

Distributional Reinforcement Learning based Integrated Decision Making and Control for Autonomous Surface Vehicles

Xi Lin, Paul Szenher, Yewei Huang, and Brendan Englot

arXiv:2412.09466v1 [cs.RO] 12 Dec 2024

Abstract—With the growing demands for Autonomous Surface Vehicles (ASVs) in recent years, the number of ASVs being deployed for various maritime missions is expected to increase rapidly in the near future. However, it is still challenging for ASVs to perform sensor-based autonomous navigation in obstacle-filled and congested waterways, where perception errors, closely gathered vehicles and limited maneuvering space near buoys may cause difficulties in following the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs). To address these issues, we propose a novel Distributional Reinforcement Learning based navigation system that can work with onboard LiDAR and odometry sensors to generate arbitrary thrust commands in continuous action space. Comprehensive evaluations of the proposed system in high-fidelity Gazebo simulations show its ability to decide whether to follow COLREGs or take other beneficial actions based on the scenarios encountered, offering superior performance in navigation safety and efficiency compared to systems using state-of-the-art Distributional RL, non-Distributional RL and classical methods.

Index Terms—Marine Robotics, Autonomous Vehicle Navigation, Reinforcement Learning

I. INTRODUCTION

Recent years have witnessed increasing demands for and deployments of Autonomous Surface Vehicles (ASVs), which motivates the development of autonomous navigation systems [1]. This paper considers scenarios in which multiple ASVs operate in congested waterways where buoys may exist, and aims at providing a single robust onboard ASV navigation system which can be deployed on multiple ASVs. Some existing works have considered collision avoidance for multiple vessels with simplified vehicle and environment dynamics, and assumed very accurate perception in simulation [2], [3]. Other works have verified algorithm performance in field experiments, but with only one autonomous vehicle and other vessels driven by humans or following pre-defined trajectories [4], [5], [6], without demonstrating performance in multi-ASV navigation scenarios.

In this letter, we propose a novel and computationally efficient ASV navigation system that can work with onboard LiDAR and odometry sensors, and we demonstrate its performance in a Gazebo based simulator [7] that is capable of simulating complex multi-ASV navigation scenarios with re-

This work was supported by the Office of Naval Research, Grants N00014-20-1-2570 and N00014-24-1-2522.

The authors are with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA
 $\{xlin26, pszenher, yhuang85, benglot\}@stevens.edu$

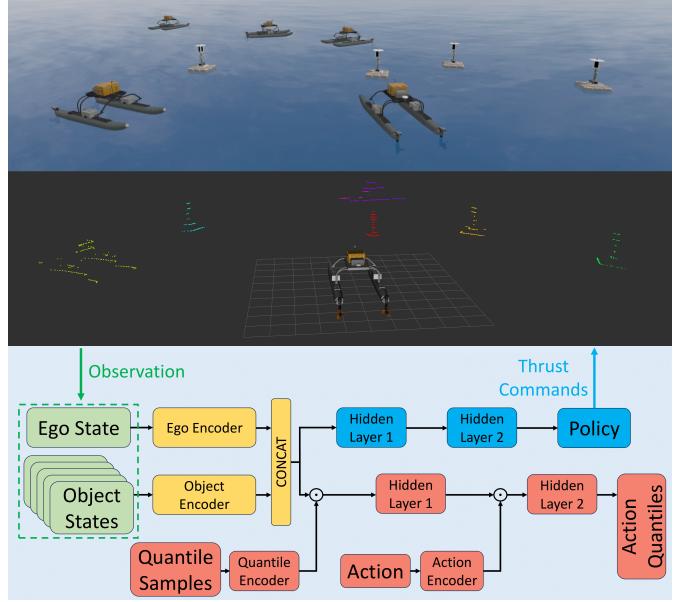


Fig. 1: **The proposed navigation system.** The top figure shows the perspective in the Gazebo simulation. The middle figure visualizes the segmentation result of LiDAR point clouds received by the lower right vehicle in the top figure. The bottom figure illustrates the decision making and control module of the proposed system.

alistic environment dynamics, vehicle models, and perceptual sensing.

The framework of our proposed system is shown in Figure 1, which uses a Distributional Reinforcement Learning (Distributional RL) agent for decision making and control of the vehicle given perceptual information. The Distributional RL agent can be represented as a deep neural network model, and learns the distributions of cumulative rewards related to different actions via trial-and-error learning [8]. The robustness of Distributional RL policies in the multi-ASV navigation problem has been demonstrated in our prior work [9], which uses Implicit Quantile Networks (IQN) [10].

However, the limitations of our prior work [9] include a highly simplified point mass vehicle model, an idealized vortical flow model for environmental disturbances, and a perfect perception assumption. Realistic wind and wave effects on motion and perception were not considered. In addition, IQN was implemented with a predefined discrete set of motion commands for vehicle control, and was not able to output arbitrary actions in a continuous action space. In recent years, Distributional RL algorithms with actor-critic structures [11], [12], [13] have been developed to work over continuous action

domains. Adopting this idea, we develop a novel ASV decision making and control policy based on IQN employed within an actor-critic framework, which we denote AC-IQN, for continuous control in scenarios involving congested multi-vehicle encounters.

The Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) [14] specify rules that apply to two-vessel encounter situations, but it faces challenges such as creating conflicting actions in multi-vessel cases [15] and limited maneuvering space near static obstacles like buoys [9]. In addition, the perceived states of nearby objects may not be precise, and can cause false judgements concerning COLREGs. To address these issues, we design a reward function that encourages COLREGs compliant behaviors, but also does not penalize other collision avoidance behaviors that are beneficial to navigation safety and efficiency. The software implementation of our approach is available at https://github.com/RobustFieldAutonomyLab/Distributional_RL_Decision_and_Control. The main contributions of this letter are summarized as follows:

- Proposal of an ASV decision making and control solution based on AC-IQN for navigating in congested multi-vehicle environments, under the influence of wind and wave disturbances on ASV motion and perception.
- Design of a novel reward function that trains policies capable of both following and deviating from COLREGs as needed for safe and efficient ASV behavior.
- Extensive evaluations in high-fidelity Gazebo simulations show superior navigation safety and efficiency performance over state-of-the-art Distributional RL, non-Distributional RL and classical baselines.

The rest of this letter is organized as follows. A review of relevant literature is given in Section II. The environment dynamic model of the Gazebo simulator and the vehicle control settings we use are introduced in Section III. Section IV introduces the perception module of the proposed system. Section V describes decision and control with AC-IQN and other RL baselines. Section VI shows experimental results. Section VII concludes the letter and discusses future work.

II. RELATED WORKS

Multi-vessel collision avoidance problems have been widely studied with the consideration of COLREGs. Naeem et al. [16] and Chiang et al. [17] developed COLREGs-compliant algorithms based on Artificial Potential Field (APF) and rapidly-exploring random tree (RRT) respectively by generating virtual obstacles around other vessels to prevent COLREGs-violating actions; Cho et al. [3] developed a rule-based system that specifies the roles of vessels in multi-vessel encounters and utilizes the probabilistic velocity obstacle method for collision avoidance. However, system performance of these works is only illustrated in highly simplified simulations without modeling of environmental disturbances and perceptual error. Kuwata et al. [4] proposed an approach based on Velocity Obstacles (VO) and demonstrated its performance in field tests involving an embodied ASV and manually-driven vessels in an environment with dimensions of about a thousand meters, but the performance in congested scenarios with multiple ASVs

is not studied. Hagen et al. [5] presented a Model Predictive Control (MPC) strategy that chooses actions which minimize the cost of COLREGs-compliant collision avoidance, showing field experiment results involving scenarios with two-vehicle encounters only.

Some works have targeted situations where it is difficult to guarantee navigation safety when following COLREGs, proposing alternative strategies. Cho et al. [18] and Jia et al. [19] proposed intent inference mechanisms on other encountered vehicles to decide if evasive actions are needed. Jeong et al. [15] focused on the multi-ASV collision avoidance problem in highly congested scenarios, and proposed to use multi-objective optimization for action selection based motion attributes of other ASVs. These works provide insight into the problem where the ego vehicle needs to exhibit different motion patterns based on the scenarios encountered, under the ideal assumption of no wind and wave disturbances and no perceptual error. Additionally, environments containing static obstacles are not considered in these works.

In recent years, Deep Reinforcement Learning (DRL), given its ability to adapt to complex dynamic scenarios, has been used to develop collision avoidance policies for ASVs assuming no external disturbances to vehicles. Zhao et al. [2] presented a DRL policy-gradient agent for ASV path following and COLREGs-compliant collision avoidance task. Meyer et al. [20] trained a proximal policy optimization (PPO) based controller that can navigate among other path-following vessels and static obstacles. Li et al. [21] used APF to improve the performance of a DQN agent in ASV control problems by enabling continuous actions and providing richer reward signals. Heiberg et al. [22] incorporated collision risk indices (CRIs) into the reward function to guide the learning of a COLREGs-compliant RL agent. Wei et al. [23] proposed a multi-agent reinforcement learning approach that promotes collaborative collision avoidance behaviors among vessels.

III. PROBLEM STATEMENT

In this work, we focus on the autonomous navigation problem of ASVs in relatively dense and congested maritime environments. Each ASV is required to navigate to a specified goal location given only measurements from onboard LiDAR and odometry sensors, while avoiding collisions with other surrounding ASVs and buoys.

We use Virtual RobotX (VRX) [7] as the evaluation platform, which is a realistic Gazebo based marine simulation environment developed by Open Robotics [24] and used in the Maritime RobotX Competition [25]. VRX adopts a dynamic model from [26], which is shown as follows:

$$\begin{aligned} \mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \\ \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}. \end{aligned} \quad (1)$$

In the formula, $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ are the 6-dimensional position and velocity vectors. $\boldsymbol{\nu}_r$ is the vessel velocity with respect to the fluid. $\boldsymbol{\tau}$, $\boldsymbol{\tau}_{\text{wind}}$, and $\boldsymbol{\tau}_{\text{wave}}$ are the forces and moments from propulsion, wind and waves.

VRX [7] models the perturbations of wind and waves on in-water objects, and perception error derives from the need to continually segment and track both static and moving obstacles

from raw LiDAR data. We use the default settings in VRX [7], where zero-mean Gaussian noise with 0.01 standard deviation is applied to the LiDAR point cloud, and no noise is added to the odometry measurements.

We use the WAM-V ASV model provided by VRX as the deployment vessel, the motion of which is controlled by thrusts from left and right propellers. We fix the angle of each propeller to zero such that the thrust always aligns with the hull, and the turning motion is achieved by the difference between thrusts. For each propeller, the maximum forward thrust is set to 1000.0 N, and the maximum backward thrust is set to 500.0 N. We denote the thrust range as $[-500.0, 1000.0]$ N. With the given thrust range, the vehicle achieves a maximum forward speed of about 3.3 m/s and a maximum backward speed of about 2.3 m/s.

IV. PERCEPTION PROCESSING

Given LiDAR and odometry measurements, we obtain the state information of the ego vehicle and objects in its immediate surroundings as shown in the following equations:

$$\mathbf{s}_{ego} = [p_x^{goal}, p_y^{goal}, v_x, v_y, w, T_{left}, T_{right}] \quad (2)$$

$$\mathbf{s}_{object} = [\mathbf{o}_1, \dots, \mathbf{o}_n], \mathbf{o}_i = [p_x^i, p_y^i, v_x^i, v_y^i, r_i]. \quad (3)$$

In the ego state \mathbf{s}_{ego} , p_x^{goal} and p_y^{goal} are the x and y coordinates of the given goal location, v_x and v_y are the x and y components of the linear velocity, w is the yaw component of the angular velocity, and T_{left} , T_{right} are thrusts of the left and right propellers. To simplify the perception, we consider each perceived object to be a cylindrical shape. The state of the i -th perceived object consists of its position, $[p_x^i, p_y^i]$, velocity, $[v_x^i, v_y^i]$, and radius r_i . Both ego state and object states are expressed in the frame of the ego vehicle.

The algorithm we use to extract object information from the LiDAR point cloud is based on [27] and described in Algorithm 1, where $R(\cdot)$ computes the distance between a LiDAR point and the LiDAR center, $M(\cdot)$ and $m(\cdot)$ are the max and min operators respectively, and θ is a given threshold to decide whether a point belongs to a cluster. An example of point cloud segmentation is shown in Figure 1, where different clusters are shown in different colors. Given that velocity estimates of distant clusters are less reliable due to the influence of environmental disturbances, we discard LiDAR reflections beyond 20 meters.

After segmentation, each cluster is regarded as an object, and the centroid and radius of the cluster are used as the position and radius of the object. To estimate the velocity of the object, we project the clusters from the vehicle frame at the last time step to the current frame using a relative transformation computed with odometry measurements. For simplicity, we assume that a perceived object with estimated speed smaller than 0.5 m/s is not a vehicle and we do not include it in the COLREGs checking process mentioned in Section V. The resulting ego state and object states will be used in the decision making and control module.

V. DECISION MAKING AND CONTROL WITH RL

A. Problem Formulation

A Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ is used to describe the ASV navigation problem, where \mathcal{S} and \mathcal{A} are the

Algorithm 1: LiDAR point cloud segmentation

Input: LiDAR point clouds P , **Output:** Clusters set C

$i = 0, C[i].centroid = 0.0, C[i].radius = 0.0$

for each point j in P

if j belongs to an existing cluster **then continue**

Initialize a new queue $q \leftarrow \emptyset$, $q.push(j)$, $i = i + 1$

while q is not empty

$k = q.top()$, $C[i].add(k)$, $q.pop()$

Update $C[i].centroid$ and $C[i].radius$ with k

for each point p that is a neighbor of k

$d_1 = M(R(p), R(k)), d_2 = m(R(p), R(k))$

if $\arctan \frac{d_2 \sin \alpha}{d_1 - d_2 \cos \alpha} > \theta$ **then** $q.push(k)$

state space and action space of agents interacting with the environment. $P(\cdot|s, a)$ is the state transition function describing the evolution of environment state, which reflects the dynamics in Equation (1). The reward function $R(s, a)$ generates scalar value signals that indicate the preference towards state-action pairs. At each time step t , based on the observation of the current state s_t , each agent selects an action a_t according to the policy π . Then the environment transitions to the next state s_{t+1} according to $P(\cdot|s_t, a_t)$ and each agent receives a reward $r_{t+1} = R(s_{t+1}, a_{t+1})$. The action value function $Q^\pi(s, a)$ is defined as the expected cumulative reward of taking action a at state s and following the policy π thereafter, where the discount factor $\gamma \in [0, 1)$ reflects the importance of future rewards.

$$Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a] \quad (4)$$

The objective is to obtain an optimal policy π_* that maximizes $Q^\pi(s, a)$ for all state-action pairs, and the resulting optimal action value function $Q^{\pi_*}(s, a)$ satisfies the Bellman optimality equation $Q^{\pi_*}(s, a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} Q^{\pi_*}(s', a')]$.

B. Action Commands

The action commands in each control time step are the variations in the propellers' thrusts. For methods that output discrete actions, the action command of each propeller is chosen from a given action set. We observed that increasing the number of actions in the discretized action space causes difficulties in learning stable control policies, and chose the action set $\mathbf{A} = \{-1000.0, -500.0, 0.0, 500.0, 1000.0\}$ N/s. For methods that operate in continuous action space, each action $a \in [-1000.0, 1000.0]$ N/s. When applying actions, thrusts are clipped within the thrust range noted in Sec. III.

C. Deep Reinforcement Learning

DQN [28] uses a deep neural network model to approximate the action value function $Q(s, a)$, and train it by optimizing the loss based on Temporal Difference (TD) error.

$$\mathcal{L}_{DQN} = \mathbb{E}[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \quad (5)$$

Rainbow [29] outperforms DQN in the Atari 2600 benchmark by incorporating techniques that suppress overestimation and improve stability and efficiency in learning value functions. For continuous control problems, the action space needs to be discretized for DQN and Rainbow to be applied.

DDPG [30] is an actor-critic method that is applicable to continuous action space. It maintains a critic model $Q(s, a|\theta^Q)$ that can be learned with a loss function similar to Equation (5), and an actor model $\mu(s|\theta^\mu)$, the parameters of which are updated using the following policy gradient.

$$\nabla_{\theta^\mu} J \approx \mathbb{E}[\nabla_a Q(s, a|\theta^Q)|_{a=\mu(s)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)] \quad (6)$$

SAC [31] further improves exploration with an objective function (7) that involves the entropy of the learned policy, and uses a stochastic actor with enhanced robustness.

$$J = \sum_{t=0}^T \mathbb{E}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))] \quad (7)$$

D. Distributional Reinforcement Learning

Instead of the expected return $Q^\pi(s, a)$, Distributional RL algorithms [32] learn the return distribution $Z^\pi(s, a)$, where $Q^\pi(s, a) = \mathbb{E}[Z^\pi(s, a)]$, and the distributional Bellman equation, $Z^\pi(s, a) \stackrel{D}{=} R(s, a) + \gamma Z^\pi(s', a')$, is considered. When the dynamics of the operating environment are highly uncertain, the randomness of the collected reward samples adversely affect the accuracy of the learned action value. Distributional RL methods can mitigate this problem by modeling and learning the distribution of cumulative rewards.

Implicit Quantile Networks (IQN) [10] express the return distribution with a quantile function $Z_\tau := F_Z^{-1}(\tau)$, where $\tau \sim U([0, 1])$, and represents the policy as follows.

$$\pi(s) = \operatorname{argmax}_a \frac{1}{K} \sum_{k=1}^K Z_{\tau_k}(s, a), \tau_k \sim U([0, 1]) \quad (8)$$

Parameters of the IQN policy model can be learned by optimizing the loss defined in Equation (11). The outputs of the quantile function, Z_{τ_k} , are referred to as action quantile values. IQN also requires a discretized action space.

$$\delta^{\tau_i, \tau'_j} = r + \gamma Z_{\tau'}(s', \pi(s')) - Z_\tau(s, a) \quad (9)$$

$$\rho_\tau^\kappa(u) = |\tau - \mathbf{1}_{\{u < 0\}}|(\mathcal{L}_\kappa(u)/\kappa),$$

$$\text{where } \mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases} \quad (10)$$

$$\mathcal{L}_{\text{IQN}} = \frac{1}{N'} \sum_{i=1}^N \sum_{j=1}^{N'} \rho_\tau^\kappa(\delta^{\tau_i, \tau'_j}) \quad (11)$$

E. Actor Critic Implicit Quantile Networks

In this work, we develop a novel decision making and control policy for ASV navigation based on IQN within an actor-critic framework, which we denote AC-IQN. Compared to policies based on traditional RL in Sec. V-C, the proposed policy uses Distributional RL, which can be more robust to the marine operating environment and its uncertainties from motion disturbances, perception errors and unknown intents of other vehicles. The proposed policy also exhibits superior maneuverability over the policy based on IQN in Sec. V-D, which is crucial for deployment in congested scenarios.

The learning algorithm of the proposed decision making and control policy based on AC-IQN is shown in Algorithm 2. The Critic Z_θ is similar to the original IQN policy model, which outputs action quantile values given quantile samples and a state-action pair. When computing the Critic loss gradient $\nabla_\theta L$, the policy output of the Actor $\pi_{\phi'}$, and quantile samples $\{\tau_i|i = 1, \dots, N\}$ and $\{\tau_j|j = 1, \dots, N'\}$, drawn from the

Algorithm 2: AC-IQN model update

Input: Critic Z_θ , Actor π_ϕ , replay buffer \mathcal{M}

Sample a minibatch $\{(s, a, r, s')_i\}_{i=1}^M$ from \mathcal{M}

Compute gradient of the critic loss $\nabla_\theta L$, where

$$L = \frac{1}{M} \sum_{i=1}^M \left[\frac{1}{N'} \sum_{i=1}^N \sum_{j=1}^{N'} \rho_{\tau_i}^\kappa(r + \gamma Z_{\theta'}^{\tau_j}(s', \pi_{\phi'}(s')) - Z_\theta^{\tau_i}(s, a)) \right]$$

Compute the policy gradient $\nabla_\phi J$, where

$$\nabla_\phi J = \mathbb{E}\left[\frac{1}{N} \sum_{i=1}^N \nabla_a Z_\theta^{\tau_i}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)\right]$$

Update critic and actor network parameters

$$\theta \leftarrow \theta + \alpha_\theta \cdot \nabla_\theta L, \phi \leftarrow \phi + \alpha_\phi \cdot \nabla_\phi J$$

Update target network parameters every k iterations

$$\theta' \leftarrow \beta\theta + (1 - \beta)\theta', \phi' \leftarrow \beta\phi + (1 - \beta)\phi'$$

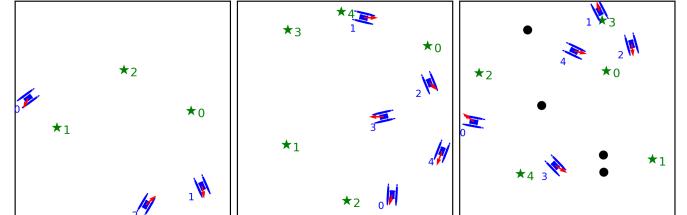


Fig. 2: **Example training scenarios.** The velocity of each vehicle is indicated by the red arrow, goal positions are plotted as green stars, and buoys are shown as black circles.

TABLE I: Curriculum training process.

Timesteps (million)	1st	2nd	3rd	4st	5st	6st
Number of robots	3	4	5	5	5	5
Number of buoys	0	0	0	2	3	4
Min distance between start and goal	30.0	35.0	40.0	40.0	40.0	40.0

uniform distribution $U([0, 1])$, are used. The loss function ρ_τ^κ is defined in Equation (10).

Policy gradient $\nabla_\phi J$ is used to update the Actor π_ϕ , which can be computed according to the chain rule as shown in Algorithm 2. The expectation over state is approximated with those from the sampled minibatch.

F. Training RL agents

We trained our AC-IQN agent, as well as IQN, SAC, DDPG, Rainbow and DQN agents which serve as comparative baselines, and used them in experiments described in Section VI. The SAC and Rainbow agents are based on the implementations of [33] and [34]. Due to the high computational expense of Gazebo's realistic environment simulation, RL agents were trained in a simplified 2D environment. As shown in Equation (12), a simplified three Degree-of-Freedom (DoF) dynamic model described in [26] is used, which only includes surge, sway, and yaw DoFs.

$$\mathbf{M}_{RB}\dot{\mathbf{v}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \quad (12)$$

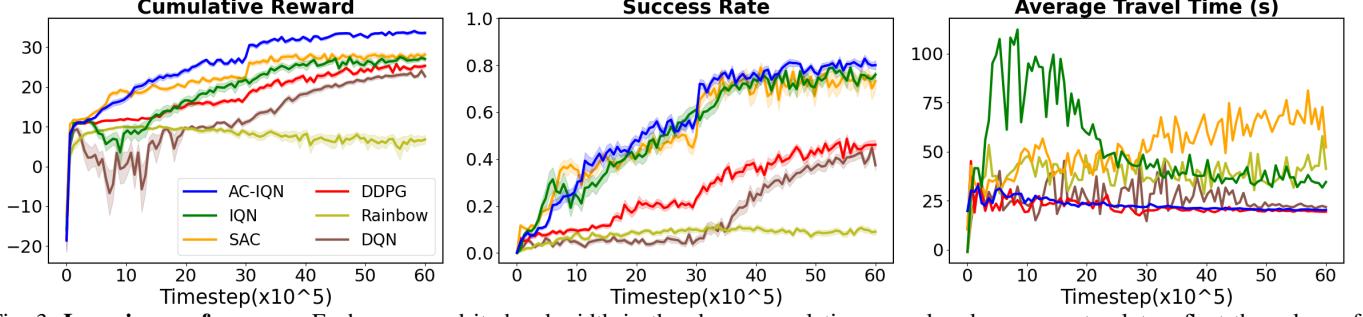


Fig. 3: **Learning performance.** Each curve and its band width in the above cumulative reward and success rate plots reflect the values of the mean and standard error. To compute the average travel time, we only include data from robots that successfully reach their goals.

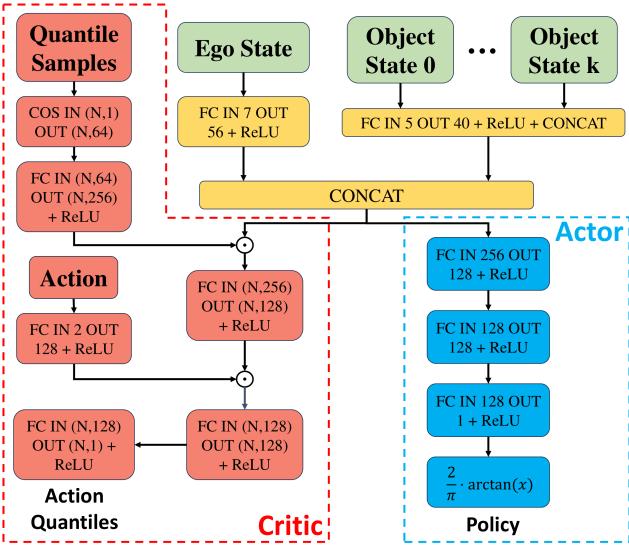


Fig. 4: **AC-IQN network architecture.** FC, ReLU, COS, \odot and CONCAT stand for Fully Connected Layer, Rectified Linear Unit, Cosine Embedding Layer, element-wise product and concatenation of tensors. The numbers after IN and OUT are the input and output dimension of a layer.

To maintain efficient training processes, the perceptual information defined in Eqs. (2) and (3) was directly given, and τ_{wind} and τ_{wave} were assumed to be zero. We introduced noisy perception, defined in Eqs. (13) - (15), to increase the robustness of trained RL agents to motion disturbances and perception errors.

$$P_o = P + w_p, \quad w_p \sim \mathcal{N}(0, \Sigma_p) \quad (13)$$

$$V_o = V + w_v, \quad w_v \sim \mathcal{N}(0, \Sigma_v) \quad (14)$$

$$R_o = R \cdot (r_{mean} + (1 - r_{mean}) \cdot w_r / \pi), \quad w_r \sim \mathcal{V}(0, \kappa) \quad (15)$$

In the above equations, P , V , and R are the ground truth position, velocity and radius of the perceived objects. Position noise w_p and velocity noise w_v are drawn from zero-mean Gaussian distributions. Since the perceived radius is bounded by the actual enclosing radius of the object, we model the radius noise w_r with a von Mises distribution, which is a close approximation to the wrapped normal distribution and lies within $[-\pi, \pi]$. As the training schedule in Table I shows, the complexity of the randomly generated training environment gradually increases as the training proceeds. Example training scenarios are visualized in Figure 2. Similar to our prior work [9], we only maintain one RL model during training, which

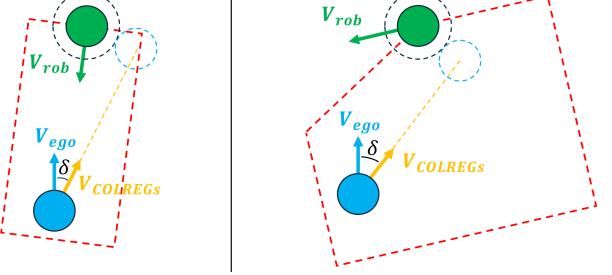


Fig. 5: **Head-on and crossing scenarios.** Velocities of the ego vehicle that are consistent with COLREGs requirements are plotted in yellow. The COLREGs compliant velocity of each vehicle is computed by viewing it as the ego vehicle.

is shared with all vehicles for individual decision making and control tasks.

$$\begin{aligned} r_t = & r_{step} + r_{forward} + r_{COLREGs} \cdot I(s_t \in \mathcal{S}_{COLREGs}) \\ & + r_{collision} \cdot I(s_t \in \mathcal{S}_{collision}) + r_{goal} \cdot I(s_t \in \mathcal{S}_{goal}) \end{aligned} \quad (16)$$

The reward function we use in training is shown in Equation (16), where $I(\cdot)$ is the indicator function, $r_{step} = -0.1$, $r_{forward} = ||p_{t-1}^{goal}|| - ||p_t^{goal}||$, $r_{collision} = -5.0$, $r_{goal} = 10.0$. We considered three cases of two-vehicle encounters where each vehicle shall follow the behavior specified by COLREGs [14] unless other actions are needed to avoid collisions: (1) Overtaking; (2) Head-on; (3) Crossing. For case (1), the vessel at the back shall move out of the way of the vessel being overtaken. $r_{collision}$ penalizes actions leading to collisions, and thus promotes behaviors that conform to the rule of case (1). For case (2), each vessel shall alter course to starboard and pass on the port side of the other. For case (3), the vessel that has the other on its starboard side shall give way. To promote behaviors that follow the rules of cases (2) and (3), which are visualized in Figure 5, each vehicle, ego , will be checked against its closest vehicle, rob : 1) ego is determined to be in a head-on situation if it is in the head-on zone of rob , and the absolute value of the angle from V_{rob} to V_{ego} is greater than $3\pi/4$; 2) ego is the give way vehicle in the crossing situation if it is in the crossing zone of rob , and the angle from V_{rob} to V_{ego} is in $[\pi/4, 3\pi/4]$. If ego is in situation 1) or 2), which we denote as $s_t \in \mathcal{S}_{COLREGs}$, then $r_{COLREGs} = -0.1 \cdot \max(\delta, 0.0)$ will be applied to ego , where $\delta \in [-\pi, \pi]$ is the angle from V_{ego} to $V_{COLREGs}$ (clockwise is positive). $\mathcal{S}_{collision}$ and \mathcal{S}_{goal} are situations where ego collides with other objects and reaches the goal respectively.

The network structure for the AC-IQN agent is shown in Figure 4. The Cosine Embedding Layer computes a feature $[\cos(\pi \cdot 0 \cdot \tau), \dots, \cos(\pi \cdot 63 \cdot \tau)]$ for each quantile sample τ . The outputs of Actor are numbers in $(-1.0, 1.0)$, which are then multiplied by 1000.0 to match the action range. The structure of the networks used by IQN, DDPG and DQN agents are mostly the same as AC-IQN, except for necessary modifications to match the corresponding input and output.

The learning performances of the AC-IQN, IQN, SAC, DDPG, Rainbow and DQN agents are shown in Figure 3. Considering the effect of random seed selection on training, we train 30 models with different random seeds for each agent on an Nvidia RTX 3090 GPU and show the general performance. AC-IQN shows a clear advantage in the cumulative reward obtained during the training process relative to other methods, and achieves the highest success rate with a minimal amount of travel time. IQN and SAC achieve similar levels of safety performance as AC-IQN, but at the cost of longer average travel time in general. On the contrary, DDPG and DQN can complete navigation tasks efficiently, but have much lower success rates than AC-IQN. Rainbow does not succeed in learning safe and efficient policies, which may result from the effects of substantial observation noise and uncertainty in multi-vehicle interactions on the performance of Rainbow components like multi-step return and prioritized replay.

VI. EXPERIMENTS

Each navigation system evaluated in our experiments requires the integration of the perception module introduced in Sec. IV and a decision making & control agent into the Robot Operating System (ROS 2) framework, and we use all six RL methods described in Sec. V in the experiments. For each RL method, we choose the model with average training success rate performance of all trained models for use in our experiments. In addition to RL agents, we also include two classical methods, Artificial Potential Fields (APF) and Model Predictive Control (MPC), which also consider COLREGs in ASV navigation and are described in the following paragraphs, as baselines.

The experiments were run on an AMD Ryzen threadripper 3970X CPU and used the `sydney_regatta` environment of the VRX simulator. We approximate the scenario of Fig. 5 in [7] and set the average wind speed to 10 m/s. The wave parameters of period and gain ratio, defined in [7], are respectively set to 5.0 sec. and 0.3 to approximate the conditions of the inland water at `sydney_regatta`. We performed six sets of experiments with an increasing level of complexity reflected in the number of vehicles and buoys, where each set includes 100 experiments for each navigation system. Examples of experiment scenarios are visualized in Figure 6. For an experiment episode, all vehicles are equipped with the same navigation system, and the episode is considered failed if a collision happens or the travel time of any vehicle exceeds 90 seconds.

The APF agent we implemented is based on [35] and [16]. COLREGs-compliant behaviors are promoted by generating virtual obstacles to prevent vehicles from entering COLREGs-violating locations. The repulsive forces $U_{\text{rep}} =$

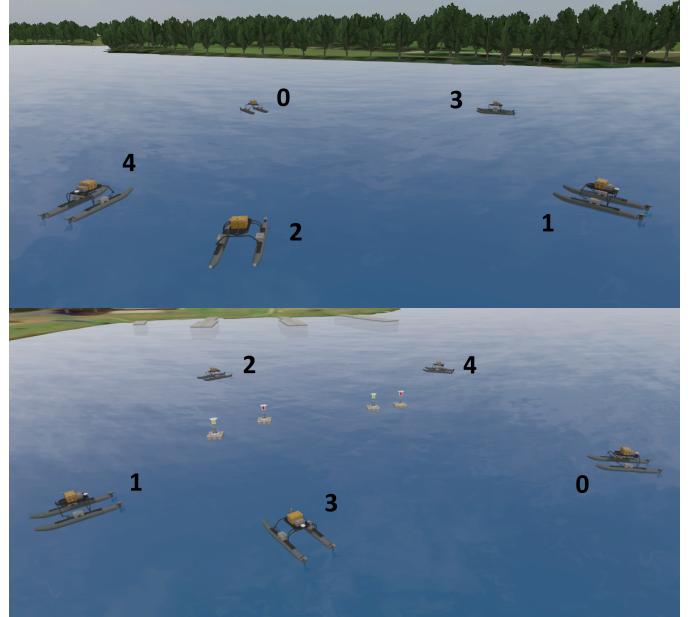


Fig. 6: Example VRX experiment episodes (with and without buoys). The initial vehicle poses and the buoy positions are shown.

$0.5 \cdot k_{\text{rep}} \cdot (1/d(X, X_o) - 1/d_0)^2 \cdot d^2(X, X_g)$ are generated when $d(X, X_o) \leq d_0$. X and X_g are positions of the robot and goal, and X_o are the positions of perceived objects and virtual obstacles. The attractive force $U_{\text{att}}(X) = 0.5 \cdot k_{\text{att}} \cdot d^2(X, X_g)$. We use $k_{\text{att}} = 50.0$, $k_{\text{rep}} = 500.0$, $d_0 = 15.0$, and total force $F = -\nabla U_{\text{att}}(X) - \nabla U_{\text{rep}}(X)$. To map the force to an action command, we forward simulate all combinations of action commands from the action set **A** defined in Sec. V-B for one control time step, and choose the one that minimizes the difference between velocity at the next step and F .

The MPC agent we have implemented is similar to [5]. The idea is to predict scenarios $k \in \{1, 2, \dots, N_s\}$ corresponding to different motion commands and states of objects $i \in \{1, 2, \dots, N_o\}$ in the future horizon from t_0 , and choose the control behavior $k^*(t_0) = \arg \min_k \mathcal{H}^k(t_0)$, where $\mathcal{H}^k(t_0) = \max_i \max_{t \in \mathcal{D}(t_0)} (\mathcal{C}_i^k(t) \mathcal{R}_i^k(t) + \kappa_i \mathcal{M}_i^k(t) + \lambda_i \mathcal{T}_i^k(t)) + f(u_m^k, \mathcal{X}_m^k)$. As defined in [5], $\mathcal{C}_i^k(t)$ are $\mathcal{R}_i^k(t)$ are collision cost and collision risk factor, $\mathcal{M}_i^k(t)$ and κ_i are the COLREGs violation cost and a tuning parameter, $\lambda_i \mathcal{T}_i^k(t)$ is the COLREGs-transitional cost, and $f(u_m^k, \mathcal{X}_m^k)$ is the cost of maneuvering effort. For each combination of actions from the action set **A** in Sec. V-B, we simulate the scenario for a horizon of $T = 5$ control time steps, and choose the one that minimizes the cost $\mathcal{H}^k(t_0)$. We set $\mathcal{C}_i^k(t) = 50.0$, $\kappa_i = 1.0$, $\mathcal{M}_i^k(t) = 10.0$, and $\lambda_i = 2.0$ when the situations corresponding to the cost terms happen, otherwise these values are set to zero. Other terms remain the same as defined in [5].

Our experimental results are shown in Figure 8 and Table II. In general, the AC-IQN based system achieves the highest success rate, with minimal average travel time. APF and MPC based systems exhibit worse safety performance than RL based systems, especially in complex cases. RL agents have been trained to be robust to the influences of environmental disturbances on ASV motion and perception, and learn safety-preserving behaviors during the training process. Among RL

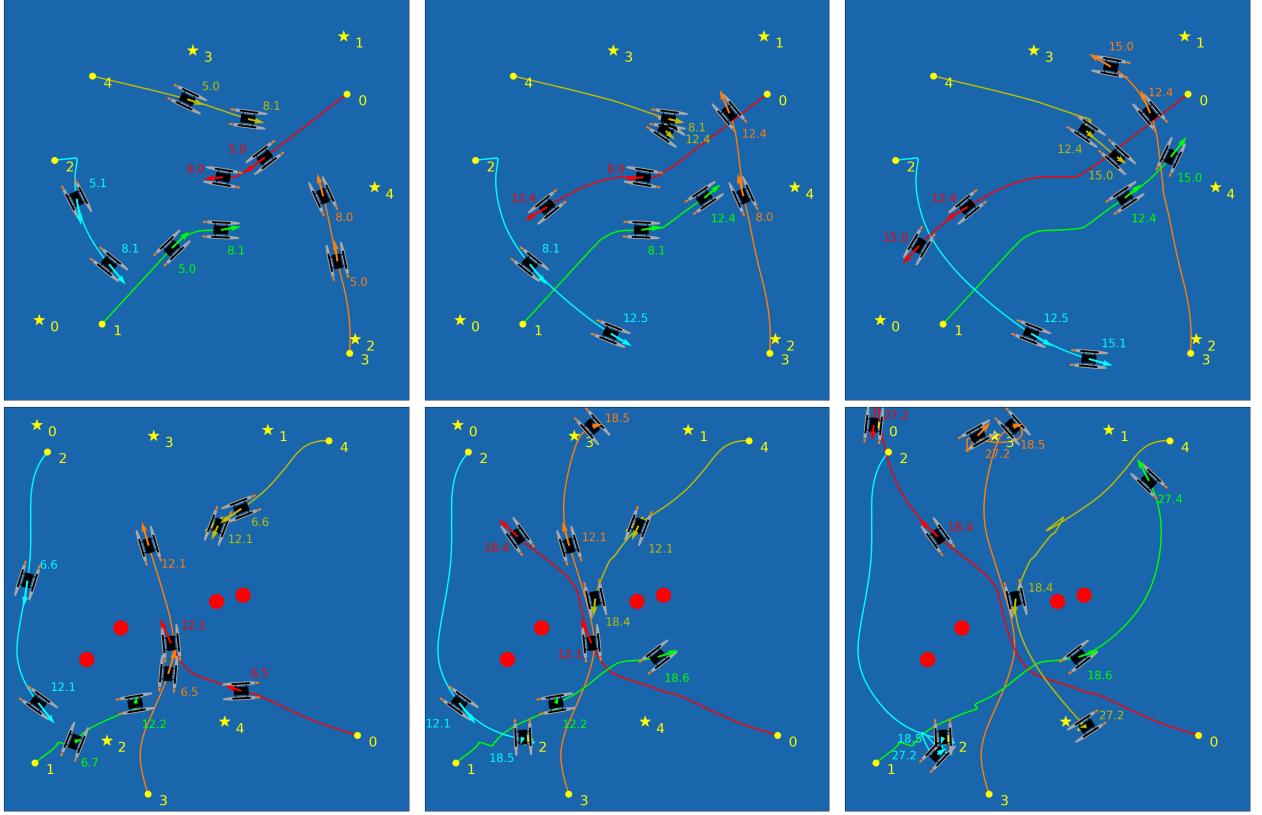


Fig. 7: Trajectories of AC-IQN based system in VRX experiments shown in Fig. 6. Yellow dots and stars indicate the start and goal positions of the vehicles, and the indices next to them match those in Fig. 6. Vehicle poses are depicted with the vehicle icons shown. Vehicle velocities and timestamps (in seconds) are shown as arrows and numbers with one decimal place in corresponding colors.

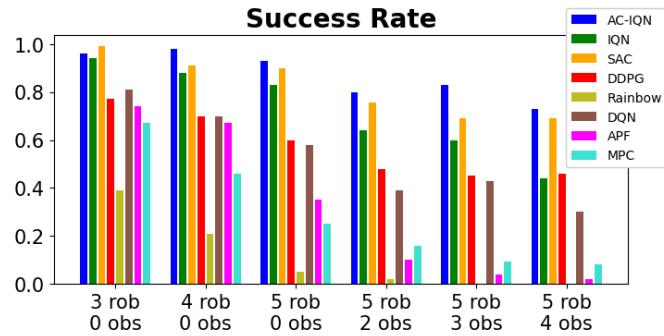


Fig. 8: Success rates in VRX experiments. Labels on the x -axis indicate the number of vehicles (rob) and buoys (obs) used in corresponding sets of experiments.

agents, the SAC based system demonstrates the most competitive safety performance with AC-IQN throughout all sets of experiments, but SAC requires over 40% more travel time on average than the latter. Accordingly, AC-IQN offers superior performance in complex multi-ASV navigation scenarios by combining continuous space maneuverability and the efficacy of Distributional RL in highly uncertain environments.

The initial configurations of two example experiment episodes are shown in Fig. 6, and the behaviors of our AC-IQN based system are demonstrated in Fig. 7. In each plot of Fig. 7, we show vehicle poses and velocities at two timestamps to clarify the relative movements. There are slight differences in timestamps because trajectories and corresponding timestamps are recorded by vehicles individually, and we visualize those

Avg. travel time (s)	3 rob 0 obs	4 rob 0 obs	5 rob 0 obs	5 rob 2 obs	5 rob 3 obs	5 rob 4 obs
AC-IQN	17.22	17.85	20.22	21.06	21.97	22.81
IQN	29.19	30.06	36.77	36.65	41.21	39.64
SAC	24.09	24.75	29.96	29.50	31.54	32.35
DDPG	18.22	18.24	20.51	20.89	21.34	21.58
Rainbow	36.46	36.56	43.33	47.32	—	—
DQN	19.39	23.38	21.91	22.24	23.40	23.25
APF	22.37	23.07	25.76	23.82	24.62	24.53
MPC	17.96	19.10	20.71	28.80	33.16	33.61

TABLE II: Travel time data from VRX experiments. Rainbow based system fails completely in sets 5 rob 3 obs and 5 rob 4 obs.

with the smallest time differences.

As shown in the plots on the top row of Fig. 7, five vehicles are navigating with no buoys present in the environment. Vehicles 0 and 1 approach each other in a head-on manner, and both decide to turn right at around 5.0 seconds to avoid a potential collision. Vehicle 4 gets close to vehicle 3 from the port side of the latter, then it stays and gives way to vehicle 3 from around 8.0 seconds to 12.5 seconds. After vehicle 3 passes, vehicle 4 approaches another vehicle, vehicle 1, from the port side of the other again, and it slows down to wait for clearance. Therefore, the AC-IQN based system shows the capability of generating actions that follow COLREGs.

At bottom of Fig. 7, four buoys exist and form a passage in the middle. Due to the existence of buoys, the motions of vehicles are constrained and it would be difficult and even unsafe to follow COLREGs. It can be seen from the plots on

the bottom row of Fig. 7 that vehicles 3, 0, and 1 approach the passage from the lower side in sequence, and vehicle 4 approaches from the upper side. The vehicles pass in a first-in-first-out manner, where the one being closest to the passage gets to cross while others slow down and give way. Being the last one waiting in the line, vehicle 1 decides to detour to save time. This episode shows that when facing difficulties in following COLREGs in a complex environment, the AC-IQN based system can maneuver in a way that is beneficial to navigation safety and efficiency.

VII. CONCLUSION AND FUTURE WORK

We propose a novel ASV autonomous navigation system that integrates a perception module that works with onboard LiDAR and odometry sensors, and a decision making and control module using Distributional RL that can generate arbitrary control commands in continuous action space. The proposed system is extensively evaluated in realistic Gazebo simulations against seven baseline systems based on state-of-the-art Distributional RL, non-Distributional RL and classical methods, and demonstrates superior performance in navigation safety and efficiency over baseline systems. In future work, our system performance may be further improved by introducing risk sensitivity, and by conducting real-world ASV field tests with onboard LiDAR, GPS and IMU.

REFERENCES

- [1] A. Vagale, R. Ouchekh, R. T. Bye, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles I: A Review," *Journal of Marine Science and Technology*, pp. 1–15, 2021.
- [2] L. Zhao and M.-I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, p. 106436, 2019.
- [3] Y. Cho, J. Han, and J. Kim, "Efficient COLREG-compliant collision avoidance in multi-ship encounter situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1899–1911, 2020.
- [4] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGs, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2013.
- [5] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Mpc-based collision avoidance strategy for existing marine vessel guidance systems," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7618–7623.
- [6] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, "The branching-course model predictive control algorithm for maritime collision avoidance," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1222–1249, 2019.
- [7] B. Bingham, C. Aguero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar, "Toward maritime robotic simulation in gazebo," in *Proceedings of MTS/IEEE OCEANS Conference*, Seattle, WA, October 2019.
- [8] M. G. Bellemare, W. Dabney, and M. Rowland, *Distributional reinforcement learning*. MIT Press, 2023.
- [9] X. Lin, Y. Huang, F. Chen, and B. Englot, "Decentralized multi-robot navigation for autonomous surface vehicles with distributional reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 8327–8333.
- [10] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1096–1105.
- [11] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, T. Dhruba, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," in *International Conference on Learning Representations*, 2018.
- [12] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 11, pp. 6584–6598, 2021.
- [13] R. Singh, K. Lee, and Y. Chen, "Sample-based distributional policy gradient," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 676–688.
- [14] International Maritime Organization, "Convention on the international regulations for preventing collisions at sea, 1972 (COLREGs)," 1972.
- [15] M. Jeong and A. Q. Li, "Motion attribute-based clustering and collision avoidance of multiple in-water obstacles by autonomous surface vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 6873–6880.
- [16] W. Naeem, S. C. Henrique, and L. Hu, "A reactive colregs-compliant navigation strategy for autonomous maritime navigation," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 207–213, 2016.
- [17] H.-T. L. Chiang and L. Tapia, "Colreg-rrt: An rrt-based colregs-compliant motion planner for surface vehicle navigation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.
- [18] Y. Cho, J. Kim, and J. Kim, "Intent inference-based ship collision avoidance in encounters with rule-violating vessels," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 518–525, 2021.
- [19] C. Jia, J. Ma, B. de Vries, and W. M. Kouw, "Bayesian inference of collision avoidance intent during ship encounters," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [20] E. Meyer, A. Heiberg, A. Rasheed, and O. San, "Colreg-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning," *Ieee Access*, vol. 8, pp. 165 344–165 364, 2020.
- [21] L. Li, D. Wu, Y. Huang, and Z.-M. Yuan, "A path planning strategy unified with a colregs collision avoidance function based on deep reinforcement learning and artificial potential field," *Applied Ocean Research*, vol. 113, p. 102759, 2021.
- [22] A. Heiberg, T. N. Larsen, E. Meyer, A. Rasheed, O. San, and D. Varagnolo, "Risk-based implementation of colregs for autonomous surface vehicles using deep reinforcement learning," *Neural Networks*, vol. 152, pp. 17–33, 2022.
- [23] G. Wei and W. Kuo, "Colregs-compliant multi-ship collision avoidance based on multi-agent reinforcement learning technique," *Journal of Marine Science and Engineering*, vol. 10, no. 10, p. 1431, 2022.
- [24] Open Robotics. <https://www.openrobotics.org/>.
- [25] The Maritime RobotX Challenge. <https://robotx.org/about/>.
- [26] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.
- [27] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 163–169.
- [28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [29] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
- [32] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 449–458.
- [33] J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, Y. Su, H. Su, and J. Zhu, "Tianshou: A highly modularized deep reinforcement learning library," *Journal of Machine Learning Research*, vol. 23, no. 267, pp. 1–6, 2022.
- [34] K. Arulkumaran, "Rainbow," 2019. [Online]. Available: <https://github.com/Kaixhin/Rainbow>
- [35] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu, "Improved artificial potential field method applied for AUV path planning," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–21, 2020.