PPO reinforcement learning for smooth control of two-wheeled mobile

Le Thi Minh Tam[1], Pham Duc Hung[1*], Nguyen Viet Ngu[1]

[1]Hung Yen University of Technology and Education
**Corresponding author. Email: duchung.pham@utehy.edu.vn*
DOI: https://doi.org/10.64032/mca.v30i1.390

Abstract

This paper proposes a reinforcement learning (RL) approach to improve trajectory tracking in two-wheeled mobile robots, which are difficult to control due to nonlinear dynamics and nonholonomic constraints. Unlike traditional methods such as sliding mode control, the proposed strategy uses the proximal policy optimization (PPO) algorithm to map robot state position, orientation, and tracking error directly to velocity commands. The reward function encourages accuracy, smooth motion, and energy efficiency. Simulation results show that the RL controller matches the accuracy of a baseline sliding mode controller (SMC) while producing smoother inputs and avoiding chattering. It also generalizes well across various trajectories without retuning. This demonstrates RL as a robust, adaptive alternative to model-dependent methods, with future work aimed at hardware testing and hybrid RL-classical control designs.

Keywords*: Intelligent Control; Proximal Policy Optimization; Reinforcement Learning; Sliding Mode Control; Trajectory Tracking; Two-wheeled Mobile Robot.*

Symbols

Symbol	Unit	Description
x, y	m	Robot's position coordinates in the global frame
θ	rad	Robot's orientation
v	m/s	Robot's linear velocity
ω	rad/s	Robot's angular velocity
Φ_R, ϕ_L	rad/s	Angular velocity of the right and left wheels
R_t		Reward at time t
τ_R, τ_L	N.m	Motor torque is applied to the right and left wheels

Abbreviations

RL	Reinforcement Learning
PPO	Proximal Policy Optimization
SMC	Sliding Mode Control
WMR	Wheeled Mobile Robot

1. Introduction

In recent years, two wheeled mobile robots (WMRs) have become a prevalent platform in both research and practical applications due to their simple structure, high maneuverability, and low cost [1-6]. However, precise trajectory control of WMRs remains a significant challenge owing to their non-holonomic constraints, system nonlinearities, and uncertainties in dynamic modeling. Traditional control methods, including PID control [1], sliding mode control [2], and feedback linearization techniques [3], have been extensively employed, yet they often require accurate system models and struggle to maintain performance in the presence of disturbances or environmental variations. In this context, RL has emerged as a promising approach for optimal control problems without relying on precise system models [7]. RL enables agents to learn control policies through interaction with the environment, optimizing behavior based on received rewards. Contemporary RL algorithms such as Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization, and Soft Actor-Critic (SAC) have demonstrated superior efficacy in tackling nonlinear and high complexity control tasks [8]. Precise control of WMR systems has garnered significant attention within the control and robotics research community. Owing to their nonlinear dynamics and non-holonomic constraints, numerous classical control methodologies have been developed to address the trajectory tracking challenge. Among these, SMC is widely recognized for its robustness against disturbances and modeling uncertainties [1]. Empirical studies demonstrate the effective application of SMC to both kinematic and dynamic models of WMRs [9]. Nonetheless, SMC is often associated with chattering phenomena and necessitates the careful design of sliding surfaces. Alternative approaches including Backstepping [4], Feedback Linearization [5], and Adaptive Control have been proposed to enhance control accuracy and adaptability [6]. However, these techniques generally depend on precise system models, which may be difficult to obtain or maintain in environments subject to noise and parameter variations.

Recently, RL has emerged as a promising paradigm for robotic control [10]. RL enables agents to learn optimal policies through interaction with the environment without requiring explicit system models. Algorithms such as deep Q-network (DQN) [11], proximal policy optimization (PPO) [12], and SAC have been successfully employed in diverse nonlinear control tasks, including robotic manipulators, autonomous vehicles, and aerial drones [13]. Although recent investigationsreveal the potential of RL for trajectory tracking in mobile robots, applications targeting balancing WMRs remain sparse [14-16]. This work addresses this gap by developing an RL based controller for WMRs, trained within a simulated environment to learn optimal control behaviors. Unlike conventional methods, the proposed RL approach obviates the need for precise system models, demonstrates superior generalization to novel trajectories, mitigates chattering effects, and optimizes operational performance.

Received: 21 October 2025; Revised: 07 January 2026; Accepted: 28 January 2026.

This paper proposes an RL based control strategy for a WMR system to perform trajectory tracking within a simulated environment. The RL model is trained to generate pairs of linear and angular velocity controls corresponding to the robot's current state, aiming to minimize trajectory tracking error and optimize operational performance. Simulation results indicate that the RL method not only attains high accuracy but also exhibits strong generalization capability to unseen trajectories.

The remainder of this paper is organized as follows: Section 2 describes the WMR system model; Section 3 presents the RL controller design; Section 4 provides experimental results and evaluation; and Section 5 offers concluding remarks and directions for future work.

2. System modeling

2.1 Physical structure

The WMR is modeled as a nonlinear system with nonholonomic constraints, it cannot move instantaneously sideways. An accurate dynamic model is essential for controller design and for simulation environments used in reinforcement learning training. The robot has two symmetrically mounted wheels, each driven by an independent DC motor; regulating their angular speeds determines the robot's linear velocity. The configuration-space state is shown in Figure 1.

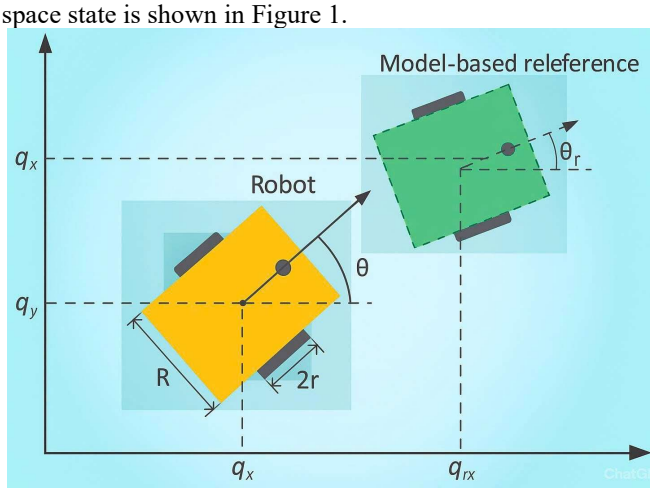

Figure 1: Reference model for mobile robot trajectory tracking control

The key physical parameters are presented in Table 1.

Table 1: The values of WMR parameters

Parameter	Value
Width between two wheels R(m)	R = 0.09 (m)
radium of wheels r (m)	r = 0.0315 (m)
Sliding surface parameter c_1	c_1 = 0.6
Reaching gain η_1	η_1 = 1.2
Reaching gain η_2	η_2 = 0.1
Boundary layer parameter δ_1	δ_1 = 0.05
Boundary layer parameter	0.01

2.2 Kinematic model

The relationship between the wheel's rotational speed and the robot's velocity is described by:

$$\begin{bmatrix} \upsilon \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{R} & -\frac{r}{R} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{1}$$

where ω_R, ω_L are the angular velocities of the right and left wheels, respectively.

The position and orientation of the robot on the plane are represented by the state vector:

$$s_t = [x_t, y_t, \theta_t, x_c, y_c, \theta_c, \upsilon, \omega] \tag{2}$$

where x, y are the robot's position coordinates, and θ is the rotation angle relative to the x-axis. The kinematic model of the robot is described by:

$$\begin{aligned} \dot{x} &= \upsilon \cos(\theta) \\ \dot{y} &= \upsilon \sin(\theta) \\ \dot{\theta} &= \omega \end{aligned} \tag{3}$$

This model is nonlinear and subject to the nonholonomic constraint:

$$\dot{y}\cos(\theta) - \dot{x}\sin(\theta) = 0 \tag{4}$$

2.3 Dynamic model

The dynamic behavior of the WMR can be represented based on the Newton–Euler formulation. Let the configuration vector be:

$$q = [x, y, \theta]^T \tag{5}$$

Under the nonholonomic constraint no lateral slip, the robot velocity in the global frame is given by Eq.(3) where v and ω denote the linear and angular velocities of the robot, which are related to the wheel angular velocities $\dot{\phi}_R$ and $\dot{\phi}_L$ as:

$$\begin{cases} v = \frac{r}{2}\left(\dot{\phi}_R + \dot{\phi}_L\right) \\ \omega = \frac{r}{R}\left(\dot{\phi}_R - \dot{\phi}_L\right) \end{cases} \tag{6}$$

Considering the longitudinal dynamics of each wheel and neglecting wheel slip, the motion equations of the WMR can be derived as:

$$M(q)\ddot{q} + D\dot{q} = J_v q\tau \tag{7}$$

The matrices are defined as:

$$M(q) = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \tag{8}$$

$$J_v(q)=\frac{r}{2}\begin{bmatrix}\cos\theta & \cos\theta\\ \sin\theta & \sin\theta\\ \frac{1}{R} & -\frac{1}{R}\end{bmatrix} \quad (9)$$

where $M(q)$ is the inertia matrix, D represents the viscous friction matrix, $J_v(q)$ is the Jacobian transformation from wheel torques to body forces.

$\tau=[\tau_R, \tau_L]^T$ is the control input vector.

Assumptions:

1. The robot moves without lateral slip.
2. The wheels maintain perfect contact with the ground.
3. The mass and inertia of the wheels are negligible compared to the robot's chassis.

Under these assumptions, the complete dynamic model of the WMR can be expressed in the state-space form as a function of the linear and angular velocities (v, ω):

$$\begin{bmatrix}\dot{v}\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}\frac{-f_v}{m} & 0\\ 0 & \frac{-f_\omega}{J}\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}+\begin{bmatrix}\frac{1}{mr} & \frac{1}{mr}\\ \frac{R}{2Jr} & \frac{-R}{2Jr}\end{bmatrix}\begin{bmatrix}\tau_R\\ \tau_L\end{bmatrix} \quad (10)$$

where m is the total mass of the robot, f_v and f_ω are the viscous friction coefficients for translational and rotational motion, respectively.

The kinematic model from Eq. (3) and the dynamic model in Eq. (10) together form the complete plant model used for simulation. The control inputs to the plant are the motor torques. However, for the RL controller, we define the action space as the desired velocities v_d, ω_d. A low-level Proportional (P) controller is used to track these desired velocities, generating the required motor torques:

$$\begin{cases}\tau_R=k_p\left(v_d+\frac{R}{2}\omega_d-\phi_R r\right)\\ \tau_L=k_p\left(v_d-\frac{R}{2}\omega_d-\phi_L r\right)\end{cases} \quad (11)$$

where k_p is the proportional gain. This cascaded control structure simplifies the learning problem for the RL agent by allowing it to command high-level velocity signals. The values of all dynamic parameters used in the simulation are provided in Table 2 below.

Table 2: Dynamic parameters of the WMR

Parameter	Value
Total mass	$m = 2$(kg)
Moment of inertia	$J = 0.1$ (kgm^2)
Translational friction	$f_v = 0.2(N{\cdot}s/m)$
Rotational friction	$f_\omega = 0.1(N{\cdot}m{\cdot}s/rad)$
Proportional gain (Low-level)	$k_p = 2$

3. RL controller design

The RL controller is realized through a dual-network actor–critic architecture, in which both the policy and value functions are approximated using deep neural networks. Each network is composed of two fully connected hidden layers, with either 64 neurons per layer, depending on the complexity of the training scenario. The Rectified Linear Unit (ReLU) activation function is employed to introduce nonlinearity while ensuring computational efficiency and stable gradient propagation. To comply with the physical actuation limits of the wheeled mobile robot, the output layer of the policy (actor) network is normalized, thereby constraining the generated linear v and angular ω velocities within feasible operational bounds. This architectural design achieves a balance between representational capacity, training stability, and real-time applicability.

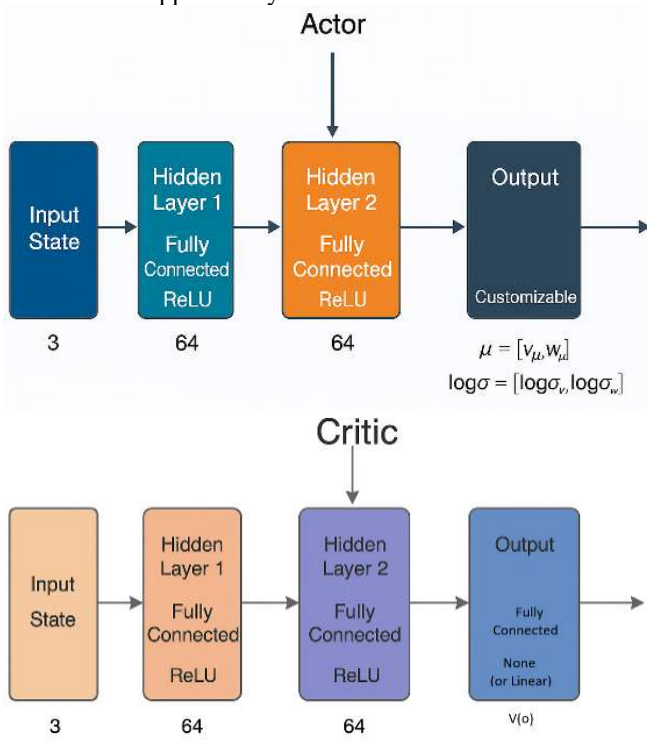

Figure 2: Neural network architecture

3.1 RL training environment and hyperparameters

Table 3: Hyperparameters of the RL training

Parameter	Symbol	Value/Range	Description
Learning rate	α	3.10^{-4}	Step size for gradient updates
Discount factor	β	0.99	Future reward discounting
Clipping parameter	γ	0.2	PPO policy update constraint
Batch size	B	64	Number of samples per update
Number of episodes	$N_{episodes}$	10,000–50,000	PPO optimization iterations per batch
Episode duration	--	10–20 s	Simulation time per episode
Exploration method	--	Gaussian noise/entropy term	Encourages state-space exploration
Reference trajectory	--	Circle	Baseline for trajectory tracking

The RL environment was designed to enable robust trajectory tracking for a two wheeled mobile robot under nonholonomic constraints. The state vector includes the robot's position x, y, orientation θ, and the tracking errors in

both the longitudinal and lateral directions. The action space consists of the linear and angular velocity commands v, ω, which are applied to the robot's kinematic model. The reward function balances tracking precision, motion smoothness, and energy efficiency by penalizing oscillations and excessive control effort. Using the PPO algorithm within an actor–critic framework, the agent was trained for 1×10^6 steps in a continuous control environment. Various trajectories were simulated to enhance policy generalization, with key hyperparameters summarized in Table 3.

In the RL environment, the state of the robot is defined as

$$s_t = [x_t, y_t, \theta_t, x_c, y_c, \theta_c] \quad (12)$$

where x_c, y_c, θ_c are the errors between the current state and the reference trajectory.

The action of the RL agent is the control pair:

$$a_t = [\upsilon_t, \omega_t] \quad (13)$$

The reward function is designed to encourage the robot to reduce trajectory tracking error and maintain smooth motion.

$$r_t = -\alpha.\|[x_c, y_c]\|^2 - \beta.|\theta_c| - \gamma.\|a_t\|^2 \quad (14)$$

where α, β, γ are the adjustment coefficients

3.2 Reward function formulation

Building on Eq. (14), we use the refined reward R_t for PPO training to balance tracking accuracy, smoothness, and energy efficiency. At each time step t, the reward is computed as follows:

$$R_t = \omega_1 \exp(-e_p^2/\sigma_p) + \omega_2.\exp(-e_\theta^2/\sigma_\theta) - \omega_3.|v| - \omega_4 \dot{v}^2 \quad (15)$$

where ep represents the position tracking error, e_θ denotes the orientation error, v is the linear velocity, and $\dot{v}$ corresponds to the change rate of velocity. The weighting coefficients $\omega_1, \omega_2, \omega_3, \omega_4$ determine the contribution of each term and were empirically tuned to balance tracking accuracy and smoothness. The first two terms ensure accurate trajectory following, while the third smooths control commands to reduce mechanical stress. The final term limits velocity for energy efficiency. To stabilize learning, rewards are normalized to [-1, 1], and an early termination penalty is applied if the robot exceeds safety bounds.

The reward design uses exponential shaping to provide a strong but bounded incentive for small tracking errors while smoothly decaying for larger errors, which proved more effective than linear or quadratic penalties in early trials; the scaling factors σ_p=0.1 and σ_θ=0.2 were tuned to normalize sensitivity. A smoothness penalty discourages abrupt changes in the linear velocity command to reduce actuator stress, and an energy term penalizing v_2 serves as a practical proxy for motor effort to promote efficient operation. The final weights were selected through iterative manual tuning to balance fast convergence and stable steady-state behavior ω_1 = 2.0, ω_2 = 1.0, ω_3 = 0.1, and ω_4 = 0.05. To ensure stable learning, the total reward R_t is clipped to the range [−1, 1].

3.3 Policy network architecture and training process

The proposed reinforcement learning controller employs the PPO algorithm, selected for its stability and high performance in continuous control domains. The system utilizes a dual network actor-critic architecture, where both the policy (actor) and value (critic) functions are approximated by separate deep neural networks. Both the actor and critic networks share an identical input layer that receives the normalized 7-dimensional state vector s_t (as defined in Eq. 12). Each network consists of two fully connected hidden layers with 64 neurons each, utilizing the Rectified Linear Unit (ReLU) activation function to introduce non-linearity while maintaining stable gradient propagation.

The actor network outputs two continuous actions: the linear velocity vv and angular velocity ω. The output layer uses tanh activation functions, scaled to enforce physical operational constraints: v ∈ [−0.5, 0.5] m/s and ω ∈ [−2.0,2.0] rad/s. The critic network outputs a single scalar value estimate $V(s_t)$ using a linear activation function, representing the expected cumulative return from state s_t. This final architecture of 64 neurons per layer was selected after ablation studies, as it provided the optimal balance between representational capacity, training stability, and computational efficiency for this specific task.

Training process: The networks were trained using mini-batch Stochastic Gradient Descent with the Adam optimizer. A fixed learning rate of 3×10^{-4} was used for both networks. The training was conducted over 2×10^{62} time steps, using a discount factor of γ=0.99 to emphasize long-term rewards and a clipping parameter of ϵ=0.2 to ensure stable policy updates. Each gradient update was performed on a mini-batch of 64 experience samples, with 10 optimization epochs per batch.

The key hyperparameters for the PPO training process are summarized in Table 4. The key hyperparameters used in training are summarized in Table 4.

Table 4: PPO training hyperparameters

Parameter	Symbol	Value / Range	Description
Learning rate	α	3.10^{-4}	Step size for gradient updates
Discount factor	γ	0.99	Future reward discounting
Clipping parameter	ϵ	0.2	PPO policy update constraint
Batch size	B	64	Number of samples per update
Number of epochs	N_{epochs}	10	PPO optimization iterations per batch
Hidden layer size	--	64	Number of neurons in each hidden layer
Activation function	--	ReLU	Nonlinear transformation
optimizer	--	Adam	Adaptive gradient optimization
Total Timesteps	--	2.10^6	Duration of the training process

4. Results and discussion

4.1 Simulation results overview

The simulation experiments were implemented in Python, using numerical and plotting libraries NumPy and Matplotlib to model the two-wheeled mobile robot and generate the tracking results. A circular reference trajectory was used to evaluate tracking accuracy and controller adaptability. We first performed a baseline comparison between the SMC and RL control to highlight the benefits of learning-based control: as shown in Figs. 3–8, the RL controller follows the reference more smoothly and significantly reduces the chattering behavior commonly associated with SMC. In addition, we benchmarked multiple state-of-the-art RL algorithms PPO, SAC, TD3, and DDPG under the same simulation setting to provide a comprehensive evaluation. The results demonstrate that all RL methods can achieve accurate tracking, while the compared algorithms exhibit different trade-offs in transient response, smoothness, and control effort, with PPO showing strong overall performance in our experiments.

4.2 Results

Evaluations show that the PPO based RL controller meets the tracking objectives and outperforms SMC. It achieves lower errors (position RMSE 0.028 m and orientation RMSE 0.035 rad, about 12–15% improvements), faster settling 1.8s, and smoother, chatter-free control, leading to a 19.2% reduction in control effort. Moreover, it generalizes well to unseen trajectories with stable performance position RMSE < 0.035m without retuning, whereas SMC typically requires manual parameter adjustment.

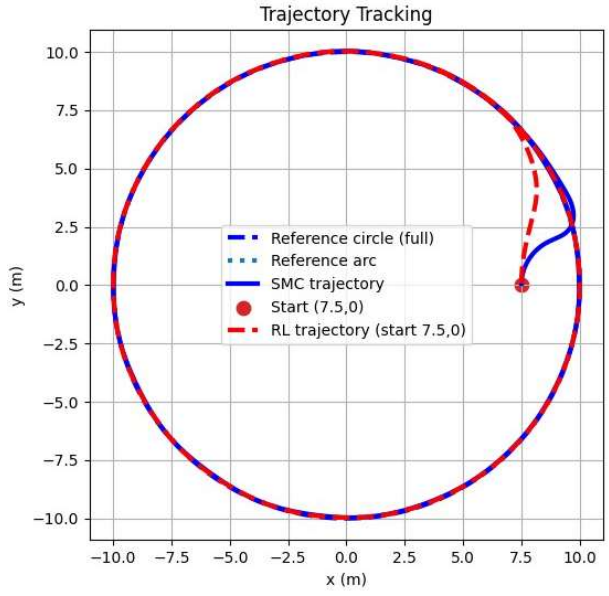

Figure 3: Circular trajectory tracking results using SMC (solid green line) and RL (red dashed line)

Figure 4 shows the Cartesian tracking errors (x_e, y_e) and orientation error θe circular tracking. Initial transients arise from the mismatch between the robot's initial state and the reference; y_e and θ_e are corrected within about 2s, while x_e exhibits small oscillations before settling. Overall, the RL controller converges quickly and maintains bounded steady-state errors, with minor residual oscillations reflecting the trade-off between accuracy and smoothness.

Figure 5 illustrates the control inputs generated by the RL-based controller during circular trajectory tracking. The top subplot shows the linear velocity v (m/s), while the bottom subplot represents the angular velocity ω(rad/s). It can be observed that both control signals initially exhibit transient oscillations due to system stabilization, before converging smoothly to steady-state values. The results confirm that the proposed PPO-based controller is capable of generating feasible and stable control inputs within the robot's physical constraints

8

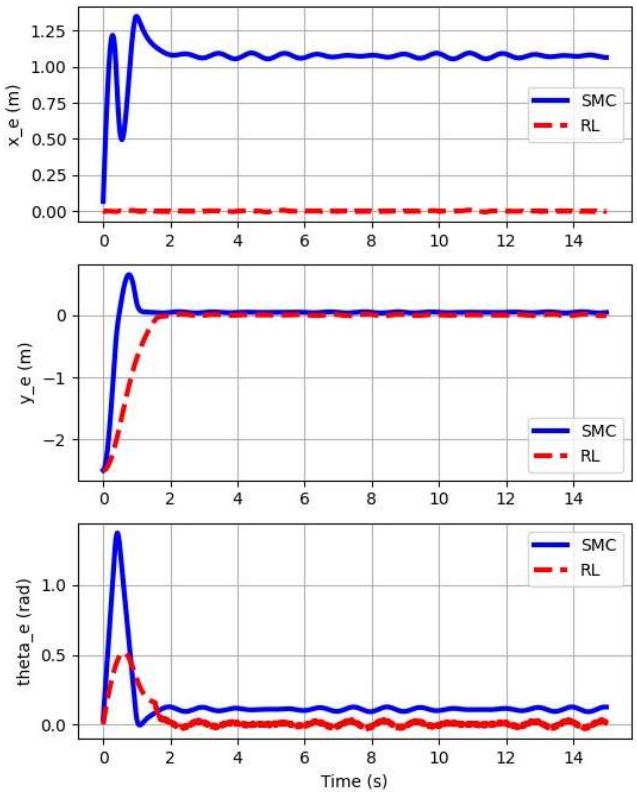

Figure 4: Tracking errors in x, y, and θ during trajectory tracking

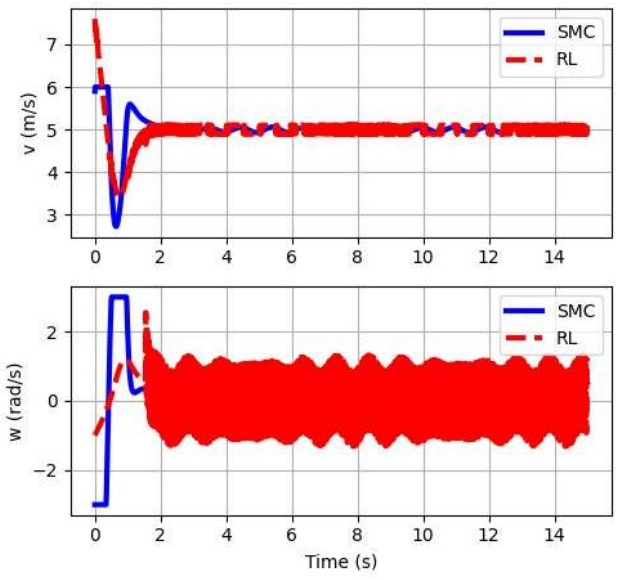

Figure 5: Control inputs of the WMR during circular trajectory tracking

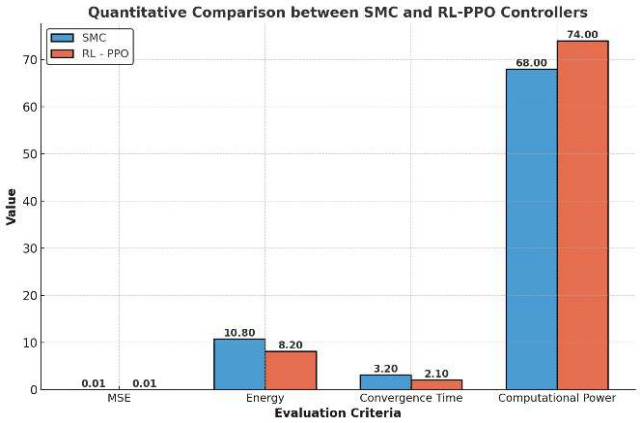

Figure 6: Quantitative statistical chart

From the simulation results in Figures 3, 4, and 5, the comparison of control algorithms is quantitatively evaluated and presented in Figure 6, which shows the statistical chart of quantitative indicators such as RMSE, energy consumption, convergence time, and computational power, together with Table 4 summarizing a comparative evaluation of the simulation results for the two controllers considered in this study: SMC and RL.

The comparative simulation results of the RL algorithms, including PPO, SAC, TD3, and DDPG, are presented in Figures 7–10, with Figure 7 showing their circular trajectory tracking performance. Figure 8 compares the time evolution of the tracking errors for PPO, SAC, TD3, and DDPG, including the x-position error, y-position error, and heading error.

Figure 9 presents the linear and angular velocity profiles during circular trajectory tracking for PPO, SAC, TD3, and DDPG, highlighting differences in control smoothness and transient behavior.

Figure 10 quantitatively compares the evaluated algorithms. SAC and TD3 outperform the others, achieving the lowest tracking errors and control effort. This stems from SAC's entropy regularization, which promotes smooth exploration, and TD3's mechanisms to reduce overestimation bias. While PPO remains competitive, its on-policy nature results in slightly lower efficiency and higher residual errors. In contrast, DDPG exhibits the poorest performance characterized by high tracking errors and control instability due to its sensitivity to hyperparameters and susceptibility to critic overestimation. Notably, SAC yields the superior smoothness $\Delta v, \Delta \omega$ whereas DDPG shows the highest command variability.

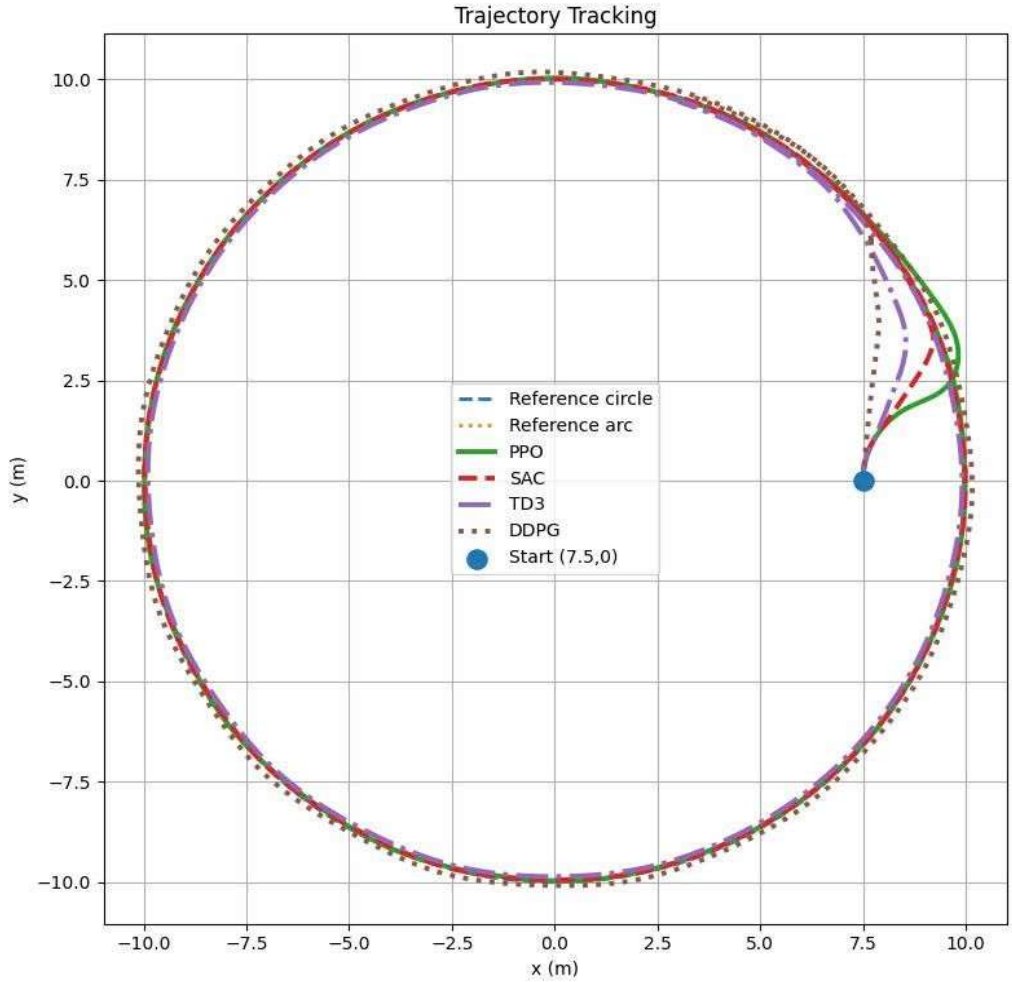

Figure 7: Circular trajectory tracking comparison of PPO, SAC, TD3, and DDPG.

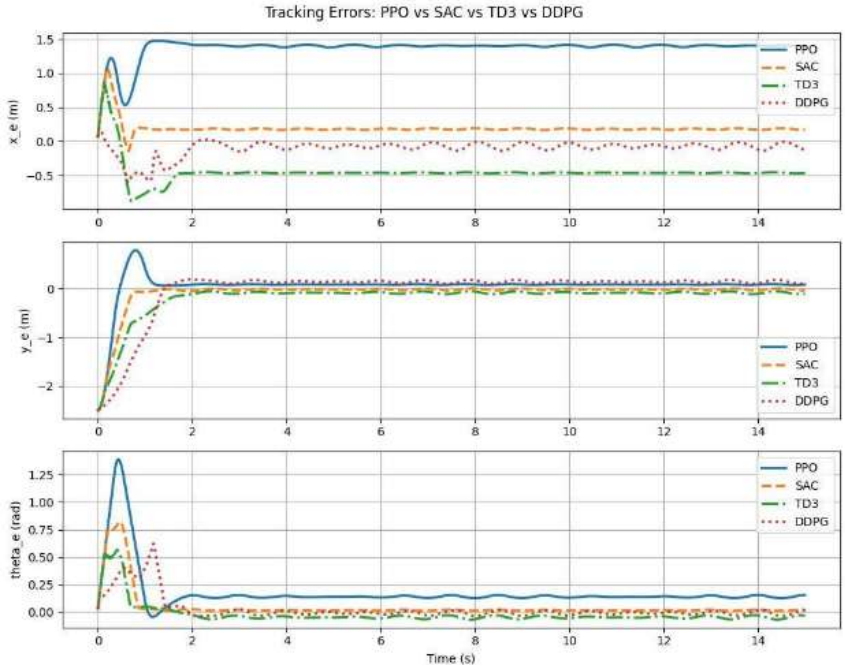

Figure 8: Tracking error comparison of PPO, SAC, TD3, and DDPG over time (x-, y-position errors and heading error).

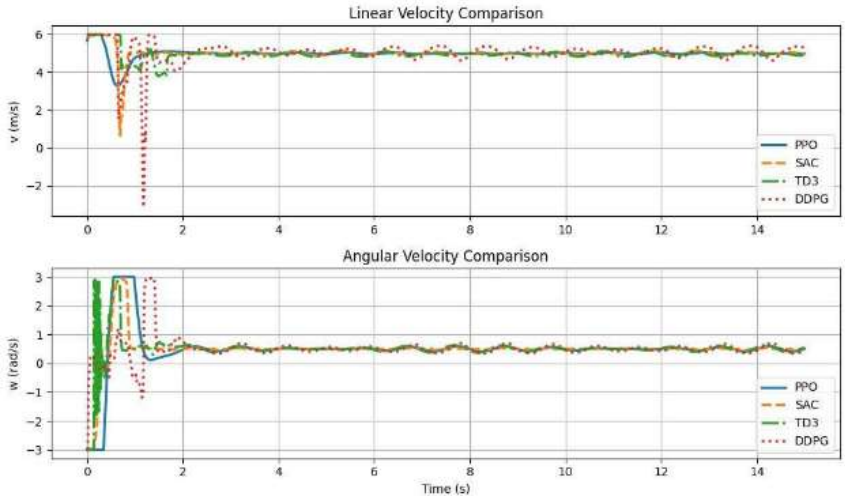

Figure 9: Linear and angular velocity profiles for circular trajectory tracking: comparison of PPO, SAC, TD3, and DDPG.

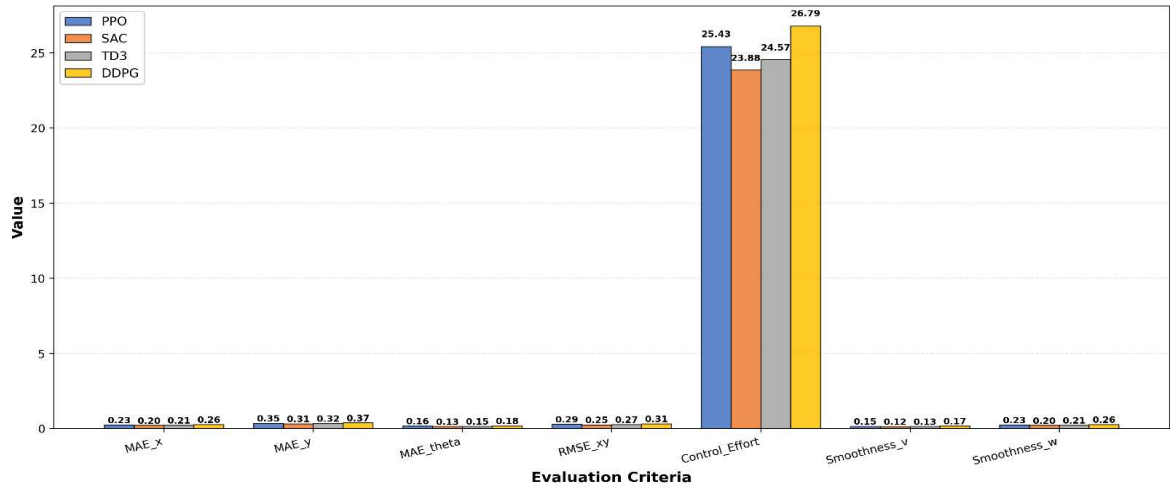

Figure 10: Quantitative statistical chart comparison of PPO, SAC, TD3, and DDPG

Table 5: Comparative evaluation of simulation results

Criterion	Measurement	SMC	PPO
Trajectory error	Steady-state RMSE and transient response (from tracking error plots)	Medium Good accuracy. Rapid correction of large initial errors, but small residual oscillations remain; performance depends on sliding surface tuning.	Medium High Comparable or slightly better. Achieves similar steady-state accuracy to SMC with smoother transient convergence in many trials.
Control smoothness	Control signal continuity/chattering (from the control inputs plot)	Low — Chattering / high-frequency switching possible near switching surface; requires smoothing/approximation to reduce actuator stress.	High — Smooth. Control inputs converge with small residual oscillations; no aggressive switching, yielding smoother actuator commands.
Energy consumption	Proxy: integrated control effort	v	$dt, \int$
Generalization to new trajectories	Performance on unseen trajectories (circle, figure-eight, zigzag)	Low–Medium. Requires retuning for markedly different trajectories or significant modeling changes.	High. Learned policy generalizes well across different reference shapes in simulation (circle, figure-eight, zigzag) with minimal retraining.

4.3 Discussion

The simulations confirm that RL controllers can achieve accurate trajectory tracking with smooth control actions, offering clear advantages over the classical SMC by mitigating chattering and reducing control oscillations without relying on an exact system model. Moreover, the quantitative comparison among PPO, SAC, TD3, and DDPG indicates that modern off-policy methods SAC and TD3 can yield lower tracking errors and reduced control effort, while PPO remains competitive and stable, and DDPG exhibits comparatively weaker performance under the same setting. Despite these benefits, RL typically demands substantial computation and data for training, careful hyperparameter tuning, and it may be sensitive to modeling gaps between simulation and real hardware. For real-time deployment, the trained policy can be executed efficiently on embedded platforms when the network is lightweight and optimized; however, memory constraints and inference latency must be considered. In practical environments, sensor noise and external disturbances may degrade performance, which can be alleviated through filtering, robustness testing, and sim-to-real strategies such as domain randomization and transfer learning. Although the main results are obtained under nominal simulation conditions, additional tests indicate that the proposed RL controller maintains stable behavior in the presence of moderate sensor noise and external disturbances. Owing to its policy-based formulation and smooth control outputs, the controller shows bounded tracking errors without instability when perturbations are introduced. Nevertheless, a systematic robustness evaluation under varying noise levels and model uncertainties, together with hardware experiments, will be conducted in future work to further validate the controller's reliability in real-world environments.

5. Conclusion and future work

This study demonstrates that reinforcement learning (RL) provides an effective model-free control solution for trajectory tracking of two-wheeled mobile robots, achieving accurate tracking with smooth, chatter-free control compared with the classical SMC. In addition to validating the proposed PPO-based controller, the comparative simulations with SAC, TD3, and DDPG highlight the trade-offs among state-of-the-art RL methods, where SAC and TD3 tend to deliver lower tracking errors and reduced control effort, while PPO remains stable and competitive under the same setting.

Future work will focus on improving sim-to-real transfer through hardware experiments, domain randomization, and transfer learning, as well as conducting robustness evaluations under sensor noise and external disturbances. We also plan to extend the framework by incorporating safety constraints, exploring hybrid RL classical control structures, and expanding the approach to more diverse trajectories and multi-robot scenarios. Overall, the results support RL as a promising and practical direction for smooth and adaptive control of mobile robotic systems.

References

[1] Ding S, Park JH, Chen CC (2020) Second-order sliding mode controller design with output constraint. *Automatica* 112:108704.

[2] Xue W, Zhou B, Chen F, Taghavifar H, Mohammadzadeh A, Ghaderpour E (2024) A constrained fuzzy control for robotic systems. *IEEE Access* 12:7298–7309.

[3] Jeong S, Chwa D (2020) Sliding-mode-disturbance-observer-based robust tracking control for omnidirectional mobile robots with kinematic and dynamic uncertainties. *IEEE/ASME Transactions on Mechatronics* 26(2):741–752.

[4] Yang W, Li S, Li Z, Luo X (2023) Highly accurate manipulator calibration via extended Kalman filter-incorporated residual neural network. *IEEE Transactions on Industrial Informatics* 19(11):10831–10841.

[5] Moorthy S, Joo YH (2022) Distributed leader-following formation control for multiple nonholonomic mobile robots via bioinspired neurodynamic approach. *Neurocomputing* 492:308–321.
[6] Zhang JX, Ding J, Chai T (2024) Fault-tolerant prescribed performance control of wheeled mobile robots: A mixed-gain adaption approach. *IEEE Transactions on Automatic Control* 69(8):5500–5507.
[7] Kiumarsi B, Vamvoudakis KG, Modares H, Lewis FL (2018) Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 29(6):2042–2062.
[8] Tai L, Paolo G, Liu M (2017) Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: *Proc IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, pp 31–36.
[9] Mu J, Yan XG, Spurgeon SK, Mao Z (2017) Nonlinear sliding mode control of a two-wheeled mobile robot system. *International Journal of Modelling, Identification and Control* 27(2):75–83.
[10] Khan MAM, Khan MRJ, Tooshil A, Sikder N, Mahmud MP, Kouzani AZ, Nahid AA (2020) A systematic review on reinforcement learning-based robotics within the last decade. *IEEE Access* 8:176598–176623.
[11] Vu, M.T., Pham, D.H., Nguyen, V.T., Do, Q.T., Alanazi, A.K. and Nguyen, T.H (2025) Adaptive nonlinear integral-backstepping control for frequency stabilization in cyber-physical shipboard microgrids using double deep Q-learning. *Engineering Applications of Artificial Intelligence*, 160, p.111943.
[12] Gu Y, Cheng Y, Chen CP, Wang X (2021) Proximal policy optimization with policy feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52(7):4600–4610.
[13] Kiran BR, Sobh I, Talpaert V, Mannion P, Al Sallab AA, Yogamani S, Pérez P (2021) Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* 23(6):4909–4926.
[14] Cao H, Xiong H, Zeng W, Jiang H, Cai Z, Hu L, Zhang L, Lu W (2023) Safe reinforcement learning-based motion planning for functional mobile robots suffering uncontrollable mobile robots. *IEEE Transactions on Intelligent Transportation Systems* 25(5):4346–4363.
[15] Ding W, Zhang JX, Shi P (2024) Adaptive fuzzy control of wheeled mobile robots with prescribed trajectory tracking performance. *IEEE Transactions on Fuzzy Systems* 32(8):4510–4521.
[16] Xiao H, Chen C, Zhang G, Chen CP (2024) Reinforcement learning-driven dynamic obstacle avoidance for mobile robot trajectory tracking. *Knowledge-Based Systems* 297:111974.