Towards Safe Maneuvering of Double-Ackermann-Steering Robots with a Soft Actor-Critic Framework

Kohio Deflesselle[1*], Mélodie Daniel[1*], Aly Magassouba[2], Miguel Aranda[3] and Olivier Ly[1]

***Abstract*— We present a deep reinforcement learning framework based on Soft Actor-Critic (SAC) for safe and precise maneuvering of double-Ackermann-steering mobile robots (DASMRs). Unlike holonomic or simpler non-holonomic robots such as differential-drive robots, DASMRs face strong kinematic constraints that make classical planners brittle in cluttered environments. Our framework leverages the Hindsight Experience Replay (HER) and the CrossQ overlay to encourage maneuvering efficiency while avoiding obstacles. Simulation results with a heavy four-wheel-steering rover show that the learned policy can robustly reach up to 97% of target positions while avoiding obstacles. Our framework does not rely on handcrafted trajectories or expert demonstrations.**

I. Introduction

Recent advances in deep reinforcement learning (DRL) for robotics have mainly focused on navigation and locomotion tasks for robots such as omni-wheeled [1], quadruped [2], or biped systems [3]. By contrast, non-holonomic robots with Ackermann steering, and in particular double-Ackermann-steering mobile robots (DASMRs), remain underexplored despite their relevance in many applications such as agriculture, industrial logistics, and urban mobility [4]. Controlling DASMRs is significantly more complex than controlling differential-drive robots, since DASMRs cannot rotate in place and often require non-trivial maneuvers, such as temporarily moving away from the goal, to achieve correct alignment.

In the aforementioned applications, safety is as important as maneuvering performance. Robots must not only reach target positions precisely, but also avoid obstacles, adapt to uncertain dynamics, and operate robustly in cluttered environments. Classical planners such as the Timed Elastic Band (TEB) [5] address safety through conservative trajectory generation, but they are highly sensitive to parameter tuning and often produce overly cautious or oscillatory behaviors [6]. For DASMRs, the problem is compounded by their non-holonomic constraints: limited degrees of freedom and the need to coordinate both front and rear steering make collision-free maneuvers especially challenging [7].

DRL represents a promising alternative, but most existing approaches either target holonomic robots or simpler non-holonomic systems like differential-drive platforms [8]. Reward functions based solely on distance or heading errors often penalize safe detours (cf. Fig. 1) that are required to align correctly [9]. Imitation learning or curriculum learning [10] are not tailored to ensure both maneuvering efficiency and obstacle-avoidance in DASMRs.

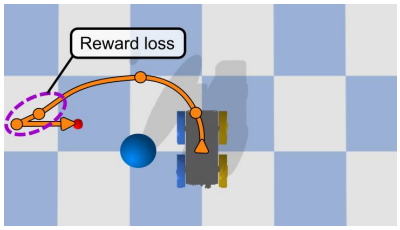

Fig. 1: Maneuvering a DASMR with DRL is challenging, as it requires temporary reward loss (circled in purple), which makes classical approaches sub-optimal. The red dot indicates the desired goal, the blue sphere denotes the obstacle, and the orange curve illustrates a feasible trajectory that avoids the obstacle while reaching the goal.

In this work, we propose an end-to-end DRL framework based on Soft Actor-Critic (SAC) combined with Hindsight Experience Replay (HER) [11] designed for DASMRs. Beyond enabling precise maneuvering, our framework explicitly addresses safety, with simulation validation in multiple scenarios featuring obstacles. We obtain promising results in this validation which showcase the potential of our framework.

II. Problem Statement

We consider a DASMR, controlled to make a maneuver to reach a desired 2D position $\boldsymbol{X_d} = (x_d, y_d)$. The DASMR is a non-holonomic platform. Its configuration includes $(\boldsymbol{X_c}, \theta_c)$, where $\boldsymbol{X_c} = (x_c, y_c)$ denotes the center position and θ_c the orientation of its longitudinal axis. However, only two DOF (motion along the longitudinal axis, and change of orientation) can be directly controlled, and the DASMR cannot rotate without moving forward or backward. The robot is initially positioned at the center of an 8-meter side square workspace P. The objective is to control the longitudinal velocity v and angular velocity $\omega_c = \dot{\theta}_c$ of the robot's center so that it reaches $\boldsymbol{X_d}$ while avoiding a fixed, known obstacle located at $\boldsymbol{X_o}$.

We restrict the workspace to a single obstacle to isolate the core difficulty of maneuvering under non-holonomic constraints while ensuring safety. This simplification is consistent with existing work such as FootstepNet [3], which showed that such an approach can then be extended with a global planner (e.g., A*) to handle cluttered environments.

A key challenge is enabling the robot to generate feasible maneuvers without relying on prior expert knowledge or predefined trajectories. The problem is formulated as a Markov Decision Process [12]. At each discrete time step t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to its policy π, and receives a reward $r_t = \mathcal{R}(s_t, a_t)$ as

*These authors equally contributed

[1] Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800 F-33400 Talence, France. [2] Cyber-physical Health and Assistive Robotics Technologies (CHART), University of Nottingham. [3]Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, 50018 Zaragoza, Spain. Corresponding author: Mélodie Daniel, e-mail: melodie.daniel@u-bordeaux.fr.

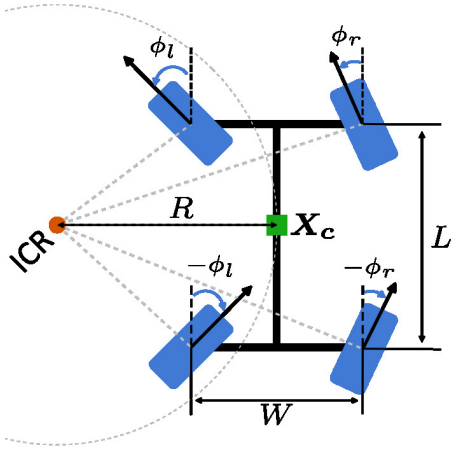

Fig. 2: DASMR rotating around an instantaneous center of rotation (ICR).

feedback. The policy π is learned in simulation, with success defined by reaching the target position $\boldsymbol{X_d}$ within a distance threshold d_{th} while avoiding collisions with $\boldsymbol{X_o}$.

III. Method

In our framework, a DRL agent controls v and ω_c. At the beginning of each training episode, the robot is initialized in the center of a square workspace containing a single fixed obstacle, and must reach $\boldsymbol{X_d}$.

A. DRL Background

In robotics, actor-critic algorithms have been successfully used in several control tasks [3], [13], [14]. Actor-critic algorithms rely on the interaction between two dense neural networks (DNNs): an actor and a critic [14]. The actor μ, also called the policy network, selects an action a_t based on the current state s_t, following $a_t = \mu(s_t)$. The critic Q, also called the Q-network, estimates the expected return of the state-action pair (s_t, a_t) by computing the Q-value $Q^\pi(s_t, a_t)$. The critic is updated using the temporal difference learning and the Bellman equation [15], with $Q_t^\pi(s_t, a_t) = r_t + \gamma \mathbb{E}[Q_{t+1}^\pi(s_{t+1}, a_{t+1})]$. The actor is updated by maximizing the expected Q-value.

SAC, introduced in [16], jointly optimizes cumulative rewards and policy entropy to promote exploration and improve training stability. To further improve learning efficiency, the recently proposed CrossQ algorithm [17] extends SAC by eliminating the need for target networks [12], through batch normalization layers. As CrossQ demonstrated promising experimental results [13], our DRL agent uses the SAC algorithm, enhanced with the CrossQ overlay.

B. State Space

The spinning velocities of the left and right wheels are ω_l and ω_r. The steering angles of the left and right wheels are ϕ_l and ϕ_r, and the corresponding steering velocities $\dot{\phi}_l$ and $\dot{\phi}_r$. The robot's center linear velocity in the world frame is $\boldsymbol{V_c} = (\dot{x}_c, \dot{y}_c) = v(\cos\ \theta_c, \sin\ \theta_c)$. At t, we define the DRL agent's $\boldsymbol{s_t}$ as ($\boldsymbol{X_c}$, $\boldsymbol{X_d}$, θ_c, ω_l, ω_r, ϕ_l, ϕ_r, $\dot{\phi}_l$, $\dot{\phi}_r$, $\boldsymbol{V_c}$, ω_c, $\boldsymbol{X_o}$) $\in \mathbb{R}^{16}$.

C. Action Space

At t, the DRL agent outputs an action $\boldsymbol{a_t} = (v, \omega_c) \in [-1, 1]$, representing normalized high-level velocity commands for the robot's center. These values are then scaled by the robot's respective velocity limits to obtain the actual control inputs.

Following the double Ackermann geometry (cf. Fig. 2) in a symmetric configuration (meaning rear steering angles are the inverse of front steering angles), ω_l, ω_r, ϕ_l, and ϕ_r can be calculated from v and ω_c, relative to the robot's current instantaneous center of rotation (ICR) as:

$$\begin{cases} \phi_l = tan^{-1}\frac{L/2}{R-W/2},\ \phi_r = tan^{-1}\frac{L/2}{R+W/2} & \text{if } \omega \neq 0 \\ \phi_l = \phi_r = 0 & \text{otherwise} \end{cases} \tag{1}$$

where R is the radius to the ICR defined as $R = \frac{v}{\omega_c}$ if $\omega_c \neq 0$, L is the wheelbase of the robot and W is the track of the robot. Similarly, we can compute ω_l and ω_r as:

$$R_l = \sqrt{(R - W/2)^2 + (L/2)^2} \tag{2}$$

$$R_r = \sqrt{(R + W/2)^2 + (L/2)^2} \tag{3}$$

$$\begin{cases} \omega_l = \frac{\omega_c}{r} R_l,\ \omega_r = \frac{\omega_c}{r} R_r & \text{if } \omega_c \neq 0 \\ \omega_l = \omega_r = \frac{v}{r} & \text{otherwise} \end{cases} \tag{4}$$

where r is the radius of the wheels, and R_l, R_r are the left and right wheels' radii to the ICR.

D. Reward Function

In our formulation, we adopt a sparse reward scheme to avoid the reward loss typically induced by classical dense functions based on Euclidean distance. Such functions tend to penalize the non-trivial maneuvers and temporary reward loss that are often necessary for DASMRs to align correctly with the goal.

To make this sparse formulation trainable, we employ HER [11], which addresses the challenge of learning under sparse rewards by relabeling unsuccessful episodes. In this approach, some of the states actually reached are retrospectively treated as if they were the intended goals. This mechanism provides meaningful reward signals even when the original goal is not achieved, thereby enabling convergence.

On top of this, we introduce additional reward shaping terms to explicitly encode safety:

$$\mathcal{R}(s_t, a_t) = \begin{cases} -1 & \text{if } \|\boldsymbol{X_d} - \boldsymbol{X_c}\| > d_{th} \\ 1 & \text{if } \|\boldsymbol{X_d} - \boldsymbol{X_c}\| \leq d_{th} \\ -10 & \text{if closestDistance}(\boldsymbol{X_o}) < 0.05 \text{ m} \\ -100 & \text{if } \boldsymbol{X_c} \notin P \end{cases} \tag{5}$$

where closestDistance$(\cdot)$ returns the closest distance between the robot and the obstacle. All norms and distances are Euclidean.

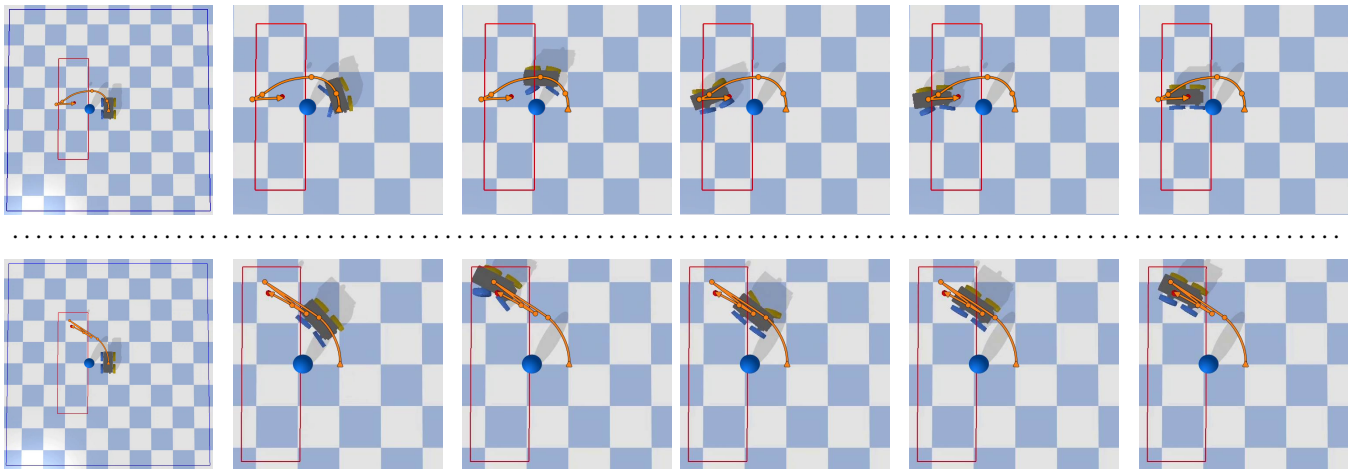

Fig. 3: Each of the two rows shows a different maneuver executed by our agent in simulation. The red dot indicates the goal, while the blue sphere denotes the obstacle.

IV. Experimental Results

A. Environment setup

We considered the Shadow Runner RR100 EDU rover as our mobile robot platform. The RR100 weighs approximately 100 kg and is 65 cm wide, 90 cm long, and 80 cm high. The DRL agent controlling the robot was trained using the PyBullet simulator, in an environment defined as follows: the robot is placed at coordinates $[0, 0]$, and $\boldsymbol{X_o}$ is set to $(0.0, 0.8)$. When the environment is reset, the robot is reset to the initial configuration, and a new $\boldsymbol{X_d}$ is sampled from a 4×1.2 m^2 goal space to the left of the obstacle, with $x \in [-2, 2]$ and $y \in [0.8, 2]$. This setup ensures that most sampled targets require the robot to perform non-trivial maneuvers while avoiding the obstacle.

A training episode is considered successful when the DRL agent reaches $\boldsymbol{X_d}$ within d_{th}, regardless of orientation. If the robot fails to reach $\boldsymbol{X_d}$ within a time step limit or drives outside the boundaries of its workspace, the episode is truncated, and the environment is consequently reset.

TABLE I: SAC, CrossQ, and HER parameters used on SBX during training and testing phases.

Parameter	Value
Nb. layers	2
Actor Hidden size	256
Critic Hidden size	1024
$\alpha_A = \alpha_C$	3e-4
Replay buffer size	1,000,000
Batch size N	256
γ	0.99
N. sampled goal	16
d_{th}	0.15 m
Training random seed	9527
Goal selection strategy	Future
Number of sampled goals	16

TABLE II: Benchmarking results for Seen and Unseen targets and $d_{\text{th}} = 15$ cm: with the success rate (SR) in %, the average error (AE) (the standard deviation σ) in m, and the SR weighted by path length (SPL).

Approach	Seed	Metrics		
		SR ↑	AE(σ) ↓	SPL ↑
Standard DRL [18]	Seen	29	0.33 (0.19)	0.17
	Unseen	24	0.43 (0.40)	0.14
Ours	Seen	**97**	**0.17** (0.21)	**0.90**
	Unseen	**95**	**0.15** (0.13)	**0.87**

B. Training setup

The DRL agent was trained in a single, non-vectorized simulation environment for 300,000 time steps, where each episode was limited to 800 time steps before truncation. The agent interacted with its environment at a frequency of 40 Hz. The training progress was monitored by logging the average episode reward and success rate (SR) every 10 episodes, both computed using a sliding window of 100 episodes. The training was carried out on a desktop computer equipped with an AMD Ryzen Threadripper PRO 7985WX 64-Cores (AMD Zen 4) CPU, 128 GB of memory, along with an Nvidia RTX 4090 GPU. The DRL algorithms were implemented using the Stable Baselines JAX (SBX) library, leveraging its CrossQ implementation built upon the SAC algorithm. Hyperparameters and network architecture details for SAC, CrossQ, and HER are provided in Table I, and were kept consistent in all experiments unless otherwise specified.

C. Preliminary simulation results

To evaluate the effectiveness of our framework, we conducted simulation experiments against a classical DRL framework [18] in two settings: (i) the same configuration as during training, and (ii) a new configuration obtained with a different random seed for goal sampling. Each framework was quantitatively evaluated with the success rate (SR), and the average distance error (AE) and the standard deviation to $\boldsymbol{X_d}$. We also evaluated the average Success weighted by (normalized inverse) Path Length (SPL) [19]. Higher SPL values indicate trajectories that are not only successful but also closely aligned with the shortest possible path. Achieving a high SPL is particularly challenging for DASMRs, as reaching the target position often requires complex maneuvers that deviate from the optimal path.

The results in Table II show a clear advantage of our framework over a standard DRL baseline [18]. In both seen and unseen targets, our framework achieves promising success rates (97% and 95%), compared to less than 30% for the baseline. This indicates that our framework generalizes well beyond the training parameters. The SPL values demonstrate that trajectories are not only successful but also significantly

more efficient. Two examples of the achieved trajectories are presented in Fig. 3. Importantly, the performance gap between seen and unseen settings remains small, suggesting that our framework learns robust maneuvering strategies rather than overfitting to specific goal configurations.

V. Conclusion

This work demonstrates that our DRL framework, leveraging SAC, CrossQ, and HER, can handle the maneuvering and safety challenges of DASMRs. Preliminary results show a success rate up to 97% and the emergence of obstacle-avoidance behaviors, confirming the potential of our framework for real-world deployment. Future work will extend the evaluation to real-world settings.

Acknowledgment

This work was supported by the French Government under the France 2030 program through the National Research Agency (ANR) grant reference ANR-24-PEAE-0002 and the IdEx University of Bordeaux / RRI ROBSYS grant. It was also funded by the Nouvelle-Aquitaine Region through the MIRAE project.

References

[1] A. Mehmood, I. Shaikh, and A. Ali, "Application of Deep Reinforcement Learning for Tracking Control of 3WD Omnidirectional Mobile Robot," *Information Technology and Control*, vol. 50, no. 19, pp. 507–521, 2021.

[2] Y. Zhang, J. Zeng, *et al.*, "Dual-Layer Reinforcement Learning for Quadruped Robot Locomotion and Speed Control in Complex Environments," *Applied Sciences*, vol. 14, no. 19, 2024.

[3] C. Gaspard, G. Passault, *et al.*, "FootstepNet: an Efficient Actor-Critic Method for Fast On-line Bipedal Footstep Planning and Forecasting," in *IEEE/RSJ IROS*, 2024, pp. 13 749–13 756.

[4] M. Deremetz, R. Lenain, *et al.*, "Path tracking of a four-wheel steering mobile robot: A robust off-road parallel steering strategy," in *ECMR*, 2017, pp. 1–7.

[5] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *IEEE/RSJ IROS*, 2017, pp. 5681–5686.

[6] B. Magyar, N. Tsiogkas, *et al.*, "Timed-Elastic Bands for Manipulation Motion Planning," *IEEE RA-L*, vol. 4, no. 4, pp. 3513–3520, 2019.

[7] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robotics*. MIT Press, 2005.

[8] W. Zhang, S. Wang, *et al.*, "DRL-DCLP: A Deep Reinforcement Learning-Based Dimension-Configurable Local Planner for Robot Navigation," *IEEE RA-L*, vol. 10, no. 4, pp. 3636–3643, 2025.

[9] L. Lazzaroni, F. Bellotti, *et al.*, "Deep Reinforcement Learning for Automated Car Parking," in *Springer APPLEPIES*, vol. 1036, 2022, pp. 125–130.

[10] H. Xue, B. Hein, *et al.*, "Using Deep Reinforcement Learning with Automatic Curriculum Learning for Mapless Navigation in Intralogistics," *Applied Sciences*, vol. 12, no. 6, 2022.

[11] M. Andrychowicz, F. Wolski, *et al.*, "Hindsight Experience Replay," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] T. P. Lillicrap, J. J. Hunt, *et al.*, "Continuous Control with Deep Reinforcement Learning," in *PMLR ICLR*, 2016.

[13] C. Gaspard, M. Duclusaud, *et al.*, "FRASA: An End-to-End Reinforcement Learning Agent for Fall Recovery and Stand Up of Humanoid Robots," in *IEEE ICRA*, 2025, pp. 15 994–16 000.

[14] M. Daniel, A. Magassouba, *et al.*, "Multi Actor-Critic DDPG for Robot Action Space Decomposition: A Framework to Control Large 3D Deformation of Soft Linear Objects," *IEEE RA-L*, vol. 9, no. 2, pp. 1318–1325, 2024.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*. MIT Press, 2018.

[16] T. Haarnoja, A. Zhou, *et al.*, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *PMLR ICML*, vol. 80, 2018, pp. 1856–1865.

[17] A. Bhatt, D. Palenicek, *et al.*, "CrossQ: Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity," in *PMLR ICLR*, 2024.

[18] V. R. F. Miranda, A. A. Neto, *et al.*, "Generalization in Deep Reinforcement Learning for Robotic Navigation by Reward Shaping," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 6, pp. 6013–6020, 2024.

[19] P. Anderson, A. X. Chang, *et al.*, "On Evaluation of Embodied Navigation Agents," *CoRR*, vol. abs/1807.06757, 2018.