Journal of Marine Science and Technology

Volume 27 | Issue 4 Article 1

CONTROL METHOD FOR PATH FOLLOWING AND COLLISION AVOIDANCE OF AUTONOMOUS SHIP BASED ON DEEP REINFORCEMENT LEARNING

Luman Zhao
Post Doc., Department of Naval Architecture and Ocean Engineering, Seoul National University, Republic of Korea

Myung-Il Roh
Professor, Department of Naval Architecture and Ocean Engineering, Research Institute of Marine Systems Engineering, Seoul National University, Republic of Korea, miroh@snu.ac.kr

Sung-Jun Lee
Graduate Student, Department of Naval Architecture and Ocean Engineering, Seoul National University, Republic of Korea.

Recommended Citation
Zhao, Luman; Roh, Myung-Il; and Lee, Sung-Jun (2019) "CONTROL METHOD FOR PATH FOLLOWING AND COLLISION AVOIDANCE OF AUTONOMOUS SHIP BASED ON DEEP REINFORCEMENT LEARNING," *Journal of Marine Science and Technology*: Vol. 27: Iss. 4, Article 1.
DOI: 10.6119/JMST.201908_27(4).0001
Available at: https://jmstt.ntou.edu.tw/journal/vol27/iss4/1

CONTROL METHOD FOR PATH FOLLOWING AND COLLISION AVOIDANCE OF AUTONOMOUS SHIP BASED ON DEEP REINFORCEMENT LEARNING

Acknowledgements

This work was supported by Research Institute of Marine Systems Engineering of Seoul National University, Republic of Korea.

Journal of Marine Science and Technology, Vol. 27, No. 4, pp. 293-310 (2019)
DOI: 10.6119/JMST.201908_27(4).0001

CONTROL METHOD FOR PATH FOLLOWING AND COLLISION AVOIDANCE OF AUTONOMOUS SHIP BASED ON DEEP REINFORCEMENT LEARNING

Luman Zhao[1], Myung-Il Roh[2], and Sung-Jun Lee[3]

Key words: autonomous ship, path following, collision avoidance, line-of-sight guidance, reinforcement learning, proximal policy optimization.

ABSTRACT

Autonomous ships have received increasing attention in the maritime industry. The development of a real-time path following and collision avoidance system complying with the International Regulations for Preventing Collisions at Sea (COLREGs) is crucial to the development of autonomous ships. In this study, we proposed a novel deep reinforcement learning (RL) algorithm to improve the efficacy and efficiency of the path following and collision avoidance system. To verify the proposed algorithm, we conducted simulations of an autonomous ship under unknown environmental disturbances to adjust its heading in real time. A three-degree-of-freedom dynamic model for the autonomous ship was developed, and a line-of-sight (LOS) guidance system was used to guide the autonomous ship along a predefined path. Then, a proximal policy optimization (PPO) algorithm was implemented for the problem. We found that, after applying the advanced deep-RL method, an autonomous ship could learn the safest and most economical avoidance behavior through repeated trials. The simulation results showed that the proposed algorithm guaranteed collision avoidance with encountered moving ships while ensuring that the ship followed a predefined path. Additionally, the algorithm demonstrated that it could manage complex scenarios with various encountered ships in compliance with COLREGs, showing excellent adaptability to unknown complex environments.

Paper submitted 01/17/19; revised 03/22/19; accepted 05/17/19. Author for correspondence: Myung-Il Roh (e-mail address: miroh@snu.ac.kr).

[1] *Post Doc., Department of Naval Architecture and Ocean Engineering, Seoul National University, Republic of Korea.*
[2] *Professor, Department of Naval Architecture and Ocean Engineering, Research Institute of Marine Systems Engineering, Seoul National University, Republic of Korea.*
[3] *Graduate Student, Department of Naval Architecture and Ocean Engineering, Seoul National University, Republic of Korea.*

I. INTRODUCTION

1. Background

With the increasing demand for automation and self-governance of certain ship operations, autonomous systems with various applications have been explored in the maritime industry for many years. An autonomous system is one that can operate for long periods of time without human control. Such systems have the potential to bring huge changes to the maritime industry. Marine collisions can cause extreme harm to human lives and huge financial losses to ship owner; Additionally, they can lead to destructive environmental effects. Recent reports indicated that more than 80% of marine collision accidents were caused by human decision failure. Autonomous systems should be able to significantly reduce human-based errors (Chiang and Tapia, 2018).

In this study, we focus on the problems of following a predefined planar path and maneuvering to avoid collisions with encountered ships, constituting a part of autonomous ship development. With autonomous system development, path following and collision avoidance remain largely unsolved problems. Whereas several studies using various approaches have been extensively conducted in this field, most have not accurately take the moving obstacles, environmental disturbances, and COLREGs rules or they do not simultaneously solve the problems.

2. Related Studies

Traditionally, path following systems are functionally divided into three subsystems that must be implemented on board (Fossen, 2011): guidance, navigation, and control systems. To accomplish autonomous ship operation, one needs to know where the ship needs to go (guidance system), where it is (navigation system), and what to do to get there (control system). In the field of path following, the look-ahead line-of-sight (LOS) guidance law is an efficient method used to achieve convergence to a desired path. An overview of LOS guidance law for marine crafts can be found in the study of Fossen et al. (2003). Lekkas (2014) mainly focused on path planning in combination with the LOS guidance law to solve various types of problems. Oh and Sun (2010) proposed a model predictive control (MPC) method, which combined LOS guidance law with path following control for a surface vessel. An alternative adaptive control approach corresponding to the LOS guidance law was investigated by Fossen and Lekkas (2017).

In this study, we consider not only the path following problem, but also collision avoidance, which raises two problems: motion planning and corresponding control forces computation.

Most studies have only focused on motion planning. Motion planning for collision avoidance aims to find an admissible collision-free path between the initial and goal configurations, given environmental conditions with obstacles, and initial and goal configurations. Motion planning encompasses a wide range of algorithms, which can be divided into two categories: local and global methods. Local methods, such as dynamic window (DW) methods, only consider solutions optimal at the current time step, whereas global methods consider the full configuration space. Examples of global methods are A*, rapidly-exploring random trees (RRTs), and hybrid-state A*. To increase the path optimality and reduce unpredictability with the A* algorithm, a global motion planning methods can be used to guide the RRTs. Thus, Loe (2008) proposed a hybrid approach with the A* guided RRTs for global motion planning and the DW algorithm for local collision avoidance. Kuwata et al. (2014) and Stenersen (2015) utilized the velocity obstacle (VO) algorithm as a collision avoidance strategy for a surface vessel to determine safe velocity ranges for avoiding motion obstacles. A proportion differentiation controller was used to complete several scenarios under COLREGs requirements. Whereas the VO approach has the advantage of guaranteeing safe navigation, the reactive actions of the encountered vessel are neglected. Hence, the path generated by this approach may be limited in practice. To address this issue, the research conducted by Zhao et al. (2016) presented a collision avoidance strategy based on the optimal reciprocal collision avoidance (ORCA) algorithm, which is an extensional formulation of the VO concept. Their work revealed that the ORCA algorithm had better performance than VO. However, environmental conditions were considered. Chiang and Tapia (2018) proposed an RRT-based motion planning method for collision avoidance system on an autonomous surface vessel with COLREGs compliance. However, this method did not consider the environmental disturbances caused by waves and ocean currents.

After generating a collision-free path, the next step adopts a control system. There have been several studies on analytic controls and reinforcement learning (RL) methods for collision avoidance. MPC, a popular analytic control method, can compute an optimal trajectory based on obstacles' motion prediction, taking uncertainty into account. It considers the dynamic model as a cost function and constraints in an optimization problem (Johansen et al., 2016). Similar research (Hagen et al., 2018) related to collision avoidance used the MPC method to comply with COLREGs. However, there were two significant drawbacks of the MPC formulation: exorbitant online computational requirements and an inability to consider the uncertainties in the optimal control calculation.

Ernst et al. (2009) compared the MPC method to the Q iteration-based RL method. Simulation results showed that MPC was slightly less robust than RL from the numerical perspective, but had a slight advantage in terms of accuracy. Whereas analytic control methods have shown acceptable performance in certain applications, the performance of these methods is often limited, because the dynamic systems can be too complicated to be properly modeled in practical applications. Moreover, the rapid development of artificial intelligence has spurred great interest in various task autonomous. Path following and collision avoidance for autonomous ships is one of those tasks. Instead of designing the collision-free path and control systems separately, several approaches have used RL to model the complex interactions between the ship and encountered ships. RL has excellent capacity to adapt complex systems and is capable of self-learning, which provides the researcher with powerful algorithms to handle extremely complex systems under an unknown environment. Q-network, a value-based RL method for the collision avoidance system, has been developed by Cheng and Zhang (2018). However, their research only focused on static obstacles and did not consider environmental disturbances.

In this study, we utilize a policy-based RL method for motion planning combined with a control system for an autonomous ship operating in an unknown ocean environment considering encountered ships with respect to COLREGs compliancy. The advantages of the proposed algorithm are that it is extensible and easy to operate in terms of various environmental condition and COLREGs regulations compliancy. Extant studies rarely combine path following with collision avoidance of moving obstacles. Thus, we particularly focus on the problem of integrating path following with collision avoidance for an autonomous ship.

The remainder of this paper is organized as follows. In section 2, the kinematic and kinetic models of an autonomous ship are presented, and environmental disturbances are considered in the simulation model. Section 3 introduces the control algorithm design using the RL method. The implementation of the path following and collision avoidance system is shown in Section 4. Section 5 presents the application of the proposed RL control algorithm and simulation results in detail. Finally, the last section concludes this paper.

II. SYSTEM FORMULATION

1. Modeling of Autonomous Ship

In this study, a simplified three-degree-of-freedom (3-DOF) vessel dynamic model is used to describe the autonomous ship motions in the horizontal plane (i.e., surge, sway, and yaw) (Fossen, 2011). The rigid body kinematic equation is

$$\dot{\eta} = \mathbf{R}(\psi)\nu, \tag{1}$$

where $\eta = [x, y, \psi]^T$ represents the earth-fixed position and heading angle, $\nu = [u, v, r]^T$ represents the vessel-fixed velocities, and $\boldsymbol{R}(\psi)$ is the rotation matrix from the earth-fixed frame to the vessel-fixed frame. With the ship speed, $V =$

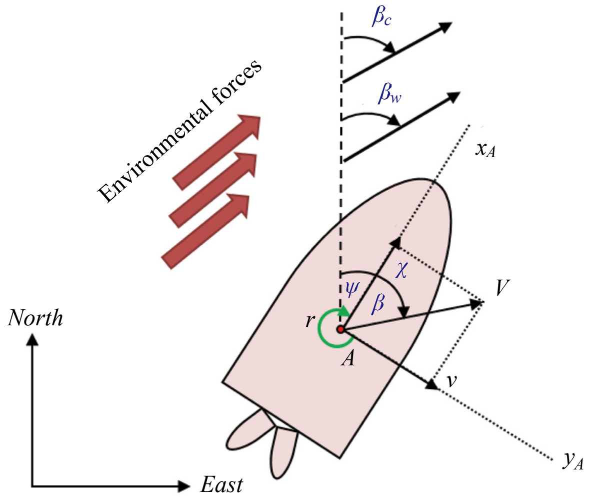


Fig. 1. Schematic representation of the ship kinematic variables.

$\sqrt{(u^2+v^2)}$, we define the course angle, $\chi = \psi + \beta$, and the sideslip angle, $\beta = \arcsin(v/V)$, as illustrated in Fig. 1. Note that the heading angle and course angle are equal when there is no sideslip. The dynamic equation for the autonomous ship can be expressed in the following form:

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} = \boldsymbol{\tau} + \boldsymbol{\tau}_{environmental\ forces}, \tag{2}$$

where $\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A$ is the mass matrix comprising the rigid-body mass and hydrodynamic added mass. $\mathbf{C}(v) = \mathbf{C}_{RB} + \mathbf{C}_A(v)$ is the matrix comprising the rigid-body and hydrodynamic Coriolis and centripetal matrices. The rigid-body mass matrix, $\mathbf{M}_{RB}$, and the rigid-body Coriolis and the centripetal matrix, $\mathbf{C}_{RB}$, have the following form:

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix}, \mathbf{C}_{RB} = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix}, \tag{3}$$

where m is the ship's mass, and I_z denotes the moment of inertia about the z-axis. The hydrodynamic added mass, $\mathbf{M}_A$, and hydrodynamic Coriolis and centripetal matrix, $\mathbf{C}_A(v)$, are expressed as

$$\mathbf{M}_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix}, \mathbf{C}_A(v) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u & 0 \end{bmatrix}. \tag{4}$$

These matrices contain the constant maneuvering coefficients of the ship.

$\mathbf{D}(v)$ is the nonlinear damping matrix, which can be defined as a sum of linear and nonlinear damping, $\mathbf{D}(v) = \mathbf{D}_L + \mathbf{D}_{DL}(v)$, where X_u, Y_v, Y_r, N_v, N_r, $X_{|u|u}$, $Y_{|v|v}$, $Y_{|v|r}$, $N_{|v|v}$, and $N_{|v|r}$ are maneuvering coefficients defined using the Society of Naval Architects and Marine Engineers notation.

$$\mathbf{D}_L = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix},$$
$$\mathbf{D}_{NL}(\mathbf{v}) = \begin{bmatrix} -X_{|u|u}|u| & 0 & 0 \\ 0 & -Y_{|v|v}|v| & -Y_{|v|r}|v| \\ 0 & -N_{|v|v}|v| & -N_{|v|r}|v| \end{bmatrix} \tag{5}$$

We assume that the ship has only one control input (i.e., rudder angle δ) and maintains constant propeller speed. The control force, $\boldsymbol{\tau}$, takes the following form:

$$\boldsymbol{\tau} = \begin{bmatrix} X_\delta \delta \\ Y_\delta \delta \\ N_\delta \delta \end{bmatrix}, \tag{6}$$

where X_δ, Y_δ, and N_δ are the rudder coefficients associated with the surge, sway force, and yaw moment, respectively. $\boldsymbol{\tau}_{environmentalforces}$ refers to the sum of environmental forces.

2. Environmental Forces

Environmental forces act upon a ship, affecting its motion. In this study, environmental forces are represented by three components: wind, current, and wave forces. The longitudinal wind force, F_{x_wind}, lateral wind force, F_{y_wind}, and wind moment, M_{z_wind}, can be computed as follows (Journee and Massie, 2000):

$$\begin{aligned} F_{x\_wind} &= \frac{1}{2}\rho_{air} A_T C_{wx}(\alpha_{rw}) V_{rw}^2 \\ F_{y\_wind} &= \frac{1}{2}\rho_{air} A_L C_{wy}(\alpha_{rw}) V_{rw}^2 \quad , \\ M_{z\_wind} &= \frac{1}{2}\rho_{air} A_L L C_{wN}(\alpha_{rw}) V_{rw}^2 \end{aligned} \tag{7}$$

where ρ_{air} is the density of air, A_T and A_L are the transverse and lateral projected wind area, respectively. L is the length of the ship. The wind speed and direction determine the longitudinal and lateral wind forces and the yawing moment on the ship. The wind load coefficients, C_{wx}, C_{wy}, and C_{wN} are parameterized in terms of relative wind direction.

The relative wind direction, α_{rw}, and speed, V_{rw}, are defined by the wind direction, β_w, and wind speed, V_w, as follows:

$$\begin{aligned} \alpha_{rw} &= \psi - \beta_w \\ V_{rw} &= \sqrt{u^2_{rw} + v^2_{rw}} \end{aligned}, \tag{8}$$

where the components of relative velocity in the x and y di-

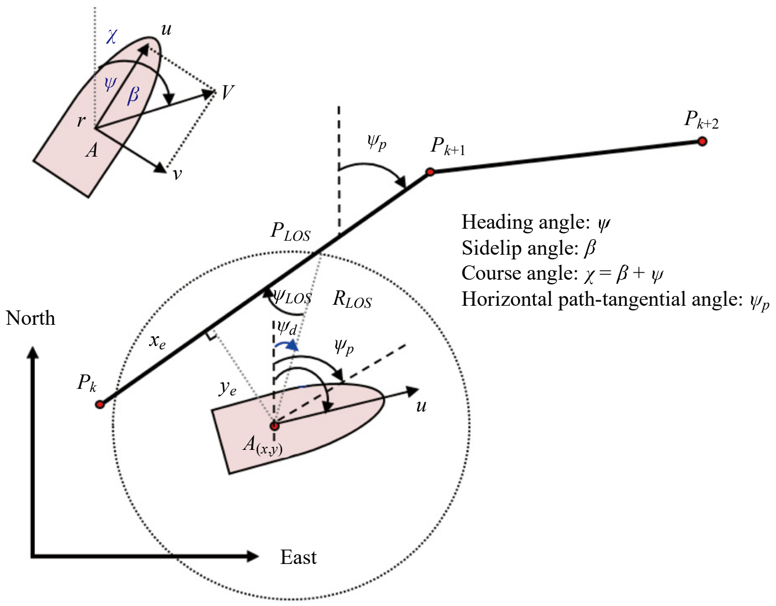


Fig. 2. Diagram of LOS guidance geometry for a straight line.

rections are

$$\begin{aligned} u_{rw} &= u - u_w = u - V_w \cos\alpha_{rw} \\ v_{rw} &= v - v_w = v - V_w \sin\alpha_{rw} \end{aligned}. \tag{9}$$

Similarly, based on the velocity vector synthesis method, the relative current velocity exerted by the current on the ship can be calculated from

$$\begin{aligned} u_{rc} &= u - u_c = u - V_c \cos(\psi - \beta_c) \\ v_{rc} &= v - v_c = v - V_c \sin(\psi - \beta_c) \end{aligned}, \tag{10}$$

where β_c is the current direction, and V_c is the speed of the ocean current.

The influence of wave interference is mainly divided into first- and second-order wave forces, which can be seen as a linear wave superimposed by a large number of regular waves of different frequencies and heights. In this study, we only consider the second-order wave drift force, which affects autonomous ship position and orientation. The wave force and moment can be calculated as follows:

$$\begin{aligned} F_{x\_wave} &= \frac{1}{2}\rho_{water}\xi_{wave}^2 gLC_{wavex}(\alpha_{rwave}) \\ F_{y\_wave} &= \frac{1}{2}\rho_{water}\xi_{wave}^2 gLC_{wavey}(\alpha_{rwave}) \\ M_{z\_wave} &= \frac{1}{2}\rho_{water}\xi_{wave}^2 gL^2 C_{waveN}(\alpha_{rwave}) \end{aligned}, \tag{11}$$

where ξ_{wave} is the wave height, and α_{rwave} is the relative wave direction. C_{wavex}, C_{wavey}, and C_{waveN} represent the coefficients of the second-order wave drift force and yawing moment, respectively. The dynamic equation of the autonomous ship motion can be rewritten using relative velocities as

$$\mathbf{M}\dot{\mathbf{v}}_{rc} + \mathbf{C}(\mathbf{v}_{rc})\mathbf{v}_{rc} + \mathbf{D}(\mathbf{v}_{rc})\mathbf{v}_{rc} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave}. \tag{12}$$

3. LOS Guidance System

Path following is the task of following a predefined path, usually specified in terms of waypoints. Each waypoint is defined using coordinates (x_k, y_k) for $k = 1$ and 2. For an autonomous ship, it means that the ship should pass through the waypoint (x_i, y_i) with the desired heading angle. A frequently used method for path following is LOS guidance. To avoid large drift when switching at the desired heading angle, and to provide a proper desired heading angle to the controller, the commanded LOS heading is fed through a reference model. The diagram of the LOS guidance system is shown in Fig. 2, where the LOS position, P_{LOS}, is the point along the path at which the vessel should be point. It is located somewhere along the straight line connecting the current waypoint, $P_k(x_k, y_k)$, and the next one, $P_{k+1}(x_{k+1}, y_{k+1})$. Let the ship's current position be located at the center of a circle with a radius of n times the ship length. The circle intersects the straight line between $P_k(x_k, y_k)$ and $P_{k+1}(x_{k+1}, y_{k+1})$ at two points, and P_{LOS} is selected as the point closest to the next waypoint, P_{k+1}.

Consider a straight line path defined by the two waypoints, $P_k(x_k, y_k)$ and $P_{k+1}(x_{k+1}, y_{k+1})$. Then, the path-tangential angle can be adjusted as follows:

$$\psi_p = \arctan(y_{k+1} - y_k, x_{k+1} - x_k). \tag{13}$$

Hence, for a ship located at the position (x, y), the alongtrack and cross-track errors can be computed as the orthogonal distance to the path-tangential reference frame defined by the point, P_k:

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} \cos\psi_p & \sin\psi_p \\ -\sin\psi_p & \cos\psi_p \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix}. \tag{14}$$

One of the control objectives for straight line path following becomes $\lim_{t\to\infty} y_e = 0$. Driving y_e to zero directs the velocity toward the intersection point, P_{LOS}, which corresponds to the desired direction. Based on the LOS guidance law, the desired course angle is separated into two parts:

$$\begin{aligned} \psi_d &= \psi_p + \psi_{LOS} \\ &= \psi_p + \arctan\left(\frac{-y_e}{\Delta}\right) \end{aligned}, \tag{15}$$

and

$$\psi_{LOS} = \arctan\left(\frac{-y_e}{\Delta}\right), \tag{16}$$

where Δ represents the look-ahead distance and takes values between 1.5 and 2.5 of the ship's length (Fossen, 2011). ψ_{LOS} ensures that the velocity is directed toward the point on the path.

In the presence of external disturbances, the heading angle error, ψ_e, becomes:

$$\psi_e = \psi_d - \beta - \psi. \tag{17}$$

Combining the above equations, the cross-track and heading angle errors can be explicitly stated by the following equation:

$$\begin{bmatrix} y_e \\ \psi_e \end{bmatrix} = \begin{bmatrix} -\sin\psi_p & \cos\psi_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_k \\ y - y_k \\ \psi_p + \psi_{LOS} - \beta - \psi \end{bmatrix}, \tag{18}$$

where y_e and ψ_e are the cross-track and heading angle error, respectively. The control objective of path following is to drive these two errors to zero.

III. CONTROLLER DESIGN FOR AUTONOMOUS SHIPS

In this section, we present the definitions and theoretical background of the controller design used in this study. The main objectives are to make an autonomous ship avoid collisions with encountered ships and to ensure that the position of the autonomous ship converges to and follows the predefined path after encountering other ships. The brain of the path following and collision avoidance system is the controller. The controller measures the process variables concerning the analysis module of the autonomous ship and gives control commands to actuators to correct errors between the process variables and desired values.

1. Problem Formulation

The path following and collision avoidance problem is defined in the context of the sequential decision-making problem by considering the controller configuration with the encountered ships. During the training, all the current process variables of the autonomous ship can be observed, and it can be evaluated whether the encountered ships are at a safe distance. Based on the observation space, self-play trials are conducted to determine the control strategy under various training processes. When the training process is completed, the autonomous ship is capable of automatically navigating along a predefined path and arriving at the destination while avoiding collisions with encountered ships under the commands of the controller.

At each time step t, the controller has access to the observation vector and computes the collision-free control command that drives the ship from the current position to the destination. The observation vector of the system is divided into two parts: s_t^T and s_t^O, $[s_t^O, s_t^T] \in s_t$, where s_t^O denotes the autonomous ship observation vector and s_t^T is the observation vector related to the encountered ships. Given the observation vector s_t, the autonomous ship computes a control command a_t sampled from a stochastic policy, $\pi_\theta(a_t|s_t)$, with the policy parameter θ. All of the terms used here will be redefined in the next section.

$$a_t \sim \pi_\theta(a_t \mid s_t) \tag{19}$$

Therefore, path following and collision avoidance problem formulated as a sequential decision-making problem. The objective of the controller design is to find an optimal policy.

2. Deep RL Setup

The sequential decision-making problem can be formulated as a Markov decision process in an RL framework, as illustrated in Fig. 3. The decision-maker (i.e., autonomous ship), which is called an agent, executes an action in the environment, and the environment, in turn, yields a new state and reward. The terms "agent", "environment", and "action" are used instead of "autonomous ship", "analysis module", and "control signal", respectively. More formally, the agent and environment interact at sequences of time steps, $t = 0, 1, 2, \ldots$ At each time step, the agent receives the state of the environment $s_t \in S$, where S is the set of possible states. It executes an action $a_t \in A(s_t)$ following a policy, where $A(s_t)$ is the set of actions available in state s_t. One time step later, the agent receives a numerical reward from the environment, $r_{t+1} \in R$, and finds itself in a new state s_{t+1}. The mapping from states to actions is called the policy (denoted as π_θ). $\pi_\theta(a|s)$ represents the probability that $a_t = a$ if $s_t = s$ (Sutton and Barto, 2015). The reward, R, is the feedback that informs the agent about the immediate quality of its actions. The goal of the agent is to maximize the sum of the

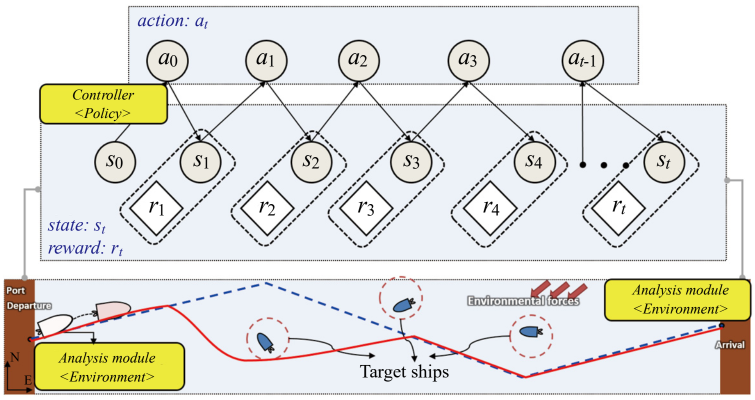


Fig. 3. Configuration of the RL framework for the path following and collision avoidance system.

rewards (return) received from the environment over the entire procedure. The return, $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma \in [0, 1]$ is the discount rate. The state value function, $V_\pi(s) = E[r_t|s_t = s]$ is the expected return for following policy π_θ from the state s_t. The state action value function, $Q_\pi(s, a) = E[r_t|s_t = s, a_t = a]$ is the expected return for selecting action a_t in state s_t and then following policy π_θ (Sutton and Barto, 2015).

As shown in Fig. 3, the policy can be formulated as a controller that observes states and applies actions to the agent (i.e., autonomous ship). The aim of the agent is to find an optimal policy, which can maximize the sum of the rewards (i.e., return) received during the interaction with the environment. Thus, the autonomous ship can follow its predefined path and avoid collisions with encountered ships.

1) State Space

Assume that the state space in this study can be fully observed. We define the state as the information the agent receives about the environment at a given time step. As mentioned in the previous section, state s_t consists of the state of the autonomous ship s_t^O, and the state related to the encountered ships s_t^T. It can be expressed as

$$s_t^O = \begin{bmatrix} y_e & \psi_e & \dot{\psi}_e & \chi_e & \dot{\chi}_e & \|P_{goal} - P\|_2 & \tilde{\phi} & \delta & \dot{\delta} & L \end{bmatrix} \tag{20}$$

and

$$s_t^T = \begin{bmatrix} P_{obstacle_i} & V_{obstacle_i} & \|P - P_{obstacle_i}\|_2 & \|\chi - \chi_{obstacle_i}\|_2 & l_i \end{bmatrix}. \tag{21}$$

The autonomous ship state s_t^O, comprises 10 elements as shown in Eq. (20). Here, y_e is the cross error, ψ_e is the heading angle error, $\dot{\psi}_e$ is the angular velocity of the heading angle error, and χ_e is the course angle error. These values can be calculated using Eq. (2). $||P_{goal} - P||_2$ represents the distance between the position of the autonomous ship and the destination. $\tilde{\varnothing}$ is the relative angle between the course angle of the autonomous ship and angle pointing to the destination from the ship. The rudder angle, δ, and rudder angular velocity, $\dot{\delta}$, are also considered as a part of the state space. As the length of the autonomous ship L may have a specific impact on the action space, it is included in the state space.

The observation vector of the encountered ships, s_t^T, contains the positions, $P_{obstaclei}$, and velocities, $V_{obstaclei}$ of the encountered ships in the local frame attached to the autonomous ship. The relative distances between the autonomous ship and encountered ships, $||P - P_{obstaclei}||_2$, and the relative angles between the autonomous ship and encountered ships, $||\chi - \chi_{obstaclei}||_2$, are also contained in the vector s_t^T. Here, χ is the course angle of the autonomous ship and $\chi_{obstaclei}$ is the course angle of the encountered ships. Additionally, the lengths of the encountered ships, l_i, are included in the state space. i represents the number of the encountered ships.

2) Action Space

As was mentioned, the state space comprising the autonomous ship inertial coordinates, whereas the action is related to rudder angle. We divide the permissible rudder angle, δ, into a set of three discrete values: $a \in \{-20, 0, 20\}$. Because an autonomous ship is an underactuated system, which has been formulated in the previous section, the control vector can be expressed as $\boldsymbol{\tau} = [X_\delta \delta \, Y_\delta \delta \, N_\delta \delta]$.

3) Reward Design

The reward function is computed as the sum of the rewards accumulated in each episode, where the reward is a measurement of action quality. The reward function can be specified to reward the agent for approaching its goal and penalize the agent for collision with encountered ships. The reward functions can be divided into two parts: the collision avoidance reward functions, related to the encountered ships, and the path following reward functions, designed to constrain the autonomous ship to follow a predesigned path.

First, the distance reward $R_{distance}$ is designed to guide the autonomous ship to achieve the destination. It can be expressed

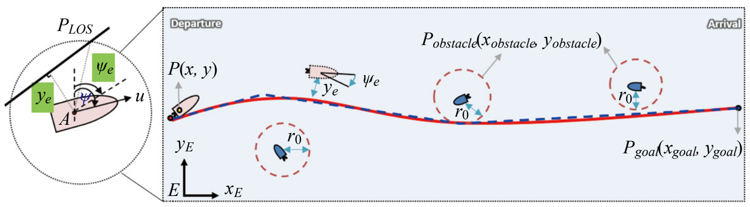


Fig. 4. Schematic representation of the variables for the reward functions.

mathematically as:

$$R_{distance} = -\lambda_{distance} \left\| P_{goal} - P \right\|_2 , \tag{22}$$

where P_{goal} and P are the position of the destination and current position of the autonomous ship, respectively (see Fig. 4). $\lambda_{distance}$ is a hyperparameter. When the ship approaches the destination, the more substantial distance reward value is imposed on the agent.

When the autonomous ship collides with the encountered ships in the range of a circle with radius r_0, it is penalized by the collision reward $r_{collision}$. The radius, r_0, can be regarded as the minimum passing distance between the autonomous ship and its encountered ships.

$$R_{collision} = \begin{cases} -r_{collision} & if \left\| P - P_{obstacle_i} \right\|_2 < r_0 \\ 0 & otherwise \end{cases} , \tag{23}$$

where $P_{obstacle_i}$ represents the current position of the encountered ships.

To avoid the drift phenomenon, the linear velocity of sway v has to be smaller than the surge velocity u. As a result, the drift reward function can be formulated as follows:

$$R_{drift} = \begin{cases} -r_{drift} & if \left| u \right| < \left| v \right| \\ 0 & otherwise \end{cases} , \tag{24}$$

where r_{drift} refers to the drift reward value in case of the drift phenomenon occurrence. As the autonomous ship has to avoid the encountered ships in compliance with COLREGs, the related reward function $R_{COLREGs}$ has to be added:

$$R_{COLREGs} = \begin{cases} r_{COLREGs} & if\ turn\ right \\ -r_{COLREGs} & otherwise \end{cases} . \tag{25}$$

The course angle error ψ_e and cross error y_e in Eq. (18) are considered in another two reward functions. To encourage the autonomous ship to follow the predesigned path, the course angle error and cross error must converge to zero. When calculating the course angle error reward within a small range $|\psi_e| < |\psi|$, we propose an exponential reward function to model it. If the heading angle error and the heading angular velocity error equal to zero, which means that there is no deviation between the autonomous ship and its path, the agent receives the maximum reward at the current time step. The cross error reward function is similar to the heading angle reward function.

$$R_{heading} = \begin{cases} \exp(-k_d((\psi_e)^2 + (\dot{\psi}_e)^2)) & if \left| \psi_e \right| < \left| \psi \right| \\ -r_{heading\_err} & otherwise \end{cases} \tag{26}$$

and

$$R_{cross} = \begin{cases} \exp(-k_c((y_e)^2 + (\dot{y}_e)^2)) & if \left| y_e \right| < \left| y \right| \\ -r_{cross\_err} & otherwise \end{cases} , \tag{27}$$

where k_d and k_c define the parameters of the exponential function, which relate to the convergence speed. $r_{heading\_err}$ and r_{cross_err} are positive values when the autonomous ship deviates from the path at a relatively large angle.

3. Network Architecture

The network comprising the critic network (value function) and policy network (policy function). The critic network is used to predict the state value function for each state, and the policy network is used to predict the action.

As shown in Fig. 5, to represent the policy network, we use a fully-connected (FC) multilayer perceptron with two hidden layers consisting of 64 and 32 hidden units with *tanh* nonlinearities outputting the probability over the action space. In the process of training, the state is transmitted to the neural network, and the agent selects and executes an action according to the policy with the highest probability. Training of the critic and policy networks (see Fig. 6) is performed by defining the surrogate loss functions for each network. Then, back-propagate gradients computed with the unified surrogate loss function are used to update the weights of the network. We refer to the network trained with this approach as the clipped proximal policy optimization (PPO) algorithm. Fig. 6 shows the network architecture. The observable state is fed into two FC layers; the outputs of critic network and policy network are the state value function (green) and action (orange).

4. Training Process

In this section, we focus on learning path following and collision avoidance policies, which perform robustly and effectively

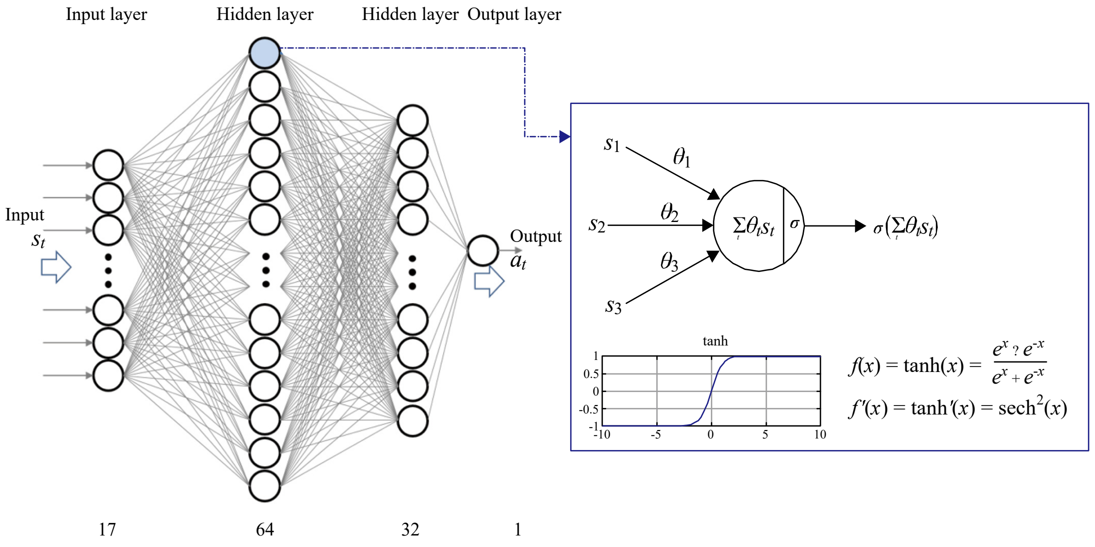


Fig. 5. Flow diagram of the neural network architecture.

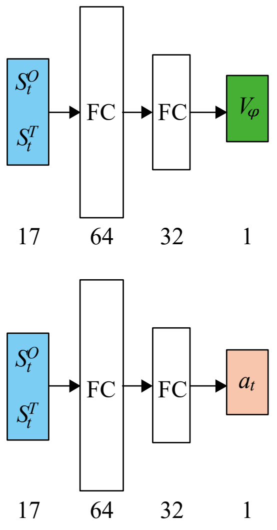


Fig. 6. Network architecture.

in various scenarios with encountered ships.

Policy gradient (PG) methods directly optimize the policy parameters θ by following the direction of the gradient of the expected return with respect to the policy parameters, which can be directly estimated from samples. However, traditional PG methods are sensitive to the choice of step size and have poor sample efficiency. To eliminate these disadvantages, a proximal policy optimization (PPO) method is proposed to constrain the step size of the policy update during training. PPO is an extension of the policy gradient method. It uses a clipped surrogate objective function as the policy network loss function, which is formulated as follows (Schulman et al., 2017):

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \tag{28}$$

and

$$L^{PPO}(\theta) = \hat{\mathrm{E}}\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right]. \tag{29}$$

In Eq. (28), $\pi_\theta(a_t|s_t)$ is the probability of the action under the current policy with the policy parameters θ; $\pi_{\theta old}(a_t|s_t)$ is the probability of the action under the previous policy. Thus, $r_t(\theta)$ is the ratio of the probabilities under the current and previous policies. It is greater than 1 when the action is more probable for the current policy, and it is between 0 and 1 when the action is less probable for the current policy than for the previous one. However, if $r_t(\theta)$ takes large values, the gradient steps may become overly large.

To deal with this, we can find that the surrogate loss function, $L^{PPO}(\theta)$, gains a penalty term so that $r_t(\theta)$ is clipped between $1 - \varepsilon$ and $1 + \varepsilon$ in Eq. (29). Therefore, it updates the policy conservatively by clipping the policy ratio within a small range around 1. $\ddot{E}_t$ denotes the empirical expectation over time steps, $\ddot{A}_t$ is the estimated advantage at time t, and ε is a hyperparameter, which is usually set to 0.1 or 0.2. Value targets are calculated based on the generalized advantage estimation (GAE) (Schulman et al., 2015). It is defined as the difference between the state action value function and state value function.

As shown in the pseudo code for the clipped PPO algorithm, at each iteration, the agent (i.e., autonomous ship) collects T time steps of the state values (where T is much less than the episode length) and runs the policy for T time steps. Then, we construct the surrogate loss function, $L^{PPO}(\theta)$, on these sampled episodes; the loss function is optimized with the Adam optimizer for $E\pi$ epochs. By taking the gradient ascent step on this loss with respect to the network parameters, the action is led to obtain a higher reward. The state value function $V\varphi(s_t)$, which is used

Table 1. Hyperparameters for the clipped PPO algorithm.

Parameter		Value
Discounted rate	γ	0.99
Lambda	λ	0.95
Time steps	$T_{\max}$	5120
The epoch of actor network	E_π	10
Clipping hyper parameter	ε	0.2
Learning rate	lr_θ	2e-
The epoch of critic network	E_V	10
Learning rate	lr_φ	1e-3

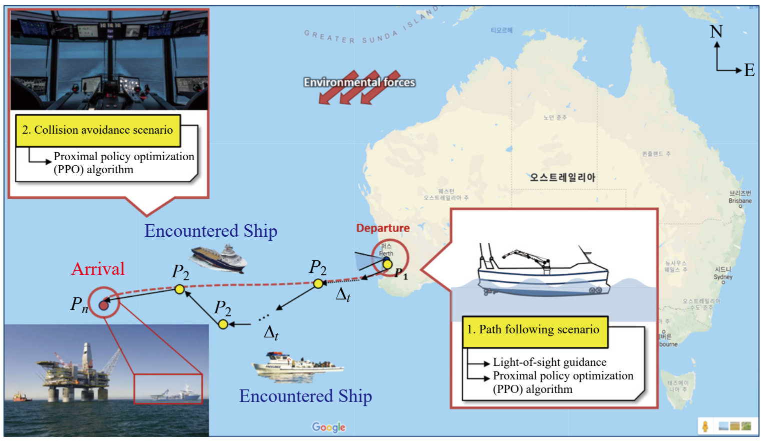


Fig. 7. Implementation of the path following and collision avoidance system.

as a baseline to estimate advantage $\ddot{A}_t$, is approximated by the critic network with the parameter φ. Then, we construct the mean squared error loss, $L^V(\varphi)$ for $V\varphi(s_t)$, and optimize it with the Adam optimizer for E_V epochs. We update $\pi_\theta(a_t|s_t)$ and $V\varphi(s_t)$ in actor and critic networks independently, and their parameters θ and φ are not shared, because we have found that using two separated networks leads to better results in practice.

For completeness, the algorithm for iteratively updating policy and value function is given below:

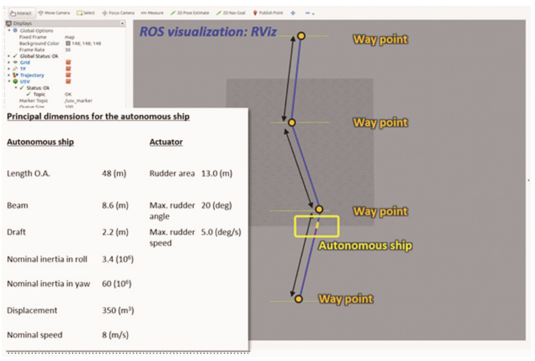
Pseudo code for the clipped PPO algorithm (adapted from (Long et al., 2017; Schulman et al., 2017))

```
1  Initialize policy network $\pi_\theta$ and value function $V\varphi(s_t)$ and set
   hyperparameters as shown in Table 1.
2  for iteration = 1, 2, …, do
3    Run policy $\pi_\theta$ for $T$ time steps, collecting states $s_t$, where $t \in [0, T]$
4    Estimate advantages using GAE $\hat{A}_t = \sum_{l=0}^{T} (\gamma\lambda)^l \delta_t$, where
     $\delta_t = r_t + \gamma V_\varnothing(s_{t+1}) - V_\varnothing(s_t)$
5    break, if $\sum_{l=0}^{N} T > T_{\max}$
6    $\pi_{old} \leftarrow \pi_\theta$
7    // Update policy
8    for $j = 0, 1, \ldots, E_\pi$ do
9      $r_t(\theta) = \dfrac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$
10     $L^{PPO}(\theta) = \hat{\mathrm{E}}\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right]$
11     Update $\theta$ with $lr_\theta$ by Adam (Kingma and Ba, 2014) w.r.t $L^{PPO}(\theta)$
12   end for
13   // Update value function
14   for $k = 1, 2, \ldots, E_V$ do
15     $L^V(\phi) = -\sum_{t=1}^{T}\left(\sum_{t'>t} r^{t'-t} r_{t'} - V_\varnothing(s_t)\right)^2$
16     Update $\varphi$ with $lr_\varphi$ by Adam w.r.t $L^{PPO}(\varphi)$
17   end for
18 end for
```

Table 1 presents the hyperparameters used in the simulations.

IV. IMPLEMENTATION OF PATH FOLLOWING AND COLLISION AVOIDANCE SYSTEM

In this study, we consider an autonomous ship assigned to converge to a predefined path specified by a path planner and to avoid collisions with encountered ships. Fig. 7 illustrates the overall implementation with two missions: predefined path



Principal dimensions for the autonomous ship

Autonomous ship		Actuator	
Length O.A.	48 (m)	Rudder area	13.0 (m)
Beam	8.6 (m)	Max. rudder angle	20 (deg)
Draft	2.2 (m)	Max. rudder speed	5.0 (deg/s)
Nominal inertia in roll	3.4 (10^6)		
Nominal inertia in yaw	60 (10^6)		
Displacement	350 (m^3)		
Nominal speed	8 (m/s)		

Fig. 8. Simulation setup for the first phase.

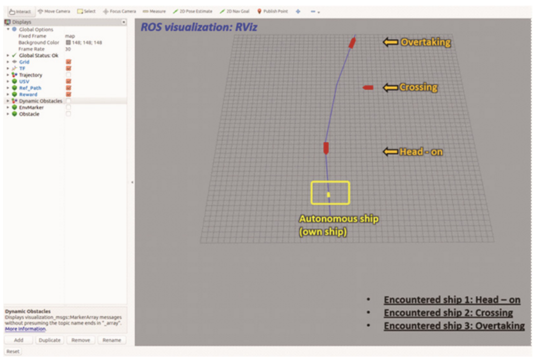

ROS visualization: RViz
Overtaking
Crossing
Head – on
Autonomous ship (own ship)
Encountered ship 1: Head – on
Encountered ship 2: Crossing
Encountered ship 3: Overtaking

Fig. 9. Simulation setup in the second phase.

following and collision avoidance with the encountered ships.

To visualize the simulation results, the 3D visualization tool RViz provided by Robot Operating System was used. It allows visualizing the simulated environment, including the autonomous ship, generated path, and encountered ships. The calculation module interacts with RViz to set the position and orientation of the autonomous ship.

In real-world implementations, the environment is often dynamic, changing unpredictably. To handle this, the path planner is required to generate various paths. We divided the training process into two phases, which accelerates the policy convergence and allowing agents to get a higher reward.

During the first phase, we train the autonomous ship to follow a randomly generated path without any encountered ships. It allows the autonomous ship to improve its training speed in the presence of the encountered ships. The generated path considered here is composed of a collection of randomly created waypoints, as shown in Fig. 8. During the training process, the positions and numbers of the waypoints are generated randomly. The principal dimensions of the autonomous ship with its actuator are shown on the left side of Fig. 8 (Perez and Mogens, 2002).

The objective of controlling an autonomous ship is to follow the randomly generated path without deviation. In this task, the rudder angle of the autonomous ship is limited to three choices: a positive angle of a fixed magnitude, a negative angle of the same magnitude, or zero. The reward functions described in Section 3.2.3, except for the collision reward function, are given on every time step until the destination (last waypoint) is reached. That means that the episode is finished. After completing the given training iteration, the optimal policy can be obtained.

When the autonomous ship achieves reliable performance, we save the trained policy and proceed to the second phase. Based on the trained neural network, the policy is further updated in the second phase, when the autonomous ship is assigned to follow the randomly generated path with the encountered ships. To simplify the problem, we assume that there are three encountered ships, representing different scenarios at each segment path: head-on scenario, crossing scenario, and overtaking scenario. While following the path, the autonomous ship encounters these three types of ships. Fig. 9 illustrates the training process setup in the second phase.

In each episode, the autonomous ship follows the path and avoids the encountered ships. Rudder angle is applied by the autonomous ship until the destination is reached. Then, the autonomous ship is restored to its initial position, and the new episode begins.

A diagram of encountered ships avoidance, as defined by COLREGs (COLREGs, 1972), is shown in Fig. 10. The autonomous ship is requested to be the give-way vessel, and the encountered ships are designed to be the stand-on vessels. If the distance between the give-way vessel and stand-on vessel is in a dangerous range, COLREGs are applied. As shown in Fig. 10(a), in the case of head-on scenario, the autonomous ship should change course to starboard and pass the encountered ship on its port side to avoid it; then, return to the original path after confirming safety. The diagram of the crossing scenario is shown in Fig. 10(b); the optimal strategy corresponds to a course offset toward starboard side until the encountered ship is passed at a safe distance on the autonomous ship port side. Finally, the overtaking scenario is shown in Fig. 10(c). The autonomous ship can either pass starboard or port of the encountered ship, depending on the COLREGs. In these scenarios, the encountered ships do not respect their responsibility to keep away.

V. SIMULATION RESULTS

The performance analysis of the path following and collision avoidance system was conducted using the proposed control algorithm for various environmental conditions. Two types of control objectives were selected. One forces the autonomous ship position to converge to the designed path by forcing the yaw angle to converge to the LOS angle. The other controls the autonomous ship to avoid the encountered ships with respect to COLREGs compliancy, while ensuring the following of the predefined path.

1. Path Following Scenario

1) Simulation Result of the Path Following Scenario

The proposed control algorithm was applied to path following in real environment using wind and current data to examine the effectiveness and practicality of the approach. The simulation

Table 2. Environmental conditions for the path following scenario.

Case	Num. of way points	Wave condition			Current direction (deg)	Wind direction (deg)	Training & testing		
		Direction (deg)	Height (m)	Period (sec)			Network model	Training iteration	Mean cross error (m)
1-1	4	-	-	-	-	-	A	0	40.2
1-2								150	1.5
1-3		90	1.0	10	90	90	A	-	7.3
1-4		45	1.0	10	45	45	A_1	20	1.7
1-5	6	0	1.0	10	0	0	A_1	-	2.1

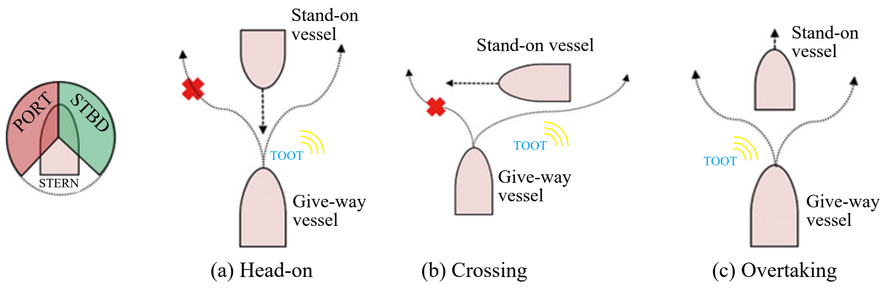


Fig. 10. Encounter ships avoidance, as defined by COLREGs.

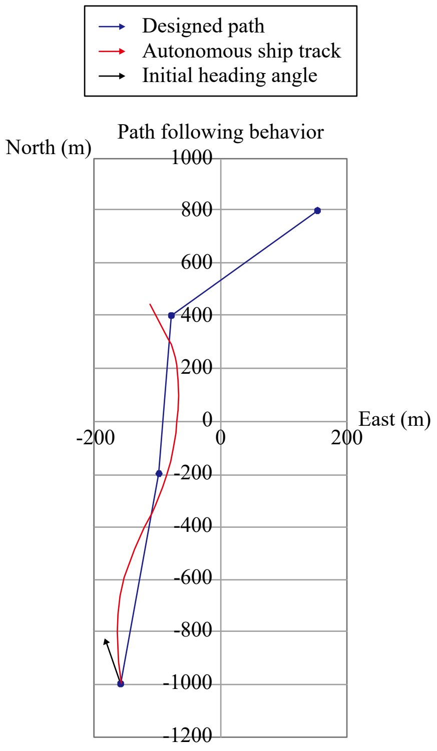


Fig. 11. Predefined path following of the autonomous ship in Case 1-1.

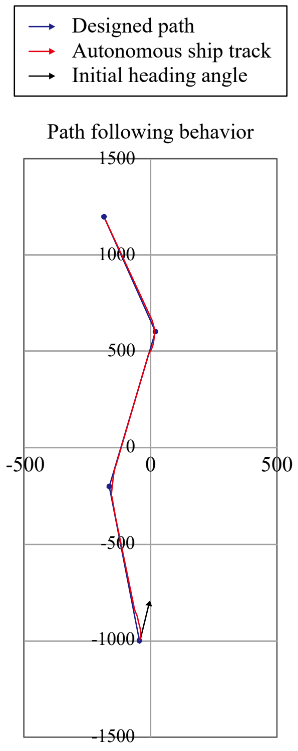


Fig. 12. Predefined path following of the autonomous ship in Case 1-2.

environment was implemented using the Python software package. The environment includes the dynamic model of the autonomous ship, randomly generated paths, and simulated wave, wind, and sea current disturbances. The integrated time step was set to $dt = 0.1$ s. Several simulation cases (see Table 2) considering a variety of environmental conditions were conducted. The following variables values were considered: wave velocity direction: {0°, 45°, 90°}; wind velocity direction: {0°, 45°, 90°}; current velocity direction: {0°, 45°, 90°}. The wind and sea current velocities' upper bounds were set to 30.0 m/s and 1.0

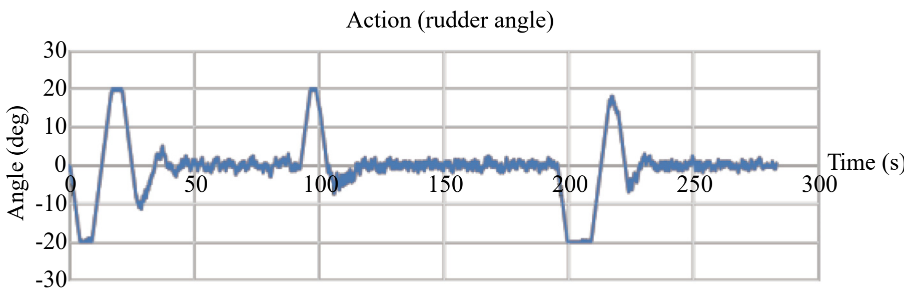


Fig. 13. Rudder angle of the autonomous ship as a function of time in Case 1-2.

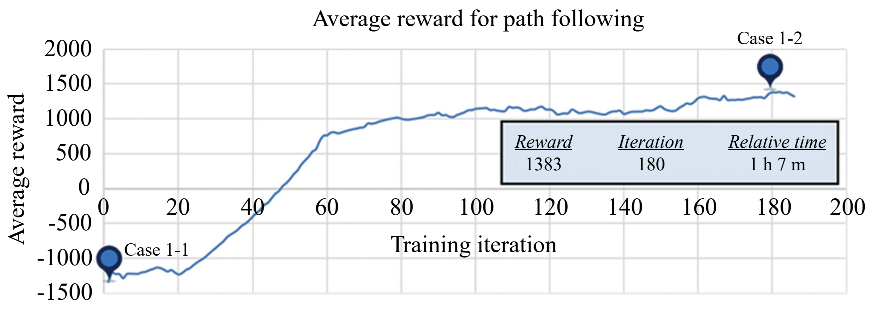


Fig. 14. Average reward during training.

m/s, respectively. The resultant disturbance forces and torque were collected in $\boldsymbol{\tau}_{wind} = [F_{x\_wind}, F_{y\_wind}, M_{z\_wind}]^T$ and $\boldsymbol{\tau}_{wave} = [F_{x\_wave}, F_{y\_wave}, M_{z\_wave}]^T$. The detailed description of the calculation can be found in Section 2.2. The related parameters of the environmental forces used in the following cases were obtained from the Oil Companies International Marine Forum (OCIMF).

At each training iteration, the agent exploits the policy to generate trajectories until the maximum of $T_{max} = 5120$ time steps is reached. Samples are then randomly selected from the collected data. The selected sampled mini-batch (= 64) is used to construct the surrogate loss function, optimized with the Adam optimizer for E_π ($E_\pi = 10$) epochs. Average reward is computed as the sum of the rewards accumulated in each episode, where the path following reward functions follow the rules defined in Section 3.2.3. An episode ends when the destination is reached, or the ship is too far from the path.

Figs. 11 and 12 illustrate the simulation results of the position of the autonomous ship following the predefined path in Cases 1-1 and 1-2. These simulation results represent the pre-training and post-training with an iteration number of 150. The network model, A, was trained without environmental forces. During this training, paths were generated randomly by connecting four way-points. The initial heading angle of the autonomous ship was also defined randomly. According to the simulation results, the capability of successfully following the path and reaching the destination is apparent.

The performance of the rudder angle in Case 1-2 is depicted in Fig. 13. It shows that, when passing through each waypoint, the rudder angle is set to maximum.

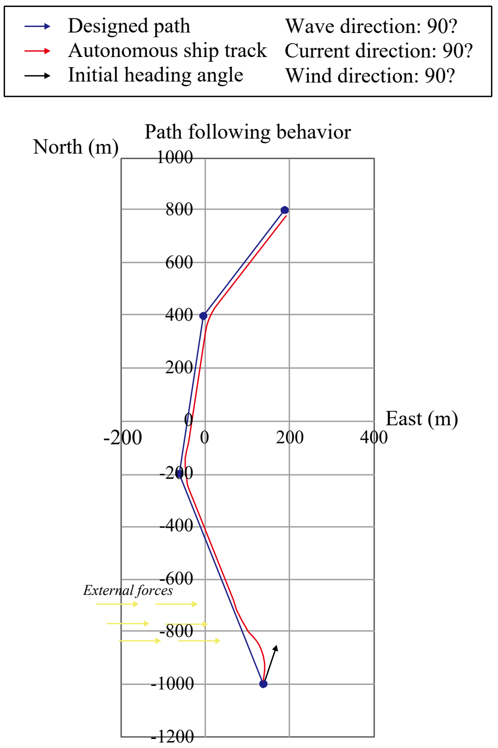


Fig. 15. Path following of the autonomous ship on a designed path in Case 1-3.

Fig. 14 shows the average total reward during the path following training. We can find that the reward increases smoothly. Case 1-1 and Case 1-2 are marked in Fig. 14.

However, the trained model A was used for the simulation of

Table 3. Environmental conditions for comparison scenario.

Case	Wave condition			Current direction (°)	Wind direction (°)	Proposed (PPO)		PID (fully tuned)	
	Direction (°)	Height (m)	Period (s)			Network model	Mean cross error (m)	Network model	Mean cross error (m)
2-1	-	-	-	-	-	A	1.14	C	3.26
2-2	45	1.0	10	45	45	A	6.11	C	9.37

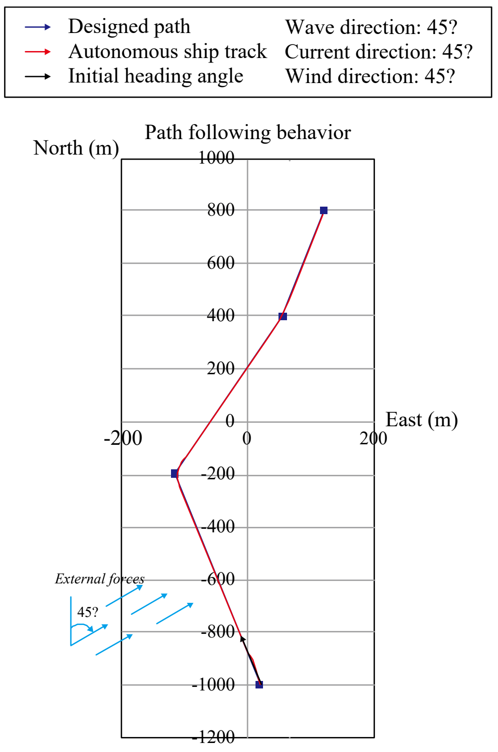


Fig. 16. Path following of the autonomous ship on a designed path in Case 1-4.

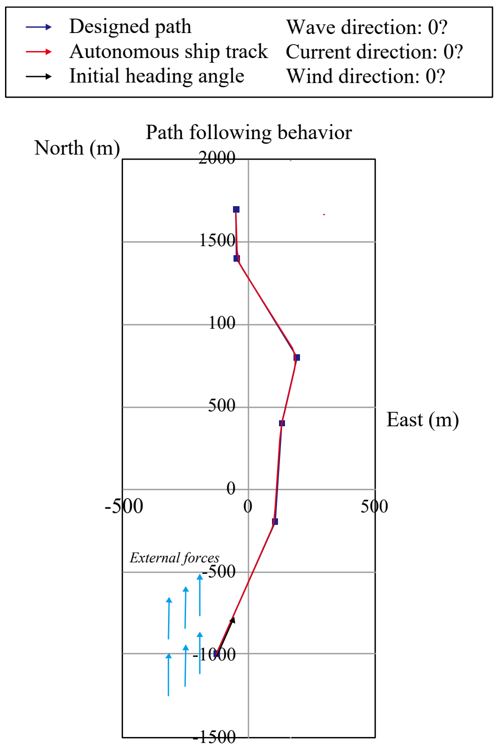


Fig. 17. Path following of the autonomous ship on a designed path in Case 1-5.

path following when the environmental forces, including waves, wind, and sea current, were taken into account. The simulation results of Case 1-3 are shown in Fig. 15. The autonomous ship deviates from the designed path regularly and fails to follow the path.

Based on the trained model A, we continuously trained the model under similar environmental conditions in Case 1-4. Thus, we obtained the trained model, A_1 after 20 iterations. Fig. 16 shows the simulation results using the network model, A_1 in Case 1-4; one can see that the autonomous ship follows the designed path successfully under the environmental forces.

The environmental conditions, such as wave amplitude, period, and direction, current and wind direction, can induce different motions of the autonomous ship. To evaluate the effectiveness of the model A_1 under various environmental forces, the following environmental conditions were set in Case 1-4: wave direction (45°); current direction (45°); and wind velocity direction (45°).

In Case 1-5, the wave, current, and wind directions are equal to 0 degrees; the training paths are more complicated than in the previous cases because two more waypoints were added (total of six waypoints). As shown in Fig. 17, the autonomous ship can successfully follow the designed path without training. According to these simulation results, we can conclude that one of the advantages of the proposed algorithm is its excellent performance in the unknown environmental disturbances.

2) *Comparison of the Proposed and Proportional-Integral-Derivative (PID) Algorithms*

To evaluate the performance of the proposed algorithm, the result of the proposed algorithm for the path following is compared with the PID algorithm. The mean cross errors of the following two cases are compared in Table 3.

To illustrate the performance of the proposed and PID algorithms, simulations with curved paths are sampled. The simulation result of the position of the autonomous ship is shown in Fig. 18. In Case 2-1, we can see the proposed algorithm using the network model A, which was trained in straight paths without the environmental forces, had good capabilities along with the curved path. Then we applied the model A to Case 2-2 that the

Table 4. Parameters for the encountered ships.

Encountered ship types		Details		
		Dimension (radius)	Initial position	Velocity
Ship 1	Head-on	$R \in [25, 150]$	P_H	V_H
Ship 2	Crossing	$R \in [25, 150]$	P_C	V_C
Ship 3	Overtaking	$R \in [25, 150]$	P_O	V_O

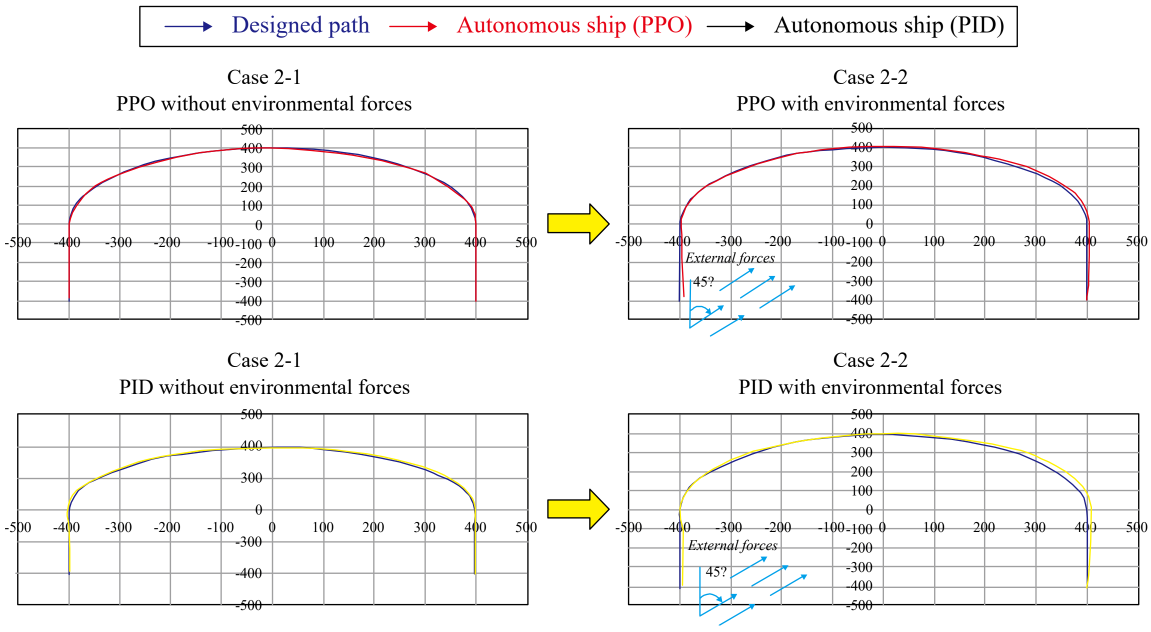


Fig. 18. Comparison of the proposed and PID algorithms for the curved-path following.

environmental forces were taken into account. The result shows the average mean cross error of 6.11 m. To highlight the performance of the proposed algorithm we performed the other simulation using the tuned PID algorithm under the same conditions. The PID algorithm was first tuned in Case 2-1 without the environmental forces and then applied to Case 2-2 with the environmental forces. The simulation result shows that the PID algorithm performed well in Case 2-1. However when we applied it to Case 2-2, it has a big deviation from the desired path. Furthermore, the results show an average mean cross error of 9.37 m.

In stable environments, the PID algorithm exhibits nearly ideal performance. When exposed to unknown dynamics, however, the PID algorithm can be far from optimal (Koch et al., 2018). Consequently, simulations demonstrate that the proposed algorithm outperforms the PID algorithm in the presence of unknown dynamics.

2. Collision Avoidance Scenario

Before proposing the collision avoidance formulation, certain assumptions should be made to simplify the training process. There are three encountered ships that represent different scenarios at each segment path: head-on scenario, crossing scenario, and overtaking scenario. The specifications of the encountered ships are listed in Table 4. Each encountered ship is regarded as a circle with radius R. R is randomly selected between 25 and 150 m. Additionally, it is assumed that, if the autonomous ship does not take avoidance actions, it collides with the encountered ship. Thus, the initial positions of the encountered ships should be well-designed. As all the encountered ships are designed to be the stand-on vessels, their velocities are set to be constant. During the training process, the head-on ship is located on the first segment path with a random velocity. When the autonomous ship successfully avoids the first head-on ship and passes the first turning waypoint, the crossing ship starts moving with a constant velocity V_C. Similarly, the overtaking ship starts moving with a relatively slow velocity, V_O, as soon as the autonomous ship passes the second turning waypoint.

To demonstrate COLREGs compliance, we trained the RL agent to avoid encounter ships using the clipped PPO algorithm. According to Section 3.2, the state input s_t conprises the state of the autonomous ship observed by itself s_t^O, and s_t^T, which is defined by the encountered ships. The output of the network is the rudder angle, and the reward function consists of the collision avoidance reward function and path following reward function. It is recalled that the autonomous ship collides with the target ships in the range of a circle with radius r_0. To calculate the radius r_0, we assume that in the head-on scenario,

Table 5. Environmental conditions for the collision avoidance scenario.

Case	Wave condition			Current direction (°)	Wind direction (°)	Training & testing		
	Direction (°)	Height (m)	Period (s)			Network model	Training iteration	Training time (h)
3-1	90	1.0	10	90	90	B	1,580	27
3-2	45	1.0	10	45	45	B	-	-

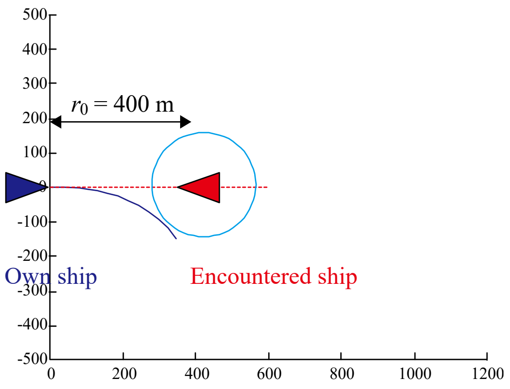


Fig. 19. Minimum distance between an autonomous ship and an encountered ship in head-on scenario.

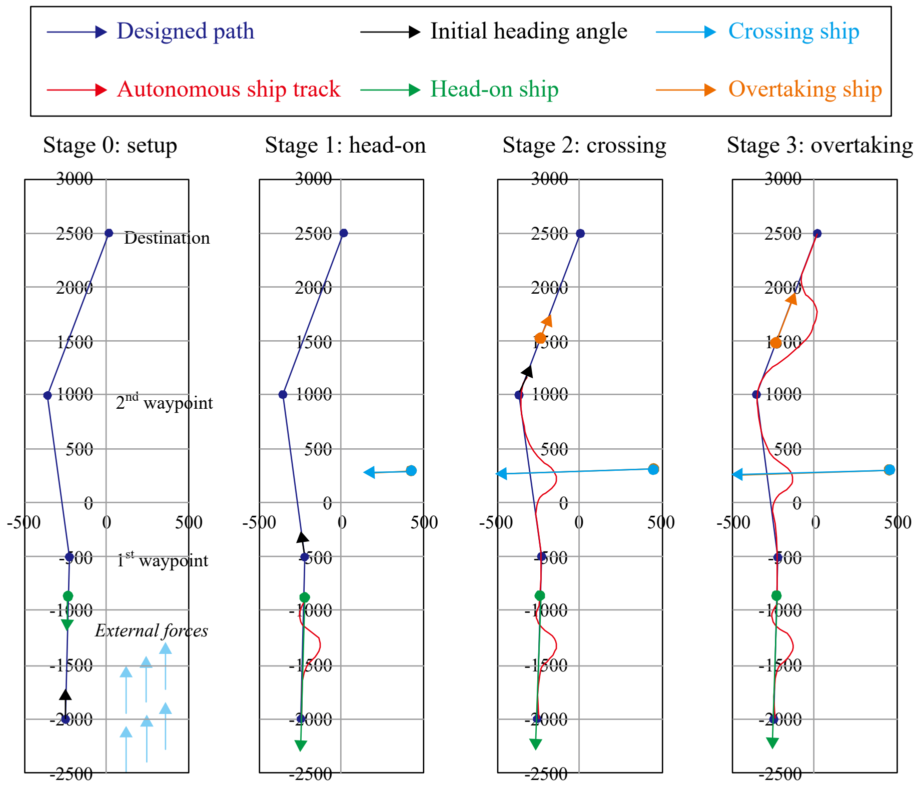


Fig. 20. Training process of collision avoidance with three encountered ships.

the autonomous ship (blue) continuously turns 90° with the maximum rudder angle (20°), to avoid the encountered ship (red). In this situation, we can obtain the minimum distance 400 m to guarantee safe navigation, as shown in Fig. 19.

The cases presented in Table 5 suggest that the RL agent is trained for the cases considering environmental forces.

Fig. 20 illustrates the training process of collision avoidance with three encountered ships. Training starts with a head-on

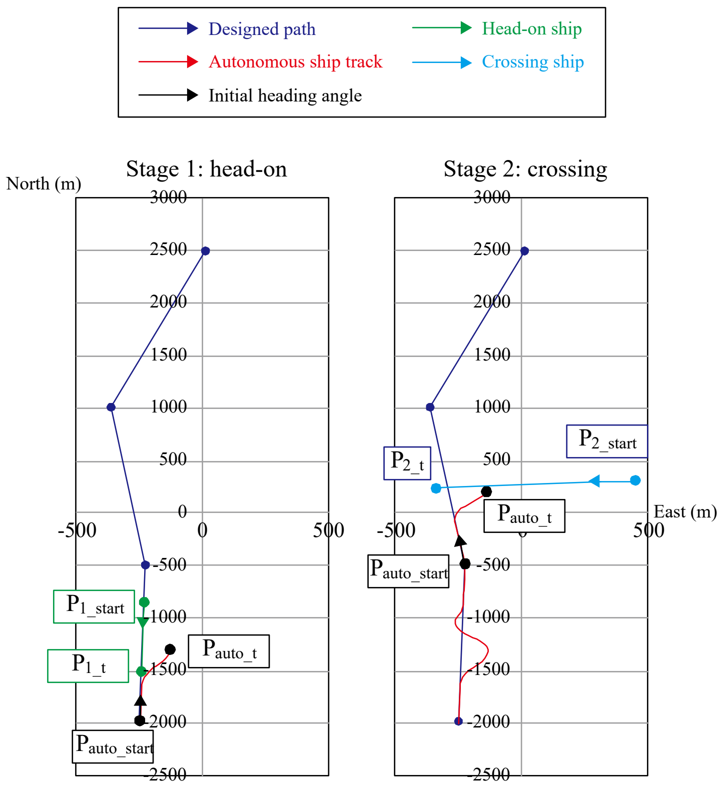


Fig. 21. Head-on and crossing simulations.

scenario, where the black and green arrows represent the initial heading angle of the autonomous ship and head-on ship, respectively. When in the head-on stage, the autonomous ship (red) passes the head-on vessel (green) on its port side; then, it returns back to the original path (blue). The first waypoint arrival of the autonomous ship triggers the crossing scenario. When the crossing ship (light blue arrow) starts approaching the path, the autonomous ship (red) makes a course change to avoid it. The course changes to starboard (in compliance with COLREGs), as the course change to port may increase the hazard. Stage 3 is the overtaking scenario: to overtake a slower ship (orange arrow), the autonomous ship changes its course to starboard to keep away from the slower ship. Finally, the autonomous ship reaches its destination.

During the head-on and crossing training processes, it is not clear whether the autonomous ship collides with the encountered ship from the above graphs because of the lack of the time co ordinate resolution. Therefore, we add Fig. 21 to illustrate the intermediate process when two ships meet at the same time. In the head-on scenario, when the head-on ship arrives at P_{1_t}, the autonomous ship maneuvers to avoid a collision and crosses abaft of the head-on ship. In the second case, when the crossing ship arrives from the starboard side, the autonomous ship changes course to starboard and passes with the crossing ship on her port side.

Fig. 22 illustrates the simulation result of collision avoidance with three types of encountered ships using the network model

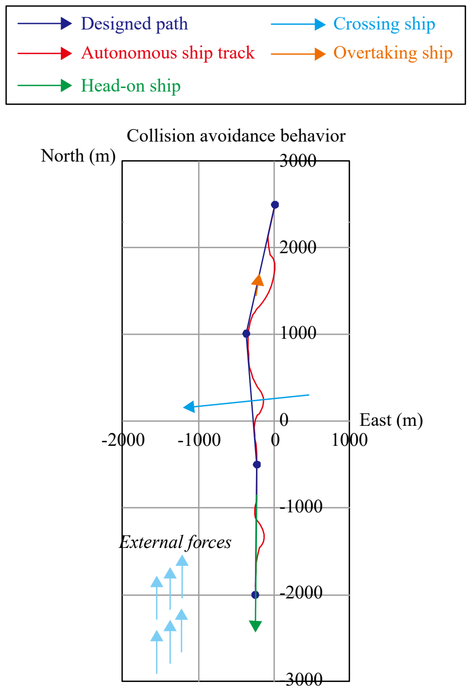


Fig. 22. Trajectories of the autonomous ship and three encountered ships in Case 3-1.

B. In Fig. 23, the corresponding time dependence of the rudder

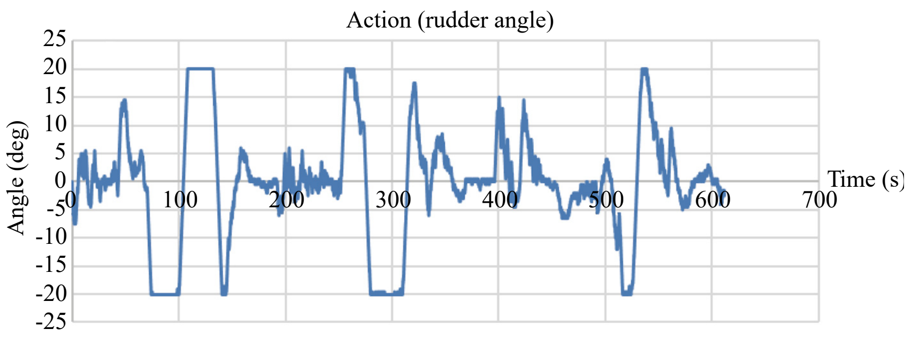


Fig. 23. Rudder angle of the autonomous ship as a function of time in Case 3-1.

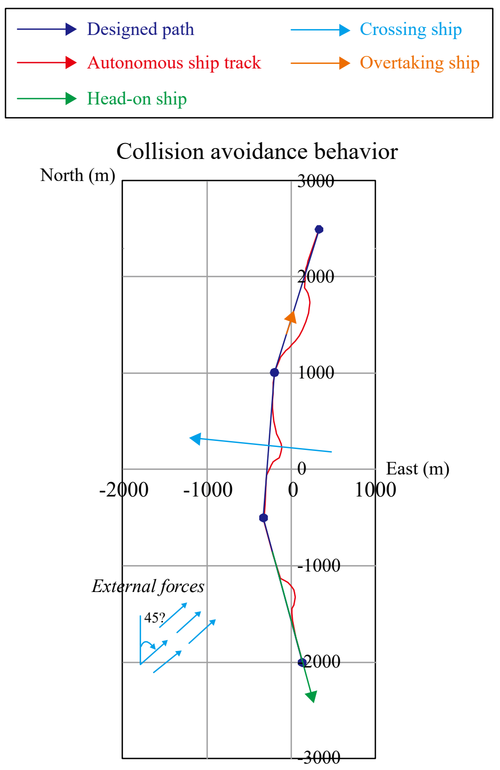


Fig. 24. Trajectories of the autonomous ship and three encountered ships in Case 3-2.

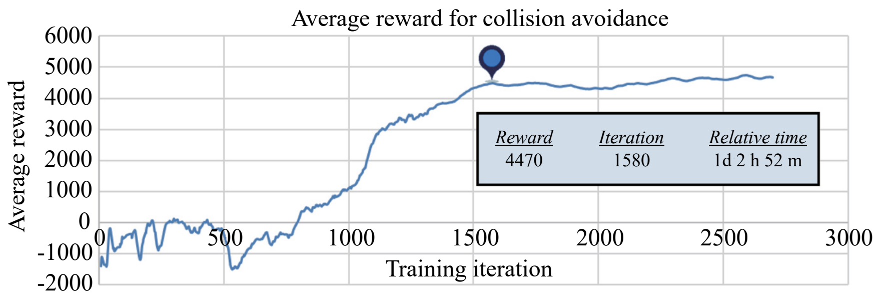


Fig. 25. Average reward for collision avoidance.

angle of the autonomous ship is presented. The control behavior corresponds to the course offset toward the starboard side until all the encountered ships pass at a safe distance on the autonomous ship's port side.

Therefore, we can conclude that the proposed algorithm for controlling the autonomous ship allows it to avoid various encountered ships. Additionally, based on the network model B, we performed simulations under different environmental conditions. Simulation results of Case 3-2 (see Fig. 24) show that the proposed algorithm is practical and can safely manage complex scenarios with various environmental disturbances.

Fig. 25 shows the average reward for collision avoidance in Case 3-1. The average reward converges to a maximum value after approximately 1580 iterations.

VI. CONCLUSION AND FUTURE WORKS

A real-time path following and collision avoidance system complying with COLREGs was developed with this study. We used the clipped PPO reinforcement learning algorithm to solve the problem. To provide a practical simulation environment, a 3-DOF dynamic model of the autonomous ship was developed.

First, we applied the path following algorithm for the autonomous ship by considering various environmental conditions. The mean cross error between the autonomous ship and predesigned path was approximately 1.77 m. The algorithm demonstrated that it could manage complex scenarios with excellent adaptability to the unknown complex environment. Moreover, we compared the proposed algorithm with a traditional model-free algorithm, PID, to evaluate the superiority of this study.

We simultaneously applied the path following and collision avoidance algorithms complying with COLREGs; the simulation results showed that the proposed algorithm guaranteed collision avoidance with moving encountered ships while ensuring the following a predefined path.

Future works will concentrate on implementing a multi-ship collision avoidance system. To achieve this, a multi-agent neural network will be developed. All the ships will be able to take actions to avoid each other. Additionally, optimization for the path planning problem will be added in future works. Furthermore, to improve the interpretability of the proposed algorithm, we will combine it with the analytical control method.

ACKNOWLEDGEMENTS

This work was supported by Research Institute of Marine Systems Engineering of Seoul National University, Republic of Korea.

REFERENCES

Chiang, H.-T. and L. Tapia (2018). COLREG-RRT: An RRT-Based COLREGS-Compliant motion planner for surface vehicle navigation. IEEE Robotics and Automation Letters 3(3), 2024-2031.

Cheng, Y. and W. D. Zhang (2018). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing 272, 63-73.

Ernst, D., M. Glavic, F. Capitanescu and L. Wehenkel (2009). Reinforcement learning versus model predictive control: a comparison on a power system problem. IEEE Transactions on Systems, Man, and Cybernetics-Part B 39(2), 517-529.

Fossen, T. I., M. Breivik and R. Skjetne (2003). Line-of-Sight path following of underactuated marine craft. Proceedings of the 6th IFAC MCMC 36(21), 211-216.

Fossen, T. I. (2011). Handbook of Marine Craft Hydrodynamics and Motion Control. John Wiley and Sons Ltd.

Fossen, T. I. and M. Lekkas (2017). Direct and indirect adaptive integral line-of-sight path-following controllers for marine craft exposed to ocean currents. International Journal of Adaptive control and signal processing 31, 445-463.

Hagen, I. B., D. K. M. Kufoalor, E. F. Brekke and T. A. Johansen (2018). MPC-based collision avoidance strategy for existing marine vessel guidance systems. 2018 IEEE International Conference on Robotics and Automation (ICRA), 7618-7623.

IMO (1972), International Regulations for Preventing Collisions at Sea (COLREGs), International Maritime Organization.

Johansen, T. A., T. Perez and A. Cristofaro (2016). Ship collision avoidance and COLREGS compliance using simulation-based control behaviour selection with predictive hazard assessment. IEEE Transactions on Intelligent Transportation Systems 17(12), 3407-3422.

Journee, J. M. J. and W. W. Massie (2000). Offshore Hydromechanics. Delft university of Technology.

Kingma, D. P. and J. Ba (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Koch, W., R. Mancuso, R. West and A. Bestavros (2019). Reinforcement learning for UAV attitude control. ACM Transactions on Cyber-Physical Systems 3(2), 22.

Kuwata, Y., M. T. Wolf, D. Zarzhitsky and T. L. Huntsberger (2014). Safe maritime autonomous navigation with COLREGS, using velocity obstacles. IEEE Journal of Oceanic Engineering 39(1), 110-119.

Lekkas, A. M. (2014). Guidance and path-planning systems for autonomous vehicles. Ph.D. Thesis, Norwegian University of Science and Technology, Norway.

Loe, Ø. A. G. (2008). Collision avoidance for unmanned surface vehicles. Master theses, Norwegian University of Science and Technology, Norway.

Long, P. X., T. X. Fan, X. Y. Liao, W. X. Liu, H. Zhang and J. Pan (2017). Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. arXiv preprint arXiv:1709.10082.

Oh, S. R. and J. Sun (2010). Path following of underactuated marine surface vessels using line-of-sight based model predictive control. Ocean Engineering 37, 289-295.

Perez, T. and B. Mogens (2002). Mathematical Ship Modeling for Control Applications. Technical Report.

Schulman, J., P. Moritz, S. Levine, M. I. Jordan and P. Abbeel (2015). High-dimensional Continuous Control using Generalized Advantage Estimation. arXiv preprint arXiv: 1506. 02438.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov (2017). Proximal Policy Optimization Algorithm. arXiv preprint arXiv: 1707. 06347.

Stenersen, T. (2015). Guidance System for Autonomous Surface Vehicles. Master Thesis, Norwegian University of Science and Technology, Norway.

Sutton, R. S. and A. G. Barto (2015). Reinforcement learning: an introduction. The MIT press.

Zhao, Y. X., W. Li and P. Shi (2016). A real-time collision avoidance learning system for unmanned surface vessels. Neurocomputing 182, 255-266.