




Harnessing traditional controllers for fast-track training of deep reinforcement learning control strategies

Md Shadab Alam & Ignacio Carlucho


To cite this article: Md Shadab Alam & Ignacio Carlucho (18 Jun 2024): Harnessing traditional controllers for fast-track training of deep reinforcement learning control strategies, Journal of Marine Engineering & Technology, DOI: [10.1080/20464177.2024.2367276](https://doi.org/10.1080/20464177.2024.2367276)

To link to this article: <https://doi.org/10.1080/20464177.2024.2367276>

 View supplementary material [↗](#)

 Published online: 18 Jun 2024.

 Submit your article to this journal [↗](#)

 Article views: 24

 View related articles [↗](#)

 View Crossmark data [↗](#)



Harnessing traditional controllers for fast-track training of deep reinforcement learning control strategies

Md Shadab Alam ^a and Ignacio Carlucho^b

^aDepartment of Industrial Design, Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands; ^bSchool of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, United Kingdom

ABSTRACT

In recent years, Autonomous Ships have become a focal point for research, specifically emphasizing improving ship autonomy. Machine Learning Controllers, especially those based on Reinforcement Learning, have seen significant progress. However, addressing the substantial computational demands and intricate reward structures required for their training remains critical. This paper introduces a novel approach, “Harnessing Traditional Controllers for Fast-Track Training of Deep Reinforcement Learning Control Strategies,” aimed at bridging conventional maritime control methods with cutting-edge DRL techniques for vessels. This innovative approach explores the synergies between stable traditional controllers and adaptive DRL methodologies, known for their complexity handling capabilities. To tackle the time-intensive nature of DRL training, we propose a solution: utilizing existing traditional controllers to expedite DRL training by cloning behavior from these controllers to guide DRL exploration. We rigorously assess the effectiveness of this approach through various ship maneuvering scenarios, including different trajectories and external disturbances like winds. The results unequivocally demonstrate accelerated DRL training while maintaining stringent safety standards. This approach has the potential to bridge the gap between traditional maritime practices and contemporary DRL advancements, facilitating the seamless integration of autonomous systems into naval operations, with promising implications for enhanced vessel efficiency, cost-effectiveness, and overall safety.

ARTICLE HISTORY

Received 15 December 2023
Accepted 8 June 2024

KEYWORDS

Reinforcement learning;
Behavioural cloning;
Autonomous vehicle; Path
following; MMG model;
Traditional control

1. Introduction

In recent years, remarkable strides in computational capabilities and the emergence of innovative Machine Learning (ML) and Deep Learning (DL) techniques have opened up a vast array of possibilities for Artificial Intelligence (AI)-based control systems. This transformation has been particularly noticeable in the automation field, where Reinforcement Learning (RL) has gained substantial traction as a focal point for research and exploration.


A significant body of literature already delves into utilising Deep Reinforcement Learning (DRL) controllers for under-actuated ships. Some contributions have successfully demonstrated the applicability of these controllers in tasks such as path following (Jose et al. 2023; Sudha et al. 2023), collision avoidance (Alam et al. 2023), and even adherence to COLREGs (Collision Regulations at Sea) (Meyer et al. 2020). However, it is essential to acknowledge that the computational demands of training these algorithms are substantial (Ladosz et al. 2022). Since RL algorithms acquire knowledge based on past experiences, considering states, actions, and reward pairings, the training process often necessitates an extended duration.

Conventional autopilots rely on line of sight (LOS) guidance systems and proportional-integral-derivative (PID) controllers to achieve waypoint tracking for path following (Moreira et al. 2007; Lekkas and Fossen 2012; Mohan and Somayajula 2023). Traditional methods are often favoured in scenarios with well-defined paths and visible obstacles, where system dynamics are predictable, and the control objectives can be expressed through mathematical relationships. A path-planning algorithm is used on top of this controller

layer to determine the desired waypoints and path modifications in the presence of static and dynamic obstacles. However, dynamically updating the path in real time is a significant challenge that requires robustness and computational power. With the emergence of AI-based control strategies, a single controller can perform control and path planning functions without significant real-time computations. This approach has shown promise for specific applications, such as active heave compensation (Zinage and Somayajula 2020, 2021) and dynamic positioning system (Lee et al. 2020).

Reinforcement Learning (RL) techniques (Sutton and Barto 2018) have gained prominence for addressing path-planning challenges across diverse domains. Several studies have explored RL's potential in this context, each with distinct contributions. For example, Wang et al. (2018) applied Q-learning to path planning, successfully avoiding static obstacles but neglecting vessel dynamics. Meanwhile, Shen et al. (2019) focussed on collision avoidance using deep Q-learning, albeit with waypoint tracking limitations and a lack of real-world validation. In contrast, Sivaraj et al. (2022, 2023) employed Deep Q-networks (DQN) for precise path and heading control of a KVLCC2 tanker in various conditions, outperforming PID controllers. Similarly, Sudha et al. (2023) applied Proximal Policy Optimization (PPO) algorithm, while Jose et al. (2023) utilised Deep Deterministic Policy Gradient (DDPG) algorithm using Stable Baseline 3 (Raffin et al. 2021) for ship path following, showcasing their effectiveness against disturbances and compared it against a PD controller with ILOS guidance system. Chen et al. (2019) enhanced path following for under-actuated cargo ships using Q-learning, outperforming

CONTACT Md Shadab Alam  m.s.alam@tue.nl

 Supplemental data for this article can be accessed online at <https://doi.org/10.1080/20464177.2024.2367276>.

traditional algorithms. Woo et al. (2019) developed a DDPG-based steering controller and evaluated it on an unmanned surface vehicle (WAM-V) but did not compare it to conventional methods. Additionally, Martinsen and Lekkas (2018) applied DDPG for path following and transfer learning in multiple vessel scenarios, showing superior rewards compared to Line-of-Sight (LOS) guidance. Zhou et al. (2019) used DQN for USV path planning, emphasizing kinematics and collision avoidance. Lastly, Zhao et al. (2019) implemented the PPO algorithm for path following and COLREGs adherence, comparing it to the traditional PID controller.

One of the primary challenges in the realm of Reinforcement Learning (RL) lies in the necessity for a substantial number of interactions with the environment. A recent study by Andres et al. (2023) addressed this challenge by employing imitation learning from a replay buffer, thereby enhancing sample efficiency in Procedurally Generated Content (PCG) environments. Similarly, Morales and Sammut (2004) successfully demonstrated the functionality of a flight system by combining reinforcement learning and behavioural cloning. In a comprehensive literature review, Bain and Sammut (1995) clearly illustrated that by capturing traces of human behaviour, it is feasible to construct efficient and robust controllers. Additionally, Model-Based Reinforcement Learning (MBRL) has also yielded intriguing results. For instance, Lambert et al. (2019) showcased that low-level control of a quadrotor can be attained through MBRL techniques. Although they achieved successful quadrotor control, it's noteworthy that they did not compare their MBRL controller with other available controllers in the same study. In another recent work, Wang et al. (2022) provided a comprehensive review of the latest advancements in theoretical analyses, algorithms, and applications of Multi-Agent Reinforcement Learning (MARL) (Ahmed et al. 2022).

Despite the advancements in utilising DRL-based controllers for ship path following, several limitations and gaps remain in the existing literature. Primarily, the incorporation of complex reward functions requires a human expert. Furthermore, the design process can take significant time and effort, as well as a deep understanding of the problem at hand, potentially hindering widespread adoption and applicability. Moreover, the computational demands associated with training DRL algorithms, coupled with the sensitivity of hyperparameters, pose significant challenges in achieving convergence within a reasonable timeframe. Furthermore, the extended time required to reach convergence underscores the pressing need to reduce the number of iterations for efficient training. Addressing these limitations, our study introduces a novel approach to streamline the training process of DRL-based ship controllers. By leveraging insights from traditional Dynamic Positioning (DP) controllers to expedite learning and enhance sample efficiency, our methodology offers a pragmatic solution to mitigate computational burdens and accelerate convergence.

The remainder of the paper is structured as follows: In Section 2, we delve into the ship's dynamics, the operation of the Reinforcement Learning algorithm, and the associated tools and libraries. Section 3 provides insights into input and output states and the training process. In Section 4, we examine the controller's performance in calm waters and the presence of winds. We present a comparison of the controller with behavioural models in Section 5. Finally, Section 6 summarises the study's findings and discussions, along with a glimpse into future research directions.

2. Background

2.1. Ship dynamics

The study utilises the Krisco Container Ship (KCS) vessel to conduct simulations. The ship's dynamics are mathematically represented

using the MMG (Maneuvering Modeling Group) model developed by Yasukawa and Yoshimura (2015). The model employs 3-DOF non-linear equations of motion to calculate the ship's maneuvering motions, including surge, sway, and yaw. These equations are solved iteratively at each time step using an implicit solver called Runge-Kutta. The commanded rudder angle, denoted as δ_c , is provided as an input at each time step. The non-dimensional coefficients used in the model were obtained from Yoshimura and Masumoto (2012). Alternatively, these coefficients can be determined through system identification methods applied to data collected from freely running ship models, as discussed by Vijay and Somayajula (2022) and Deogaonkar et al. (2023).

The mathematical model of the ship follows the Equation (1).

$$\begin{aligned} (m + m_x)\dot{u} - mvr - mx_G r^2 &= X_H + X_R + X_P \\ (m + m_y)\dot{v} + mx_G \dot{r} + mur &= Y_H + Y_R \\ (I_{zz} + J_{zz})\dot{r} + mx_G v + mx_G ur &= N_H + N_R \end{aligned} \quad (1)$$

Further dynamics of the ship can be referred from Deraj et al. (2023), Alam (2023) and Yasukawa and Yoshimura (2015).

2.2. Proportional-derivative (PD) controller

The Proportional-Derivative (PD) controller is a fundamental control mechanism widely used in various engineering applications. It combines two control actions: proportional (P) and derivative (D) terms. The proportional term provides an output signal proportional to the current error, which is the difference between the desired setpoint and the actual process variable. The derivative term, on the other hand, generates an output proportional to the rate of change of the error. Combining these two terms allows the PD controller to effectively regulate the system's behaviour and improve its response characteristics.

Mathematically, the output of a PD controller can be expressed as:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}, \quad (2)$$

Where:

- $u(t)$ is the controller output,
- $e(t)$ is the error signal (difference between setpoint and process variable),
- K_p is the proportional gain, and
- K_d is the derivative gain.

The proportional gain K_p determines the magnitude of the controller's response to the current error, while the derivative gain K_d influences the response based on the rate of change of the error. Proper tuning of these gains is essential to achieve the desired control performance, balancing stability, overshoot, and response time.

2.3. ILOS guidance system

The Integrated Line Of Sight (ILOS) guidance system is a navigation and control mechanism for guiding missiles, rockets, and uncrewed vehicles (UVs). It combines various sensing, computation, and actuation components to achieve accurate trajectory tracking and target interception. The ILOS system derives its name from maintaining the line of sight (LOS) between the vehicle and its target while integrating additional functionalities for enhanced performance.

The operating principle of the ILOS guidance system revolves around continuously adjusting the vehicle's trajectory to ensure that the LOS with the target remains constant. This is achieved by incorporating feedback from sensors such as inertial measurement units (IMUs), GPS receivers, and vision-based systems to estimate the

vehicle's position, velocity, and orientation relative to the target. The guidance algorithm then computes control commands to steer the car along the desired trajectory, compensating for disturbances such as wind gusts and target maneuvers.

Mathematically, the guidance law implemented in the ILOS system can be expressed as follows:

$$\boldsymbol{\tau}(t) = K_p \cdot \boldsymbol{e}(t) + K_d \cdot \frac{d\boldsymbol{e}(t)}{dt}, \quad (3)$$

Where:

- $\boldsymbol{\tau}(t)$ is the control torque or thrust vector,
- $\boldsymbol{e}(t)$ is the LOS error vector,
- K_p is the proportional gain matrix, and
- K_d is the derivative gain matrix.

2.4. Reinforcement learning

Reinforcement Learning (RL) is a type of machine learning where agents learn to make optimal decisions by interacting with an environment to accumulate rewards (Sutton and Barto 2018). Through trial and error, the agent learns which actions to take in a particular state to maximise rewards. This learning environment is known as a Markov decision process (MDP), suitable for representing ship dynamics. The agent's policy, denoted by $\pi(s)$, governs the action in a state s . An episode is terminated when a maximum number of time steps is reached or if any termination condition is satisfied. The agent's goal is to maximise the cumulative reward obtained in each episode, known as the episode returns.

In RL, the value function $V(s)$ is defined as the expected sum of discounted rewards obtained from a given state, as shown in Equation (4). Here $\gamma \in [0, 1]$ denotes the discount factor that adjusts the weightage given to rewards obtained in future time steps.

$$V(s) = \mathbb{E}_\pi [r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{T-t} r_T \mid s_t = s] \quad (4)$$

Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) is a DRL algorithm that uses the actor-critic framework to extend

Q-learning to a continuous action space. The policy and Q-value functions are estimated by neural networks, namely the actor and critic networks. The actor-network directly represents the agent's policy in policy gradient methods. The actor-network takes as input the current state and outputs the action based on the policy encoded by the network. The critic network takes both the state and action as input to predict the Q-value. DDPG also makes use of target networks like the DQN algorithm.

In DDPG, noise is added to the action to aid the agent in exploring more during training. The behavioural policy can be denoted as follows:

$$a_t = \mu(s_t) + \mathbb{N}_t \quad (5)$$

Where \mathbb{N}_t is the noise added to the policy and $\mu(s_t)$ is the current policy. Noise is usually added through a correlated Ornstein-Uhlenbeck process or an uncorrelated Gaussian distribution.

The critic network uses squared TD error as its loss function. The actor network uses the policy gradient loss, which is given below. In Equation (6), θ^Q denotes the parameters of the critic network, and θ^μ denotes the parameters of the actor-network.

$$\nabla_{\theta^\mu} J = [\nabla_a Q(s, a \mid \theta^Q)]_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu) \Big|_{s=s_t} \quad (6)$$

The DDPG algorithm is specified below in Algorithm 1.

3. Methodology

3.1. Observation states

An observationstate in Deep Reinforcement Learning (DRL) refers to a snapshot or representation of the environment that an agent uses to make decisions. It contains essential information about the surroundings, enabling the agent to choose appropriate actions to achieve its objectives.

In this study, four distinct observation states play a pivotal role in assisting the agent achieve a successful episode termination. These four observation states comprise:

Algorithm 1 DDPG algorithm

- 1: Initialise actor and critic networks and target networks with random parameters θ^μ and θ^Q
 - 2: Initialise target networks with parameters $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
 - 3: Initialise an empty experience replay buffer \mathcal{D}
 - 4: **for** Episode = 1, 2 \dots N **do**
 - 5: **for** $t = 1, 2 \dots T$ **do**
 - 6: Choose action a_t according to (5)
 - 7: Obtain reward r_t and next state s_{t+1}
 - 8: Add transition (s_t, a_t, r_t, s_{t+1}) to replay buffer \mathcal{D}
 - 9: End episode if termination conditions are met
 - 10: Sample M random transitions from \mathcal{D}
 - 11: Update the critic network using the loss:
 - 12: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i \mid \theta^Q))^2$ where
 - 13: $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^Q)$
 - 14: Update the actor-network:
 - 15: $\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a \mid \theta^Q) \Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s \mid \theta^\mu) \Big|_{s_i}$
 - 16: **if** time step % target update frequency ==0 **then**
 - 17: Update target networks:
 - 18: $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$
 - 19: $\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$
 - 20: **end if**
 - 21: **end for**
 - 22: **end for**
-

- (1) Cross-track error
- (2) Course angle error
- (3) Distance to the destination
- (4) Yaw rate

It is worth noting that these four specific observation states, as elucidated in Alam and Somayajula (2024), are both essential and adequate for enabling an under-actuated ship to navigate to its intended destination effectively.

3.2. Action states

The commanded rudder angle, δ_c , is the action the agent can choose at every step. In this study, the DDPG or PD agent gives continuous action states between $[-35^\circ, 35^\circ]$. So, at every timestep, the RL agent executes one of the actions. The agent only controls the commanded rudder angle, and the actual rudder angle (δ) still varies smoothly as governed by Equations (7) and (8).

$$T_R \dot{\delta} + \delta = \delta_c \quad (7)$$

$$\dot{\delta} = \begin{cases} \frac{\delta_c - \delta}{T_R} & \text{if } \left| \frac{\delta_c - \delta}{T_R} \right| \leq \dot{\delta}_{\max} \\ \dot{\delta}_{\max} & \text{if } \frac{\delta_c - \delta}{T_R} > \dot{\delta}_{\max} \\ -\dot{\delta}_{\max} & \text{if } \frac{\delta_c - \delta}{T_R} < -\dot{\delta}_{\max} \end{cases} \quad (8)$$

In this study, the non-dimensional rudder time-constant T_R is taken as 0.1, and $\dot{\delta}_{\max}$ is chosen as 5° per second for the full-scale ship.

3.3. Rewards

The role of rewards in reinforcement learning is paramount for guiding the agent's behaviour. In this study, a reward of +100 is bestowed upon the agent upon successful arrival at the destination; otherwise, it remains zero. Mathematically, the reward function is defined as:

$$\text{reward} = \begin{cases} +100 & \text{if the agent successfully reaches the destination,} \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

The design of the reward function aims to equip the agent with sparse signals. While such rewards are straightforward to implement, they

Table 1. Hyperparameters for DDPG.

Hyperparameter	Value
Actor Learning rate	0.0005
Actor Decay steps	40000
Actor Decay Rate	0.8
Critic Learning rate	0.003
Critic Decay steps	40000
Critic Decay Rate	0.7
Actor Hidden layers	(64,64)
Observation fc layer params	(32,32)
Action fc layer params	(16,16)
Joint fc layer params	(64,64)
Discount factor(γ)	0.95
Sample batch size	128
Replay buffer size	1000000
Maximum time steps	160
Time step interval Δt	0.3
PD controller episodes (PD eps)	2000
Total number of episodes	5000
Mean of noise	0
The standard deviation of noise	0.15
Number of noisy episodes	11000
Update frequency(time steps)	10
Target network update frequency	1
Target update rate (τ)	0.01
Random seed number	65220

provide scant information to the agent during the learning process. This lack of information may result in prolonged training times or convergence issues (Hare 2019).

Actions are selected based on maximising the expected reward, often referred to as the Q-value. The Q-value for a given state-action pair (s, a) is computed using the Bellman equation (10):

$$Q(s, a) = \mathbb{E} \left[r + \gamma \max_{a'} Q(s', a') \mid s, a \right], \quad (10)$$

where r_t represents the immediate reward obtained after taking action a in state s at time t , and γ is the discount factor determining the importance of future rewards. This equation encapsulates the agent's foresight, considering both the immediate and future rewards when selecting actions.

Consequently, the expected reward diminishes if the agent chooses a longer route instead of the optimal path. This underscores the imperative of acquiring efficient policies that yield higher expected rewards, thereby aligning with the goal of attaining the destination with maximal efficiency. By prioritising actions that lead to

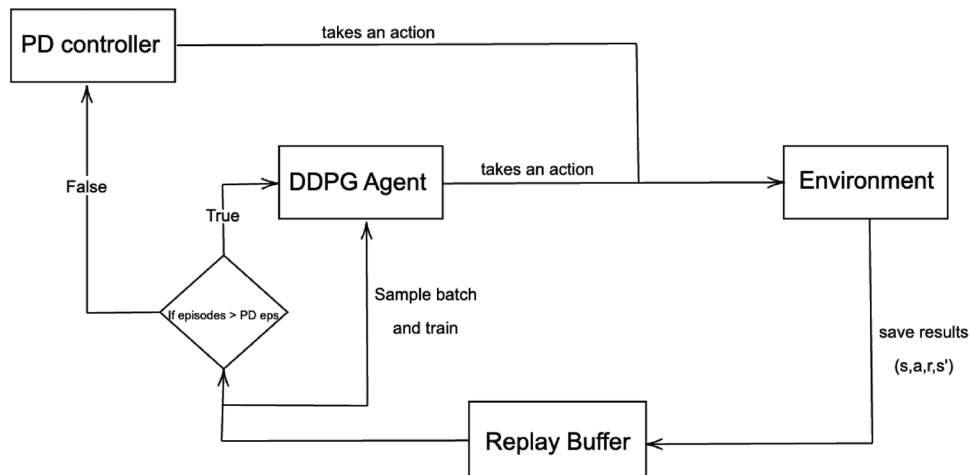


Figure 1. Training process of the RL agent.

the shortest duration to the destination, the agent implicitly learns to follow the track effectively, optimising its trajectory toward achieving the task objective. This reinforcement learning framework ensures that the agent continually refines its decision-making process to navigate the environment efficiently, even in the absence of explicit rewards or penalties associated with track adherence.

3.4. Training process

At the start of each episode, a random destination is chosen between $8L$ and $28L$. The agent starts from the origin, having an initial surge velocity of $1U$ with no sway velocity and no acceleration in any direction.

During the training procedure, the PD controller, equipped with the ILOS guidance system, generates the desired rudder angle denoted as δ_c , which the agents then utilise. The observational state (s), action (a), rewards (r), and future state (s') are subsequently recorded in a buffer for future reference as illustrated in Figure 1. After a specific number of training episodes, the RL agents determine the commanded rudder angle, which becomes the input for their training. To facilitate effective learning, the replay buffer is allocated

ample storage capacity to retain the trajectory of the PD controller until the training process concludes.

It's worth noting that, aside from receiving a destination reward of $+100$, the agent does not receive any other rewards. These rewards do not influence the trajectory between the starting point and the destination reward, as seen in previous works such as Chun et al. (2021), Zhou et al. (2019), Deraj et al. (2023), and others. Given that the replay buffer contains stored trajectories from the PD controller, the RL agent learns from these trajectories and explores previously stored ones.

3.5. Termination condition

These conditions prompt the agent to halt after a specific time duration or upon meeting particular criteria. In this investigation, we adopted the identical termination criteria as outlined in Deraj et al. (2023), which are as follows:

- When the number of timesteps reaches 160.
- When the agents successfully reach its destination.

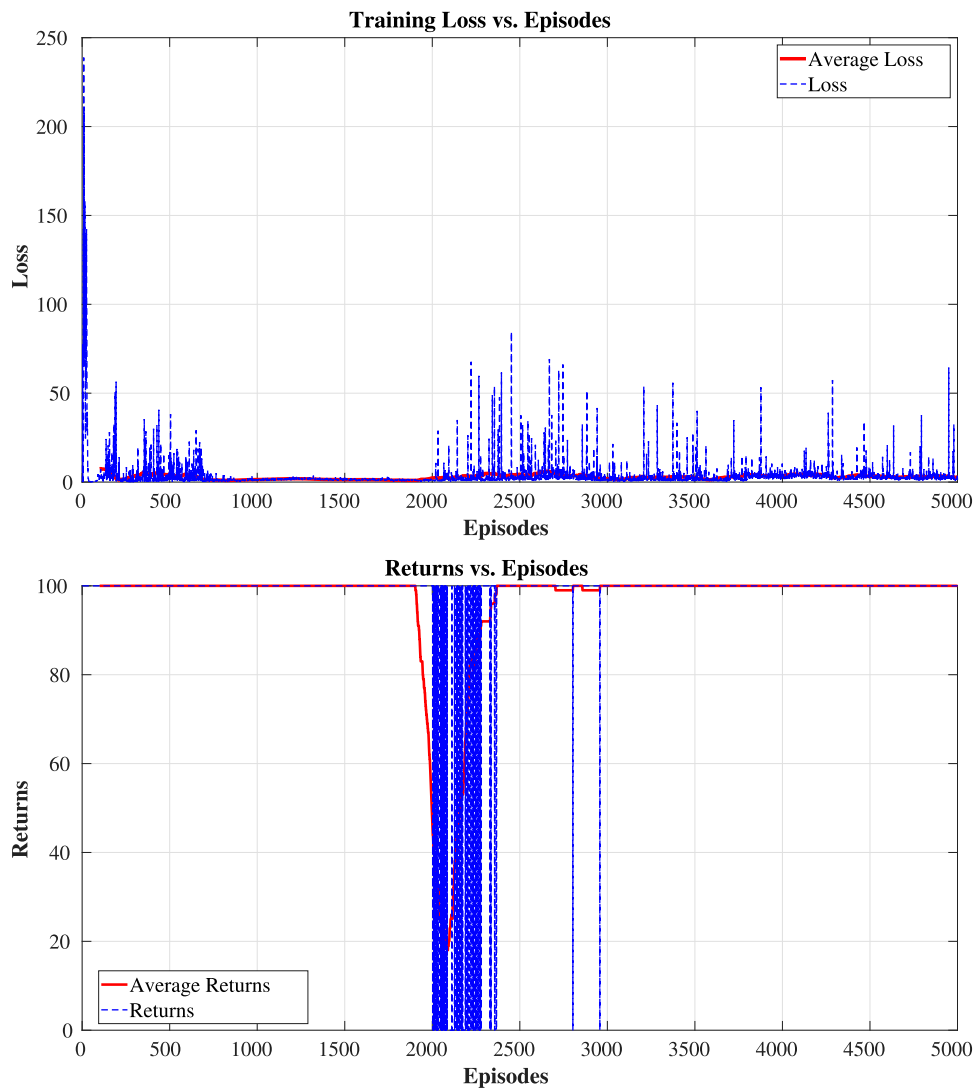


Figure 2. Training loss and returns for the model.

- When the agent crosses the destination, but the velocity vector does not align with the direction of the destination.

4. Results

4.1. Tools and libraries

The code implemented utilises the RL framework offered by the TensorFlow library called *TensorFlow Agents* (Hafner et al. 2017). This framework was specifically developed to establish consistent benchmarks for RL research. It offers a user-friendly platform for building RL environments, enabling the deployment and training of custom RL agents with minimal complexity. It is also important to note that this study used TensorFlow version 2.9.1, which allows GPU operations to be made deterministic. This means that the results produced in the study can be reproduced on any computer by training the agent with the same random seed value.

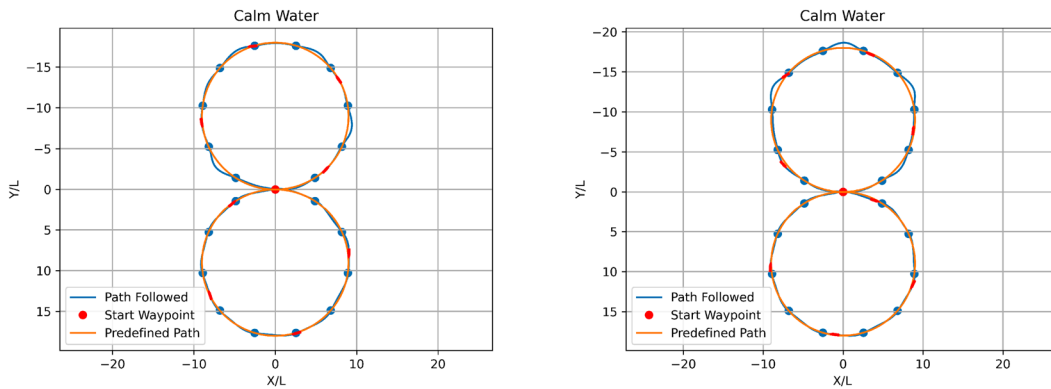
4.2. Hyperparameters

The agent undergoes calibration to ensure satisfactory waypoint tracking by tuning the hyperparameters of the Deep Deterministic Policy Gradient (DDPG) algorithm. This calibration process involves fine-tuning parameters such as the learning rate, exploration noise, and neural network architecture to optimise the agent's performance in navigating the environment. Specifically, the learning rate is chosen as an exponentially decaying function to facilitate efficient convergence during training.

The hyperparameters used for the DDPG model are detailed in Table 1, while the functions governing these hyperparameters can be referenced in TensorFlow (2023). These parameters play a crucial role in shaping the agent's learning dynamics and influence its ability to adapt to different environmental conditions.

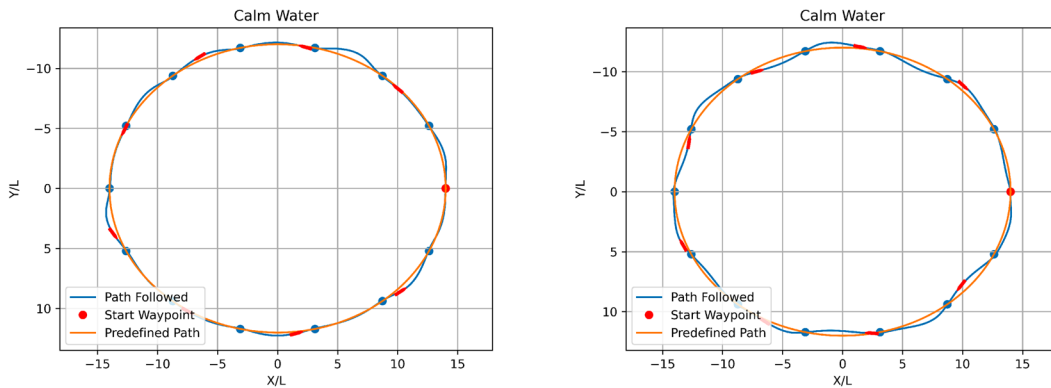
To evaluate the agent's performance, various maneuvers for waypoint tracking are simulated, allowing for a comprehensive assessment of its navigation capabilities. The training progress is monitored through visualisations of training losses and episode returns, as illustrated in Figure 2. These plots provide insights into the agent's learning trajectory and its ability to achieve the task objective over successive episodes.

An interesting observation emerges from analysing the episode returns: when the PD controller is responsible for providing the commanded rudder angle, the agent consistently reaches the destination within each episode. However, upon switching to the RL controller to generate the commanded rudder angle, there is an initial learning phase during which the agent struggles to reach the destination. This initial difficulty can be attributed to the exploration-exploitation dilemma inherent in reinforcement learning, where the agent must balance between exploring new strategies and exploiting known successful actions. During this phase, the agent may initially choose suboptimal actions, leading to deviations from the desired trajectory. Nevertheless, with continued training, the agent gradually learns to navigate towards the destination more effectively, eventually achieving a consistent reward of +100 in every episode.



(a) Eight Maneuver (bottom first) ($d_{RMSE} = 0.3549L$)

(b) Eight Maneuver (top first) ($d_{RMSE} = 0.3817L$)



(c) Ellipse Port turn ($d_{RMSE} = 0.3975L$)

(d) Ellipse Starboard turn ($d_{RMSE} = 0.3602L$)

Figure 3. Simple maneuver in Calm Water. (a) Eight Maneuver (bottom first) ($d_{RMSE} = 0.3549L$), (b) Eight Maneuver (top first) ($d_{RMSE} = 0.3817L$), (c) Ellipse Port turn ($d_{RMSE} = 0.3975L$) and (d) Ellipse Starboard turn ($d_{RMSE} = 0.3602L$).

This transition highlights the agent's capacity to adapt its control strategy based on environmental feedback and underscores the effectiveness of reinforcement learning in enabling autonomous decision-making in dynamic environments. Overall, these findings demonstrate the agent's capability to learn and refine its navigation skills, ultimately achieving the desired task objective of waypoint tracking.

The code associated with this training has been made available through a [Github repository](#)

4.3. Calm water

Upon completing successful agent training, the agent underwent testing to traverse paths discretised into an optimal number of waypoints, assuming that this selection accurately represents the path. Notably, information concerning subsequent waypoints is relayed to the neural network only upon the agent reaching the preceding waypoint, as outlined in the observation specifications (see Section 3.1), thereby finalising the trajectory. The agent underwent testing under different maneuvering conditions, as illustrated in Figures 3 and 4. The depicted trajectories include 'eight' and 'ellipse' maneuvers in depicted in Figure 3. It is evident from the Figure 3 that the agent successfully tracked the designated waypoints. To further check the effectiveness of the controller, the agent was given 'cardioid', 'sine', and 'astroid' maneuvers which are illustrated in Figure 4. From the figures, it becomes quite evident that the agent was successfully able to maneuver even the complex shapes.

The root mean square cross-track error for a given trajectory is mathematically defined as:

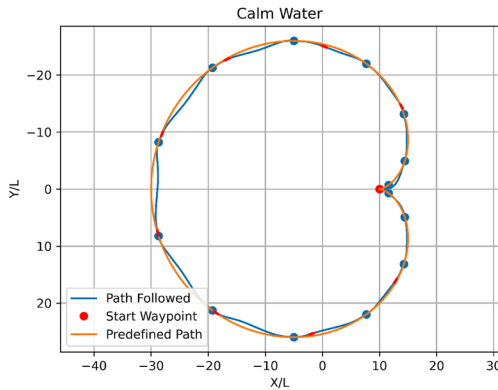
$$d_{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^N d_c^2(n\Delta t)} \quad (11)$$

Here, N represents the number of time steps within the trajectory, and $d_c(n\Delta t)$ signifies the cross-track error's value at the n^{th} time step. The figures demonstrate different turning directions, one clockwise and the other counterclockwise, which result from the inherent flow asymmetry affecting the rudder.

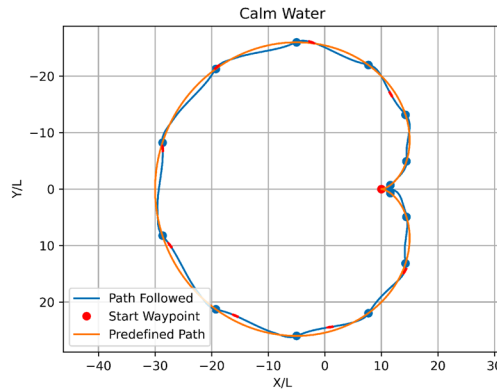
4.4. Presence of winds

The agent undergoes further testing under the influence of external disturbances, specifically in the form of wind. The wind forces are integrated into Equation (1) on the right-hand side. These forces and moments caused by the wind are modelled and added to Equation (1). The wind's non-dimensional velocity is V_w , and its direction is represented by β_w . The wind direction is defined as the angle between the wind's direction and the positive X-axis of the Global Coordinate System (GCS). The components of non-dimensional wind velocity relative to the Body Coordinate System (BCS) of the ship are defined as follows in Equation (12):

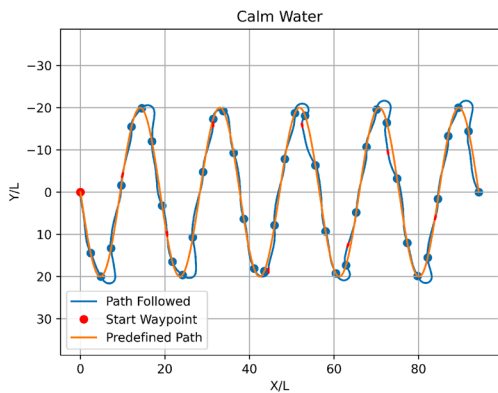
$$\begin{aligned} u_w &= V_w \cos(\beta_w - \psi) \\ v_w &= V_w \sin(\beta_w - \psi) \end{aligned} \quad (12)$$



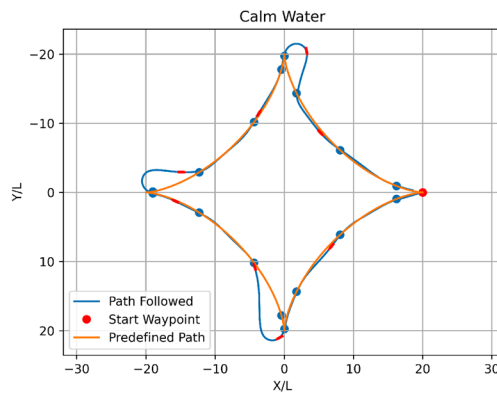
(a) Cardioid Port turn ($d_{RMSE} = 0.4287L$)



(b) Cardioid Starboard turn ($d_{RMSE} = 0.5053L$)



(c) Sine maneuver ($d_{RMSE} = 1.0294L$)



(d) Astroid maneuver ($d_{RMSE} = 1.3002L$)

Figure 4. Complex maneuver in Calm Water. (a) Cardioid Port turn ($d_{RMSE} = 0.4287L$), (b) Cardioid Starboard turn ($d_{RMSE} = 0.5053L$), (c) Sine maneuver ($d_{RMSE} = 1.0294L$) and (d) Astroid maneuver ($d_{RMSE} = 1.3002L$).

Table 2. Wind force parameters.

Parameter	Value
A_x	0.1064
A_y	0.7601
C_{wx}	$\cos(\gamma_w)$
C_{wy}	$\sin(\gamma_w)$
$C_{w\psi}$	$0.5 \sin(\gamma_w)$
ρ_a	1.225 kg/m^3
ρ	1025 kg/m^3

The non-dimensional ship velocity components relative to the wind can be expressed as $u_{rw} = u - u_w$ and $v_{rw} = v - v_w$. The magnitude of the non-dimensional relative wind velocity U_{wr} and the relative wind angle γ_w with respect to the vessel are determined by Equation (13):

$$U_{wr} = \sqrt{u_{rw}^2 + v_{rw}^2} \quad (13)$$

$$\gamma_w = \arctan 2(-v_{rw}, -u_{rw})$$

Subsequently, the non-dimensional wind force components W_x , W_y , and the non-dimensional wind yaw moment W_ψ are computed as

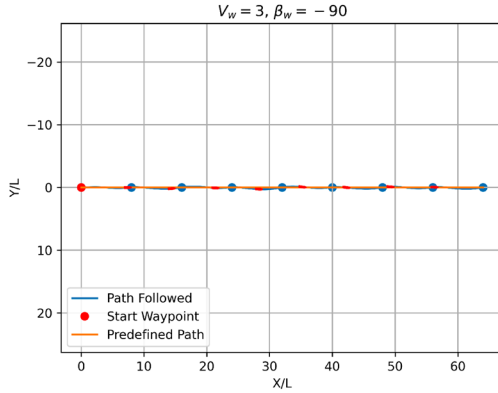
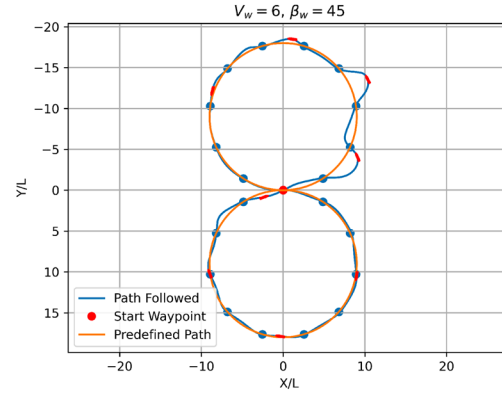
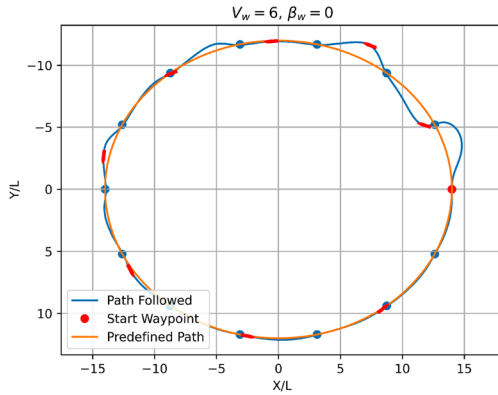
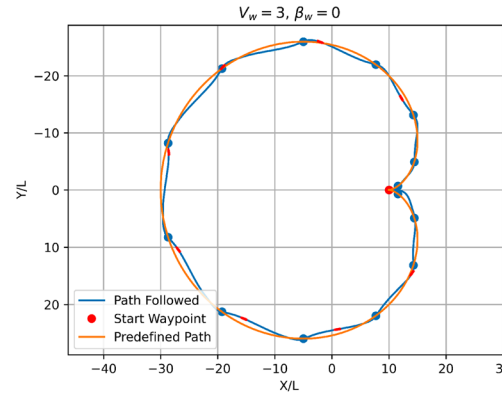
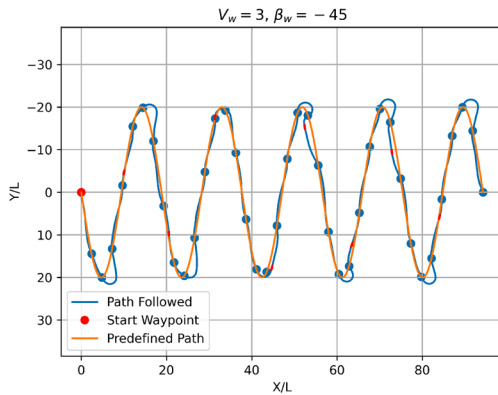
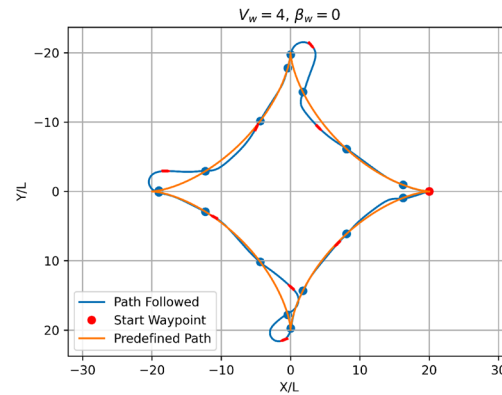
(a) Straight Line ($d_{RMSE} = 0.1154L$)(b) Eight Maneuver (bottom first) ($d_{RMSE} = 0.7512L$)(c) Ellipse Port Turn ($d_{RMSE} = 0.6123L$)(d) Cardioid Maneuver ($d_{RMSE} = 0.4906L$)(e) Sine Maneuver ($d_{RMSE} = 1.0542L$)(f) Astroid Maneuver ($d_{RMSE} = 1.2831L$)

Figure 5. Different maneuver in the presence of wind. (a) Straight Line ($d_{RMSE} = 0.1154L$), (b) Eight Maneuver (bottom first) ($d_{RMSE} = 0.7512L$), (c) Ellipse Port Turn ($d_{RMSE} = 0.6123L$), (d) Cardioid Maneuver ($d_{RMSE} = 0.4906L$), (e) Sine Maneuver ($d_{RMSE} = 1.0542L$) and (f) Astroid Maneuver ($d_{RMSE} = 1.2831L$).

Table 3. Wind force cases.

Case	Path	v_w	β_w	d_{RMSE}
1	Line	3	$-\pi/2$	0.1254L
2	Eight	6	$\pi/4$	0.6196L
3	Ellipse	6	0	0.6123L
4	Cardioid	3	0	0.4906L
5	Sine	3	$-\pi/4$	1.0542L
6	Astroid	3	0	1.2831L

depicted in Equation (14):

$$\begin{aligned}
 W_x &= C_{wx}(\gamma_w) \frac{\rho_a}{\rho} A_x U_{wr}^2 \\
 W_y &= C_{wy}(\gamma_w) \frac{\rho_a}{\rho} A_y U_{wr}^2 \\
 W_{\psi} &= C_{w\psi}(\gamma_w) \frac{\rho_a}{\rho} A_y L_{OA} U_{wr}^2
 \end{aligned} \quad (14)$$

Here, A_x and A_y represent the hull's non-dimensional lateral and longitudinal projected areas above water in the yz and xz planes in the BCS, respectively. Note that the non-dimensional factor for the projected areas A_x and A_y is taken as Ld_{em} . The non-dimensional wind coefficients C_{wx} , C_{wy} , and $C_{w\psi}$ are assumed to be functions of the

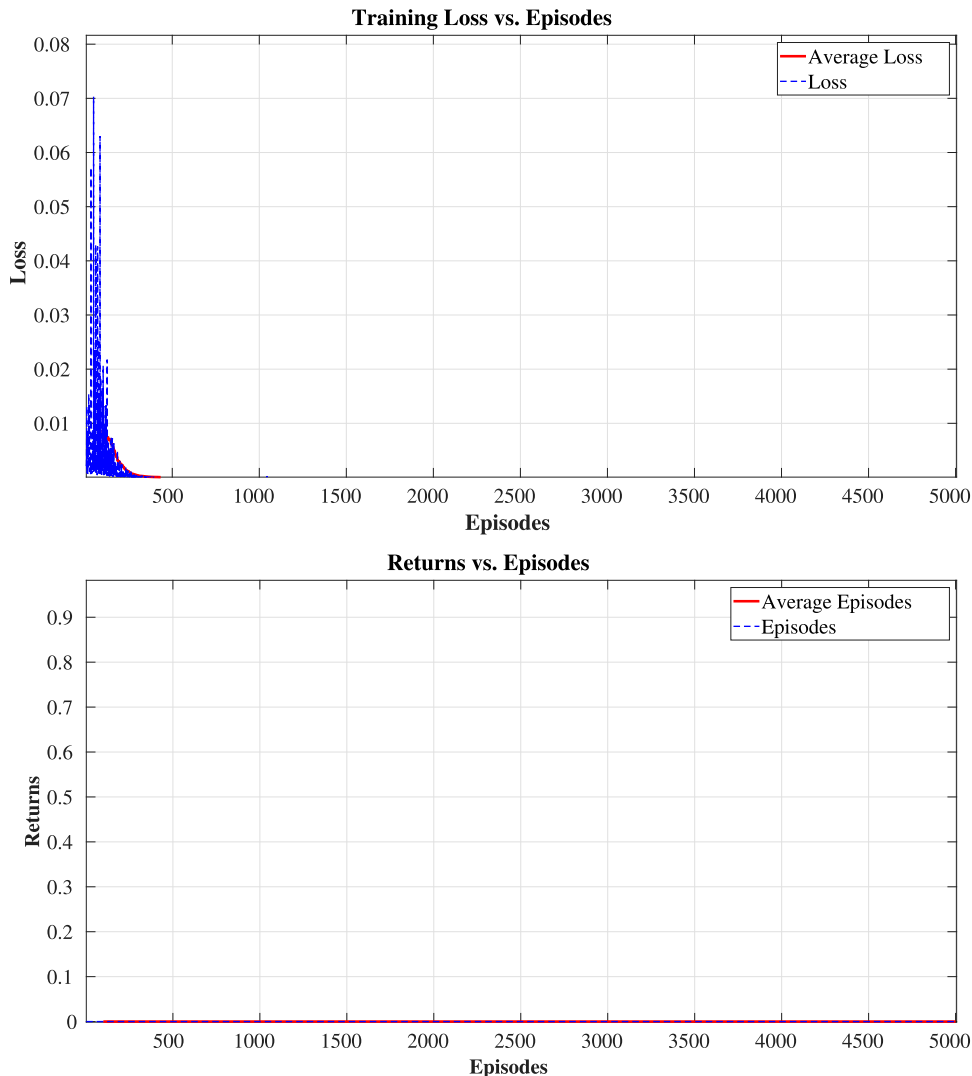
relative wind direction γ_w . ρ_a represents the air density, and ρ is the water density. L_{OA} stands for the non-dimensional overall length of the vessel, normalised concerning the length between perpendiculars L . Table 2 presents the wind parameter values considered in this study.

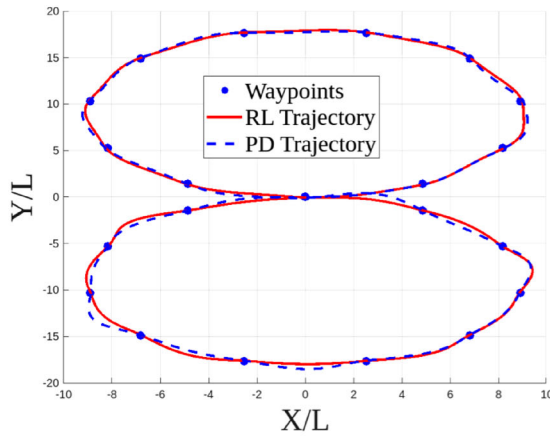
As illustrated in Figure 5, the ship effectively followed waypoints even when exposed to strong winds. The wind speed and direction are detailed in Table 3.

5. Discussion

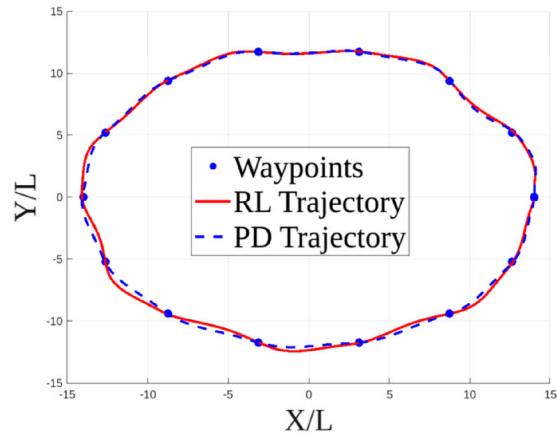
An agent was trained using hyperparameters identical to those used in generating the results showcased in Figures 3 and 4, as outlined in detail in Table 1, and utilising the same random seed number. The only deviation in this instance was the absence of the PD controller with an ILOS guidance system when populating the replay buffer. Unlike the previous setup, where both the trajectory of the RL agent and the guidance information from the PD controller were stored in the replay buffer, only the trajectory of the RL agent was stored in this modified configuration.

The outcomes of this modified configuration are depicted in Figure 6, which illustrates the returns and loss plot. Evidently, the agent failed to undergo any meaningful learning and was unable to reach the destination throughout the training phase, as indicated by

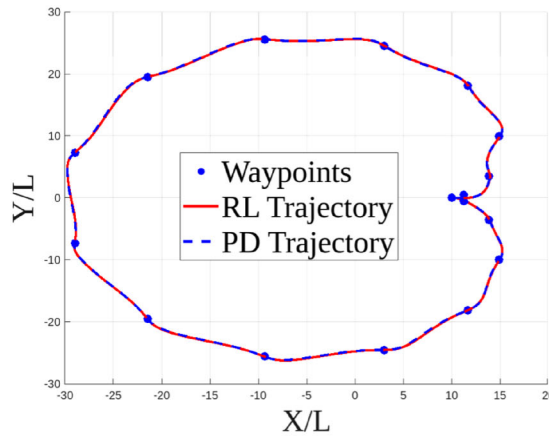

Figure 6. Training loss and returns for the model trained without PD controller.



(a) Eight Trajectory (RL: $d_{RMSE} = 0.3549L$; PD: $d_{RMSE} = 0.4189L$)



(b) Ellipse Trajectory (RL: $d_{RMSE} = 0.3602L$; PD: $d_{RMSE} = 0.3723L$)



(c) Cardioid Trajectory (RL: $d_{RMSE} = 0.4910L$; PD: $d_{RMSE} = 0.4927L$)

Figure 7. Comparison of the controller in different maneuver. (a) Eight Trajectory (RL: $d_{RMSE} = 0.3549L$; PD: $d_{RMSE} = 0.4189L$), (b) Ellipse Trajectory (RL: $d_{RMSE} = 0.3602L$; PD: $d_{RMSE} = 0.3723L$) and (c) Cardioid Trajectory (RL: $d_{RMSE} = 0.4910L$; PD: $d_{RMSE} = 0.4927L$).

the returns plot. Examination of the plots reveals a consistent value of zero for all instances, signifying that the model never acquired the ability to navigate successfully to the destination. This indicates a significant deficiency in exploration, despite using the same reward function and training scenarios as before. The loss plot further confirms that the agent did not learn anything over the entire episode duration, as the loss quickly approached zero. It's worth noting that the difference in the two networks in the DDPG motivates the DDPG agent to learn the optimal behaviour, but since the reward was consistently zero, the agent failed to learn anything.

This stands in stark contrast to the earlier scenario, highlighting the critical role played by the PD controller with an ILOS guidance system in facilitating effective learning and goal achievement during the training process. The absence of the guidance system severely hindered the agent's ability to explore and learn, underscoring the importance of integrating guidance mechanisms in reinforcement learning setups to ensure successful training outcomes.

5.1. Comparison with a PD based controller

In this section, we evaluate the path following capabilities of the Deep Deterministic Policy Optimisation (DDPG) agent in comparison

with a Proportional Derivative (PD) controller. The PD controller is employed to ensure the convergence of the vessel's heading. The proportional (K_p) and derivative (K_d) gains are set to the same values utilised during the training of the DDPG agent, as outlined in Section 3.4. These parameter values are specified in Table 1.

The Root Mean Square Error (RMSE) metric reveals that the RL agent outperformed the PD controller, indicating superior performance. The DRL agent's capacity for both exploration and exploitation facilitated its enhanced performance compared to the conventional PD controller. This is evident from Figure 7(a), where the trajectory's RMSE of the RL agent was 18% lower than that of the PD controller. Additionally, as depicted in Figure 7(b,c), the RL agent consistently exhibited slightly superior performance compared to the PD controller. This underscores the efficacy of the DRL approach in navigating the vessel trajectory.

6. Conclusion and future studies

This investigation delves into the fine-tuning study that addresses the optimisation of training time for a Deep Reinforcement Learning (DRL) based controller, particularly in scenarios where a pre-existing controller with proficiency at waypoint tracking exists. The essence

of this study lies in leveraging the iterative trial-and-error nature of the DRL agent, which possesses the inherent capability to assimilate. Since the DRL agent learns through iterative trial and error processes, it has the potential to acquire a more unrefined trajectory than the currently available controllers.

The crux of this research extends beyond its current scope, projecting a future trajectory that involves the future and various other integrations of diverse controllers to augment the efficacy of the neural network training further. By introducing multiple controllers, each proficient in specific aspects of maritime navigation, the training process stands controllers could be employed to benefit from a more comprehensive and diverse set of experiences. This diversified training approach has the potential to equip the DRL agent with a broader range of skills and heightened adaptability to navigate through dynamic and complex environments.

Moreover, the study envisions an expansion of the training landscape by enhancing network training further. Additionally, the introduction of external environmental factors such as waves and currents. Integrating these real-world complexities into the training regimen can significantly enhance the DRL-based controller's robustness. Navigating maritime scenarios characterised by the unpredictability of waves and currents poses a distinct set of challenges, and incorporating these challenges into the training process ensures that the DRL agent is well-equipped to handle many real-world scenarios. Another noteworthy dimension explored in this study involves training the DRL agent to navigate with precision and a heightened awareness of collision avoidance. Instilling collision avoidance capabilities in the agent makes it adept at evading stationary and moving obstacles, contributing to safer and more reliable navigation. Additionally, it can be incorporated into the training process. The agent can also be trained to evade collisions with both stationary and moving obstacles and to adhere to COLREGs rules, which govern the conduct of vessels at sea, which can be instilled in the DRL agent through training. This regulatory compliance further solidifies the agent's ability to navigate by established maritime norms and standards.

Furthermore, the research suggests incorporating Automatic Identification System (AIS) data into the training process. The DRL agent can emulate more human-like decision-making processes by utilising AIS-based data. This infusion of real-world maritime data adds a layer of authenticity to the agent's learning experience, making its decision-making more contextually relevant and reflective of the intricacies of actual naval operations. Ultimately, this augmentation elevates the DRL-based controller's overall efficiency, aligning it more closely with human decision-making paradigms in maritime scenarios. Furthermore, training the agent using AIS-based data can facilitate more human-like decision-making, thus enhancing the controller's efficiency.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Data availability statement

The datasets generated during and analysed during the current study are available from the corresponding author upon reasonable request.

ORCID

Md Shadab Alam  <http://orcid.org/0000-0001-9184-9963>

References

Ahmed IH, Brewitt C, Carlucho I, Christianos F, Dunion M, Fosong E, Garcin S, Guo S, Gyevar B, McInroe T, et al. 2022. Deep reinforce-

- ment learning for multi-agent interaction. *AI Commun.* 35(4):357–368. doi: [10.3233/AIC-220116](https://doi.org/10.3233/AIC-220116).
- Alam MS. 2023. Data driven control for marine vehicle maneuvering [PhD thesis]. doi: [10.13140/RG.2.2.16194.15045](https://doi.org/10.13140/RG.2.2.16194.15045).
- Alam MS, Somayajula A. 2024. Importance of states and rewards in designing reinforcement learning environments for autonomous ships (under review).
- Alam MS, Sudha SKR, Somayajula A. 2023. AI on the water: applying drl to autonomous vessel navigation. *arXiv preprint arXiv:2310.14938*.
- Andres A, Schäfer L, Villar-Rodriguez E, Albrecht SV, Del Ser J. 2023. Using offline data to speed-up reinforcement learning in procedurally generated environments. *arXiv preprint arXiv:2304.09825*.
- Bain M, Sammut C. 1995. A framework for behavioural cloning. In: *Machine intelligence 15*. p. 103–129.
- Chen C, Chen X-Q, Ma F, Zeng X-J, Wang J. 2019. A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Eng.* 189:106299. doi: [10.1016/j.oceaneng.2019.106299](https://doi.org/10.1016/j.oceaneng.2019.106299).
- Chun D-H, Roh M-I, Lee H-W, Ha J, Yu D. 2021. Deep reinforcement learning-based collision avoidance for an autonomous ship. *Ocean Eng.* 234:109216. doi: [10.1016/j.oceaneng.2021.109216](https://doi.org/10.1016/j.oceaneng.2021.109216).
- Deogaonkar VV, Jadhav AK, Ramachandran K, Somayajula AS. 2023, June. Data driven identification of ship maneuvering coefficients. *International Conference on Offshore Mechanics and Arctic Engineering (Vol. 86878, p. V005T06A050)*. American Society of Mechanical Engineers.
- Deraj R, Kumar RS, Alam MS, Somayajula A. 2023. Deep reinforcement learning based controller for ship navigation. *Ocean Eng.* 273:113937. doi: [10.1016/j.oceaneng.2023.113937](https://doi.org/10.1016/j.oceaneng.2023.113937).
- Hafner D, Davidson J, Vanhoucke V. 2017. Tensorflow agents: efficient batched reinforcement learning in tensorflow. *arXiv preprint arXiv:1709.02878*.
- Hare J. 2019. Dealing with sparse rewards in reinforcement learning. *arXiv preprint arXiv:1910.09281*.
- Jose J, Alam MS, Somayajula AS. 2023. Navigating the ocean with drl: Path following for marine vessels. *arXiv preprint arXiv:2310.14932*.
- Ladosz P, Weng L, Kim M, Oh H. 2022. Exploration in deep reinforcement learning: a survey. *Inform Fusion.* 85:1–22. doi: [10.1016/j.inffus.2022.03.003](https://doi.org/10.1016/j.inffus.2022.03.003).
- Lambert NO, Drew DS, Yaconelli J, Levine S, Calandra R, Pister KS. 2019. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robot Autom Lett.* 4(4):4224–4230. doi: [10.1109/LSP.2016](https://doi.org/10.1109/LSP.2016).
- Lee D, Lee SJ, Yim SC. 2020. Reinforcement learning-based adaptive PID controller for DPS. *Ocean Eng.* 216:108053. doi: [10.1016/j.oceaneng.2020.108053](https://doi.org/10.1016/j.oceaneng.2020.108053).
- Lekkas AM, Fossen TI. 2012. A time-varying lookahead distance guidance law for path following. *IFAC Proc Vol.* 45(27):398–403. doi: [10.3182/20120919-3-IT-2046.00068](https://doi.org/10.3182/20120919-3-IT-2046.00068).
- Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Martinsen AB, Lekkas AM. 2018. Curved path following with deep reinforcement learning: results from three vessel models. In: *OCEANS 2018 MTS/IEEE Charleston*; IEEE. p. 1–8.
- Meyer E, Heiberg A, Rasheed A, San O. 2020. Colreg-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning. *IEEE Access.* 8:165344–165364. doi: [10.1109/Access.6287639](https://doi.org/10.1109/Access.6287639).
- Mohan A, Somayajula AS. 2023. Analyzing robustness and accuracy of different controllers for underactuated ships. In: *2023 4th International Conference for Emerging Technology (INCET)*; IEEE. p. 1–6.
- Morales EF, Sammut C. 2004. Learning to fly by combining reinforcement learning with behavioural cloning. *Proceedings of the Twenty-First International Conference on Machine Learning*. p. 76.
- Moreira L, Fossen TI, Soares CG. 2007. Path following control system for a tanker ship model. *Ocean Eng.* 34(14–15):2074–2085. doi: [10.1016/j.oceaneng.2007.02.005](https://doi.org/10.1016/j.oceaneng.2007.02.005).
- Raffin A, Hill A, Ernestus M, Gleave A, Kanervisto A, Dormann N. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Technol.* 22(268):1–8.
- Shen H, Hashimoto H, Matsuda A, Taniguchi Y, Terada D, Guo C. 2019. Automatic collision avoidance of multiple ships based on deep q-learning. *Appl Ocean Res.* 86:268–288. doi: [10.1016/j.apor.2019.02.020](https://doi.org/10.1016/j.apor.2019.02.020).
- Sivaraj S, Dubey A, Rajendran S. 2023. On the performance of different deep reinforcement learning based controllers for the path-following of a ship. *Ocean Eng.* 286:115607. doi: [10.1016/j.oceaneng.2023.115607](https://doi.org/10.1016/j.oceaneng.2023.115607).
- Sivaraj S, Rajendran S, Prasad LP. 2022. Data driven control based on deep q-network algorithm for heading control and path following of a ship in calm water and waves. *Ocean Eng.* 259:111802. doi: [10.1016/j.oceaneng.2022.111802](https://doi.org/10.1016/j.oceaneng.2022.111802).
- Sudha SKR, Alam MS, Reddy B, Somayajula AS. 2023. Comparison of path following in ships using modern and traditional controllers. *arXiv preprint arXiv:2310.14940*.
- Sutton RS, Barto AG. 2018. *Reinforcement learning: an introduction*. MIT Press.
- TensorFlow. 2023. *Tensorflow agent ddpq*. [accessed 2023 Sep 17].

- Vijay A, Somayajula A. 2022. Identification of hydrodynamic coefficients using support vector regression. In: OCEANS 2022-Chennai; IEEE. p. 1–7.
- Wang C, Zhang X, Li R, Dong P. 2018. Path planning of maritime autonomous surface ships in unknown environment with reinforcement learning. In: International Conference on Cognitive Systems and Signal Processing; Springer. p. 127–137.
- Wang X, Zhang Z, Zhang W. 2022. Model-based multi-agent reinforcement learning: recent progress and prospects. arXiv preprint arXiv:2203.10603.
- Woo J, Yu C, Kim N. 2019. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean Eng.* 183:155–166. doi: [10.1016/j.oceaneng.2019.04.099](https://doi.org/10.1016/j.oceaneng.2019.04.099).
- Yasukawa H, Yoshimura Y. 2015. Introduction of MMG standard method for ship maneuvering predictions. *J Mar Sci Technol.* 20(1):37–52. doi: [10.1007/s00773-014-0293-y](https://doi.org/10.1007/s00773-014-0293-y).
- Yoshimura Y, Masumoto Y. 2012. Hydrodynamic force database with medium high speed merchant ships including fishing vessels and investigation into a manoeuvring prediction method. *J Jpn Soc Nav Archit Ocean Eng.* 14:63–73.
- Zhao L, Roh M-I, Lee S-J. 2019. Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. *J Mar Sci Technol.* 27(4):1.
- Zhou X, Wu P, Zhang H, Guo W, Liu Y. 2019. Learn to navigate: cooperative path planning for unmanned surface vehicles using deep reinforcement learning. *IEEE Access.* 7:165262–165278. doi: [10.1109/Access.6287639](https://doi.org/10.1109/Access.6287639).
- Zinage S, Somayajula A. 2020. A comparative study of different active heave compensation approaches. *Ocean Syst Eng.* 10(4):373.
- Zinage S, Somayajula A. 2021. Deep reinforcement learning based controller for active heave compensation. *IFAC-PapersOnLine.* 54(16):161–167. doi: [10.1016/j.ifacol.2021.10.088](https://doi.org/10.1016/j.ifacol.2021.10.088).