Proceedings of the ASME 2023 43rd International Conference on Ocean, Offshore and Arctic Engineering OMAE2024
June 9-14, 2024, Singapore EXPO, Singapore

OMAE2024-126768

DEEP REINFORCEMENT LEARNING FOR SHIP COLLISION AVOIDANCE AND PATH TRACKING

Amar Nath Singh[1,*], Akash Vijayakumar[1], Shankruth Balasubramaniyam[1], Abhilash Somayajula[1],

[1]Marine Autonomous Vessels Lab
Department of Ocean Engineering
Indian Institute of Technology Madras, Chennai, India

ABSTRACT

This research focuses on collision avoidance between ships in the maritime industry, addressing human decision errors as a significant contributor to ship-to-ship collision. Emphasizing the crucial role of automation in maritime operations, the study employs advanced algorithms, specifically Deep Q Network, to enhance dynamic collision avoidance capabilities. By automating decision-making processes, the research aims to significantly reduce collision occurrences between ships, highlighting these algorithms as promising within the realm of deep reinforcement learning. Utilizing a three-degree-of-freedom dynamic model and Krisco field ship as a benchmark hull, rigorous numerical simulations validate the proposed model's accuracy. The reinforcement learning agent, trained on this dynamic model, strives to optimize collision avoidance and waypoint tracking, demonstrated through numerical simulations and model experiments with a scaled version for a comprehensive evaluation of its maritime safety efficacy.

Keywords: Deep Reinforcement Learning, Collision Avoidance, Path Tracking, Deep Q learning, KCS, Path Following, DQN, CRI

1. INTRODUCTION

The maritime industry plays a pivotal role in global trade and transportation, facilitating the movement of goods and people across oceans. As the volume of maritime traffic escalates, the risk of ship collisions becomes an increasingly critical concern. Human errors contribute significantly to maritime accidents, posing dangers to lives, the environment, and the price range of delivery proprietors.

The traditional methods of collision avoidance, relying heavily on rule-based systems and manual navigation, have limitations in adapting to the dynamic and complex nature of maritime environments. Recent advancements in reinforcement learning (RL), a subfield of artificial intelligence (AI) have empowered autonomous systems to make intelligent decisions through learning from their complex environment.

This paper focuses on using Deep Q-Network (DQN) to help ships keep away from collisions and live on course more precisely.

While preceding research has explored AI strategies for avoiding limitations at sea, this study takes a step in addition. It seeks to develop a DQN agent-based controller that allows ships to comply with their meant paths, considering the complicated dynamics of vessels and the effect of environmental forces. They have looked at plans to test and validate the overall performance of this technique through simulations and actual international experiments.

As the maritime industry enters a new era, incorporating advanced technologies like reinforcement learning-based navigation can greatly improve ship safety and navigation accuracy. The objective of this research is to provide valuable insights for making maritime navigation more secure and efficient, thereby ensuring smooth global trade.

1.1 REINFORCEMENT LEARNING

In the Artificial Intelligence domain, reinforcement learning (RL) emerges as a powerful paradigm that mimics the way humans learn through trial and error. At its core, RL provides a framework for training intelligent agents to make sequential decisions in an environment, learning optimal strategies by receiving feedback in the form of rewards or penalties. Unlike supervised learning, where models are trained on labelled datasets with explicit input-output pairs, reinforcement learning operates in dynamic, uncertain environments. RL agents navigate these environments by taking actions to maximize cumulative rewards over time, adapting their behaviour based on the consequences of their actions. This intrinsic ability to learn from interactions makes RL particularly well-suited for addressing complex problems where explicit guidance may be limited or unavailable.

*Corresponding author

2. SHIP DYNAMICS MODEL

The major components of reinforcement learning implementation are the agent, environment and the interaction between them. The agent explores the environment and learns from it. The experience of the agent after learning, is utilized to choose the optimized behaviour. This paper deals with a maritime environment where the dynamic model of the ship acts as an agent. In this paper, the dynamic model of Krisco container ship (KCS) is chosen as an agent for running reinforcement learning simulation [1].

The parameters of the KCS vessel are given in Table 1. The ship dynamics are mathematically modeled with the help of the MMG (Maneuvering Modeling Group) model (Yasukawa and Yoshimura, 2015)[2]. The 3-DOF non-linear equations of motion are used to solve for ship manoeuvring motions, including surge, sway and yaw motions. The equations of motion are solved progressively at each time step as an initial value problem using a Runge–Kutta implicit solver. The commanded rudder angle δ is provided as an input at each time step. The MMG model used for simulating ship maneuvering (Yoshimura and Masumoto, 2012)[3] is as shown .

$$(m + m_x)\dot{u} - mvr - m_x G r^2 = X$$

$$(m + m_y)\dot{v} + mxG\dot{r} + mur = Y$$

$$(I_{zz} + J_{zz})\dot{r} + mxG\dot{v} + mxGu\dot{r} = \tau$$

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

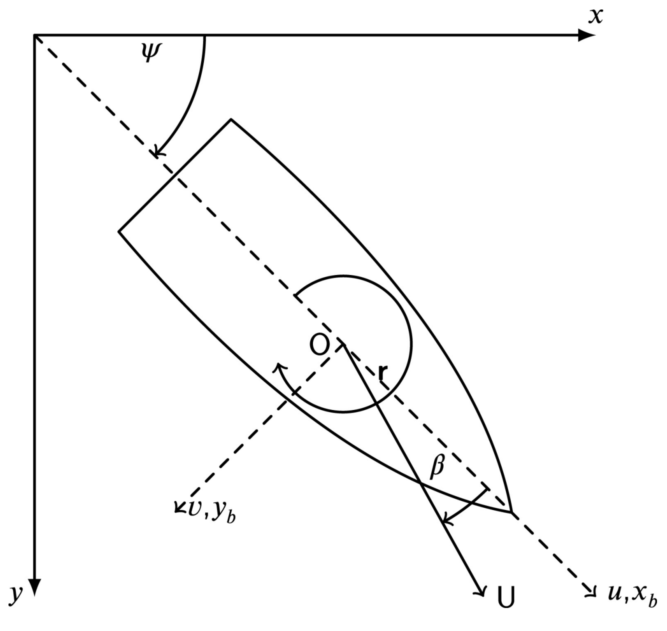

FIGURE 1: Dynamic Obstacle Case 3.

TABLE 1: KCS Ship Parameters

Ship Parameter	Value
Length between perpendiculars (L)	230 m
Length overall (LOA)	232.5 m
Depth moulded (d)	19 m
Beam (B)	32.2 m
Draft (d_{em})	10.8 m
Displacement	53,330.75 tons
LCG (x_G)	-3.408 m
Radius of gyration	57.5 m
Design speed (U)	12.347 m/s

3. DEEP Q-LEARNING ARCHITECTURE

Deep Q-Learning (DQN) is a reinforcement learning algorithm that combines Q-learning with deep neural networks to handle complex environments [4]. The vanilla DQN architecture involves the use of a deep neural network to approximate the Q-function, denoted as $Q(s, a; \theta)$, where s is the state, a is the action, and θ represents the parameters of the neural network.

3.1 Q-Value Update

The Q-value update in DQN is based on the temporal difference (TD) error. The TD error is defined as the difference between the current Q-value ($Q(s, a; \theta)$) and the target Q-value ($r + \gamma \max_{a'} Q(s', a'; \theta^-)$), where r is the reward, γ is the discount factor, s' is the next state, and θ^- represents the parameters of a target network.

The Q-value update equation is given by:

$$\delta = r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \tag{1}$$

The loss function for training the neural network is defined as the mean squared TD error:

$$\mathcal{L}(\theta) = \mathbb{E}\left[(\delta - Q(s, a; \theta))^2\right] \tag{2}$$

3.2 Neural Network Architecture

The neural network architecture typically consists of layers that process the state input and output the Q-values for each action. Let s be the state input, and $Q(s, a; \theta)$ be the output for each action a. The neural network is trained to minimize the loss function $\mathcal{L}(\theta)$.

3.3 Experience Replay

To improve stability and sample efficiency, DQN uses experience replay. Experience replay involves storing past experiences (state, action, reward, next state) in a replay buffer and randomly sampling batches during training. This helps break the temporal correlations in the data and improves the efficiency of the learning process.

3.4 Target Network

To stabilize training, DQN uses a target network with parameters θ^-. The target network is a delayed copy of the Q-network, and its parameters are updated periodically to the current Q-network parameters.

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{3}$$

where τ is a small update rate.

3.5 HYPER PARAMETERS

The hyperparameters used for running all the scenarios mentioned in this paper is given in the Table 2.

Parameter	Value
Num Episodes	7000
Model	Multi-Layer Perceptron [128, 128]
Activation Function	Tanh
Optimizer	Adam
Max Episode Steps	240
Learning Starts	100000
Initial Learning Rate	0.001
Final Learning Rate	5e-7
Learning Rate Scheduler	Exponential
Train Frequency	10
Gradient Steps	-1
Batch Size	128
Tau	0.02
Gamma	0.98
Buffer Size	100000
Target Update Interval	1
Exploration Fraction	0.8
Exploration Initial Eps	1
Exploration Final Eps	0.1

TABLE 2: DQN Hyperparameters

4. COLLISION RISK

A Collision Risk Index (CRI) is needed to define a reward function for the RL environment so that the ship agent can be penalized for increasing the risk of collision and rewarded for reducing it. The collision risk index(CRI) represents the risk of collision of the vessel with the target vessel. There are many collision risk indices available for identifying the risk of collision between two vessels. In this paper a collision risk index based on DCPA and TCPA values has been utilized.

4.1 COLLISION RISK INDEX

This method considers "distance of the closest point of approach (DCPA)" and "time to the closest point of approach (TCPA)" as the primary factors that affect the risk of ship collision [5] [6]. Additionally, this approach can accurately calculate the CRI for both static and dynamic obstacles.

$$\phi = \arctan\left(\frac{Y}{X}\right)$$
$$\lambda = \arctan\left(\frac{V_y}{V_x}\right)$$
$$\gamma = \lambda - \phi - \pi$$

where:

$\mathbf{X}$ = Relative position of obstacle to vessel in X-axis
$\mathbf{Y}$ = Relative position of obstacle to vessel in Y-axis
$\mathbf{V_x}$ = Relative Velocity of obstacle to vessel in X-axis
$\mathbf{V_y}$ = Relative Velocity of obstacle to vessel in Y-axis

The Distance of Closest Point of Approach (4) is the minimum separation distance between two moving objects. It is the closest distance they come to each other during their respective trajectories.

$$DCPA = |\sin(\gamma)| \cdot D \tag{4}$$

The Time to Closest Point of Approach (5) is the time it takes for two moving objects to reach their closest point. It is often expressed as a relative time, indicating how long it will take for the objects to reach their DCA. The TCPA can be calculated using the relative velocity of the two objects and the DCA.

$$TCPA = \frac{|\cos(\gamma)| \cdot D}{V} \tag{5}$$

where:

$\mathbf{D}$ = Distance to the obstacle
$\mathbf{V}$ = Magnitude to Relative Velocity of obstacle to vessel

CRI value can be calculated using Equation (6)

$$\text{CRI} = \begin{cases} 0 & \text{if TCPA} < 0 \\ 1 & \text{if } D < R\text{s} \\ e^{-|a \cdot DCPA + b \cdot TCPA|} & \text{otherwise} \end{cases} \tag{6}$$

where:

$\mathbf{a}$ = 0.2
$\mathbf{b}$ = 0.1
$\mathbf{R}$s = Safety distance; Twice of ship length

5. PATH FOLLOWING

During the course of the simulation, the vessel is in calm waters. It is provided with a set of waypoints to navigate from the start to the goal state, represented as a Reinforcement Learning (RL) environment. The parameters employed to guide the vessel towards its destination while staying close to the intended path include the distance to the goal, vessel heading, cross-track error, and angular velocity. These factors collectively form the observation state used for model training. The distance to the goal ensures steady progress towards the destination, while the heading angle ensures alignment with the goal direction. The cross-track error indicates the perpendicular distance from the vessel to the intended path, helping to stay on course. Additionally, managing angular velocity helps maintain stability by minimizing yaw-rate deviation.

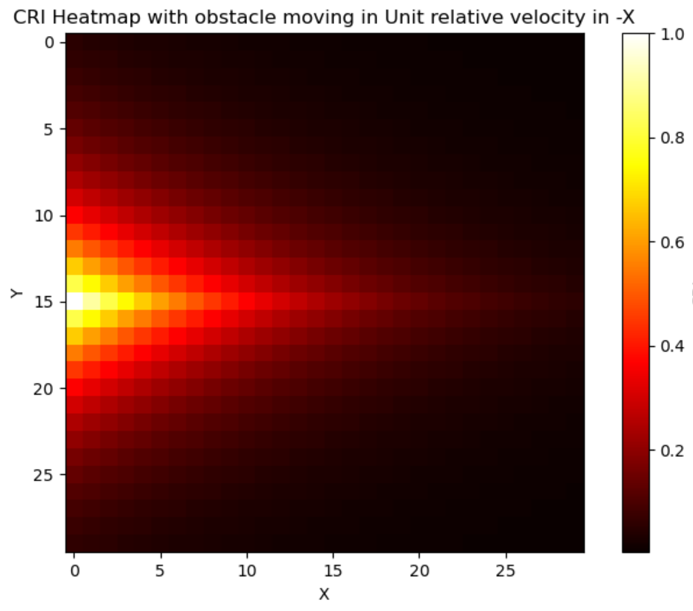

FIGURE 2: CRI Heatmap where all obstacles have unit relative velocity in negative X-axis and vessel is at position (0,15)

5.1 ENVIRONMENT

The Path Following environment used in this paper is an RL simulation environment that is designed to test the ability of autonomous vessels to navigate towards a goal point from a starting point. The vessel is spawned at Origin with a heading on the X-axis. The goal point is also located at a specified point in the environment, and the vessel's objective is to navigate to the goal point.

The goal point is calculated using the distance from the origin to the goal point D_g and the angle from the X-axis θ_g. The coordinates of the goal point (x_g, y_g) can be obtained as follows:

$$x_{\text{goal}} = D_g \cos(\theta_g)$$
$$y_{\text{goal}} = D_g \sin(\theta_g)$$

The goal point given to the vessel is the same throughout an episode of simulation, and it changes as the episode changes. The change in goal point with episodes is according to D_g and θ_g sampling using the probability density function of the uniform distribution.

The probability density function (pdf) of the uniform distribution over the interval $[a, b]$ used in this paper is given by:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \le x \le b \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Where:

$\mathbf{a}$ = 8.0 times ship length

$\mathbf{b}$ = 50.0 times ship length

5.2 CROSS-TRACK ERROR (CTE)

The vessel is allowed to follow the track the path given by the waypoints in each episode of the RL simulation, which essentially means the vessel has to reduce the deviation from the track, the vessel has to follow. The parameter which defines the deviation from the track is known as the cross-track error(CTE). The cross-track error (CTE), denoted by e_{CT}, is the minimum distance between the vessel's current position and the line joining the waypoints. The cross-track error (e_{CT}) can be expressed using the cross-product of vectors. Let $\mathbf{p}_{\text{initial}} = \begin{bmatrix} x_{\text{initial}} \\ y_{\text{initial}} \end{bmatrix}$, $\mathbf{p}_{\text{goal}} = \begin{bmatrix} x_{\text{goal}} \\ y_{\text{goal}} \end{bmatrix}$, and $\mathbf{p}_{\text{current}} = \begin{bmatrix} x_{\text{current}} \\ y_{\text{current}} \end{bmatrix}$ represent the initial, final, and current positions, respectively.

The cross-track error can be calculated using the cross product of vectors $\mathbf{v}_{\text{to\_goal}}$ and $\mathbf{v}_{\text{current}}$, where:

$$\mathbf{v}_{\text{to\_goal}} = \mathbf{p}_{\text{goal}} - \mathbf{p}_{\text{initial}}$$

$$\mathbf{v}_{\text{current}} = \mathbf{p}_{\text{current}} - \mathbf{p}_{\text{initial}}$$

For our case: $\mathbf{p}_{\text{initial}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
The cross-track error is then given by:

$$e_{\text{CT}} = \frac{||\mathbf{v}_{\text{to\_goal}} \times \mathbf{v}_{\text{current}}||}{||\mathbf{v}_{\text{to\_goal}}||}$$

5.3 DISTANCE TO GOAL

The distance (D_{g}) from the vessel's current position to the final point can be calculated using the Euclidean distance formula:

$$D_{\text{g}} = \sqrt{(x_{\text{goal}} - x_{\text{current}})^2 + (y_{\text{goal}} - y_{\text{current}})^2}$$

5.4 COURSE ANGLE ERROR

The course angle error defines the deviation of the velocity of the ship from a line joining the initial and goal point, denoted by θ_{error}. It is the angle between the vessel's global velocity vector and the line joining the vessel's position to the final point. It can be calculated using the dot product:

$$\theta_{\text{error}} = \arccos\left(\frac{\mathbf{v} \cdot \mathbf{d}}{||\mathbf{v}|| \, ||\mathbf{d}||}\right)$$

where:

$\mathbf{v}$ = Velocity vector of the vessel

$\mathbf{d}$ = Vector along the line joining the vessel's position to the goal

5.5 OBSERVATION SPACE

In Deep Q-Networks (DQN) reinforcement learning, the observation space represents the information the agent receives from the environment at each time step when the agent interacts with the environment. The observation space is crucial for the agent to make decisions and learn optimal strategies.

Components of Observation Space

The observation space for a path-following scenario can include the following components:

- **Cross-Track Error (e_{CT}):** The minimum distance between the vessel's current position and the path connecting the initial and final waypoints.
- **Course Angle Error (θ_{error}):** The angle between the vessel's global velocity vector and the line connecting the vessel's position to the final point.
- **Goal Distance (D_{g}):** The distance from the vessel's current position to the final point.
- **Angular Velocity (r):** The angular velocity of the ship.

In the context of path following, the observation space can be defined as:

$$\text{Observation Space} = [e_{\mathrm{CT}}, \theta_{\mathrm{error}}, D_{\mathrm{i}}, r]$$

Where:

$$\mathbf{D}_{\mathrm{i}} = 8 + 10 \cdot \frac{D_{\mathrm{g}} - a}{b - a}$$

$$\mathbf{a} = \text{Minimum goal distance used for training}$$

$$\mathbf{b} = \text{Maximum goal distance used for training}$$

Given that D_{g} can take on large values, to stabilize training, we use D_{i}, a scaled value of D_{g} within the range of 8 to 18.

5.6 ACTION SPACE

In DQN, the action space is often referred to as the "action state." The Q-function, which the DQN is learning to approximate, takes the current state of the environment and an action as input and outputs a Q-value. The agent selects actions based on these Q-values. The action space used in this paper contains the rudder angle of the vessel.

The rudder angle of the ship is represented as a discrete set of actions. The list gives the possible actions:

$$\text{actions} = [-35^\circ, -20^\circ, 0^\circ, 20^\circ, 35^\circ]$$

Here, each element in actions represents a discrete rudder angle that the agent can choose. The angles are in degrees.

5.7 REWARD FUNCTION

In reinforcement learning, the reward is a crucial signal guiding the agent's learning process. For the given information, four reward functions are defined:

$$R_1 = 1.3 \cdot \exp(-10 \cdot |\theta_{\mathrm{error}}|) - 0.3$$
$$R_2 = 2.0 \cdot \exp(-0.08 \cdot e_{\mathrm{CT}}^2) - 1.0$$
$$R_3 = -0.25 \cdot D_{\mathrm{i}}$$
$$R_4 = \begin{cases} 0.0 & \text{if } \cdot D_{\mathrm{g}} > 0.5, \\ 20.0 & \text{otherwise.} \end{cases}$$

5.8 DISCUSSION AND RESULTS

This paper presents an implementation of path following in an RL environment using the DQN algorithm. Instead of employing a proportional derivative integral (PID) controller or any other traditional method, the DQN algorithm is utilized to track waypoints from the initial to the target position. The results achieved by training the RL simulation through 7000 runs are depicted in the figures below.

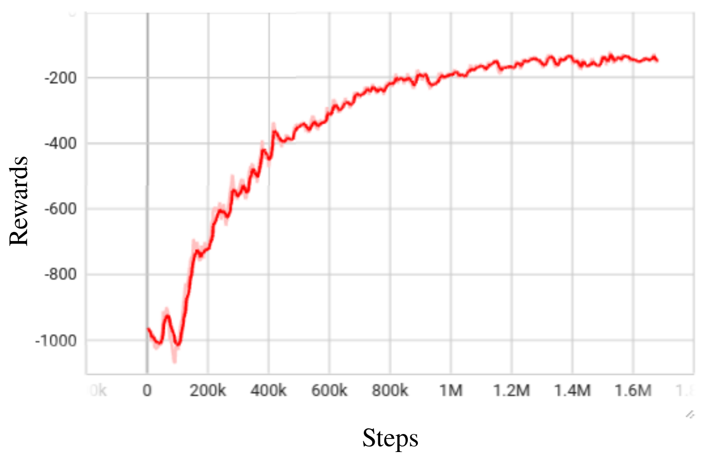

FIGURE 3: Steps vs Rewards Plot.

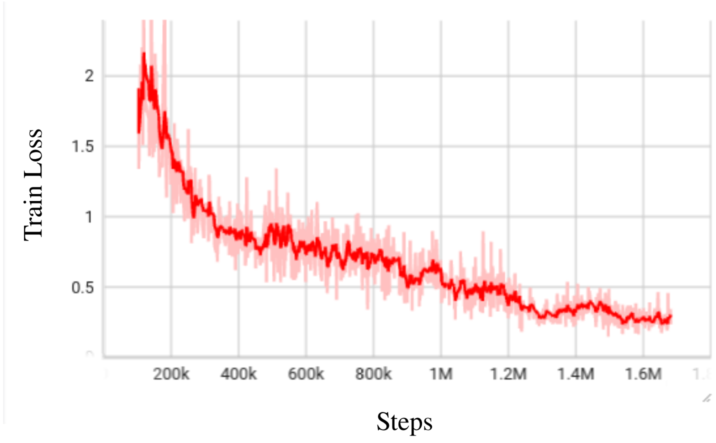

FIGURE 4: Steps vs Train Loss Plot.

6. COLLISION AVOIDANCE FOR STATIC OBSTACLES

In this particular setting, obstacle vessels are randomly generated along the path of vessels. These obstacles remain stationary, indicating that they do not exhibit any movement. This segment addresses strategies for addressing static obstacle avoidance by implementing Deep Q Network (DQN) [7].

6.1 ENVIRONMENT

The vessel is spawned at the origin at the origin with a random heading angle and random goal. Given the number of obstacles, every obstacle has a 90 percent chance of spawning in this environment, ensuring that agents learn to tackle scenarios without obstacles. The coordinate of the obstacle can be calculated using

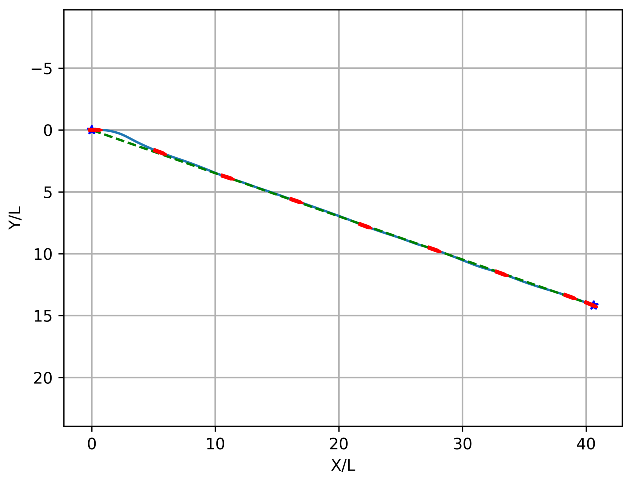

FIGURE 5: Path Following Case 1.

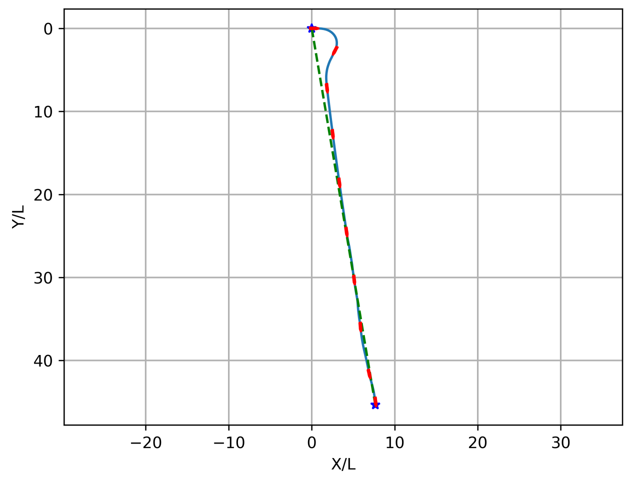

FIGURE 6: Path Following Case 2.

D_o and θ_l.

$$X_o = D_o \cos(\theta_l)$$
$$Y_o = D_o \sin(\theta_l)$$

D_o is sampled using the probability density function of the uniform distribution with $a = 0$, b = Distance between goal and initial point.
θ_l is the angle from the X-axis to the line joining the goal and initial point.

The generalized policy of the agent is ensured by adding random noise to the position of obstacles.

6.2 OBSERVATION SPACE

The observation states are determined by identifying obstacles with the maximum Criticality Index (CRI) value, designating them as the most crucial obstacles. The Observation States derived from the path following the process will remain unchanged, with the addition of two new values: η and θ. Therefore, the updated observation space is represented as follows:

$$\text{Observation Space} = [e_{\text{CT}}, \theta_{\text{error}}, d_{\text{goal}}, r, \eta, \theta]$$

$$\eta = \frac{R - D}{R}$$

Here, e_{CT} represents the cross-track error, θ_{error} denotes the heading error, d_{goal} signifies the distance to the goal, r represents the angular velocity of the ship, D is the distance between the vessel and obstacle, R is maximum range to which vessel can see obstacles, and θ angle between velocity vector of vessel. The variable η denotes the normalized distance between the vessel and an obstacle within its detection range. It aims to scale distance values during training to prevent exceedingly high magnitudes. As the obstacle recedes far beyond the vessel's observable range, η diminishes significantly, approaching zero. Conversely, the numerator and denominator nearly equalize as the obstacle draws closer, leading η towards unity.

If there are no Obstacles in the vessel's field of view, We use a dummy obstacle with $D = R$ and $\theta = 0.0$.

6.3 REWARD FUNCTION

Using the same reward functions from the path following in addition to a collision penalty, if the agent gets collided, the episode is truncated with a collision penalty of -200.0 Reward values can calculated as follows:

$$R_1 = 1.3 \cdot \exp(-10 \cdot |\theta_{\text{error}}|) - 0.3$$
$$R_2 = 2.0 \cdot \exp(-0.08 \cdot e_{\text{CT}}^2) - 1.0$$
$$R_3 = -0.25 \cdot D_{\text{i}}$$
$$R_4 = \begin{cases} 0.0 & \text{if } \cdot D_{\text{g}} > 0.5, \\ 20.0 & \text{otherwise.} \end{cases}$$
$$R_5 = \begin{cases} 0.0 & \text{if } \cdot D > 2.0, \\ -200.0 & \text{otherwise.} \end{cases}$$

6.4 DISCUSSION AND RESULTS

Instead of using rule-based collision avoidance, where we have to manually define the dynamics of interaction, the DQN algorithm will learn the dynamics on its own and find a safe path around the obstacle. The results obtained through running 7000 episodes of RL simulation are shown in these figures.

Observing Figure 10, it becomes evident that when the obstacle is distant, the vessel maintains its intended path, given the negligible value of η. However, as the obstacle nears, the vessel initiates obstacle avoidance maneuvers, indicated by η approaching unity.

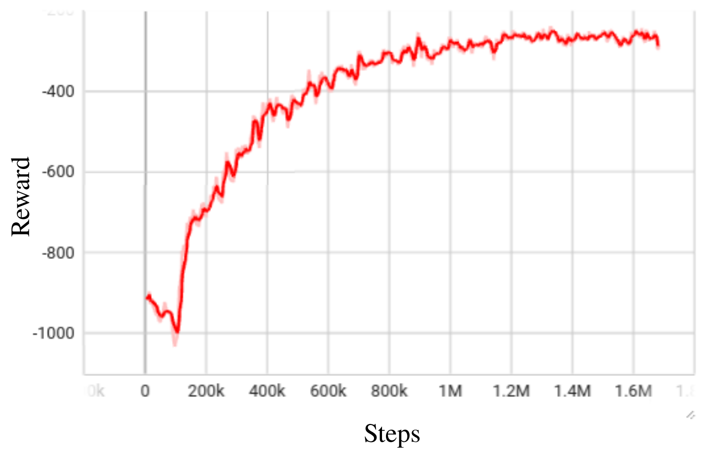

FIGURE 7: Steps vs Rewards Plot.

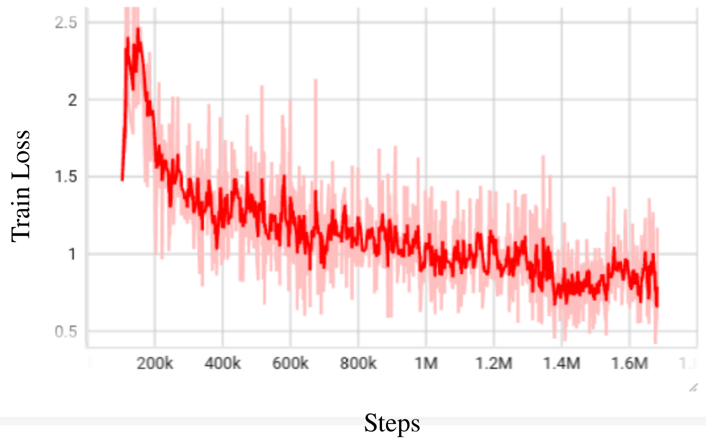

FIGURE 8: Steps vs Train Loss Plot.

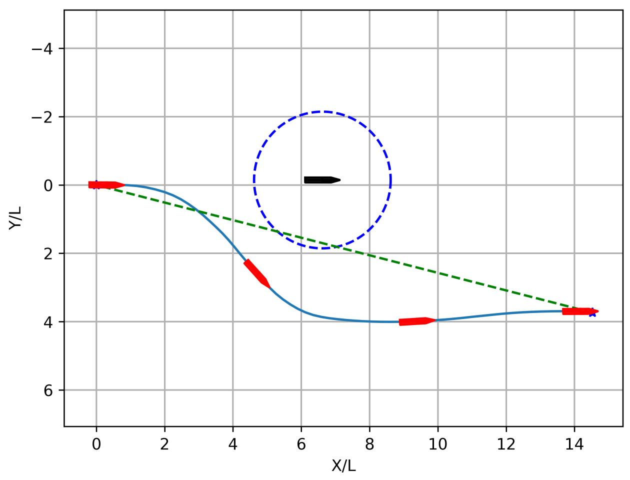

FIGURE 9: Static Obstacle Case 1.

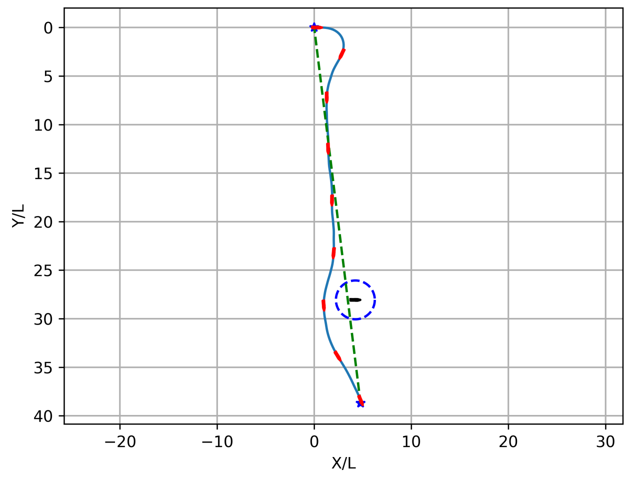

FIGURE 10: Static Obstacle Case 2.

7. COLLISION AVOIDANCE FOR DYNAMIC OBSTACLES

This section presents a DQN approach to handling Collision Avoidance and Path Following in an environment with moving Obstacles [8].

7.1 ENVIRONMENT

In this scenario, the vessel is generated at a starting point with a random direction and a random destination, similar to the process of avoiding stationary obstacles. There is a 90 percent chance of spawning each obstacle, if there are any, in this environment. This ensures that the agents learn how to navigate without obstacles and are well-equipped to handle situations with obstacles.

If an obstacle is spawned, it will have a 10 percent chance of having zero velocity, making it a static obstacle; this is to ensure training over static obstacles as well. The velocity of obstacles is randomly sampled using probability density function (7) with a = -2.5 and b = 2.5.

The quantities that define an obstacle state are $[X_o, Y_o, V_{ox}, V_{oy}]$.

Where:

$$\mathbf{V_{ox}} = \text{Velocity of obstacle in X-axis}$$
$$\mathbf{V_{oy}} = \text{Velocity of obstacle in Y-axis}$$

7.2 OBSERVATION SPACE

The observation states derived from the path following will remain unchanged. New observation states representing obstacles in the environment will be generated from a potential field based on the positions and velocities of all obstacles in the arena. The relative position of the obstacles in the vessel's body frame is calculated as follows:

$$\begin{bmatrix} X_r \\ Y_r \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} X_o - X_a \\ Y_o - Y_a \end{bmatrix}$$

$$D = \sqrt{X_r^2 + Y_r^2}$$

$$\lambda = \arctan\left(\frac{Y_r}{X_r}\right) - \pi$$

The relative velocity of the obstacles in the vessel's body frame is calculated as follows:

$$\begin{bmatrix} V_{rx} \\ V_{ry} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} V_{ox} - V_{ax} \\ V_{oy} - V_{ay} \end{bmatrix}$$

The Potential due to moving obstacles can be calculated as follows [9]:

$$F_{px} = K \cdot \sum\left(\frac{1}{D} - \frac{1}{R}\right) \cdot \frac{1}{D^2} \cdot \cos\lambda$$

$$F_{py} = K \cdot \sum\left(\frac{1}{D} - \frac{1}{R}\right) \cdot \frac{1}{D^2} \cdot \sin\lambda$$

$$F_{vx} = R \cdot \sum \frac{V_{rx}}{D}$$

$$F_{vy} = R \cdot \sum \frac{V_{ry}}{D}$$

Where:

R = Maximum distance to which vessel can see obstacles

ψ = Heading angle of the vessel

$$\text{Observation Space} = [e_{\text{CT}}, \theta_{\text{error}}, D_{\text{g}}, r, F_{px}, F_{py}, F_{vx}, F_{vy}]$$

If no obstacles are nearby, $F_{px} = 0$, $F_{py} = 0$, $F_{vx} = 0$, and $F_{vy} = 0$ are padding values.

7.3 REWARD FUNCTION

Introducing the CRI value of the most critical obstacles as a reward for learning a policy that aims to minimize the CRI value along the path. In this case, when the vessel detects a collision with an obstacle, the episode is not truncated, but a continuous penalty is given. Reward values can calculated as follows:

$$R_1 = 1.3 \cdot \exp(-10 \cdot |\theta_{\text{error}}|) - 0.3$$

$$R_2 = 2.0 \cdot \exp(-0.08 \cdot e_{\text{CT}}^2) - 1.0$$

$$R_3 = -0.25 \cdot D_{\text{i}}$$

$$R_4 = \begin{cases} 0.0 & \text{if } \cdot D_{\text{g}} > 0.5, \\ 20.0 & \text{otherwise.} \end{cases}$$

$$R_5 = -5.0 \cdot CRI$$

7.4 DISCUSSION AND RESULTS

In this paper, a potential field method is incorporated with the Deep Q Network (DQN) algorithm to run the simulations in the dynamic collision avoidance RL environment so that the agent executes a safe maneuver on encountering a dynamic obstacle ahead. This method can capture the velocity interaction of multiple moving obstacle information and take appropriate maneuvers. The vessel selects the shortest path while evading obstacles, as depicted in Figure 13. When encountering a velocity obstacle, it maneuvers behind the obstructing vessel to avoid a collision, as shown in Figure 14. As the vessel moves away from the obstacle, its influence diminishes due to the D^2 term in the potential equation. The results of training for 7000 simulations in the environment are illustrated in the figure.

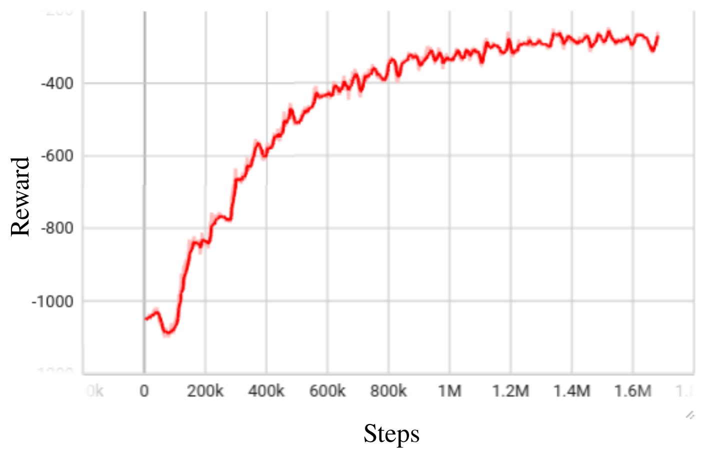

FIGURE 11: Steps vs Rewards Plot.

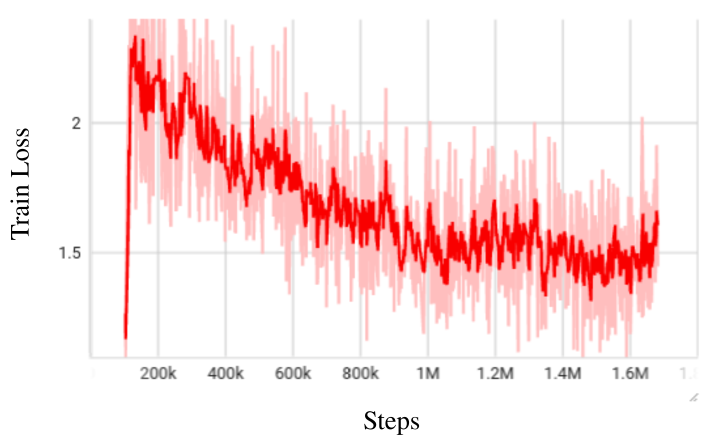

FIGURE 12: Steps vs Train Loss Plot.

8. CONCLUSION

The DQN policy is used in this paper for collision avoidance, where satisfactory results are obtained. Here, the DQN algorithm has been implemented for all major scenarios in maritime autonomous navigation. In all scenarios, satisfactory results are obtained by running the RL simulations. A comparative study needs to be done with the other policies, such as PPO and DDPG. Since there is not much information about the individual moving obstacle as the cumulative reward is considered, the trained RL model may not prefer the optimal path to dynamically avoid the obstacles. As a future scope of this paper, the discretization part of the environment can be worked upon, by introducing cost map architecture also so that whole information about the simulation environment can be captured. The dynamic obstacle avoidance scenario can be enhanced by inculcating the Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs) into the observation space so that the algorithm will be well equipped for real-world deployment.

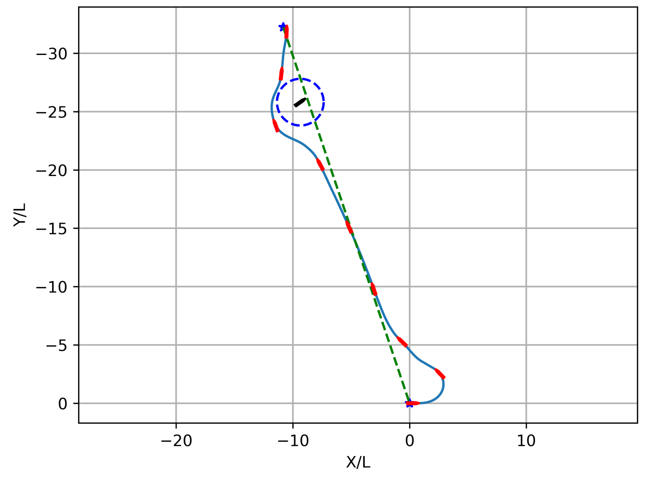

FIGURE 13: Dynamic Obstacle Case 1.

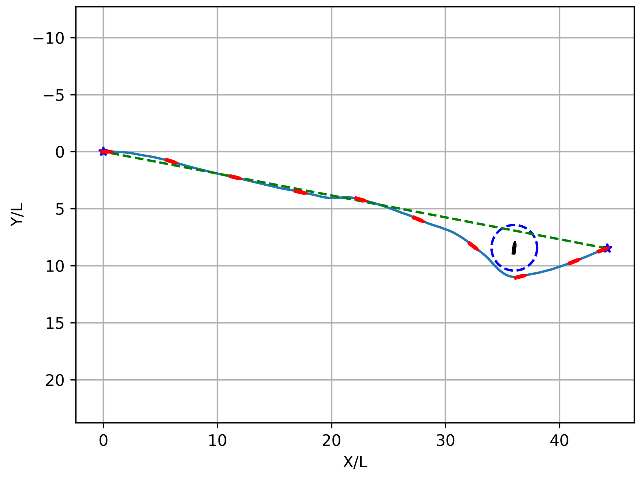

FIGURE 14: Dynamic Obstacle Case 2.

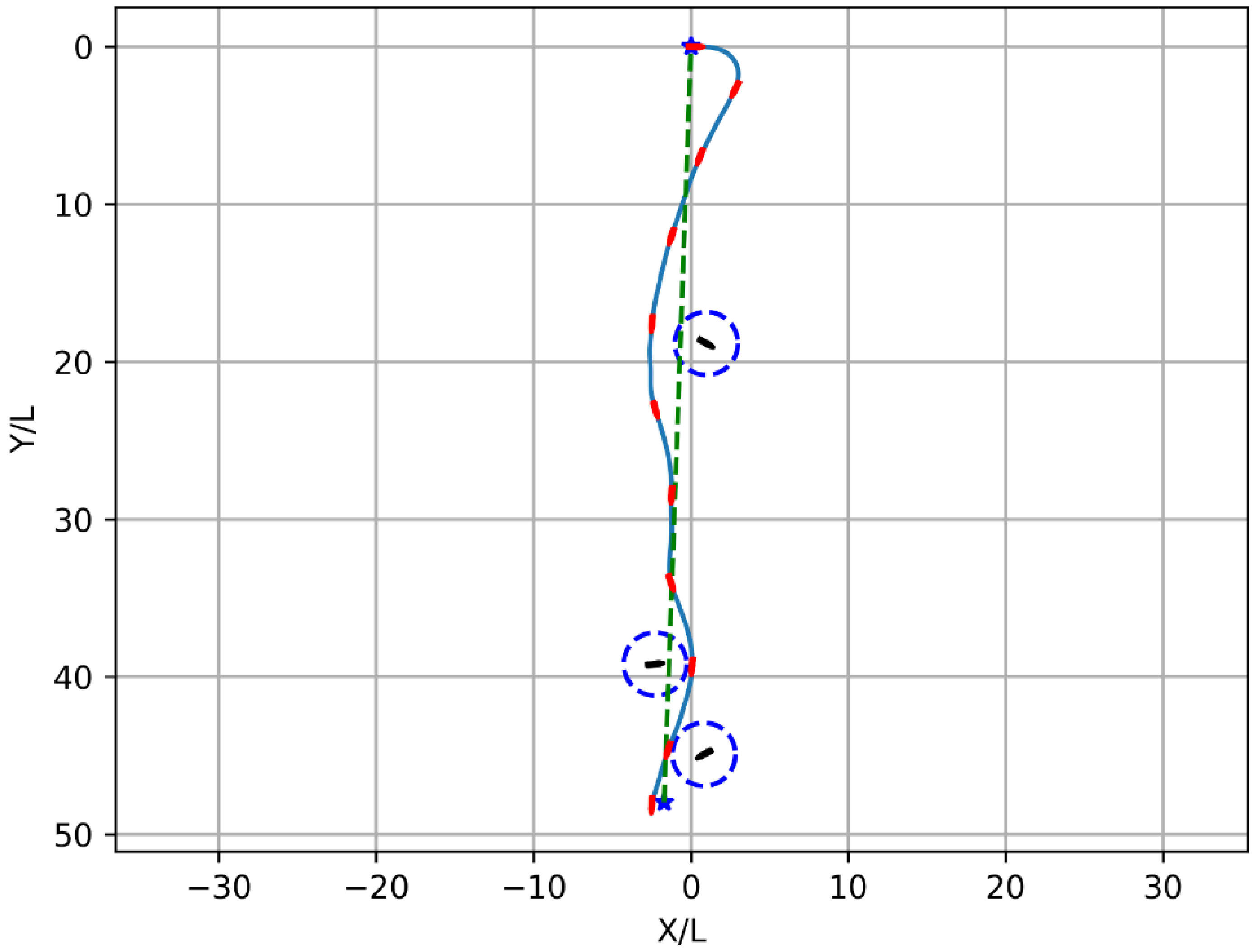

FIGURE 15: Dynamic Obstacle Case 3.

REFERENCES

[1] Deraj, Rohit, Kumar, R.S. Sanjeev, Alam, Md Shadab and Somayajula, Abhilash. "Deep reinforcement learning based controller for ship navigation." *Ocean Engineering* Vol. 273 (2023): p. 113937. DOI https://doi.org/10.1016/j.oceaneng.2023.113937. URL https://www.sciencedirect.com/science/article/pii/S0029801823003219.

[2] Yoshimura, H. Yasukawa • Y. "Introduction of MMG standard method for ship maneuvering predictions." *Springer Link* (2014)DOI https://doi.org/10.1007/s00773-014-0293-y. URL https://link.springer.com/article/10.1007/s00773-014-0293-y.

[3] Masayoshi Yoshimura, Yusuke Matsunaga, Yusuke Akamine. "An Exact Estimation Algorithm of Error Propagation Probability for Sequential Circuits." *J Stage* (2012)DOI https://doi.org/10.2197/ipsjtsldm.5.63. URL https://www.jstage.jst.go.jp/article/ipsjtsldm/5/0/5_0_63/_article.

[4] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan and Riedmiller, Martin. "Playing Atari with Deep Reinforcement Learning." (2013). URL 1312.5602.

[5] Li, Weifeng, Zhong, Lufeng, Xu, Yang and Shi, Guoyou. "Collision Risk Index Calculation Based on an Improved Ship Domain Model." *Journal of Marine Science and Engineering* Vol. 10 No. 12 (2022). DOI 10.3390/jmse10122016. URL https://www.mdpi.com/2077-1312/10/12/2016.

[6] Ha, Jisang, Roh, Myung-Il and Lee, Hye-Won. "Quantitative calculation method of the collision risk for collision avoidance in ship navigation using the CPA and ship domain." *Journal of Computational Design and Engineering* Vol. 8 No. 3 (2021): pp. 894–909. DOI 10.1093/jcde/qwab021. URL https://academic.oup.com/jcde/article-pdf/8/3/894/37931378/qwab021.pdf, URL https://doi.org/10.1093/jcde/qwab021.

[7] Qian, Yubin, Feng, Song, Hu, Wenhao and Wang, Wanqiu. "Obstacle avoidance planning of autonomous vehicles using deep reinforcement learning." *Advances in Mechanical Engineering* Vol. 14 No. 12 (2022): p. 16878132221139661. DOI 10.1177/16878132221139661. URL https://doi.org/10.1177/16878132221139661, URL https://doi.org/10.1177/16878132221139661.

[8] Yang, Xiao and Han, Qilong. "Improved DQN for Dynamic Obstacle Avoidance and Ship Path Planning." *Algorithms* Vol. 16 No. 5 (2023). DOI 10.3390/a16050220. URL https://www.mdpi.com/1999-4893/16/5/220.

[9] Zhu, Qidan, Yan, Yongjie and Xing, Zhuoyi. "Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing." *Sixth International Conference on Intelligent Systems Design and Applications*, Vol. 2: pp. 622–627. 2006. DOI 10.1109/ISDA.2006.253908.