Autonomous Gain Tuning for Differential Drive Robots Targeting Control using Soft Actor-Critic

Chao-Chung Peng*
Department of Aeronautics and Astronautics
National Cheng Kung University
Tainan, Taiwan
ccpeng@mail.ncku.edu.tw

Meng-Huan Chiang
Department of Aeronautics and Astronautics
National Cheng Kung University
Tainan, Taiwan
P46114361@gs.ncku.edu.tw

Yi-Ho Chen
Department of Aeronautics and Astronautics
National Cheng Kung University
Tainan, Taiwan
p48101500@gs.ncku.edu.tw

***Abstract*—Differential drive robots (DDRs) belong to a unique category of mobile robots that regulate their speed and direction by independently adjusting the speeds of two wheels. Due to their high maneuverability, DDRs can execute various missions requiring precise positioning and navigation. To guide DDRs in target tracking, the Approximate Pose Increment Control (APIC) is applied to provide reference direction and speed control based on the Line of Sight (LOS) guidance principle. However, the ordinary APIC is unable to consider the physical constraints on the DDRs, potentially resulting in poor tracking performance when the target is near the DDR. One of the most common failure scenarios is the "deadlock loop", which prevents DDRs from reaching the target and causes them to keep circling near it due to a too-large turning radius. To address this issue, the Reinforcement Learning (RL) based APIC is proposed, allowing the system to learn optimal actions from the environment to reach the goal. In this approach, a Soft Actor-Critic (SAC) agent is trained to dynamically adjust two gain values in APIC based on real-time observations. The proposed method not only enhances the targeting performance of APIC but also provides an expert guidance law for imitation learning.**

Keywords—differential drive robotics, targeting control, reinforcement learning.

I. INTRODUCTION (*HEADING 1*)

Differential drive robots (DDRs) are a unique type of mobile robot that regulate their speed and direction by independently adjusting the speeds of two wheels. By doing so, DDRs can execute rotations in confined spaces, move laterally, and execute sharp turns. Their remarkable maneuverability and adaptability make DDRs a dependable choice, particularly in scenarios demanding precise navigation, such as warehousing and indoor logistics environments [1]. Approximate pose increment control (APIC) [2] is a control law proposed to guide DDRs to track their targets. In this method, the direction of DDRs are forced to point towards their target and provide a good performance of tracking moving targets.

In this paper, a reinforcement learning (RL) strategy is integrated [3] with the recently presented APIC targeting application. It is well known that the RL is a subfield of machine learning where an agent learns to take appropriate actions when interacting with an environment. Designing rules of reward function is compulsory to provide agents with learning from the environment. Therefore, the goal of agents is to learn a policy or a decision model based on the designated rewards with respect to the environment. RL has been applied to a wide range of domains and has successfully trained agents to play complex games and handle control problems [4].

To apply reinforcement learning to the rule-based APIC algorithm, a game-like environment based on the DDR model is created. In this game environment, DDRs are equipped with a weapon, and the goal of this game is to destroy targets using this weapon. To highlight the main difficulty of the DDR targeting gaming, an additional physical constraint is imposed on the DDR to increase the difficulty of the game. Due to the constraint, simulations will be conducted to illustrate that a pure APIC algorithm will struggle to destroy a fixed target. To enhance the targeting performance, a soft actor-critic (SAC) [5] algorithm is introduced to integrate with the APIC. This integration enables the agent to dynamically adjust two important gains within APIC in real-time. The primary advantage of this proposed method lies in the agent's ability to adapt its motion based on various observations, resulting in improved performance and addressing existing challenges in APIC. A couple of simulations are provided to demonstrate the advantages and effectiveness of the RL-SAC-based APIC.

The rest of this article is organized as follows. In Section II, the descriptions of a DDR are given. A brief review of the recently presented APIC algorithm is introduced and the main drawback of the existing APIC will be highlighted in Section III. In Section IV, the RL-SAC with consideration of the environment, reward functions, and the training framework are presented. The details comparison of the APIC and the RL-SAC-based online gain tuning APIC is presented in Section V. Finally, Section VI concludes this article.

II. DESCRIPTION OF DIFFERENTIAL DRIVE ROBOTS

Fig. 1 shows the configuration of DDR. Based on the coordinates defined in the figure, the kinematics of DDR can be described as:

$$\begin{aligned} \dot{X}_Q(t) &= v_Q \cos\phi(t) \\ \dot{Y}_Q(t) &= v_Q \sin\phi(t) \\ \dot{\phi}(t) &= \omega_Q \end{aligned} \tag{1}$$

where $\dot{X}_Q$, $\dot{Y}_Q$ are the derivative of DDRs' positions in the inertial frame (X_Q,Y_Q) respectively. $\dot{\phi}(t)$ is the derivative of the heading angle $\phi(t)$ of DDR with respect to the inertial x-axis. v_Q and ω_Q represent the translational and rotational velocity in body fame, respectively, which can be calculated by:

$$\begin{bmatrix} v_Q \\ \omega_Q \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 \\ 2\alpha^{-1} & -2\alpha^{-1} \end{bmatrix} \begin{bmatrix} v_R(t) \\ v_L(t) \end{bmatrix} \triangleq \mathbf{M} \begin{bmatrix} v_R(t) \\ v_L(t) \end{bmatrix} \tag{2}$$

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00056

where $v_R(t)$, $v_L(t)$ and α are the right/left wheel speeds and half of the wheel track width, respectively. The heading speed and direction of DDR are controlled by the right/left wheel speeds. According to (2), given command v_Q and ω_Q, the wheel speeds $v_R(t)$ and $v_L(t)$ can be derived by

$$\begin{bmatrix} v_R(t) \\ v_L(t) \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} v_Q \\ \omega_Q \end{bmatrix} \tag{3}$$

Eq. (3) can be used to generate commands that drive the motors, thereby achieving speed control.

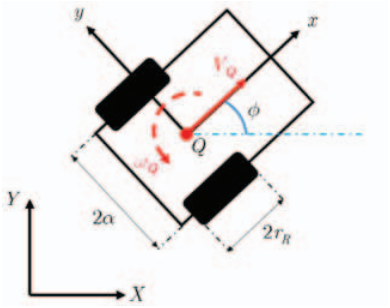


Fig. 1 Configuration of the Differential Drive Robot

III. REVIEW OF APPROXIMATE POSTURE INCREMENT CONTROL

There are many methods proposed to guide DDRs in tracking their targets [1]. The APIC [2] is an approach that considers simple kinematics analyses together with discrete-time dynamics approximation. The main goal of APIC is to force the direction of DDRs to head/pursue toward the targets' location. Using this straightforward design, the control law can guarantee its stability without selecting a proper Lyapunov function and provide a good tracking performance. A bit different from the task presented in [2], the purpose of this research is to make the target fall in the blue attack zone, as illustrated in Fig. 2.

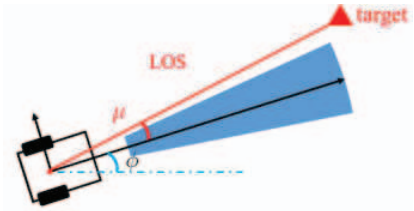


Fig. 2 Definition of line of sight (LOS) and bearing angle μ

Referred to the kinematics equation in (1), the locations of DDRs presented in the discrete-time form are modeled as follows:

$$\begin{aligned} X_Q(k+1) &= X_Q(k) + dt \cdot v_Q(k)\cos\phi(k) \\ Y_Q(k+1) &= Y_Q(k) + dt \cdot v_Q(k)\sin\phi(k) \\ \phi(k+1) &= \phi(k) + dt \cdot \omega_Q(k) \end{aligned} \tag{4}$$

where dt represents the sampling time interval. The posture increment is defined as:

$$\begin{bmatrix} \Delta t_X(k) \\ \Delta t_Y(k) \\ \Delta R(k) \end{bmatrix} = \begin{bmatrix} dt \cdot v_Q(k)\cos\phi(k) \\ dt \cdot v_Q(k)\sin\phi(k) \\ dt \cdot \omega_Q(k) \end{bmatrix} \tag{5}$$

The goal of APIC is to find a proper posture increment defined in (5) at the current time instant such that the controlled DDR can track the posture of the target $(X_{REF}, Y_{REF}, \phi_{REF})$ precisely.

This requirement can be expressed by

$$\begin{aligned} X_Q(k+1) &= X_Q(k) + \Delta t_X(k) \to X_{REF}(k+1) \\ Y_Q(k+1) &= Y_Q(k) + \Delta t_Y(k) \to Y_{REF}(k+1) \\ \phi(k+1) &= \phi(k) + \Delta R(k) \to \phi_{REF}(k+1) \end{aligned} \tag{6}$$

APIC obtains this posture increment through the line of sight (LOS). As shown in Fig. 2, the LOS is the straight line connecting from target and to controlled robot, and it can be calculated by:

$$e_{LOS}(k) \triangleq \begin{bmatrix} X_{REF}(k) \\ Y_{REF}(k) \end{bmatrix} - \begin{bmatrix} X_Q(k) \\ Y_Q(k) \end{bmatrix} \tag{7}$$

The direction of the LOS can be obtained by its unit vector:

$$e_{LOS}^{unit}(k) = e_{LOS}(k) / \left\| e_{LOS}(k) \right\| \tag{8}$$

The moving tangential direction of the controlled robot can be expressed by:

$$F_{Dir}(k) = \begin{bmatrix} \cos(\phi(k)) \\ \sin(\phi(k)) \end{bmatrix} \tag{9}$$

With (8) and (9), the included angle between the target and the controlled robot can be calculated through:

$$\Delta R(k) = sign(\eta_{cross}) \cdot \cos^{-1}\left(\frac{\left\langle F_{Dir}(k), e_{LOS}^{unit}(k) \right\rangle}{\left\| F_{Dir}(k) \right\| \cdot \left\| e_{LOS}^{unit}(k) \right\|} \right) \tag{10}$$

where

$$\eta_{cross} \triangleq \begin{bmatrix} F_{Dir}(k) \\ 0 \end{bmatrix} \times \begin{bmatrix} e_{LOS}^{unit}(k) \\ 0 \end{bmatrix} \tag{11}$$

The final control law of APIC is then described as:

$$\begin{aligned} v_Q(k) &= K_P^{V_Q} \cdot \left\| e_{LOS}(k) \right\| \\ \omega_Q(k) &= K_P^{\omega_Q} \cdot \Delta R(k) \end{aligned} \tag{12}$$

where $K_P^{V_Q}$ and $K_P^{\omega_Q}$ are positive gain values that can be selected by different means. A higher $K_P^{V_Q}$ represents higher closure velocity, and a larger $K_P^{\omega_Q}$ means faster decrement of error between the direction of the controlled robot and LOS.

Through the analysis of the DDR kinematics, APIC is capable of providing appropriate direction and speed control [2]. However, the absence of physical constraints of the DDR, that is, non-zero wheel speeds, could lead to suboptimal targeting performance in certain conditions. Due to the controller limitation, the wheel speeds can only be controlled in a specific range with minimum value $V_{\min}$ and maximum value $V_{\max}$. When the difference between $V_{\min}$ and $V_{\max}$ is small, the turning radius of DDRs can be large, causing DDR

to be stuck in the so-called " deadlock loop", which is shown in Fig. 3. In other words, when the target lies within the turning radius, the controlled DDR will keep orbiting the target and will not be able to target it eventually. Therefore, to address this issue, the paper proposes an online gain-tuning method to improve the targeting performance of the APIC, which will be introduced in the next section.

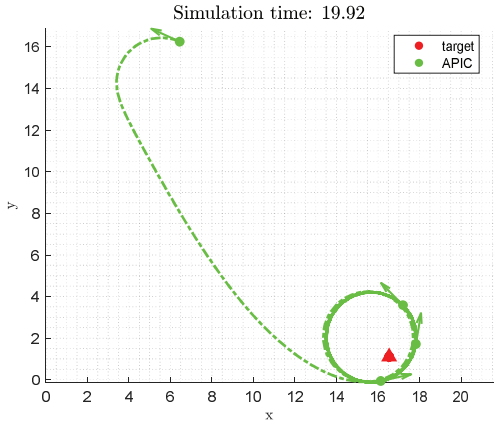


Fig. 3 The "deadlock loop" problem in APIC

IV. INTEGRATED WITH REINFORCEMENT LEARNING AGENT

The deadlock loops problem in DDR is difficult to characterize and solve using analytical methods, leading to challenges in deriving suitable pairs of gain values $K_P^{V_Q}$ and $K_P^{\omega_Q}$ to avoid such a condition. Therefore, this research introduces Reinforcement Learning (RL) to enhance the tracking performance of APIC. During the tracking process, $K_P^{V_Q}$ and $K_P^{\omega_Q}$ are adaptively tuned by RL based on current observation, as shown in Fig. 4. RL is a type of machine-learning algorithm that learns from interaction. An agent takes actions in response to observations from the environment and adjusts its behavior based on the rewards from the environment. The primary objective of the RL agents is to maximize the accumulated reward from the environment. After a specific learning process, the RL can be taken as a powerful and efficient strategy. Hence, the RL is applied to various problems [3], such as game-playing and robotics control [4].

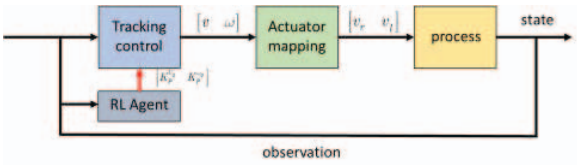


Fig. 4 The proposed structure in this research

This research applies the soft actor-critic (SAC) [5] method for the RL training framework. The SAC is an off-policy actor-critic algorithm that considers entropy when learning during exploration. The term "entropy", is used to measure the uncertainty in a system, which is defined as:

$$\mathcal{H}(\pi) = -\sum \pi(a \mid s) \log \pi(a \mid s) \tag{13}$$

In the SAC, entropy is added to the critic to encourage the actor to act as randomly as possible so the agent can explore the environment deeply. For this reason, SAC is stable in contrast to another off-policy method and can be a promising candidate in real-world robotics tasks [5]. In addition, SAC can be used in continuous action space. All of these characteristics make SAC an ideal algorithm for our task.

This work integrates SAC with APIC by letting SAC agents adjust $K_P^{V_Q}$ and $K_P^{\omega_Q}$ with different observations. The training environment and the reward function are introduced sequentially in this chapter.

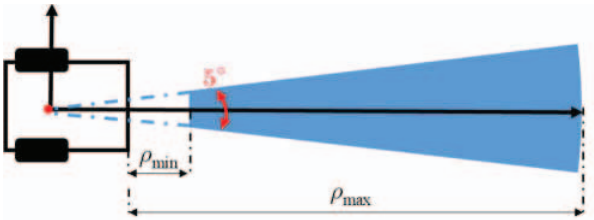


Fig. 5 The effective zone of the weapon

A. Environment

To train SAC agents, a DDR environment is built. The SAC agent is going to control a DDR based on APIC and the kinematic of DDR is calculated and updated through (1)-(4). The task of the agent is to destroy a fixed target with the weapon equipped on the DDR. To complete the mission, the agent needs to bring the targets to its attack zone, which is a blue area in front of the DDR shown in Fig. 5. The attack zone is defined by the distance between $\rho_{\min}$ and $\rho_{\max}$, within a sector spanning 5 degrees. When the target falls into this zone, the DDR will attack the target automatically. The minimum speed of a single wheel is restricted to $[V_{\min}, V_{\max}]$, where $V_{\min} > 0$, which means it can only move forward.

The output of the RL agent is two gains: $K_P^{V_Q} \in [0,10]$ and $K_P^{\omega_Q} \in [0,100]$, respectively. To reduce the frequency of making decisions, down-sampling is applied in the environment. That is, each step in the environment is composed of several timesteps. The overall parameters of the environment are shown in TABLE I.

The observation space is composed of 10 elements. The first four elements are relative position at the time step t and the last time step $t-1$ of the agent and the target. The relative position can be calculated by (7). Other elements in observation space are $\dot{X}_Q$, $\dot{Y}_Q$, ϕ, $\dot{\phi}$ and distance between target and agent denotes ρ. The last element is the bearing angle μ, which represents the angle between the heading direction and LOS as shown in Fig. 2. The bearing angle μ is the absolute value of $\Delta R(k)$ and it is calculated by (7)~(10).

B. Reward function

Reward functions are crucial in the training of RL agents. In this research, two reward functions are developed to help the RL-SAC training.

The first reward function is the hit reward r_H. This reward is given to the agent each time the DDR successfully hits the target with its weapon, which can be expressed as:

$$r_H = \begin{cases} 1, & \text{when targets fall into the weapon zone} \\ 0, & \text{else} \end{cases} \tag{14}$$

TABLE I. PARAMETER OF THE ENVIRONMENT

Parameter	value
Time step interval dt	0.01 (s)
Environment step interval	0.1 (s)
Minimum/maximum velocity $[V_{min}, V_{max}]$	$[5,10]$ (m/s)
Wheel radii r_R, r_L	0.1 (m)
Wheel track width α	0.1 (m)
Weapon range $[\rho_{min}, \rho_{max}]$	$[1,6]$ (m)
Health of target	100
Damage of weapon	1.5 (per 0.01s)

With the addition of a hit reward, the agent can consider the attack zone during control to prevent the deadlock loop. However, the agent may spend more time to destroy the target since the hit reward is a sparse reward that is only given when the condition is satisfied. In this way, the agent may result in some redundant action. To address the issue, a shaped reward is also introduced to make each action more efficient.

Advantage improvement hit reward is a dense reward, which means it will be given to the agent at each step. The advantage can be calculated as follows:

$$A = \frac{-\mu}{\pi} e^{\frac{-|\rho-\rho_w|}{k}} \tag{15}$$

where $\mu \in [0°, 180°]$ is the bearing angle and ρ is the Euclidean distance between the agent and the target. ρ_w is the ideal attack distance and is set to 5 in this environment. Eq. (15) can be visualized in Fig. 6. As the figure shows, the advantage gets larger when μ is small, which means the agent's direction pointing towards the target. When $\mu > 90°$, the target is behind the target, in this situation, the agent needs to increase the distance ρ and decrease μ at the same time to get more advantage.

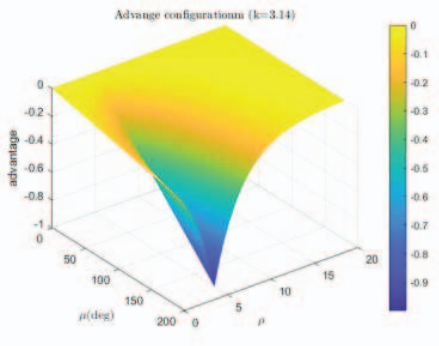


Fig. 6 Plot of advantage

The goal of the advantage improvement reward is to encourage the agent to improve its advantage at each step and penalize it when the action decreases the advantage. To accomplish this, the reward is defined as:

$$r_A = A(k-1) - A(k) \tag{16}$$

As a consequence, the overall reward of a step is:

$$r_{step} = r_H + r_A \tag{17}$$

C. Training Framework

The training framework in this paper was developed in Pytorch (v2.1.0). The SAC agent will be trained in an off-policy way. Usually, a replay buffer is introduced into RL training to stabilize the training process. In this research, we used Prioritize Experience Replay (PER) [6] to accelerate the training process. PER assigns priority to each data stored in the replay buffer. The data with higher priority is more likely to be sampled in the training process. A common approach to assigning priorities is to use the Temporal Difference (TD) error. In this research, the loss of the SAC critic as the metric for assigning priorities was employed. The overall training framework is illustrated in Fig. 7.

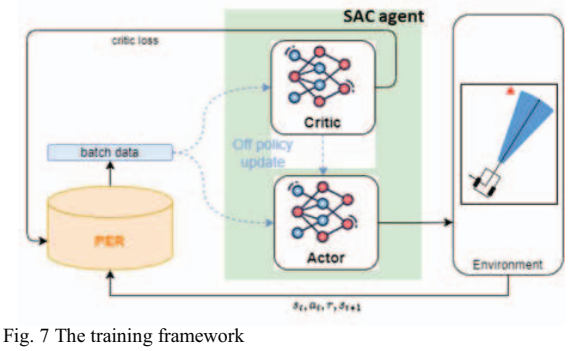


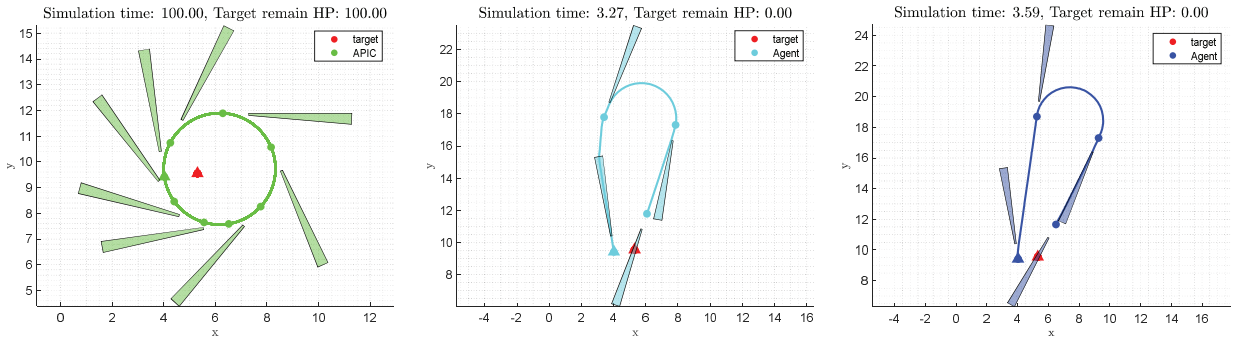
Fig. 7 The training framework

During the training process, the target will be placed randomly in the environment at the start of each episode. The maximum number of steps in one episode is set to 100 to prevent too much sampling data in a single episode. After 100 steps, the environment will truncated and reset automatically. At the start of the training, the network was initialized, and the agent would randomly sample an action from the action space. These randomly moved data will be stored in PER. After sampling 10,000 random data, the training will begin, and the current actor (or policy) will be used to collect new data. The other hyperparameters of the SAC agent are listed in TABLE II.

Fig. 8 Goal achievement rate in training history

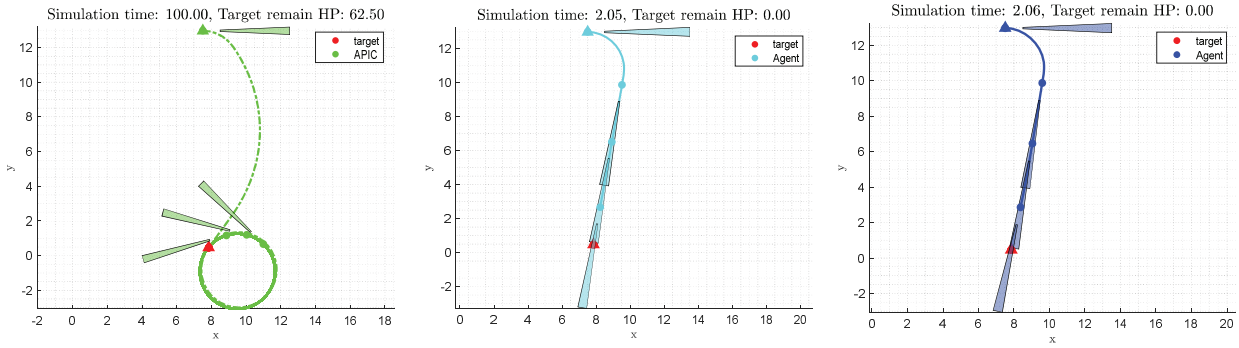
V. SIMULATION RESULTS

In this chapter, the training process of the autonomous gain-tuning agent will be described. Two agents are trained in this research: one is trained with the hit reward function (14) only, and the other is trained by both hit and advantage improvement rewards in (17). The performance of these agents and ordinary APIC are also demonstrated and compared in this chapter.



(a) (b) (c)

Fig. 9 Trajectory of the different models in case 1: (a) is the trajectory of ordinary APIC, which is stuck into a deadlock loop. (b) and (c) are autonomous gain-tuning APIC agents, which are trained with reward functions (14) and (17) respectively. Both agents can avoid the deadlock loop problem and destroy the target in a similar way.



(a) (b) (c)

Fig. 10 Trajectory of the different models in case 2: (a) is the trajectory of ordinary APIC, which is stuck into a deadlock loop again because of rapid closure. (b) and (c) are autonomous gain-tuning APIC agents, which are trained with reward functions (14) and (17) respectively. Both of agents could lower the closure velocity to destroy the target on the first attempt.

TABLE II. HYPER-PARAMETERS USED IN SAC TRAINING

Parameter	value
Optimizer	Adam
Size of Replay Buffer	10e+5
Batch Size	256
Discount factor γ	0.99
# hidden layers	2
Hidden layer size	256
Activation function	ReLu
Learning rate	2e-4
Initial Temperature	0.1
Soft update parameter τ	5e-3

A. *Training process and performance*

During the training process, we evaluate the performance of the current policy in every 2000 training steps. In each evaluation, the current policy interacts with the environment for 10,000 environment steps, and the goal achievement rate of the agent will be calculated based on the result. The performance of using rewards (14) and (17) in the training process are shown in Fig. 8, both the training can converge within 10,0000 steps.

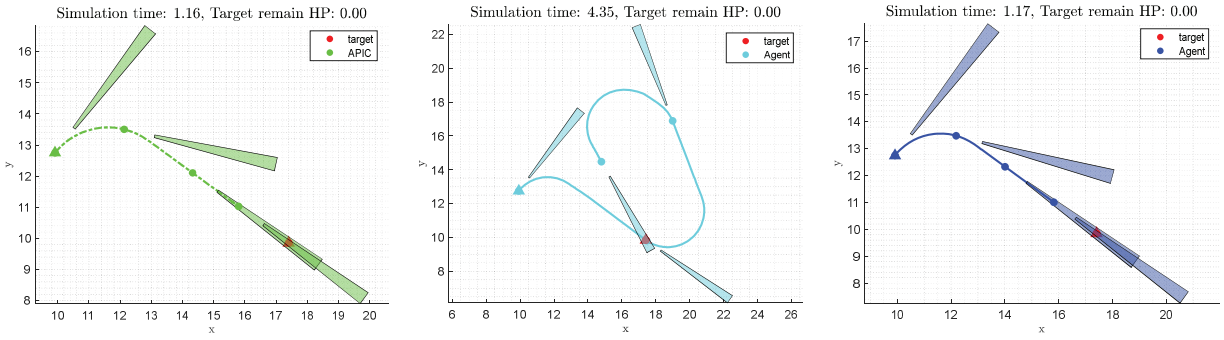
While the convergence of the training process may appear similar, distinctions in the training reward function are reflected in the tracking performance of the agents. The tracking performance is tested by allowing the agent to take 10,000 steps to take action. The initial positions of the target and the agent are randomly set, and a random seed is used to ensure fairness in the performance test. We count how many times the agent destroys the targets and calculate the average time the agent spends to destroy each target. The result is listed in TABLE III. . The result shows that the agent trained by reward (17) can destroy more targets in limited steps, and take less time on average.

B. *Numerical Case studies*

In this section, the targeting trajectories of different initial conditions (ICs) are displayed for comparison purposes under different models. The ordinary APIC with fixed $K_P^{V_Q}$ and $K_P^{\omega_Q}$ is taken as the comparison benchmark, where $\left(K_P^{V_Q}, K_P^{\omega_Q}\right) = \left(3.74,\ 98.77\right)$. In the following, a couple of comparisons will be given to illustrate the targeting behavior under different initial conditions.

TABLE III. COMPARISON OF DIFFERENT AGENTS' PERFORMANCE

Training reward function	Destroyed targets	Average destroyed time
Ordinary APIC	0	-
Hit reward only	380	2.54 (s)
Hit and advantage improvement reward	412	2.36 (s)



(a) (b) (c)

Fig. 11 Trajectory of the different models in case 3: (a) is the trajectory of ordinary APIC, which rapidly destroys the target. (b) and (c) are autonomous gain tuning APIC agents, which are trained with reward functions (14) and (17) respectively. The latter agent can destroy the target a lot faster in this case.

1) Case 1 IC: large bearing angle with a short distance

The first case is that the target is initially set at a short distance at the right-hand side of the controlled DDR. The initial heading of the DDR is pointing upward. This is a classical case of the deadlock loop. As shown in Fig. 9, the ordinary APIC tended to use the maximum turning speed because the included angle between its heading and LOS was large initially. However, due to the physical constraint, it started to orbit the target without hitting it. In contrast, the autonomous gain tuning agents select a small value of $K_P^{\omega_Q}$ at the first, which allows the DDR to elongate the distance. After the distance is far enough, it increases to make a rapid turn and attacks the target with its weapon. The overall process took about 3.5 seconds for both agents.

2) Case 2 IC: large bearing angle with a large distance

Secondly, the target is put at a relatively far distance from the controlled DDR. In this case, the ordinary APIC could successfully damage the target. However, the fixed value of $K_P^{V_Q}$ the DDR resulted in a rapid closure, causing the attack time to be too short to destroy the target during the first attempt. Once the DDR passed the targets, it started to make a rapid turn and became trapped in the deadlock loop again, resulting in mission failure, as shown in Fig. 10. On the other hand, the autonomous gain tuning agents can reduce the value of $K_P^{V_Q}$, thereby decreasing the rate of closure. This ensures sufficient time for the weapon to destroy the targets. Consequently, both agents can successfully destroy the targets on the first attempt.

3) Case 3 IC: Small bearing angle with a large distance

In this initial condition, the APIC can complete the mission, as shown in Fig. 11. This time, the agent trained only by hit reward took the longest time to destroy the target. This result emphasizes the importance of the advantage improvement reward. With the shaped dense reward, the RL training framework can optimize each action of the agent, leading to better performance.

VI. CONCLUSION

The paper proposes a hybrid guidance law for DDRs, which combines the rule-based APIC and AI-based action selection. Since the ordinary APIC doesn't take the physical constraints into consideration, DDRs may get trapped in a problem called the deadlock loop problem, leading to failure in the tracking tasks. This study integrates RL with APIC control laws of DDRs. In the proposed framework, an SAC agent is going to tune the two important gains $K_P^{V_Q}$ and $K_P^{\omega_Q}$ dynamically. To train the agent effectively, two reward functions are proposed. The training process can converge fast and provide a good performance in evaluation. The simulation results show that in the proposed framework, gain-tuning APIC will no longer fall into the deadlock loop problem. The proposed hybrid guidance law improves the tracking capabilities of APIC and also offers an expert control law for imitation learning, which can lower the expenses of human data collection. The imitation learning result can serve as a pre-trained AI model that can be further improved by further training

ACKNOWLEDGMENT

The authors would like to express their deepest gratitude to the National Cheng Kung University (NCKU), and the National Science and Technology Council under the project numbers NSTC 111-2923-E-006 -004 -MY3 and 112-2221-E-006-104-MY3 for their financial support in carrying out this research project.

REFERENCES

[1] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in Proceedings., IEEE International Conference on Robotics and Automation, 1990: IEEE, pp. 384-389.

[2] C.-C. Peng, "Approximate Posture Increment Control for the Targeting Task of Differential Drive Robots," in *2023 IEEE 6th International Conference on Knowledge Innovation and Invention (ICKII)*, 2023: IEEE, pp. 755-760.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine,* vol. 34, no. 6, pp. 26-38, 2017.

[4] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.

[5] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905,* 2018.

[6] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952,* 2015.