2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SwarmNav: Swarm Robotics Navigation in Dynamic and Dense Environments via Reinforcement Learning

Shengbo Li[1], Chuanjie Lv[1], Xiangqian Yuan[1], Liming Xu[1], Xinyang Liu[1], Zongzhi Zhu[2], Gang Xu[1,*] and Yong Liu[1,*]

***Abstract*— Collision avoidance and navigation in dynamic and dense environments remain highly challenging for swarm robotics. To address this, we propose SwarmNav, a novel goal-region amplification navigation policy that leverages LiDAR-based position data to generate velocity commands guiding robots toward their goals while actively avoiding obstacles. SwarmNav is trained within a deep reinforcement learning actor-critic framework. In this framework, the reward function integrates a goal-region amplification term with the reciprocal velocity obstacles formulation, enabling goal-directed navigation under dynamic obstacle uncertainty. Extensive simulations demonstrate that SwarmNav significantly outperforms state-of-the-art approaches, including both reinforcement learning-based and traditional velocity obstacle-based methods, in terms of success rate and computational efficiency. Real-world experiments across diverse scenarios further confirm its effectiveness in dynamic and dense environments.**

I. INTRODUCTION

Swarm robotics has increasingly been applied across a wide range of domains, including search and rescue [1], [2], agricultural automation [3], [4], warehouse management [5], [6], and environmental monitoring [7], [8]. However, enabling effective swarm navigation in dynamic and dense environments remains highly challenging, due to the need to perceive complex surroundings with unknown and moving obstacles, extract meaningful information from sensor data, and adapt navigation policies to diverse scenarios.

To address these challenges, considerable research has focused on developing methods for collision avoidance and navigation in swarm robotics. Among them, velocity obstacle (VO)-based methods [9] represent typical early approaches in dynamic environments. However, these methods are limited by computational efficiency, reliance on cooperative dynamic agents, and susceptibility to deadlock in symmetric situations, leading to inefficient navigation in dynamic environments with motion uncertainty. Several extensions have been proposed to address these limitations, such as the variable responsibility strategy introduced by Guo et al. [10] to improve collision avoidance efficiency and the diversion strategy proposed by Xu et al. [11] to resolve deadlocks in symmetric scenarios. On the other hand, some reinforcement learning-based methods have been investigated to tackle navigation challenges in swarm robotics under dynamic and dense environments. For example, Han et al. [12] proposed a deep reinforcement learning-based method, which incorporates the reciprocal velocity obstacles (RVO) formulation into the reward function to ensure safe navigation in swarm robotics. Building on this work, Chen et al. [13] expanded the RVO representation by developing a novel spatio-temporal network comprising a temporal state encoder for sequence features and a reciprocal spatial state encoder. However, the aforementioned methods still struggle to ensure safe navigation in dynamic dense obstacle environments. To this end, Huang et al. [14] addressed multi-robot navigation in environments with dense dynamic obstacles by incorporating the VO formulation into the reward function of a deep reinforcement learning framework. Similarly, Martinez et al. [15] employed the VO formulation by leveraging nonlinear opinion dynamics to adapt to varying levels of agent cooperation, thereby enhancing the safety of swarm robotics navigation in dynamic and dense environments.

Nevertheless, in dynamic and dense environments, reinforcement learning-based methods still exhibit poor generalization, hindering the deployment of a single model to larger swarms and causing the safety of swarm navigation to degrade rapidly with increasing swarm size. To address these limitations, we propose SwarmNav, a novel navigation framework for dynamic collision avoidance in swarm robotics. The overall training framework of SwarmNav is illustrated in Fig. 1. Inspired by [12] and built upon the RVO formulation, SwarmNav introduces a goal-region amplification strategy to optimize the design of reinforcement learning reward functions. In addition, it leverages LiDAR data to obtain observations of each robot's surrounding environment. This proposed method not only guides robots effectively toward their target regions but also enables them to proactively avoid other robots and dynamic obstacles with motion uncertainty, thereby achieving safe and efficient navigation for every robot in the swarm.

The main contributions are summarized as follows:

1) A reinforcement learning framework with a hierarchical training strategy, where robots first acquire basic collision avoidance skills in obstacle-free environments and then progressively adapt to dynamic obstacles in complex scenarios.
2) A novel reward function design that integrates the reciprocal velocity obstacles (RVO) formulation with a novel goal-region amplification term.
3) Extensive simulations and real-world experiments with

*Gang Xu and Yong Liu are the corresponding authors (e-mail: wuuya@zju.edu.cn, yongliu@iipc.zju.edu.cn).
[1] are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China.
[2] is with the Zhejiang Guoli Xin'an Technology Co., Ltd., Hangzhou 310027, China.
This work was supported by the the Key R&D Program of Zhejiang Province (2025C01069).

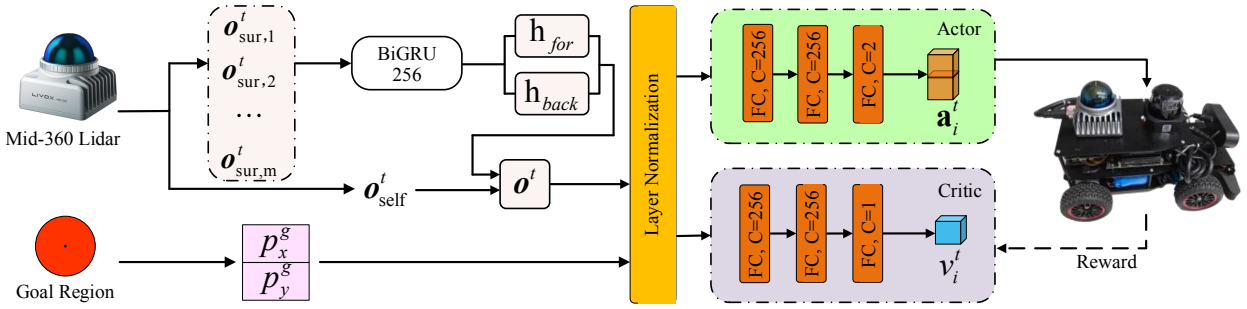


Fig. 1. Overall system architecture of the proposed SwarmNav navigation policy. Raw LiDAR observations and the goal-region amplification input are normalized before being fed into the actor and critic networks, where the critic estimates the state value and the actor outputs the velocity commands.

multiple robots demonstrate that SwarmNav significantly outperforms state-of-the-art VO-based and reinforcement learning-based methods in terms of success rate, efficiency, and robustness.

II. Related Works

Numerous approaches have been proposed for collision avoidance and navigation in swarm robotics, and most of them can be broadly categorized into centralized and decentralized approaches.

Among them, centralized methods typically rely on a central server that collects the full state information of all robots and obstacles to generate motion solutions for each robot. Representative approaches include CBS [16], which obtains feasible paths through a combination of low-level graph search and high-level conflict resolution. Another example is SIPP [17], which generates initial paths for robots via graph search and resolves conflicts based on assigned priorities. Several extensions, such as CL-CBS [18] and D-PBS [19], further improve these approaches by considering kinematic constraints. However, as the number of robots increases, the computational complexity of centralized methods grows rapidly, making it difficult to deploy them in real-world scenarios.

In contrast to centralized methods, decentralized approaches allow each robot to independently generate feasible motion commands based on its observations of the surrounding environment. In decentralized methods, VO-based approaches [9] have attracted the most attention, where the concept was first introduced in [20]. Subsequently, the RVO method [21] was proposed to address the motion oscillations of the VO method by introducing reciprocal responsibility among neighboring robots in collision avoidance. Furthermore, the optimal reciprocal collision avoidance (ORCA) method [22] was proposed to improve computational efficiency by representing the space of collision-free velocities as a set of half-planes and determining feasible velocities through linear programming. Several studies [10], [11] further mitigate deadlocks in symmetric scenarios by building upon the RVO formulation, which in turn improves the success rate of swarm robotics navigation. Nevertheless, VO-based methods typically assume perfect knowledge of surrounding robots and obstacles. This assumption becomes unrealistic in real-world environments with dynamic obstacles and motion uncertainty, thereby limiting their effectiveness in ensuring safe navigation.

Meanwhile, reinforcement learning-based methods, driven by recent advances in artificial intelligence, have emerged as a promising framework for navigation in unknown or highly dynamic environments. For instance, [23] proposed an attention-based deep reinforcement learning (DRL) approach that enables a single robot to navigate through crowded scenarios by extracting features of surrounding pedestrians to predict their movements and avoid collisions. [24] employed inverse reinforcement learning (IRL) to enable robots to learn from human interactions and adapt to user preferences, thereby reducing the need for manually programmed behaviors. [25] combined deep reinforcement learning with imitation learning to separately handle information from static and dynamic objects, thereby enabling more precise motion planning in complex environments. [26] utilized the VO formulation to design a reward function within a deep reinforcement learning framework, demonstrating strong collision-avoidance performance in dense dynamic environments. However, these methods mainly focus on the navigation of a single robot in scenarios with dynamic obstacles. Fortunately, many studies, such as [12], [13], [27], [28], have already focused on the problem of collision avoidance in robot swarms. In [27], the authors proposed a distributed motion planning framework in which each robot estimates its relative position and orientation based on range measurements and relative velocities, achieving robust collision avoidance in large-scale environments. Considering uncertainties in multi-robot motion arising from unobservable intentions of other agents, the work [28] proposed a decentralized collision avoidance approach that leverages a value network to estimate the time to the goal from the joint positions and velocities of an agent and its neighbors. In addition, an RL-RVO method based on deep reinforcement learning was proposed in [12], which integrates the RVO formulation into the reward function to ensure collision avoidance among robots. Building on this work, Chen et al. [13] developed a spatio-temporal network that extends the RVO formulation. The network consists of a temporal state encoder for sequence features and a reciprocal spatial state encoder that leverages a transformer to integrate information from Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional GRU (BiGRU) branches.

In contrast to existing methods, the SwarmNav approach proposed in this paper explicitly considers both interactions among multiple robots and dynamic obstacles with uncertain motions. By introducing a goal-region amplification strategy into the reward function, SwarmNav improves the success rate of individual robot navigation in densely dynamic obstacle scenarios, enhances generalization across diverse scenarios and varying swarm sizes, and increases computational efficiency.

III. Problem Statement

The navigation and collision avoidance problem in swarm robotics can be formulated as an optimization problem that seeks to determine a sequence of optimal velocities for each robot, aiming to minimize travel time while satisfying collision avoidance constraints. For a fleet of robots and dynamic obstacles in the same environment, consider an individual robot and its m neighbors (including other robots and dynamic obstacles). Let $\mathbf{p}^t$ and $\mathbf{v}^t$ denote the position and velocity of the robot at time t, respectively, and let $\mathbf{p}_j^t$ and $\mathbf{v}_j^t$ denote the position and velocity of its j-th neighbor. Given the robot's proprioceptive observation $\mathbf{o}_{\text{self}}^t$ and the exteroceptive observation of its surroundings $\mathbf{o}_{\text{sur}}^t$, the policy neural network π_θ outputs the action $\mathbf{a}^{t+1}$ for the next time step. This action $\mathbf{a}^{t+1}$ aims to minimize the difference between the robot's actual velocity and its desired velocity at time $t+1$, while satisfying collision avoidance constraints. Thus, the navigation and collision avoidance problem of an individual robot can be formulated as the following constrained optimization problem:

$$\begin{aligned} \arg\min_{\pi_\theta} \ & \left\| \mathbf{v}^{t+1} - \mathbf{v}_{\text{des}}^{t+1} \right\|, \\ \text{s.t.} \quad & \Delta\mathbf{v}^{t+1} \sim \pi_\theta(\mathbf{a}^{t+1} \mid \mathbf{o}_{\text{self}}^t, \mathbf{o}_{\text{sur}}^t), \\ & \|\mathbf{v}^{t+1}\| \le v_{\max}, \\ & \|\mathbf{p}^{t+1} - \mathbf{p}_j^{t+1}\| > R_c + R_c^j, \\ & \forall j \in [1, m], \end{aligned} \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector, and R_c and R_c^j are the collision radii of the robot and its j-th neighbor, respectively. All the robots share the same navigation policy π_θ and find the optimal velocity independently. In addition, all dynamic obstacles are assumed to move randomly, and their motions are uncertain to the robots.

IV. Methodology

In this section, we first summarize the RVO formulation, then formulate the observation and action spaces of our proposed method. Subsequently, we present the DRL network architecture, the goal-region amplification strategy, and the design of the novel reward function.

A. Reciprocal Velocity Obstacle

The reciprocal velocity obstacle (RVO) [21] defines an area of relative velocities between two robots that would lead to a collision if maintained. Selecting velocities outside this area avoids collisions. Based on this principle, we incorporate the RVO formulation as a component in the reward function design. As shown in Fig. 2, for disc-shaped robots A and B with radii R_A and R_B, their positions and velocities are denoted by $\mathbf{p}_A$, $\mathbf{p}_B$, $\mathbf{v}_A$, and $\mathbf{v}_B$, respectively. Thus, the velocity obstacle area $VO_B^A(\mathbf{v}_B)$ is defined as:

$$VO_B^A(\mathbf{v}_B) = \{\mathbf{v}_A \,|\, \lambda(\mathbf{p}_A, \mathbf{v}_{AB}) \cap (B \oplus -A) \neq \emptyset\}, \tag{2}$$

where the relative velocity $\mathbf{v}_{AB} = \mathbf{v}_A - \mathbf{v}_B$, $\lambda(\mathbf{p}_A, \mathbf{v}_{AB})$ denotes the ray starting at $\mathbf{p}_A$ in the direction of $\mathbf{v}_{AB}$, and $\oplus$ denotes the Minkowski sum. Therefore, robot A avoids collision by selecting a velocity outside the $VO_B^A(\mathbf{v}_B)$ area. Furthermore, the reciprocal velocity obstacle area $RVO_B^A(\mathbf{v}_B, \mathbf{v}_A)$ can be obtained from $VO_B^A(\mathbf{v}_B)$ by translating its apex from $\mathbf{v}_B$ to $\mathbf{v}_p = \frac{\mathbf{v}_A + \mathbf{v}_B}{2}$, which is formulated as:

$$RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v}_A' \,|\, 2\mathbf{v}_A' - \mathbf{v}_A \in VO_B^A(\mathbf{v}_B)\}. \tag{3}$$

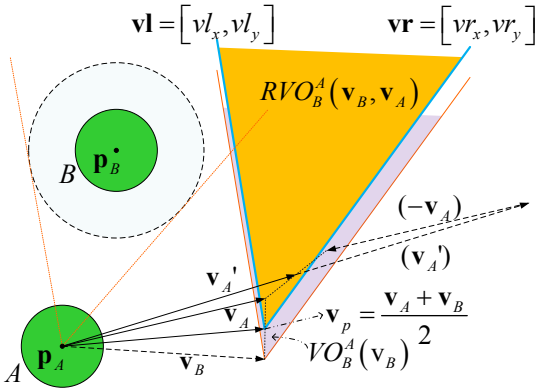


Fig. 2. The reciprocal velocity obstacle $RVO_B^A(\mathbf{v}_B, \mathbf{v}_A)$ of robot B to robot A.

Particularly, the RVO can be represented by the vector $\mathbf{c} = [\mathbf{v}_p, \mathbf{vl}, \mathbf{vr}]$, where $\mathbf{v}_p = [v_x, v_y]$ is the apex coordinate, and $\mathbf{vl} = [vl_x, vl_y]$ and $\mathbf{vr} = [vr_x, vr_y]$ represent the directions of the left and right rays, respectively. Thus, the vector $\mathbf{c}$ is subsequently incorporated as a component of the observation space in our reinforcement learning framework, representing the reciprocal velocity obstacle area between the robot and its neighboring robots or obstacles.

B. Observation Space and Action Space

The observation space of each agent at time t consists of its own state and the observations of the m nearest surrounding robots or obstacles, denoted as $\mathbf{o}^t = [\mathbf{o}_{\text{self}}^t, \mathbf{o}_{\text{sur},j}^t]$ with $j = 1, \ldots, m$. Specifically, the vectors $\mathbf{o}_{\text{self}}^t$ and $\mathbf{o}_{\text{sur},j}^t$ are defined as follows:

$$\begin{cases} \mathbf{o}_{\text{self}}^t = \left[\mathbf{v}^t, \theta^t, \mathbf{v}_{\text{des}}^t, R_c\right], \\ \mathbf{o}_{\text{sur},j}^t = \left[\mathbf{c}_j^t, d_j^t, r_j^t\right]. \end{cases} \tag{4}$$

Here, $\mathbf{o}_{\text{self}}^t$ represents the state of the robot itself, including its current velocity $\mathbf{v}^t$, orientation θ^t, desired velocity $\mathbf{v}_{\text{des}}^t$, and virtual radius R_c for collision avoidance. Meanwhile, $\mathbf{o}_{\text{sur},j}^t$ denotes the observation of the j-th closest neighbor, including the reciprocal vector $\mathbf{c}_j^t$ derived from the RVO area, the relative distance d_j^t between the robot and its j-th neighbor, and the collision-risk indicator r_j^t, defined as follows:

$$r_j^t = \frac{1}{t_j^t + D}, \tag{5}$$

where D is a constant value and t_j^t is the predicted time-to-collision with the j-th neighbor. Note that d_j^t and r_j^t are estimated from LiDAR measurements. Since sensor accuracy diminishes with distance, the LiDAR detection range is limited to $r_{\max}$ meters. The robot selects the m closest robots or obstacles within this range, padding with zeros if fewer than m objects are detected to maintain a consistent observation-space dimension.

The action space is defined as the change in the robot's velocity within the plane, *i.e.*, $\mathbf{a}^t = [\Delta v_x^t, \Delta v_y^t]$, representing the velocity increments in the x and y directions. In addition, the robot's velocity is constrained within the range $[v_{\min}, v_{\max}]$. The relationship between the velocities at times t and $t+1$ is given by:

$$\mathbf{v}^{t+1} = \mathbf{v}^t + \mu \cdot \mathbf{a}^t, \tag{6}$$

where μ is a parameter controlling the range of the velocity increment.

C. Neural Network Architecture

Fig. 1 outlines our network architecture. We use a BiGRU module, consisting of two GRUs that process the inputs in both forward and backward directions. At each step, the exteroceptive measurements $\mathbf{o}_{\text{sur},j}^t$ (a series of vectors) are first sorted in ascending order of r_j^t and then in descending order of the distance d_j^t, and are fed into the BiGRU. Two final hidden states, h_{for} and h_{back}, are obtained from the forward and backward GRUs, respectively. This output is concatenated with the proprioceptive measurement $\mathbf{o}_{\text{self}}^t$ to form the integrated fixed-length observation $\mathbf{o}^t$. We then apply layer normalization [29] to this fixed-length observation. The normalized output serves as the input to two neural networks. The first is the policy actor π_θ, which consists of two hidden fully connected (FC) layers using Tanh as the activation function. The final layer outputs a two-dimensional vector representing the velocity increment. The second network corresponds to the policy critic V_ψ, which also consists of two hidden fully connected layers using ReLU as the activation function. Its final layer outputs a scalar value that evaluates each state-action pair.

D. Goal-Region Amplification

To enable the robots to actively avoid dynamic obstacles in the environment while moving toward their goal positions, we propose a goal-region amplification strategy to improve the reward function. As illustrated in Fig. 3, we define the red circular region, centered at the goal position $\mathbf{p}_{goal} = [p_x^g, p_y^g]$ with radius R_{goal}, as the goal region. The goal region is then expanded into a larger virtual goal region with a radius of M_{goal}, given by

$$M_{goal} = \mathscr{K} \cdot R_{goal}, \tag{7}$$

where $\mathscr{K}$ is an amplification coefficient. Next, we describe how the amplified virtual goal region is used to generate a triangular reward region $A_{\text{re}} = [\mathbf{p}_A, \mathbf{p}_1, \mathbf{p}_2]$ that guides the robots toward their goal positions. Here, $\mathbf{p}_1$ and $\mathbf{p}_2$ are auxiliary vertices defined with respect to the current robot position $\mathbf{p}_A = [p_x^A, p_y^A]$. Specifically, if $|p_x^A - p_x^g| \le M_{goal}$ and $p_x^A \neq p_x^g$, then

$$\mathbf{p}_1 = [p_x^g,\, p_y^g + M_{goal}], \quad \mathbf{p}_2 = [p_x^g,\, p_y^g - M_{goal}]. \tag{8}$$

Similarly, if $|p_y^A - p_y^g| \le M_{goal}$ and $p_y^A \neq p_y^g$, then

$$\mathbf{p}_1 = [p_x^g + M_{goal},\, p_y^g], \quad \mathbf{p}_2 = [p_x^g - M_{goal},\, p_y^g]. \tag{9}$$

As illustrated in Fig. 3(a), when $|p_x^A - p_x^g| \le M_{goal}$, the green region corresponds to A_{re}. In this case, if the robot's current velocity direction lies within A_{re}, the angles θ_1 and θ_2 are computed from $\mathbf{p}_A$, $\mathbf{p}_1$, $\mathbf{p}_2$, and the X-axis, and are subsequently used to determine the reward value. The computation under the condition $|p_y^A - p_y^g| \le M_{goal}$ is carried out in a similar manner.

Furthermore, when the dynamic obstacle enters the reward region A_{re}, a purple penalty region A_{pen} is generated, as shown in Fig. 3(b). Specifically, A_{pen} is defined by the tangent lines from the robot's current position to the dynamic obstacle. The robot's orientation θ^t is then evaluated with respect to A_{re} and A_{pen} to determine the corresponding reward value. Whether θ^t lies within A_{re} or A_{pen} is determined by the parameter $\mathscr{K}$. As $\mathscr{K}$ increases, the reward region expands, allowing the robot to receive positive rewards earlier. However, overly large values of $\mathscr{K}$ may lead to unstable feedback due to uncertain obstacle motions, which in turn can affect the accuracy and stability of the robot's decision-making. Therefore, we empirically set $\mathscr{K} = 4$ in our implementation to balance responsiveness and stability.

E. Reward Function

An essential challenge in reinforcement learning methods is the design of an effective reward function that can guide the agent to acquire the desired behaviors. In this work, we propose a novel reward function to encourage the robot to move safely and efficiently toward its goal while satisfying the given velocity constraints. The proposed reward function is composed of the following terms.

1) RVO Reward: The reward is given by

$$r_{rvo}^t = \begin{cases} a_1 - b \cdot \mathrm{dis}_{\text{des}}^t & \text{if } \mathbf{v}_t \notin \text{RVO or } \xi > 5, \\ a_2 - c_1 \cdot (\xi + f)^{-1} & \text{if } \mathbf{v}_t \in \text{RVO and } \xi > 0.1, \\ -c_2 \cdot (\xi + f)^{-1} & \text{if } \xi \le 0.1, \end{cases} \tag{10}$$

where $\mathrm{dis}_{\text{des}}^t = \left\| \mathbf{v}^t - \mathbf{v}_{\text{des}}^t \right\|$ denotes the deviation between the robot's actual velocity $\mathbf{v}^t$ at time t and the desired velocity $\mathbf{v}_{\text{des}}^t$. The desired velocity $\mathbf{v}_{\text{des}}^t$ is defined as $[v_{\max} \cdot \cos(\alpha^t), v_{\max} \cdot \sin(\alpha^t)]$ when no collision occurs, and as $[0,0]$ otherwise. α^t is the angle between the vector $\mathbf{p}_{goal} - \mathbf{p}^t$ and the positive X-axis. ξ is the anticipated shortest time until a collision with another robot or an obstacle under the current velocity. Parameters a_1, a_2, b, c_1, c_2, and f are adjustable constants during training: a_1 and a_2 are baseline rewards that encourage desirable behaviors (typically positive values), b penalizes deviation from the desired velocity, while c_1 and c_2 weight the term $(\xi + f)^{-1}$. Whether $\mathbf{v}^t$ lies within the RVO

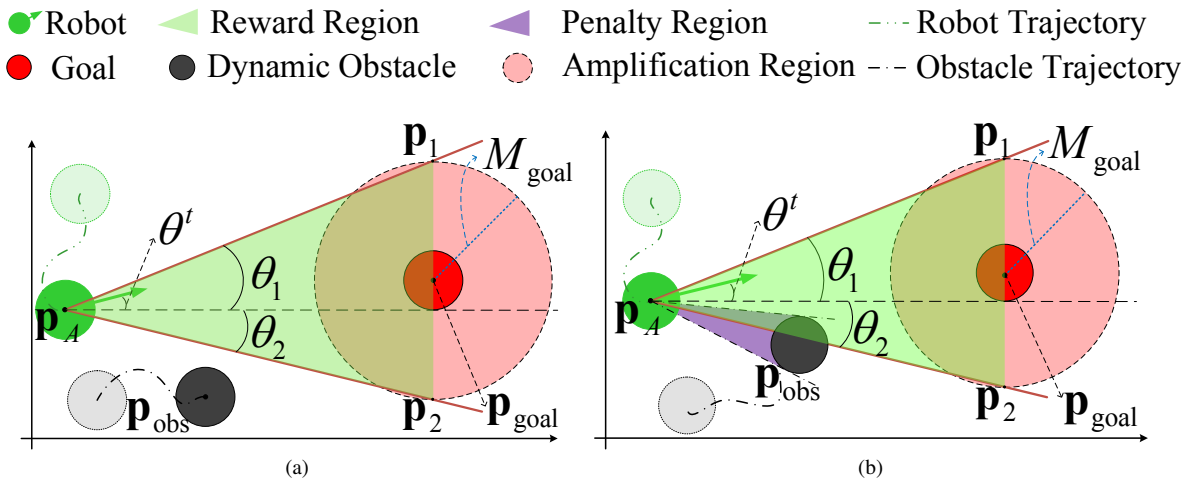


Fig. 3. Geometric illustration of the proposed goal-region amplification in the reward function. (a) Scenario without any obstacles in the reward region. (b) Scenario with obstacles entering the reward region, resulting in the generation of a penalty region.

area is evaluated using three vectors, $\mathbf{v}_p$, $\mathbf{vl}$, and $\mathbf{vr}$, with the following expression:

$$\begin{cases} \mathbf{v}^t \in \text{RVO} := (\mathbf{v}^t - \mathbf{v}_p) \times \mathbf{vl} \geq 0 \wedge (\mathbf{v}^t - \mathbf{v}_p) \times \mathbf{vr} \leq 0, \\ \mathbf{v}^t \notin \text{RVO} := (\mathbf{v}^t - \mathbf{v}_p) \times \mathbf{vl} < 0 \vee (\mathbf{v}^t - \mathbf{v}_p) \times \mathbf{vr} > 0, \end{cases} \tag{11}$$

where $\wedge$ and $\vee$ denote logical AND and OR, and $\times$ denotes the vector product.

Taken together, r^t_{rvo} drives the policy to prefer admissible velocities that remain outside the RVO region whenever possible, to track the desired velocity $\mathbf{v}^t_{\text{des}}$ in collision-free situations, and to promptly decelerate or reorient as the time-to-collision ξ decreases. By coupling the distance-to-desired-velocity term with an inverse time-to-collision penalty, this reward provides a proactive safety margin around nearby agents and obstacles, stabilizes behavior near the RVO boundary, and reduces oscillations and near-collision dithering.

2) Advance Planning: The reward is given by

$$r^t_{ap} = \begin{cases} h \cdot |\theta_{\max} - \theta^t| & \text{if } \theta^t \in A_{\text{re}} \text{ and } \theta^t \notin A_{\text{pen}}, \\ -h \cdot |\theta_{\max} - \theta^t| & \text{if } \theta^t \in A_{\text{pen}}, \\ 0 & \text{otherwise}, \end{cases} \tag{12}$$

where h is a constant coefficient, and $\theta_{\max} = max(|\theta_1|, |\theta_2|)$. Note that θ_1 and θ_2 are obtained from our goal-region amplification strategy. r^t_{ap} serves as the reward for encouraging the robot to avoid obstacles in advance, thereby improving the smoothness of the trajectory.

3) Reaching the Goal: The reward is given by

$$r^t_g = \begin{cases} g & \text{if reach the goal}, \\ 0 & \text{otherwise}, \end{cases} \tag{13}$$

where g is a constant positive reward that encourages robots to reach the goal region.

4) Passive Collision Avoidance: The reward is given by

$$r^t_c = \begin{cases} -g & \text{if collide with robots or obstacles}, \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

where r^t_c is the penalty for collisions with other robots or obstacles, which discourages unsafe behaviors. Here, g is the same constant as defined in Section IV-E.3, but used with a negative sign to penalize collisions.

5) Overall Reward: Integrating the four reward components described above, the overall reward function is defined as:

$$r^t_{\text{overall}} = r^t_{rvo} + r^t_{ap} + r^t_g + r^t_c. \tag{15}$$

F. Training Process

In this paper, we utilize PPO [30] to iteratively train and update the collision avoidance policy. This algorithm consists of an actor network π_θ with learning rate l_a and a critic network V_ψ with learning rate l_v. Each robot makes decisions based on its local observations to determine the next action. Since all robots share identical capabilities, we employ the same actor and critic networks for all agents, thereby reducing the overall training cost.

We adopt a staged training strategy [31]. In the first stage (e_1 epochs), training is conducted in a circular environment of size 10 m $\times$ 10 m containing n_1 robots, without additional obstacles; the robots are treated as moving obstacles to each other. Only the rewards r^t_{rvo}, r^t_g, and r^t_c are applied to ensure that robots learn to avoid collisions and establish connections to the goal region. In the second stage (e_2 epochs), n_3 dynamic obstacles are introduced, and the policy continues to be trained with n_2 robots in the same environment. The goal region is amplified, and the reward r^t_{ap} is incorporated to guide the robots in predicting obstacle motion and anticipating potential factors that may affect navigation. This design improves both the collision avoidance success rate and the quality of the resulting trajectories.

V. Experiment Results

In this section, we compare the proposed method with several state-of-the-art approaches in simulations across different scenarios and swarm sizes. Furthermore, we also conduct real-world experiments to validate the effectiveness and generalizability of the proposed method.

A. Training Details and Setup

We use a fleet of robots, each with a radius of 0.2 m, to train collision-avoidance strategies in a 10 m $\times$ 10 m envi-

ronment, with a maximum robot speed of 1.5 m/s. Notably, each robot independently performs localization and navigation while operating without any communication with other robots. Furthermore, the radius of dynamic obstacles is set to 0.1 m, with a maximum speed of 0.3 m/s. The robots' navigation policy is trained in simulation and implemented using PyTorch (Python 3.8). Specifically, in a circular scenario, each robot is uniformly placed on a circle, and its goal position is set directly opposite its starting point. In addition, the hyperparameters used during training are listed in Table I; these were selected based on preliminary tests and optimized to ensure stable and efficient training. Training is performed on an NVIDIA RTX 3090 GPU and requires approximately 8 hours to complete. Moreover, the performance of all methods is evaluated using the following four metrics:

- *Success Rate*: The fraction of trials (out of 100 independent trials under the same conditions) in which all robots successfully reach their goal positions.
- *Average Speed*: The average speed of all robots over the successful trials in 100 independent experiments.
- *Average Travel Distance*: The average distance traveled by all robots over the successful trials in 100 independent experiments.
- *Average Computation Time*: The average computation time per iteration for generating the next-step velocity commands in 100 independent experiments.

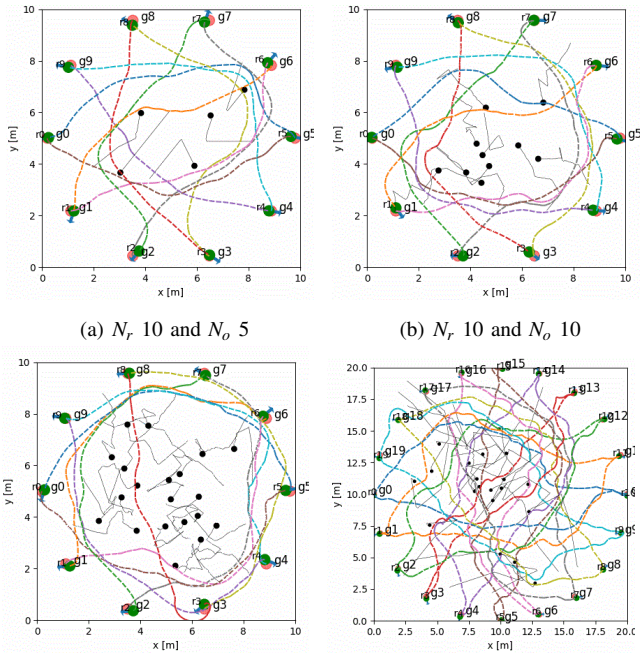
TABLE I
THE HYPERPARAMETERS OF THE TRAINING PROCESS

Para.	**Val.**	**Para.**	**Val.**	**Para.**	**Val.**	**Para.**	**Val.**
a_1	0.3	b	1.0	a_2	0.3	c_1	1.2
c_2	3.0	f	0.2	g	6.0	h	0.9
D	0.2	l_a	2e-7	e_1	1000	$\mathscr{K}$	4
R_c	0.3	l_v	5e-6	e_2	2000	μ	1.0
R_{goal}	0.2	$r_{\max}$	4	$v_{\min}$	-1.5	$v_{\max}$	1.5
m	5	n_1	4	n_2	10	n_3	5

These settings form the basis for the extensive experiments presented in the next subsection, which are used to evaluate the effectiveness of the proposed method in both simulation (including comparisons with state-of-the-art approaches) and real-world environments.

B. Simulation Results

In the simulations, we first compare the proposed method with four baseline approaches—RL-RVO [12], PCA [11], ORCA [22], and RVO [21]—across different circular scenarios and swarm sizes to evaluate its performance. Note that all dynamic obstacles move randomly in an omnidirectional manner. For each condition, 100 independent trials are conducted to collect statistics for the evaluation metrics. The results are reported in Table II, where N_r and N_o denote the number of robots and obstacles, respectively. Moreover, Fig. 4 shows the trajectories of all robots for four representative circular scenarios generated by our method. As shown, our method enables all robots to successfully reach their goal positions across different swarm sizes and dense obstacle environments.



(a) N_r 10 and N_o 5 (b) N_r 10 and N_o 10

(c) N_r 10 and N_o 20 (d) N_r 20 and N_o 20

Fig. 4. Trajectories generated by our SwarmNav in four representative scenarios, where all robots successfully reach their goal positions.

From Table II, it can be observed that, compared with the other four baseline methods, our SwarmNav achieves the highest success rate in almost all scenarios. In contrast, the success rates of the four baseline methods vary across different scenarios. The only exception occurs in the absence of dynamic obstacles, where symmetry favors PCA's diversion strategy; even in this case, SwarmNav still achieves the second-highest success rate and maintains consistently strong performance overall. It should also be noted that the success rates of ORCA and RVO drop to zero in the absence of dynamic obstacles due to deadlocks occurring under symmetrical conditions. In terms of average speed and average travel distance, PCA and RL-RVO achieve the best performance in almost all scenarios. This is because these methods do not account for the uncertain motion of dynamic obstacles and thus adopt more aggressive velocities. In contrast, our SwarmNav considers these uncertainties and therefore tends to adopt slightly more cautious speeds toward the goal positions. By comparison, ensuring that all robots safely reach their respective goal positions is the most important metric in swarm robotics navigation. Regarding the average computation time, Table II shows that our proposed method achieves the shortest value in this metric, except for one case, and is followed by RL-RVO. This indicates that reinforcement learning-based approaches significantly outperform traditional methods such as PCA and RVO in computation time for collision avoidance and navigation in robot swarms. In summary, under circular scenarios with dynamic obstacles exhibiting motion uncertainty, the proposed SwarmNav consistently achieves superior performance compared with the four baseline methods, particularly in terms of success rate and computational efficiency. These results demonstrate the effectiveness and robustness of the

TABLE II

COMPARISON OF SEVERAL METRICS OVER DIFFERENT EXPERIMENT SETTINGS.

Environments		Success Rate					Average Speed (m/s)					Average Travel Distance (m)					Average Computation Time (ms)				
Map (m^2)	N_r / N_o	RL-RVO	PCA	ORCA	RVO	Ours	RL-RVO	PCA	ORCA	RVO	Ours	RL-RVO	PCA	ORCA	RVO	Ours	RL-RVO	PCA	ORCA	RVO	Ours
5 × 5	5 / 0	1.00	1.00	0.00	1.00	**1.00**	1.04	**1.26**	-	1.19	1.05	5.62	5.57	-	**5.46**	5.82	0.09	14.34	-	13.92	**0.07**
5 × 5	10 / 0	0.85	**1.00**	0.00	0.00	0.99	0.84	**1.17**	-	-	0.73	6.23	**5.84**	-	-	7.03	0.17	20.34	-	-	**0.17**
5 × 5	10 / 2	0.85	0.63	0.83	0.76	**0.96**	0.84	**1.00**	0.36	0.98	0.78	**6.43**	9.81	7.26	8.56	7.43	**0.17**	23.36	0.62	22.66	0.19
10 × 10	10 / 0	1.00	1.00	0.00	0.00	**1.00**	1.01	**1.31**	-	-	1.18	10.85	**10.49**	-	-	11.23	0.20	19.67	-	-	**0.18**
10 × 10	10 / 5	0.74	0.91	0.71	0.77	**0.96**	0.98	**1.23**	0.85	1.17	1.11	**11.50**	12.82	14.27	12.86	12.02	0.20	26.10	0.51	25.67	**0.18**
10 × 10	10 / 10	0.54	0.66	0.41	0.69	**0.87**	0.94	**1.16**	0.76	1.14	1.13	12.34	13.39	17.27	13.61	**12.34**	0.20	29.98	0.69	29.56	**0.18**
10 × 10	20 / 5	0.44	0.28	0.33	0.20	**0.93**	0.84	**1.12**	0.70	1.04	0.92	**12.34**	18.20	20.38	19.64	14.05	0.42	29.41	0.83	30.06	**0.36**
20 × 20	20 / 0	0.82	**1.00**	0.00	1.00	0.95	0.96	**1.35**	-	1.24	1.22	**21.65**	25.87	-	25.94	22.68	0.36	20.31	-	21.49	**0.35**
20 × 20	20 / 5	0.63	0.76	0.83	0.24	**0.87**	0.99	**1.32**	0.83	1.21	1.18	**22.09**	24.24	25.44	25.35	22.90	0.38	33.21	0.39	34.90	**0.35**
20 × 20	20 / 10	0.52	0.68	0.72	0.31	**0.73**	0.99	**1.28**	0.83	1.20	1.18	**22.53**	25.02	27.43	27.62	23.67	0.38	40.07	0.44	38.92	**0.38**
20 × 20	20 / 20	0.22	0.00	0.00	0.00	**0.52**	1.01	-	-	-	**1.18**	**25.02**	-	-	-	25.15	0.41	-	-	-	**0.41**
20 × 20	30 / 10	0.16	0.48	0.49	0.13	**0.53**	0.98	**1.25**	0.82	1.15	1.18	**25.09**	26.84	31.38	29.26	26.18	0.53	41.21	0.54	41.90	**0.48**

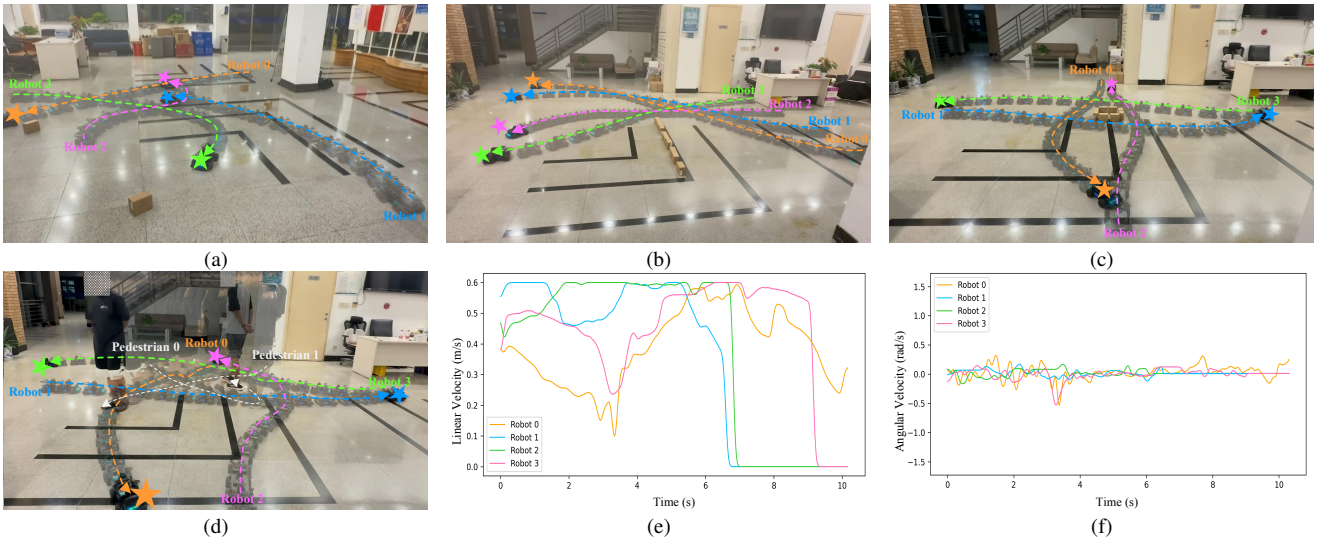


Fig. 5. Real-world experiment results. (a) Random scenario with four static obstacles. (b) Corridor scenario with four static obstacles. (c) Circular scenario with four static obstacles. (d) Circular scenario with two pedestrians. (e) Linear velocity variations in the circular scenario with two pedestrians. (f) Angular velocity variations in the circular scenario with two pedestrians.

proposed method.

We further test large-scale robot swarm navigation in circular scenarios. As shown in Table III, when the swarm size reaches 60, RL-RVO succeeds only 7 times out of 100 trials, whereas our method still maintains 58 successful trials. This demonstrates that our SwarmNav possesses superior generalization capability.

TABLE III

LARGE-SCALE ROBOT SWARM NAVIGATION

Robot Number	Methods	Success Rate	Average Speed (m/s)
40	RL-RVO	0.25	1.05
	Ours	**0.74**	**1.24**
60	RL-RVO	0.07	1.03
	Ours	**0.58**	**1.26**

C. Real-world Experiments

In the real-world experiments, we aim to further validate the generalization capability of our SwarmNav from simulation to reality. Specifically, we use four Ackermann-model vehicles for testing, each with dimensions of 0.3 m × 0.2 m, with a maximum linear velocity of 0.6 m/s, a maximum angular velocity of 1.1 rad/s, and a maximum acceleration of 0.5 m/s^2. Each vehicle is equipped with a Jetson Nano computing platform running Ubuntu 18.04 with ROS Noetic. In addition, each vehicle is equipped with a Mid-360 LiDAR to provide environmental perception data and 10 Hz global localization information obtained using the Point-LIO algorithm [32].

The real-world experiments are conducted in a 5 m × 5 m environment, considering four challenging scenarios: a random scenario with static obstacles, a corridor scenario with static obstacles, a circular scenario with static obstacles, and a circular scenario with dynamic obstacles. Note that we simulate dynamic obstacles using pedestrians moving at a constant speed of 0.3 m/s. Finally, the experimental results for the four scenarios are shown in Fig. 5, where Fig. 5(a)–(d) depict the motion trajectories of the veicles and pedestrians in the four scenarios, and Fig. 5(e) and (f) show the variations of the linear and angular velocities of each vehicle in the circular scenario with dynamic obstacles. As shown in Fig. 5, although our method is trained in a simulation environment and deployed across four real-world scenarios, it is still able to ensure that all robots safely reach their respective destinations. This demonstrates the excellent generalization and robustness of the proposed method. In particular, in the

circular scenario with dynamic obstacles, Fig. 5(e) and (f) show that Robots 0 and 3 exhibit noticeable fluctuations in their linear and angular velocities. These fluctuations occur between 3 and 4 seconds after departing toward their destinations, which corresponds to the moment they encounter pedestrians, indicating that each robot proactively adjusts its speed to avoid pedestrians. Overall, the real-world experiments further validate the generalization capability of our SwarmNav and demonstrate its robustness in real-world conditions.

VI. Conclusion

In this paper, we propose a novel swarm robot navigation method that enables each individual robot to reach its target position quickly and safely in dense and dynamic obstacle environments. We train the collision-avoidance strategy of each robot through a hierarchical reinforcement learning framework, allowing them to handle dynamic obstacles with motion uncertainty. Within this framework, we introduce a novel goal-region amplification strategy and incorporate the RVO formulation into the reward function design. Extensive simulations and real-world experiments demonstrate the effectiveness and strong generalization capability of our SwarmNav. In future work, we plan to explore more complex dynamic scenarios with obstacles of varied shapes.

References

[1] S. Saxena, R. Jais, and D. R. Sona, "Search and rescue operations using robots demonstrating swarm behaviour," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020, pp. 115–119.

[2] Y. Li, Y. Gao, S. Yang, and Q. Quan, "Swarm robotics search and rescue: A bee-inspired swarm cooperation approach without information exchange," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1127–1133.

[3] L. Abhang, A. Gummadi, R. Changala, V. A. Vuyyuru, S. R, and I. I. Raj, "Swarm intelligence for multi-robot coordination in agricultural automation," in *2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2024, pp. 455–460.

[4] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[5] D. Chaudhari and S. A. Bhosale, "A review on management of logistics using swarm robotics," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 0371–0373.

[6] J. Wen, L. He, and F. Zhu, "Swarm robotics control and communications: Imminent challenges for next generation smart logistics," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, 2018.

[7] M. Duarte, J. Gomes, V. Costa, T. Rodrigues, F. Silva, V. Lobo, M. M. Marques, S. M. Oliveira, and A. L. Christensen, "Application of swarm robotics systems to marine environmental monitoring," in *OCEANS 2016 - Shanghai*, 2016, pp. 1–8.

[8] T. Tosik, J. Schwinghammer, M. J. Feldvoß, J. P. Jonte, A. Brech, and E. Maehle, "Mars: A simulation environment for marine swarm robotics and environmental monitoring," in *OCEANS 2016 - Shanghai*, 2016, pp. 1–6.

[9] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Systems*, vol. 7, no. 01, pp. 55–64, 2019.

[10] K. Guo, D. Wang, T. Fan, and J. Pan, "Vr-orca: Variable responsibility optimal reciprocal collision avoidance," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4520–4527, 2021.

[11] G. Xu, Y. Chen, J. Cao, D. Zhu, W. Liu, and Y. Liu, "Multivehicle motion planning with posture constraints in real world," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 4, pp. 2125–2133, 2022.

[12] 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.

[13] L. Chen, Y. Wang, Z. Miao, M. Feng, Z. Zhou, H. Wang, and D. Wang, "Reciprocal velocity obstacle spatial-temporal network for distributed multirobot navigation," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 11, pp. 14 470–14 480, 2024.

[14] J. Huang, J. Zeng, X. Chi, K. Sreenath, Z. Liu, and H. Su, "Velocity obstacle for polytopic collision avoidance for distributed multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3502–3509, 2023.

[15] D. Martinez-Baselga, E. Sebastián, E. Montijano, L. Riazuelo, C. Sagüés, and L. Montano, "Avocado: Adaptive optimal collision avoidance driven by opinion," *IEEE Transactions on Robotics*, vol. 41, pp. 2495–2511, 2025.

[16] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.

[17] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 5628–5635.

[18] L. Wen, Y. Liu, and H. Li, "Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints," *Robotics and Autonomous Systems*, vol. 150, p. 103997, 2022.

[19] X. Zhang, G. Xiong, Y. Wang, S. Teng, and L. Chen, "D-pbs: Dueling priority-based search for multiple nonholonomic robots motion planning in congested environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6288–6295, 2024.

[20] 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.

[21] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[22] 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.

[23] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.

[24] M. Kollmitz, T. Koller, J. Boedecker, and W. Burgard, "Learning human-aware robot navigation from physical interaction via inverse reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 025–11 031.

[25] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671–5677.

[26] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2700–2719, 2023.

[27] R. Han, S. Chen, and Q. Hao, "A distributed range-only collision avoidance approach for low-cost large-scale multi-robot systems," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8020–8026.

[28] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.

[29] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:28695052

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[32] D. He, W. Xu, N. Chen, F. Kong, C. Yuan, and F. Zhang, "Point-lio: robust high-bandwidth light detection and ranging inertial odometry," *Advanced Intelligent Systems*, vol. 5, no. 7, p. 2200459, 2023.