

Reinforcement Learning-Based Control for Smooth and Accurate Circular Trajectory Tracking in Mobile Robots

Le Thi Minh Tam[✉], Nguyen Viet Ngu^{*}, Nguyen Ba Dung[✉]

Hung Yen University of Technology and Education, Vietnam

^{*}Corresponding author. Email: ngunguyenviet77@gmail.com

ARTICLE INFO		ABSTRACT
Received:	15/05/2026	This paper investigates reinforcement learning-based control for smooth and accurate trajectory tracking of a nonholonomic mobile robot. Three reinforcement learning algorithms, Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Soft Actor-Critic (SAC), are implemented and evaluated, with conventional PID and Sliding Mode Control (SMC) controllers included as comparative baselines. The robot is modeled using unicycle kinematics, and the tracking task is formulated through body-frame longitudinal, lateral, and heading errors. A reward function is designed to penalize tracking errors, radial deviation, and excessive control effort while encouraging forward progression. Simulations are conducted primarily on a circular reference trajectory, with a figure-eight trajectory included for additional comparison. The results show that all controllers can follow the reference trajectories, although their tracking accuracy and motion smoothness differ. DQN exhibits noticeable control switching, while PPO produces smoother actions but retains tracking deviations. Among the evaluated controllers, SAC achieves lower tracking errors and smaller control variations. These findings suggest that SAC is a promising approach for accurate and smooth trajectory tracking in mobile robots.
Revised:	30/06/2026	
Accepted:	29/07/2026	
Online First:	22/10/2026	
Published:		
KEYWORDS		
Mobile robot;		
Trajectory tracking;		
Reinforcement learning;		
Motion smoothness;		
Soft Actor-Critic.		

DOI: <https://doi.org/10.54644/jte.2026.2455>

Copyright © JTE. This is an open access article distributed under the terms and conditions of the [Creative Commons Attribution-NonCommercial 4.0 International License](#) which permits unrestricted use, distribution, and reproduction in any medium for non-commercial purpose, provided the original work is properly cited.

1. Introduction

Mobile robots have become indispensable in various sectors, including warehouse logistics, service robotics, and autonomous transportation [1]. In these applications, the ability to follow a predefined trajectory with high precision is a fundamental requirement for operational safety and efficiency. Among various path types, circular trajectories are of particular interest as they frequently appear in tasks such as docking, surveillance, and maneuvering in constrained spaces [2]. Despite their prevalence, achieving circular trajectory tracking remains a significant challenge due to the nonlinear kinematic constraints of mobile robots and the continuous need to regulate both linear and angular velocities to compensate for lateral deviations [3].

Traditionally, trajectory tracking control has relied on model-driven approaches such as PID and Model Predictive Control (MPC). While PID controllers are widely used for their simplicity, their performance often degrades under nonlinear conditions or when facing parameter uncertainties [4]. MPC offers superior constraint handling and predictive capabilities, yet it requires a high-fidelity mathematical model and involves heavy computational overhead. Recently, reinforcement learning (RL) has emerged as a robust alternative for robot control [5]. Model-free RL methods, including PPO and SAC, have demonstrated great potential in handling high-dimensional, continuous control tasks without requiring explicit system dynamics [6].

Despite recent advancements, a critical gap remains in the existing RL-based navigation literature [7]. Most current studies prioritize task completion or reward maximization, often neglecting the smoothness of the resulting motion [8]. Consequently, the learned policies frequently exhibit oscillatory or jerky control actions, commonly referred to as "chattering," which can lead to mechanical wear and instability in real-world deployments [9]. This issue is particularly pronounced when tracking curved trajectories at varying speeds, where small errors in heading can lead to rapid accumulation of tracking

inaccuracies [10]. To address this gap, this paper investigates a reinforcement learning-based control framework for smooth and accurate trajectory tracking. The reward function incorporates position, heading, and radial tracking errors, together with control effort and forward progression. This formulation aims to balance tracking accuracy and control economy during trajectory tracking. Previous studies have investigated robot path planning using instruction-guided probabilistic roadmaps [11] and deep reinforcement learning [12]. Deep reinforcement learning has also been applied to path tracking control of four-wheel independent-steering vehicles [13], while collision-avoidance motion planning for nonholonomic robots in dynamic environments has been addressed in [15]. These studies provide relevant background on robot navigation and control, although their objectives differ from the trajectory tracking problem considered in this work.

The main contributions of this work are summarized as follows:

1. A reward function is formulated to jointly penalize tracking errors and control effort while encouraging forward progression along the reference trajectory.
2. A comparative evaluation of three reinforcement learning algorithms (DQN, PPO, and SAC) is conducted, with PID and SMC included as conventional control baselines. The controllers are assessed in terms of tracking accuracy and motion smoothness under circular and figure-eight reference trajectories
3. Simulation results compare the tracking accuracy and motion smoothness of the evaluated controllers, showing that SAC achieves lower tracking errors and smaller control-input variations under the considered conditions.

The remainder of this paper is organized as follows. Section 2 formulates the kinematic model and the path-tracking problem. Section 3 details the methodology, including the RL architecture and reward design. Section 4 describes the experimental setup and training parameters. Section 5 presents and discusses the comparative results. Finally, Section 6 provides concluding remarks and future research directions.

2. Problem Formulation

2.1. Kinematic model and circular reference trajectory

The mobile robot considered in this study is modeled as a nonholonomic unicycle-type system, which is widely adopted in the mobile robotics literature due to its ability to capture the essential kinematic constraints of wheeled robots [1], [2]. Figure 1 illustrates (a) the geometric formulation of the circular path-tracking problem and (b) the kinematic model of the mobile robot. The mobile robot, modeled by the pose $q = (x, y, \theta)$, is required to follow a circular reference trajectory defined by the reference pose $q_r = (x_r, y_r, \theta_r)$ and radius R .

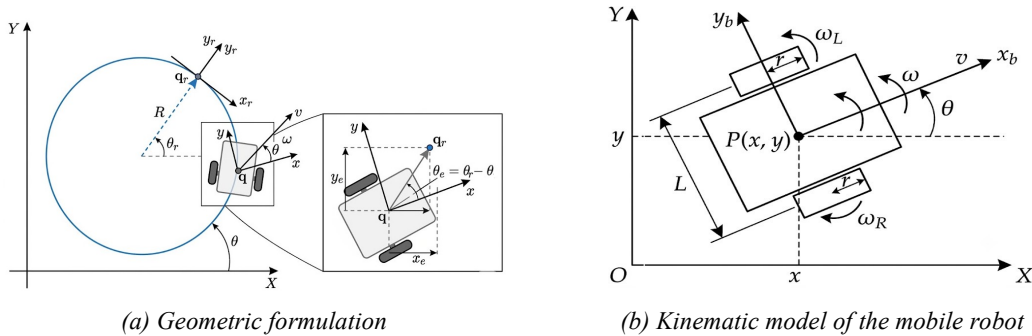


Figure 1. Geometric and Kinematic Models for Mobile Robot circular trajectory tracking:
(a) Geometric formulation of circular trajectory tracking; (b) Kinematic model of the mobile robot.

Let the robot pose in the inertial frame be denoted by: $q(t) = [x(t) \ y(t) \ \theta(t)]^T$, where $x(t)$ and $y(t)$ are the Cartesian coordinates, and $\theta(t)$ is the heading angle. The control inputs are the linear velocity ($v(t)$) and angular velocity ($w(t)$). The robot kinematics are given by:

$$\dot{x}(t) = v(t) \cos \theta(t), \quad \dot{y}(t) = v(t) \sin \theta(t), \quad \dot{\theta}(t) = w(t) \quad (1)$$

The desired path is a circle of radius $R > 0$, centered at the origin. Let $\phi_r(t)$ denote the angular parameter of the reference point moving on the circle. The reference trajectory is defined as

$$x_r(t) = R \cos \phi_r(t), \quad y_r(t) = R \sin \phi_r(t) \quad (2)$$

with the corresponding tangential orientation

$$\theta_r(t) = \phi_r(t) + \frac{\pi}{2} \quad (3)$$

Assuming uniform angular motion along the circle, the reference angular rate is chosen as:

$$\dot{\phi}_r(t) = w_r \quad (4)$$

which implies the reference linear velocity

$$v_r = R w_r \quad (5)$$

Therefore, the reference pose evolves according to

$$\dot{x}_r(t) = v_r \cos \theta_r(t), \quad \dot{y}_r(t) = v_r \sin \theta_r(t), \quad \dot{\theta}_r(t) = w_r \quad (6)$$

Equations (2)–(6) define a circular motion with constant curvature $1/R$, which is a meaningful benchmark for curved-trajectory tracking because it continuously excites both translational and rotational dynamics [1], [2].

2.2. Tracking-error representation

To express the tracking problem in a control-oriented form, the pose error is defined in the robot body-fixed frame. Let the position mismatch in the inertial frame be:

$$\Delta p(t) = \begin{bmatrix} x_r(t) - x(t) \\ y_r(t) - y(t) \end{bmatrix} \quad (7)$$

Using the rotation matrix associated with the robot heading:

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (8)$$

The body-frame tracking errors are defined as:

$$\begin{bmatrix} x_e(t) \\ y_e(t) \end{bmatrix} = R^T \theta(t) \begin{bmatrix} x_r(t) - x(t) \\ y_r(t) - y(t) \end{bmatrix} \quad (9)$$

that is,

$$x_e(t) = (x_r - x) \cos \theta + (y_r - y) \sin \theta \quad (10)$$

$$y_e(t) = -(x_r - x) \sin \theta + (y_r - y) \cos \theta \quad (11)$$

The heading error is defined as:

$$\theta_e(t) = \text{wrap}(\theta_r(t) - \theta(t)) \quad (12)$$

where $\text{wrap}(\cdot)$ maps the angle to $(-\pi, \pi]$.

Differentiating Equations (9)–(12) and using Equation (1) and (6), the error dynamics can be written as:

$$\dot{x}_e = w y_e - v + v_r \cos \theta_e \quad (13)$$

$$\dot{y}_e = -w x_e + v_r \sin \theta_e \quad (14)$$

$$\dot{\theta}_e = w_r - w \quad (15)$$

Equations (13)–(15) show that the trajectory tracking problem is nonlinear and strongly coupled: the lateral error, longitudinal error, and heading error interact through both the robot inputs and the reference motion. This body-frame formulation is standard in nonholonomic trajectory tracking and provides a mathematically convenient basis for controller synthesis [2].

2.3. Control objective

For circular trajectory tracking, it is additionally useful to quantify the deviation from the desired radius. We therefore define the radial error as:

$$e_r(t) = \left| \sqrt{x^2(t) + y^2(t)} - R \right| \quad (16)$$

The overall objective is to design a control policy π such that:

$$x_e(t) \rightarrow 0, \quad y_e(t) \rightarrow 0, \quad \theta_e(t) \rightarrow 0 \quad (17)$$

while keeping $e_r(t)$ small throughout the motion and avoiding excessive control activity. When Equation (17) holds, the robot pose converges to the reference pose on the circular trajectory, implying accurate geometric tracking. However, in practical applications, accurate tracking alone is insufficient; the generated velocity commands should also remain smooth to reduce actuator stress and suppress chattering-like motion. For this reason, the tracking problem addressed in this paper is formulated as a multi-objective optimization task that seeks to jointly minimize path-tracking error and control effort.

3. Methodology

3.1. Reinforcement learning formulation

The circular trajectory tracking task is formulated as a discounted Markov decision process (MDP)

$$M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma) \quad (18)$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability induced by the robot dynamics, r is the reward function, and $\gamma \in (0, 1)$ is the discount factor.

At each time step t , the agent observes the tracking state:

$$s_t = [x_e(t), y_e(t), \theta_e(t)]^\top \quad (19)$$

which compactly characterizes the robot's deviation from the desired circular trajectory. This state representation is sufficient for feedback decision-making because x_e , y_e , and θ_e directly determine the tracking condition in the robot frame.

For PPO and SAC, the action space is continuous:

$$a_t = [v_t, w_t]^\top, \quad v_t \in [-v_{\max}, v_{\max}], \quad w_t \in [-w_{\max}, w_{\max}] \quad (20)$$

For DQN, the action space must be discrete. Hence, the admissible control set is quantized as

$$A_t = \{(v^{(i)}, w^{(j)}) \mid v^{(i)} \in V, w^{(j)} \in \Omega\} \quad (21)$$

where V and Ω are finite sets of linear and angular velocity levels, respectively. The discrete action taken by DQN is thus an index pointing to a predefined pair $v^{(i)}, w^{(j)}$. This distinction is central to the present study because action quantization may limit the controller's ability to generate small corrective commands near the reference path, thereby affecting tracking smoothness.

3.2. Reward design and optimization objective

The design of the reward function is the key mechanism through which reinforcement learning incorporates control objectives into the policy optimization process. In this work, the reward is constructed to encourage accurate trajectory tracking, radial consistency with the desired circle, moderate control effort, and forward progression along the tangent direction. The instantaneous reward is defined as:

$$r_t = -k_p e_{p,t} - k_\theta |\theta_{e,t}| - k_r |e_{r,t}| - k_u (v_t^2 + w_t^2) + k_f v_t \cos \theta_{e,t} + \beta \mathbf{1}_{\{|e_{r,t}| < \delta_r\}} \quad (22)$$

The reward function penalizes position, heading, and radial tracking errors, together with control-input magnitude, while encouraging forward progression along the reference trajectory. Motion smoothness is evaluated separately through variations in consecutive control commands

$$\text{Where } e_{p,t} = \sqrt{x_{e,t}^2 + y_{e,t}^2} \quad (23)$$

where $e_{p,t}$ is the Euclidean tracking error in the robot frame, $k_p, k_\theta, k_r, k_u, k_f > 0$ are weighting coefficients, $\beta > 0$ is a bonus constant, δ_r is a radial tolerance, and $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function.

The first three penalty terms in Equation (22) directly reduce the position, heading, and radial tracking errors. The quadratic control penalty $k_u (v_t^2 + w_t^2)$ regularizes the input magnitude, which discourages unnecessarily aggressive actuation. The progression term $k_f v_t \cos \theta_{e,t}$ rewards motion aligned with the reference tangent, thereby preventing the trivial behavior of remaining nearly stationary to avoid penalties. Finally, the bonus term reinforces stable motion near the desired circle.

The long-term objective of the agent is to maximize the expected discounted return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{N-1} \gamma^t r_t \right] \quad (24)$$

Since r_t in Equation (22) is the negative of a weighted tracking-and-effort cost plus a progression incentive, maximizing $J(\pi)$ is equivalent to learning a policy that balances tracking accuracy, radial consistency, and control economy over the entire episode. This formulation is especially suitable for circular tracking, where small instantaneous errors may accumulate over time if the controller does not account for the long-term consequences of each action.

3.3. Learning algorithms

Among model-free deep reinforcement learning methods, DQN, PPO, and SAC are representative baselines for value-based, on-policy policy-gradient, and off-policy entropy-regularized learning, respectively. These methods have been widely adopted in sequential decision-making and robot control tasks [3]–[6].

3.3.1. Deep Q-Network

DQN approximates the action-value function $Q(s, a)$ by a neural network and selects actions according to the greedy policy induced by the learned Q-function. The optimal action-value function satisfies the Bellman optimality equation [9].

$$Q^*(s_t, a_t) = \mathbb{E} \left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t, a_t \right] \quad (25)$$

The DQN network parameters ψ_D are trained by minimizing the temporal-difference loss:

$$L_{\text{DQN}}(\psi_D) = \mathbb{E} \left[(y_t - Q(s_t, a_t; \psi_D))^2 \right] \quad (26)$$

$$\text{with target: } y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \psi_D^-) \quad (27)$$

where ψ_D^- denotes the parameters of the target network. In the present problem, DQN can only operate on the discretized action set in Equation (21). Therefore, although it can learn a stable policy, its finite action resolution may produce switching behavior when the robot is close to the desired trajectory.

3.3.2. Proximal Policy Optimization

PPO parameterizes the stochastic policy $\pi_\eta(a \mid s)$ where η denotes the policy network parameters. The policy is updated using a clipped surrogate objective to constrain excessively large policy updates. The PPO objective is given by [6].

$$L_{\text{PPO}}^{\text{clip}}(\eta) = \mathbb{E}_t \left[\min \left(\rho_t(\eta) \hat{A}_t, \text{clip}(\rho_t(\eta), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \quad (28)$$

where $\rho_t(\phi)$ is the probability ratio defined in Equation (29), $\hat{A}_t$ is the estimated advantage function, and $\epsilon > 0$ is the clipping parameter.

$$\rho_t(\eta) = \frac{\pi_\eta(a_t \mid s_t)}{\pi_{\eta_{\text{old}}}(a_t \mid s_t)} \quad (29)$$

$\hat{A}_t$ is the estimated advantage function, and $\epsilon > 0$ is the clipping parameter. Because PPO operates in the continuous action space defined by (20), it can generate finer velocity commands than DQN. However, as an on-policy method, its performance may be more sensitive to rollout quality and sample efficiency.

3.3.3. Soft Actor-Critic

SAC is an off-policy actor-critic algorithm derived from the maximum-entropy RL framework. In addition to maximizing expected return, SAC also maximizes the entropy of the policy, which promotes robust exploration and smoother stochastic control [5]. The SAC objective can be written as:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{N-1} \gamma^t (r_t + \alpha \mathcal{H}(\pi(\cdot \mid s_t))) \right] \quad (30)$$

where $\alpha > 0$ is the SAC entropy temperature parameter, $\mathcal{H}(\cdot)$ denotes policy entropy, and N is the number of time steps in an episode. Let ψ_Q and η denote the parameters of the SAC Q-network and policy network, respectively. The soft Q-target is defined as follows:

$$y_t = r_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi} [Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1} | s_{t+1})] \quad (31)$$

and the policy is updated by minimizing

$$L_{\text{SAC}}(\eta) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_{\eta}} [\alpha \log \pi_{\eta}(a_t | s_t) - Q_{\psi_Q}(s_t, a_t)] \quad (32)$$

Because SAC combines continuous control with entropy regularization, it is expected to be better suited for tracking curved trajectories, tasks in which both precision and actuation smoothness are important.

Figure 2 presents the overall reinforcement learning framework used for circular path tracking. The agent receives the tracking-error state x_e, y_e, θ_e generates the control action v, w and interacts with the robot environment system to maximize a reward function that balances tracking accuracy and motion smoothness.

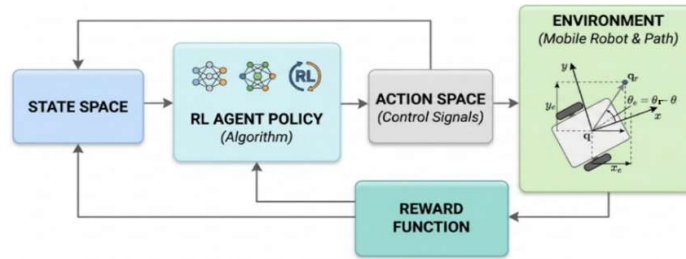


Figure 2. Reinforcement learning framework and experimental setup for circular trajectory tracking

4. Simulation Setup

To evaluate the effectiveness of the proposed reinforcement learning framework, all algorithms were trained and tested in the same circular trajectory tracking environment. The simulation uses a sampling time of $\Delta t = 0.01$ s and each episode lasts 15 s, corresponding to 1500 control steps. The reference trajectory is a circle of radius $R = 10$ m with reference angular velocity $w_r = 0.5$ rad/s and linear velocity:

$$v_r = R w_r = 5 \text{ m/s} \quad (33)$$

For continuous-control methods, the action bounds are defined as:

$$v_t \in [-6, 6] \text{ m/s}, \quad w_t \in [-3, 3] \text{ rad/s} \quad (34)$$

For DQN, the control space is discretized into seven uniformly spaced linear velocity levels, $V = -6, -4, -2, 0, 2, 4, 6$ m/s, and seven angular velocity levels, $\Omega = -3, -2, -1, 0, 1, 2, 3$ rad/s resulting in $|\mathcal{A}_t| = 7 \times 7 = 49$ discrete actions.

4.1. Training Configuration

The neural networks used in PPO, SAC, and DQN employ multilayer perceptrons with two hidden layers of 64 neurons each and ReLU activation functions. PPO is trained with a rollout length of 1,024, a batch size of 256, a learning rate of 3×10^{-4} a discount factor of $\gamma = 0.99$, and a generalized advantage estimation parameter of $\lambda = 0.95$. SAC uses a replay buffer of 200,000 transitions, a batch size of 256, a learning rate of 3×10^{-4} a soft update coefficient of $\tau = 0.005$, and automatic entropy tuning. DQN is trained with a replay buffer of 100,000 samples, a learning rate of 10^{-3} a batch size of 256, a target network update interval of 1,000 steps, and an ϵ greedy exploration schedule with the exploration probability decreasing from 1.0 to 0.05. The reward function in Equation (22) uses the weighting coefficients $k_p, k_{\theta}, k_r, k_u, k_f, |e_{r,t}|$ together with the bonus coefficient β and radial tolerance δ_r . These parameters are kept identical across DQN, PPO, and SAC to ensure a consistent reward formulation during training.

4.2. Evaluation Protocol

After training, each controller is evaluated under the same initial condition and reference trajectory. The robot starts outside the desired circle and is required to converge to the circular path while maintaining stable motion. During evaluation, the learned policies are executed deterministically, and the robot trajectory, tracking errors, and control inputs are recorded over one episode for direct comparison.

4.3. Performance Metrics

The comparative analysis is based on both tracking accuracy and motion smoothness. Let $x_{e,t}$, $y_{e,t}$ and $\theta_{e,t}$ denote the longitudinal, lateral, and heading tracking errors at time step t , respectively, and let N denote the number of time steps in an episode. The mean absolute errors are computed as:

$$\text{MAE}_x = \frac{1}{N} \sum_{t=1}^N |x_{e,t}|, \quad \text{MAE}_y = \frac{1}{N} \sum_{t=1}^N |y_{e,t}| \quad (35)$$

$$\text{MAE}_\theta = \frac{1}{N} \sum_{t=1}^N |\theta_{e,t}| \quad (36)$$

The overall position-tracking error is measured by the root-mean-square error:

$$\text{RMSE}_{xy} = \sqrt{\frac{1}{N} \sum_{t=1}^N (x_{e,t}^2 + y_{e,t}^2)} \quad (37)$$

To assess control economy, the average control effort is defined as:

$$C_u = \frac{1}{N} \sum_{t=1}^N (v_t^2 + w_t^2) \quad (38)$$

Finally, motion smoothness is evaluated through the mean variation of the control inputs:

$$S_v = \frac{1}{N-1} \sum_{t=1}^{N-1} |v_{t+1} - v_t|, \quad S_w = \frac{1}{N-1} \sum_{t=1}^{N-1} |w_{t+1} - w_t| \quad (39)$$

These metrics are applied consistently to DQN, PPO, SAC, PID, and SMC under both reference trajectories, providing a common basis for evaluating tracking accuracy, control effort, and motion smoothness.

5. Results and Discussion

5.1. Trajectory tracking performance

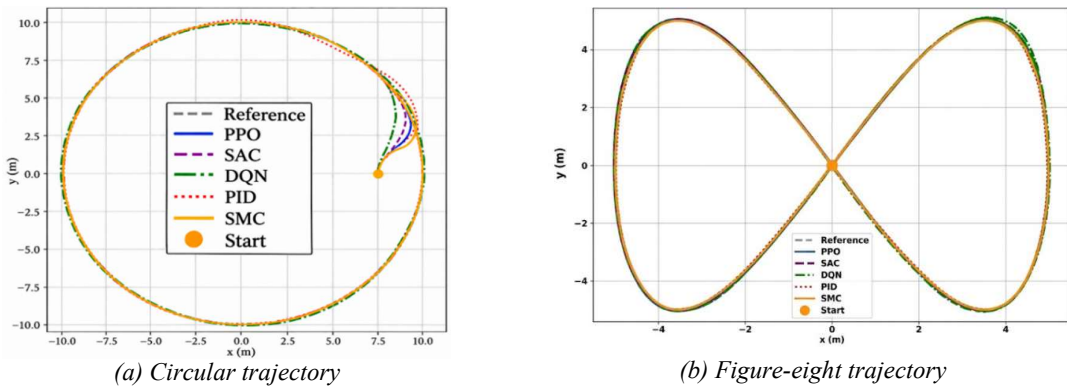


Figure 3. Comparison of trajectory tracking performance of PID, SMC, DQN, PPO, and SAC under two reference trajectories: (a) Circular trajectory and (b) Figure-eight trajectory.

Figure 3 compares the trajectory tracking performance of five controllers (PID, SMC, DQN, PPO, and SAC) for two reference trajectories: a circular path and a figure-eight path. As shown in Figure 3a, all controllers guide the robot toward the desired circular trajectory, although their tracking accuracy differs. PID exhibits larger deviations during the initial convergence phase, while SMC improves the response but retains visible oscillations. Among the reinforcement learning methods, SAC follows the reference trajectory more closely than DQN and PPO. Similar trends are observed for the figure-eight trajectory in Figure 3b, where SAC maintains smaller tracking deviations around curved and crossing regions. These observations indicate that SAC provides favorable tracking performance among the evaluated controllers.

5.2. Tracking error analysis

Figure 4 compares the longitudinal x_e , lateral y_e , and heading θ_e tracking errors of PID, SMC, DQN, PPO, and SAC under the circular and figure-eight trajectories. PID exhibits the largest transient and steady-state errors, while SMC improves tracking performance but still shows noticeable oscillations. DQN and PPO achieve better accuracy than the conventional controllers, although DQN presents stronger error fluctuations. SAC consistently maintains the lowest tracking errors and the smoothest responses for both trajectories, demonstrating the best overall tracking performance.

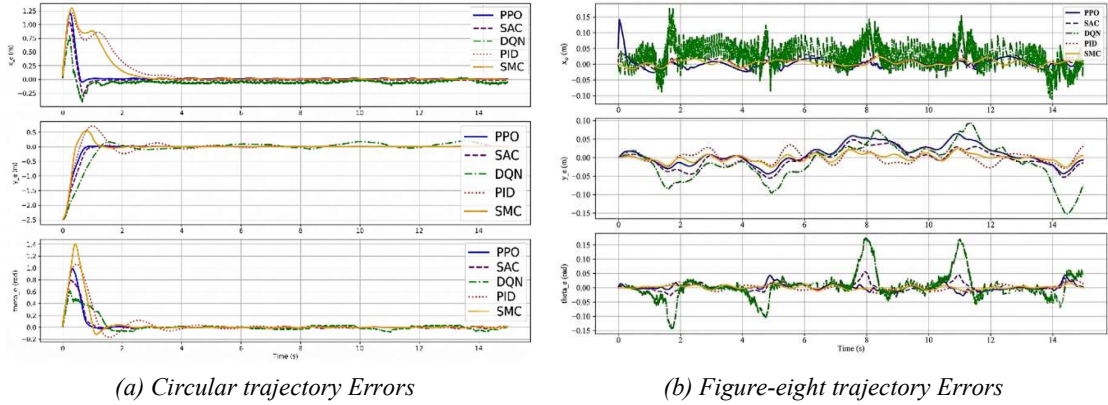


Figure 4. Comparison of longitudinal, lateral, and heading tracking errors of PID, SMC, DQN, PPO, and SAC for (a) Circular trajectory Errors and (b) Figure-eight trajectory Errors

5.3. Control input behavior

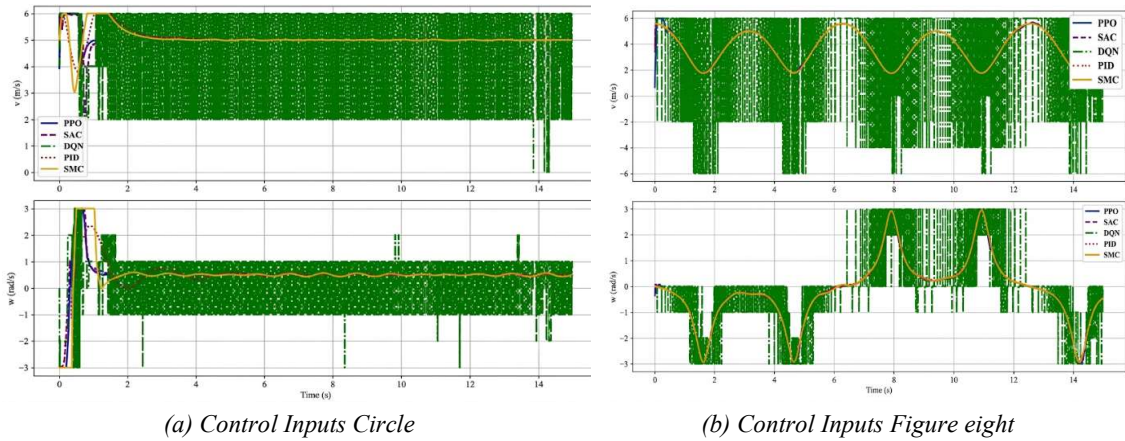


Figure 5. Comparison of linear and angular control inputs generated by PID, SMC, DQN, PPO, and SAC under (a) circular and (b) figure-eight trajectories.

Figure 5 compares the linear and angular control inputs generated by DQN, PPO, and SAC, together with PID and SMC as conventional baselines. PID and SMC produce relatively smooth signals but

require larger control adjustments during transient motion. DQN exhibits frequent switching associated with its discrete action space, whereas PPO produces smoother commands. SAC maintains more stable control inputs with smaller variations under both reference trajectories, indicating a favorable balance between tracking accuracy and control smoothness.

5.4. Motion smoothness

Figure 6 compares motion smoothness based on linear and angular acceleration variations. PID and SMC exhibit noticeable transient variations, while DQN shows larger fluctuations associated with discrete action switching. PPO improves motion smoothness compared with DQN, whereas SAC maintains smaller acceleration variations under both reference trajectories. These results suggest that SAC provides smoother control behavior in the considered simulation scenarios.

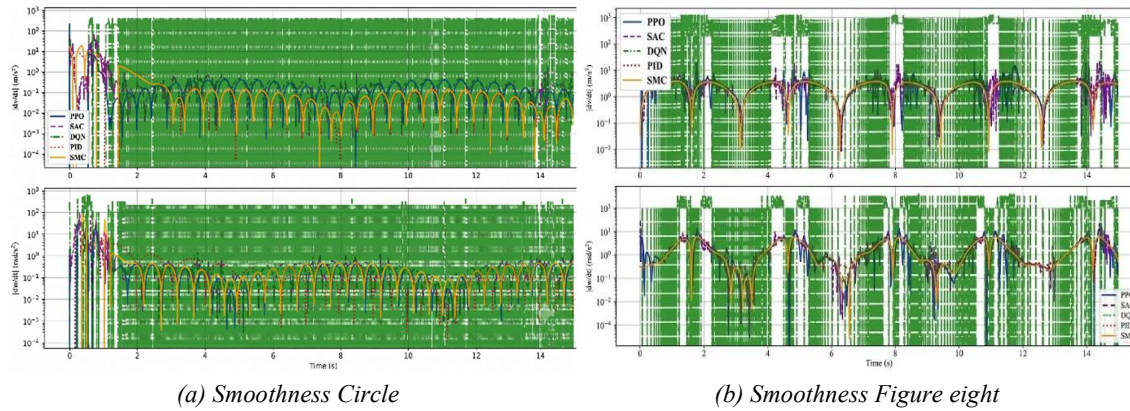


Figure 6. Comparison of motion smoothness based on linear and angular acceleration variations of PID, SMC, DQN, PPO, and SAC under (a) circular and (b) figure-eight trajectories.

Overall, Figures 3–6 demonstrate the tracking performance and motion smoothness of the three reinforcement learning algorithms compared with the conventional PID and SMC baselines. Among the reinforcement learning methods, SAC achieves the most favorable balance between tracking accuracy and control smoothness under both reference trajectories.

6. Conclusion and Future Work

This paper presented a comparative study of three reinforcement learning algorithms, DQN, PPO, and SAC, for trajectory tracking of a nonholonomic mobile robot, with PID and SMC included as conventional control baselines. Although all controllers successfully followed the reference paths, SAC achieved the best overall performance with the lowest tracking errors, the smoothest control inputs, and the highest motion stability. These results demonstrate the effectiveness of continuous-action reinforcement learning for accurate and smooth mobile robot trajectory tracking. Future work will focus on obstacle avoidance in dynamic environments and experimental validation on a real mobile robot platform.

Conflict of Interest

The authors declare that they have no conflict of interest.

REFERENCES

- [1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2011.
- [2] J. A. Abdulsahab and D. J. Kadhim, "Classical and Heuristic Approaches for Mobile Robot Path Planning: A Survey," *Robotics*, vol. 12, no. 4, Art. no. 93, 2023, doi: 10.3390/robotics12040093.
- [3] G. Yang and Y. Guo, "Deep Reinforcement Learning Based Mobile Robot Navigation in Crowd Environments," in *Proc. 21st Int. Conf. Ubiquitous Robots (UR)*, 2024, pp. 248–254, doi: 10.1109/UR61395.2024.10597510.
- [4] Y. Zhang, C. Li, G. Zhang, R. Zhou, and Z. Liang, "Research on the Local Path Planning for Mobile Robots Based on PRO-Dueling Deep Q-Network (DQN) Algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 8, pp. 381–389, 2023, doi: 10.14569/IJACSA.2023.0140842.

- [5] T. Haamoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, ser. Proceedings of Machine Learning Research (PMLR), vol. 80, 2018, pp. 1861–1870.
- [6] T. M. T. Le, D. H. Pham, and V. N. Nguyen, "PPO Reinforcement Learning for Smooth Control of Two-Wheeled Mobile Robot," *J. Meas., Control Autom.*, vol. 30, no. 1, pp. 57–65, 2026.
- [7] Y. Zhu, X. Wang, and H. Li, "Deep Reinforcement Learning of Mobile Robot Navigation: A Review," *Sensors*, vol. 25, no. 11, Art. no. 3394, 2025, doi: 10.3390/s25113394.
- [8] X. Yu, Y. Fan, S. Xu, and L. Ou, "A Self-adaptive SAC-PID Control Approach Based on Reinforcement Learning for Mobile Robots," arXiv:2103.10686, 2021.
- [9] T. Nakamura, M. Kobayashi, and N. Motoi, "Path Planning for Mobile Robot Considering Turnabouts on Narrow Road by Deep Q-Network," *IEEE Access*, vol. 11, pp. 19111–19121, 2023, doi: 10.1109/ACCESS.2023.3247730.
- [10] H. Xu, T. Terakawa, and M. Komori, "Deep-Reinforcement-Learning-Based Trajectory Tracking Control for Slidable-Wheel Omnidirectional Mobile Robot," *J. Adv. Mech. Des., Syst., Manuf.*, vol. 19, no. 4, Art. no. JAMDSM0031, 2025, doi: 10.1299/jamdsm.2025jamdsm0031.
- [11] J. Bao and R. Yonetani, "Path Planning using Instruction-Guided Probabilistic Roadmaps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2025, pp. 1–7.
- [12] H. Liu, Y. Shen, S. Yu, Z. Gao, and T. Wu, "Deep Reinforcement Learning for Mobile Robot Path Planning," arXiv:2404.06974, 2024.
- [13] X. Hua, T. Zhang, X. Cheng, and X. Ning, "Path Tracking Control for Four-Wheel Independent Steering and Driving Vehicles Based on Improved Deep Reinforcement Learning," *Technologies*, vol. 12, no. 11, Art. no. 218, 2024, doi: 10.3390/technologies12110218.
- [14] Y. Hu and S. X. Yang, "A Novel Knowledge-Based Genetic Algorithm for Robot Path Planning in Complex Environments," arXiv:2209.01482, 2022.
- [15] L. Zhao, J. Zhao, Z. Liu, Y. Wang, and H. Liu, "Solving the Real-Time Motion Planning Problem for Non-Holonomic Robots With Collision Avoidance in Dynamic Scenes," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9429–9436, Oct. 2022, doi: 10.1109/LRA.2022.3194313.

Le Thi Minh Tam received the B.S. degree in Control and Automation Engineering from Thai Nguyen University of Technology, Vietnam, in 2000, and the Ph.D. degree in Control Engineering and Automation from the Military Technical Academy, Vietnam, in 2018. The author is affiliated with the Faculty of Electrical and Electronic Engineering, Hung Yen University of Technical and Education, Vietnam. The author's research interests include intelligent control systems, deep learning, robotics, and advanced control algorithm development.

Email: leminhtamutehy@gmail.com. ORCID:  <https://orcid.org/0009-0000-5147-5370>.

Nguyen Viet Ngu received the B.S. degree in Agricultural and Rural Electrification from Vietnam Academy of Agriculture, Vietnam, in 2001, and the Ph.D. degree in Electrical Systems and Automation from Hohai University, Nanjing, China, in 2013. The author is affiliated with the Faculty of Electrical and Electronic Engineering, Hung Yen University of Technical and Education, Vietnam. The author's research interests include advanced power electronics control, renewable energy system optimization, robotics, and intelligent control methods.

Email: ngunguyenviet77@gmail.com. ORCID:  <https://orcid.org/0009-0004-6012-4846>.

Nguyen Ba Dung is currently pursuing graduate studies in Automation at Hung Yen University of Technology and Education, Vietnam. The author's research interests include automation and control engineering.

Email: dungnguyen14012003@gmail.com. ORCID:  <https://orcid.org/0009-0003-1976-9899>.