Ocean Engineering 273 (2023) 113937

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Deep reinforcement learning based controller for ship navigation

Rohit Deraj [a,b], R.S. Sanjeev Kumar [a,b], Md Shadab Alam [b], Abhilash Somayajula [b,*]

[a] Department of Marine Technology, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
[b] Department of Ocean Engineering, Indian Institute of Technology (IIT) Madras, Chennai, India

ARTICLE INFO

Keywords:
Reinforcement learning
Autonomous vessel
Ship maneuvering
Path-following
Deep Q-network
MMG model
Deep learning

ABSTRACT

A majority of marine accidents that occur can be attributed to errors in human decisions. Through automation, the occurrence of such incidents can be minimized. Therefore, automation in the marine industry has been receiving increased attention in the recent years. This paper investigates the automation of the path following action of a ship. A deep Q-learning approach is proposed to solve the path-following problem of a ship. This method comes under the broader area of deep reinforcement learning (DRL) and is well suited for such tasks, as it can learn to take optimal decisions through sufficient experience. This algorithm also balances the exploration and the exploitation schemes of an agent operating in an environment. A three-degree-of-freedom (3-DOF) dynamic model is adopted to describe the ship's motion. The Krisco container ship (KCS) is chosen for this study as it is a benchmark hull that is used in several studies and its hydrodynamic coefficients are readily available for numerical modeling. Numerical simulations for the turning circle and zig-zag maneuver tests are performed to verify the accuracy of the proposed dynamic model. A reinforcement learning (RL) agent is trained to interact with this numerical model to achieve waypoint tracking. Finally, the proposed approach is investigated not only by numerical simulations but also by model experiments using 1:75.5 scaled model.

1. Introduction

Human error accounts for roughly 80%–85% of all marine-related accidents (Baker and McCafferty, 2005). These incidents put human lives in danger and also have a potential to cause damage to the environment. In addition, the owners of the ships involved in such incidents also face significant financial losses. The recent Ever Given incident in the Suez Canal, 2021 is just one such example, where the vessel could not maintain its path under the influence of strong winds. With the recent advancements in the artificial intelligence (AI) it is now possible to explore automation solutions for the maritime industry to reduce the occurrence of such incidents. The progress in AI technology, particularly reinforcement learning (RL), now offers a new solution to address the demands of ship path following and trajectory tracking.

In RL, agents are trained on a reward and penalty system. The agent is rewarded for actions that allow the attainment of a goal and penalized for actions that have detrimental effects. Through experience, the agent seeks the optimal policy that consistently chooses actions leading to higher rewards and avoids actions that lead to lower rewards. Such a control approach falls under the class of data driven controllers where no model of the system being controlled is needed. Rather the system dynamics and the appropriate control strategy is learned by the agent as it tries to optimize its interaction with the system.

Traditional RL requires the storing of the action choosing policy of the agent in the form of Q-tables. Q-tables document the expected reward for a large table of scenarios covering all possible combinations of discrete states that the agent can be in and the actions it can choose. This table is updated at every time step when the agent moves to a new state through a chosen action from previous state. The advancement in the area of deep learning has given rise to deep reinforcement learning (DRL) that incorporate neural networks to store the policy of the agent (Mnih et al., 2013). This replacement of Q-tables by neural networks allows for a more efficient storage of the policy and potential application to more complex problems. While the traditional RL based on Q-tables requires both state space and action space to be discrete, with deep reinforcement learning it is now possible to explore solutions for scenarios where the state space and action space may be discrete or continuous (Perera et al., 2015).

Traditional autopilots use line of sight (LOS) guidance systems in conjunction with proportional–integral–derivative (PID) controllers to achieve waypoint tracking for path following (Lekkas and Fossen, 2012; Moreira et al., 2007). Over this layer of controller is a typical path

* Corresponding author.
E-mail addresses: rohitdr@stud.ntnu.no (R. Deraj), sanjeev.k.r.sudha@ntnu.no (R.S.S. Kumar), oe21s007@smail.iitm.ac.in (M.S. Alam), abhilash@iitm.ac.in (A. Somayajula).
URL: https://www.doe.iitm.ac.in/abhilash (A. Somayajula).

https://doi.org/10.1016/j.oceaneng.2023.113937
Received 6 December 2022; Received in revised form 24 January 2023; Accepted 10 February 2023
Available online 27 February 2023

Nomenclature

α_R	Effective inflow angle to rudder
β	Drift angle
β_w	Direction of wind with respect to X-axis of GCS
χ_e	Course angle error
δ	Actual rudder angle
δ_c	Commanded rudder angle
$\dot{e}$	Rate of change of heading error
$\dot{\delta}$	Rudder rate
$\dot{\delta}_{max}$	Maximum rudder rate
$\dot{\psi}_d$	Rate of change of desired heading angle
ϵ	Probability with which the RL agent takes a random action
η	Ratio of propeller diameter to the rudder height
γ_R	Flow straightening factor of hull
γ_w	Relative wind angle with respect to ship
κ	An experimental constant for expressing u_R
π	Policy of Markov Decision Process
ψ	Current heading angle
$\psi_{desired}$	Desired heading angle
ρ	Density of water
ρ_a	Density of air
θ	Weight and biases of neural network
ε	Ratio of wake fraction at propeller and rudder positions
a	Action of Markov Decision Process
a_0	Constant curve fitting parameter for determining propeller thrust coefficient
a_1	Linear curve fitting parameter for determining propeller thrust coefficient
a_2	Quadratic curve fitting parameter for determining propeller thrust coefficient
a_H	Rudder force increase factor
A_R	Rudder Area
A_x	Lateral projected areas of the hull
A_y	Longitudinal projected areas of the hull
B	Beam of the vessel
$C_{w\psi}$	Yaw wind coefficient
C_{wx}	Surge wind coefficient
C_{wy}	Sway wind coefficient
d	Depth moulded
D_p	Propeller diameter
d_c	Cross track error
d_{em}	Draft
d_{wp}	Distance to destination waypoint
e	Heading error
f_α	Rudder lift gradient coefficient
J	Advance coefficient
K_d	Derivative gain
K_p	Proportional gain
K_T	Propeller thrust coefficient
L	Length between perpendiculars
L_{OA}	Length overall
l_R	Correction of flow straightening factor to yaw rate
N	External yaw moment
n	Propeller revolution rate
N_H	Hull hydrodynamic yaw moment
N_R	Yaw moment due to rudder
R	Cumulative sum of reward at end of episode
r	Yaw rate
r_1, r_2, r_3	Rewards associated with cross-track error, course angle error and distance to goal
s	Observation state of Markov Decision Process
t	Thrust deduction factor
T_R	Rudder time constant
t_R	Steering resistance deduction factor
U	Design speed
u	Surge velocity
u_{rw}, v_{rw}	Velocity components of the ship relative to the wind
U_R	Rudder inflow velocity
u_R	Longitudinal inflow velocity component to rudder
U_{wr}	Relative wind velocity
v	Sway velocity
v_R	Lateral inflow velocity component to rudder
V_w	Velocity of wind
w	Effective wake fraction
W_ϕ	Yaw wind moment
W_x	Surge wind force
W_y	Sway wind force
X	External surge force
x, y	Coordinate of current position of ship in GCS
x_G	Longitudinal center of gravity of ship (LCG)
x_i, y_i	Coordinate of initial position of ship in GCS
x_g, y_g	Coordinate of goal/next waypoint in GCS
X_H	Hull hydrodynamic surge force
x_H	Longitudinal coordinate of acting point of the additional lateral force component induced by steering
X_P	Surge force due to propeller
X_R	Surge force due to rudder
x_R	Location of the rudder with respect to midship
Y	External sway force
Y_H	Hull hydrodynamic sway force
Y_R	Sway force due to rudder

planning algorithm that dictates the waypoints to be followed and also specifies changes in path in the presence of static and dynamic obstacles. Dynamic path planning is still a significant challenge in terms of robustness as well as with respect to computational power needed to perform it in real time. With the advent of AI based control strategies, these two functions of control and path planning can be achieved with a single controller that does not need to perform significant computations in real time. This has already been shown to be promising for certain applications like active heave compensation (Zinage and Somayajula, 2021).

Several studies have started investigating the application of RL methods for path planning. Wang et al. (2018) used the Q-learning algorithm and demonstrated that the trained agent is successfully able to plan the path and reached the destination while avoiding the static obstacles. However, the study did not consider the dynamics of the

vessel and was limited to planning a path that avoided static obstacles. Shen et al. (2019) used deep Q learning algorithm for collision avoidance of multiple ships and demonstrated that three ships in simulations effectively avoid each other in a restricted space. The dynamics of ships in simulations was taken to be governed by the Nomoto's first order model. Later these results were validated through experiments in a rectangular basin. The study showed that the vessels were able to effectively avoid each other in a restricted basin, but it was also seen that the waypoints could not be effectively tracked. Ability to track waypoints without the presence of obstacles was not investigated in the study. Sivaraj et al. (2022) developed a Deep Q-network (DQN) for path following and heading control of ship in calm water and waves for a KVLCC2 tanker. This study relied on training different agents for tracking different headings. Chen et al. (2019) developed a Q-learning based model using Nomoto's first order equation to implement a practical path following algorithm taking into account the high inertia of under-actuated cargo ships. The study showed that the Q-learning based model outperforms RRTs (Rapidly-exploring random tree) and A* with shorter path length and smoother turns. Woo et al. (2019) trained a DDPG (Deep Deterministic Policy Gradient) based steering controller that was used in conjunction with a vector field guidance law to achieve path following. The policy at different levels of training were compared in simulations and also through experiments on an unmanned surface vehicle (WAM-V). However, the performance of the steering controller was not compared with any of the traditional control approaches. Martinsen and Lekkas (2018) trained a DDPG model for straight line path following and used transfer learning to follow curve paths for three different vessels. The effectiveness of the developed controllers is demonstrated through simulations. It was shown that DRL based guidance could accumulate better rewards than the traditional LOS based guidance. Zhou et al. (2019) used a deep Q-network (DQN) for path planning of a single USV and USV formation. Being focused on path planning alone, the study focused on kinematics of the vessels rather than a dynamic model. It was shown through simulations that the developed method could effectively avoid collision with static obstacles while maintaining a formation between three USVs.

Subsequently, several researches have also tried to combine path planning along with COLREGs (Convention on the International Regulations for Preventing Collisions at Sea) compliant behavior for autonomous ships. Guo et al. (2020) implemented a DDPG based path planning agent to choose a continuous actions of rudder angle and speed, for autonomous path planning compliant with COLREGs. However, the details of the dynamics of the vessel used for training were not provided. Artificial potential field (APF) in conjunction with DDPG method is explored and the study reported that this method converged faster and required fewer training episodes and lesser time. Shen and Guo (2016) used an actor–critic algorithm for path-following. The reward functions in the study are based on the error calculated with respect to a reference course. Later they also extended their method to use limiting lines and navigational polygons to prevent ship collisions (Shen et al., 2019). Layek et al. (2017) compared DDPG and NAF (Normalized Advantage Function) based models, both outperforming random search based control when the ship tries to pass through a specified gate, with random initial orientations. The study focuses on the kinematics of the vessel and ignores the dynamics in the simulation. Zhao et al. (2019) developed a PPO (Proximal Policy Optimisation) algorithm to navigate a ship simulated by the 3-DOF dynamic model using LOS guidance system. The study also compared the DRL controller against a traditional PID controller and concluded that the RL controller resulted in a smaller cross track error. Heiberg et al. (2022) also developed a PPO algorithm for path following and obstacle avoidance using collision risk theory. Zhao and Roh (2019) also proposed multi-ship collision avoidance based on DRL by categorizing the target ships based on the region in which they are present with reference to the autonomous ship. It was demonstrated that the ship can follow some of the COLREG rules.

Table 1
KCS ship parameters.

Ship parameter	Value
Length between perpendiculars (L)	230 m
Length overall (L_{OA})	232.5 m
Depth moulded (d)	19 m
Beam (B)	32.2 m
Draft (d_{em})	10.8 m
Displacement	53330.75 tons
LCG (x_G)	−3.408 m
Radius of gyration	57.5 m
Design speed (U)	12.347 m/s

Most of the studies have focused on application of DRL to obstacle avoidance and path planning problems and very few have attempted to study the ability of DRL based methods to undertake path following and compare them with traditional methods. Particularly, implementation in practical experiments are scarce. There is still some way to go before RL based controllers can be seamlessly integrated into practice because the dynamics in training is usually simplified in simulations and environmental forces are neglected (Cui et al., 2017). Therefore, this study explores the development of a DQN based controller for path following task of a ship whose dynamics is defined by a strongly non-linear model, and investigates its performance in the presence of significant environmental forces. The simulated results are also validated against field experiments.

The rest of the paper is organized as follows. Section 2 deals with the dynamics of the ship. The non-linear equations of motions and the simulated maneuvering tests are discussed in detail. Section 3 introduces the basics of reinforcement learning, Q-learning algorithm and the DQN algorithm. Section 4 illustrates on how the algorithm mentioned in Section 3 is applied to the problem of ship path following problem. Section 5 describes the results obtained and further analyzes the performance of the DQN agent by evaluating various maneuvers. Section 6 describes the experimental validation of the RL agent trained in simulations. Section 7 discusses modeling wind forces and analysis of path following in the presence of wind. Based on the results, the salient features of the DRL controller are highlighted in Section 8. This section also compares the performance of RL agent against a traditional PD controller. Finally, Section 9 summarizes the results and discussion of this study.

2. Ship dynamic model

The KCS vessel is chosen for running numerical simulations and evaluating the control algorithms used in this research. The ship dynamics are mathematically modeled with help of the MMG (Maneuvering Modeling Group) model (Yasukawa and Yoshimura, 2015). The 3-DOF non-linear equations of motion are used to solve for ship maneuvering motions including surge, sway and yaw motions. The equations of motion are solved progressively at each time step as an initial value problem using a Runge–Kutta implicit solver. The commanded rudder angle δ_c is provided as an input at each time step. The particulars of the KCS vessel used for simulating the ship dynamics are provided in Table 1.

Two coordinate systems are defined to track the vessel. The first system is a global coordinate system (GCS) that is an earth fixed coordinate frame with its z-axis pointed down. The second is a body coordinate system (BCS) that is fixed to the body and moves with the vessel. The origin of the BCS is located at the intersection of midship, centerline and waterline of the vessel with its x-axis pointed towards the bow, y-axis pointed towards starboard and z-axis pointed towards the keel of the vessel. Both these coordinate frames are shown in Fig. 1. The heading angle ψ is defined as the angle between the x-axes of the GCS and BCS frames. The position and orientation of the vessel is denoted by $\boldsymbol{\eta} = [x_o, y_o, \psi]^T$ where x_o and y_o denote the position of

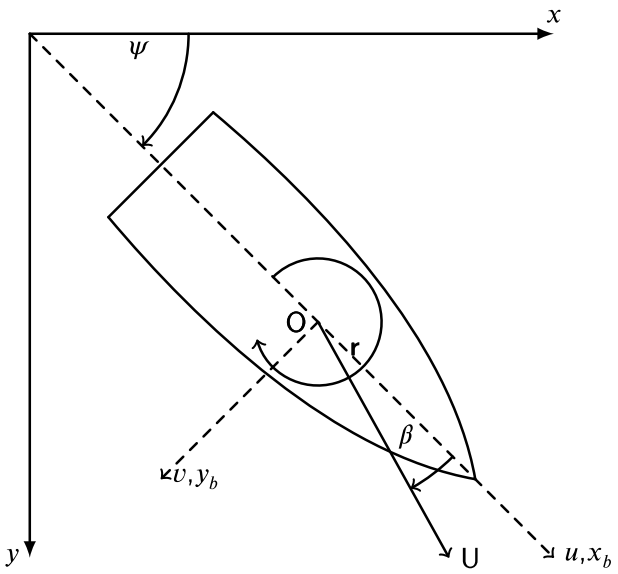


Fig. 1. Representation of ship kinematic variables.

the origin of BCS expressed in GCS. The velocity vector of the body in BCS is given by $\mathbf{V} = [u, v, r]^T$ where u, v and r represent surge, sway and yaw velocities of the vessel respectively and are expressed in BCS frame. The ship's speed is given by $U = \sqrt{u^2 + v^2}$. The drift angle (β) is defined as the angle between the total velocity vector and the longitudinal direction of ship and is given by $\beta = \tan^{-1}(-v/u)$. The kinematics of ship motion are represented by (1)

$$\dot{\boldsymbol{\eta}} = [R(\psi)]\mathbf{V} \tag{1}$$

where $[R(\psi)]$ represents the rotation matrix given by

$$[R(\psi)] = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Any vector in BCS when pre-multiplied by the rotation matrix will result in the same vector expressed in GCS.

The MMG model used for simulating ship maneuvering (Yoshimura and Masumoto, 2012) is shown in (3).

$$\begin{aligned} &(m + m_x)\dot{u} - mvr - mx_G r^2 = X \\ &(m + m_y)\dot{v} + mx_G\dot{r} + mur = Y \\ &(I_{zz} + J_{zz})\dot{r} + mx_G\dot{v} + mx_G ur = N \end{aligned} \tag{3}$$

In (3), m is the mass of the KCS ship, I_{zz} is the second mass moment of inertia in yaw and m_x, m_y and J_{zz} are the surge and sway added masses and yaw added mass moment of inertia respectively. X, Y and N represent the external surge, sway forces and yaw moment acting on the vessel expressed in BCS coordinate frame.

The surge and sway equations are non-dimensionalized by dividing both sides of the surge and sway equations by $\frac{\rho}{2}U^2 L d_{em}$ where ρ is the density of sea water, L is the length of the vessel, U is the design speed of the vessel and d_{em} is the draft of the vessel. Similarly, the yaw equation of motion is non-dimensionalized by divided both sides by $\frac{\rho}{2}U^2 L^2 d_{em}$. This non-dimensionalization is consistent with the prime-II system of normalization described by Fossen (1999). The non-dimensionalizing factors for various quantities is shown in Table 2. The resulting non-dimensional equations of motion are shown in (4)

$$\begin{aligned} &(m' + m'_x)\dot{u}' - m'v'r' - m'x'_G r'^2 = X' \\ &(m' + m'_y)\dot{v}' + m'x'_G\dot{r}' + m'u'r' = Y' \\ &(I'_{zz} + J'_{zz})\dot{r}' + m'x'_G\dot{v}' + m'x'_G u'r' = N' \end{aligned} \tag{4}$$

Table 2
Prime-II system of normalization.

Parameter	Prime-II system
Length	L
Velocity	U
Angular velocity	U/L
Time	L/U
Acceleration	U^2/L
Angular acceleration	U^2/L^2
Mass	$0.5\rho L^2 d_{em}$
Force	$0.5\rho U^2 L d_{em}$
Moment	$0.5\rho U^2 L^2 d_{em}$

Table 3
Ship parameters.

Parameter	Non-dimensional value
Surge added mass (m_x)	0.006269
Sway added mass (m_y)	0.155164
Yaw added mass moment (J_{zz})	0.009268
Yaw mass moment of inertia (I_{zz})	0.011432
Mass of the vessel (m)	0.182280

where $(.)'$ denotes a non-dimensionalized quantity with the non-dimensional factor as specified in Table 2. For simplicity in notation, the prime is avoided in the rest of the paper and all quantities presented henceforth are assumed to be non-dimensionalized according to the factors shown in Table 2. The non-dimensional mass and added mass terms are specified in Table 3.

The non-dimensional external forces and moments can be decomposed into components due to hull, propeller and rudder as shown in (5).

$$\begin{aligned} X &= X_H + X_R + X_P \\ Y &= Y_H + Y_R \\ N &= N_H + N_R \end{aligned} \tag{5}$$

where the subscripts H, R and P represent the hull, rudder, and propeller effects respectively.

2.1. Forces due to hull

The external fluid forces and moments acting on the hull can be expressed as a Taylor series in u, β and r as shown in (6). The non-dimensional hull hydrodynamic coefficients for the KCS vessel used in (6) were obtained from Yoshimura and Masumoto (2012) and are reported in Table 4. However, it is also possible to determine these coefficients through system identification methods applied to data collected from free running ship models (Vijay and Somayajula, 2022).

$$\begin{aligned} X_H &= X_0 u^2 + X_{\beta\beta}\beta^2 + (X_{\beta r} - m_y)\beta r \\ &\quad + X_{rr}r^2 + X_{\beta\beta\beta\beta}\beta^4 \\ Y_H &= Y_\beta \beta + (Y_r - m_x)r + Y_{\beta\beta\beta}\beta^3 + Y_{\beta\beta r}\beta^2 r \\ &\quad + Y_{\beta rr}\beta r^2 + Y_{rrr}r^3 \\ N_H &= N_\beta \beta + N_r r + N_{\beta\beta\beta}\beta^3 + N_{\beta\beta r}\beta^2 r \\ &\quad + N_{\beta rr}\beta r^2 + N_{rrr}r^3 \end{aligned} \tag{6}$$

2.2. Forces due to propeller

The non-dimensional propeller thrust acting in the surge direction can be calculated using (7)

$$X_p = 2(1-t)K_T {D_p}^4 n^2 \frac{L}{d_{em}} \tag{7}$$

where D_p is the propeller diameter, n is the propeller revolution rate, K_T is the propeller thrust coefficient and t is the thrust deduction factor. The thrust deduction factor t is included to capture the effect of

Table 4
Ship hydrodynamic coefficients.

Parameter	Non-dimensional value
X_0	−0.0167
$X_{\beta\beta}$	−0.0549
$X_{\beta r} - m_y$	−0.1084
X_{rr}	−0.0120
$X_{\beta\beta\beta\beta}$	−0.0417
Y_β	0.2252
$Y_r - m_x$	0.0398
$Y_{\beta\beta\beta}$	1.7179
$Y_{\beta\beta r}$	−0.4832
$Y_{\beta rr}$	0.8341
Y_{rrr}	−0.0050
N_β	0.1111
N_r	−0.0465
$N_{\beta\beta\beta}$	0.1752
$N_{\beta\beta r}$	−0.6168
$N_{\beta rr}$	0.0512
N_{rrr}	−0.0387

Table 5
Propeller parameters.

Parameter	Non-dimensional value
t	0.207
D_p	0.03435
n	35.86
w	0.355
a_0	0.5228
a_1	−0.4390
a_2	−0.0609

additional resistance observed when the propeller is operating behind the hull as compared to bare hull resistance. The thrust coefficient K_T is obtained by curve fitting the propeller open water test data and for the KCS vessel is given by (8)

$$K_T = a_0 + a_1 J + a_2 J^2 \tag{8}$$

Here J represents the advance coefficient and is given by (9)

$$J = \frac{u(1-w)}{nD_p} \tag{9}$$

where w represents the effective wake fraction. The effective wake fraction w accounts for the reduction in inflow fluid velocity to the propeller when operating in the wake of the hull. The values of propeller parameters are specified in Table 5.

2.3. Forces due to rudder

The surge and sway forces and the yaw moment due to the rudder are given by

$$\begin{aligned} X_R &= -(1-t_R)F_N sin\delta \\ Y_R &= -(1+a_H)F_N \cos\delta \\ N_R &= -(x_R + a_H x_H)F_N \cos\delta \end{aligned} \tag{10}$$

where δ is the instantaneous rudder angle, t_R is the steering resistance deduction factor, x_R is the non-dimensional location of the rudder with respect to midship and x_H is the non-dimensional position of the point where additional lateral force acts. The values of these parameters for the KCS vessel have been taken from Yoshimura and Masumoto (2012) and are given in Table 6. F_N is the non-dimensional rudder normal force and can be calculated as shown in (11)

$$\begin{aligned} F_N &= \frac{A_R}{Ld_{em}} f_\alpha U_R^2 \sin\alpha_R \\ \alpha_R &= \delta - \tan^{-1}\frac{-v_R}{u_R} \end{aligned} \tag{11}$$

where A_R is the rudder area, U_R is the non-dimensional rudder inflow velocity and f_α is the gradient of rudder lift coefficients. f_α can be approximated as a function of rudder aspect ratio (λ), as shown in (12).

Table 6
Rudder force parameters.

Parameter	Value
$\frac{A_R}{Ld_{em}}$	0.0182
t_R	0.258
a_H	0.361
x_H	−0.436
x_R	−0.5
l_R	−0.755
γ_R (starboard)	0.492
γ_R (port)	0.338
κ	0.633
ϵ	0.956
λ	2.164
η	0.7979

$$f_\alpha = \frac{6.13\lambda}{(2.25+\lambda)} \tag{12}$$

v_R and u_R in (11) are the lateral and in-line velocity components ($U_R = \sqrt{v_R^2 + u_R^2}$) and can be computed as shown in (13).

$$\begin{aligned} u_R &= \epsilon(1-w) \\ &\times \sqrt{\eta\left(1+\kappa\left(\sqrt{\left(1+8\frac{K_T}{\pi J^2}\right)}-1\right)\right)^2 + (1-\eta)} \\ v_R &= \gamma_R(v + rl_R) \end{aligned} \tag{13}$$

In (13), l_R is the correction of flow straightening factor to yaw-rate, γ_R is the flow straightening factor of hull and the value for these constants are given in Table 6. η is the ratio of propeller diameter to the rudder height $\left(\eta = \frac{D_p}{h_R}\right)$, ϵ and κ are constants identified from experiments performed by Yoshimura and Masumoto (2012).

2.4. Rudder angle variation

In practical scenarios the rudder angle cannot be varied instantaneously as this would mean that the rudder rate $\dot{\delta}$ would be very large. In this study, the actual rudder angle δ is governed by a first order equation as shown in (14) where δ_c denotes the commanded rudder angle. The actual rudder angle δ evolves slowly for a step change in commanded rudder angle δ_c. A lower value of T_R leads to a faster response time.

$$T_R\dot{\delta} + \delta = \delta_c \tag{14}$$

In addition, the rudder rate $\dot{\delta}$ is also saturated at $\dot{\delta}_{max}$. So the rudder-rate is given by:

$$\dot{\delta} = \begin{cases} \frac{\delta_c - \delta}{T_R} & \text{if } \left|\frac{\delta_c-\delta}{T_R}\right| \le \dot{\delta}_{max} \\ \dot{\delta}_{max} & \text{if } \frac{\delta_c-\delta}{T_R} > \dot{\delta}_{max} \\ -\dot{\delta}_{max} & \text{if } \frac{\delta_c-\delta}{T_R} < -\dot{\delta}_{max} \end{cases} \tag{15}$$

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.

2.5. Turning circle maneuver

The turning circle maneuver is one of the standard tests performed to check the maneuvering capabilities of a vessel. In this test, the ship is initially oriented at heading $\psi = 0°$ and a constant 35° rudder is executed, either towards starboard or port. The steady turning diameters observed from the numerical model developed in this study were

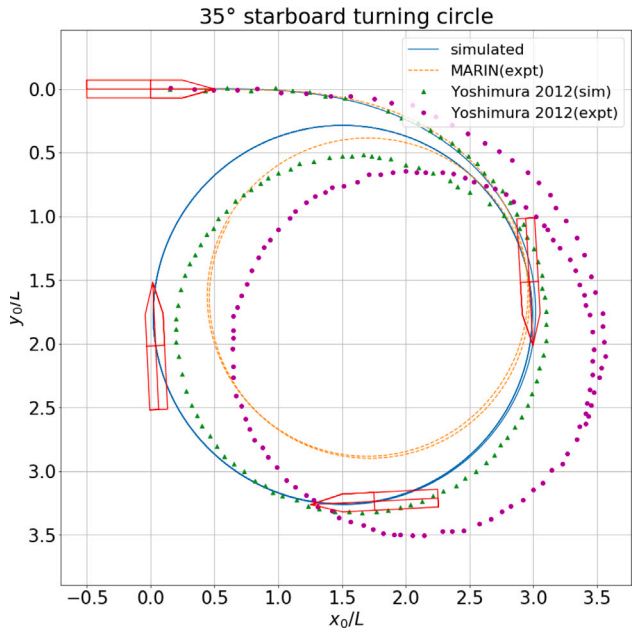


Fig. 2. Starboard 35° turning circle.

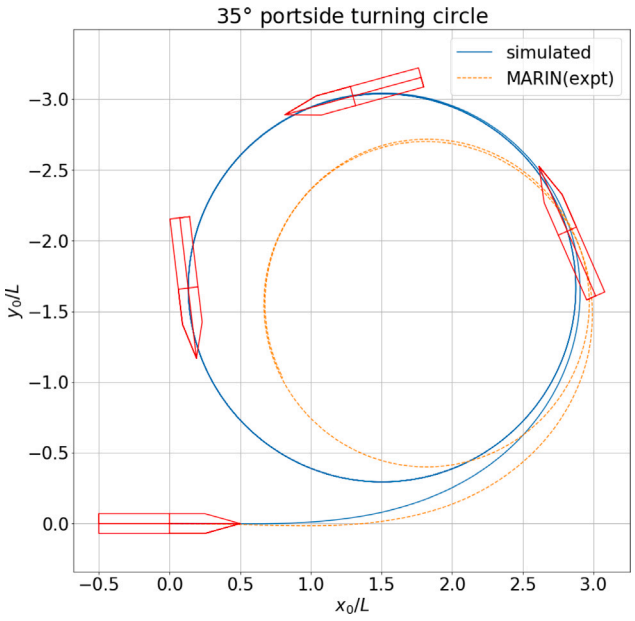


Fig. 3. Port 35° turning circle.

observed to be $2.97L$ and $2.74L$ for starboard and port side turning circles respectively. The port and starboard turning circles differ due to asymmetry in flow through the rudder. The simulated results were compared with its corresponding experimental turning circle tests for the same vessel and are shown in Figs. 2 and 3. The experimental data was obtained from SIMMAN 2021 workshop (experiments were performed by MARIN) and by digitizing plots from Yoshimura and Masumoto (2012). While the experiments in Yoshimura and Masumoto (2012) were performed at 1:75.5 scale, the MARIN experiments for SIMMAN 2021 workshop were conducted at 1:37.89 scale.

It can be seen that the steady turning diameter from the experiments and simulations of Yoshimura and Masumoto (2012) agree fairly well with the results of this study. The minor differences observed in the simulations of both studies can be partly attributed to digitization errors as the plots in Yoshimura and Masumoto (2012) were not of high resolution. In addition the inclusion of smooth rudder variation through a first order system as described above was not considered by Yoshimura and Masumoto (2012) and might also have contributed to the difference.

It is seen that the steady turning diameter observed from SIMMAN 2021 experiments is smaller than the values predicted from both (Yoshimura and Masumoto, 2012) and this study. It is well known that the lift and drag generated by an airfoil section depend strongly on the Reynolds number. Since Froude scaling is followed when performing scaled model experiments, the Reynolds number of the flow over the rudder will be significantly different as the scale ratio varies. Therefore, the rudder forces will not be dynamically similar as the scale ratios change. The difference between the experimental turning circle maneuvers of SIMMAN 2021 workshop and Yoshimura and Masumoto (2012) can be attributed to these scale effects.

2.6. Zig-Zag test

In the Zigzag maneuver test, the rudder angle δ is kept at a constant value say δ_0 until the ship's heading angle reaches the specified value ψ_0. When heading ψ reaches ψ_0, the rudder is reversed to $-\delta_0$. Now the rudder is held constant at $-\delta_0$ until the ship's heading reaches $-\psi_0$, when the rudder direction is reversed again. In 20°-20° zigzag maneuver the δ_0 and ψ_0 values are both 20°. Zig-zag maneuver helps us

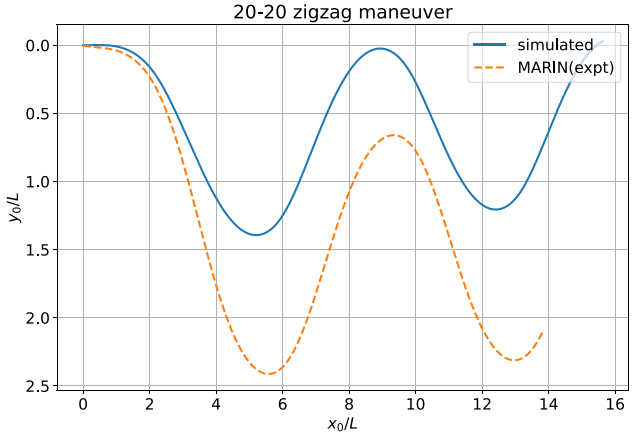


Fig. 4. Rudder angle (δ) and ship's heading (ψ).

to understand the course-changing ability of the ship (how fast the ship can change its heading on command). One of the crucial quantities to note in this maneuver is the *overshoot angle*. It is the maximum heading reached by the ship beyond ψ_0 after the rudder change is applied. A low overshoot angle indicates better controllability of the ship.

Fig. 4 shows the ship's position during the simulated zigzag maneuver compared with the SIMMAN 2021 experimental data for the same test. While the experiment on a free running model allows the vessel to heel upon the application of rudder, this effect is not incorporated in the simulations, which are restricted to three degrees of freedom. This and the scale effects are believed to be some of the reasons for the differences between the simulations and the experiments. Fig. 5 shows the variation of the ship's heading and the rudder angle with time. The maximum ship heading was observed to be 30.53° which indicates an overshoot of 10.53°.

3. Deep-Q learning

RL is a machine learning technique which involves intelligent agents that learn to take optimal decisions by interacting with an environment with a goal of accumulating rewards. The agent can try a set of actions

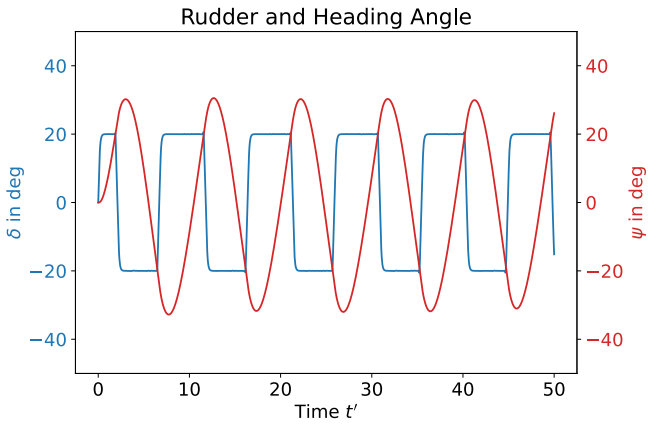


Fig. 5. Rudder angle (δ) and ship's heading (ψ).

at any certain state and it learns with experience (Sutton and Barto, 2018). Choosing a particular action from a particular state will result in rewards that can be positive or negative. The agent progressively gets better at the task by trial and error mechanism with the rewards as feedback. The agent starts the episode at a specified state then executes a series of actions moving through various states till the episode is terminated. An episode is terminated either when the agent has successfully reached the goal state or when a terminating condition (usually denoting failure) is satisfied. The motive of the agent is to maximize the *return* obtained in an episode defined as the cumulative reward obtained in that episode.

The current state of the agent depends only on the previous state and the action chosen at the previous state. This kind of an environment is called a Markov decision process (MDP). Typical ship dynamics involves consideration of retardation functions that model the memory of the fluid. However, in a calm water condition this can be relaxed and can be represented as an MDP, which makes it a suitable learning environment for the RL framework.

3.1. Q-learning

Q-Learning is a fundamental RL algorithm where every action executable at a particular state has a value associated with it called the Q-value. The Q-value for any action–state pair represented as $Q(s, a)$., is the expected cumulative reward gained by the agent by performing an action a at a certain state s and choosing optimal actions thereon. The agent learns the Q-values through experience from numerous trials.

A major limitation of Q-learning is that it only works when the action space and the state space are discrete, which is not so for most real-life problems. The number of Q-values needed will be large if a continuous action space is discretized finely. To overcome this problem, the Q-table that keeps track of the Q-values for every state–action pair in Q-learning can be substituted by a function approximator. Artificial neural networks (ANN) are commonly used as function approximators due to their powerful representational capacity and this has led to a new set of RL algorithms called deep reinforcement learning (DRL).

3.2. Deep Q-network

In deep Q-learning, the state variables are input to an ANN known as DQN (Deep Q-network) which outputs the Q-values for each corresponding action. Fig. 6 illustrates this behavior. Unlike Q-learning, DQN can be used for a continuous state space as well. Just like Q-learning, the action which corresponds to the highest Q-value is executed. Since the Q-value is now a function of the parameters (weights and biases) of the network represented by θ, the Q-value can be represented as $Q(s, a; \theta)$.

The network parameters have to be updated after every episode to improve the Q-values output by the network, and thus improving the policy. Network parameters are updated by backpropagation with the help of an optimizer that tries to minimize the loss function given by (16).

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[(y - Q(s, a; \theta))^2\right] \tag{16}$$

$\mathbb{E}_{s,a,r,s'}[.]$ represents the expected value over all the possible state transitions (s, a, r, s'). Each state transition (s, a, r, s') denotes a transition of the agent from state s to s' by taking an action a and earning a reward r in the process. Note that y is the sum of the observed reward $r(s, a)$ during the transition from s to s' and the expected reward thereon assuming that the agent chooses the action with the highest Q-value and is shown in (17). Note that the latter denotes the expected rewards from future and is discounted by a factor $\gamma \leq 1$ to adjust the weightage the agent gives to rewards at current time step and expected future rewards.

$$y = r(s, a) + \gamma \max_{a'} Q(s', a'; \theta') \tag{17}$$

Note that $(y - Q(s, a; \theta))$ is called the TD (temporal difference) error. y is the target value and it is computed with the help of a target network. The target network is a copy of the Q-network but one which is updated at a slower rate by using a soft update factor τ. In (17), θ' represents the parameters of the target network.

In each episode the agent starts from an initial state and takes a sequence of actions determined by the policy given by $\pi(s) = \max_a Q(s, a; \theta)$. This sequence of actions continue until the agent reaches the goal or if a termination condition is met. The termination condition is invoked when the ship has an along track distance greater than the goal location and its velocity is pointed away from the goal point. This is described in detail later (in Section 4.4). However, if the ship is unable to reach the destination within a maximum number of time steps, then too the episode is terminated.

After each episode state transitions observed by the agent are sampled and added to an experience replay buffer. The replay buffer consists of multiple transitions from both the most recent episode and the previous episodes. As the number of transitions in the replay buffer reach the maximum number of transitions that can be stored, the older transitions are replaced by transitions from newer episodes. A mini batch of transitions is randomly chosen from the replay buffer for updating the network after each episode, which ensures that the updates are unbiased and the transitions are reused in the training process. After each episode the training process updates the parameter set θ to minimize the loss function defined in (16) for the mini batch of transitions sampled from the replay buffer.

With sufficient training, the model will learn to minimize the loss function, which means that the neural network will learn to predict the Q-values in such a way that the transitions observed in the training data will generally correspond to the choice of the action with the largest Q-values. However, care must be taken to not over-train the model as this can lead to poor generalization when the agent experiences scenarios that it was not exposed to in the training data.

3.3. Epsilon-Greedy algorithm

In a RL problem the agent must trade-off between exploration and exploitation to arrive at the optimal policy. Exploration refers to the agent exploring different actions to observe its effect on the cumulative reward. This allows the agent to explore the environment and to look for more optimal action sequences that it might not have learned so far. On the other hand, once a policy has been established leading to the goal, the agent will try to reinforce this policy by choosing actions based on this policy. This behavior is known as exploitation that refers to the tendency of the agent to maximize its rewards based on its experience.

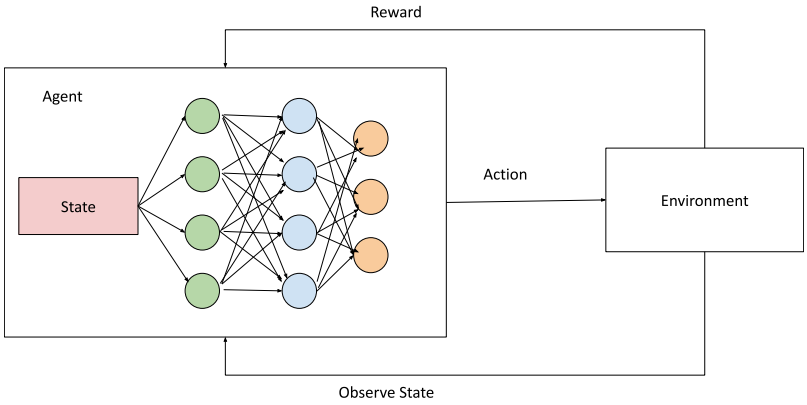


Fig. 6. DQN algorithm flowchart.

Both exploration and exploitation are equally important. Without exploitation the agent will never learn to settle onto the optimal policy and would keep wandering. However, without exploration, the agent may be stuck on an sub-optimal policy.

The ϵ-Greedy algorithm is one common approach used to balance between exploration and exploitation tendencies of the agent. A parameter ϵ is introduced, which is the probability with which the agent takes a random action from any particular state. So the agent takes the action corresponding to the highest Q-value with probability $1-\epsilon$ (exploitation) and chooses random actions with probability ϵ (exploration). This policy in this case can be mathematically represented as:

$$\pi(s) = \begin{cases} \arg\max_a Q(s,a) \text{with a probability of } 1-\epsilon \\ \text{random action with a probability of } \epsilon \end{cases} \tag{18}$$

The DQN algorithm described above is expressed in a condensed form in Algorithm 1. In this study, tensorflow framework is used to model the RL agent (Abadi et al., 2015).

Algorithm 1 DQN algorithm

Initialise Q-network and target network with random parameters θ_0
Initialise an empty experience replay buffer D
for Episode = 1,2...N **do**
  for t=1,2...T **do**
    Choose random action a_t as per (18)
    Obtain reward r_t and next state s_{t+1}
    Add transition (s_t, a_t, r_t, s_{t+1}) to replay buffer D
    End episode if termination conditions are met
    if update frequency is a multiple of time step **then**
      Sample M random transitions from D
      Compute loss as below using y_i from (17)
      $L(\theta) = \sum_{i=0}^{M} \frac{1}{M}[y_i - Q(s_i, a_i; \theta)]^2$
      Update the network using the computed loss
      Update target network as: $\theta' = \tau\theta + (1-\tau)\theta'$
    end if
  end for
end for

4. Implementation of DQN algorithm for ship navigation

This section discusses the details of DQN framework applied to ship navigation. An observation state of the ship is provided as an input to the Q-network at every time step based on which the DQN agent takes actions to maneuver the ship towards the goal position. Fig. 7 shows the schematic representation of the problem statement.

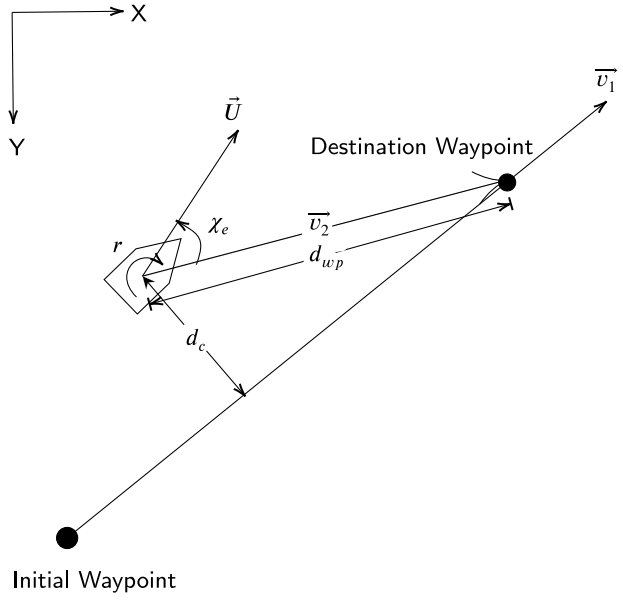


Fig. 7. Representation of the problem statement.

4.1. Observation states

The observation state represents the current state of the agent that is provided as an input to the Q-network at every time step. It is based on this information that the agent decides what action to execute. It is important to note that observation states should be independent of the waypoint to be applicable for any generically chosen waypoint or path formed by waypoints. In this study four observation states consisting of four variables: *cross-track error (d_c), course angle error (χ_e), distance to destination (d_{wp})* and *yaw rate(r)* are chosen.

4.1.1. Cross-track error

The initial and destination (or goal) waypoint coordinates are denoted by (x_i, y_i), (x_g, y_g) respectively. The ship coordinates at an intermediate time step are denoted by (x, y). Cross-track error (d_c) is defined as the perpendicular distance of the current ship's coordinates from the line joining initial and destination waypoints.

Two vectors $\hat{v}_1$ and $\vec{v}_2$ are defined as shown in Fig. 7. $\hat{v}_1$ is a unit vector in the direction of the line joining the initial and destination waypoints and $\vec{v}_2$ is the vector pointing towards the destination from current ship location. It can be seen from Fig. 7 that the cross-track

error d_c is given by the cross product of the two vectors, $\hat{v}_1$ and $\vec{v}_2$.

$$\begin{aligned}\vec{v}_1 &= (x_g - x_i)\hat{i} + (y_g - y_i)\hat{j} \\ \hat{v}_1 &= \frac{\vec{v}_1}{|\vec{v}_1|} \\ \vec{v}_2 &= (x_g - x)\hat{i} + (y_g - y)\hat{j} \\ d_c &= \vec{v}_2 \times \hat{v}_1\end{aligned} \tag{19}$$

Note that a deviation of the vessel to the left of the line joining the waypoints will result in a negative cross track error and a deviation to the right will result in a positive cross track error.

4.1.2. Course-angle error

The course angle error is defined as the angle between the direction of ship's instantaneous velocity and the direction of the desired heading and is calculated as shown in (20). It is computed by taking the smallest signed angle (ssa) of the difference in the direction of ship's instantaneous velocity and the direction of the desired heading towards the goal waypoint (Fossen, 1999).

$$\chi_e = ssa\left(arctan2(\vec{U}) - arctan2(\vec{v_2})\right) \tag{20}$$

Here $\vec{U}$ is the ship's instantaneous velocity represented in global coordinates and can be calculated as shown in (21).

$$\begin{aligned}\vec{U} &= \dot{x}\vec{i} + \dot{y}\vec{j} \\ \dot{x} &= u\cos(\psi) - v\sin(\psi) \\ \dot{y} &= u\sin(\psi) + v\cos(\psi)\end{aligned} \tag{21}$$

4.1.3. Distance to destination

The distance to the destination waypoint d_{wp} is defined as shown in (22).

$$d_{wp} = \sqrt{(x_g - x)^2 + (y_g - y)^2} \tag{22}$$

4.2. Action states

The commanded rudder angle, δ_c is the action that the agent can choose at every time step and has been divided into a set of three discrete values $\delta_c \in [-35°, 0°, 35°]$. As the Q-values output corresponds to the number of actions, the Q-network computes three Q-values. So, at every time-step, the DQN agent executes one of the three actions. Note that the agent only controls the commanded rudder angle and the actual rudder angle (δ) still varies smoothly as governed by (14) and (15).

4.3. Reward structure

The reward functions for a RL setup considerably influence its decision making. For this problem, rewards were chosen for the cross-track error, the course angle and the distance to the destination waypoint. The reward associated with cross-track error (d_c) is given by

$$r_1 = 2\exp\left(\frac{-d_c^2}{12.5}\right) - 1 \tag{23}$$

The function ensures that the reward value falls within the range of -1 to 1 for each time step, and a graphical representation of the curve can be observed in Fig. 8. The coefficient of the exponential term is chosen such that for $|d_c| > 3$, the reward associated with cross track error becomes negative. The reward associated with the course angle error (χ_e) is given by

$$r_2 = 1.3\exp\left(-10\left|\chi_e\right|\right) - 0.3 \tag{24}$$

At every time step, the function has been selected in a way that the reward value remains within the range of -0.3 to 1 as shown in Fig. 9. The coefficient of the exponential term is chosen such that for $|\chi_e| > 8.5^0$, the reward associated with the course angle error becomes negative. Note that the cross-track error has d_c^2 as an argument to the exponential function while the course angle error has $|\chi_e|$ as the argument. This ensures that the reward decreases at a faster rate for cross-track error than the course angle error. This choice of the two reward functions enables the agent at each time step to prioritize reducing cross-track error over reducing the course angle error.

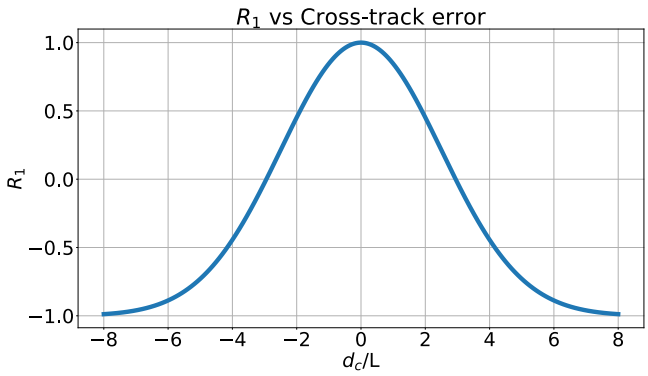


Fig. 8. Cross track reward.

It should be noted that the overall reward during the training process should be negative to ensure that the agent tries to reach the goal quickly and does not loiter around the environment to accumulate more rewards. To ensure this, a third reward associated with distance to the destination waypoint (d_{wp}) is introduced and is given by

$$r_3 = \frac{-d_{wp}}{4} \tag{25}$$

The coefficient $1/4$ is found to be sufficient to result in an overall negative reward for the episode. Based on this reward structure it can be seen that when the goal waypoint is farther than $4L$ from the current position, the agent will aim to reduce the distance to goal first rather than focusing on keeping course. This value of 4 is arrived at by observing that the total episode reward is negative. It is found that the factor of 4 along with the reward structure taken for r_1 and r_2 provide a path that smoothly decreases course angle error and cross track error as the distance reduces. Other values are also possible but they will lead to a different rate of decrease of cross track error and course angle error with distance. For example a $-d_{wp}/8$ reward with r_1 and r_2 being the same will cause the cross track error to be weighted more than distance to goal when the vessel is $8L$ away from the goal waypoint. However for $-d_{wp}/4$ this reduces the transition point to $4L$ away from the goal waypoint.

The reward at time step m is denoted by r_m and is given as the sum of the rewards from each component at that time step. The total reward for the episode (also known as episode return) is obtained by summing the reward accumulated by the agent from all time steps and is shown in (26).

$$\begin{aligned}r_m &= r_1 + r_2 + r_3 \\ R &= \sum_{m=0}^{n} r_m\end{aligned} \tag{26}$$

In (26), r_m is the reward at time step m and R is the episode return, which is the cumulative sum of rewards obtained at each time step(see Figs. 8 and 9).

4.4. Training process

In each episode, the ship starts from the origin (initial waypoint), oriented along the positive X-axis ($\psi = 0$) of the GCS and with an initial velocity of design speed in the surge direction. The ship has no initial acceleration in any of the 3-DOF and no initial velocity in the sway and

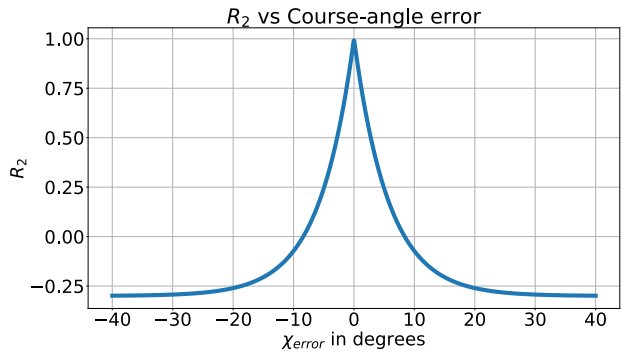


Fig. 9. Course angle reward.

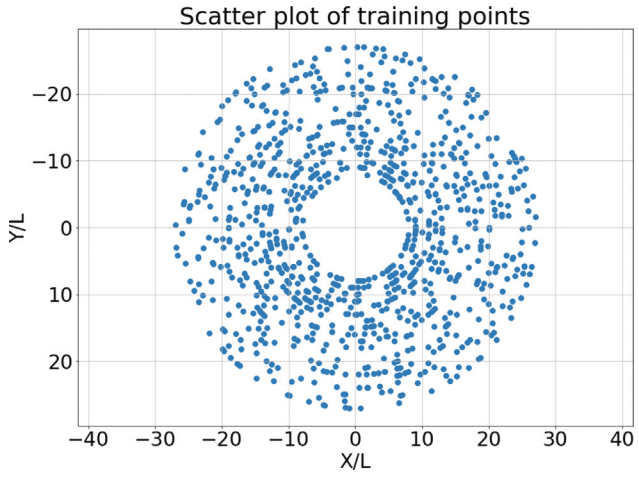


Fig. 10. Scatter plot of thousand random points with radius between $8L$–$28L$.

yaw motions. ϵ - greedy algorithm is used while training to allow for ample exploration. ϵ linearly decays from 1 to 0 throughout a majority of the training episodes and it is kept at 0 towards the end of training to allow for exploitation behavior.

In each training episode, the destination is chosen randomly, whereas the initial waypoint, velocity and heading are fixed at $(0, 0)$, 1 and 0° respectively. The radial distance of the destination waypoint from the initial waypoint is sampled from a uniform distribution between $8L$ to $28L$. The direction of the destination waypoint is sampled from a uniform distribution between 0 and 2π. Fig. 10 shows a plot of 1000 randomly sampled goal coordinates.

4.4.1. Episode termination

In every episode of training, the aim of the agent is to accumulate maximum possible return which subsequently navigates the ship towards the goal. A tolerance of 0.5L centered around the destination waypoint is specified. As long as the ship is able to enter this region, the positive termination condition is satisfied and the episode is considered to be successful. A terminal reward of +100 is given when this positive termination condition is achieved. However, when an untrained agent tries this it may not be successful. So, certain conditions need to be formulated to determine if the particular agent is still capable of reaching the destination point or if the episode is a failure and has to be terminated to avoid wasting time. The negative termination condition is given as follows:

$\vec{v_1} \cdot \vec{v_2} < 0$ and $\vec{U} \cdot \vec{v_2} < 0$

The first condition is only based on the ship's current position and does not depend on the ship's current velocity. In Fig. 11 it can be seen that the angle between $\vec{v_1}$ and $\vec{v_2}$ becomes obtuse, which also indicates that the dot product becomes negative, as soon as the ship crosses the line perpendicular to the line joining the initial and the destination waypoints. The second condition takes into consideration the direction of ship's velocity vector. Even if the ship has crossed the destination point, the episode is not terminated if the ship's velocity has a component that is still directed towards the destination point.

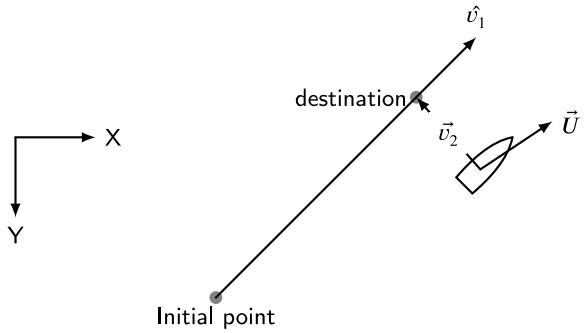


Fig. 11. Visualizing the terminating condition.

Table 7
Hyperparameters.

Hyperparameter	Value
Initial learning rate	0.001
Decay steps	5000
Decay rate	0.5
Hidden layers	(64,64)
Discount factor(γ)	0.95
Sample batch size	128
Replay buffer size	100000
Activation function	tanh
Maximum time steps	160
Time step interval Δt	0.3
Number of episodes	7000
Update frequency(time steps)	20
Target network update frequency(time steps)	1
Target update rate (τ)	0.01

4.5. Hyperparameters of the network

The agent is calibrated for satisfactory waypoint tracking by tuning the hyperparameters of the DQN. The agent is then tested by simulating various maneuvers for waypoint tracking. The learning rate is chosen as exponential decaying function. The hyperparameters for the model are given below in Table 7. The training losses and episode returns averaged over 100 episodes are shown in Fig. 12. The policy at 8000 episodes is chosen to strike a balance between under-fitting and over-fitting and will be tested in the next sections for path following. It is also important to note that this study has used TensorFlow version 2.9.1 which allows the 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 (TensorFlow, 2022). The code associated with this training has been made available through a Github repository.

5. Calm water results

5.1. Waypoint tracking

The performance of the RL agent is tested by analyzing waypoint tracking and path following in different situations. A starting state is chosen where the vessel is oriented along the global x-axis and initialized with unit non-dimensional velocity in the surge direction.

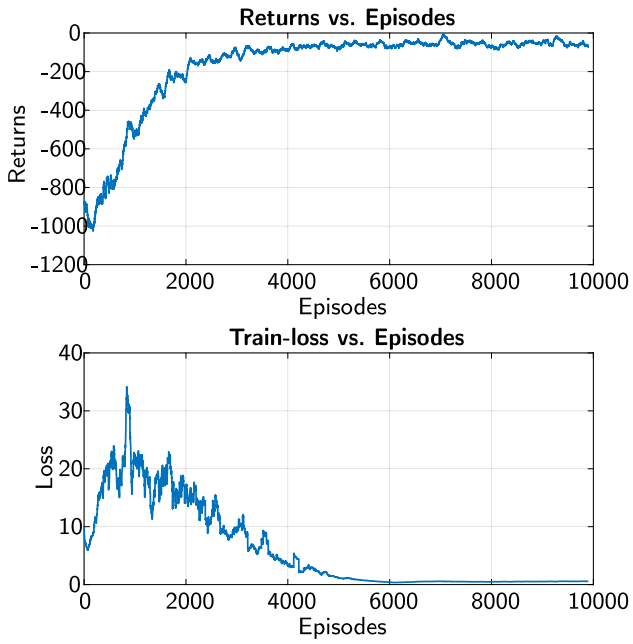


Fig. 12. Training returns and loss for the model.

The ability of the agent to track points in all four quadrants is investigated by specifying waypoints $(10L, 10L)$, $(10L, -10L)$, $(-10L, -10L)$ and $(-10L, 10L)$ with the same initial starting state as mentioned above in each case. Fig. 13 shows that the trained agent is successful in reaching the destination points in all four quadrants. It is also observed that the path is sufficiently smooth and that the agent tries to reduce the cross-track error and course angle error simultaneously.

5.2. Path following through waypoint tracking

Since the RL agent is successful in being able to guide the ship to the destination waypoint, it can be directed to follow complex paths that are discretized into adequate number of waypoints assuming that the chosen set of waypoints describe the path effectively.

To test the turning ability of the agent, circles of radius $6L$ and $12L$ are defined by discretizing them into 8 and 12 waypoints respectively as shown in Fig. 14. In each case the vessel starts at the intersection of the circle with the positive x-axis and is initially oriented towards the negative y-axis. Consecutive waypoints are provided such that the agent tries to track the circle in the anti-clockwise direction. It can be seen that the agent is able to follow the outer circle with radius $12L$ well. However, the deviation is larger for the shorter circle of radius $6L$. Although the agent is trained for tracking waypoints at least $8L$ away from initial waypoints, it is able track the $12L$ radius circle well, where the distance between waypoints is slightly greater then $5L$. However, the performance is worse while tracking a $6L$ radius circle, where the distance between waypoints is about $4L$. The root mean square cross track error for a trajectory is defined as

$$d_{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} d_c^2(n\Delta t)} \tag{27}$$

where N is the number of time steps in the trajectory and $d_c(n\Delta t)$ denotes the value of the cross track error at the nth time step. It seen from Fig. 14 that the d_{RMSE} for the larger circle is less than the corresponding value for smaller circle.

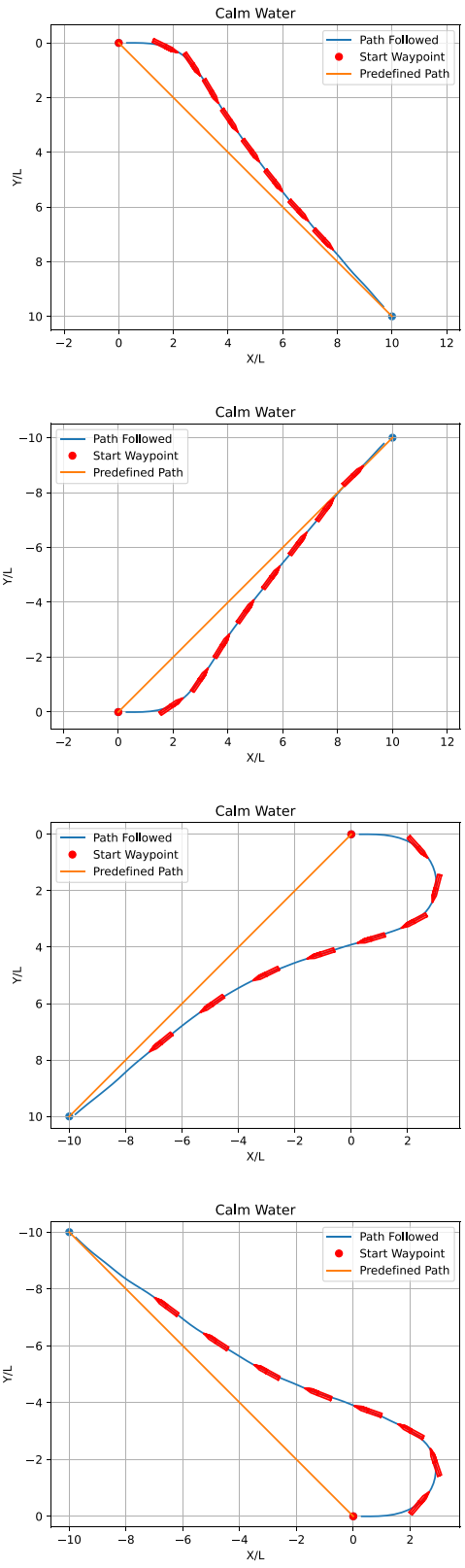
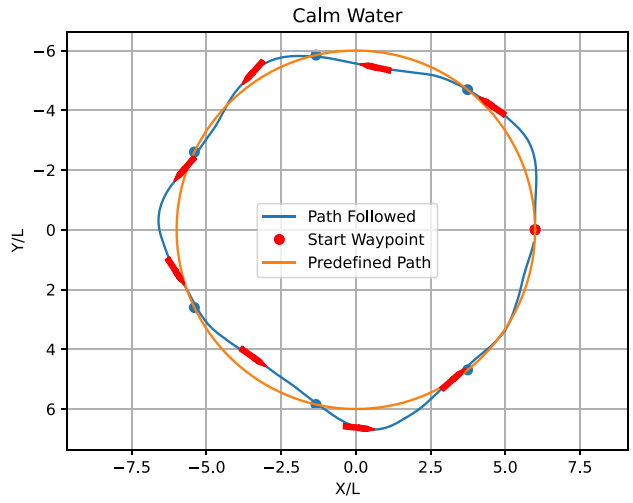
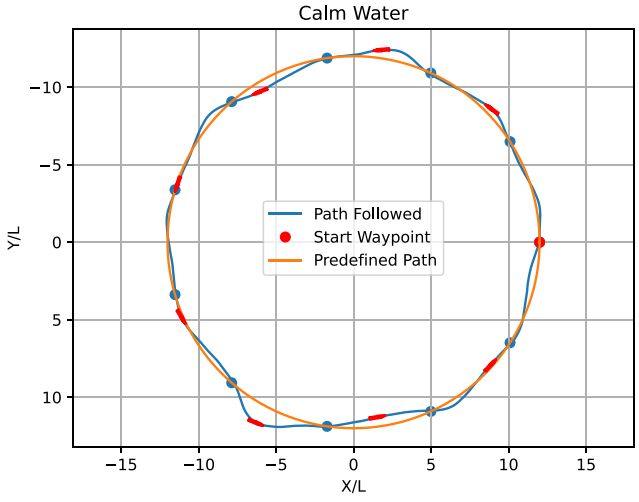


Fig. 13. Single waypoint tracking.

Fig. 15 shows the ability of the agent to track an elliptical path. An ellipse of major and minor axes of lengths $28L$ and $24L$ respectively is discretized into 15 waypoints. The ship starts at $(x, y) = (14L, 0)$ and is initially oriented along the negative Y-axis ($\psi = -\pi/2$). The



(a) Eight ($d_{RMSE} = 0.6095L$)



(b) Eight ($d_{RMSE} = 0.5395L$)

Fig. 14. Circle maneuver waypoint tracking.

distance between the waypoints is about $5L$ and the local radius of curvature along the path varies from about $10L$ to $16L$. It can be seen that the agent is able to track the elliptical path successfully with a $d_{RMSE} = 0.2969L$.

Finally, a path in the shape of "eight" is defined and is discretized into 23 waypoints. This path was chosen to observe the effect of asymmetry in the port and starboard turns due to the propeller rotation. It can be seen from Fig. 16 that the ship starts at origin with a heading of ψ=0 and completes the lower circle of radius $9L$ first before finishing the upper circle of the same radius to complete the eight maneuver successfully. The propeller rotation asymmetry does not seem to impact the tracking performance as the turns towards both sides are tracked equally well.

6. Experiments

In order to validate the simulation results against experiments, a scaled free running model of the KCS model is equipped with the policy learnt by the RL agent. Fig. 17 shows the scaled ship model deployed in the lake. The vehicle is equipped with a brushless DC motor that controls the propeller and a stepper motor that controls the rudder. The propeller RPM is kept fixed while the commanded rudder angle is controlled by the output of the RL agent policy. Fig. 18 displays the SBG Ellipse-A IMU and Ardusimple simpleRTK2B-SBC GPS system, both of which are equipped in the vehicle. Furthermore, Table 8 lists the

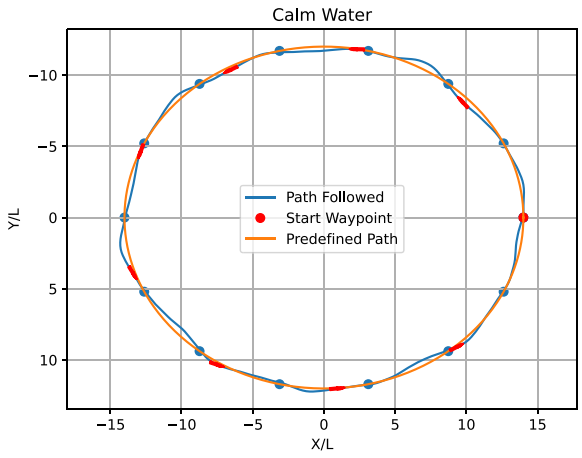


Fig. 15. Ellipse maneuver waypoint tracking ($d_{RMSE} = 0.2969L$).

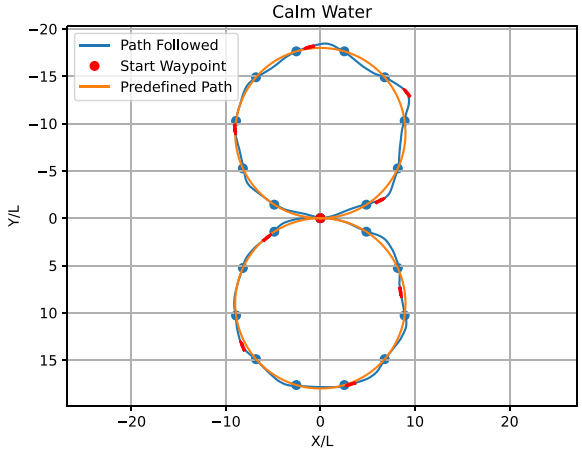


Fig. 16. Eight ('8') maneuver waypoint tracking ($d_{RSME} = 0.4557L$).

Fig. 17. 1:75.5 scaled model of KCS.

Table 8
Sensors onboard.

Sensor	ROS standard Message	Frequency (Hz)
SBG Ellipse-A (IMU)	sensor_msgs/Imu	100
simpleRTK2B SBC	sensor_msgs/NavSatFix	10

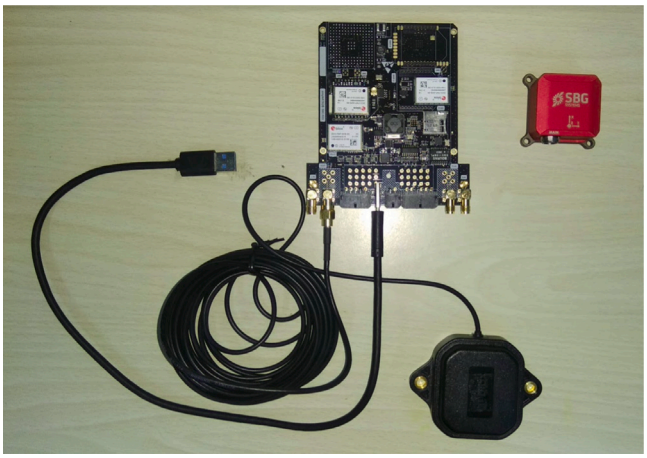


Fig. 18. Sensors used: simpleRTK2B SBC(left) and SBG Ellipse-A (right).

Fig. 19. Waypoints and the Trajectory: white represents the square generated by the waypoints, red represents the trajectory traversed by the vessel.

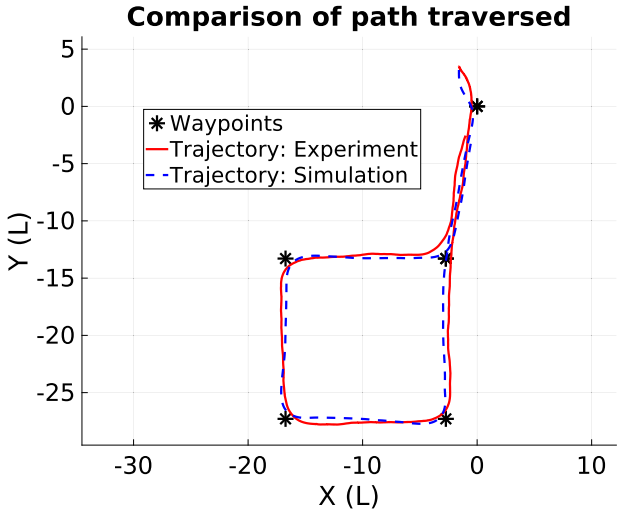


Fig. 20. Comparison of the path traversed by the model in experiment and simulation (Simulation $d_{RMSE} = 0.6286L$ Experimental $d_{RMSE} = 0.8257L$).

sampling rate and the message type published by the sensor. The sensor measurements are fused using a Kalman filter. The vehicle is tested at IIT Madras Lake with a span of 300 m and a width of 100 m (see Table 8). Due to the limitation of the space in the lake, the same maneuvers as described in the previous section could not be undertaken.

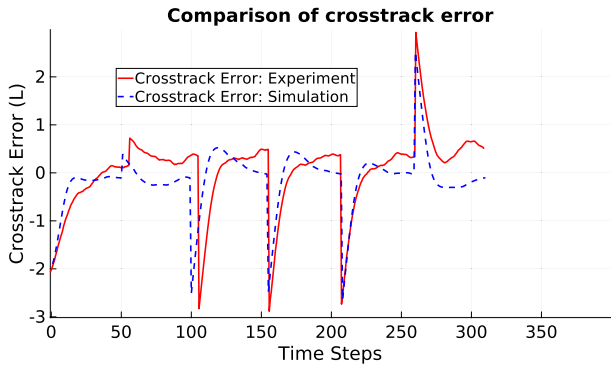


Fig. 21. Comparison of crosstrack error between experiment and simulation.

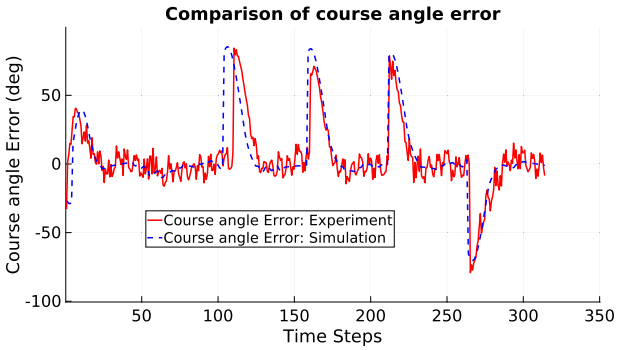


Fig. 22. Comparison of course-angle error between experiment and simulation.

Therefore, the simulations and experiments were performed for a set of waypoints that trace a 40 m square in the center of the lake. The GPS coordinates of the waypoints are listed in Table 9. The first and last waypoints are the same and are also chosen as the datum.

The experimental trajectory followed by the vehicle is shown in Fig. 19, where the white lines denote the straight lines between the waypoints defined in Table 9 and the red line represents the actual trajectory followed by the vessel. A comparison of results from the experiments with those from simulations are shown in Fig. 20. The variation of the cross-track error, defined in (19), is shown in Fig. 21. The corresponding variation of the course angle error, defined in (20), is shown in Fig. 22. The model tries to track the setpoint $d_c = 0$ for every edge of the square until the waypoint at the vertex is tracked. The circle of acceptance was chosen to be $3L$ for both simulation and experiments while tracking the waypoints. It can be seen that the course angle error from the simulations agree well with the corresponding values from experiments. It can be seen from Fig. 21 and Fig. 22 that the time of switching of waypoints also agrees between the simulations and experiments. The cross-track error curves between the simulation and experiment are slightly different. The simulations demonstrate a higher first overshoot of cross-track error after the switching of the waypoint, while the experiments do not exhibit such a significant overshoot. The difference can be attributed to the minor differences in dynamics of the simulation and the actual vessel and the presence of mild gust wind in the lake during the experiments. Overall the simulated trajectories and the experimental trajectories agree well and validates the model of the craft and the DQN based controller.

7. Performance with wind forces

In real ocean the vessel will not be operating in calm waters and would be subjected to environmental disturbances that arise due to wind, waves and currents. While all three disturbances are important

Table 9
Waypoints.

Waypoints	Latitude (deg)	Longitude (deg)
WP1	12.994224	80.239620
WP2	12.993859	80.239544
WP3	12.993474	80.239544
WP4	12.993474	80.239151
WP5	12.993859	80.239151
WP6	12.993859	80.239544
WP7	12.994224	80.239620

for a real ship, in this study the effect of wind disturbance on the vessel will be studied to understand the disturbance rejection capability of the controller.

7.1. Modeling wind forces

So far, the path following ability for an autonomous ship in the absence of any external environmental forces has been investigated using the DQN agent. This section discusses the ability of the agent to follow a desired path in the presence of moderate and severe wind conditions. Since the observation states do not include the wind parameters, the agent cannot take anticipatory actions (feedforward control) to compensate for wind forces and rather only reacts when it deviates significantly from the desired path (feedback control).

Wind forces and moments are modeled and added to the right hand side of (5). The non-dimensional velocity of the wind is denoted by V_w and the direction of wind is given by β_w. The direction of wind is defined as the angle the wind direction makes with respect to positive X-axis of GCS. The components of non-dimensional wind velocity with respect to the BCS of the ship are then defined as shown in (28).

$$\begin{aligned} u_w &= V_w \cos(\beta_w - \psi) \\ v_w &= V_w \sin(\beta_w - \psi) \end{aligned} \tag{28}$$

The non-dimensional velocity components of the ship relative to the wind can be defined as $u_{rw} = u - u_w$ and $v_{rw} = v - v_w$. The magnitude of non-dimensional relative wind velocity U_{wr} and relative wind angle with respect to the vessel γ_w are given by (29).

$$\begin{aligned} U_{wr} &= \sqrt{u_{rw}^2 + v_{rw}^2} \\ \gamma_w &= \arctan 2(-v_{rw}, -u_{rw}) \end{aligned} \tag{29}$$

The non-dimensional wind force components W_x, W_y and the non-dimensional wind yaw moment W_ψ can then be calculated as shown in (30)

$$\begin{aligned} W_x &= C_{wx}(\gamma_w)\frac{\rho_a}{\rho}A_x U_{wr}^2 \\ W_y &= C_{wy}(\gamma_w)\frac{\rho_a}{\rho}A_y U_{wr}^2 \\ W_\psi &= C_{w\psi}(\gamma_w)\frac{\rho_a}{\rho}A_y L_{OA} U_{wr}^2 \end{aligned} \tag{30}$$

where A_x and A_y represent the non-dimensional lateral and longitudinal projected areas of the hull 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 is the density of air and ρ is the density of water. L_{OA} represents the non-dimensional overall length of the vessel, non-dimensionalized with respect to the length between perpendiculars L. The values of wind parameters considered in this study are shown in Table 10.

The wind speed and the direction are varied to simulate different scenarios. The performance of the controller in the presence of constant wind is evaluated.

Table 10
Wind force parameters.

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

Table 11
Wind force cases.

Case	Path	v_w	β_w	d_{RMSE}
1	Line	6	$\pi/2$	$0.1227L$
2	Line	6	$-\pi/2$	$0.1615L$
3	Line	9	$\pi/2$	$0.9110L$
4	Line	9	$-\pi/2$	$0.9938L$
5	Ellipse maneuver	6	0	$0.8456L$
6	Eight maneuver	6	$\pi/4$	$0.3941L$

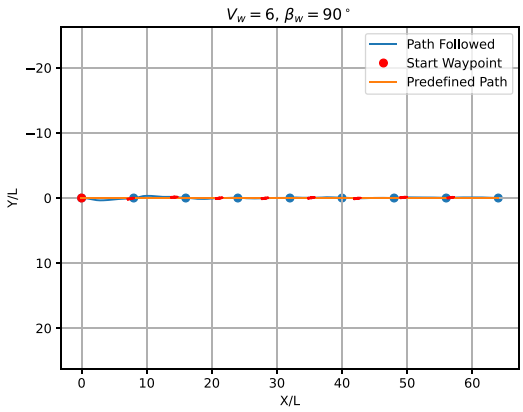
7.2. Results

Although several wind speeds and directions are investigated, six specific cases are shown in Table 11. The first four cases correspond to straight line paths and the fifth and sixth cases correspond to the ellipse and the eight maneuvers respectively. The non-dimensional wind speed and direction of wind and the observed d_{RMSE} values for each case are reported in Table 11. Fig. 23 shows the path tracked for each of the cases listed in Table 11. It can be seen that for straight line paths, the vessel initially deviates by a small amount in moderate winds ($V_w = 6U$) but later recovers to track the path successfully. Stronger winds lead to largest initial deviations and causes the d_{RMSE} to be multiple times larger than that observed in moderate winds. Cases 5 and 6 depict ellipse and eight maneuvers in the presence of a steady wind. Fig. 23 shows that the ellipse and eight maneuvers are also followed successfully in the presence of strong winds. In the ellipse maneuver too the beam wind results in a significant initial deviation. However, the model eventually is able to reach all the waypoints and complete the desired path. These simulations demonstrate the ability of the controller to reject the disturbances effectively.

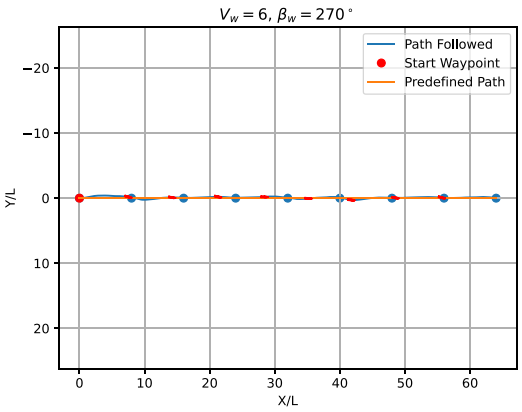
8. Discussion

This section discuss some of the salient features of the data driven RL controllers demonstrated in the previous sections.

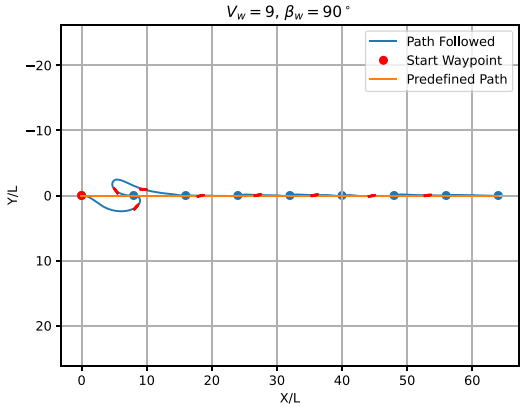
Although the yaw rate r is present in the observation state vector, it does not have a reward associated with it. However, it has been found that excluding the yaw rate from the observation state affects the ship's trajectory significantly. Two agents are trained independently to highlight the effect of the yaw rate on the ability of the agent to track waypoints. One agent is trained with yaw rate included in the observation states, while the other agent is trained with yaw rate excluded from the observation states. All other hyperparameters are kept the same between the models and are the same as reported in Table 7. Fig. 24 shows the trajectories tracked by both agents for the same waypoints. It can be seen that the agent that has the yaw rate in the observation state performs better than the other agent. This indicates that the agent, without any knowledge of the dynamics of the vessel or any reward associated with yaw rate, is able to correlate yaw rate with rudder angle and uses this information in addition to the other observation states while selecting a rudder action. While this might seem trivial from the knowledge of the dynamics of the vessel, the ability of the agent to make this connection without any knowledge of the dynamics and without any reward associated with yaw rate demonstrates the capability of the data driven methods to



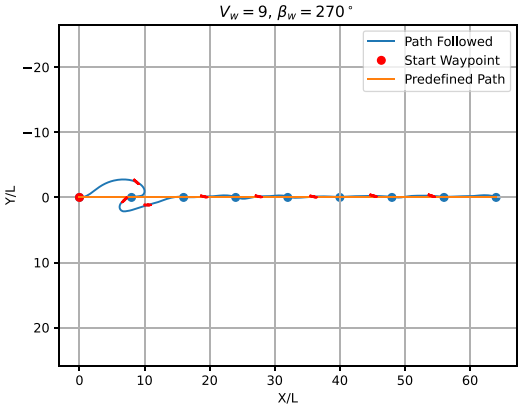
(a) Straight Line Path ($d_{RMSE} = 0.1227L$)



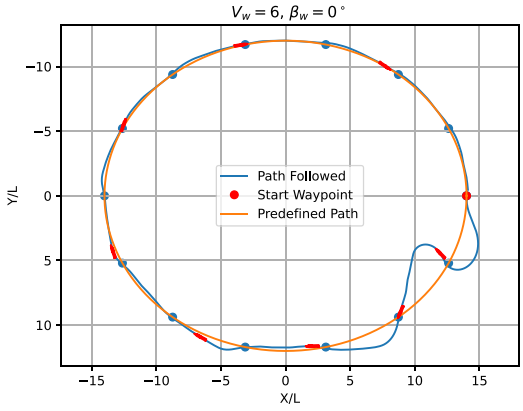
(b) Straight Line Path ($d_{RMSE} = 0.1615L$)



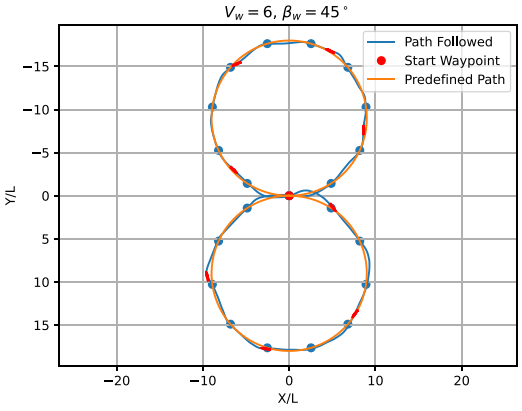
(c) Straight Line Path ($d_{RMSE} = 0.9110L$)



(d) Straight Line Path ($d_{RMSE} = 0.9938L$)



(e) Ellipse ($d_{RMSE} = 0.8456L$)



(f) Eight ($d_{RMSE} = 0.3941L$)

Fig. 23. Different cases for path following in presence of constant and uniform wind.

infer information from the environment beyond that explicitly expected by their reward structure.

In order to determine the ability of the RL agent to control the vessel in the presence of wind, several combinations of wind speed and wind direction are simulated for the straight line maneuver. In each case the vessel starts at $(0, 0)$ with a initial heading of $\psi = 0$ and tries to track a straight line along the x-axis. The ratio of the d_{RMSE} values in wind to the corresponding d_{RMSE} value in calm water is plotted in Fig. 25 for two different wind speeds and nine different wind directions. It can be seen from Fig. 25 that for low wind speed $V_w = 6U$, the RL agent is able to track the waypoints well for all wind directions and the error is no more than 20% as to compared to calm water conditions. When wind speed increases to $V_w = 9U$, wind direction of $\pm 90°$ experiences a greater deviation than observed in calm water. It can be observed from Fig. 23 that the maximum deviation occurs at the beginning of the episode where the wind is incident along the beam direction and results in a significant deviation before the vessel is able to returns to the path and follow it effectively.

It can also be seen from Fig. 23 that for straight line paths in the presence of a high wind speed $V_w = 9U$, a 10% deviation is observed in d_{RMSE} for a wind heading direction of $-90°$ as compared to $+90°$ when trying to follow a straight line. This can be attributed to the asymmetry

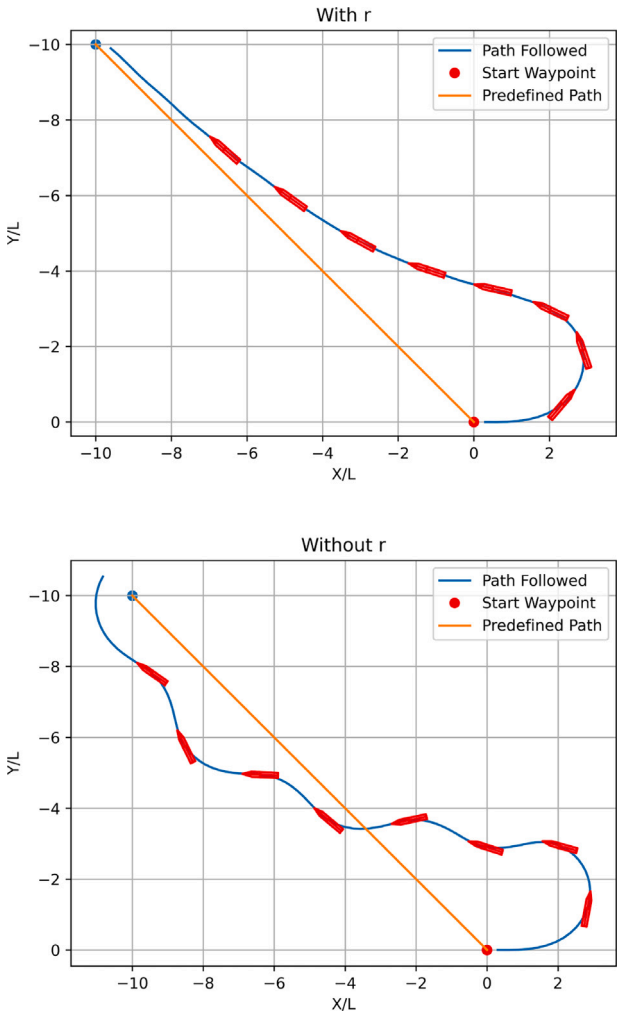


Fig. 24. Comparison of waypoint tracking with and without yaw rate(r) in the observation state.

in the rudder dynamics when port and starboard rudder angles are applied. Due to the vessel having a single screw propeller and rotating in one direction, the flow past the rudder is not symmetric. This effect is captured in the rudder dynamics (see Section 2.3). While this effect is not significant for moderate wind speeds of $V_w = 6$, it is significant for the higher wind speed $V_w = 9$ where the wind forces and moments are much stronger.

8.1. Comparison with a PD based controller

In this section the DQN agent is compared with a PD controller to compare their path following ability. This study uses a Proportional Derivative (PD) controller to ensure convergence of the vessel's heading ψ to the desired heading angle ψ_d, where the desired heading angle (ψ_d) is obtained from integral line of sight (ILOS) guidance law (Fossen, 2021). The error e is defined as the difference between the reference value ψ_d and the current heading angle ψ and is shown in (31). The derivative of the error with respect to time is given by (32)

$$e = \psi_d - \psi \tag{31}$$

$$\dot{e} = \dot{\psi}_d - \dot{\psi} = -r \tag{32}$$

where $\dot{\psi}_d$ is taken to be 0 for the control implementation. This is a fair approximation as the rate of change of desired heading angle $\dot{\psi}_d$

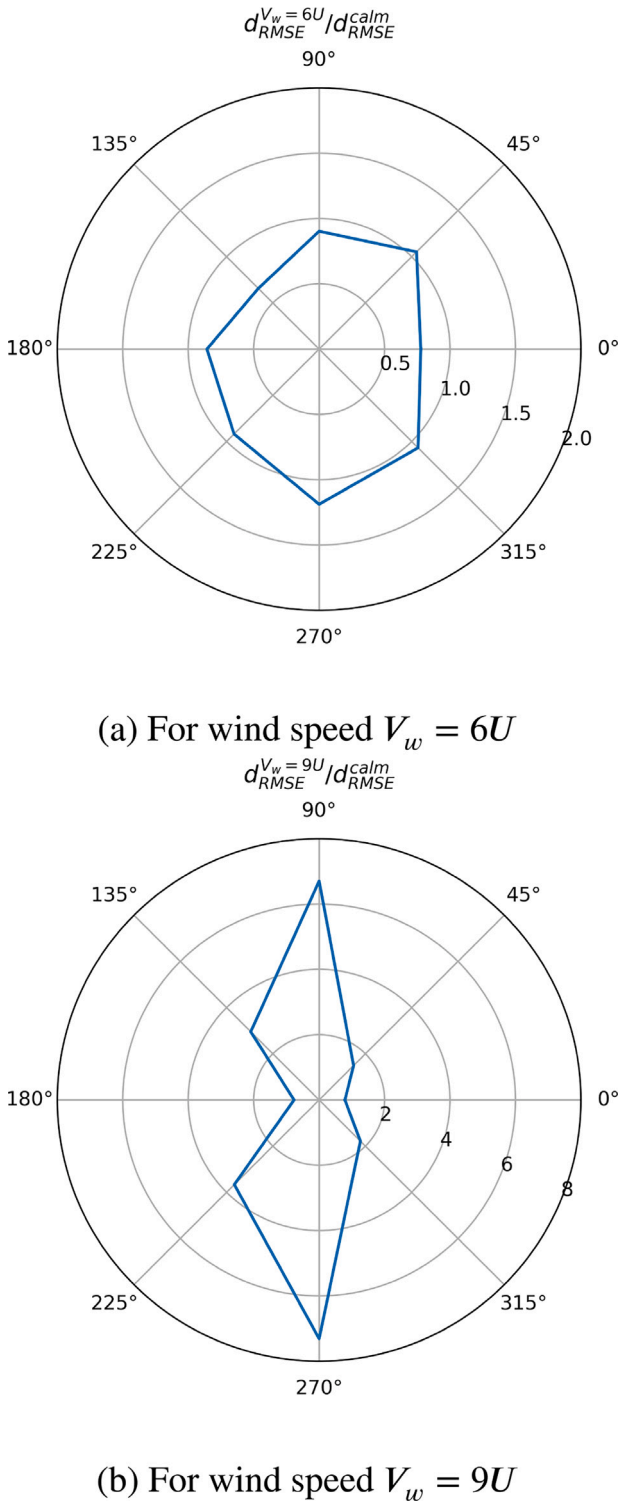
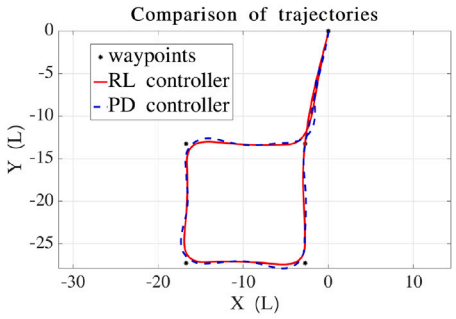


Fig. 25. Relative cross track error for straight line maneuver with wind speeds $V_w = 6, 9$.

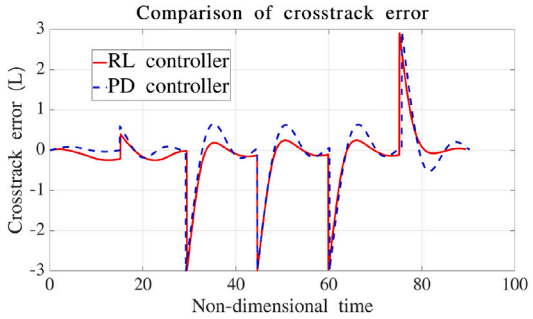
is governed by the outer guidance loop and varies slowly. Thus the proportional derivative (PD) control law can be expressed as

$$\delta_c = K_p e + K_d \dot{e} = K_p(\psi_d - \psi) - K_d r \tag{33}$$

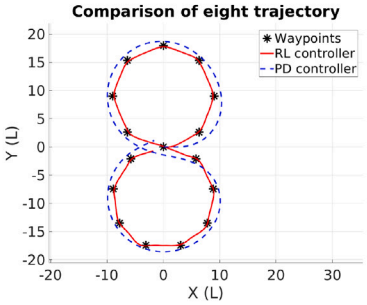
where δ_c is the commanded rudder angle. K_p is the proportional gain and K_d is the derivative gain of the controller. These controller gains



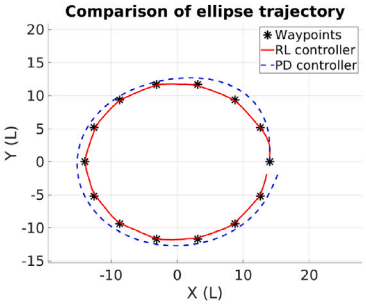
(a) Square Trajectory (RL: $d_{RMSE} = 0.6693L$ PD: $d_{RMSE} = 0.6959L$)



(b) Comparison of crosstrack error between RL controller and PD controller for a square trajectory



(c) Eight Trajectory (RL: $d_{RMSE} = 0.369L$ PD: $d_{RMSE} = 1.990L$)



(d) Ellipse Trajectory (RL: $d_{RMSE} = 0.246L$ PD: $d_{RMSE} = 1.019L$)

Fig. 26. Comparison of the path traversed by the model in PD controller and RL controller in simulation.

are tuned in such a way that deviation of vessel's trajectory from the desired square trajectory as shown in Fig. 26(a) is minimal. $K_d = 4.0$ and $K_p = 1.7$ were found to provide the best performance among the values investigated in the tuning process. It can be seen from Figs. 26(a) and 26(b) that the RL controller performance is similar to the PD controller with the root mean square cross track error being marginally (4%) smaller for the DRL controller. Further experiments were performed on an ellipse and eight trajectory and it was found that the root mean square cross track error for PD controller is 5 times the corresponding value from DRL controller in case of eight trajectory (Fig. 26(c)). This same value is about 4 times for the ellipse trajectory (Fig. 26(d)). Thus it can be seen that the traditional controller performance parallels the RL controller on paths for which the traditional controller gains are tuned. However, the same traditional controller is unable to perform as well as the RL controller on different paths for which its gains were not tuned.

9. Conclusion and future studies

This study has implemented a DRL based controller for path following of a ship using waypoints. A DQN agent was trained for this task and was provided rewards associated with cross-track error, course angle error and distance to goal waypoint. It was observed that even thought the agent did not receive any reward for yaw rate, it was a key observation state that was needed for the agent to effectively track the goal waypoint. The DQN agent demonstrated that it could successfully track destination waypoints in calm waters. It was also demonstrated that the agent could track complex paths like ellipses and "eight" maneuvers when discretized by waypoints.

In the presence of very strong winds up to six times the ship speed (about 250 km/hr for a full scale ship under consideration), the agent did not exceed more than 20% of the cross track errors observed in calm waters. In speeds of up to nine times the ship speed the cross-track errors did show a significant increase in beam wind conditions. It is also found that the DRL based controller can provide a performance as good as the traditional controllers in terms of path following.

In the future the existing DRL framework shall be improved to include both obstacle and collision avoidance and following of COLREGs. This will allow multiple objectives to be optimized at once and be codified into a policy that is computationally inexpensive to implement in real time. These constructs of multi-objective control will also be tested experimentally to better understand the nuances of the traditional and modern controllers. Further environmental disturbances such as waves and currents will also be explored in the future studies.

CRediT authorship contribution statement

Rohit Deraj: Conceptualization, Software, Writing – original draft. **R.S. Sanjeev Kumar:** Methodology, Software, Investigation, Writing – original draft. **Md Shadab Alam:** Methodology, Software, Validation, Investigation, Data curation, Writing – review & editing, Visualization. **Abhilash Somayajula:** Conceptualization, Validation, Investigation, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partially funded by the Science and Engineering Research Board (SERB) India - SERB Grant CRG/2020/003093 andNew Faculty Initiation Grant of IIT Madras. This work is also supported through the proposed Center of Excellence for Marine Autonomous Systems (CMAS), IIT Madras setup under the Institute of Eminence Scheme of Government of India.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/ Software available from tensorflow.org.

Baker, C., McCafferty, D., 2005. Accident database review of human element concerns: What do the results mean for classification? In: Proc. Int Conf.'Human Factors in Ship Design and Operation, RINA Feb. Citeseer.

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.

Cui, R., Yang, C., Li, Y., Sharma, S., 2017. Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. IEEE Trans. Syst. Man Cybern. Syst. 47 (6), 1019–1029.

Fossen, T.I., 1999. Guidance and control of ocean vehicles. (Doctors Thesis). University of Trondheim, Norway, Printed By John Wiley & Sons, Chichester, England, ISBN: 0 471 94113 1.

Fossen, T.I., 2021. Handbook of Marine Craft Hydrodynamics and Motion Control. John Wiley & Sons.

Guo, S., Zhang, X., Zheng, Y., Du, Y., 2020. An autonomous path planning model for unmanned ships based on deep reinforcement learning. Sensors 20 (2), 426.

Heiberg, A., Larsen, T.N., Meyer, E., Rasheed, A., San, O., Varagnolo, D., 2022. Risk-based implementation of COLREGs for autonomous surface vehicles using deep reinforcement learning. Neural Netw. 152, 17–33.

Layek, A., Vien, N.A., Chung, T., et al., 2017. Deep reinforcement learning algorithms for steering an underactuated ship. In: 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI, IEEE, pp. 602–607.

Lekkas, A.M., Fossen, T.I., 2012. A time-varying lookahead distance guidance law for path following. IFAC Proc. Vol. 45 (27), 398–403.

Martinsen, A.B., Lekkas, A.M., 2018. Curved path following with deep reinforcement learning: Results from three vessel models. In: OCEANS 2018 MTS/IEEE Charleston. IEEE, pp. 1–8.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Moreira, L., Fossen, T.I., Soares, C.G., 2007. Path following control system for a tanker ship model. Ocean Eng. 34 (14–15), 2074–2085.

Perera, L.P., Ferrari, V., Santos, F.P., Hinostroza, M.A., Guedes Soares, C., 2015. Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance. IEEE J. Ocean. Eng. 40 (2), 374–387.

Shen, H., Guo, C., 2016. Path-following control of underactuated ships using actor-critic reinforcement learning with MLP neural networks. In: 2016 Sixth International Conference on Information Science and Technology. ICIST, IEEE, pp. 317–321.

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.

Sivaraj, S., Rajendran, S., Prasad, L.P., 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.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

TensorFlow, 2022. TensorFlow Blog, URL https://blog.tensorflow.org/2022/05/whats-new-in-tensorflow-29.html/.

Vijay, A., Somayajula, A., 2022. Identification of hydrodynamic coefficients using support vector regression. In: OCEANS Conference 2022. Chennai.

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, pp. 127–137.

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.

Yasukawa, H., Yoshimura, Y., 2015. Introduction of MMG standard method for ship maneuvering predictions. J. Mar. Sci. Technol. 20 (1), 37–52.

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. Naval Archit. Ocean Eng. 14, 63–73.

Zhao, L., Roh, M.-I., 2019. COLREGs-compliant multiship collision avoidance based on deep reinforcement learning. Ocean Eng. 191, 106436.

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), 293–310.

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.

Zinage, S., Somayajula, A., 2021. Deep reinforcement learning based controller for active heave compensation. IFAC-PapersOnLine 54 (16), 161–167.