

Decentralized Multi-Robot Navigation for Autonomous Surface Vehicles with Distributional Reinforcement Learning

Xi Lin, Yewei Huang, Fanfei Chen, and Brendan Englot

Abstract—Collision avoidance algorithms for Autonomous Surface Vehicles (ASV) that follow the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) have been proposed in recent years. However, it may be difficult and unsafe to follow COLREGs in congested waters, where multiple ASVs are navigating in the presence of static obstacles and strong currents, due to the complex interactions. To address this problem, we propose a decentralized multi-ASV collision avoidance policy based on Distributional Reinforcement Learning, which considers the interactions among ASVs as well as with static obstacles and current flows. We evaluate the performance of the proposed Distributional RL based policy against a traditional RL-based policy and two classical methods, Artificial Potential Fields (APF) and Reciprocal Velocity Obstacles (RVO), in simulation experiments, which show that the proposed policy achieves superior performance in navigation safety, while requiring minimal travel time and energy. A variant of our framework that automatically adapts its risk sensitivity is also demonstrated to improve ASV safety in highly congested environments.

I. INTRODUCTION

Reliable collision avoidance is crucial to the deployment of Autonomous Surface Vehicles (ASV), which is still a challenging problem [1], especially in congested environments. The Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) [2] defines rules for collision avoidance between vessels, which have been used to develop obstacle avoidance solutions in multi-ASV navigation scenarios [3], [4], [5], [6]. However, following COLREGs may cause conflicting actions when the roles of each ego vehicle with respect to different neighboring vehicles are in conflict, especially in congested waters [7]. In addition to the risk of collision with other vehicles, static obstacles that exist in the marine environment, such as buoys or rocks, may pose threats to navigation safety. To avoid collision with them, an ASV may need to maneuver in a way that does not comply with COLREGs, which primarily consider the relationship to nearby vehicles. Additionally, the motion of marine autonomous vehicles is affected by current flows [8], thus following COLREGs may not always be safe when navigating in strong current. The objective of this work is to develop a safe decentralized ASV collision avoidance policy for the multi-ASV navigation problem in congested marine environments where unknown static obstacles and currents exist.

Deep Reinforcement Learning (DRL) combines the ability of Reinforcement Learning to adapt to unknown environ-

Xi Lin, Yewei Huang and Brendan Englot are with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA, {xlin26, yhuang85, benglot}@stevens.edu.

Fanfei Chen is with Aescap, New York, NY, USA.

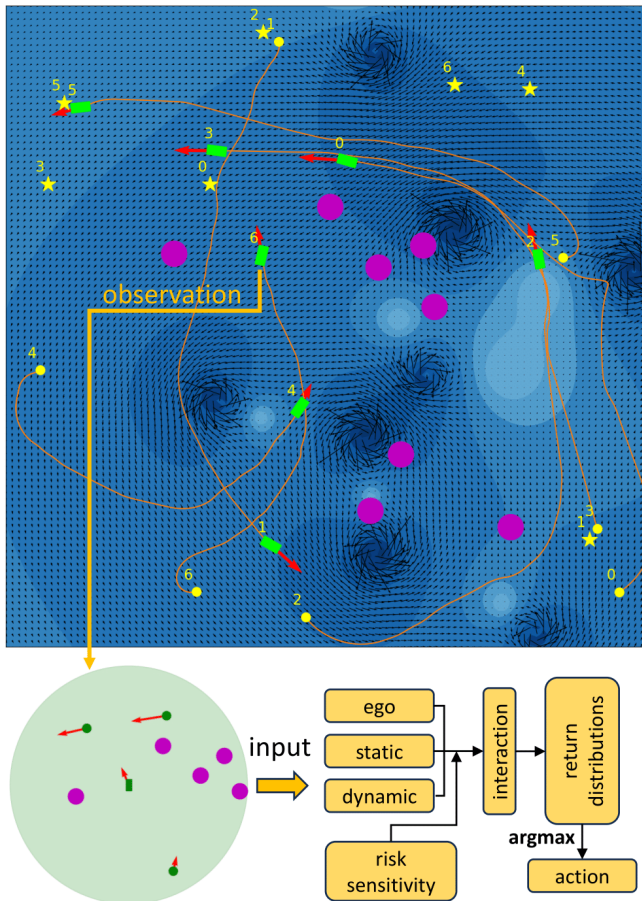


Fig. 1: Our decentralized decision making framework.

ments through trial-and-error learning [9] with the strong representation capability of deep neural networks, and has been applied to various robotics control problems, including to ASVs [10], [11]. Different from DRL that aims to learn the expected return of actions, Distributional Reinforcement Learning (Distributional RL) learns *return distributions* [12], enabling the deployment of risk-sensitive policies by distorting return distributions with risk measures [13] for enhanced safety performance.

Building on our previous work on single ASV sensor-based navigation with Distributional RL [14], we propose a Distributional RL based policy for decentralized multi-ASV collision avoidance with the following contributions: (1) We design a policy network structure that encodes interaction relations among the ego vehicle, static obstacles and other vehicles, and develop a decentralized multi-agent training framework that induces a coordinated collision avoidance policy by parameter sharing among training agents. (2) The

proposed policy based on Distributional RL is compared against a traditional DRL policy and classical methods, including Artificial Potential Fields (APF) and Reciprocal Velocity Obstacles (RVO), in simulation experiments. The results indicate that the proposed framework achieves the highest success rate over all levels of environmental complexity, with nearly the smallest amount of travel time and energy cost. When equipped with a proposed variant of our algorithm that automatically adapts risk sensitivity, the navigation safety of the proposed policy can be clearly improved in highly congested environments. (3) Our approach has been made freely available at <https://github.com/RobustFieldAutonomyLab/Multi-Robot-Distributional-RL-Navigation>.

II. RELATED WORKS

Traditional non-RL-based methods are widely used to solve the problem of navigating multiple robots in a decentralized manner. Since the original proposal of using Artificial Potential Fields (APF) in mobile robot navigation [15], it has also been used in multi-robot scenarios, by considering other robots as moving obstacles and generating repulsive forces according to their positions and velocities [16], [17]. Fiorini et al. [18] applied the concept of Velocity Obstacles to the dynamic obstacle avoidance problem, which identifies the set of velocities leading to future collisions with moving obstacles. Reciprocal Velocity Obstacles (RVO) [19], [20] and Optimal Reciprocal Collision Avoidance (ORCA) [21], [22] improved the performance of velocity obstacle methods on the multi-robot navigation problem by selecting velocities that exhibit reciprocal obstacle avoidance behaviors. Kim et al. [23] proposed to predict the future motion of neighboring agents with past observations of them by Ensemble Kalman Filtering, showing the robustness of the method to noisy, low-resolution sensors, and missing data. To solve the freezing robot problem when navigating in cluttered environments, Trautman et al. [24], [25] modeled the interactions among agents with Gaussian processes to achieve cooperative navigation.

Decentralized multi-agent collision avoidance solutions based on Deep Reinforcement Learning (DRL) have been developed in recent years. Chen et al. [26], [27] proposed DRL based methods that aim at minimizing the expected time to goal while avoiding collisions, and can show socially aware behaviors in pedestrian-rich environments. Everett et al. [28] deal with varying numbers of observed agents by using long short-term memory (LSTM) structures, which improves the policy performance when the number of agents increases. Long et al. [29] employed a centralized learning and decentralized execution framework that trains agents capable of navigating in complex environments with a large number of robots. Chen et al. [30], [31] modeled the higher-order interactions among agents, which consider not only the direct interactions between the ego agent and other agents, but also the indirect effects of interactions among other agents on the ego agent. In addition to the observations of other agents, Liu et al. [32] included an additional occupancy

grid map or an angular map to encode static obstacle information, improving navigation performance in environments cluttered with static obstacles. Han et al. [33] trained DRL agents that exhibit reciprocal collision avoidance behaviors by designing a reward function based on RVO.

Compared to DRL, Distributional RL methods capture *return distributions* instead of only the expected return [12], and these methods have been used to train risk-sensitive agents for a variety of navigation tasks [34], [35], [36], [37], [14] in recent years. These works focus on the single robot navigation problem in environments with static obstacles or dynamic obstacles moving along pre-defined trajectories. We explore the application of Distributional RL to the *decentralized multi-robot navigation problem* in this paper.

III. PROBLEM SETUP

Multiple robots are required to navigate in a simulated marine environment with no prior information describing current flows and static obstacles. At each time step t , each robot receives an observation of its own state, and of the states of static obstacles and other robots within the detection range, which can be expressed as a combined state vector $\mathbf{s}_t = [\mathbf{s}_{ego}^t, \mathbf{s}_{static}^t, \mathbf{s}_{dynamic}^t]$, where \mathbf{s}_{ego} contains the goal position and ego velocity, \mathbf{s}_{static} contains the position and radius of detected obstacles, and $\mathbf{s}_{dynamic}$ contains the position and velocity of other detected robots.

$$\mathbf{s}_{ego} = [p_x^{goal}, p_y^{goal}, v_x, v_y] \quad (1)$$

$$\mathbf{s}_{static} = [\mathbf{o}_s^1, \dots, \mathbf{o}_s^m], \quad \mathbf{o}_s^i = [p_{sx}^i, p_{sy}^i, r_s^i] \quad (2)$$

$$\mathbf{s}_{dynamic} = [\mathbf{o}_d^1, \dots, \mathbf{o}_d^n], \quad \mathbf{o}_d^i = [p_{dx}^i, p_{dy}^i, v_{dx}^i, v_{dy}^i] \quad (3)$$

The pose of each robot can be represented as (x, y, θ) , where (x, y) are the global Cartesian coordinates of the robot, and θ is its orientation. As shown in Fig. 1, \mathbf{s}_t is expressed in the robot frame. The objective is to obtain a policy $\pi(\mathbf{s})$ that minimizes the expected travel time to a specified goal, while avoiding collisions with static obstacles and other robots.

$$\begin{aligned} & \arg \min_{\pi(\mathbf{s})} \mathbb{E}[t_{goal} | \mathbf{s}_0, \pi], \\ & s.t. \quad \|\mathbf{p}_s^{i,t}\| > r_s^{i,t} + r_{rob} \quad \forall t, \quad \mathbf{o}_s^{i,t} \in \mathbf{s}_{static}^t \\ & \quad \|\mathbf{p}_d^{i,t}\| > 2 \cdot r_{rob} \quad \forall t, \quad \mathbf{o}_d^{i,t} \in \mathbf{s}_{dynamic}^t \\ & \quad \|\mathbf{p}^{goal,t_{goal}}\| \leq d_{thres} \end{aligned} \quad (4)$$

In the above formula, $\mathbf{p}_s = [p_{sx}, p_{sy}]$, $\mathbf{p}_d = [p_{dx}, p_{dy}]$, r_{rob} is the collision radius of the robot, and the robot is considered to reach its goal if its Euclidean distance to the goal is within the threshold d_{thres} .

The simulation environment we use is similar to our prior work [14]. The Rankine vortex model [38] is utilized to simulate current flows, where the angular velocity of the vortex core $\Omega = \Gamma / (2\pi r_0^2)$.

$$v_r = 0, \quad v_\theta(r) = \frac{\Gamma}{2\pi} \begin{cases} r/r_0^2, & \text{if } r \leq r_0 \\ 1/r, & \text{if } r > r_0 \end{cases} \quad (5)$$

The kinematic model described in [39] is used to model the effect of current flow on robot motion.

$$\frac{d\mathbf{X}(t)}{dt} = \mathbf{V}(t) = \mathbf{V}_C(\mathbf{X}(t)) + \mathbf{V}_S(t) \quad (6)$$

$\mathbf{V}(t)$ is the total velocity of the robot, $\mathbf{V}_C(\mathbf{X}(t))$ is the current flow velocity at robot position $\mathbf{X}(t)$, and $\mathbf{V}_S(t)$ is the robot steering velocity. The robot action at time step t is $\mathbf{a}_t = (a_t, w_t)$, where a is the rate of change in the magnitude of $\mathbf{V}_S(t)$, and w is the rate of change in the direction of $\mathbf{V}_S(t)$. Then the navigation policy can be expressed as $\pi : \mathbf{s} \rightarrow (a, w)$. Linear acceleration a and angular velocity w may be selected as follows: $a \in \{-0.4, 0.0, 0.4\}$ m/s², and $w \in \{-0.52, 0.0, 0.52\}$ rad/s. The forward speed is clipped to $[0, v_{max}]$.

IV. APPROACH

A. Reinforcement Learning

We formulate the problem as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where \mathcal{S} and \mathcal{A} are the sets of states and actions of the agents described in Section III. $P(s'|s, a)$ is the state transition function reflecting the environment dynamics introduced in Section III, and is unknown to the agent due to the unobservable current flow and intents of the other agents. $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each time step t , given the observation of the current state s_t , the agent chooses an action a_t , which causes the transition to the state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and receiving the reward $r_{t+1} = R(s_{t+1}, a_{t+1})$. The action value function $Q^\pi(s, a)$ is defined as the expected return of taking action a at state s and following the policy π thereafter.

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \quad (7)$$

An optimal policy π_* maximizes $Q^\pi(s, a)$, and the resulting optimal action value function $Q^{\pi^*}(s, a)$ satisfies the Bellman optimality equation (8).

$$Q^{\pi^*}(s, a) = \mathbb{E} [r_{t+1} + \gamma \max_{a'} Q^{\pi^*}(s', a')] \quad (8)$$

The reward function we use is shown as follows, where $r_{step} = -1.0$ and $r_{forward,t} = \|\mathbf{p}^{goal,t-1} - \mathbf{p}^{goal,t}\|$ encourage the agent to move towards the goal, $\mathbf{1}$ is the indicator function, and $r_{collision} = -50.0$ or $r_{goal} = 100.0$ is given when the agent reaches the goal or collides, respectively.

$$r_t = r_{step} + r_{forward,t} + \mathbf{1}_{collision}(s_t) \cdot r_{collision} + \mathbf{1}_{reach_goal}(s_t) \cdot r_{goal} \quad (9)$$

B. Traditional DRL

DQN [40] represents the action value function as a neural network model $Q(s, a; \theta)$, and learns the network parameters θ by optimizing the loss \mathcal{L}_{DQN} computed with samples in the format of (s, a, r, s') from experiences.

$$\mathcal{L}_{DQN} = \mathbb{E} [(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \quad (10)$$

C. Distributional RL

Instead of the expected return $Q^\pi(s, a)$, Distributional RL algorithms [12] learn the return distribution $Z^\pi(s, a)$, where $Q^\pi(s, a) = \mathbb{E}[Z^\pi(s, a)]$, and the distributional Bellman equation is considered.

$$Z^\pi(s, a) \stackrel{D}{=} R(s, a) + \gamma Z^\pi(s', a') \quad (11)$$

Our planner is based on implicit quantile networks (IQN) [13], which expresses the return distribution with a quantile function $Z_\tau := F_Z^{-1}(\tau)$, where $\tau \sim U([0, 1])$, and enables the incorporation of a distortion risk measure $\beta : [0, 1] \rightarrow [0, 1]$ to compute a risk distorted expectation $Q_\beta(s, a)$ as well as a risk-sensitive policy π_β . We use $K = 32$.

$$Q_\beta(s, a) = \mathbb{E}_{\tau \sim U([0,1])} [Z_{\beta(\tau)}(s, a)] \quad (12)$$

$$\pi_\beta(s) = \operatorname{argmax}_a \frac{1}{K} \sum_{k=1}^K Z_{\beta(\tilde{\tau}_k)}(s, a), \quad \tilde{\tau}_k \sim U([0, 1]) \quad (13)$$

Network parameters of the IQN model can be learned by optimizing the loss \mathcal{L}_{IQN} . We use $N = N' = 8$, $\kappa = 1.0$.

$$\delta^{\tau_i, \tau'_j} = r + \gamma Z_{\tau'}(s', \pi_\beta(s')) - Z_\tau(s, a) \quad (14)$$

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

$$\text{where } \mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases} \quad (15)$$

$$\mathcal{L}_{IQN} = \frac{1}{N'} \sum_{i=1}^N \sum_{j=1}^{N'} \rho_{\tau_i}^\kappa(\delta^{\tau_i, \tau'_j}) \quad (16)$$

D. Training Decentralized Policy in Multi-Agent Scenarios

The network architecture we use for the IQN agent is shown in Fig. 2. To create a fixed-size input for the model, the observed static obstacles and other robots are sorted according to the distance to the ego robot, and maximum $m = 5$ static obstacles and $n = 5$ other robots closest to the ego robot are considered in the input. The input is padded with zeros if the number of observed static obstacles and other robots are smaller than m and n . Observations of the ego robot, static obstacles and other robots are transformed into corresponding latent features through the observation encoder. To encode risk sensitivity, the Conditional Value at Risk (CVaR) (17) is used, which scales the quantile sample τ with the CVaR threshold ϕ , and uses it to compute a set of cosine features (18). After observation features are combined with risk sensitivity features, they are passed through two fully connected layers to model interactions among the ego robot, static obstacles and other robots, and outputs the return distributions of actions. The DQN agent uses the same network model as Fig. 2, except that risk sensitivity and the quantile encoder are not used, and its output is composed of action values.

$$f(\tau; \phi) = \phi \tau, \quad \phi \in (0, 1), \quad \tau \sim U([0, 1]) \quad (17)$$

$$[\cos(\pi \cdot 0 \cdot f(\tau; \phi)), \dots, \cos(\pi \cdot 63 \cdot f(\tau; \phi))] \quad (18)$$

The process of training IQN and DQN agents is described in Algorithm 1. At the beginning of every training episode, a new training environment with randomly generated robots, vortices and static obstacles is created according to the training schedule shown in Table I, which is designed to gradually increase the complexity of the environment as training proceeds. Example training environments with different levels of complexity are shown in Fig. 3. The radius of each static obstacle is set to 1.0 meters.

Algorithm 1: Policy training algorithm

$\mathcal{R}, \{s_i | Rob_i \in \mathcal{R}\} \leftarrow$ initialize env, $l_{episode} \leftarrow 0$
 $\mathcal{M} \leftarrow \emptyset, \pi \leftarrow$ DQN or IQN, $\mathcal{L} \leftarrow \mathcal{L}_{DQN}$ or \mathcal{L}_{IQN}
for $t = 1, \dots, t_{total}$ **do**
 $\epsilon \leftarrow f_{linear}(t, t_{total}, \epsilon_{max}, \epsilon_{min})$
 $\mathcal{A} \leftarrow \{a_i = f_{\epsilon-greedy}(\pi_{\theta}(s_i), \epsilon) | Rob_i \in \mathcal{R}\}$
 $\{(s'_i, r_i) | Rob_i \in \mathcal{R}\} \leftarrow$ execute \mathcal{A} in env
 $\mathcal{M} \leftarrow$ add $\{(s_i, a_i, r_i, s'_i) | Rob_i \in \mathcal{R}\}$
 $\mathcal{R} \leftarrow \mathcal{R} \setminus \{Rob_i | Rob_i \text{ reaches goal or collides}\}$
 if $t \bmod t_{learn_freq} = 0$ **then**
 $\mathcal{B} = \{(s, a, r, s')_k | k = 1, \dots, n_{batch}\} \leftarrow$
 sample batch from \mathcal{M}
 $\theta \leftarrow$ optimize $\mathcal{L}(\mathcal{B}, \pi_{\theta})$ w.r.t θ
 if $t \bmod t_{eval_freq} = 0$ **then**
 evaluate π_{θ}
 if $l_{episode} > l_{episode_max}$ or $\mathcal{R} = \emptyset$ **then**
 $\mathcal{R}, \{s_i | Rob_i \in \mathcal{R}\} \leftarrow$ reset env, $l_{episode} \leftarrow 0$
 else
 $s_i \leftarrow s'_i$ for all $Rob_i \in \mathcal{R}$, $l_{episode} += 1$

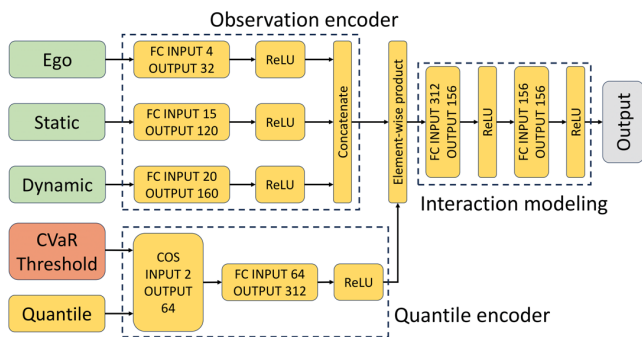


Fig. 2: **IQN network model.** FC, COS and ReLU stand for fully connected layer, cosine embedding layer, and rectified linear unit. Outputs of the model are the return distributions of actions.

To induce coordinated obstacle avoidance behavior among robots, we maintain only one learning model, which is shared with all robots during their individual decision making processes. We use $\phi = 1.0$ for the IQN model during training. An ϵ -greedy policy is employed in training, where the exploration rate ϵ decays linearly from $\epsilon_{max} = 0.6$ to $\epsilon_{min} = 0.05$ in the first 25% of training time steps, and remains the same thereafter. In each step, the experience tuples (s, a, r, s') of all robots are inserted in to the replay buffer \mathcal{M} , which is then used to give a batch of samples \mathcal{B} for model training. If a robot reaches its goal or collides with static obstacles or other robots, it will be removed from the environment. The current episode ends when it is longer than $l_{episode_max} = 1000$ or no robot exists in the environment.

For every $t_{eval_freq} = 60,000$ steps, the learning model is evaluated in a set of predefined evaluation environments to understand its learning performance. We create ten random environments for each level of complexity in Table I, resulting in a total of sixty evaluation environments. We train thirty IQN models and thirty DQN models with different random

TABLE I: **Training schedule.** Hyperparameters used to create training environments.

Timesteps (million)	1st	2nd	3rd	4st	5st	6st
Number of robots	3	5	7	7	7	7
Number of vortices	4	6	8	8	8	8
Number of obstacles	0	0	2	4	6	8
Min distance between start and goal	30.0	35.0	40.0	40.0	40.0	40.0

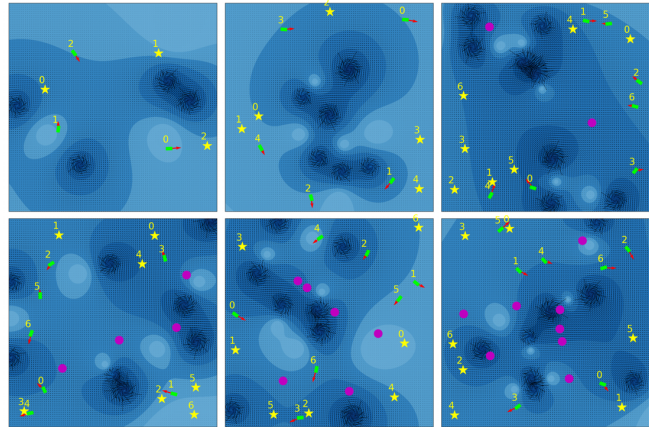


Fig. 3: **Training environments.** Examples of random environments generated according to the training schedule (environments of increasing difficulty are shown from left to right, top to bottom). The initial poses and velocities of the robots are indicated with green rectangles and red arrows.

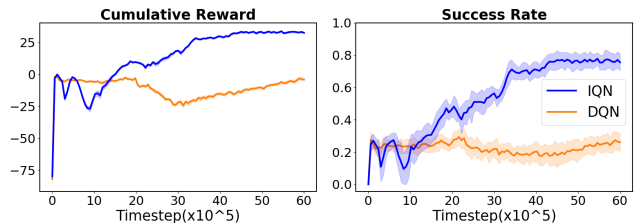


Fig. 4: **Evaluation performance during training.** Solid lines and bandwidths indicate the mean and standard error over the results of all learning models.

seeds on an Nvidia RTX 3090 GPU, and show their general performance in Fig. 4. If not all robots reach goals at the end, the evaluation episode is considered failed, hence IQN achieves a significantly higher level of safety even in highly congested environments.

V. EXPERIMENTS

We evaluate the performance of two randomly selected IQN and DQN models, as well as baseline approaches which will be introduced as follows, on two sets of simulation experiments. The first set of experiments purely focuses on dynamic obstacle avoidance and does not contain any static obstacles. 500 randomly generated environments of different levels of difficulty (100 per level) are used. The number of robots range from 3 to 7, and the number of vortices range from 4 to 8. The start and goal of each robot are randomly generated such that the distance is at least 40.0 meters. In the second set of experiments, static obstacles are also included when creating 500 evaluation environments, where the number of static obstacles ranges from 4 to 8. If

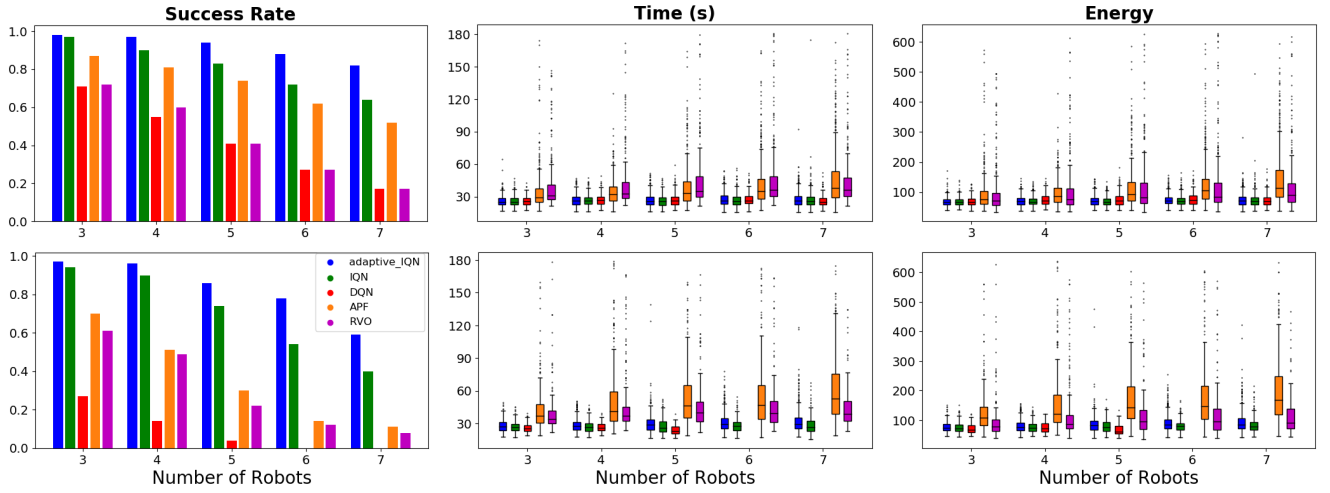


Fig. 5: **Experimental results.** Top and bottom rows show the results of experiments without and with static obstacles, respectively. Time and energy plots show the distributions of travel time and energy consumption of the robots in successful episodes.

not all robots successfully reach their goals within 3 minutes, the test episode is considered failed. The energy consumption of a robot in an experiment episode is computed by summing up the magnitude of all actions executed during the process.

In addition to the greedy policy, we also test the *risk sensitive policy* used in our prior work [14], which adapts the CVaR threshold ϕ according to the distance to obstacles.

$$\phi = \begin{cases} \min(d(X, X_O))/d_0, & \text{if } \min(d(X, X_O)) \leq d_0 \\ 1.0, & \text{if } \min(d(X, X_O)) > d_0 \end{cases} \quad (19)$$

X is the position of the ego robot, and X_O are the positions of all obstacles and other robots.

In addition to IQN and DQN, we also evaluate the performance of two classical methods. Fan et al. [17] designed an improved Artificial Potential Field (APF) method to deal with both static and dynamic obstacles. The attractive potential field U_{att} creates a force that drives the ego robot to the goal.

$$U_{\text{att}}(X) = \frac{1}{2} k_{\text{att}} \cdot d^2(X, X_g) \quad (20)$$

The repulsive potential field U_{rep1} creates a repulsive force to avoid colliding with static obstacles.

$$U_{\text{rep1}}(X) = \begin{cases} U_o(X), & \text{if } d(X, X_o) \leq d_0 \\ 0, & \text{if } d(X, X_o) > d_0 \end{cases} \quad (21)$$

$$\text{where } U_o(X) = \frac{1}{2} k_{\text{rep}} \left(\frac{1}{d(X, X_o)} - \frac{1}{d_0} \right)^2 d^n(X, X_g)$$

For dynamic obstacles, an additional repulsive field based on the relative velocity to the ego robot is used.

$$U_{\text{rep2}}(X, V) = \begin{cases} U_r(X) + U_r(V), & \text{if } d(X, X_r) \leq d_0 \\ 0, & \text{if } d(X, X_r) > d_0 \end{cases}$$

$$\text{where } U_r(X) = \frac{1}{2} k_{\text{rep}} \left(\frac{1}{d(X, X_r)} - \frac{1}{d_0} \right)^2 d^n(X, X_g), \\ U_r(V) = k_v \frac{1}{d(X, X_r)} v_{ao} \quad (22)$$

In the above equations, X , X_g , X_o , and X_r are positions of the robot, goal, static and dynamic obstacles respectively, and v_{ao} is the relative velocity component in the direction from the ego robot to the dynamic obstacle. We use $k_{\text{att}} =$

50.0, $k_{\text{rep}} = 500.0$, $k_v = 1.0$, and $n = 2$. The total force $F = -\nabla U_{\text{att}}(X) - \nabla U_{\text{rep1}}(X) - \nabla U_{\text{rep2}}(X, V)$. As in our implementation of APFs in previous work [14], we choose the angular velocity that results in the closest alignment with vector F in a single control step, and we choose the linear acceleration closest to the scaled component of F in the direction of robot velocity, as the control action.

Van den Berg et al. [19] proposed Reciprocal Velocity Obstacles (RVO). For disc-shaped robots A and B , the velocity obstacle $VO_B^A(\mathbf{v}_B)$ is the set of velocities of A , \mathbf{v}_A , that will lead to collision with B moving at velocity \mathbf{v}_B . Denote a ray starting at \mathbf{p} and pointing in the direction of \mathbf{v} as $\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + t\mathbf{v} | t \geq 0\}$, then $VO_B^A(\mathbf{v}_B)$ can be expressed in Equation (23), where \oplus is the Minkowski sum, and $-A$ refers to A reflected about its reference point.

$$VO_B^A(\mathbf{v}_B) = \{\mathbf{v}_A | \lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B) \cap B \oplus -A \neq \emptyset\} \quad (23)$$

To solve the problem of oscillatory motions when choosing the velocity that avoids the VO s created by other robots, RVO is introduced, which translates the apex of $VO_B^A(\mathbf{v}_B)$ to $(\mathbf{v}_A + \mathbf{v}_B)/2$. Since no passively moving obstacles exist in our problem, the combined reciprocal velocity obstacle for each robot is the union of all the RVO s generated by other robots and the VO s generated by static obstacles.

$$RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v}'_A | 2\mathbf{v}'_A - \mathbf{v}_A \in VO_B^A(\mathbf{v}_B)\}. \quad (24)$$

The velocity \mathbf{v}'_i that minimizes the penalty function (25) is chosen to be the next velocity of robot i .

$$\text{penalty}_i(\mathbf{v}'_i) = w_i \frac{1}{tc_i(\mathbf{v}'_i)} + \|\mathbf{v}'_i - \mathbf{v}_i^{\text{pref}}\| \quad (25)$$

$$\mathbf{v}'_i = \arg \min_{\mathbf{v}''_i \in AV^i} \text{penalty}_i(\mathbf{v}''_i) \quad (26)$$

We use $w_i = 0.2$ for all robots, and set $\mathbf{v}_i^{\text{pref}}$ as the velocity of maximum robot speed, with heading directed towards its goal. $tc_i(\mathbf{v}'_i)$ is the expected time to collision with the robot's combined reciprocal velocity obstacle, and AV^i is the set of admissible velocities. Similar to our application of APFs, we choose the angular velocity that results in

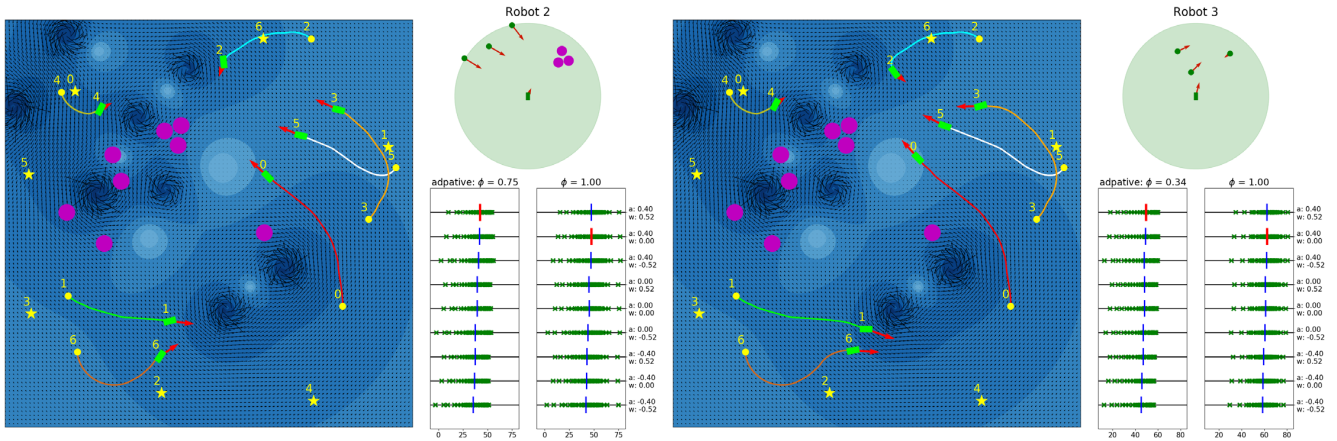


Fig. 6: **Performance of adaptive IQN agents in congested scenarios.** The resulting distributions of adaptive IQN and greedy IQN, in two representative decision-making instances from the same mission, are shown as return values corresponding to sampled quantiles. The mean of each distribution is marked as a vertical bar, and the selected actions are colored in red.

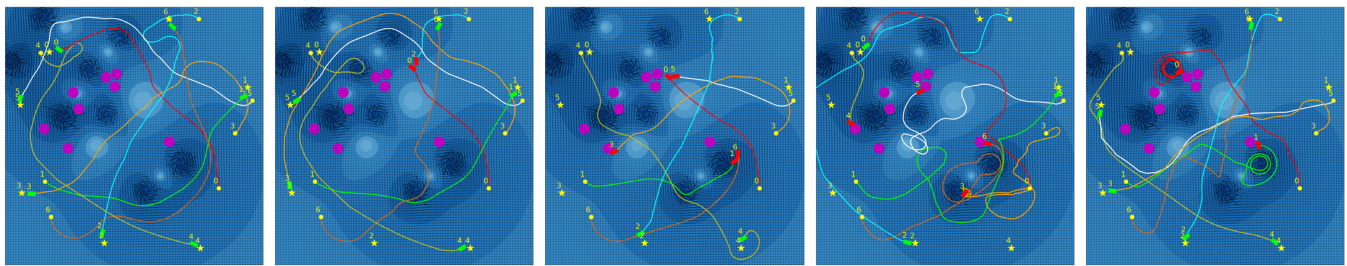


Fig. 7: **Planned trajectories.** Trajectories of adaptive IQN, greedy IQN, DQN, APF, and RVO are shown from left to right, for a representative set of start/goal locations. We mark robots that reach their goals in green, and those that failed in red.

the closest alignment with vector \mathbf{v}_i^j in a single control step, and we choose the linear acceleration leading to the speed closest to that of \mathbf{v}_i^j in a single control step, as the control action. The RVO agent of our work is based on the implementation in [41], which can be conveniently integrated into our simulation environment used in this study.

Our experimental results are summarized in Fig. 5. Adaptive IQN achieves the highest success rates across different levels of environment complexity, while consuming nearly the minimum amount of travel time and energy needed to reach goals. Fig. 6 and Fig. 7 visualize an experiment episode where seven robots, eight vortices and eight static obstacles exist. In the left plot of Fig. 6, robot 2 faces several static obstacles on the left and three robots coming from the right, while its goal is basically in the direction of its current speed. With a higher risk sensitivity, the adaptive IQN agent chooses to turn left to avoid getting into a more congested state with higher collision risk, while the greedy IQN agent would keep moving in the current direction towards the goal. Subsequently, after robot 2 turns left, robot 3 faces head-on collision risk with robot 2, and the adaptive IQN agent directs robot 3 to turn left in this instance to get away from the head-on situation. In comparison, Fig. 7 shows that greedy IQN failed in this episode due to the collision between robot 0 and robot 2. Thus, adaptive IQN can effectively enhance the safety in congested environments, which is shown as the increase in the difference of success rate between adaptive IQN and greedy IQN, while keeping the travel time and energy consumption small. It can be seen from Fig. 5 that

DQN requires the minimum travel time and energy cost in its successful episodes, and its planned trajectories are straight compared to that of other methods. This evidence indicates the greedy nature of DQN, the success rate of which drops significantly as the environment becomes more complex. APF and RVO are vulnerable to current disturbances of robot motions, which cause the ASVs to struggle near vortices, and to fall victim to collisions with static obstacles in Fig. 7.

VI. CONCLUSION

We propose a Distributional RL based decentralized multi-ASV collision avoidance policy, which is deployed in simulated congested marine environments filled with unknown static obstacles and vortical current flows (inspired by the challenges of whitewater rafting). The interactions among decentralized decision making agents as well as with static obstacles are considered in the policy network, which is shared by all agents during training to induce the coordinated avoidance of collisions, and robustness to current disturbances. Compared to traditional DRL, APF and RVO, the Distributional RL policy dominates in navigation safety, which can be further enhanced by adapting the risk sensitivity, while also achieving nearly the minimum amount of travel time and energy consumption.

ACKNOWLEDGMENT

This research was supported by the Office of Naval Research, grants N00014-20-1-2570 and N00014-21-1-2161.

REFERENCES

- [1] A. Vagale, R. Oucheikh, 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] International Maritime Organization, "Convention on the international regulations for preventing collisions at sea, 1972 (COLREGs)," 1972.
- [3] 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.
- [4] D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2402–2409.
- [5] L. Zhao and M.-I. Roh, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, p. 106436, 2019.
- [6] 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.
- [7] 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.
- [8] T. Lolla, P. Haley Jr, and P. Lermusiaux, "Path planning in multi-scale ocean flows: Coordination and dynamic obstacles," *Ocean Modelling*, vol. 94, pp. 46–66, 2015.
- [9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [10] Y. Cheng and W. Zhang, "Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels," *Neuro-computing*, vol. 272, pp. 63–73, 2018.
- [11] J. Woo and N. Kim, "Collision avoidance for an unmanned surface vehicle using deep reinforcement learning," *Ocean Engineering*, vol. 199, p. 107001, 2020.
- [12] 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.
- [13] 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.
- [14] X. Lin, J. McConnell, and B. Englot, "Robust unmanned surface vehicle navigation with distributional reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 6185–6191.
- [15] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [16] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18,382–18,390, 2017.
- [17] 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.
- [18] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.
- [19] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.
- [20] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009, pp. 177–187.
- [21] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium (ISRR)*. Springer, 2011, pp. 3–19.
- [22] J. Alonso-Mora, A. Breitenmoser, M. Ruffi, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Springer, 2013, pp. 203–216.
- [23] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.
- [24] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 797–803.
- [25] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2153–2160.
- [26] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.
- [27] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1343–1350.
- [28] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052–3059.
- [29] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.
- [30] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.
- [31] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10,007–10,013.
- [32] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671–5677.
- [33] R. Han, S. Chen, S. Wang, Z. Zhang, R. Gao, Q. Hao, and J. Pan, "Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5896–5903, 2022.
- [34] J. Choi, C. Dance, J.-E. Kim, S. Hwang, and K.-S. Park, "Risk-conditioned distributional soft actor-critic for risk-sensitive navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8337–8344.
- [35] D. Kamran, T. Engelgeh, M. Busch, J. Fischer, and C. Stiller, "Minimizing safety interference for safe and comfortable automated driving with distributional reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1236–1243.
- [36] C. Liu, E.-J. van Kampen, and G. C. de Croon, "Adaptive risk-tendency: Nano drone navigation in cluttered environments with distributional reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7198–7204.
- [37] X. Lin, P. Szenher, J. D. Martin, and B. Englot, "Robust route planning with distributional reinforcement learning in a stochastic road network environment," in *20th International Conference on Ubiquitous Robots (UR)*, 2023, pp. 287–294.
- [38] D. J. Acheson, *Elementary Fluid Dynamics*. Acoustical Society of America, 1991.
- [39] T. Lolla, P. F. Lermusiaux, M. P. Uecker mann, and P. J. Haley, "Time-optimal path planning in dynamic flows using level set equations: theory and schemes," *Ocean Dynamics*, vol. 64, pp. 1373–1397, 2014.
- [40] 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.
- [41] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1082–1097, 2018.