International Journal of
Network Dynamics and Intelligence (IJNDI)

Scilight

Article

DRL-RVO: Learning to Avoid Collision in Crowd Using Reciprocal Velocity Obstacles

Liyunong Yang and Liang Hu *

School of Intelligence Science and Engineering, Harbin Institute of Technology, Shenzhen 518000, China
* Correspondence: l.hu@hit.edu.cn
Received: 28 December 2025; Revised: 14 March 2026; Accepted: 15 April 2026; Published: 22 June 2026

Abstract: This paper proposes DRL-RVO, a Deep Reinforcement Learning (DRL) framework incorporating the traditional Reciprocal Velocity Obstacles (RVO) for safe navigation. In this framework, the conventional RVO algorithm, which is widely used for dynamic obstacle avoidance in autonomous systems, is enhanced by integrating a DRL model to dynamically adapt to the behaviors of other human participants in the shared environment. Specifically, the weight coefficient of the RVO algorithm is updated using DRL in real time, improving navigation performance in dynamic environments shared with human participants. To evaluate the effectiveness of the proposed DRL-RVO algorithm, a comparative analysis is performed against state-of-the-art algorithms. The experimental results demonstrate the superior performance of the DRL-RVO algorithm in terms of navigation success rate, real-time adaptability, and efficiency, highlighting the potential benefits of integrating DRL into traditional RVO-based methods for autonomous navigation in dynamic environments.

Keywords: collision avoidance; deep reinforcement learning; reciprocal velocity obstacle

1. Introduction

Advanced by machine intelligence and robotics technologies, unmanned robots and vehicles are envisioned to share space and interact with humans in the near future. Traditional navigation approaches consider moving agents as static obstacles [1–4] or react to them through a one-step lookahead [5–7], which results in short-sighted, unsafe and unnatural behaviors. To navigate dense crowds in a socially appropriate manner, robots need to understand human behavior and follow their rules of cooperation [8–10].

In general, it is difficult to know motion intentions and preferences of humans or human-controlled systems, which poses a great challenge for safe interaction between autonomous and manned systems. Some data-driven trajectory prediction methods are proposed to model the interactions between the crowd [11–14]. Earlier work integrates these prediction models into the decision-making process through two separate steps, planning safe paths after predicting the future possible trajectories of others [15]. However, the probabilistic evolution of the crowd after a few steps can expand to the entire space in dense environments, leading to the freezing problem [16]. To solve this problem, a large number of works sought to jointly plan plausible paths for all decision-makers in a cooperative setting [16–18]. Nevertheless, when applied to dense environments, these methods suffer from the randomness of neighbors' behaviors as well as high computational costs.

As an alternative, reinforcement learning frameworks have been used to train computationally efficient policies that implicitly encode the interactions and cooperation between agents. Despite significant progress in [19–21], existing models still face two key limitations: (i) the collective influence of the crowd is often represented through simplified aggregations of pairwise interactions, such as the max-min operator [19] or LSTM [22], which may not fully capture the complexity of all interactions; (ii) most existing methods primarily focus on one-way influence from humans to the robot, overlooking the interactions within the crowd that could indirectly impact the robot's behavior. These limitations undermine the effectiveness of cooperative planning in dense and complex environments. High-order Crowd-Robot Interaction (CRI) modeling is exploited to address these limitations and have been incorporated into the DRL navigation framework in [23,24]. However, these RL-based approaches lack of training efficiency and interoperability, and always assume that the surrounding human participants show the same and fixed avoidance intention at the beginning, which is often inconsistent with real scenarios.

More recently, some methods have proposed to combine reinforcement learning algorithms with traditional VO or RVO algorithms, but they generally use VO (RVO) to shape the reward function [25–27] or simply to replace the RL exploration process [28] rather than directly applying the traditional algorithms to robot control. The RVO method assumes that all participants share the responsibility of collision avoidance, which is generally hard to know exactly, and thus, the common practice uses a simplified equal responsibility assumption. DRL method

directly generates control commands of navigation, such as degree speed, etc., from the initial values of the shared environment, which is not very efficient in densely populated environments with high randomness. To fill the gap between the two methods, we propose DRL-RVO, which exploits the merits of the DRL algorithm and the traditional obstacle avoidance algorithm RVO. Without assuming the extent of cooperation/non-cooperation of the participants, we use DRL algorithm to learn an optimal cooperative coefficient of the generalized RVO that the autonomous robot should use in its environment. The main contribution of this paper is summarized as follows:

- We propose a novel framework DRL-RVO that adjusts the cooperative intent of the autonomous robot by learning from the crowd behaviors in the surroundings using the integrated DRL and RVO algorithm;
- We design a dedicated structure that reduces the framework to a one-dimensional optimization problem, accelerating the training process greatly;
- Extensive simulation experiments demonstrate the superior performance of our DRL-RVO. It outperforms other comparative methods in complex dynamic scenarios where different human participants have different reaction times and avoidance intentions

2. Related Work

2.1. Reciprocal Velocity Obstacle

Reciprocal Velocity Obstacles (RVO) enhance the Velocity Obstacle (VO) [29] framework by enabling cooperative collision avoidance among agents [5]. Unlike VO, which assumes passive obstacles, RVO requires agents to share the responsibility in avoiding collisions. Each agent adjusts its velocity as a weighted average of its current motion and collision-free velocities derived from the predicted paths of neighbors, ensuring safe and oscillation-free trajectories. RVO has been further extended into Optimal Reciprocal Collision Avoidance (ORCA) [6], which optimizes velocity selection by applying linear programming with half-plane constraints in velocity space. Despite its advantages, RVO and its variants assume perfect sensing and do not account for real-world uncertainties in localization and control. To address these limitations, some methods introduce probabilistic versions, such as PRVO [30], to handle Gaussian uncertainties more effectively. However, these methods tend to be overly cautious, which can result in less efficient navigation, especially in dense or highly dynamic environments.

2.2. Deep Reinforcement Learning

DRL-based collision avoidance methods can leverage extensive training experiences, making them highly effective in managing complex scenarios with improved efficiency and robustness. Specifically, collision avoidance with DRL (CADRL) employs a value network to generate a collision-free path toward the goal [19]. Some approaches focus on the sensor-level collision avoidance policy that maps raw sensor data to the robot control vector with end-to-end training [21,31]. Different from sensor-level methods, agent-level approaches make high-level decisions and path planning based on global environment models (such as maps, mission objectives) other than raw sensor data [20].

Unlike sensor-level methods, agent-level DRL approaches use environment models rather than relying directly on raw sensor data, enabling high computational efficiency and greater flexibility in terms of handling various sensor types and kinematic/dynamic characteristics [20]. However, these methods require the input data dimensions to be fixed. To circumvent such a limitation, some techniques assume a constant upper bound of the number of obstacles [32]. Recurrent Neural Networks (RNNs) are able to deal with a varying number of moving obstacles, as demonstrated in [22], using an LSTM model to create a fixed-size feature vector to represent the environment and to handle the state of each neighbor in reverse order of their distance from the robot. However, such a fixed-size vector probably gives more weight to the final section of the input sequences, which limits the performance in the obstacle-dense environment. To address this issue, the socially aware RL (SARL) introduces [23] a socially attentive network consisting of three main modules: interaction, pooling, and planning. It uses a self-attention mechanism to assess the relative importance of dynamic obstacles in the environment. Our work builds on these models and combines the RVO algorithm with DRL algorithms to deal with highly random crowd scenes more efficiently.

2.3. Combining Deep Reinforcement Learning with RVO

Some recent studies have combined reinforcement learning (RL) with traditional VO or RVO algorithms. However, these methods generally either use VO/RVO to shape the RL reward function [25–27] or to guide RL exploration [28], rather than directly controlling the robot. As a result, the RVO parameters in these methods are typically fixed or predetermined, and do not adapt to the real-time behavior of other participants.

In contrast, our proposed DRL-RVO dynamically adjusts the cooperative coefficient of the RVO in real time.

By integrating DRL to directly update RVO weights based on the current environment, DRL-RVO can efficiently navigate densely populated and highly dynamic scenarios.

3. System Setup and Problem Statement

3.1. System Framework

In our system, a group of omnidirectional drive robots, which represent 'humans', and a controlled agent, which represents 'robot', navigate in a shared workspace along x and y directions. Each human and the robot have no communication with the others but can sense the surrounding agents within a certain range. The information observed is the radius, current position, and current velocity. In addition to the observable statement above, the full state also includes the goal position and preferred velocity. Unlike most learning-based methods whose navigation policy output is the direct control vector of the robot (e.g. speed), the output control of our approach is a cooperation coefficient, which is also called the weight of RVO, used to determine the effort of agents to avoid collision. Then the coefficient is passed into the RVO method to calculate the speed of the robot agent at the next moment. For other human agents, RVO is directly used to select the optimal speed for mutual obstacle avoidance. The structure of our method is shown in Figure 1.

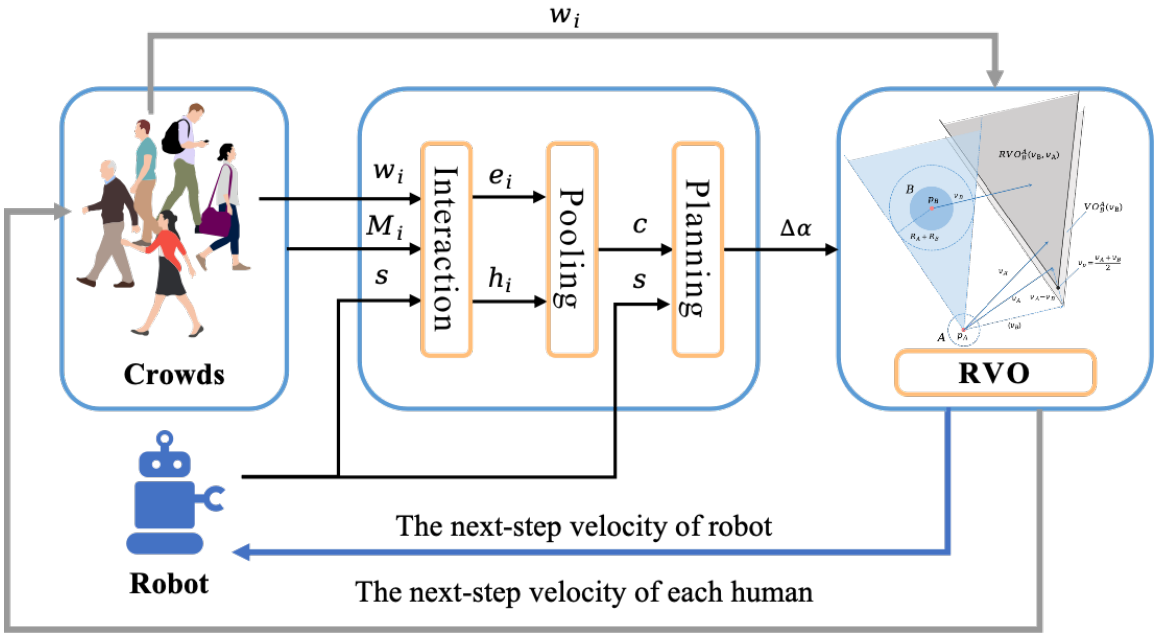


Figure 1. Structure of our method. RVO is used for getting the next-step velocity of the robot and for each human avoiding collision of each other. The variables passed into the network are: w_i is the state of each human i, M_i is the map tensor centered at each human i, and s is the state of the robot. $\Delta\alpha$ is the increment of the weight coefficient of RVO, which is obtained by learning. For simplicity, we ignore the subscript time t.

3.2. Reciprocal Velocity Obstacle

This section summarizes the geometrical definition of the VO and RVO. Reciprocal Velocity Obstacle (RVO) is an extension of the traditional Velocity Obstacle (VO) used for multi-agent collision avoidance. While VO is typically employed for dynamic obstacle avoidance by a single robot, RVO is designed for a group of robots to avoid mutual collisions.

Given two robots, A and B, with radiuses R_A and R_B, the positions and velocities denoted as p_A, p_B, v_A, v_B, respectively. The VO area of the robot A relative to the robot B is defined as:

$$VO_B^A(v_B) = \{v_A | \lambda(p_A, v_A - v_B) \cap B \oplus -A \neq \varnothing\} \tag{1}$$

where $\lambda(p_A, v_A - v_B)$ denotes the a ray starting at p_A and heading in the direction of the relative velocity of A and B, i.e., $(v_A - v_B)$, and $\oplus$ denotes the Minkowski sum. The formula above indicates that if $v_A \in VO_B^A(v_B)$, a collision between A and B is expected in the future. If v_A is outside the velocity obstacle of B, both objects will never collide. If v_A is on the boundary of the velocity obstacle, a collision between A and B will happen at some moment. As shown in Figure 2a, the velocity obstacle is a cone with an apex at v_B.

RVO, on the other hand, extends from the VO concept geometrically for multi-agent scenarios. Instead of

choosing a new velocity for each agent that is outside the velocity obstacle of the other agent, RVO chooses a new velocity that is the average of its current velocity and a velocity that lies outside the velocity obstacle of the other agent. As illustrated in Figure 2b, the RVO of agent A relative to agent B can be expressed as:

$$RVO_B^A(v_B, v_A) = \{v_A^{'} | 2v_A^{'} - v_A \in VO_B^A(v_B)\} \tag{2}$$

In practice, it is usually implicitly assumed that each agent shares an equal effort to avoid mutual collisions above. To model the natural priorities and different cooperative levels among agents, the RVO concept can be generalized by introducing a weight coefficient α_B^A [5], which means the share of the effort agent A takes to avoid agent B. The weight is defined such that:

$$\alpha_A^B = 1 - \alpha_B^A \tag{3}$$

This means that the cooperation weight is shared between the two agents: if one agent takes more responsibility, the other takes less. The Generalized Reciprocal Velocity Obstacle (GRVO) of agent B to agent A can be defined as follows:

$$\begin{aligned} RVO_B^A(v_B, v_A, \alpha_B^A) = \\ \{v_A^{'} | \frac{1}{\alpha_B^A} v_A^{'} + (1 - \frac{1}{\alpha_B^A}) v_A \in VO_B^A(v_B)\} \end{aligned} \tag{4}$$

It can geometrically be interpreted as the apex of $VO_B^A(v_B)$ move to $(1 - \alpha_B^A)v_A + \alpha_B^A v_B$, as shown in Figure 2c. Moreover, the value of cooperation weight coefficient α ranges from 0 to 1 (0 means no intention of cooperation at all, and 1 means high cooperation intention), and the coefficient of each agent is independent of each other, that is, each agent can choose its cooperation coefficient based on its current environment, goals, and strategies.

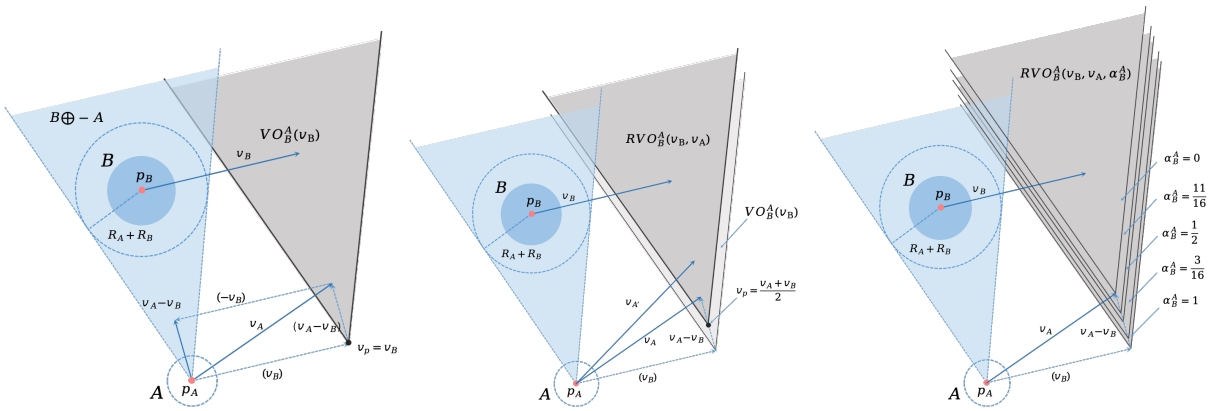


Figure 2. A diagram showing the velocity obstacle and reciprocal velocity obstacle for a pair of dynamic agents A and B.

3.3. Problem Statement

In this work, we consider a navigation task in which a robot moves towards a goal through a crowd of n humans. To tackle this challenge, we represent the uncertainty-aware behavior planning problem as a Partially Observable Markov Decision Process (POMDP). A POMDP model is defined by the tuple $< S, A, Z, P, O, R >$, where S, A, and Z denote the state space, action space, and observation space, respectively. The function $P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) = p(\mathbf{s}_{t+\Delta t}^{jn} | \mathbf{s}_t^{jn}, \mathbf{a}_t)$ is the transition probability from time t to time $t + \Delta t$, and $O(\mathbf{s}_t^{jn}, z_t) = p(\mathbf{s}_t^{jn} | z_t)$ is the observation model, describing the likelihood of receiving observation z_t given the true state. These two functions reflect the stochastic property of the motion model and uncertain sensing. The reward function $R(\mathbf{s}_t^{jn}, \mathbf{a}_t)$ evaluates the immediate benefit of taking action $\mathbf{a}_t$ in state $\mathbf{s}_t^{jn}$. Due to partial observability, where not all human states are directly accessible, the robot maintains a *belief* $b \in B$, which is the probability distribution over S. The updated belief can be derived using Bayes' rule:

$$\begin{aligned} b_t(\mathbf{s}_{t+\Delta t}^{jn}) &= p(\mathbf{s}_{t+\Delta t}^{jn} | \mathbf{z}_{t+\Delta t}, a_{t+\Delta t}, b_t) \\ &= \eta O(\mathbf{s}_{t+\Delta t}^{jn}, \mathbf{z}_{t+\Delta t}) \int_{\mathbf{s}_t^{jn} \in S} P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) b_t(\mathbf{s}_t^{jn}) \mathrm{d}\mathbf{s}_t^{jn}, \end{aligned} \tag{5}$$

where η is a normalizing factor. In practice, we approximate the belief state by directly using the available observations, including the robot's proprioceptive information and exteroceptive measurements of surrounding humans. This belief approximation enables efficient learning and decision-making without the need for explicit belief tracking.

For each agent (robot or human), the position $\mathbf{p} = [p_x, p_y]$, velocity $\mathbf{v} = [v_x, v_y]$, and radius r can be observed by the others, which is also called exteroceptive observation measurements. Furthermore, the robot is aware of its unobservable state, also called the proprioceptive observation measurements, which include the position of the target $\mathbf{p}_g = [g_x, g_y]$, the preferred speed v_{pref}, and the weight coefficient α. We assume that the velocity of the robot v_t can be achieved instantly after the control command of the speed. Let s_t denote the state of the robot, and $\mathbf{w}_t = [w_t^1, w_t^2, ..., w_t^n]$ denote the state of humans at time t. Then, $\mathbf{s}_t = [p_x^r, p_y^r, v_x^r, v_y^r, r^r, g_x^r, g_y^r, v_{pref}^r, \alpha^r]$, and $\mathbf{w}_t^i = [p_x^{ih}, p_y^{ih}, v_x^{ih}, v_y^{ih}, r^{ih}]$, where the superscript r represents the robot, while ih represents the ith human. The joint state for robot navigation is defined as $\mathbf{s}_t = [\mathbf{r}_t, \mathbf{w}_t]$.

The objective of the RL learning is to find the optimal motion planning policy, $\pi^* : \mathbf{s}_t^{jn} \mapsto \mathbf{a}_t$ ($\mathbf{a}_t = \Delta\alpha_t$ is action taken at time t), which through repeated exploratory training to guide the robot to reach the target point as soon as possible under the condition of safety, as shown in Equation (4) [23].

$$\begin{aligned} \pi^*(\mathbf{s}_t^{jn}) = \mathop{\mathbf{argmax}}_{\mathbf{a}_t} R(\mathbf{s}_t^{jn}, \mathbf{a}_t) + \\ \gamma^{\Delta t v_{pref}} \int_{\mathbf{s}_t^{jn}+\Delta t} P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) V^*(\mathbf{s}_{t+\Delta t}^{jn}) \mathrm{d}\mathbf{s}_{t+\Delta t}^{jn} \\ V^*(\mathbf{s}_t^{jn}) = \sum_{t'=t}^{T} \gamma^{t' \cdot v_{pref}} R_{t'}(\mathbf{s}_{t'}^{jn}, \pi^*(\mathbf{s}_{t'}^{jn})) \end{aligned} \tag{6}$$

where $R(\mathbf{s}_t^{jn}, \mathbf{a}_t)$ is the reward received at time t, $\gamma \in (0, 1)$ is a discount factor, and V^* is the optimal state-action value function, $P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn})$ is the transition probability from time t to time $t + \Delta t$, while a_t is implemented at time t. The preferred velocity v_{pref} is used as a normalization term in the discount factor for numerical reasons [19].

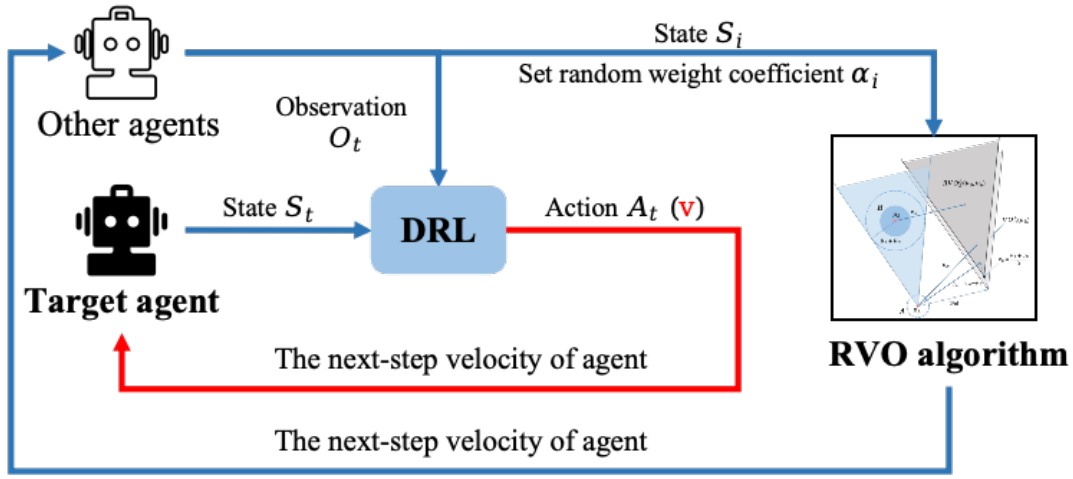
In our approach, the action $\Delta\alpha_t$ dynamically adjusts the weight coefficient, modifying the robot's collision avoidance strategy computed through the RVO framework. Following each action, the robot updates its belief based on the resulting observation and the altered transition dynamics. The forward prediction process consists of selecting $\Delta\alpha_t$ based on the current belief, updating the cooperation weight, recomputing the velocity with RVO, executing motion, and refreshing the belief accordingly. By integrating reward-driven weight adaptation within the POMDP framework, our method enables the robot to predictively and adaptively optimize its motion strategy in dense, dynamic crowds, achieving a balance between efficiency and safety in navigation.

4. DRL-RVO Framework

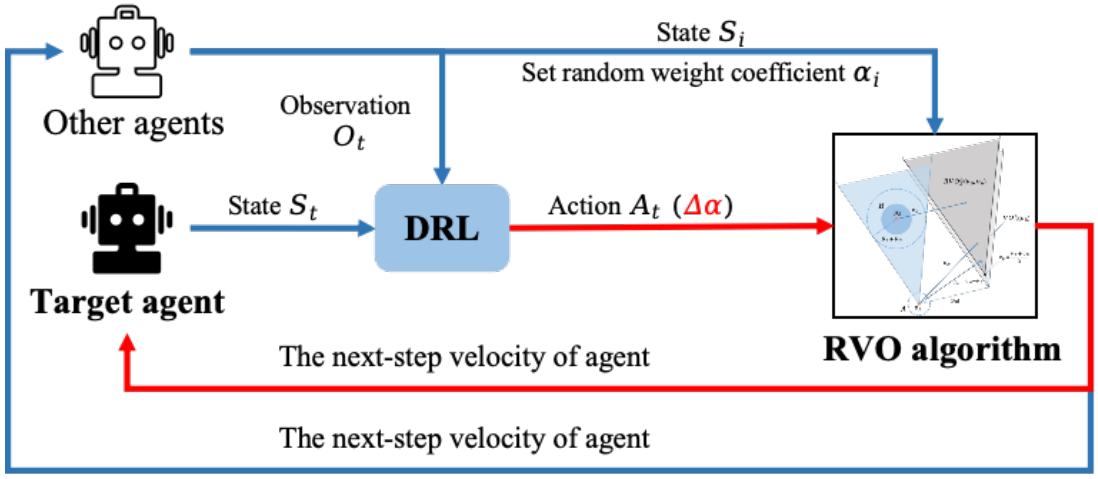
In this study, we directly employ the socially attentive network introduced in [23], which consists of three modules: interaction, pooling, and planning. This method has proven to be effective in solving complex collision avoidance problems in dynamic and crowded environments. However, there are some important distinctions in our work and the original DRL framework.

4.1. State Space and Action Space

The state space contains the proprioceptive measurements $\mathbf{r}_t = [p_x, p_y, v_x, v_y, r, g_x, g_y, v_{pref}, \alpha]$, and exteroceptive measurements $\mathbf{w}_t = [w_t^1, w_t^2, ..., w_t^n]$, and $\mathbf{w}_t^i = [p_x^i, p_y^i, v_x^i, v_y^i, r_i]$, as mentioned in Sec.III.C. The joint state for robot navigation is $\mathbf{s}_t = [\mathbf{r}_t, \mathbf{w}_t]$. The action space is different from most learning-based methods whose action space is always a probability distribution of velocity [23,25,26,32]. Our action space is a uniformly distributed increment of RVO weight coefficient of RVO, i.e., $\Delta\alpha$. Then the current RVO weight parameters of the agent update, i.e., $\alpha_{t+1} = \alpha_t + \Delta\alpha$. At each time step, in the condition of state $\mathbf{s}_t$, the robot selects the with the highest probability from an action space to navigate following the stochastic policy. In the real-world situation, the robot only moves forward because of the limited view of observation, and the range of transitional velocity is limited between 0 and 1.0 m/s. Figure 3 shows the basic frameworks of two kinds of learning-based methods.



(a)



(b)

Figure 3. The structure of (**a**) common DRL-based methods and (**b**) our method.

4.2. Input and Output of Network

The network we use has three modules: interaction, pooling, and planning. The input and output of each module are as follows, and the time index t is omitted below for simplicity.

- **Interaction module:** models the Human-Robot interactions explicitly and encodes the Human-Human interactions through coarse-grained local maps. L is the size of a neighborhood. A $L \times L \times 3$ map tensor M_i centered at each human i, which encodes the presence and velocities of neighbors, enters the interaction module with the state of each human w_i and the state of the robot s. Then those are embedded into a fixed-length vector e_i by an MLP. The e_i is fed to a subsequent MLP to obtain the feature h_i, which represents the pairwise interaction between the robot and person i.
- **Pooling module:** aggregates the interactions into a fixed-length embedding vector by a self-attention mechanism. The interaction embedding e_i is sent into mean pooling to obtain a fixed-length embedding vector e_m. Then e_i and e_m are transformed into attention score α_i through an MLP. After that, the final representation of the crowd is a weighted linear combination c of pairwise interaction vector h_i and the corresponding attention score α_i for each neighbor i, i.e., $c = \sum_{i=1}^{n} \mathbf{softmax}(\alpha_i) h_i$.
- **Planning module:** estimates the value of the joint state of the robot and crowd for social navigation. The representation of the crowd c and the state of the robot s are sent into an MLP to estimate the state value for cooperative planning.

4.3. Reward Function

The goal of DRL algorithm is to maximize the reward of a series of actions. We follow the formulation of the reward defined in [19,20,23], which considers the task accomplishments and collisions or uncomfortable distances, as shown below:

$$R_t(\mathbf{s}_t^{jn}, \mathbf{a}_t) = \begin{cases} -2 & if\ d_t < 0 \\ -0.1 + dt/2 & else\ if\ d_t < 0.2 \\ 2 & else\ if\ p_t = p_g \\ 0 & otherwise \end{cases}$$

where d_t is the minimum separation distance between the robot and the person during the period $[t - \Delta t, t]$. The reward function gives rewards when the agent reaches the goal and penalties when it collides with or gets too close to others. Our reward function encourages goal-reaching and collision avoidance, and implicitly guides the adjustment of cooperation weights. Specifically, positive rewards for successful navigation reinforce the selection of appropriate $\Delta\alpha_t$, while negative rewards for collisions or unsafe proximity penalize inappropriate weight changes. Thus, the reward signal effectively drives the learning of dynamic cooperation behaviors.

4.4. Implementation Details

4.4.1. Network Architecture

The network architecture is built to enable the robot to navigate in dynamic environments by taking into account both individual human behaviors and collective interactions within the crowd. The local map is a 4×4 grid centered at each human, with each cell having a side length of 1 m. For imitation learning, we collected 3 k episodes of demonstration using ORCA [5] and trained the policy 50 epochs with learning rate 0.01. The following configurations are used for the different DRL algorithms:

For baseline methods, we implement four existing state-of-the-art methods, ORCA [5], CADRL [19], LSTM-RL [22], SARL [23], and AEMCARL [24]. The following configurations are used for the different DRL algorithms:

- **CADRL:** For CADRL, the Multi-Layer Perceptron (MLP) dimensions are set to (150, 100, 100, 1). This configuration is used without multi-agent training, as we focus on the performance of a single agent in our evaluation.
- **LSTM-RL:** The global state dimension is set to 50, with two MLPs: the first MLP has dimensions (150, 100, 100, 50), and the second MLP is defined as (150, 100, 100, 1). In this case, multi-agent training is enabled, but the interaction module and the observation map (OM) are disabled for this specific setup, allowing the model to focus solely on the LSTM network for state-based learning.
- **SARL:** The architecture for SARL includes an MLP with dimensions (150, 100) for the first layer, followed by (100, 50) for the second layer. The attention module has dimensions (100, 100, 1), and the third MLP layer is (150, 100, 100, 1). SARL also enables multi-agent training, and incorporates a global state, but disables the observation map module to simplify the learning task. Also, in our method, the configuration of the SARL module is the same as above.
- **AEMCARL:** The architecture of AEMCARL consists of a hierarchical environment model (HEM), an adaptive perception mechanism (APM), and a Transformer-based attention module. The HEM utilizes three layers of Gated Recurrent Units (GRUs) to model dynamic environments, where each GRU refines the representation based on the previous layer's output. The APM adaptively adjusts the number of GRUs used based on perception confidence. The Transformer-based attention module extracts key interaction features among agents. The hidden units of the AEM module, the TF module, and the action module are [(100, 50)], [(150, 150, 150), nhead:2], and [(150, 100, 100, 1)], respectively. This framework supports multi-agent training while disabling the observation map module, focusing on efficient and adaptive environment modeling.

4.4.2. Hyperparameters

For training the RL agents, the following hyperparameters are used across all methods:

- **Discount Factor:** γ is set to 0.9 for all methods, ensuring a balance between immediate rewards and long-term goals.
- **Exploration Strategy:** We use an ϵ-greedy policy, where the exploration rate decays linearly from 0.5 to 0.1 in the first 5 k episodes and stays 0.1 for the remaining 4 k episodes, allowing the agent to explore the environment early in the training and then shift towards exploitation.

- **Learning Rate:** For all reinforcement learning methods, the learning rate is reduced to 0.001 to allow for finer adjustments during training.

This work assumes holonomic kinematics for the robot, meaning it can move freely in any direction. The action space is uniformly sampled between -1.0 and $+1.0$ with 0.1 as the sampling interval, which represents weight increments applied to the RVO-based collision avoidance model.

5. Experiments

5.1. Simulation Setup

We built a simulated environment in Python for robots to navigate in crowds. The simulated humans are controlled by ORCA [6], to introduce behavioral diversity we sample the parameters from a Gaussian distribution. For the weight coefficient of each human, it is not set to a fixed value of 0.5, but is randomly set to a value between 0 and 1 when the state is initialized, and the weight coefficient is unknown to the robot. Specifically, when each round is initialized, a time value t_i and a weight value α_i between 0–1 are randomly generated for each human, indicating that at the moment t_i, the RVO weight coefficient changes from 0 to α_i, as shown in Figure 4. We use this change to simulate the human's intention to avoid the robot. For instance, if t_i is small and α_i is large, it means that the human reacts quickly to the robot and has a high intention to avoid.

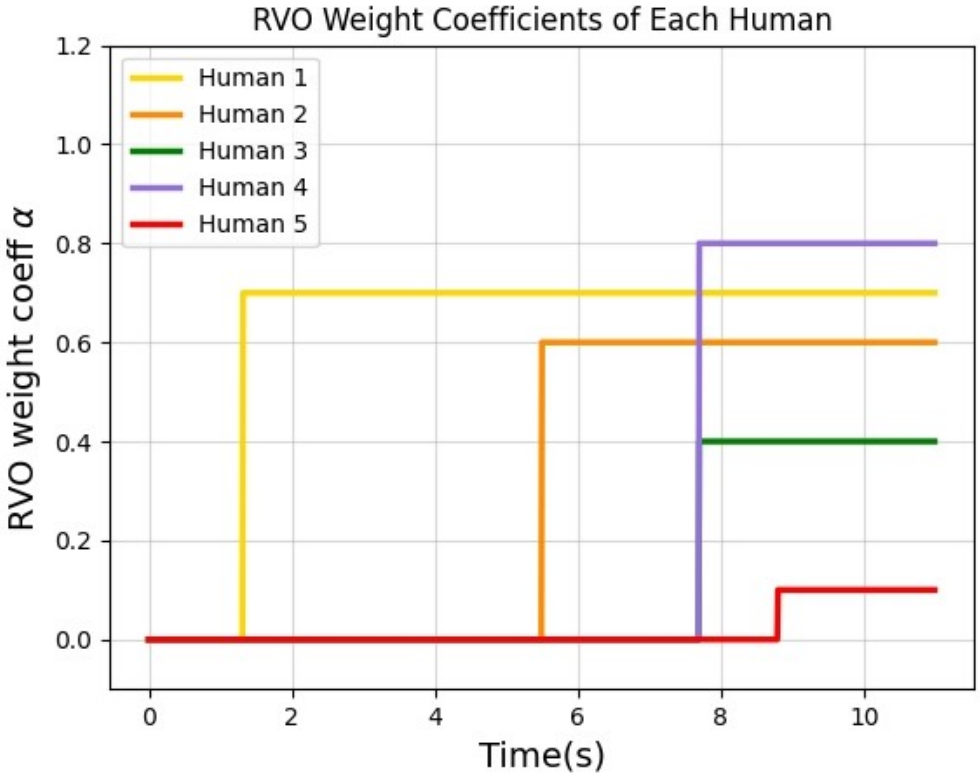


Figure 4. The changes in the RVO weight value of each human during the training process. In one episode, at a random time t_i, the weight changes from 0 to a random α_i and then keeps the value.

In training and testing, we use a circle crossing scenario, where all humans are randomly placed on a circle of radius 4 m, and their goal positions are on the other side of the same circle. Random perturbations are added to the x, y coordinates of the begin and goal positions. As mentioned in Sec.IV.C, the baseline methods of our work are ORCA [5], CADRL [19], LSTM-RL [22], SARL [23], and AMCARL [24]. Specifically, SARL incorporates unique interaction and pooling modules, which differentiate it from the other RL baselines, though the planning module remains consistent across all methods. In our implementation of LSTM-RL, we make a key modification by using joint states as inputs to the LSTM unit, instead of the original observable states of human agents, as done in [23].

To fully evaluate the effectiveness of the proposed model, we study two simulation settings: invisible and visible. The former setting in which the robot is invisible to other humans. Therefore, the simulated humans only react to other humans, not to the robot. We also remove the penalty for discomfort distance in the reward function to eliminate the additional factor of collision avoidance. This setting serves as a clean testbed for validating the model's ability to reason about human-robot and human-human interactions without affecting human behaviors.

The latter visible setting is more similar to the real-world situation where the robot and humans have mutual impacts. In both settings, the models are evaluated using 1000 random test cases. Moreover, in the real world, people often have reaction time to avoid collision and have different social behaviors, such as selfishness, adaptation, cooperation, etc. We use time-varying and random cooperation coefficients to represent this. For the robot, we set the initial avoiding coefficients to 0 in all situations.

In our experiments, the proposed DRL-RVO model achieved training completion within approximately 10 h on a NVIDIA GeForce RTX 3080, compared to around 150 h required by other reinforcement learning baselines under identical computational settings. We also measured the inference latency. DRL-RVO requires 0.0588 ± 0.016 s per step, compared to 0.0898 ± 0.002 s for CADRL and 0.0656 ± 0.005 s for SARL. These results demonstrate the superior training efficiency of our method, making it highly suitable for practical applications where computational resources and time are constrained.

5.2. Quantitative Evaluation

5.2.1. Invisible Robot

Table 1 reports the success rate, collision rate, average navigation time, and average discounted cumulative reward in the invisible test experiments. In this setting, the robot mainly relies on predicting the future trajectory of humans to avoid collisions. As expected, the ORCA method fails badly in the invisible setting due to the violation of the reciprocity assumption and the significant increase in randomness. CADRL can only consider one pair of interactions and ignore the rest. LSTM-RL, SARL, and AEMCARL all directly aggregate information from surrounding agents. We observe that some baseline methods failed to converge consistently across multiple runs. To quantify this, we define the *convergence rate* as the ratio of successful convergence in ten independent training runs. For fair comparison, we use the product of the convergence rate and the maximum success rate, which is defined as *Convergence-Weighted Success (CWS)*, as the evaluation metric.

The results show that after introducing RVO, our proposed method not only maintains a high success rate but also achieves consistent convergence in every run, highlighting its robustness and reliability in dynamic environments.

Table 1. Quantitative results in the invisible setting. "Success": The highest success rate achieved by each method, i.e., the maximum probability that the robot reaches the goal without collision. "Collision": The lowest collision rate, i.e., the minimum probability that the robot collides with another human. "Time": The average time in seconds the robot takes to reach the goal. "Reward": The discounted cumulative reward in the navigation task. "Convergence Rate": the ratio of successful convergence in ten independent training runs. "CWS": The product of the convergence rate and the maximum success rate of each method. **Bold** and underline highlight the best and second best results, respectively.

Methods	Success	Collision	Time	Reward	Convergence Rate	CWS
ORCA [5]	0.21	0.79	10.86	0.2174	1.0	0.21
CADRL [19]	0.82	0.18	**10.84**	0.4995	0.8	0.656
LSTM-RL [22]	0.95	0.03	11.95	0.5339	0.2	0.19
SARL [23]	0.97	0.02	11.26	0.5667	0.4	0.388
AEMCARL [24]	**0.99**	**0.01**	11.19	**0.5835**	0.4	0.396
DRL-RVO	0.95	0.05	11.23	0.5435	**1.0**	**0.95**

5.2.2. Visible Robot

We further evaluate the navigation performance of our models against the baselines in the visible setting. In this scenario, the robot must not only comprehend human behaviors but also actively interact with them to maximize rewards. We define discomfort frequency as t_{disc}/T, where t_{disc} represents the duration during which the separation distance d_t is less than 0.2 m. To ensure a fair comparison between the ORCA baseline and learning-based methods, we introduce an additional 0.1 m as the agent's virtual radius, promoting a safer distance for human-aware navigation, as suggested in [9]. The results in Table 2 indicate that most reinforcement learning results in the visible setting are similar to those in the unseen setting. Except for our DRL-RVO method, which has a significantly higher success rate, the remaining RL methods all have timeouts for their complex calculations. As expected, our DRL-RVO method still significantly outperforms the other methods.

Table 2. Quantitative results in the visible setting. Note that the initial weight of the robot is 0.5 when using policy 'ORCA'. **Bold** and underline highlight the best and second best results, respectively.

Methods	Success	Collision	Time	Reward	Convergence Rate	CWS
ORCA [5]	0.76	0.24	**10.52**	0.4905	1.0	0.76
CADRL [19]	0.95	0.05	10.85	0.5575	0.5	0.475
LSTM-RL [22]	0.97	0.02	11.29	0.5616	0.2	0.194
SARL [23]	0.98	0.01	11.15	0.5852	0.2	0.196
AEMCARL [24]	**0.99**	**0.00**	11.06	**0.5867**	0.3	0.297
DRL-RVO	0.96	0.04	11.13	0.5675	**1.0**	**0.96**

In addition, we also modified the number of humans to test the generalizability of our method. The number of humans used in our training was 5. We tested both invisible and visible scenarios when the human number was 1, 2, 5, 10, and 20. The experiment results are shown in Figure 5. Figure 5a shows the success and collision rates of the two scenarios. It can be seen that as the number of humans increases, the success rate decreases and the collision rate increases. However, even in the case of dense humans, the success rate of DRL-RVO can reach more than 60%. It is consistent with the common sense that the success rate is higher when the robot is visible than when it is invisible, because in the visible case, humans will respond to the robot, so that both the robot and humans will share the obstacle avoidance responsibility, which is less difficult than if the robot only makes all the obstacle avoidance responses. Figure 5b shows that in the two scenarios, the navigation time increases with the increase in the number of humans. Similarly, the navigation time in the visible scenario is shorter than that in the invisible scenario, and the navigation time increases as the number of humans increases.

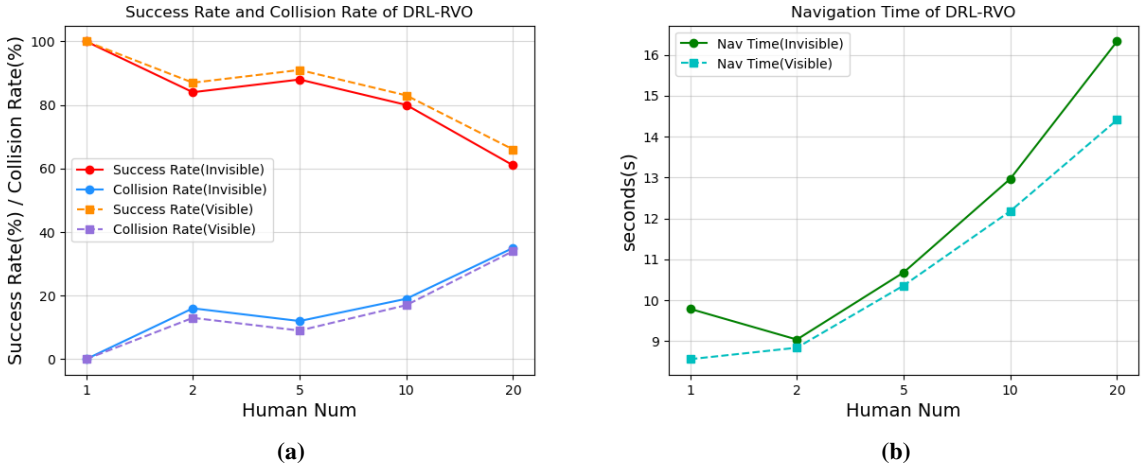
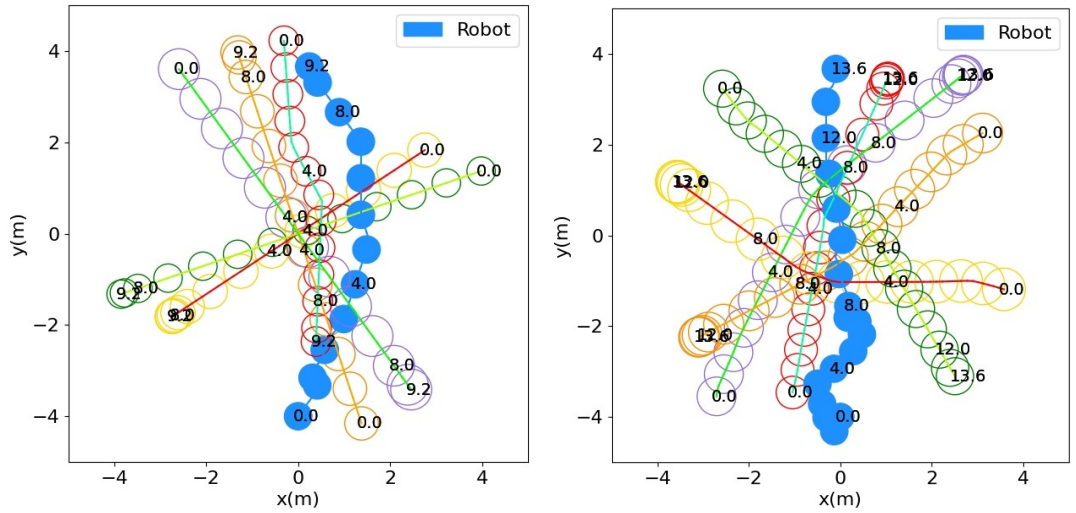


Figure 5. A comparison of (**a**) success rate, collision rate, and (**b**) navigation time for different numbers of humans in invisible and visible cases.

5.3. Qualitative Evaluation

The subsection presents the qualitative analysis of the proposed model. When the model encounters human agents located at the centre of the environment, it exhibits a distinct behaviour in contrast to the baseline methods. For our model, at the beginning of training, with randomly initialized scenarios, it is difficult for the agent to complete the crowd navigation task, and most of the terminal states will result in "Timeout" or "Collision". As training progresses, the robot quickly learns to keep a safe distance from human trajectories. It gradually understands the behavior of the crowd and plans its path based on the prediction of the pedestrians' trajectory. At the end of training, the robot's performance becomes relatively stable.

We investigate the effectiveness of our model through trajectory analysis. The SARL and the AEMCARL are first set with the initial condition that the surrounding humans' RVO weights α_{human} are fixed at 0.5, for which both methods complete the navigation task, as shown in Figure 6a,b. Then we randomly set the humans' RVO weights to change at the beginning of each episode. In this case, our method shows obvious advantages.

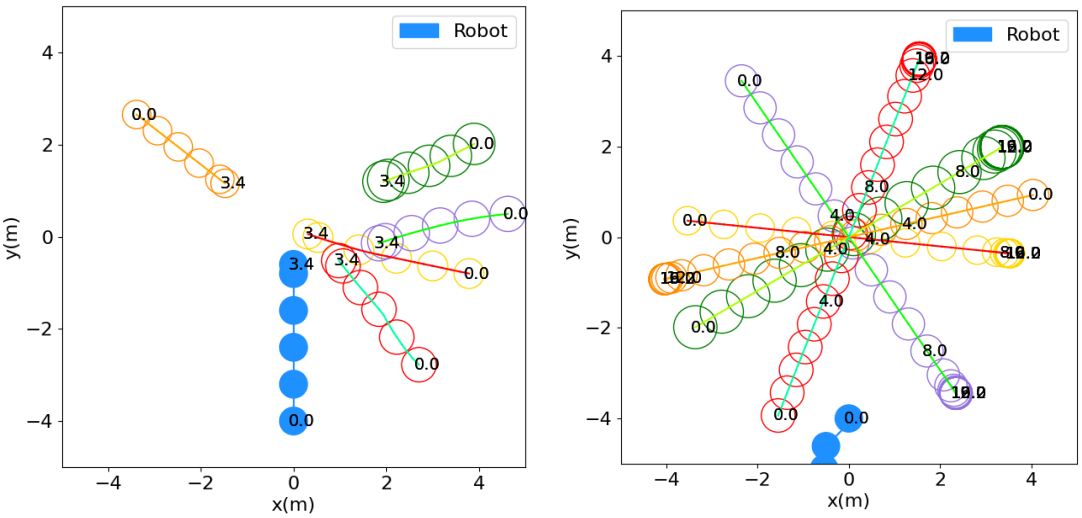


(a) SARL($\alpha_{human} = 0.5$) **(b)** AEMCARL($\alpha_{human} = 0.5$)

Figure 6. Both (**a**) SARL and (**b**) AEMCARL methods complete the navigation task successfully when all participants follow equal reciprocation.

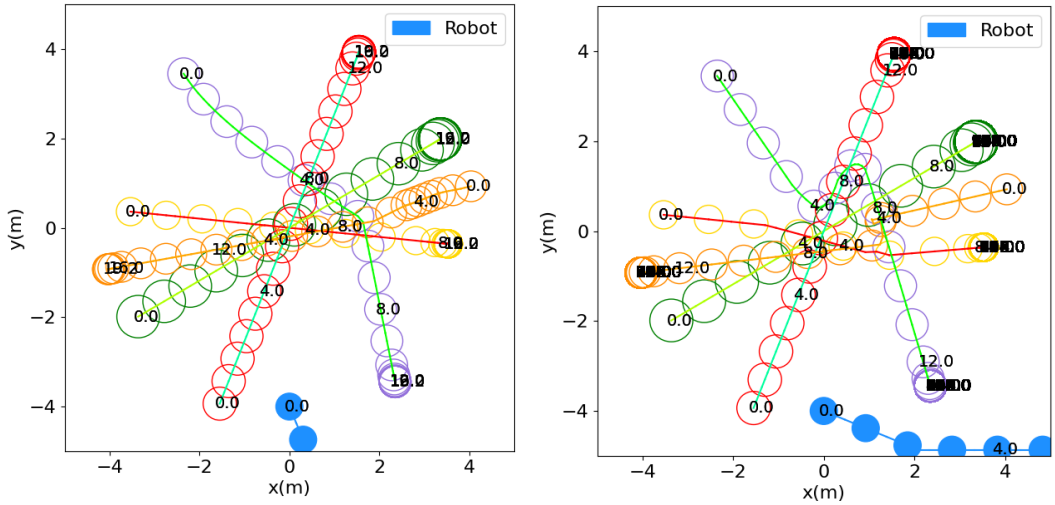
ORCA (Figure 7a) leads to a collision due to its purely reactive nature, lacking predictive capability for unseen agents. CADRL (Figure 7b), LSTM-RL (Figure 7c), SARL (Figure 7d), and AEMCARL (Figure 7e) all generate overly conservative trajectories, resulting in unnecessary detours, slowing progress toward the goal, and ultimately failing to reach the goal point after a timeout. In contrast, our DRL-RVO method (Figure 7f) achieves a balanced navigation behavior. The robot maintains smoother and safer trajectories, adapts its cooperation weight dynamically to negotiate through the crowd, and reaches the goal efficiently without unnecessary detours or close interactions. These results highlight the effectiveness of our belief-based dynamic weight adjustment under partial observability. The results show that other advanced RL methods can only work under the ideal assumption that human participants work with the same willingness to cooperate at the beginning and cannot handle a wider range of real situations–the degree of cooperation willingness of each person may vary greatly, while our method can handle more random situations.

To further evaluate the performance of the proposed algorithm, we randomized the robot's initial position and, in addition, increased the randomness of the humans' initial positions. The resulting trajectory plots are shown in Figure 8. The results demonstrate that even under heightened randomness, the proposed DRL-RVO algorithm continues to achieve robust performance.

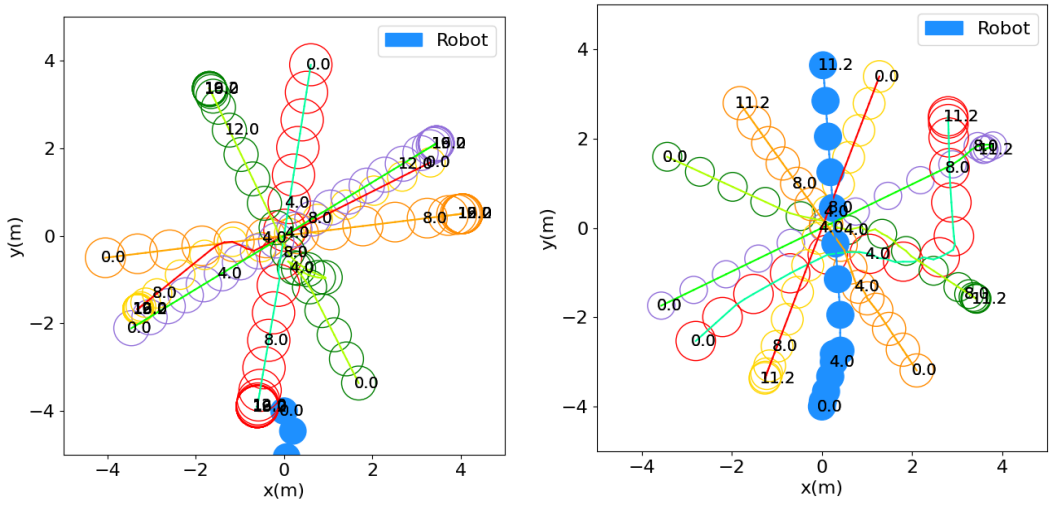


(a) ORCA (Collision) **(b)** CADRL

Figure 7. *Cont.*

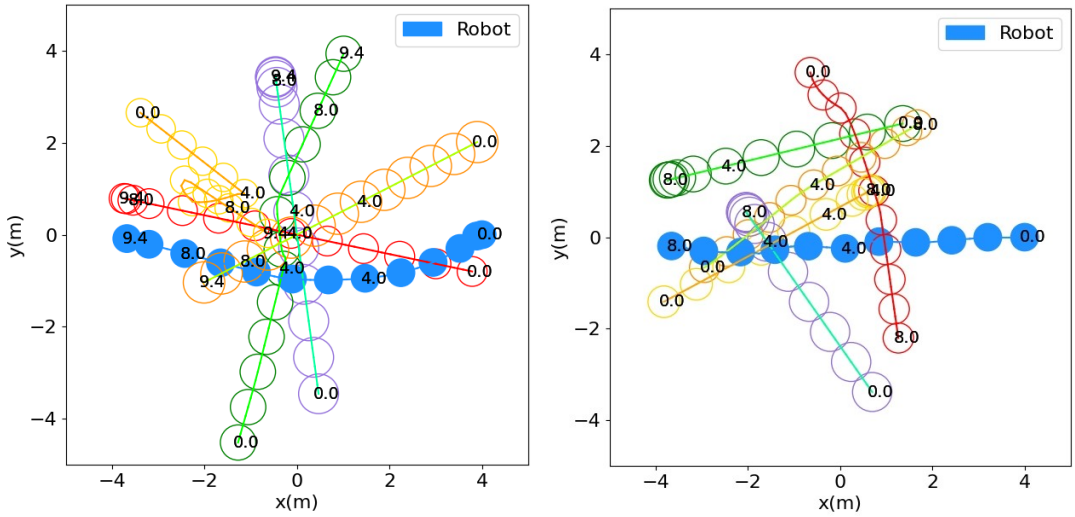


(c) LSTM-RL **(d)** SARL



(e) AEMCARL **(f)** DRL-RVO (Ours)

Figure 7. Trajectory comparison in an invisible test case. Circles are the positions of agents at the labeled times. ORCA directly causes a collision. CADRL, LSTM-RL, SARL, and AEMCARL all show avoidance and moving far away actions, but can not lead directly to the target point. Only our DRL-RVO can reach the target point as quickly as possible without collision.



(g) **(h)**

Figure 8. Trajectory of (**a**) Set random initial robot position and (**b**) Set more random humans' positions based on (**a**).

5.4. Ablation Studies

To further analyze the sensitivity of our model to environmental conditions and weighting strategies, we conducted two sets of ablation experiments.

5.4.1. Map Size

We varied the size of the environment, testing scales of 1/2, 2, 5, and 10 times the original setting, keeping other settings unchanged. This setup evaluates the robustness and adaptability of the model under different spatial densities. The results are shown in Table 3.

Table 3. DRL-RVO Performance under Different Environment Scales.

Map Scale	Success	Collision	Time
1/2	0.47 ± 0.031	0.51 ± 0.024	7.65 ± 0.214
1 (Origin)	0.95 ± 0.005	0.05 ± 0.005	11.18 ± 0.025
2	0.95 ± 0.019	0.05 ± 0.019	18.39 ± 0.269
5	0.97 ± 0.005	0.03 ± 0.005	42.08 ± 0.160
10	0.98 ± 0.000	0.02 ± 0.000	82.40 ± 0.091

Results show that as the environment becomes larger, the success rate slightly increases, while the navigation time significantly increases, reflecting the impact of environmental complexity on navigation efficiency.

5.4.2. Fixed α

To investigate the effect of dynamic weighting, we evaluated the pre-trained model with fixed robot weights of 0, 0.2, 0.5, 0.7, and 1. Interestingly, the success rate decreases when the weight is fixed at 0.7 or 1.0, while it achieves the highest value at 0.5. The comparison metrics are the same as above.

Table 4 summarizes the navigation performance under different fixed robot weights α. When $\alpha = 0$, the success rate drops to 0.01 and the collision rate rises to 0.99, indicating that the robot effectively ignores its own navigation objective. Increasing α to 0.2 significantly improves performance, and $\alpha = 0.5$ achieves the highest success rate with a low collision rate. The apparent advantage of $\alpha = 0.5$ could be attributed to statistical fluctuations in a limited set of scenarios and does not guarantee robustness across diverse or more complex environments. Interestingly, fixing α at 0.7 or 1.0 leads to a decrease in success rate, suggesting that excessive reliance on a high fixed weight reduces the robot's adaptability to dynamic environments. Navigation time remains relatively stable across different α values, except for $\alpha = 0$, where early collisions result in shorter completion time. Overall, these results highlight the importance of dynamic weighting, which balances strategy selection and ensures high success rates and safe navigation.

Table 4. Effect of Robot Weight α on Success Rate, Collision Rate, and Navigation Time.

The Weight α of Robot	Success	Collision	Time
$\alpha = 0.0$	0.01 ± 0.000	0.99 ± 0.000	8.2 ± 0.000
$\alpha = 0.2$	0.76 ± 0.010	0.23 ± 0.019	11.72 ± 0.065
$\alpha = 0.5$	0.95 ± 0.015	0.04 ± 0.005	11.33 ± 0.260
$\alpha = 0.7$	0.86 ± 0.005	0.14 ± 0.015	11.12 ± 0.033
$\alpha = 1.0$	0.89 ± 0.005	0.11 ± 0.005	11.09 ± 0.029
Dynamic α (Origin)	0.95 ± 0.005	0.05 ± 0.005	11.18 ± 0.025

6. Conclusions

This paper presents a novel DRL-RVO algorithm that integrates DRL with the traditional RVO method to achieve efficient collision avoidance in highly dynamic environments. Compared to state-of-the-art methods such as ORCA, CADRL, LSTM-RL, and SARL, the DRL-RVO algorithm dynamically adjusts the weight coefficient of RVO through learning, enabling real-time adaptation to environmental changes and significantly improving both avoidance success rates and task efficiency.

Experimental results demonstrate that the proposed DRL-RVO model not only enhances overall performance but also significantly accelerates training, achieving superior efficiency compared to conventional DRL baselines

in terms of success rate, task completion time, and training duration. Moreover, the proposed method effectively addresses the inherent limitations of RVO, which relies on fixed weight parameters, and mitigates the inefficiencies of pure DRL-based navigation approaches in dense crowd environments. In future work, we will explore the performance of our proposed method through real-world experiments.

Author Contributions: L.Yang.: conceptualization, methodology, software, data curation, writing—original draft preparation; L.Hu.: supervision, software, validation, writing—reviewing and editing. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies: No AI tools were utilized for this paper.

References

1. Borenstein, J.; Koren, Y. Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.* **1989**, *19*, 1179–1187.
2. Borenstein, J.; Koren, Y. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 572–577.
3. Borenstein, J.; Koren, Y.; et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Trans. Robot. Autom.* **1991**, *7*, 278–288.
4. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33.
5. Van den Berg, J.; Lin, M.; Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 1928–1935.
6. Van Den Berg, J.; Guy, S.J.; Lin, M.; et al. Reciprocal n-body collision avoidance. In Proceedings of the Robotics Research: The 14th International Symposium ISRR, Springer, Lucerne, Switzerland, 31 August–3 September 2009; pp. 3–19.
7. Snape, J.; Van Den Berg, J.; Guy, S.J.; et al. The hybrid reciprocal velocity obstacle. *IEEE Trans. Robot.* **2011**, *27*, 696–706.
8. Fong, T.; Nourbakhsh, I.; Dautenhahn, K. A survey of socially interactive robots. *Rob. Auton. Syst.* **2003**, *42*, 143–166.
9. Kruse, T.; Pandey, A.K.; Alami, R.; et al. Human-aware robot navigation: A survey. *Rob. Auton. Syst.* **2013**, *61*, 1726–1743.
10. Kuderer, M.; Kretzschmar, H.; Sprunk, C.; et al. Feature-Based Prediction of Trajectories for Socially Compliant Navigation. In *Robotics: Science and Systems VIII; MIT Press: Cambridge, MA, USA,* 2013.
11. Alahi, A.; Goel, K.; Ramanathan, V.; et al. Social LSTM: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971.
12. Altché, F.; de La Fortelle, A. An LSTM network for highway trajectory prediction. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 353–359.
13. Vemula, A.; Muelling, K.; Oh, J. Social attention: Modeling attention in human crowds. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 4601–4607.
14. Rudenko, A.; Palmieri, L.; Herman, M.; et al. Human motion trajectory prediction: A survey. *Int. J. Rob. Res.* **2020**, *39*, 895–935.
15. Bennewitz, M.; Burgard, W.; Cielniak, G.; et al. Learning motion patterns of people for compliant robot motion. *Int. J. Rob. Res.* **2005**, *24*, 31–48.
16. Trautman, P.; Krause, A. Unfreezing the robot: Navigation in dense, interacting crowds. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 797–803.
17. Fan, T.; Cheng, X.; Pan, J.; et al. Getting robots unfrozen and unlost in dense pedestrian crowds. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1178–1185.
18. Sathyamoorthy, A.J.; Patel, U.; Guan, T.; et al. Frozone: Freezing-free, pedestrian-friendly navigation in human crowds. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4352–4359.
19. Chen, Y.F.; Liu, M.; Everett, M.; et al. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 285–292.
20. Chen, Y.F.; Everett, M.; Liu, M.; et al. Socially aware motion planning with deep reinforcement learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1343–1350.
21. Long, P.; Fan, T.; Liao, X.; et al. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6252–6259.
22. Everett, M.; Chen, Y.F.; How, J.P. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3052–3059.

23. Chen, C.; Liu, Y.; Kreiss, S.; et al. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6015–6022.
24. Wang, S.; Gao, R.; Han, R.; et al. Adaptive environment modeling based reinforcement learning for collision avoidance in complex scenes. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 9011–9018.
25. Han, R.; Chen, S.; Wang, S.; et al. Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards. *IEEE Robot. Autom. Lett.* **2022**, *7*, 5896–5903.
26. Xie, Z.; Dames, P. DRL-VO: Learning to navigate through crowded dynamic scenes using velocity obstacles. *IEEE Trans. Robot.* **2023**, *39*, 2700–2719.
27. Chen, L.; Wang, Y.; Miao, Z.; et al. Reciprocal velocity obstacle spatial-temporal network for distributed multirobot navigation. *IEEE Trans. Ind. Electron.* **2024**, *71*, 14470–14480.
28. Ban, J.; Li, G. Training is execution: A reinforcement learning-based collision avoidance algorithm for volatile scenarios. *IEEE Access* **2024**, *12*, 116956–116967.
29. Fiorini, P.; Shiller, Z. Motion planning in dynamic environments using velocity obstacles. *Int. J. Rob. Res.* **1998**, *17*, 760–772.
30. Gopalakrishnan, B.; Singh, A.K.; Kaushik, M.; et al. PRVO: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1089–1096.
31. Fan, T.; Long, P.; Liu, W.; et al. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int. J. Rob. Res.* **2020**, *39*, 856–892.
32. Han, R.; Chen, S.; Hao, Q. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 448–454.