Ocean Engineering 189 (2019) 106299

Contents lists available at ScienceDirect

Ocean Engineering

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

A knowledge-free path planning approach for smart ships based on reinforcement learning

Chen Chen[a,b,d], Xian-Qiao Chen[a,b], Feng Ma[c,*], Xiao-Jun Zeng[d], Jin Wang[e]

[a] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
[b] Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan, China
[c] Intelligent Transportation System Center, Wuhan University of Technology, Wuhan, China
[d] School of Computer Science, University of Manchester, Manchester, UK
[e] Liverpool Logistics, Offshore and Marine (LOOM) Research Institute, Liverpool John Moores University, Liverpool, UK

ARTICLE INFO

Keywords:
Q-learning
Path planning
Smart ships
Rewards
Value function

ABSTRACT

The autonomous navigation of smart ships needs to meet their huge inertia and obey existing complex rules. A smart ship has to realise autonomous driving instead of manual operation, which consists of path planning and controlling. Toward to this goal, this research proposes a path planning and manipulating approach based on Q-learning, which can drive a cargo ship by itself without requiring any input from human experiences. At the very beginning, a ship is modelled with the Nomoto model in a simulation waterway. Then, distances, obstacles and prohibited areas are regularized as rewards or punishments, which are used to judge the performance, or manipulation decisions of the ship. Subsequently, Q-learning is introduced to learn the action–reward model and the learning outcome is used to manipulate the ship's movement. By chasing higher reward values, the ship can find an appropriate path or navigation strategies by itself. After a sufficient number of rounds of training, a convincing path and manipulating strategies will likely be produced. By comparing the proposed approach with the existing methods, it is shown that this approach is more effective in self-learning and continuous optimisation, and therefore closer to human manoeuvring.

1. Introduction and background

Since the 1970s, the combination of robot technologies and vehicles has led to the emergence of drones, unmanned vehicles and unmanned boats (Seuwou et al., 2017). Among them, the research of unmanned vehicles may be comprehensive. In contrast, there is much less research on unmanned ships, especially for cargo ships. Cargo ships are generally under-actuated due to their large tonnage, slow speed and relatively weak power. The autonomous navigation of cargo ships has to meet huge inertia and complex navigation rules, therefore the requirements for smart ships are much higher than those for unmanned vehicles. An operator of a cargo ship faces many challenges, including those associated with the dynamic environment, insufficient power and the uncertainties in perception. Hence, the artificial intelligence of cargo ship operating is considered to be very difficult to be built, the core function of which is path planning.

In the field of robotics, path planning is a prolonged topic. Artificial Potential Field (APF), A*, dynamic A*, Rapid-exploring Random Tree (RRT) and other algorithms have been studied and developed for many years. However, the above methods are generally based on models such as potential fields, a rule tree, or a probability tree. In general, these methods primarily take into account distances, smoothness and accessibility of paths without considering the dynamic characteristics of the corresponding agent. Therefore, the routes obtained by these planning methods often do not meet the requirements of ship kinematics and safety. In other words, traditional path planning methods could not be practical in the navigation of cargo ships.

At present, the development of artificial intelligence technology, especially the reinforcement learning, provides a new possibility to satisfy the requirements of the path planning of intelligent cargo ships. Reinforcement learning has attracted extensive attention in recent years, which emphasizes the learning of agents from the environment to behaviour mapping and seeks for the most correct or best action decision by maximizing value functions. Q-learning, Deep Q-Network DQN, A3C and Deep Deterministic Policy Gradient DDPG are most frequently used reinforcement learning methods (Gupta et al., 2017). Different from

* Corresponding author.
E-mail address: feng.ma.whut@gmail.com (F. Ma).

https://doi.org/10.1016/j.oceaneng.2019.106299
Received 15 January 2019; Received in revised form 1 June 2019; Accepted 7 August 2019
Available online 27 August 2019

other artificial intelligence algorithms, reinforcement learning methods do not need any human knowledge or pre-set rules. Considering the navigation difficulty for cargo ships and the benefits offered by reinforcement learning, in this research, Q-learning is introduced to address the problem of path planning for a cargo ship. The essence of path planning based on the Q-learning system is that agents can independently find the most effective path by enumerating possible solutions, which might be closer to human manipulating intelligence. The only prerequisite is to build a computing environment which is consistent with or close to the real world. Via the proposed approach, this research not only provides a novel way for smart ships to sail autonomously by considering all the characteristics of a cargo ship, but also presents a new application scenario for reinforcement learning.

The above idea is realised and implemented in this paper. Firstly, a ship agent is introduced in a simulation waterway. Based on Q-learning, an action-return mode is adopted for the evaluation and calculation of path under the constraints of safety and efficiency. Then, an appropriate path can be found after sufficient training.

The remaining part of this paper is organised as follows. Section 2 reviews relevant literatures. Then Section 3 proposes a Q-learning based path planning approach. After that, Section 4 validates the approach in simulations. Finally, the conclusion on the use of the approach is given in Section 5.

2. Literature review

2.1. Path planning methods for ships

Compared with the path planning in the field of robots and manipulators on lands, there are only a limited number of studies on the path planning of cargo ships. For many years, the A* algorithm has been the dominant approach in the relevant research. A Swiss boat named Avalon (Erckens et al., 2010) is capable of generating a persuasive path to a given destination and avoiding both static and dynamic obstacles based on the A* algorithm. Langbein (Langbein et al., 2011) from Ulm University made use of the A* algorithm to develop a long-term path routing approach for autonomous sailboats. In particular, such an approach had been validated in a model test. Li et al. (2017) developed an Autonomous Underwater Vehicle (AUV) optimal path planning method on a basic of A* for seabed terrain matching navigation to explore the underwater world. In short, A* is a popular choice for path finding, which is flexible. The algorithm given in (Hart et al., 1968) uses a heuristic function to estimate the distance from the current point to the end in the graph and determine its searching direction accordingly. Although A* is very efficient for avoiding static obstacles, it might not be very suitable for dynamic environments. Apart from the A* algorithm, APF is another popular method in the path planning. Ma and Chen (2018) adopted the APF model to describe the collision potential caused by buoys, piers and encountered vessels and then estimated the collision probabilities. Xue et al. (2011) applied an APF-based method in ship automatic navigation, which can find a promising route and avoid collision. In general, the APF model is elegant and considered to be practical in path planning. However, the major issue of the APF model is that the agent might fall into local minima. In this occasion, it is very difficult for the agent to reach its destination. Chen (Chen et al., 2018) developed an extension of RRT algorithm to overcome the actual demand of multi-waypoint path planning for unmanned ship. The RRT method is a sampling-based expansion algorithm. At each step of the tree-growth, given the generated sample (random seed), the growth of the tree is attracted to this seed until a certain branch of the tree can reach the destination (Dong et al., 2017). In addition to A* and APF, other methods are also used in Unmanned Surface Vessel (USV) and AUV. Alvarez et al. (2004) presented a genetic algorithm (GA) for path planning of an AUV, which turns out to be efficient. However, the computing speed of this GA based approach is too slow to meet the requirement of real-time updating. Cheng and Liu (2007) applied a genetic annealing algorithm to trajectory optimisation based on the ship dynamic collision avoidance space model. Petres et al. (2007) proposed a novel fast marching-based approach for the path planning of AUV, which takes the control constraints of AUV into consideration in two-dimensional calculation. Liu and Bucknall (2015) suggested a constrained fast marching method to solve the problem of USV formation path planning in dynamic environments. Dynamic Window Approach (DWA) and Nearness diagram (ND), are also popular approaches in the path planning or collision avoidance for robots. However, they are not very common in the applications of vessels (Fox et al., 1997; Minguez and Montano, 2003).

2.2. Reinforcement learning

As discussed previously, although these algorithms have their own advantages in path planning, none of them fully take consideration of the dynamic characteristics and path rationality. Therefore, this research introduces the Q-learning algorithm to address the problem of path planning, which considers the path planning as a continuous optimisation problem under the trade-off between gain and loss. In 1956, Bellman (1956) proposed a dynamic programming method, which laid the foundation of reinforcement learning. Watkins (Watkins and Dayan, 1992) put forward Q-learning in 1989, which is the most commonly used reinforcement learning algorithm. Q-learning is a form of model-free reinforcement learning, which can also be viewed as a method of asynchronous dynamic programming (DP). The model of Q-learning is very elegant and refined. It provides agents with the capability of learning to act optimally by experiencing the consequences of actions, without requiring them to build maps of the domains.

Q-learning was known as a Markov decision process (Bellman, 1957) with the environment fully observable, which is described as the variables below (Sato et al., 1988):

$\boldsymbol{S}$ is a finite set of possible states.
$\boldsymbol{A}$ is a finite set of actions.
$\boldsymbol{P}$ is a state transition probability matrix, where $\boldsymbol{P}(\boldsymbol{S}_t, \boldsymbol{S}_{t+1}, \boldsymbol{a}_t)$ is the probability of arriving in state $\boldsymbol{S}_{t+1}$ when performing action in state $\boldsymbol{S}_t$
$\boldsymbol{R}$ is a reward function.
$\boldsymbol{\gamma}$ is a discount rate, $0 \le \boldsymbol{\gamma} \le 1$.
$\boldsymbol{\pi} : \rightarrow \boldsymbol{S} \times \boldsymbol{A}$ is a state transaction function.

In Q-learning, a series of different stages or episodes constitute the experience of an agent. In the t^{th} episode, the agent has the following steps: 1) Observe its current state $\boldsymbol{S}_t$, 2) Select and perform an action $\boldsymbol{a}_t$, 3) Observe the subsequent state $\boldsymbol{S}_{t+1}$, 4) Receive an immediate return value $\boldsymbol{R}_t$, 5) The Q value is adjusted by using the learning factor $\boldsymbol{\alpha}_n$, and the Q value is obtained according to the following formula:

$$\boldsymbol{Q}(\boldsymbol{S}_t\ ,\ \boldsymbol{a}_t) \leftarrow \boldsymbol{Q}(\boldsymbol{S}_t\ ,\ \boldsymbol{a}_t) + \boldsymbol{\alpha}[\boldsymbol{R}_t + \boldsymbol{\gamma}\boldsymbol{Q}(\boldsymbol{S}_{t+1}, \boldsymbol{a}_{t+1}) - \boldsymbol{Q}(\boldsymbol{S}_t, \boldsymbol{a}_t)] \tag{1}$$

where $\boldsymbol{Q}(\boldsymbol{S}_t\ ,\ \boldsymbol{a}_t)$ is the value of an action $\boldsymbol{a}_t$ executed by the agent; $\boldsymbol{R}_t$ is the immediate reward; $\boldsymbol{\alpha}$ **is** a learning rate and $\boldsymbol{\gamma}$ is the discount factor.

When this process is sufficient, the agent generates a memory that can select actions to maximize the total future reward in different states. Q-learning combines Monte Carlo sampling and Dynamic Programming bootstrapping. Therefore, it can learn online after each step and from incomplete sequence, with low variance and high efficiency. Compared with the path planning methods briefly described previously, Q-learning has been applied in complex and challenging environments, such as enlarged state spaces, increased computational complexity, energy requirements, safety issues and many other constraints. Demircan et al. (2011) took advantage of Q-learning to find out an optimum route for electrical energy transmission lines under specified criteria. Konar et al. (2013) provided a Q-learning-based path planning algorithm for a mobile robot with respect to time, the number of states traversed and energy consumption. Zolfpour et al. (Zolfpour-Arokhlo et al., 2014) presented a multi-agents reinforcement learning model for a route planning system to address the vehicle delay problems by studying the weights of various components in road network environments such as weather, traffic, road safety, and fuel capacity to create a priority route plan for vehicles between cities of Malaysia. Li (Li et al., 2008) presented a reinforcement learning algorithm using linear function approximation to generate an optimal path by controlling the choice of four moving actions of the microrobot. Zhang et al. (2009) developed a reinforcement learning path-following control strategy based on approximate policy iteration for a real mobile robot. Compared with traditional methods, this method offers better convergence and a higher path tracking accuracy.

There are very few applications of reinforcement learning in the field of waterway. Gaskett et al. (1999) used the Q-learning algorithm to make AUV control its propeller according to input commands and sensor information in order to find the target autonomously. El-Fakdi and Carreras (2013) demonstrated the feasibility of reinforcement learning techniques in underwater cable tracking tasks for AUV. Yoo and Kim (2016) used reinforcement learning for marine vehicles path optimisation in ocean environments without considering obstacles. Yin Cheng and Weidong Zhang (Cheng and Zhang, 2018) realised the obstacle avoidance for unmanned ships based on DQN, and the vessel had no knowledge of the disturbance of the environment. In this method, only a small neighbouring area is taken into consideration during the navigation that makes the whole path planning elegant and concise.

From the above discussion, it can be concluded that there is very little research on automatic manoeuvring of ships by using reinforcement learning. The existing methods of ship route planning are traditional ones in the field of robots, which might not take all the factors, including dynamic characteristics, efficiency and rules of ships, into consideration. Reinforcement learning is a trial-and-error learning algorithm, in which the agent interacts with the environment in real-time and tries constantly to obtain an appropriate strategy. Based on an appropriate reward function, such a method can be used to address the problem that needs to make continuous decisions to achieve an objective. The path planning and control of ships are such problems that require continuous decision-making. Therefore, this research proposes reinforcement learning to address the problems. In particular, taking into account the ship kinematics equation in the learning process, it enables the automatic self-driving of ships, filling the gap between reinforcement learning and the manoeuvring of smart cargo ships.

3. A proposed approach

The objective of this research is not to find a shortest path for a cargo ship, which often is not an appropriate path for the corresponding ship, but to find a practical path which takes into consideration of the dynamic characteristic and a certain criterion. This problem can be addressed by Q-learning, which enables a smart ship agent to learn convincing actions in a dynamic environment with no prior-knowledge. To make the simulation agree with the dynamic characteristics of a cargo ship, the given research uses the first-order Nomoto model to establish the initial state Q-table. Based on the risk of the ship collision with an obstacle in simulation, some other regulations, a reward function is formulated. Subsequently, the Q-learning principle is used for repeated training. After sufficient training, the artificial intelligence of path planning is established, and a meaningful and reasonable path can be found. To verify the performance of the proposed approach, such a path is compared with those obtained the traditional A * and RRT algorithms.

3.1. Modelling of the Q-table

In this research, a state space of the ship agent is formulated based on its position and heading. In addition, an action space is associated with its rudder angle. It is known that the position and heading of the corresponding ship in a certain time can be inferred by the first-order Nomoto model. Such a model was proposed in 1957 by Nomoto et al. (1957) who regarded various motion changes of a ship caused by the steering as the response relationship. That is, the input is rudder angle and the output is the ship motion change. Based on such a relationship, the first order Nomoto equation is then proposed as the first order *KT* equation, which can be expressed by.

$$T\dot{r} + r = K\delta \tag{2}$$

with the notation

$$\dot{\psi} = r \tag{3}$$

Where ψ is the heading of a ship, Eq. (2) can be written as

$$T\ddot{\psi} + \dot{\psi} = K\delta \tag{4}$$

where K is the turning ability coefficient, T is the turning lag coefficient, r is the yaw rate, and δ is the rudder angle.

The first order Nomoto model has been widely employed in simulating the movement of ships. The yaw dynamics is characterized by coefficients K and T, which can be identified from standard manoeuvring tests (Liu et al., 2017).

Taking the ship as a rigid body, when the ship steers at any rudder angle δ, the bow of the ship rotates at a certain angle and the yaw rate is r. The above formula can be seen as the bow rotation equation of the ship when it steers.

Assuming that the initial conditions are $t = 0$, $\delta = \delta_0$ and $r = 0$, then yaw rate r at any moment t can be computed through the first-order KT equation (Zhang et al., 2012).

$$r = K\delta_0\left(1 - e^{-t/T}\right) \tag{5}$$

Since the yaw rate r is actually the time derivative of Ψ, ship heading angle Ψ can be obtained as

$$\Psi = K\delta_0\left(t - T + T\cdot e^{-t/T}\right) \tag{6}$$

In order to describe the motion of a ship, a ship motion coordinate system is established, as shown in Fig. 1. In this figure, G represents the position of the centre of gravity of the ship, XOY indicates the hydrostatic water plane, O is the origin, X_{OG} and Y_{OG} represent the projection of the centre of ship gravity G on the X and Y axes respectively, ψ is the heading of ship, and δ indicates the ship rudder angle. The position of

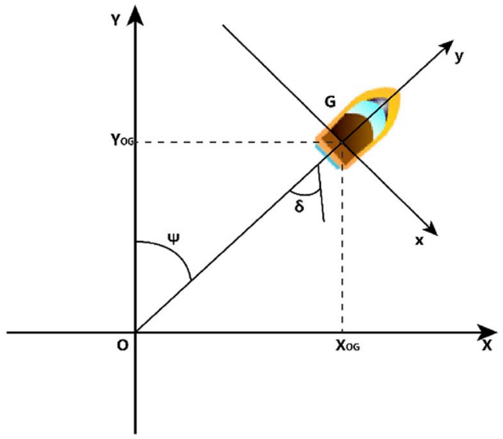

Fig. 1. A ship motion coordinate system.

$$\begin{cases} x(t) = x(0) + \int v\cdot\sin\psi dt \\ y(t) = y(0) + \int v\cdot\cos\psi dt \end{cases} \tag{7}$$

the ship at any moment can be calculated by Eq. (7).where $x(0)$ *and* $y(0)$ are the initial positions, v is the ship speed, and ψ is the heading of ship. With the help of these equations above, the position and heading can always be inferred.

To enable the reinforcement learning, in this research, the return function is defined as follows. If the distance from the obstacle is greater than the minimum collision distance, it is set to 1. If the distance from the obstacle is smaller than the minimum collision distance, it is set to –1000. In this occasion, the system will quit and start learning again. This is also in line with the navigation rules. In this way, it is possible to obtain the corresponding return value of the ship in any state after taking any action, so as to establish the Q-table of the corresponding ship.

3.2. Learning process

Through the modelling in Section 3.1, the selection of ship actions and the state of the next step can be linked by the first-order KT model. Hence, Q-learning can be introduced to establish autonomous path planning intelligence. The learning process of Q-learning can be designed to start from any initial state and then select actions according to an action policy. After taking the selected action, the agent observes the following state and finds the reward, and then updates the Q-value of the previous state and action based on the maximum Q-value and reward of the new state. The Q-learning action policy employs the ε-greedy policy, which balances "exploration" and "exploitation" (Sutton and Barto, 1998). Exploitation refers to select an action with the largest value function. Exploration, on the other hand, means that other attempting actions still have a chance to be chosen. This will make the agent learn experience from the environment, ensuring that the agent searches all possible actions to avoid being trapped in local optimal actions. The target policy is a greedy policy, which is also a deterministic policy. Only when the value function is maximized, the probability is 1 and the other action probability is set to be 0. The algorithm is given as follows:

Algorithm 1. Q-learning Algorithm

In summary, the Q-learning algorithm presented stores Q-values at each state for the optional action. When the learning process is completed, all the states are determined; the Q-table can be used for the coming path-planning applications.

4. A case study

To verify the effectiveness of the proposed method, the PyCharm platform was used to establish a simulation environment of a waterway. In this platform, a Q-learning-based path planning function was implemented by Python. Subsequently, the RRT and A* methods were also used to plan the paths respectively in the same platform and under the same scenario, and the results of the three methods were compared. For the given case study, it is worth highlighting that dynamic characteristics of a ship are different from those of a vehicle. In fact, a cargo ship always tries to maintain its speed and direction, since speed change and sharp turn would hurt its engine or could lead to capsize. Basically, the engine speed of a cargo ship stays on a constant value in most cases, and the rudder maintains a small angle in turning. In order to simulate the dynamic characteristics of a ship, it is fair to simplify the model, and make the assumption that the simulated ship always sails on an almost constant speed.

4.1. An experimental platform

This research established a virtual navigation environment for ships, which is shown in Fig. 2. The size of this map is set to 800 × 600, where the bottom-left corner is taken as the origin (0, 0). One pixel stands for 4.71 m in the real world. In this map, the orange area is considered as the land, and the blue area is considered as a waterway, which is 400 in width, and 600 in length. It should be pointed out that the left-bottom marginal point of this waterway is (200, 0). In this case study, there are 4 objective scenarios designed for this experiment. The four scenarios are labelled from 1 to 4, each of which represents the number of obstacles in the corresponding scenario. Therefore, there are 4 obstacles in Scenario 4, which is most complicated. The size of the obstacles is set to be 100 × 50. In Scenario 1, there is only obstacle 1 located at the lower left corner (300, 200). In Scenario 2, obstacle 2 is added at (400, 450). Similarly, obstacle 3 has been added in Scenario 3 at (350, 325), obstacle 4 has been added in Scenario 4 at (400, 100). At the same time, a cargo ship is simulated with a length L of 59 and a width W of 30. Certainly other sizes are also applicable in this test. The initial position of this ship is (400, 30), and its initial heading angle $\boldsymbol{\Psi}$ is set to zero.

Different from a land robot, cargo ships must overcome the problems of huge inertia and weak driving power. Meanwhile, the speed of a cargo

Algorithm 1 *Q-learning Algorithm*

```
1:  Initialize $\mathbf{Q(s,a)}$, $\forall \mathbf{s} \in \mathbf{S}, \mathbf{a} \in \mathbf{A(s)}$, arbitraily
2:  Assign a learning factor $\boldsymbol{\alpha}$, a discount factor $\boldsymbol{\gamma}$
3:  Repeat (for each episode)
4:      Initialize state $\boldsymbol{S_t}$
5:      Repeat (for each step of episode)
6:          Choose $\boldsymbol{a_t}$ from $\boldsymbol{S_t}$ according to ε-greedy policy,
7:          Take action $\boldsymbol{a_t}$, observe $\boldsymbol{r_t}$ and next state $\boldsymbol{S_{t+1}}$
8:          Update Q-value by using Eq. (1)
9:          $\boldsymbol{S}$=$\boldsymbol{S'}$, $\boldsymbol{a} = \boldsymbol{a'}$
10:     Until $\boldsymbol{S}$ is terminal
11: Until all $\boldsymbol{Q(s,a)}$ is of convergence
12: Output the last policy:
        $\pi(s) = \underset{a}{argmax}\, Q(s,a)$
```

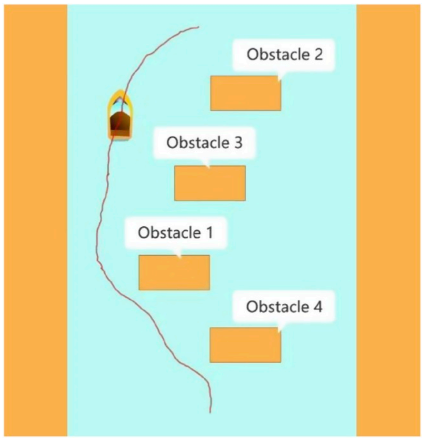

Fig. 2. Path taken by Q-learning algorithm.

ship cannot be changed easily. Sharp turns in narrow spaces are not possible in practice. Therefore, cargo ships generally need large open waters for operating. In the experiment shown in Fig. 2., the available spaces are generally 5–8 time the length of the ship itself. Such a scenario might be difficult enough for a cargo ship.

4.2. Path planning based on the proposed approach

To simplify the training process, the map was rasterized and divided into grids at the size of 5×5, and the heading is represented as 20 steps, as the interval is 18°. Hence, the state space of the ship is related to its coordinates and headings, which can be represented by

$$[(x_1, y_1, \psi_1), (x_2, y_2, \psi_2), \cdots, (x_n, y_n, \psi_n)] \tag{8}$$

Where x is the X-coordinate, y is the Y-coordinate and ψ is the heading of the ship.

As discussed previously, the speed of the simulated ship can be set to be constant. Therefore, the rudder angle is the only action option for the ship. In fact, the steering angle of a cargo ship generally does not have many choices, which is between ±35°. The Q-learning algorithm requires the action space of an agent being discrete, and therefore the discretization process is applied. Without loss of generality but simplifying the model and computing, the action space of the ship is set to 5 options, [-35, −15, 0, 15, 35]. That is, the ship can select only five rudder angles in any state in this case study.

Taking Scenario 4 (the most complicated one) of Fig. 2 as an example, according to the size of the map, the shape of the waterway and the dynamic characteristics of the cargo ship, and a rule that the ship cannot collide with the land and obstacles during the simulation; the reward value is defined by the following rules:

(1) If $300 - W/2 < x < 400 + W/2$, and $200 - L/2 < y < 250 + L/2$, the ship has collided with obstacle 1, $r = -1000$. The learning process will exit and re-start.
(2) If $400 - W/2 < x < 500 + W/2$, and $450 - L/2 < y < 500 + L/2$, the ship has collided with obstacle 2, $r = -1000$. The learning process will exit and re-start.
(3) If $350 - W/2 < x < 450 + W/2$, and $325 - L/2 < y < 375 + L/2$, the ship has collided with obstacle 3, $r = -1000$. The learning process will exit and re-start.
(4) If $400 - W/2 < x < 500 + W/2$, and $100 - L/2 < y < 150 + L/2$, the ship has collided with obstacle 4, $r = -1000$. The learning process will exit and re-start.
(5) If $x - W/2 < 200$, or $x + W/2 > 600$, the ship has collided with the land, $r = -1000$. The learning process will exit and re-start
(6) If $y - L/2 < 0$, or $y + L/2 > 600$, the ship has sail out of the map, $r = -1000$. The learning process will exit and re-start
(7) If $400 < x + W/2 < 450$, or $y + L/2 > 550$, the ship has reached the destination, $r = 1000$.
(8) Other situations, the ship can be considered as sailing properly in the simulation world, $r = 1$.

In this experiment, the ship manoeuvrability coefficients are given as follows: $K = 0.08$, $T = 10.8$. In fact, other K and T values are also applicable, which can be set up according to any specific vessel. If the loading has been changed, the corresponding K and T should be altered accordingly. On the other hand, the learning factor α is set as 0.3 and the discount factor γ as 0.99; the variation of these values might change the learning speed and sometimes need some pre-analysis and simulation to determine. In this research, the learning factor α and the discount factor γ are assigned with typical values. The Q-table are updated every 20 s so as to reduce the computation, and the value function update formula can be presented as:

$$Q(S_t, a_t) \leftarrow Q(S_t, a_t) + 0.3 \times [R_t + 0.99 \times Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t)] \tag{9}$$

Table 1 gives a table of the value function of an episode, which is calculated every 20 s. For example, in the last state 14, $x - W/2 < 200$. It means that the ship collides with the shore and the value function is −300, the training ends and restarts.

During 300,000 times of learning, the value function table was continuously updated, and the policy with the largest value function was improved. Finally, the path planned by this algorithm is shown by the red line in Fig. 2. In fact, the learning can be implemented forever. The more training, the more satisfied result would be found.

4.3. Path planning taken rules and economic efficiency into consideration

In practice, navigation rules and economic efficiency are also vital in the navigation of a cargo ship. The unique advantage of Q-Learning or Reinforcement Learning algorithms is that these factors can be modelled as reward, hence the behaviours of agent can be altered accordingly.

In the same scenario, Q-learning, RRT and A* algorithms are used for path planning respectively. In fact, many kinds of improved RRT and A* have been proposed in the applications of vessels under different scenarios. It is difficult to enumerate them all in our research. Therefore, for length and simplicity reasons, only the original forms of RRT and A* are discussed in this research. Then the paths planned by the three methods under the four scenarios are shown in Figs. 3–6. In particular, the distance between the ship and the destination can be considered as a special kind of reward to generate a new path in these figures, which is represented as purple dash line, and denoted as Q-learning under distance constraints. In the corresponding algorithm, the distance L between the ship and the destination is been calculated by Eq. (10) at each state, where the destination is (x_{goal}, y_{goal}).

Table 1
The Q-table of an episode.

	x	y	ψ	Action	Reward
State1	396.05	49.34	−30.51	0	3.08
State2	385.90	66.57	−30.51	0	3.17
State3	375.74	83.80	−30.51	0	3.24
State4	365.59	101.03	−30.51	0	3.28
State5	359.18	119.70	0.00	−35	1.89
State6	355.23	139.04	−30.51	−15	1.72
State7	343.65	155.29	−43.58	−15	2.18
State8	328.70	168.50	−56.65	−15	4.52
State9	311.15	177.99	−69.73	15	4.59
State10	292.39	184.92	−69.73	0	0.02
State11	273.62	191.85	−69.73	35	0.08
State12	254.11	194.84	−100.23	0	0
State13	234.43	191.29	−100.23	0	0
State14	214.75	187.74	−100.23	0	−300

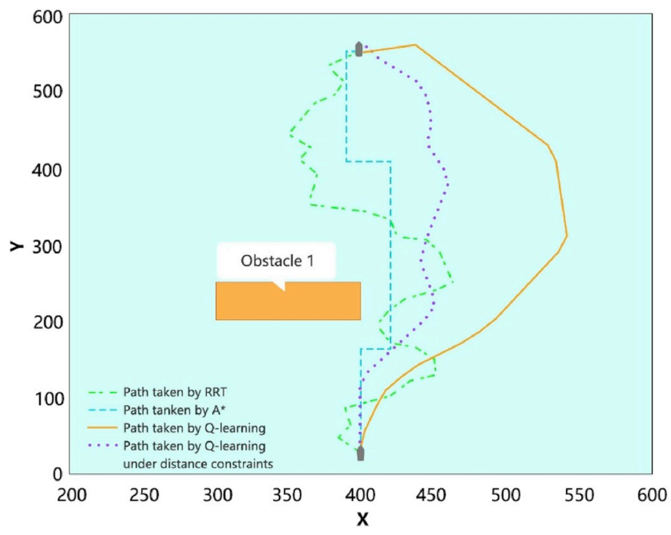

Fig. 3. Paths planned by different methods in Scenario 1.

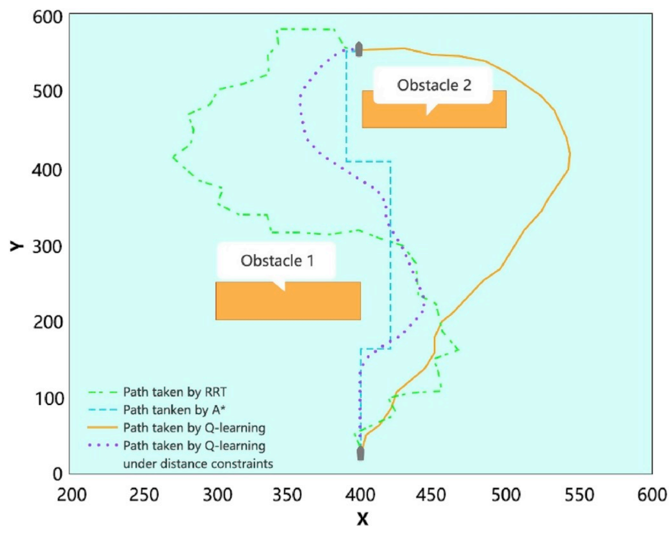

Fig. 4. Paths planned by different methods in Scenario 2.

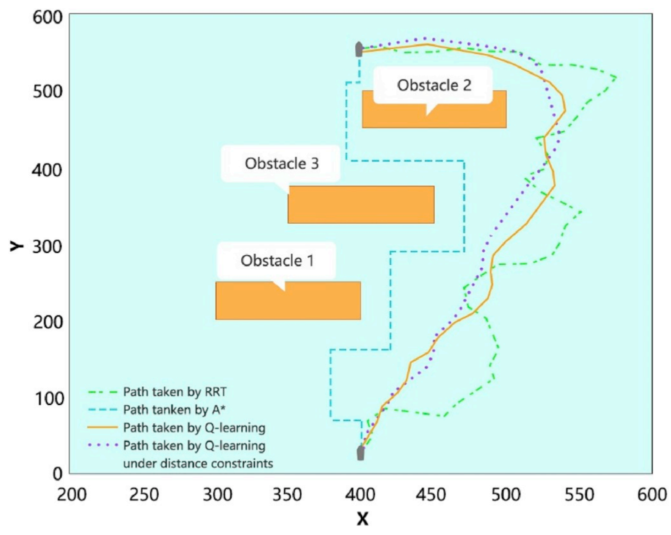

Fig. 5. Paths planned by different methods in Scenario 3.

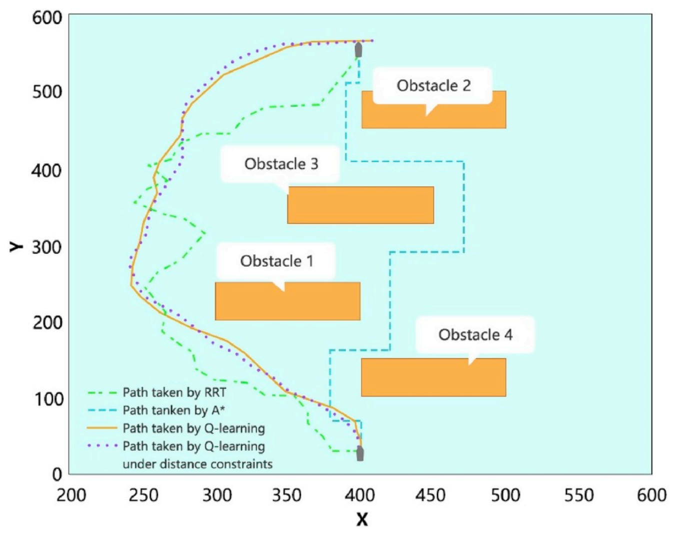

Fig. 6. Paths planned by different methods in Scenario 4.

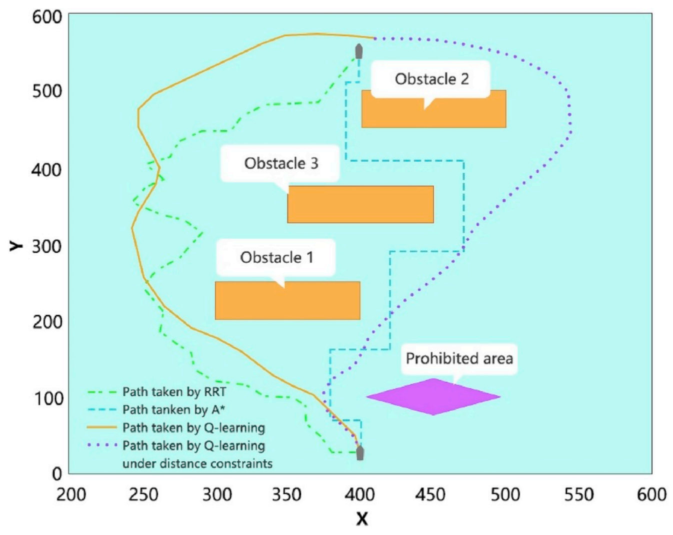

Fig. 7. Paths planned by different methods in Scenario 5.

$$L = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2} \tag{10}$$

In this occasion, path distance L expresses how close to the goal, which will cause a reward, where reward $r = 1\text{-}\ 0.003 \times L$ for each step.

In this research, the length of the paths, the number of sharp turns, the number of 90° turns planned by the three methods are compared respectively, and the results are shown in Table 2. Sharp turns refer to angles greater than 35° or angles less than −35°. Table 2 presents the distances, the number of sharp turns and the number of 90° turns produced by the three methods.

To validate that the proposed approach is also applicable in most scenarios, obstacles have been placed randomly presented in Figs. 8–11. Subsequently, the corresponding comparisons of performance are also given in Table 2.

Based on Table 2, it can be concluded that the paths planned by the A* algorithm are the shortest in Scenario 1 and Scenario 2, which are better than the ones produced by Q-learning. However, with the increasing number of obstacles, the path distance of the A* algorithm turns out to be longer than that produced by the Q-learning-based algorithms in Scenario 3 and Scenario 4. In addition, the paths of the A* algorithm in all scenarios contain many 90° sharp turns. For cargo ships,

Table 2
Comparisons among different methods.

Scenarios	Methods	The Length of Path	No. of sharp turns	No. of 90° turns
Scenario 1	A*	580	5	5
	RRT	690.72	16	0
	Q-learning	631.47	0	0
	Q-learning under distance constraints	558.88	0	0
Scenario 2	A*	600	5	5
	RRT	838.53	15	0
	Q-learning	643.55	0	0
	Q-learning under distance constraints	577.67	0	0
Scenario 3	A*	680	8	8
	RRT	831.25	18	0
	Q-learning	663.99	0	0
	Q-learning under distance constraints	662.79	0	0
Scenario 4	A*	720	10	10
	RRT	733.72	16	0
	Q-learning	677.13	0	0
	Q-learning under distance constraints	676.81	0	0
Scenario 5	A*	720	10	10
	RRT	733.72	16	0
	Q-learning	697.32	0	0
	Q-learning under distance constraints	677.62	0	0
Scenario 6	A*	760	21	21
	RRT	755.03	14	0
	Q-learning	676.33	0	0
	Q-learning under distance constraints	636.53	0	0
Scenario 7	A*	660	11	11
	RRT	725.01	10	0
	Q-learning	697.01	0	0
	Q-learning under distance constraints	637.76	0	0
Scenario 8	A*	760	7	7
	RRT	725.65	12	0
	Q-learning	677.76	0	0
	Q-learning under distance constraints	637.59		
Scenario 9	A*	760	11	11
	RRT	754.75	8	0
	Q-learning	656.48	0	0
	Q-learning under distance constraints	617.07	0	0

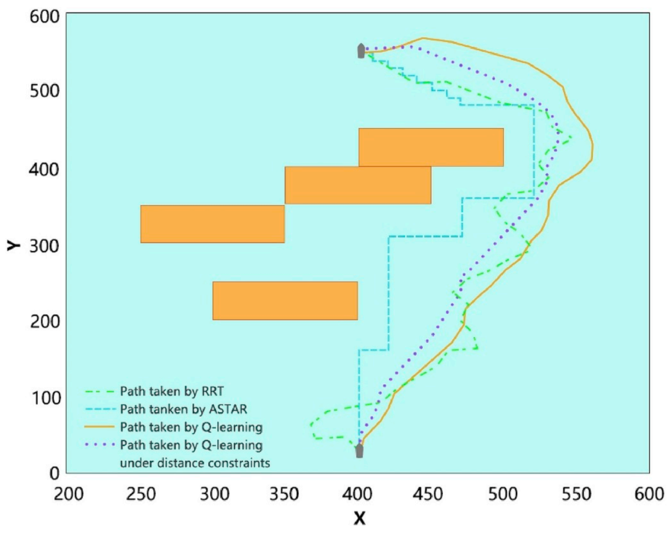

Fig. 8. Paths planned by different methods in Scenario 6.

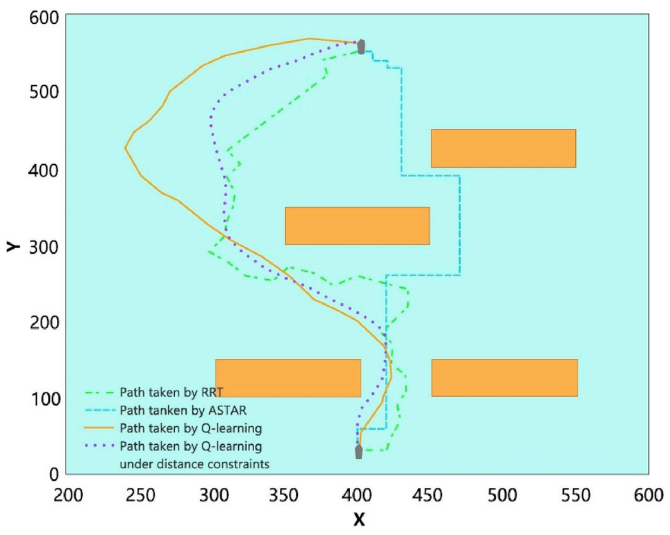

Fig. 9. Paths planned by different methods in Scenario7.

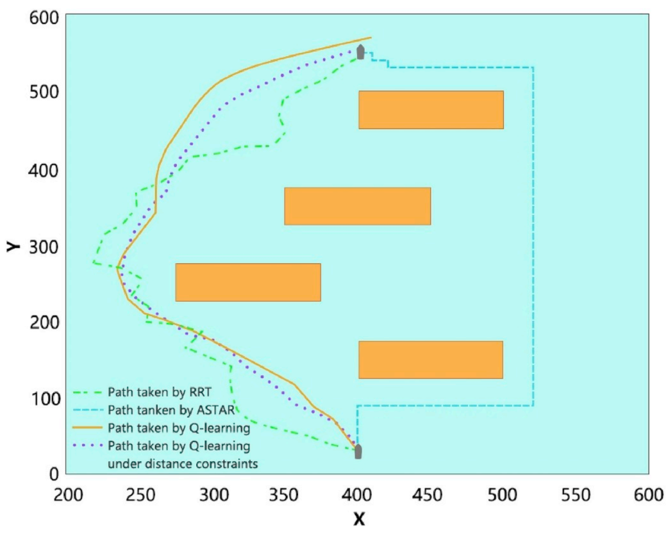

Fig. 10. Paths planned by different methods in Scenario 8.

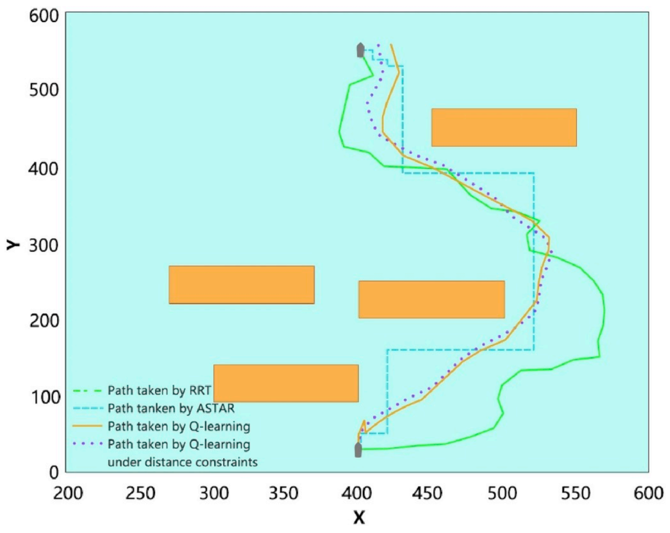

Fig. 11. Paths planned by different methods in Scenario 9.

such sharp turns are not practical or cannot be realised. Although there is no 90° turn in path distance produced by the RRT algorithm, it requires the ship steering the rudder at sharp degrees all the time, which does not conform to the reality of ship manipulating. Therefore, the path planned by the Q-learning algorithms proposed in this research is a realistic and feasible one, and therefore superior to the path produced by the RRT algorithm.

It is worth noting that the path length of Q-learning is the shortest in Scenarios 3, 4, 5, 6, 8 and 9. With the increasing of obstacles, the superiority of Q-learning algorithm becomes increasingly obvious. At the same time, this method has already taken into account the dynamic characteristics of the ship in its learning process. In fact, the output of the proposed approach not only provide a practical path for the corresponding ship, but also the sequential manipulating strategies. In other words, the controlling process of this ship has been completed simultaneously. Therefore, the path produced by this method can be easily implemented in real applications and enables the self-adjustment with the environments where the ship operates.

In particular, Scenario 5 is presented to simulate a more authentic waterway, where a prohibited area is placed (see Fig. 7). In all scenarios, the algorithm under length or distance constraints will always produce a shorter path for a ship than the basic Q-learning, even such a path might be closer to obstacles. By increasing the reward of shortening distances, such a tendency is clearer. In other words, the Q-Learning might be practical to simulate the balance or the objective trade-off of experienced operators.

In fact, Q-Learning has a drawback that it consumes much time for training. However, with the development of computer technology, such a drawback might be less obvious. In these experiments, all the training converges in less than 3 h on a normal personal computer. It can be inferred that such a procedure can be finished on a powerful server in very short time, which is applicable on a low-speed large cargo.

5. Conclusions and discussions

This research proposed a novel approach of ship path planning based on the Q-learning algorithm. To make the approach practical, the first-order Nomoto equation was used, by which the following position and heading angle of the ship can be inferred according to the present position, rudder angle and heading of the ship. Moreover, the position, rudder angle and heading of the ship are also the fundamental factors of the state space and action space for the ship. According to the characteristics of the simulation waterway, the action reward value was defined and the value function updating formula was proposed. In particular, distance to the destination, prohibited areas, and other rules can be modelled as in the reward function to simulate the trade-off of human. In the learning or training, the value function table was continuously updated. Finally, a rational path can always be found. The feasibility and effectiveness of the proposed approach was validated and compared with the traditional path planning methods, A* and RRT algorithms. This research provides an effective route planning method for ships manoeuvring.

In the simulation process of this experiment, there is no other dynamic obstacles such as ships in the waterway, which should be taken into consideration in future research. Moreover, DQN or other strategy gradient methods which are considered to be capable of addressing the problem of the space explosion are worth investigation and experiment. Furthermore, ship collision avoidance rules could be taken into consideration in the ship agent model reward function during the process of learning.

Acknowledgements

The first author is financed by National Key R&D Program 2018YFB1601503, China. This research is also supported by the project of Intelligent Ship Testing and Verification, 2018/473, China.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.oceaneng.2019.106299.

References

Alvarez, A., Caiti, A., Onken, R., 2004. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. IEEE J. Ocean. Eng. 29, 418–429.
Bellman, R., 1957. A Markovian decision process. J. Math. Mech. 679–684.
Bellman, R., 1956. Dynamic programming and Lagrange multipliers. Proc. Natl. Acad. Sci. 42, 767–769.
Chen, X., Liu, Y., Hong, X., Wei, X., Huang, Y., 2018. Unmanned ship path planning based on RRT. In: International Conference on Intelligent Computing. Springer, pp. 102–110.
Cheng, X., Liu, Z., 2007. Trajectory optimization for ship navigation safety using genetic annealing algorithm. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. IEEE, pp. 385–392.
Cheng, Y., Zhang, W., 2018. Neurocomputing Concise deep reinforcement learning obstacle avoidance for underactuate d unmanne d marine. vessels 272, 63–73. https://doi.org/10.1016/j.neucom.2017.06.066.
Demircan, S., Aydin, M., Durduran, S.S., 2011. Finding optimum route of electrical energy transmission line using multi-criteria with Q-learning. Expert Syst. Appl. 38, 3477–3482.
Dong, Y., Zhang, Y., Ai, J., 2017. Experimental test of unmanned ground vehicle delivering goods using RRT path planning algorithm. Unmanned Syst. 5, 45–57.
El-Fakdi, A., Carreras, M., 2013. Two-step gradient-based reinforcement learning for underwater robotics behavior learning. Robot. Auton. Syst. 61, 271–282.
Erckens, H., Büsser, G.-A., Pradalier, C., Siegwart, R.Y., 2010. Navigation strategy and trajectory following controller for an autonomous sailing vessel. IEEE RAM 17, 47–54.
Fox, D., Burgard, W., Thrun, S., 1997. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine 4 (1), 23–33. https://doi.org/10.1109/100.580977.
Gaskett, C., Wettergreen, D., Zelinsky, A., 1999. Reinforcement learning applied to the control of an autonomous underwater vehicle. In: Proceedings of the Australian Conference on Robotics and Automation (AuCRA99).
Gupta, J.K., Egorov, M., Kochenderfer, M., 2017. Cooperative multi-agent control using deep reinforcement learning. In: International Conference on Autonomous Agents and Multiagent Systems. Springer, pp. 66–83.
Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107.
Konar, A., Goswami, I., Singh, S.J., Jain, L.C., Nagar, A.K., 2013. A deterministic improved Q-learning for path planning of a mobile robot. IEEE Trans. Syst. Man, Cybern. Syst. 43, 1141–1153.
Langbein, J., Stelzer, R., Frühwirth, T., 2011. A Rule-Based Approach to Long-Term Routing for Autonomous Sailboats. Springer Berlin Heidelberg.
Li, J., Li, Z., Chen, J., 2008. Reinforcement learning based precise positioning method for a millimeters-sized omnidirectional mobile microrobot. In: International Conference on Intelligent Robotics and Applications. Springer, pp. 943–952.
Li, Y., Ma, T., Chen, P., Jiang, Y., Wang, R., Zhang, Q., 2017. Autonomous underwater vehicle optimal path planning method for seabed terrain matching navigation. Ocean Eng 133, 107–115.
Liu, Y., Bucknall, R., 2015. Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. Ocean Eng 97, 126–144.
Liu, Y., Noguchi, N., Ali, R.F., 2017. Simulation and test of an agricultural unmanned airboat maneuverability model. Int. J. Agric. Biol. Eng. 10, 88–96.
Ma, F., Chen, Y.-W., 2018. Probabilistic assessment of vessel collision risk: an evidential reasoning and artificial potential field-based method. In: Multi-Criteria Decision Making in Maritime Studies and Logistics. Springer, pp. 123–149.
Minguez, J., Montano, L., 2003. Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. IEEE Transactions on Robotics & Automation 20 (1), 45–59. https://doi.org/10.1109/TRA.2003.820849.
Nomoto, K., Taguchi, T., Honda, K., Hirano, S., 1957. On the steering qualities of ships. Int. Shipbuild. Prog. 4, 354–370.
Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., Lane, D., 2007. Path planning for autonomous underwater vehicles. IEEE Trans. Robot. 23, 331–341.
Sato, M., Abe, K., Takeda, H., 1988. Learning control of finite Markov chains with an explicit trade-off between estimation and control. IEEE Trans. Syst. Man. Cybern. 18, 677–684.
Seuwou, P., Banissi, E., Ubakanma, G., Sharif, M.S., Healey, A., 2017. Actor-network theory as a framework to analyse technology acceptance model's external variables: the case of autonomous vehicles. In: International Conference on Global Security, Safety, and Sustainability. Springer, pp. 305–320.
Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: an Introduction. MIT press.
Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Mach. Learn. 8, 279–292.
Xue, Y., Clelland, D., Lee, B.S., Han, D., 2011. Automatic simulation of ship navigation. Ocean Eng 38, 2290–2305.
Yoo, B., Kim, J., 2016. Path optimization for marine vehicles in ocean currents using reinforcement learning. J. Mar. Sci. Technol. 21, 334–343.
Zhang, J., Yan, X., Chen, X., Sang, L., Zhang, D., 2012. A novel approach for assistance with anti-collision decision making based on the International Regulations for

Preventing Collisions at Sea. Proc. Inst. Mech. Eng. M J. Eng. Marit. Environ. 226, 250–259.

Zhang, P., Xu, X., Liu, C., Yuan, Q., 2009. Reinforcement learning control of a real mobile robot using approximate policy iteration. In: International Symposium on Neural Networks. Springer, pp. 278–288.

Zolfpour-Arokhlo, M., Selamat, A., Hashim, S.Z.M., Afkhami, H., 2014. Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms. Eng. Appl. Artif. Intell. 29, 163–177.