Journal of Marine Science and Technology (2021) 26:509–524
https://doi.org/10.1007/s00773-020-00755-0

ORIGINAL ARTICLE

Automatic ship collision avoidance using deep reinforcement learning with LSTM in continuous action spaces

Ryohei Sawada[1] · Keiji Sato[1] · Takahiro Majima[1]

Received: 30 April 2020 / Accepted: 23 July 2020 / Published online: 3 August 2020

Abstract

This paper presents an automatic collision avoidance algorithm for ships using a deep reinforcement learning (DRL) in continuous action spaces. Obstacle zone by target (OZT) is used to compute an area where a collision will happen in the future based on dynamic information of ships. Agents of DRL detects the approach of multiple ships using a virtual sensor called the grid sensor. Agents learned collision avoidance maneuvering through Imazu problem, which is a scenario set of ship encounter situations. In this study, we propose a new approach for collision avoidance with a longer safe passing distance using DRL. We develop a novel method named inside OZT that expands OZT to improve the consistency of learning. We redesign the network using the long short-term memory (LSTM) cell and carried out training in continuous action spaces to train a model with longer safe distance than the previous study. The bow cross range in collision detection proposed in this paper is effective to COLREGs-compliant collision avoidance. The trained model has passed all scenarios of Imazu problem. The model is also validated by a test scenario which includes more ships than each scenario of Imazu problem.

Keywords Collision avoidance · Multiple ships · Reinforcement learning · OZT

1 Introduction

In recent years, there has been a lot of research and development on automated ships. The regulation of Maritime Automatic Surface Ship (MASS) is under discussion at the International Maritime Organization (IMO) [1]. For automation of maritime transportation, it is important to improve the safety of navigation. It is reported that collision accidents of ships were mainly caused by human errors such as[2]. By supporting human or automating operations, the number of collision accidents can be decreased.

Automatic collision avoidance has been studied for a long time, and a number of algorithms have been proposed [3]. In 1980s, Imazu and Koyama utilized a dynamic programming [4–6]. In this method, the ship's speed and heading angle are defined in a discrete action space, and collision avoidance is performed by selecting the optimal action with an evaluation function based on the International Regulations for Preventing Collisions at Sea (COLREGs) and rules of collision risk assessment. Kouzuki and Hasegawa developed an fuzzy controller for collision avoidance [7]. They defined collision risk (CR) calculated from the distance to a closest point of approach (DCPA) and time to a closest point of approach (TCPA). CR is used to determine the timing to steer for collision avoidance navigation. Hu et al. studied a COLREGs-compliant path planning approach using particle swarm optimization (PSO) [8]. Kuwata et al. proposed an automatic collision avoidance method using velocity obstacles (VO) [9]. In this method, a collision is prevented by calculating a safe heading based on VOs which is calculated from velocity vectors of an own ship and a target ship. As a method without explicit implementation of the COLREGs, the Nagasawa model, which uses an evaluation function defined on the steering space with rudder angle and speed, has also been studied [10, 11]. Woerner et al. implemented the rule-based algorithm based on the COLREGs for evaluation of autonomous vessels [12]. In recent years, since techniques of machine learning are developing rapidly, reinforcement learning, which is one of the machine learning, is begun to be applied to automatic collision avoidance. There are research which used Q-learning which is one of the reinforcement learning algorithms [13–15]. In the last few years, collision avoidance methods using DRL have also been

✉ Ryohei Sawada
sawada-r@m.mpat.go.jp

[1] National Maritime Research Institute, 6-38-1, Shinkawa, Mitaka, Tokyo 181-0004, Japan

proposed. Shen et al. [16] used Deep Q-Network (DQN) for collision avoidance of multiple ships. DQN is one of the DRL algorithms. They tested trained model for encounters with up to three target ships. One of the authors also proposed an automatic collision avoidance algorithm using proximal policy optimization (PPO) and a novel virtual sensor [17]. PPO is also one of the DRL algorithms and is used in not only playing video games but also controlling robots. The trained model can avoid all encounter situations of up to three target ships and arrive at a given waypoint in simulations.

In the previous study [17], one of the authors used obstacle zone by target (OZT) [18] for collision risk assessment. OZT represents an area where a collision will happen in the future based on dynamic information of ships. The radius of OZT is defined as the safe passing distance. The previous method was able to avoid collision in complex scenarios called Imazu problem [19] . Imazu problem is a scenario set including basic situations of 2-ship encounters and rather difficult situations of 3–4-ship encounters. There was a limit to the safe passing distance which could be set for learning using the previous method. In the previous study, the model learned collision avoidance maneuvering at a safe passing distance of 0.3 NM, but in actual ship operation, ships are required to have a longer safe passing distance from target ships. Besides, in some cases of Imazu problem, the own ship had passed the front of other ships. This kind of maneuver is dangerous. In fact, as three basic encounter situations shown in Fig. 1 stipulated in the COLREGs article, ships do not perform avoidance maneuvers across in the front of other ships.

In this paper, the maneuverability of ships used in simulation is not changed in comparison with the previous studies, but the safe passing distance of an OZT is set to a larger value to accommodate a wider range of sizes of ships and improve collision avoidance maneuver of new trained models. There is no standard way to determine the safe passing distance for OZT. On the other hand, several methods for detection of ship collision were proposed. One of them is a bumper model [20]. Figure 2 shows the bumper model. Bumper model is an area surrounding a ship and gives the minimum of safe passing distance between ships. Ships navigate to keep other ships out of their bumpers. The lengths $L1$ and $L2$ in Fig. 2 are usually taken as $6.4L$ and $1.6L$, respectively, where L is the ship's length. According to Inoue [21], for international ships, the lengths $L1$ and $L2$ of bumper model are taken about twice as long as as recent ships become larger and faster. Thus, in this paper, we determine the safe passing distance for ships up to around 300 m in length and the distance is set at 0.5 NM as about twice the distance of $1.6L$. This is that $L1 = 0.5$ NM, which corresponds to the length of the bumper model for a ship with exactly 289.375 m length.

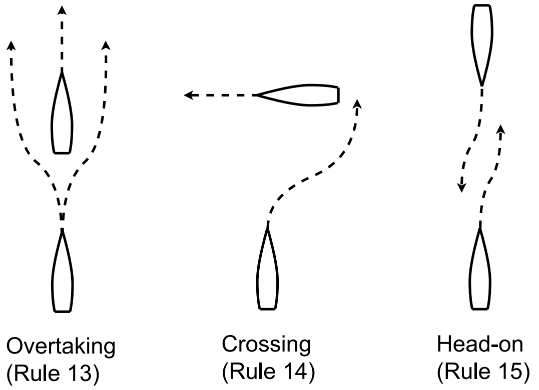
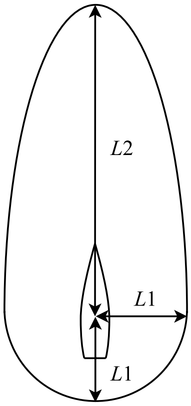


Fig. 1 Basic encounter situations and actions to avoid collision as specified in the COLREGs

The purpose of this study is to construct a new trained model which performs automatic collision avoidance with a longer safe passing distance at 0.5 NM. One possible reason why learning does not progress when the safe passing distance is increased is that the ship has to turn around too much due to the longer safe passing distance to avoid collisions and may lose the course to the given waypoint. To solve this problem, we introduce a recurrent neural networks (RNN) in continuous action spaces. The learning environment was implemented according to the OpenAI Gym [22], which is one of the standard environmental frameworks used in deep reinforcement learning research in recent years. This environment provides a flexible simulation environment of numerical simulation in various scenarios. The performance of the training model is verified by numerical simulations for the Imazu problem and a test scenario.

This paper is organized as follows: in the next section, we explain OZT and introduced a novel method named the inside OZT and the bow crossing distance to expand OZT. In Sect. 3, we explain algorithm to detect OZT and DRL, followed by a learning method and the environment of DRL in Sect. 4. The scenario used for training using DRL is also described in this section. We describe the results and evaluation of collision avoidance simulation using trained models

Fig. 2 Bumper model [20]



for Imazu problem and a test scenario in Sect. 5. Section 6 is devoted to discussion. Finally, we conclude the paper.

2 OZT

2.1 Computation of OZT

In this study, OZT is used for collision risk assessment. In actual operation, a ship's operator usually makes a decision to avoid collision based on the distance to a target ship and the change in azimuth of the target ship from the own ship. In addition, the closest point of approach (CPA) analysis is also used for collision avoidance. The time margin to collision is expressed by time to CPA (TCPA) and the distance to collision by the distance to CPA (DCPA). If DCPA is short, a ship officer will steer the ship to avoid other ships. The timing of steering will be determined by TCPA. Information obtained by CPA analysis is useful for safety navigation. However, results of CPA analysis do not tell which directions are safe. It is necessary for the operator to check whether a sufficient distance can be secured by actually changing the course of the own ship. This is the same for the automatic collision avoidance problem using TCPA and DCPA. To avoid a collision, it is necessary to predict the relative motion of the target ship and the possibility of a collision in the future. Then, we use OZT which is an area where a collision will happen in the future based on dynamic information of ships. The dynamic information including each target ship's position, speed and heading angle is assumed to be obtained from automatic identification system (AIS). There are several versions of OZT. In this study, an OZT is defined as a capsule-shaped area calculated using collision courses C_O as shown in Fig. 3. The collision course C_O of the own ship that may collide with the target ship in the future is calculated by Eq. (1) [23].

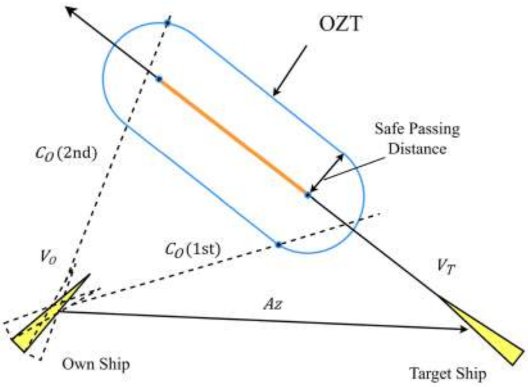


Fig. 3 Computation of OZT

$$C_O = \begin{cases} \mathrm{Az} \pm \alpha - \arcsin\left\{ \frac{V_T}{V_O} \sin\left(\mathrm{Az} \pm \alpha - C_T\right) \right\}, \\ \mathrm{Az} \pm \alpha - \pi + \arcsin\left\{ \frac{V_T}{V_O} \sin\left(\mathrm{Az} \pm \alpha - C_T\right) \right\}, \ \left(V_T > V_O\right) \end{cases} \tag{1}$$

where $\alpha = \arcsin\left(r_s/d\right)$. r_s is the safe passing distance and d is the distance between the own ship and the target ship. V_O is the own ship's speed and V_T is the target ship's speed. Az is the azimuth of the target ship's position from the own ship and C_T is the course of the target ship. Relative motion is computed when the own ship takes the collision courses C_O as follows.

$$\begin{aligned} \Delta X &= V_T \sin C_T - V_O \sin C_O, \\ \Delta Y &= V_T \cos C_T - V_O \cos C_O, \end{aligned} \tag{2}$$

$$\begin{aligned} V_R &= \sqrt{\Delta X^2 + \Delta Y^2}, \\ C_R &= \arctan \frac{\Delta X}{\Delta Y}, \end{aligned} \tag{3}$$

where V_R and C_R are the relative speed and course of a target ship against C_O, respectively. Then, DCPA and TCPA for each C_O can be obtained as follows.

$$\mathrm{DCPA} = d\left|\sin\left(C_R - \mathrm{Az} + \pi\right)\right| \tag{4}$$

$$\mathrm{TCPA} = \frac{d \cos\left(C_R - \mathrm{Az} + \pi\right)}{V_R}. \tag{5}$$

2.2 Inside OZT

We introduce Inside OZT, which expands the scope of the original OZT. OZT is an evaluation method originally introduced to determine the behavior of avoidance maneuvering in advance, and it is an indicator to be used when avoiding a collision. According to Eq. 1, to calculate the collision course, it is necessary for the distance between the own ship and the target ship to be more than the safe passing distance, and if not, OZT cannot be calculated. As a result, it is not possible to assess the risk of collision with the target ship when the ship is unavoidably close to the specified safe passing distance in congested water areas. To extend the safe passing distance, this problem should be solved. It is noted here that this characteristic is not a practical problem in usual use, especially in the case of human-operated ships. However, when OZT is used as a decision-making of collision avoidance, it is not possible to make a correct decision depending on OZT when the ship is extremely close to the target ship. To avoid this, it is necessary to supplement OZT when the distance to the target ship is less than the set safe passing distance. In this study, we develop a method to compute the supplementary OZT in a simple way. We call it the inside OZT and distinguish it from original OZT. The inside OZT is defined as that an OZT with TCPA when the distance between the ship and the target ship is less than or equal to the safe passage distance, $d \leq r_s$, an area of distance r_s or less centered on the line segment that the target ship moves from at TCPA = 0 to TCPA = $r_s/|V_O - V_T|$ on the heading direction of the target ship. The red area in Fig. 4 represents the inside OZT. The orange area in Fig. 4 shows a normal OZT (Sect. 2.1). If the speed of the own ship and the target ship are the same, the inside OZT extends to infinity farther on the target ship's heading direction. In the process of the simulation using deep reinforcement learning, when OZT of the target ship is not displayed, it is difficult to distinguish between "the case where there are no target ships around that may collide" and "the case where OZT is not displayed because the distance to the target ship is less than the safe passing distance". In some of the DRL algorithms, the value of an observed state is expressed as a value function. A vector of an observed state includes information of OZT distributions. The learning curve may be adversely affected because the value of the loss function for value functions may not decrease as the learning progresses. The value of loss function corresponds to the error between prediction by the network and actual data. Figure 5 is a comparison of the loss function during learning with and without the inside OZT. In the figures, each blue solid line shows the moving average of the loss function and the light blue area shows the standard deviation of the loss function for each iteration. By introducing the inside OZT, the value of the loss function of the value function decreases stably because the association between the reward obtained in collision and the state vector containing the OZT detection result can be consistent, then the learning becomes more stable as shown in Fig. 5.

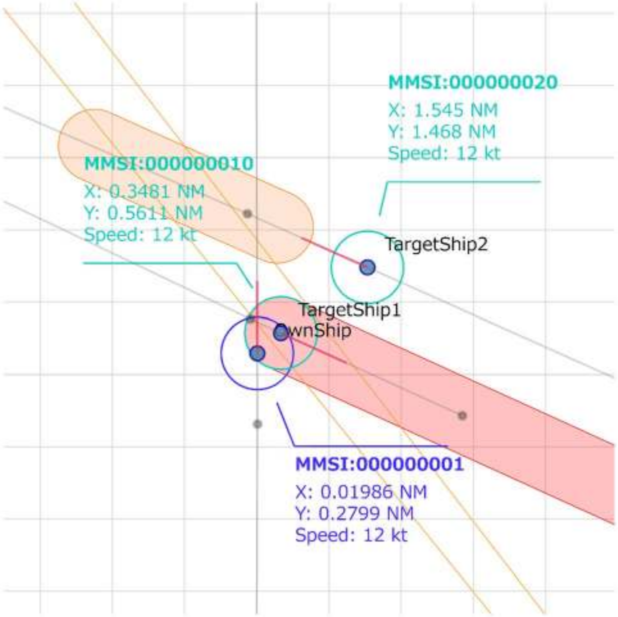


Fig. 4 Inside OZT (the red area is an inside OZT of `TargetShip1`. An OZT is shown as an orange area. Each circle surrounding a ship shows the safe passing distance set to compute OZT and the inside OZT) (colour figure online)

2.3 Bow crossing range

During actual maneuvering on the sea, it is usually avoided to pass in front of a target ship. However, if the distance between the own ship and the target ship is sufficient large, it may be possible to pass in the front of the target ship. Basically, it is known statistically that it is natural and safe for a

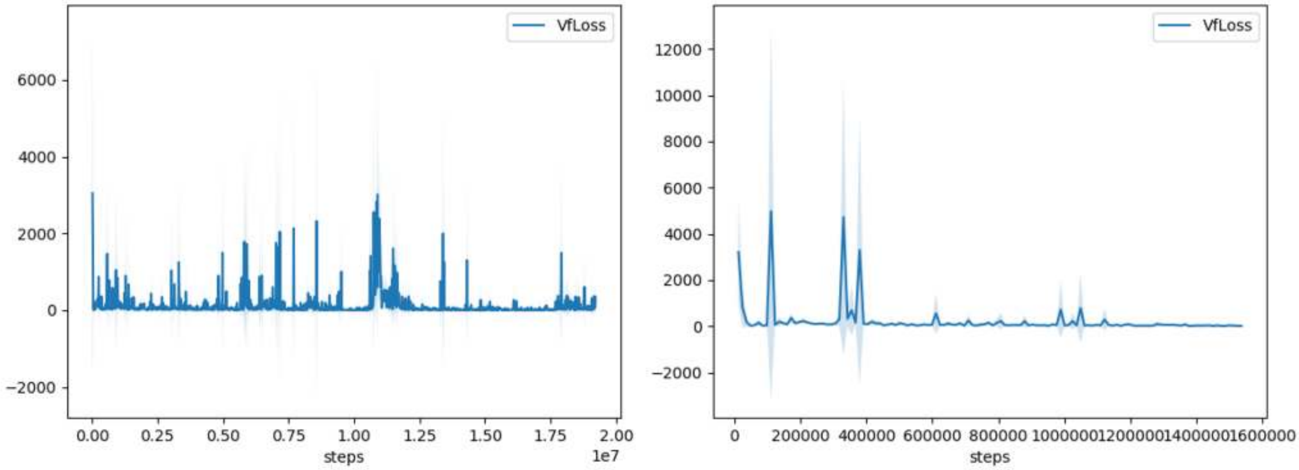


Fig. 5 Comparison of changes of the loss function for value network during learning with/without inside OZT (left: not using inside OZT, right: using inside OZT)

ship to have a longer distance ahead of it than in other directions such as both sides and stern of the ship, as represented by methods such as the bumper model, the ship domain and effective domain [24]. There are two possible ways to introduce the bow crossing range into OZT: first, there is a virtual ship that goes ahead of the target ship at the same speed and its distance is equal to the bow crossing range minus the safe passing distance. Then, the original OZT is extended by the TCPA of the collision course calculated by this virtual ship. As a simpler way, it is also possible to correspond by extending the area of OZT by subtracting the safe passing distance from the bow crossing range. In this study, we adopt the latter simple method because of the simplicity of the implementation. To perform detection of collision corresponding to OZT distribution with the bow crossing range, we additionally defined a region like the ship domain for collision detection with the bow range corresponding to the definition of OZT as the area enclosed by the solid line in Fig. 6. A collision is judged when the own ship enters this domain of target ships. As shown in Fig. 6, this area is a capsule-shaped region with a radius of the safe passing distance and the bow crossing range is set to 1.0 NM.

3 Algorithm

3.1 Detection of OZT

To process information of OZT, we use a virtual sensor called the grid sensor [17]. It is required to detect OZT and convert it into a form that can be easily used as an input of deep neural networks used in DRL. Because, networks in reinforcement learning only accept vectors, the dimension of which is fixed. However, to avoid multiple ships, the automatic collision avoidance system should track more than one OZT simultaneously and the number of ships maybe change during navigation. In addition, it also needs to be able to detect OZT with a high resolution over a wide area of several nautical miles. Ships at sea avoid a collision according to the COLREGs. The rule 8 of the COLREGs states that action to avoid a collision should be positive and made in ample time. It is necessary to observe the situation from an ample distance and choose an appropriate action in ample time. For this reason, one of the authors designed the grid sensor. The schematic diagram of detection of OZTs using a grid sensor is shown in Fig. 7. The grid sensor is a virtual sensor that extends from the center of the own ship and is separated by evenly spaced intervals of the angle direction and the radius direction in a concentric circle grid. When a grid cell overlaps with an OZT, it is judged detecting the OZT on each grid cell of the grid sensor. After detection, the grid sensor returns a state vector, the dimension of which is equal to the number of its grid cells. Each component of the state vector by the grid sensor is set to 1 if the corresponding grid cell detects OZT, 0 otherwise. In this way, the collision avoidance system recognizes OZTs as a vector with a fixed dimension regardless of the changes of OZT distribution.

3.2 Deep reinforcement learning algorithm

The proposed collision avoidance algorithm is based on DRL. DRL is a kind of machine learning, and a combination of reinforcement learning (RL) and deep learning. RL algorithm uses an environment and agents. Agents and the environment interact with each other to promote learning. Agents receive information from the environment called the state and take an action. Then, each agent receives a reward for its action and a new state of environment. Agents learn in the environment to maximize the cumulative reward. In collision avoidance tasks, an agent corresponds to the own ship and an environment consists of some components such as target ships and waypoints. There is a wide variety of DRL algorithms. The

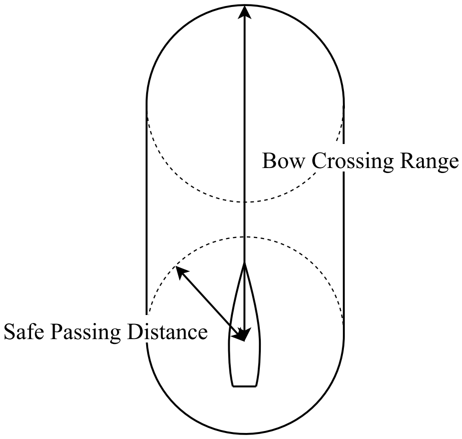


Fig. 6 Domain for collision detection

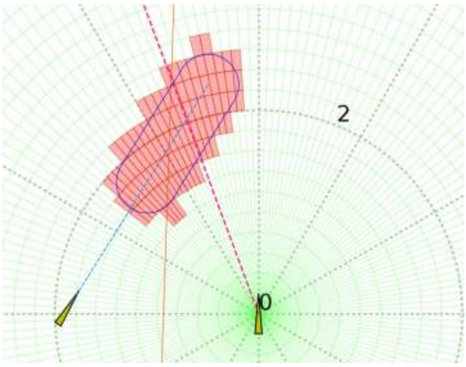


Fig. 7 Detection of OZT by the grid sensor (the unit of radius is nautical mile and size of ships' plots is 4 times of full scale)

differences in these DRL algorithms depend on how the agent learns and how it chooses its actions. For some kinds of algorithms such as on-policy Actor-Critic method, probability distributions of actions is given by a policy function (Actor) and the value of the action is represented by the value function (Critic). In DRL, these functions are represented as deep neural networks. It is called the function approximation, which is approximation of a policy function or a value function with parameterized functions. Using expressive power of deep learning, DRL algorithms can control an agent based on high-dimensional inputs like images and the grid sensor.

As mentioned above, there are many kinds of DRL algorithms depending on processes such as update policy/value functions, evaluation of a state and determination of action. In this paper, we used proximal policy optimization algorithm (PPO) [25] because PPO outperforms other algorithms such as DQN and is applied to tasks in real world including robotic locomotion. The code of PPO in this paper was implemented using DRL library machina coded by PyTorch [26] which is a deep learning framework for Python. PPO is an algorithm that has its origins in reinforcement learning of two Actor-Critic methods: Trust Region Policy Optimization (TRPO) [27], which restricts policy updates according to the Kullback–Leibler divergence of the probability distribution before and after the update, and Asynchronous Advantage Actor-Critic (A3C) [28], which uses distributed learning and the advantage to update the probability distribution. Actor-critic method uses two networks basically. One is a policy function $\pi(a_t|s_t;\theta)$ which is formulated in the form of a posterior probability distribution for a given state vector to determine a next action. The other is a value function $V(s_t;\theta_v)$ to evaluate a state of an environment, where a_t is an action and s_t is a state at step t. A policy function and a value function are parameterized by θ and θ_v, respectively. In our implementation, these two networks are independent and do not share any part of them. An update using the advantage, if without using the state-action value function directly, performs via $\nabla_\theta \log \pi_\theta\left(a_t|s_t;\theta\right)\hat{A}\left(s_t,a_t;\theta,\theta_v\right)$ obtained from an estimate of the advantage given by $\hat{A}\left(s_t,a_t;\theta,\theta_v\right) = \sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V\left(s_{t+k};\theta_v\right) - V(s_t;\theta_v)$,where k can vary from a state to a state and is upper-bounded by max step of each episode. In the implementation of machina, a set of estimates of the advantage in a batch is normalized with its variance and mean. The distinctive feature of PPO is to use the modified objective function after the idea of TRPO to update policy with the simple implementation. A primary variant of PPO called PPO-clip stabilizes policy update by Eq. 6.

$$\theta_{t+1} = \text{argmax}_\theta\, E_{s,a\sim\pi_{\theta_t}}\left[L\left(s,a,\theta_t,\theta\right)\right]. \tag{6}$$

Here, objective function L is given by

$$L\left(s,a,\theta_t,\theta\right) = \min\left(r_t\left(\theta_t,\theta,\right)\hat{A}_t, \text{clip}(r_t\left(\theta_t,\theta\right), 1-\epsilon, 1+\epsilon)\hat{A}_t(s,a)\right), \tag{7}$$

where $r_t(\theta_t,\theta) = \pi(a_t|s_t;\theta)/\pi(a_t|s_t;\theta_t)$, ϵ is a hyperparameter and $\hat{A}_t(s,a)$ is an estimate of advantage at the step t. For updates of the value function, the Monte-Carlo method is adopted. Specifically, the Monte-Carlo method minimizes the mean squared error between the cumulative discounted reward $G_t^{\mathrm{T}} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{\mathrm{T}-1} r_{\mathrm{T}}$ and the current value function $V(s_t|\theta_v)$.

4 Learning method

4.1 Environment for learning of collision avoidance

The environment consists of target ships, a waypoint and a target area. In this paper, it was assumed that the target area was an open sea with no obstacles such as coast lines, buoys. A waypoint was set as a destination of the own ship which was a controllable agent. The own ship must go to a waypoint while avoiding target ships. Target ships were set in the target area according to a set of encounter situations. The motion of ships was calculated by Nomoto's equation [29] for the heading and a primary delay equation for the rudder motion as Eq. 8 and the coordinate system of ship motion is shown in Fig. 8.

$$\begin{bmatrix}\dot{\psi}\\ \dot{r}\\ \dot{\delta}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & -1/T & K/T\\ 0 & 0 & -1/T_E\end{bmatrix}\begin{bmatrix}\psi\\ r\\ \delta\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1/T_E\end{bmatrix}\begin{bmatrix}0\\ 0\\ \delta_C\end{bmatrix}, \tag{8}$$

where ψ is a heading angle. r is a rate of turn. δ is a rudder angle of a ship and δ_C is a command rudder angle. T and T_E are time coefficients of heading and rudder motion. K is a gain. All of own and target ships had the same parameter for motion calculation assuming a kind of cargo ships. The own ship and target ships cannot change their speed. The Runge–Kutta method was used to integral motion equations.

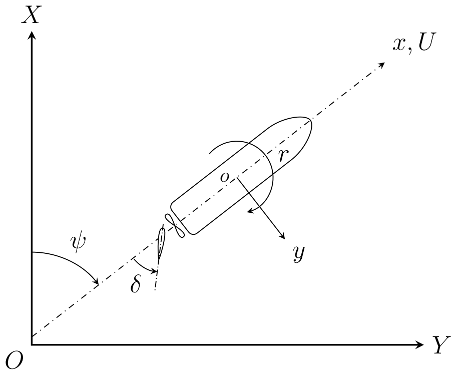


Fig. 8 Coordinate system of ship motion

Table 1 Subjects of ships for learning

K (1/s)	0.05
T (s)	50.0
T_E (s)	2.5
Ship's speed, U (kt)	12.0, 8.4 (for ships over-taken only)
Ship length between perpendiculars, L_{PP} (m)	106.0
Ship breadth, B (m)	16.2

Table 2 Configurations of environment

Subjects	Value
Safe passing distance (NM)	0.5
Grid sensor	
Angle of detection (°)	360
Grid space on angular direction (°)	2.0
Radius of sensor (NM)	12.0
Grid space on radius direction (NM)	0.2
Detection intervals (s)	10.0

In this study, two types of control methods are used to train model in the continuous action spaces: a rudder control model that outputs the command rudder angle and an autopilot model that outputs the command heading angle. For the model of rudder angle control, the command rudder angle is determined based on the current policy in the range of − 10° to +10°. The autopilot model selects a change of the heading angle of autopilot in the range of − 10° to +10° based on the current policy at each time step. Table 1 shows a summary of the vessels used in this calculation. These are the same for all ships in the simulation, whether their own or target ships.

The policy and value networks of PPO get a state vector from environment. A state vector consist of as follows: (1) OZT information by a grid sensor detection results, (2) normalized values of a heading angle, a rate of turn, a speed and a rudder angle of the own ship and (3) normalized values of an azimuth angle and a distance to a waypoint and a command rudder angle from autopilot toward a waypoint. In this algorithm, dynamic information of target ships is grasped only from a detection result of a gird sensor. Thereby, it is a feature of the proposed algorithm that the dynamic information of the target ships and position information of the own ship are not handled directory. To reduce the computational complexity, detection of a grid sensor and updating of a command rudder angle were carried out every 10 s in the simulation time, and for the motion calculation, the interval of the integration was set to 1 s. The setting of other learning environments is shown in Table 2. A grid sensor is assumed to use information provided by AIS. The practical range of shipborne AIS communication is about 12 NM [30]. We determined the radius of the grid sensor based on this. In this study, we implemented the environment using Python with OpenAI Gym which is used as a standard platform for development and evaluation for RL algorithms. By this implementation, it is easy to apply any other DRL algorithms to this environment.

4.2 Design of rewards

The rewards are designed by dividing them into two kinds of rewards: basic rewards, which are added at each step, and achievement rewards, which are given at the end of each episode. An episode is defined as a sequence from a start of a simulation to a termination of the simulation by satisfying terminal conditions. The terminal conditions of an episode are that the distance to a waypoint becomes less than or equal to a specified distance, or the simulation reaches the set maximum steps. The basic rewards *Costs* are defined as shown in Eqs. 9–12. The $\mathrm{Costs}_{\mathrm{wp}}$ calculated by Eq. 10 is a positive reward that is given more as the own ship approaches a given waypoint. For compliance with the COLREGs, a small positive reward is given as $\mathrm{Costs}_{\mathrm{starboard}}$ to encourage the own ship to pass through the area on the right side of the line connecting the initial position of the own ship with a given waypoint, so that an agent basically learns to avoid toward the starboard side. $\mathrm{Costs}_{\mathrm{stable}}$ is defined to stabilize heading control by trained models. In this study, an additional reward of − 5 is given instead of terminating the episode at the step where the collision was judged.

$$\mathrm{Costs} = \mathrm{Costs}_{\mathrm{wp}} + \mathrm{Costs}_{\mathrm{starboard}} + \mathrm{Costs}_{\mathrm{stable}}, \tag{9}$$

$$\mathrm{Costs}_{\mathrm{wp}} = 0.9 \tanh(1/d_{\mathrm{wp}}), \tag{10}$$

$$\mathrm{Costs}_{\mathrm{starboard}} = \begin{cases} 0.05, & \mathrm{Az}_{\mathrm{wp}} \geq 0 \\ 0.0, & \mathrm{Az}_{\mathrm{wp}} < 0, \end{cases} \tag{11}$$

$$\mathrm{Costs}_{\mathrm{stable}} = -0.01|r/\pi|, \tag{12}$$

where d_{wp} and $\mathrm{Az}_{\mathrm{wp}}$ are the distance and the azimuth to a given waypoint from the own ship. Results are set according to the end condition of each episode: − 50 for deviation from the target area, − 50 for a collision, and + 50 for reaching within the specified distance from a given waypoint without collisions. The scale of achievement rewards is determined based on the result of the preliminary learning carried out beforehand as the scale for that the effect of $\mathrm{Costs}_{\mathrm{starboard}}$ with a small value does not disappear, while encouraging the own ship to avoid collision.

4.3 Structure of networks and update method

The policy and value function used in PPO are represented by deep neural networks. In the present study, we set a safe distance of 0.5 NM in the discrete action space in advance, but the trained model by previous approach [17] did not reach the sufficient performance. One of the possible reasons is that networks consist of only convolutional layers and full-connected layers (FC) cannot store historical information of the environment. We introduced the recurrent neural network (RNN), which can deal with time series data. Specifically, we used the long short-term memory (LSTM), which is a kind of RNN. Among the input states, the detection result of the grid sensor and the dynamic information of the own ship and information of the waypoint are different in nature, and the results of the grid sensor needs to be processed by the convolutional layer used in image learning. On the other hand, real numerical data, such as dynamic information and information of waypoint, can be processed in the layer of all joins because of its small number of dimensions. For this reason, as shown in Fig. 9, we divided the state vectors of the grid sensor and the state vectors of other numerical information and input them separately in the convolutional layer and the full-connected layer in the input layer, respectively, and finally combined the results of each output into one network. To learn in a continuous action spaces, we introduce a network structure as shown in Fig. 9, which has different output layers from those for the discrete action spaces. In this network, the LSTM cell is placed before the output layer. In our implementation, the two networks have no shared parts. Here, the rudder control model, the network of policy has two convolutional layers, and the value function has only one convolutional layer. We used Adam [31] to update the network. The hyperparameters for PPO in continuous action spaces are provided in Table 3. The hyperparameters of the previous model in discrete action spaces are described in [17].

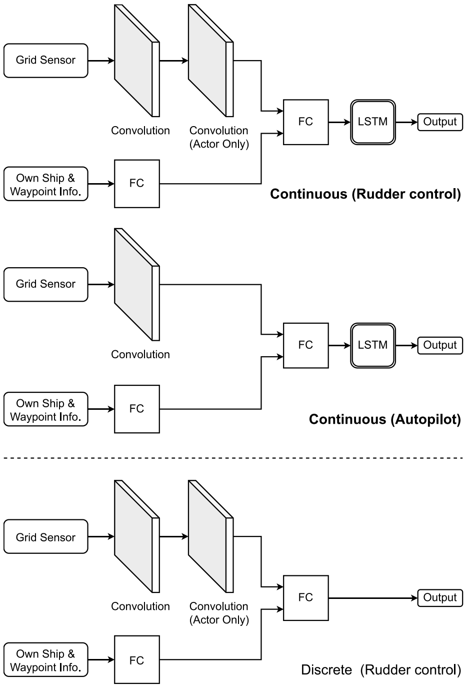


Fig. 9 Structures of networks used in PPO (top: network for continuous action space with rudder control, middle: network for continuous action space with autopilot, bottom: network for discrete action space in [17])

4.4 Scenario

The encounter situations used during learning affect the quality of the collision avoidance model. It is suitable that the set of situations includes from easy one like 1 on 1 encounters to difficult one like encounters of many ships. There is a scenario set for collision avoidance tests. Woerner et al. proposed the scenario set [32]. This scenario has six situations of ship encounters. Cai and Hasegawa proposed an evaluation method of performance of automatic collision avoidance systems [33]. In this method, they used Imazu problem [19] as a benchmark which is a set of ship encounter situations as shown in Fig. 10. Imazu problem consists of basic ship encounters of 1 on 1 and difficult situations of multiple ships. In this paper, we used Imazu problem as scenario for learning. In Fig. 10, numbers on the top left in each of boxes indicate the number of cases of Imazu problem. Each of short bars from triangles (the own ship) or circles (target ships) is a velocity vector of each ship. According to the report by Cai and Hasegawa, the problem may become easier by the elimination of the collision risk when the avoidance maneuvering by a target ship is permitted. In this paper, target ships are able to only go straight without making any changes to their course by waypoint navigation

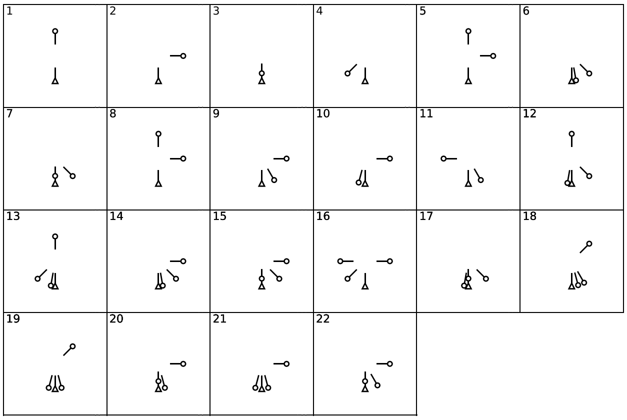


Fig. 10 Imazu problem [19]

or avoidance maneuvering. In addition, to improve the generalization performance of the learned model, one random case that 3 ships are randomly arranged is prepared, and the learning was carried out under the problem of total 23 cases including 22 cases of the Imazu problem and a random case. The position and course of each ship for every case were set so as to collide at the origin of the space fixed coordinates. In learning, 1 case out of 23 case was chosen randomly when it was initialized in each episode start, and target ship was arranged according to the configurations of each case. For the initial position and heading angle of target ships in each case in Imazu problem, see Appendix 1. Each target ship is positioned at the speed so that the TCPA is 30 mins. The own ship is positioned at (X [NM], Y [NM]) = (− 6.0, 0.0) and its heading angle is randomly set in the range of − 5° to + 5° for every episode for generalizability.

5 Results

In this section, we describe the results of automatic collision avoidance using the trained model by the proposed approach. First, we show the results for all scenarios of Imazu problem using the two trained models of continuous action spaces and the previous trained models used in the previous study [17]. This previous model is trained at the safe passing distance of 0.3 NM and OZT is not extended to the bow crossing distance in learning and this validation. Using these models, a total of 22 cases of the Imazu problem were simulated to verify the performance of collision avoidance. In all scenarios, the waypoints are placed at (X (NM), Y (NM)) = (6.0, 0.0). Trajectories by the three trained models in all 22 cases of Imazu problem are shown in Figs. 11, 12 and 13. The corresponding bar graphs of the minimum passing distance in each case of the two continuous action spaces models and the discrete action spaces model are shown in Fig. 14. In the case of the discrete action spaces model, the safe passage distance during learning was set to 0.3 NM, which results in monotonous trajectories with small heading angle changes overall. On the other hand, the two models of continuous action spaces learned at a safe distance of 0.5 NM have been learned to take a longer distance to the target ships than the previous model. From the point of view on the COLREGs, the discrete action spaces model cannot find a path to avoid ship according to the COLREGs in some cases such as case 6. The model of continuous action spaces with rudder control found paths to avoid target ships by starboard turning, which is appropriate avoidance maneuvering for almost cases. In the almost results in Figs. 11 and 12, the own ship passed behind the target ships. This can be due to the setting of bow crossing range of OZT and collision detection. However, the autopilot model may maneuver from the initial OZT such as case 12 and case 17, which deflects the course significantly at the beginning, and the minimum safe passing range is also large. For all cases of Imazu problem, the minimum passing distance of autopilot model tends to be larger than that of the rudder control model. Moreover, it can be inferred from the trajectories in Fig. 12 that the effect of Costs$_{\text{stable}}$ is longer for the autopilot model than for the rudder control model. On the other hand, the rudder control model has mastered advanced control such as turning to the destination as shown in Fig. 15. Such a drastic change of course was not observed in the trained model of discrete action spaces at all. This is a major feature of the continuous action space model. This

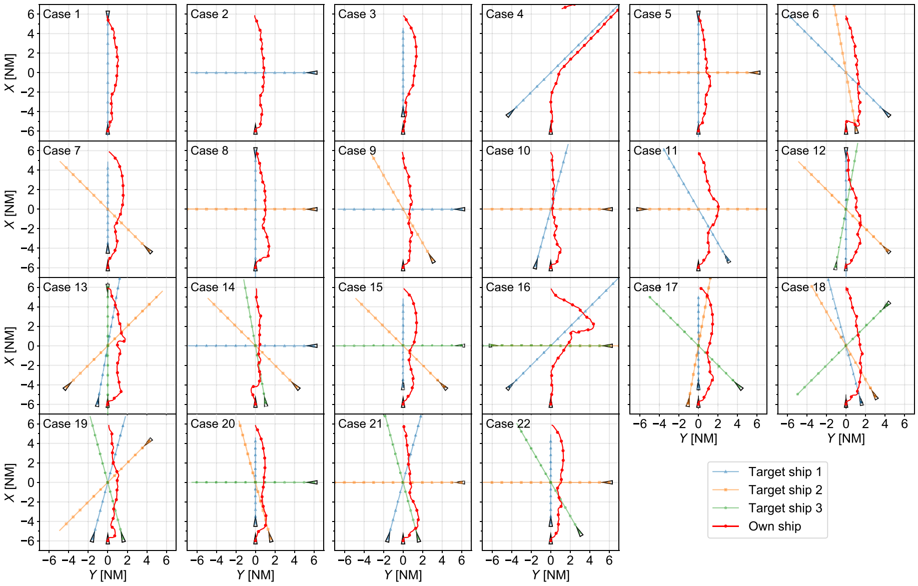


Fig. 11 Trajectories through Imazu problem using the continuous action space model with rudder control

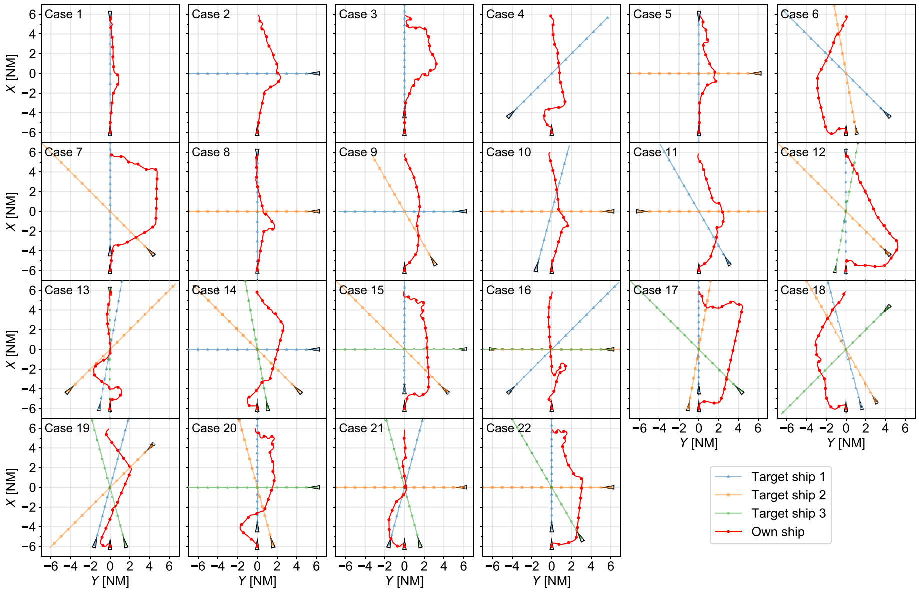


Fig. 12 Trajectories through Imazu problem using the continuous action space model with autopilot

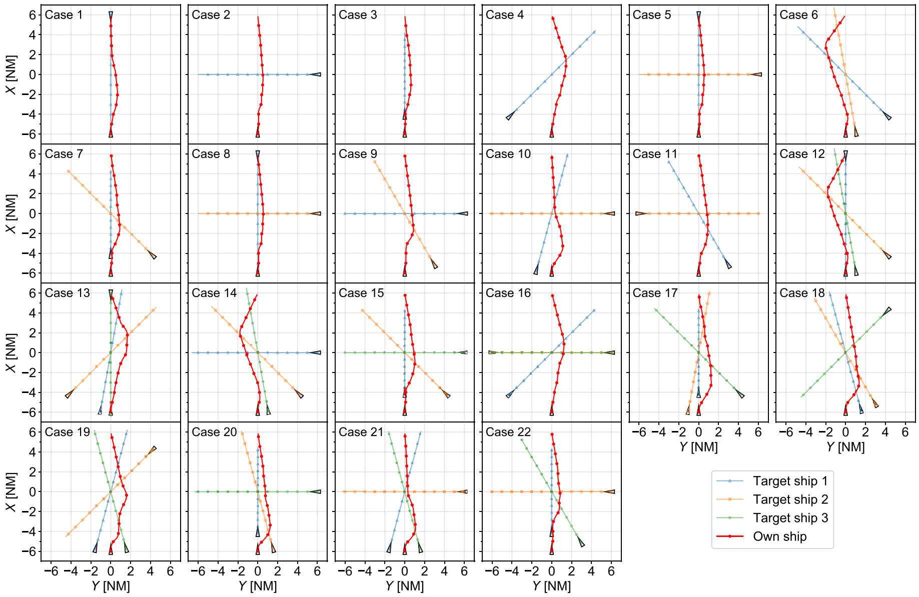


Fig. 13 Trajectories through Imazu problems using the discrete action space model with rudder control [17] (the safe passing distance is 0.3 NM)

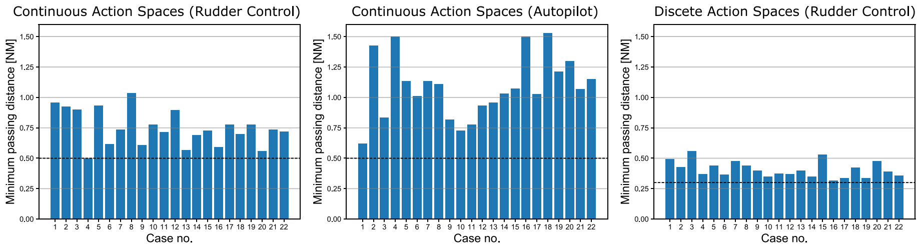


Fig. 14 Minimum passing distance of trained models (the safe passing distance for the trained models in continuous action spaces is 0.5 NM, and it for the trained model in discrete action spaces is 0.3 NM [17])

may be due to the reward design that there is no negative reward for the elapsed time, so there is a less need to hurry to arrive at the waypoint. Nevertheless, it is necessary to design the reward so that the model can be constructed in a time-efficient manner while retaining this flexible control, since it is necessary to consider the economy in actual ship operation.

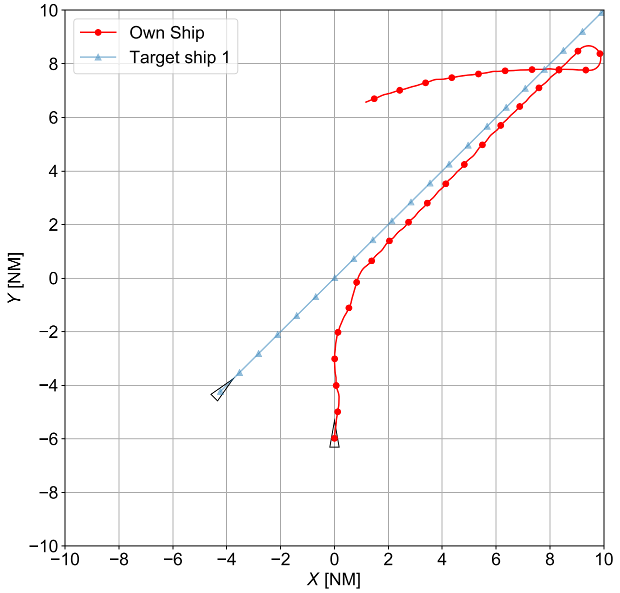


Fig. 15 Turning behavior in case 4 of Imazu problem in Fig. 11

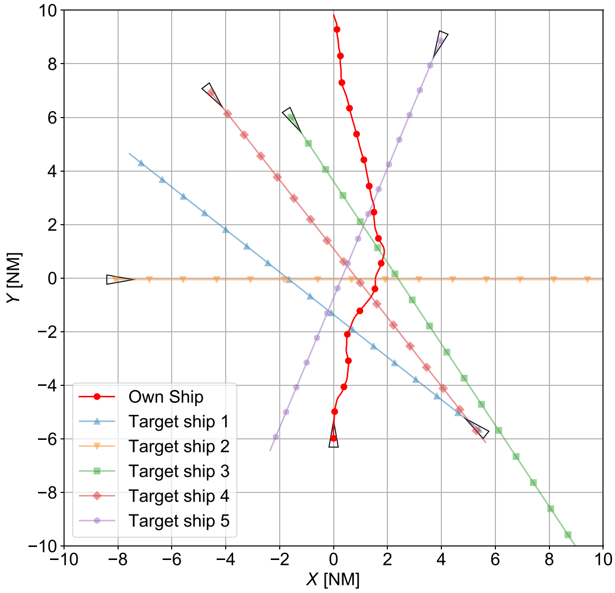


Fig. 17 Trajectory by the continuous action space model with rudder control for the test scenario

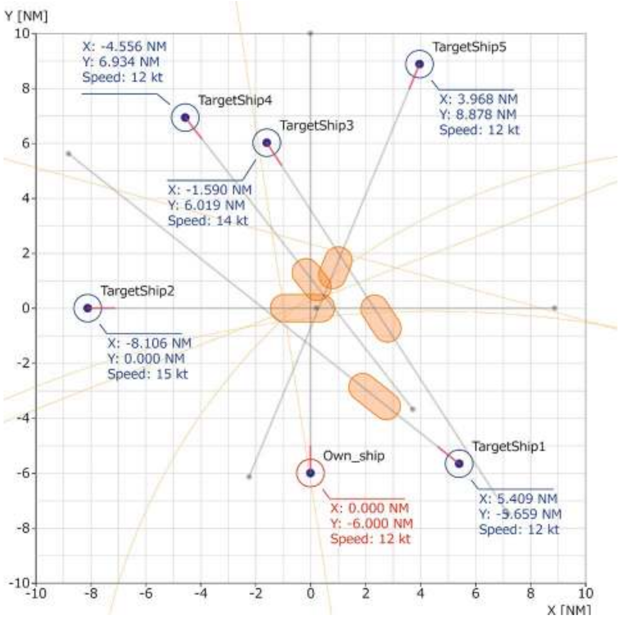


Fig. 16 Initial position and OZT distributions of the test scenario

For validation of the rudder control model, we have prepared a test scenario shown in Fig. 16. In the test scenario, there are five target ships at different speed from 12.0 to 15.0 kt. The waypoint of the own ship is set at (X [NM], Y [NM]) = (0.0, 10.0). From the initial OZT distribution, one of the desirable paths to arrive at the waypoint is way on the starboard of the own ship through the gaps of OZTs. The trajectory in Fig. 17 shows the trained model with rudder control learned such desirable way to avoid collision and arrive at the waypoint. The minimum passing distance of this scenario is 0.753 NM. Although the own ship's speed is lower than the target ships 2 and 3, the trained model avoided them. As shown in this test, the trained model with rudder control in continuous action spaces learned generalized maneuvering to avoid ships and arrive at the waypoint with the longer safe passing distance is 0.5 NM.

6 Discussion

By introducing negative rewards for the collision detection with the bow crossing range and the small positive rewards of the right-handed rewards Costs$_{\text{starboard}}$, the trained model of rudder control is able to learn to control the own ship in a way that does not violate the three rules of the COLREGs described in the Introduction. For instance, the previous trained model of discrete action spaces had crossed the front of the target ships in some cases of Imazu problem such as case Nos. 4, 6 and 12. Conversely, these results indicate that there is the possibility that the basic collision avoidance behavior for head-on, crossing and overtaking situations defined by the COLREGs can be composed of the principle for securing the bow crossing range and passing on the right side of the original course to the waypoint. Several studies by Imazu [4–6], Hu et al. [8], and Woerner et al. [12] defined detailed functions to evaluate the legality for the COLREGs. In contrast, our method shows that only OZT with the bow crossing range (and the cost term Costs$_{\text{starboard}}$ of rewards) is enough for COLREGs-compliant collision avoidance for multiple ships navigation problems. This also means we can improve methods of collision avoidance by modifying not only functions but also representations of collision detection based on the statistical models such as the bumper model. Imazu problem is a set of short-term scenarios that are easy to maneuver in response to the COLREGs, but in the test scenario prepared this time, if the ship tries to proceed in the one of the shortest path, the course that passes in the front of target ships is selected. The question here is whether the trajectory generated by the trained model which takes action in compliance with the COLREGs in short-term scenarios is acceptable in practice. The choice of such a course is determined by various factors such as the degree of congestion of the sea, the shape of the navigable area, and the distance to the waypoint. One of the future research agendas is to examine whether the trajectory from the trained model is realistic using more complex scenarios and real traffic data, while at the same time investigating the essential elements needed to bridge the difference with real ship operations in the DRL framework.

A characteristic of DRL for ship control problems is that the two control methods, rudder control and autopilot, show significant differences in the same reward design. These models differ in the number of convolutional layers in the network structure, but the presence or absence of this effect is unknown. Nevertheless, the linear trajectory inherent in the results by the model of autopilot indicates that the reward design may need to be determined according to how the vessel is controlled. The reason why the trained model of autopilot could not learn in the same network as the trained model of rudder control in continuous action spaces is that the deviation from the original course is too large for collision avoidance maneuvers, which corresponds to the reason why the discrete model could not learn. From the learning process of these two models, we consider that it is necessary to design state vectors and rewards that limit the deviation from the course to learn collision avoidance maneuvers while maintaining the performance of waypoint navigation. In the future research, we will design learning algorithms that take these factors into account.

7 Conclusion

In this study, we proposed a novel approach of automatic collision avoidance using DRL in continuous action spaces with the longer safe passing distance at 0.5 NM. By our previous approach, when the safe passing distance was set to 0.5 NM, the learning did not progress and the performance of the trained model was not satisfying. The proposed method solves this problem by developing new OZT representations, changing the network structure, and learning in continuous action spaces. We extend OZT by introducing the inside OZT and the bow crossing range. The inside OZT promotes the reduction of the loss of the value function during learning using DRL. We continued to investigate the combination of OZT detection by grid sensors and deep reinforcement learning as a method to perform collision avoidance and waypoint navigation of multiple ships at the same time. Using OZT and collision detection with the bow crossing range, the model learned safer maneuvering that the ship is less likely to cross the front of the target ships. This result shows the bow crossing range in collision detection is effectual for realizing safe and COLREGs-compliant maneuvering in a simple way. By introducing the LSTM cell in the continuous action spaces into network of PPO, the new trained model shows that it can make complex decisions such as giving up collision avoidance and returning to a waypoint in the midst of operation, which has not been seen in behaviors by trained model learned in the discrete action spaces. The trained model has found an appropriate course to weave its way through OZTs even in unknown test scenarios and has learned collision avoidance maneuvering with generalization performance.

On the other hand, the trained model of the continuous action spaces tends to have low course stability. As a result, the ship's operation may cause anxiety to the other ship. In the next stage of this research, we would like to build a model with more course stability by various approach including a review of the structure of state vectors and networks, as well as reward design. In addition, it is future work to build the automatic collision avoidance model which can deal with ship encounters including changes of ships' speeds and courses.

Acknowledgements This work was supported by JSPS KAKENHI Grant numbers 19H02371 and 20K14971.

Appendix 1: Learning settings of PPO

The hyperparameters of the neural network structure and learning used in the learning of continuous action spaces in this study are shown in Table 3. As for the learning rate, it does not decay and is fixed during learning. In this study, we used Pytorch to perform operations around neural networks on a GPU using CUDA 9.0 to shorten the learning time. Computations of the environment, including ship motion calculations and grid sensor detection, were performed on CPU. For the calculations, we used a Windows 10 desktop machine (core i7 8700K, NVIDIA GeForce 1060 6GB). Note that machina used in this study officially supports only Ubuntu as an OS, and Windows is not officially supported. For this reason, some of the implementations of multiprocessing and log output for Pytorch and Python in parallel were changed for Windows and the machina code was modified.

Table 3 Hyperparameters for PPO

Parameters	Value
Optimizer	Adam
Learning rate	
Actor	$1.0 \cdot 10^{-4}$
Critic	$3.0 \cdot 10^{-4}$
Discount *gamma*	0.995
1st convolutional layer for grid sensor	
Number of output channel	256
Kernel size	8
Stride	4
Padding	8
The number of channels of output	32
2nd convolutional layer for grid sensor	
Number of output channel	128
Kernel size	4
Stride	2
Padding	4
The number of channels of output	32
LSTM cell	
Input size	1024
Hidden size	512
Hidden units of fully connected layers for other state vector	256
Hidden units of fully connected layers for merged output	128
Size of clipping gradient norm	20.0
Nonlinearity	ReLU
Number of processes to parallel sampling	6
Batch size	8
Number of epoch in a iteration	10

Settings of Imazu problem

Imazu problem proposed in [19] has two group of encounter scenarios. Group 1 includes relative difficult situation Table 4 shows the initial settings of the Imazu problem used in learning and test of the model. The coordinates in Table 4 are the spatial fixed coordinates. Numbers of target ships in Table 4 is not always corresponding to them used in the simulations of this paper. Figure 18 shows the initial conditions for each case set in Table 4. However, to make the display of the ship easier to see, it is drawn at a size equivalent to 10 times the actual ship scale.

Table 4 Settings of Imazu Problem (own ship starts from $(X\ [NM], Y\ [NM]) = (-6.0, 0.0)$ with $\psi = 0(°)$)

Ship	Target ship 1			Target ship 2			Target ship 3		
Case no.	X (NM)	Y (NM)	ψ (°)	X (NM)	Y (NM)	ψ (°)	X (NM)	Y (NM)	ψ (°)
1	6.000	0.000	180.0	–	–	–	–	–	–
2	0.000	6.000	– 90.0	–	–	–	–	–	–
3	– 4.200	0.000	0.0	–	–	–	–	–	–
4	– 4.243	– 4.243	45.0	–	–	–	–	–	–
5	6.000	0.000	180.0	0.000	6.000	– 90.0	–	–	–
6	– 5.909	1.042	– 10.0	– 4.243	4.243	– 45.0	–	–	–
7	– 4.200	0.000	0.0	– 4.243	4.243	– 45.0	–	–	–
8	6.000	0.000	180.0	0.000	6.000	– 90.0	–	–	–
9	– 5.196	3.000	– 30.0	0.000	6.000	– 90.0	–	–	–
10	0.000	6.000	– 90.0	– 5.796	– 1.553	15.0	–	–	–
11	0.000	– 6.000	90.0	– 5.196	3.000	– 30.0	–	–	–
12	– 4.243	4.243	– 45.0	– 5.909	1.042	– 10.0	–	–	–
13	6.000	0.000	180.0	– 5.909	– 1.042	10.0	– 4.243	– 4.243	45.0
14	– 5.909	1.042	– 10.0	– 4.243	4.243	– 45.0	0.000	6.000	– 90.0
15	– 4.200	0.000	0.0	– 4.243	4.243	– 45.0	0.000	6.000	– 90.0
16	– 2.970	– 2.970	45.0	0.000	– 6.000	90.0	0.000	6.000	– 90.0
17	– 4.200	0.000	0.0	– 5.909	– 1.042	10.0	– 4.243	4.243	– 45.0
18	4.243	4.243	– 135.0	– 5.796	1.553	– 15.0	– 5.196	3.000	– 30.0
19	– 5.796	– 1.553	15.0	– 5.796	1.553	– 15.0	4.243	4.243	– 135.0
20	– 4.200	0.000	0.0	– 5.796	1.553	– 15.0	0.000	6.000	– 90.0
21	– 5.796	1.553	– 15.0	– 5.796	– 1.553	15.0	0.000	6.000	– 90.0
22	– 4.200	0.000	0.0	– 4.243	4.243	– 45.0	0.000	6.000	– 90.0

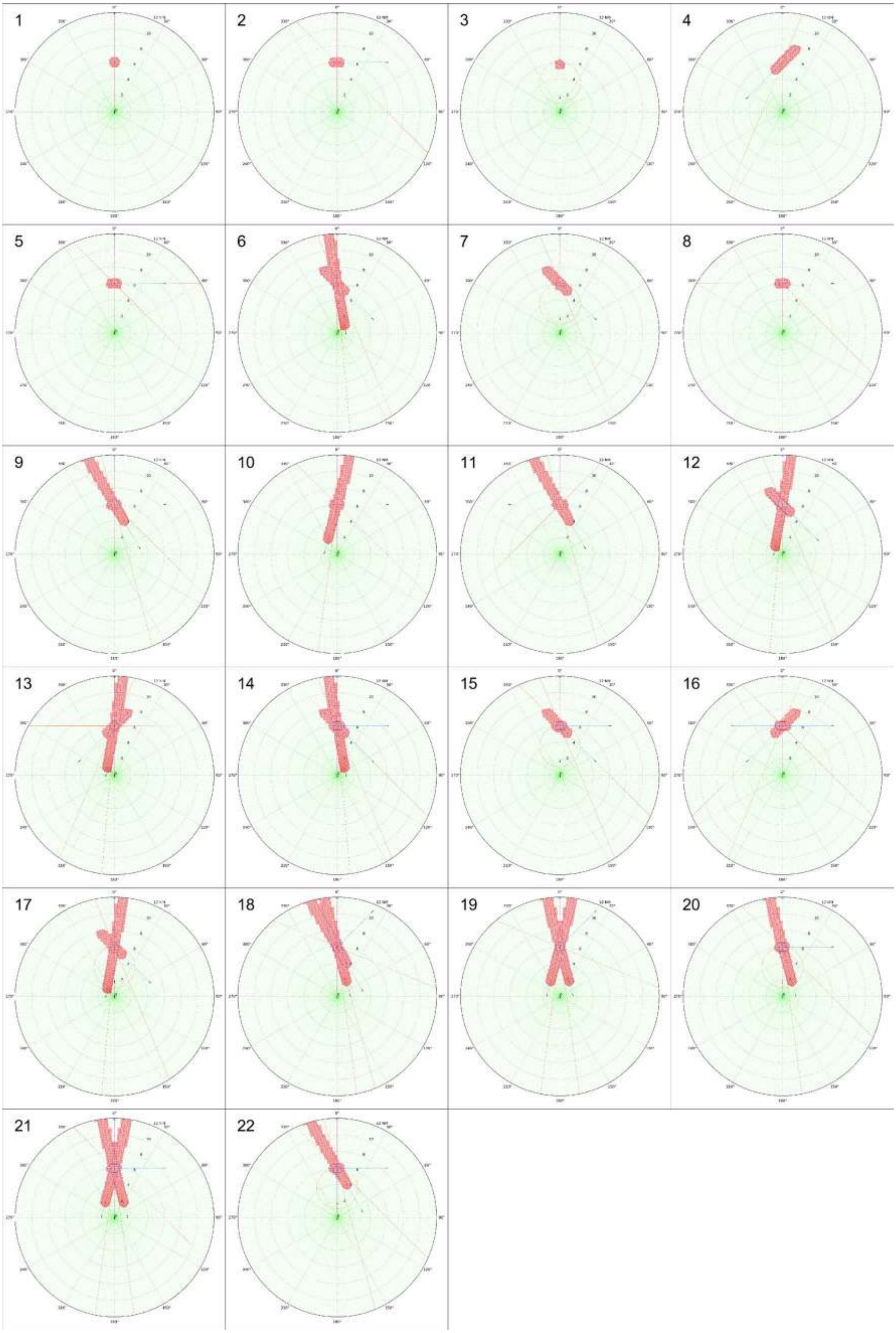


Fig. 18 The initial state of Imazu problem and OZT detection by the grid sensor (the safe passing distance of OZT is 0.5 NM)

References

1. IMO (2020) Autonomous shipping. http://www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx. Accessed 15 July
2. Chauvin C, Lardjane S, Morel G, Clostermann JP, Langard B (2013) Human and organisational factors in maritime accidents: analysis of collisions at sea using the HFACS. Accid Anal Prev 59:26–37
3. Huang Y, Chen L, Chen P, Negenborn RR, van Gelder P (2020) Ship collision avoidance methods: state-of-the-art. Saf Sci 121:451–473
4. Imazu H, Koyama T (1984) The optimization of the criterion for collision avoidance action. J Jpn Inst Navig 71:123–130 (in Japanese)
5. Imazu H, Koyama T (1985) The optimization of the criterion for collision avoidance action-II. J Jpn Inst Navig 72:23–30 (in Japanese)
6. Imazu H, Koyama T (1985) The optimization of the criterion for collision avoidance action-III. J Jpn Inst Navig 73:19–26 (in Japanese)
7. Kouzuki A, Hasegawa K (1987) Automatic collision avoidance system for ships using fuzzy control. J Kansai Soc Nav Archit Jpn 205:1–10
8. Hu L, Naeem W, Rajabally E, Watson G, Mills T, Bhuiyan Z, Salter I (2017) COLREGs-compliant path planning for autonomous surface vehicles: a multiobjective optimization approach. IFAC Pap OnLine 50(1):13662–13667 (**20th IFAC World Congress**)
9. Kuwata Y, Wolf MT, Zarzhitsky D, Huntsberger TL (2014) Safe maritime autonomous navigation with COLREGS, using velocity obstacles. IEEE J Ocean Eng 39(1):110–119
10. Nagasawa A, Hara K, Inoue K, Kose K (1993) The subjective difficulties of the situation of collision avoidance-ii : Toward the rating by simulation. J Jpn Inst Navig 88:137–144 (**In Japanese**)
11. Nakamura S, Okada N (2019) Development of automatic collision avoidance system and quantitative evaluation of the manoeuvring results. TransNav Int J Mar Navig Saf Sea Transport 13(1):133–141
12. Woerner K, Benjamin MR, Novitzky M, Leonard JJ (2019) Quantifying protocol evaluation for autonomous collision avoidance. Auton Robots 4:967–991
13. Mitsubori K, Kamio T, Tanaka T (2004) Finding the shortest course of a ship based on reinforcement learning algorithm. J Jpn Inst Navig 110:9–18
14. Tanigawa T, Kamio T, Mitsubori K, Tanaka T, Fujisaka H, Haeiwa K (2013) Modified multi-agent reinforcement learning system to find ships' courses. In: The 2013 proceedings of international symposium on nonlinear theory and its applications, vol 2, pp 487–490
15. Nakayama M, Kamio T, Mitsubori K, Tanaka T, Fujisaka H (2014) Reinforcement learning based search for ships' courses controlled by safety. In: The 2014 proceedings of international symposium on nonlinear theory and its applications, pp 28–31
16. 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
17. Sawada R (2020) Automatic collision avoidance using deep reinforcement learning with grid sensor. In: Proceedings of the 23rd Asia pacific symposium on intelligent and evolutionary systems. pp 17–32
18. Imazu H (2017) Evaluation method of collision risk by using true motion. TransNav Int J Mar Navig Saf Sea Transport 11(1):65–70
19. Imazu H (1987) Research on collision avoidance manoeuvre. Ph.D. thesis, The University of Tokyo (**In Japanese**)
20. Fujii Y (1980) A definition of the evasive domain. Navigation 65:17–22 (**In Japanese**)
21. Inoue K (2011) Theory and practice of ship handling. Seizando-shoten publishing, Tokyo (**In Japanese**)
22. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI gym. arXiv:1606.01540
23. Hayama I (2014) Computation of OZT by using collision course. Navigation 188:78–81 (**In Japanese**)
24. Tam C, Bucknall R (2010) Collision risk assessment for ships. J Mar Sci Technol 15:257–270
25. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. CoRR. arXiv:abs/1707.06347
26. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: an imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32 (NIPS 2019). Curran Associates, Inc., pp 8026–8037
27. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization. In: Proceedings of the 32nd international conference on machine learning. Volume 37 of proceedings of machine learning research, Lille, France, PMLR, pp 1889–1897
28. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of The 33rd international conference on machine learning. Volume 48 of proceedings of machine learning research, PMLR, pp 1928–1937
29. Nomoto K (1960) Analysis of kempf's standard maneuver test and proposed steering quality indices. In: Proceedings of the 1st symposium on ship manoeuvrability. pp 275–304
30. Fukuto J, Imazu H, Kobayashi E, Arimura N (2002) Report of field experiments of AIS. Navigation 151:73–78 (**In Japanese**)
31. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the 3rd international conference on learning representations (ICLR 2015), arXiv:abs/1412.6980
32. Woerner KL, Benjamin MR, Novitzky M, Leonard JJ (2016) Collision avoidance road test for COLREGS-constrained autonomous vehicles. In: OCEANS 2016 MTS/IEEE Monterey. pp 1–6
33. Cai Y, Hasegawa K (2013) Evaluating of marine traffic simulation system through imazu problem. Proc Jpn Soc Nav Archit Ocean Eng 17:191–194