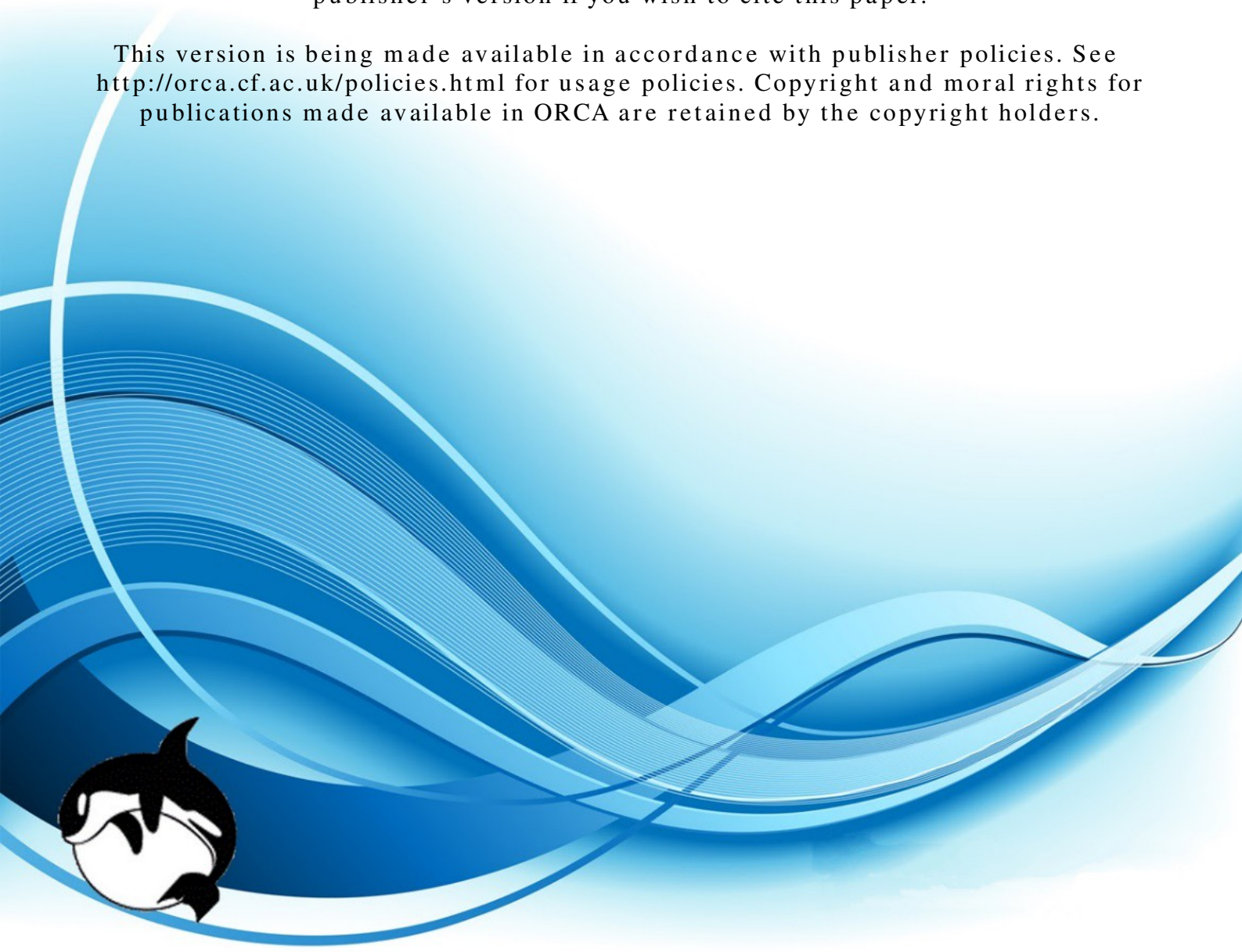


Deep Reinforcement learning with Multiple Unrelated Rewards for AGV Mapless Navigation

Boliang Cai, Changyun Wei and Ze Ji*

***Abstract*—Mapless navigation for Automated Guided Vehicles (AGV) via Deep Reinforcement Learning (DRL) algorithms has attracted significantly rising attention in recent years. Collision avoidance from dynamic obstacles in unstructured environments, such as pedestrians and other vehicles, is one of the key challenges for mapless navigation. Autonomous navigation requires a policy to make decisions to optimize the path distance towards the goal but also to reduce the probability of collisions with obstacles. Mostly, the reward for AGV navigation is calculated by combining multiple reward functions for different purposes, such as encouraging the robot to move towards the goal or avoiding collisions, as a state-conditioned function. The combined reward, however, may lead to biased behaviours due to the empirically chosen weights when multiple rewards are combined and dangerous situations are misjudged. Therefore, this paper proposes a learning-based method with multiple unrelated rewards, which represent the evaluation of different behaviours respectively. The policy network, named Multi-Feature Policy Gradients (MFPG), is conducted by two separate Q networks that are constructed by two individual rewards, corresponding to goal distance shortening and collision avoidance, respectively. In addition, we also propose an auto-tuning method, named Ada-MFPG, that allows the MFPG algorithm to automatically adjust the weights for the two separate policy gradients. For collision avoidance, we present a new social norm-oriented continuous biased reward for performing specific social norm so as to reduce the probabilities of AGV collisions. By adding an offset gain to one of the reward functions, vehicles conducted by the proposed algorithm exhibited the predetermined features. The work was tested in different simulation environments under multiple scenarios with a single robot or multiple robots. The proposed MFPG method is compared with standard Deep Deterministic Policy Gradient (DDPG), the modified DDPG, SAC and TD3 with a social norm mechanism. MFPG significantly increases the success rate in robot navigation tasks compared with the DDPG. Besides, among all the benchmarking algorithms, the MFPG-based algorithms have the optimal task completion duration and lower variance compared with the baselines. The work has also been tested on real robots. Experiments on the real robots demonstrate the viability of the trained model for the real world scenarios. The learned model can be used for multi-robot mapless navigation in complex environments, such as a warehouse, that need multi-robot cooperation.**

***Note to Practitioners*—Autonomous navigation for AGVs in complex and large-scale environments, such as factories and warehouses, is challenging. AGVs are usually centrally controlled and depend on reliable communications. However, centralized control is not always reliable due to poor signal strengths or crashes of the server, and hence unsuitable due to the requirements of accurate information of the dynamic environments and fast responses of decision making. Therefore, it is necessary for the vehicles to perform reliable decision making based on only onboard sensors and processors, for efficient and safe autonomous navigation. Existing methods, such as simultaneous localization and mapping (SLAM) and motion planning algorithms, have been widely used. However, they are neither flexible nor generalizable enough. This paper proposes a method for autonomous navigation based on reinforcement learning (RL), which allows vehicles to gain experience through cumulative rewards by continuously interacting with the environment. The RL-based controller is designed for optimising its performance in two independent aspects, namely collision avoidance and navigation, which are quantified as separate rewards. Instead of carefully hand-crafting a combined reward, our proposed approach trains the agent using the two rewards separately to obtain one optimal policy. It is clearly easier and more practical to design the individual rewards than manually combining them. Besides, the algorithm includes a mechanism for incorporating social norms to encourage the vehicles to follow the right-hand rule, such that they can avoid pedestrians or other vehicles in a socially acceptable manner. This is achieved by adding a continuous bias on the collision avoidance reward. Experiments using simulation environments and real robots suggest that the method is generalizable to multi-robot systems, while guaranteeing safety. In future research, we will focus on incorporating uncertainties of sensor readings for safe and reliable autonomous navigation.**

***Index Terms*—Collision Avoidance, Deep Reinforcement Learning, Mapless Navigation, Motion Planning, Multiagent Systems**

Boliang Cai is with the College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China and the School of Engineering, Cardiff University, Cardiff, United Kingdom (e-mail:caib4@cardiff.ac.uk)

Changyun Wei is with the College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China (e-mail: c.wei@hhu.edu.cn.)

Ze Ji (corresponding author) is with the School of Engineering, Cardiff University, Cardiff, United Kingdom (e-mail: jiz1@cardiff.ac.uk).

This work was supported in part by the National Natural Science Foundation of China under Grant 61703138, China Scholarship Council under Grant CSC202106710026, and in part by the Fundamental Research Funds for the Central Universities under Grant B200202224 and B200204036.

I. Introduction

A Major breakthrough in Reinforcement Learning (RL), in recent years, is the combination of artificial neural networks (ANN) with RL algorithms, which fostered the development of neural controlled RL algorithms, such as Deep Q-Networks (DQN), Actor Critic (AC) etc. Deep Reinforcement Learning (DRL) algorithms have gained remarkable achievements for complex tasks, such as the Go [1], video games [2] [3], robotics [4] [5] and so on. Despite the great success of DRL in different domains, DRL faces many challenges related to its efficiency and practicality. Many new methods have emerged to tackle various challenges related to the efficiency or practicality issues, such as Sim-To-Real to transfer simulation-trained agents to the real-world environments [6] [7]. On the other hand, many new algorithms have emerged in recent years, such as DDPG [6], Soft Actor Critic (SAC) [8], and Proximal Policy Optimization (PPO) [9], have attracted rising attention for solving robot control

problems. However, when a task is composed of multiple sub-tasks, the reward definitions are usually constructed by simply combining rewards of each individual state from the corresponding sub-tasks. Decoupled sub-tasks could have unrelated rewards, meaning that a simple combination of rewards can result in unbalanced weighted rewards for individual sub-tasks, causing the policy to be trained inadequately.

From the robotics perspective, mapless navigation and collision avoidance for AGVs are widely discussed in the robot control field. Classical methods for robot autonomous navigation are usually map-based algorithms that may require pre-built maps using techniques such as simultaneous localization and mapping (SLAM) [10] and path planning will be carried out using the map. The emergence of DRL algorithms, however, provides another avenue for autonomous AGV navigation in unknown scenes that allows a robot to plan its path without a pre-built map. DRL-based mapless navigation algorithms [8] [11] [12] are proposed with stronger adaptability and robustness to unknown environments.

For RL algorithms, a reward is often defined in a simplified manner with empirically determined constant values, meaning that the reward needs to be pre-designed and regulated with fine-tuned parameters. The rewards for AGV navigation tasks are usually conditioned by two environmental states, the distance of the agent towards the target and the surroundings of the agent, which are used to construct two individual rewards, namely the distance-shortening reward and the collision avoidance reward in our work. Methods such as [11] [12] [8] combine the rewards together as a single value function. However, the irreconcilable criteria or the difference between the two independent rewards make the simple combination unable to represent the optimal policy value of the agent. Therefore, a multi-feature-based policy gradient optimization algorithm considering multiple task features is proposed in this paper. Considering that the reward for each feature is dealt with separately, the value of each action of each sub-task also needs to be calculated individually. In the algorithm, multiple critic networks are used to calculate Q values using separate rewards.

On the other hand, inspired by traffic flow control in human societies, the collision avoidance reward function in our work includes a social norm element that is introduced to encourage robots to keep on one side of the path when avoiding collisions with other robots, humans, or dynamic objects, i.e. the right hand side in our case. In addition, we also introduce a modified Prioritized Experience Replay (PER) algorithm to allow efficient transition sampling in our case with two separate rewards, as mentioned above.

Our contributions are summarized as follows:

- We introduce a DRL-based approach for mapless navigation, known as Multi-Featured Policy Gradient (MFPG) that incorporates unrelated reward signals, corresponding to multiple objectives in terms of path length and safety. These rewards serve the purpose of approximating distinct state-action values. By optimizing the policy based on these values, we ensure the above-mentioned objectives of reachability and safety in AGV navigation.
- An adaptive approach is introduced to balance the impact of each state-action value in relation to the actor. To achieve this, the standard deviation of each state-action value is used as the guiding factor for determining the weight applied to the distinct policy gradients.
- A PER is used to accelerate the training in the paper, however, due to the irreconcilable criteria of the two critic networks, we extend the Prioritized Experience Replay (PER) algorithm by incorporating the approximation errors of both state-action values as the priority indicator.
- Built upon the MFPG framework, we develop an innovative technique for visualizing the preferences established by the reinforcement learning algorithm. We visualize the state-action distribution of each individual Q neural network. As a result, the preferences of the two Q neural networks can be effectively visualized, thereby enhancing the interpretability of the MFPG approach.

The remainder of this article is organized as follows. Section II introduces related work. Section III describes the working principle of our proposed MFPG algorithm. The mapless navigation and collision avoidance problems with MFPG are detailed in Section IV. Last, the experiments and discussions are provided in Section V, followed by the conclusion in Section VI.

II. Related Works

Control methods for multi-vehicle navigation tasks can be categorized into two main directions: centralized methods and decentralized methods [13]. Centralized methods for vehicle navigation have been widely applied in formation control [14], goods transportation [15] etc. Most centralized methods assume that all robots in the team can access the central server at runtime with negligible delays [16]. Therefore, the connection between the server and the robots guarantees the safety and optimality of the navigation policy [17]. However, situations of robots in the real world are more dynamic and unpredictable than the assumptions [13]. Such differences make it hard for centralized robots and servers to be deployed in large-scale fields and safety-critical scenarios, such as warehouses or many public areas. One disadvantage of centralized methods is that the high cost and complexity of such real-time communication systems will greatly impede the deployment of such systems in the aforementioned scenarios such as warehouses [13]. Delays of signals could result in failures and accidents in the working space that will be critical. Besides, such methods are highly dependent on the computing power of the central server that will drastically increase as the number of vehicles increases [17]. Therefore, the scalability of such systems will be limited, making it impractical for system upgrades with an increased number of vehicles. Centralized methods need to be fully aware of the surroundings for decision-making for all vehicles and, hence, will be more complicated to implement.

Decentralized methods, however, allow robots to make decisions based on local measurements of their surroundings independently, meaning that the robots are less dependent on the communication system, hence more robust and responsive under unreliable communication [13]. Many works, such as

Social Force [18] and Artificial Potential Field (APF) [19], have been introduced in a decentralized manner. Inspired by the behaviours of charged particles in electric fields, these methods model the interaction between robots and obstacles as the function of the distance from the controlled robot to its surroundings [20]. Besides, fuzzy logic combined with classical control theory emerges as an approach for Unmanned Aerial Vehicles (UAVs) guidance and navigation with limited computational resources [21]. These methods are efficient and simple for navigating single vehicles. However, the methods are not suitable for situations involving confrontation with multiple vehicles, and collision avoidance between vehicles is not considered in the policy. Besides, due to the simplicity of these algorithms, the parameters usually need to be tuned empirically to meet the needs of specific circumstances.

Therefore, decentralized MPC methods, such as [22] [4], are employed. In [22], a distributed path planning system is introduced for planning near-optimal solutions. The method can prevent collisions robustly and allow altruistic behaviors between vehicles that monotonically decrease the global cost function [23]. Another representative method designed for multiple vehicles is the Optimal Reciprocal Collision Avoidance (ORCA) and its variations [24]. The methods are proven to be efficient for multi-robot systems with millisecond-timescale control in simulation environments. However, such methods aim to adjust their parameters in a heuristic way according to the surrounding states of the robots. The adjustment procedure could not be completed in real time to produce an optimal policy when encountering a complex scenario. Besides, the assumptions of the approaches that fully sensor-detectable shapes, speeds, as well as the positions of other robots, make it difficult to practically apply the method in a real-world environment. In order to enhance the performance of the model in the real world, modifications are proposed to enhance the inter-communication between robots [25]. In the modified approaches, the controlled robot receives the state information from the robots nearby via a communication protocol for better state sharing [26]. However, the algorithms also encountered problems similar to those in the centralized methods. Particularly, the reliability of communication can affect the performance of the algorithm significantly.

In recent years, applications of Deep Neural Networks (DNN) have shown great potential in many challenging domains. The ability of DNNs to construct nonlinear mapping relationships in a high-dimensional space makes it suitable to be applied for robot control problems. Existing navigation algorithms based on the DNNs can be categorized into two parts: imitation learning (IL) and DRL. Imitation learning methods allow learning end-to-end mapping of policy [27] [28] in a supervised manner. The IL methods often receive sensor representations, such as Lidar readings and raw images, as the state input along with a label, which is usually the action corresponding to the state input. The sensor representations and the corresponding actions can be obtained from the demonstrations of the experts or from the simulation scenes. Besides, it is worth noting that the trained model is conditioned by the quality of the dataset [29]. Owing to that the trained model only requires the representation input, the model can be applied in both single and multi-agent tasks. In [27], video images are used as the input for the neural network, which generates actions for the robot to execute, for the task of lane-following. A conditional imitation learning approach is proposed to divide the neural networks into several branches for separate tasks [28]. These methods are simple, but cannot handle complex traffic situations, such as multi-lane and crossing, resulting in inadequate mapping from the high dimensional state input to the actions. Therefore, semantic abstraction IL was proposed, taking semantic images, such as High-Definition (HD) maps [30] and bridgeye views (BEVs) [31] as the input. Compared with the redundant raw image inputs, the semantic images provide a concise and informative abstraction of perceptual results with a better environment consistency. The properties of the semantic abstraction make the IL more generalizable and efficient, because the semantic abstraction diminishes state space and helps the DNNs learn meaningful context cues [5]. Overall, the IL methods perform well for states that are included in the training data but generalize poorly to unknown states. This is referred to as the distribution mismatch [32].

RL algorithms, on the other hand, focus on interaction between the environment and the policy, and aim to improve the policy according to the environmental feedback. DRL algorithms can be categorized into two main approaches: value-based and policy-based methods. The value-based methods, such as DQN, DDQN etc., have been widely examined in robot navigation tasks [5] [33] [34].

A deep Q-learning with graph attention representation is introduced to address the safety aspect of the lane-following task [5]. Graph attention layers and BEVs are adopted to obtain a compact state representation of the DQN algorithms for the lane-following tasks. The policy is defined to control the linear velocity of the vehicle in the discrete action space, while the steering control is performed with a typical PID controller. In [33], AUV navigation uses the earth's geomagnetic information for state representation, where a set of discrete heading angles are selected as the action space. Dueling double deep Q-learning with PER is used for AGV path planning in [34], in which an action set composed of 3 linear and 5 angular velocity values are defined. Although the value-based methods aforementioned are effective and simple, they neglect the fact that robot control in the real world is based on the continuous action space.

Different from value-based methods, policy-based methods generate continuous action commands, which allow more flexible manoeuvring in complex environments [35]. Policy-based DRL has been investigated and deployed for navigation tasks [8], [12], [20], [36]–[38]. An end-to-end DDPG navigation method proposed in [36] defines a distance-conditioned reward function and receives the position of the controlled robot and its destination as the input state. The policy generates commands for the six thrusters directly. However, further analysis of the reward is required to obtain a better navigation performance. An asynchronous advantage actor-critic structure is adopted in [37] for the exploring task in an underwater scenario. The paper proposes a multiple policy network fusion method to facilitate the AUV's exploration task. In this context,

the upper policy receives input from sonar sensor readings and determines the activation of specific sub-policies. Once activated, the policy generates necessary actions for executing navigation or avoidance manoeuvres independently. In [20], a multiplicative controller fusion-based method is introduced, integrating an APF-based method with SAC for addressing robot navigation tasks. In the method, an end-to-end actor-critic structure is adopted as one of the controllers, which takes the laser sensor readings, historical action information and the destination of the robot as input. The APF controller receives laser readings and the robot's destination for computing navigation commands. This algorithm computes the desired actions based on the outputs from both the APF module and the SAC agent. However, the method is limited, when the APF and policy controllers generate conflicting commands with separate lower variances. The above methods mainly focus on single robot navigation tasks in static scenes, which, however, cannot be directly deployed for decentralized robot navigation tasks, which present higher uncertainty and complexity during interactions between robots and obstacles.

One of the actor-critic methods applied in decentralized robot navigation is SA-CADRL [12] that considers a social norm punishment in the reward function for avoiding others. The purpose of the social norm punishment is to form social rules, such as the left- or right-hand rule, in the algorithm. In a similar method, GA3C-CADRL [39], states of other robots observed by the controlled robots are transmitted to a Long Short-Term Memory (LSTM) unit to extract the surrounding features. The features are extracted with the state of the controlled robots as the input of the policy network [12]. It is worth mentioning that both the above methods do not require inter-communication between vehicles. They present similar approaches for solving the vehicle confrontation problems. A similar pattern has also been observed in [40], wherein they employ an LSTM module to capture the dynamic properties of UAV, in conjunction with multiple critic networks. SA-CADRL [12] forms a common knowledge base between the vehicles that similar action trends will be taken to avoid collisions, while GA3C-CADRL [39] focuses on forecasting motions of other vehicles in complex dynamic situations. The Control Barrier Function (CBF) is employed in [41] as the task indicator and the work applies the mixed Nash equilibrium of the game theory in solving the deadlock problems of UAVs under the transmission obstruction scenarios. In our work, we adopt the idea of the SA-CADRL [12] for collision avoidance between vehicles and propose a continuous social norm bias to control the behaviors of social norms.

On the other hand, the reward functions in most previous works are defined as a single value function, conditioned by the state of the vehicle. However, in safety-critical tasks, a single reward function for the policy could not be sufficient to evaluate the quality of the decision. Therefore, several research works focused on this aspect and proposed methods from the reward-oriented perspective.

For example, Zhang [11] defined the reward by combining $r_{collision}$, r_{target} and r_{normal} as a single value function, where r_{normal} is the distance difference between the robot and its destination at two consecutive time steps, r_{target} represents the navigation ability of the robot but neglects the surroundings of the robot, and $r_{collision}$ is the punishment if encountering collisions with obstacles. Similarly, Chen [12] defined the reward as the social norm reward r_{norm} and collision avoidance reward r_{col}. A safety navigation tasks are framed in the context of a Constrained Markov Decision Process (CMDP) and penalty indicators are incorporated into the reward function in [42]. The rewards of these approaches are simply combined by adding them together. However, the works are not sufficient using a rough single reward value to indicate the performance for each action taken by the vehicles. Therefore, the task in our method is decomposed into two sub-tasks, where each sub-task is evaluated by a different reward value separately.

III. Multi-Feature Policy Gradients

A. Preliminaries

RL problems can be described as the discrete-time Markov Decision Processes (MDP), formalized by a set of states $\mathbf{S}$, a set of actions $\mathbf{A}$. An MDP state transition at time t can be defined as:

$$P_t = (s_t, a_t, r_{t+1}(s_t, a_t), s_{t+1}). \tag{1}$$

where $s_t, s_{t+1} \in S$ represent the states at time step t and $t+1$, $a_t \in \mathbf{A}$ is the action of state s_t and $r_t(s_t, a_t)$ is the reward of a_t in s_t at the time step t.

RL algorithms can be broadly categorized into value-based and policy-based solutions. The most well-known value-based algorithm is Q-learning, which maps actions at given states with corresponding values in a tabular manner. To alleviate the difficulties of RL algorithms in high dimensional space, researchers introduced the ANN that can be deployed to approximate the values and relationships between states and actions. The introduction of ANN significantly reduces the volume of the Q table as used in classic value-based RL. Policy-based methods use the policy gradient method for function approximation [43] that can model and optimize policies, hence mapping the relationship directly between states and actions in the continuous space. In this approach, a policy is defined as a mapping from states to actions, as follows:

$$\pi_\theta : \mathbf{S} \mapsto \mathbf{A}, \tag{2}$$

where θ represents the parameter vector of the policy network π_θ, $\mathbf{S}$ is the states and $\mathbf{A}$ is the mapped actions. An action a_t at the time-step t can be calculated from the policy π_θ with a given state s_t:

$$a_t = \pi_\theta(s_t) \tag{3}$$

In an MDP, the probability of that agent transferred from s_1 to s_T can be described as:

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} (p_{\pi_\theta}(a_t|s_t) p(s_{t+1}|s_t, a_t)), \tag{4}$$

in which, $\tau = \{s_1, a_1, \ldots, a_{T-1}, s_T\}$ is a specific trajectory of a training episode. $p_{\pi_\theta}(a_t|s_t)$ is the probability that policy π_θ chooses action a_t given state s_t. $p(s_{t+1}|s_t, a_t)$ is the

probability of reaching s_{t+1} after agent takes the action a_t at state s_t. In order to measure the value of action a_t at state s_t that contributes to the final results of each trajectory quantitatively, an expected discounted reward is introduced to calculate the value of each action a_t at state s_t. For each trajectory, the expected discounted reward can be described as:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau\sim p_\theta(\tau)}[\sum_{t}^{T}\gamma^{T-t}r_t(s_t,a_t)] \\ &= \int \pi_\theta(\tau)Q_{\pi_\theta}(\tau)\mathrm{d}\tau \\ &\approx \frac{1}{|N|}\sum_{i\in N}\sum_{t}^{T}\gamma^{T-t}r(s_{i,t},a_{i,t}) \end{aligned} \tag{5}$$

where N is the size of the sampled experience replay buffer, and $Q_{\pi_\theta}(\tau) = \sum_t \gamma^t r_t(s_t, a_t)$ is the Q value of each action a_t at state s_t.

Deep Deterministic Policy Gradient (DDPG) is a model-free algorithm that constructs the policy network, which maps from the state space to the actions of the agent. The policy optimizes its parameters according to the gradient generated from the Q-value networks.

A neural network is used for storing the Q-values corresponding to each agent state, formulated as below:

$$C_\psi : \mathbf{S}\times\mathbf{A}\mapsto Q(s,a), \tag{6}$$

where ψ represents the parameters of the critic network. The critic network updates its parameters by reducing its loss value, defined as:

$$L_\psi(P_t) = \mathbb{E}_{P_t}[Q^*(s_t,a_t) - Q(s_t,a_t)]^2, \tag{7}$$

where, $C^*_{\psi^*} = Q^*(s,a)$ is the target Q value can be calculated using a K-step-constant critic network, whose parameter ψ^* is copied from C_ψ. $Q^*(s,a)$ is the target Q value calculated by the following:

$$\begin{aligned} Q^*(s_t,a_t) &= \mathbb{E}_{P_t}[r(s_t,a_t) + \gamma\max_{a'}Q'(s_{t+1},a)] \\ &= \mathbb{E}_{P_t}[r(s_t,a_t) + \gamma Q'(s_{t+1},a'(s_{t+1}))], \end{aligned} \tag{8}$$

where Q' is a stable Q value network used to calculate target Q values, and parameters in Q' are copied from the Q network after certain training steps. Similarly, a' represents a stable policy network for assistance of calculating target Q values. Parameters in a' are updated by copying the policy network a periodically.

As the training goes on, $J(\theta)$ will gradually converge to stable values. It is expected that a better policy will get a higher expected reward. Therefore, the policy gradient methods that optimize parameters θ of policy π_θ for gaining a higher expected reward can be calculated using:

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_\tau \nabla_\theta\pi_\theta(\tau)Q(\tau)\mathrm{d}\tau \\ &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \mathrm{log}\pi_\theta(\tau)Q(\tau)] \end{aligned} \tag{9}$$

According to Equation 5 and Equation 9, the policy gradient is calculated using:

$$\nabla_\theta J(\theta) \approx \frac{1}{|N|}\sum_{i\in N}(\sum_{t=1}^{T}\nabla_\theta \mathrm{log}\pi_\theta(a_{i,t}|s_{i,t})\nabla_{\pi_\theta}Q(s_{i,t},a_{i,t})) \tag{10}$$

After calculating the gradient, the parameters are updated as below:

$$\theta' \leftarrow \theta + \lambda\nabla_\theta J(\theta). \tag{11}$$

where θ' represents the updated parameters, and λ is the learning rate for the policy update procedure.

B. Gradient Calculation for Multiple Uncombinable Rewards

In this section, we introduce our proposed work, named Multi-Feature Policy Gradient (MFPG). Rewards of mapless navigation, denoted by $r^{Nav} \in R^{Nav}$, and that of collision avoidance, denoted by $r^{CA} \in R^{CA}$, are considered separately in this work. Here, the MDP state transition is reshaped as:

$$P'_t = (s_t, a_t, r^{CA}_{t+1}, r^{Nav}_{t+1}, s_{t+1}). \tag{12}$$

Considering the uncombinability characteristic of the two rewards, the policy π_θ needs to be optimized using gradients constructed from two separate Q networks. Two policy gradients are used to optimize the policy π_θ that can be calculated as described in Section III-A.

The gradients constructed from r^{CA} and r^{Nav} are denoted by $\nabla_\theta J^{CA}(\theta)$ and $\nabla_\theta J^{Nav}(\theta)$:

$$\begin{aligned} \nabla_\theta J^{CA}(\theta) &= \nabla_\theta J(\theta)\Big|_{Q(s_t,a_t)=Q^*_{CA}(s_t,a_t)} \\ \nabla_\theta J^{Nav}(\theta) &= \nabla_\theta J(\theta)\Big|_{Q(s_t,a_t)=Q^*_{Nav}(s_t,a_t)} \end{aligned} \tag{13}$$

After policy gradients are calculated, θ is updated using:

$$\theta' = \theta + \lambda\Phi\cdot[\nabla_\theta J^{Nav}(\theta), \nabla_\theta J^{CA}(\theta)]^{\mathrm{T}}, \tag{14}$$

where $\Phi = [\phi^{Nav}, \phi^{CA}]$ represents the weights of both featured policy gradients that can adjust the performance of the model by changing the value of each individual weight. The main procedure can be seen in Algorithm 1. A brief flow chart is shown in Figure 1.

Here we provide the proof of the convergence for the MFPG. Since MFPG is derived from the DDPG [6], this paper only focuses on the equality between the gradient calculated from the original value network and the gradients from the two value networks in MFPG.

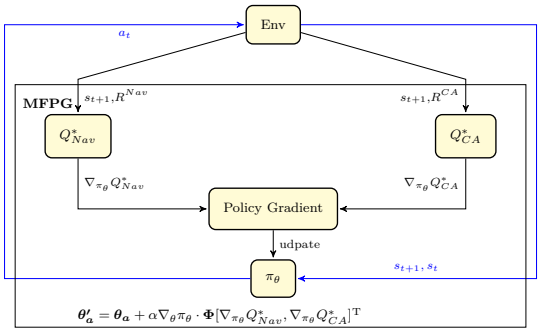


Fig. 1: Flow chart of Multi-Feature Policy Gradients.

Convergence proof for MFPG. For a single Q value function Q and two reward R^{Nav} and R^{CA}:

$$\begin{aligned} Q &= \sum_{i=t}^{T} \gamma^{i-t} R(s_t, a_t) \\ &= \sum_{i=t}^{T} \gamma^{i-t} (\phi^{CA} R^{CA}(s_t, a_t) + (\phi^{Nav} R^{Nav}(s_t, a_t)) \\ &= \phi^{CA} \sum_{i=t}^{T} \gamma^{i-t} R^{CA}(s_t, a_t) + \phi^{Nav} \sum_{i=t}^{T} \gamma^{i-t} R^{Nav}(s_t, a_t) \\ &= \phi^{Nav} Q^{Nav} + \phi^{CA} Q^{CA} \end{aligned} \tag{15}$$

It can be induced from Equation 15 that the Q value function calculated from the linear combination of the two rewards is identical to the linear combination of the two reward-constructed Q value functions. Therefore, the gradient calculated from the Q value function Q follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_\tau \nabla_\theta \pi_\theta(\tau) Q(\tau) \mathrm{d}\tau \\ &= \int_\tau \nabla_\theta \pi_\theta(\tau) (\phi^{CA} Q^{CA}(\tau) + \phi^{Nav} Q^{Nav}(\tau)) \mathrm{d}\tau \\ &= \phi^{CA} \nabla_\theta J^{CA}(\theta) + \phi^{Nav} \nabla_\theta J^{Nav}(\theta) \end{aligned} \tag{16}$$

Equation 16 meaning that the linear combination of the gradients $[\nabla_\theta J^{CA}(\theta), \nabla_\theta J^{Nav}(\theta)]$ is in coincidence with the gradient calculated from Q. The convergence proof for DDPG using a single gradient has been proposed in [6]. Therefore, the convergence ability of the MFPG is proved. □

C. Adaptive Policy Weight Calculation

In this section, we expand upon the proposed framework by introducing an adaptive gradient weight adjustment method, replacing the previous heuristic approach. This adaptive method is based on the assumption that the standard deviation of the separate state-action values is scale equivalent, i.e.:

$$\iota = \frac{\sigma(Q^*_{Nav}(s, \pi(s|\theta^*))}{\sigma(Q^*_{CA}(s, \pi(s|\theta^*))} \approx 1, \forall s \in \mathcal{S} \tag{17}$$

Here, $\sigma(\cdot)$ represents the standard deviation of $(\cdot)$. The parameter ι is the scale ratio of two state-action values, which are obtained from the target critic neural networks, indicating to what extent one critic influences the task policy in comparison to the other one. Equation 17 reveals two properties of the reward values. Firstly, the ratio ι between the two standard deviations is set approximately to 1, which implies that both rewards have equal contributions to the policy. However, it is worth noting that ι can also be set to any preferred value to obtain different behaviours. Secondly, to ensure training stability, the reward values should have a similar distribution across the entire state-action space, i.e. ι should remain

Algorithm 1: Adaptive Multi-Feature Policy Gradients

Input: Environment in and out
Output: Policy π_θ

Initialization Policy network π_θ, Critic networks $C_{\psi_{CA}}$ and $C_{\psi_{Nav}}$.;
Initialize the target networks π_{θ^*}, $C^*_{\psi^*_{CA}}$ and $C^*_{\psi^*_{Nav}}$.;
Initialize MDP transition set M with max volume m;
Initialize episode $k = 0$, max episode k_m , update counters for actor K_a, critic K_c;
Initialize scale weight: $\mathbf{w}_n = \exp_n[-1:0]$ and scale ratio queue $\mathbf{q}_n = [1 \ldots, 1]_n$;
while $k \leq k_m$ **do**
  Get current state s_t;
  Select action $a_t = \pi_\theta(s_t)$;
  Apply action to environment and get next state s_{t+1} and next reward r_{t+1}.;
  Store new MDP state transition to experience buffer $M = M \cup P'_t$;
  if $|M| \geq m$ **then**
    Sample experience replay buffer N using prioritized experience replay described in Section III-D ;
    Update critic networks $C_{\psi^{CA}}$ and $C_{\psi^{Nav}}$ by minimizing $L(\psi_{CA})$ and $L(\psi_{Nav})$;
    Calculating batch Q values $Q^*_{\psi_{CA}}$,$Q^*_{\psi_{NAV}}$ according to the sampled states and estimating Q value scale Equation 17;
    Update scale ratio queue by $\iota \mapsto \mathbf{q}_n$ and estimate the historical scale ratio η by Equation 18;
    Estimate the gradient weight $\Phi = [1/\sqrt{\eta}, \sqrt{\eta}]$;
    Calculate $\nabla_\theta J^{CA}$ and $\nabla_\theta J^{Nav}$ using Equation 13;
    $\theta' \leftarrow \theta + \lambda \Phi \cdot [\nabla_\theta J^{Nav}(\theta), \nabla_\theta J^{CA}(\theta)]^{\mathrm{T}}$;
  end
  if $k // K_a \in \mathbb{Z}^+$ **then**
    $\theta^* \leftarrow \theta$;
  end
  if $k // K_c \in \mathbb{Z}^+$ **then**
    $\psi^*_{Nav} \leftarrow \psi_{Nav}$;
    $\psi^*_{CA} \leftarrow \psi_{CA}$;
  end
  $k = k + 1$
end
return π_θ

relatively stable.

Hence, we apply historical weighting factors to $\mathbf{q_n} = [\iota_{-(n-1)}, \ldots, \iota_0]$, which is a queue storing the past n scale ratios, following a FIFO manner. The weighting factors $\mathbf{w}_n$ are the exponential of n points equidistant sampled from $[-1, 0]$, such that the weight for the nearest scale ratio ι_0 receives the highest weight of 1 and the earlier weights will undergo an exponential decrease. Specifically, the scale ratio is calculated as follows:

$$\eta = \frac{\mathbf{q_n} \cdot (\mathbf{w}_n)}{\Sigma_n((\mathbf{w}_n))}, \tag{18}$$

At the initial stage, elements inside the queue are all set to 1. Finally, the gradient weights are obtained, as follows:

$$\Phi = [1/\sqrt{\eta}, \sqrt{\eta}]. \tag{19}$$

To summarize, calculating the adaptive gradient weights requires two primary steps. Firstly, Equation 17 estimates the scale ratio of the two state-action values from a statistical perspective. The assumption $\iota \approx 1$ implies an ideal situation where both state-action values leverage the same level of influence on the policy. Secondly, the gradient weight estimation integrated with the historical scale ratio is employed to guarantee stability, as demonstrated in Equation 18. Therefore, the gradient weights Φ in Equation 19 are derived from multiple training sample batches. By following the previous steps, the adaptive gradient weight can keep the impact of separate navigation and collision avoidance state-action values on policy balanced.

D. Modified Prioritized Experience Replay

Usually, experience transitions are sampled from a memory buffer uniformly without considering their significance, hence presenting low efficiency. To address this limit, the PER is proposed to prioritize experience, so that important transitions can be replayed more frequently [44]. Conventionally, the PER algorithm was used as a sampling method aiming to improve the effectiveness in reducing the loss of the critic network in the model [45]. However, the circumstance in our approach is different from the condition proposed as with the original situation of PER. Because of the irreconcilable criteria of the two critic networks, it is difficult to compute the importance sampling weight and sampling probability. In this work, we address this problem by introducing some changes to allow the combination of two different loss functions of both critic networks in the PER framework.

When an MDP transition is selected for training, the Temporal Difference (TD) error of the transition can be calculated using Equation 7, which will be recorded as the importance sampling weight of the transition. The probability of transition $q_i \in M$ to be sampled can be calculated according to:

$$p(q_i) = \frac{L_\psi(q_i)^\alpha}{\Sigma_{q \in M} L_\psi(q)^\alpha}. \tag{20}$$

In Equation 20, α is an indicator of the priority for the corresponding transition, where uniform sampling is applied if $\alpha = 0$.

Here, a new method is proposed to calculate the sampling probability of the MDP transitions:

$$p(q_i) = \frac{(\zeta_1 L_\psi^{Nav}(q_i) + \zeta_2 L_\psi^{CA}(q_i))^\alpha}{\Sigma_{q \in M}(\zeta_1 L_\psi^{Nav}(q) + \zeta_2 L_\psi^{CA}(q))^\alpha}. \tag{21}$$

where ζ_1 and ζ_2 are the importance weights for both loss components. The values of the two weights need to satisfy $\zeta_1 + \zeta_2 = 1$, where, in this paper, ζ_1 and ζ_2 are set to 0.5. When a new transition q_i is added into the experience buffer M, its default loss value $(L_\psi^{Nav}(q_i) + L_\psi^{CA}(q_i))^\alpha$ will be set to 1.

Conventionally, the PER algorithm adopts the approximation error as the sampling priority for each sample. Here, we modify the indicator to be the linear combination of the training loss of both the critic networks. This means the accuracies of both collision avoidance and navigation state-action values of all the samples are taken into consideration. Hence, the modified PER can incorporate the sampling priority of the samples from both collision avoidance and navigation standpoints.

When the sampling of the buffer is completed, the importance sampling weight can be calculated using:

$$w_i = (\frac{1}{|N|} \cdot \frac{1}{p(q_i)})^\beta, \tag{22}$$

in which β is an annealing bias that ranges from 0.4 to 1. The detailed steps for updating the Critic networks are described in Algorithm 2:

Algorithm 2: Updating Approach for Critic Networks

Input: Critic $C_{\psi_{Nav}}$,$C_{\psi_{CA}}$, experience buffer M
Output: Updated Critic network $C_{\psi'_{Nav}}$,$C_{\psi'_{CA}}$

Initialize accumulated weighted changesΔ_{Nav} , Δ_{CA};
Sampling experience transitions from M using Equation 21;
foreach $q_i \in N$ **do**
  Calculate w_i using Equation 22;
  Compute loss of critic neural networks $L_\psi^{Nav}(q_i)$ and $L_\psi^{CA}(q_i)$;
  Calculate weight importance sampling weight w_i using Equation 22;
  Update accumulated weighted changes:
  $\Delta_{Nav} = \Delta_{Nav} + w_i \cdot L_\psi^{Nav}(q_i) \nabla_{\psi_{Nav}} Q_{Nav}(s, a)$;
  $\Delta_{CA} = \Delta_{CA} + w_i \cdot L_\psi^{CA}(q_i) \nabla_{\psi_{CA}} Q_{CA}(s, a)$;
end
update parameters:
$\psi'_{Nav} \leftarrow \psi_{Nav} + \eta \Delta_{Nav}$;
$\psi'_{CA} \leftarrow \psi_{CA} + \eta \Delta_{CA}$;
return $C_{\psi'_{Nav}}$,$C_{\psi'_{CA}}$

When a new transition is added to the replay buffer, the transitions that existed in the buffer with the minimum sampling probability will be replaced by the new transition:

$$q_{min} \Leftarrow q_{new}, \tag{23}$$

in which, the operator $\Leftarrow$ denotes that the transition on the left of the operator will be replaced by that on the right and the probability of the new one $p(q_{new})$ will be reset to 1.

IV. Mapless Navigation and Collision Avoidance

A. Parameterization

This subsection describes the environment configuration for mapless navigation in this work. Figure 2 depicts the system overview of vehicle navigation with collision avoidance. The black arrows depict the positive and negative orientations in the polar coordinate space. Following the right-handed rule, "P" represents the positive direction of rotation, while "N" represents the negative direction. The red lines represent the laser beams of a 2-D planar laser range finder. The purple triangle is the position of the target. The arrow on the green line represents the index of the laser beam on the sensor that starts from 0 at the beginning and ends at 128.

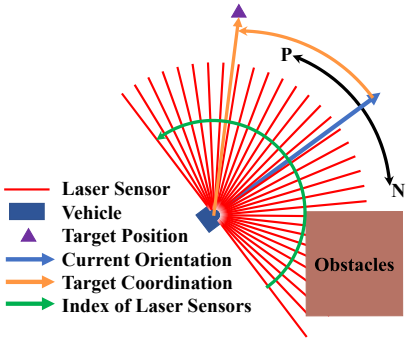


Fig. 2: Detail of vehicle navigation and collision avoidance.

The state of a vehicle can be represented by the following components:

- Laser range finder readings $S_{laser} = [l_1 \dots l_{128}]$ that represent the surrounding occupancy information of the vehicle. In this paper, 128 laser beams are used for sensing obstacles around the vehicle.
- Coordinate of the target location in the polar coordinate representation of the vehicle. This is denoted by a pair of values (ρ, ϑ), where ρ is the Euclidean distance between the vehicle and the target and ϑ is the angle between the target and the heading direction of the vehicle.
- The action ω_{t-1} that the vehicle executed in the previous state.

The Euclidean distance ρ can be calculated using:

$$\rho = \sqrt{(x_o - x_t)^2 + (y_o - y_t)^2}, \tag{24}$$

in which (x_o, y_o) and (x_t, y_t) are the coordinates of the vehicle and the target. ϑ can be calculated by:

$$\vartheta = \text{sign}(\vec{v_o} \times \vec{v_t}) \arccos(\frac{\vec{v_o} \cdot \vec{v_t}}{|\vec{v_o}||\vec{v_t}|}), \tag{25}$$

where $\vec{v_o}$ and $\vec{v_t}$ represent the vehicle motion direction and the target direction with respect to the vehicle base in the polar coordinate.

Finally, the state is represented by concatenating the above three parts, as formulated below:

$$S_t = [l_1 \dots, l_{128}, \rho, \vartheta, \omega_{t-1}]. \tag{26}$$

where a 131-dimension input vector is constructed, comprising laser range values, target location and previous action, to describe the state of the vehicle.

In this work, the action space is defined by the vehicle's angular velocity, which ranges from -1 to 1 rad/s. The vehicle's linear velocity is maintained at a constant value of 0.5 m/s.

B. Reward

In this work, rewards functions are constructed by 2 main parts, namely reward for encouraging the robot to move towards the goal, named distance-shortening, and reward for collision avoidance, which are introduced next respectively.

1) Reward function for distance-shortening: The distance-shortening reward function is defined based on the distance between the vehicle and its target. We define the condition for a robot to arrive at the goal by comparing a pre-defined threshold with the distance of the robot to the goal in this work. If the distance is less than the threshold, a fixed reward of 10 is used. Otherwise, the reward is defined as the hyperbolic tangent of the distance difference between the vehicle and its target at two consecutive time steps, namely the current state s_t and the previous state s_{t-1}. The reward can be therefore formulated as follows:

$$R^{Nav}(s_t) = \begin{cases} 100 & \rho \leq \epsilon \\ r_{rel} \in [-9.9, 9.9] & else \end{cases} \tag{27}$$

where ϵ is the threshold value used to evaluate if the vehicle reached the target and is set as 0.5 in this work. r_{rel} is defined as:

$$r_{rel} = k_1 \tanh(k_2(\rho_t - \rho_{t-1})) \tag{28}$$

where k_1, k_2 are the gain values used to control the shape of the reward curve. In this paper, k_1 and k_2 are assigned as 28 and 30 separately. This reward value is clipped by two boundary values. If $r_{rel} \geq 9.9$, R^{Nav} will be set 9.9 and if $r_{rel} \leq 9.9$, R^{Nav} will be -9.9. Equation 28 ensures that the agent will gain a positive reward if the current state s_t is closer to the target than its previous state s_{t-1}. Contrarily, a negative reward will be applied if the current state is further from the target than its previous state.

2) Reward function for Collision Avoidance: This work proposes a collision avoidance reward function based on laser range finder data that incorporates a continuous social norm adjustment mechanism. In multi-vehicle navigation and collision avoidance problems, the actions of other vehicles have an important impact on the judgement of the controlled vehicle. Here, a social norm for reducing the probability of vehicle collisions is applied to the multi-vehicle navigation and collision avoidance problem. The social norm adds an offset gain to the collision reward function to encourage the robot behaviours to be biased to match the social norm of human society.

Human societies commonly employ either the left-hand or right-hand traffic rules for controlling traffic flows. When people walk on streets, people have the tendency to move to one side of the road to avoid collisions with others, depending on if it is the traffic rule. Therefore, inspired by human behaviours, an intrinsic property for robots to obey either the left-hand or right-hand traffic rule would be helpful for traffic

flow control. This intrinsic property can be added to the models by applying a modification to the reward function to prevent vehicles from colliding with each other. In our work, we add an offset gain in order to change along with the index of the laser beams, the model is expected to perform a common tendency when confronted with other robots.

In this paper, R^{CA} is divided into 2 parts, namely a normal reward function and an offset gain distribution.

The normal collision avoidance reward function can be described as:

$$R_{laser}(l_m) = -e^{-k_l(l_m - o_l)} \tag{29}$$

where $l_m = \min(l_1, l_2, \dots l_{128})$ is the minimum laser range value. k_l and o_l are the gain and distance offsets used to determine the shape of the curve. The reward is designed as Equation 29 so that the reward will decay exponentially with the decrease of the distance to the closest obstacle from the vehicle and the reward is always negative here.

The offset gain value is introduced to mimic the behaviour of social norms and is defined by the orientation of the nearest obstacle, as follows:

$$K(z) = U - D \times (1 - z), \tag{30}$$

where $z \in [0, 1]$ is the normalized index of the laser beam with the minimum value, U denotes the upper bound of the offset gain value, and D is the range between the upper and lower bounds. In this paper, U is set to 3 and D is set to 2.5. Here a right-hand order is used, so that a higher reward will be produced if the closest obstacle is on the right-hand side of the vehicle. Likewise, a lower reward will be used if the closest obstacle is on the left-hand side. In summary, the reward for collision avoidance can be represented as:

$$R^{CA}(s_t) = \begin{cases} -100 & l_m \leq \delta \\ R_{laser}(l_m) \cdot K(z) \in [-9.9, 9.9] & else \end{cases} \tag{31}$$

δ is a threshold value used to judge if the vehicle has crashed. In this paper δ is set to 0.375.

C. Main Procedure of The Algorithm

The MFPG agent is trained in a multi-vehicle environment such that the process of experience accumulation will be accelerated. Therefore, the state of each vehicle will be passed to the agent for determining the optimal action. At the initial stage before training, transitions are first collected by conducting the APF algorithm. After the experience buffer reaches the maximum volume, the training procedure will start. The main procedure of training is illustrated in Figure 3.

At the beginning of each episode, the states of the vehicles $s_t^i (i \in \{1, \dots 4\})$, along with the previous states s_{t-1}, actions a_{t-1}, and rewards of the actions $r_t^{CA,i}, r_t^{Nav,i}$, will be sent to the PER buffer for storing transitions as well as the controller to generate the subsequent action a_t. Before the buffer is fully filled, APF will be used to instruct the vehicles to produce motions and transitions that will be appended to the experience buffer. Once the buffer is fully filled, while learning from the

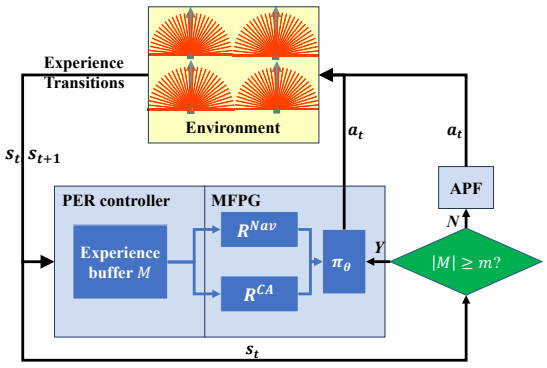
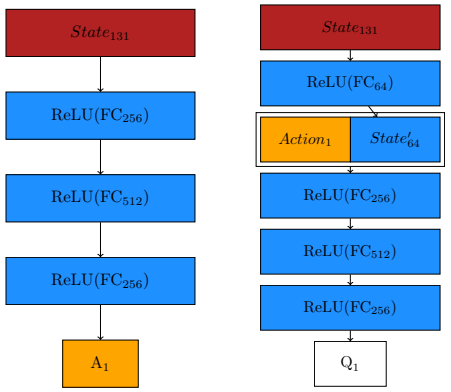


Fig. 3: Main procedure of training.



(a) Neural structure for actor. (b) Neural structure for critic.

Fig. 4: Structure of neural networks.

PER buffer, the actor neural network π_θ will start processing to produce actions for the vehicles. The structure of the actor and critic networks can be seen in Figure 4. The actor network is a simple MLP, which maps the agent state space to the regularized vehicle steering commands. The critic network takes the concatenation of the action a_{t-1} and a hidden layer that is calculated from the state s_{t-1} as the input.

At the same time, when new MDP transitions are added to the buffer, those transitions with lower probabilities will tend to be replaced, as described in Equation 23.

V. Experiments

A. Preparation

This work is tested in both simulation and real-world environments. This section introduces the test environments, and compares our work with the standard DDPG-based mapless navigation [11] and a modified DDPG algorithm for performance benchmarking in terms of both navigation and obstacle avoidance aspects. The simulation scenarios for experiments are listed below:

- an obstacle-free square cage,
- a large-scale warehouse environment with obstacles (including one additional warehouse-like environment with a larger size), and
- an unstructured complex environment (including more narrow and more obstacles).

The algorithms are implemented based on the robot operating system (ROS) and tested on Ubuntu 18.04. The platform has an Intel i7-9750H (6C12T, 2.6GHz) CPU, 16 GB memory, and a GPU of Nvidia GTX 1060Ti (6GB memory). The algorithm scripts are all implemented using Python with TensorFlow 1.14. 4 mobile robots (TurtleBot 3) are used in the simulation environment, and each is equipped with a 2-D laser radar simulated that ranges from 0.3m to 3.5m, updated at 45Hz with preset normal noises. In this paper, unless otherwise specified, all state parameters are normalized. In addition, we have also tested the work on three real robots in a real-world environment.

B. Multi-robots Navigation in Simulated Scenes

In this section, two experiments are performed in both scenarios separately. The maps used for the simulation environments are shown in Figure 5a and 5b respectively. One of the environments is an open area with no obstacles in the space (Figure 5a). Since the scene is small and without any static obstacles, the robots can sense each other more frequently than in the other environments, meaning that evaluating the performance of the methods in terms of avoiding other robots dynamically is more evident. Therefore, the focus of this scene is to test the ability of collision avoidance with other moving robots. The other environment (Figure 5b) is to simulate a warehouse-like environment, which is designed to evaluate the overall navigation performance of the robots.

Here, the standard DDPG is adopted as the baseline algorithm, based on Zhang's work [11]. The reward function in the baseline is defined as:

$$R^{baseline}(s_t) = \frac{R^{Nav}(s_t) + R^{CA}(s_t)|_{K(z)=1}}{2} \tag{32}$$

where $K(z) = 1$ means that the DDPG will have no preference in terms of the social norm. To make sure the comparison is fair, the method above is modified by incorporating the social norm into consideration in the DDPG algorithm for testing. The reward of the social-normed DDPG here can be then defined as:

$$R^{SN}(s_t) = \frac{R^{Nav}(s_t) + R^{CA}(s_t)}{2} \tag{33}$$

Here, the hyperparameters of the two algorithms are identical to the MFPG detailed in the previous section. Besides, the artificial potential field (APF) method adopted as our exploration method is also examined as a non-RL baseline.

Firstly, to examine the collision avoidance capability of the navigation model, training and evaluation are performed in the simple collision-free open space (Figure 5a). Four robots are initialized at the four corners of the room with their target destinations set at the diagonal directions of the initial positions in the room. In this environment, owing to the shape and area of the environment (5 meters on each side), the maximum number of steps for each training episode is set to 180. The learning rate of the actor network is set to $1e-2$, and that of critic networks is set to $1e-3$. Besides, Φ in Equation 14 is set to $[0.65, 0.35]$. After 750 episodes of training, each of the algorithms obtained a stable policy. To evaluate each policy, the statistical results can be seen in Table 4. The minimum covering area of each algorithm and trajectory of the vehicles of the MFPG are shown in Figures 5a. To benchmark the performance, there are a few different criteria considered in the work. First, the area of the minimum circle that covers all vehicles can be used to evaluate the control optimality of the proposed solution. This is named as the minimum covering area. The minimum covering area, along with the success rate and average target-reaching duration are all used to indicate the performance of the proposed algorithm.

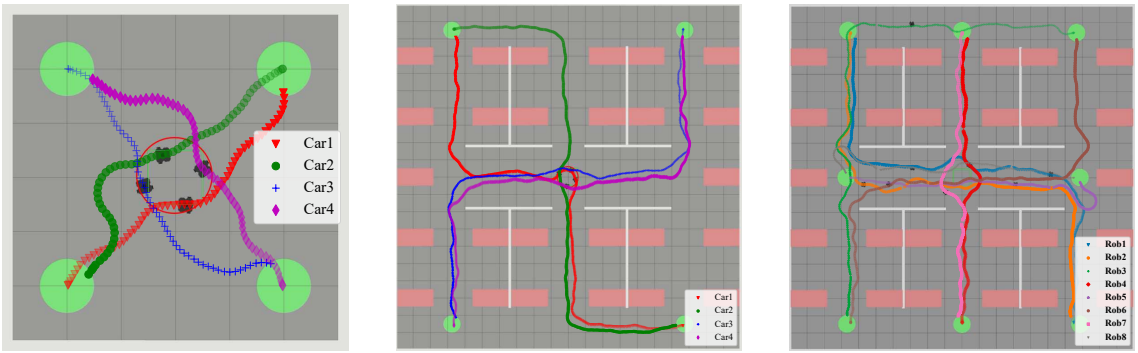
It can be seen from Table I that the proposed MFPG outperforms the standard DDPG in success rate, average duration and covering area. In comparison with the social normed DDPG, the proposed MFPG algorithm significantly improves the success rate, while the social normed DDPG performs slightly better in terms of covering area and average duration. By contrast, the APF method is substantially worse than all the other RL algorithms.

The Q values of each state-action pair can be seen in Figure 6. The change of Q^{Nav}_{MFPG} shows an upward trend, while Q^{CA}_{MFPG} declines because the networks receive negative-only rewards. The Q value of the standard DDPG grows steadily, while the trend of DDPG with the social norm is unstable during most of the training process and finally converges to a relevantly low level.

Secondly, the other experiment is performed in the simulated warehouse, aiming to test the overall navigation performance of the models in a larger-scale scene.

However, the tests show that the socially normed DDPG has performed with considerably low success rates. Therefore, the socially normed DDPG is considered not worth comparing for the task of this scenario and hence is not included in this study. The proposed method is only compared with the standard DDPG as the baseline in this experiment. In the simulated warehouse, five locations are marked for testing purposes and four robots are deployed in the environment. Four of the locations are represented by the green circles and the fifth is located in the centre of the scene (Figure 5b). The target destinations of separate robots are chosen randomly from the rest four positions. In the experiments, we only use the green circles as the candidates for the initial positions and destinations for the robots. Here, the maximum step and φ are set to 250 and 28.3, respectively. The learning rates for the networks in this scenario are set to $1e-3$ for the actor network and $5e-1$ for critic networks, respectively. The weight vector $\Phi = [\phi^{CA}, \phi^{Nav}]$ is set to $[0.45, 0.55]$, meaning that, in this scenario, the actor network will evaluate collision avoidance with a higher importance than navigation.

Figure 5b illustrates the trajectories of vehicles controlled by the MFPG in the simulated warehouse. The statistical results of the performance are shown in Table II in terms of three metrics, namely successful rate, collision distance and covering area. Here, the collision distance is defined as the Euclidean distance between the target and the location of the collision. A smaller collision distance would indicate that the vehicle managed to move closer to the destination, hence more desirable. If the vehicle reaches the target, the collision



(a) Robots navigation in an obstacle-free area.

(b) Robots navigation in the simulated warehouse.

(c) Robots navigation in the simulated warehouse with 8 robots.

Fig. 5: Trajectories of the proposed navigation model performed in the simulated environments.

TABLE I: Statistical comparison between algorithms

Method	Success Rate	Average Duration	Covering area
DDPG	94.25%	16.41	2.381
Social normed DDPG	86.0%	**14.89**	**0.785**
MFPG	**94.75%**	15.96	1.525
APF	20.375%	21.364	5.474

TABLE II: Performance Comparison between Standard DDPG, MFPG and APF

Method	Success Rate	Crash Distance		Covering area	
		mean	std	mean	std
Standard DDPG	73.267%	4.52	6.45	60.76	34.4
MFPG	**80.5%**	**2.84**	**5.27**	**5.50**	**12.23**
APF	0.5%	17.75	3.51	636.74	170.24

distance will be 0. According to Table II, the comparison of the success rates among the methods suggests that MFPG outperforms the standard DDPG. Similarly, the comparison of collision distances in the table between both algorithms suggests that MFPG has a more stable navigation performance. In addition, the covering area indicates that MFPG has a more optimal control policy than the standard DDPG. On the other hand, with regard to the Non-RL algorithm, the APF has the lowest success rate. A comparison of the success rate between Table I and Table II indicates that the APF can be easily stranded in local regions, leading to local minima.

Similarly to that of the square cage, the Q values of each state-action pair during the training are illustrated in Figure 7. The Q values of DDPG fluctuate around the zero point and finally reach the highest value at about the 15000th step, while the Q values of both the MFPG-based methods show similar trends as shown in Figure 6.

To test the generalization capability of the trained model, a test with 8 robots in the simulated warehouse is performed. 8 positions are marked in the simulated warehouse coloured in green as depicted in Figure 5c. Each robot selects a random pair of all candidate positions that are symmetric about the origin as their initial positions and destinations. Here, ROS runs at a 90Hz update rate. Other parameters are identical to the previous model. The trajectories of the eight robots are shown in Figure 5c.

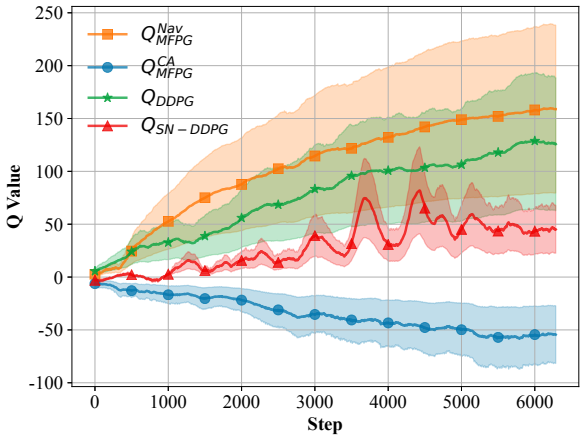


Fig. 6: Q values of each State-Action pair in the square cage environment.

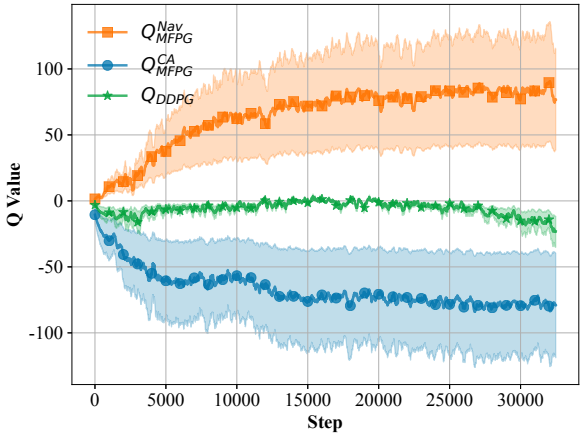


Fig. 7: Q values for each State-Action pair in the warehouse environment.

C. Benchmark and Case Study

In this section, we compare our proposed algorithms with three existing methods: the original DDPG, Soft Actor-Critic (SAC), and Twin-Delayed DDPG (TD3). We analyze the sensitivity of our proposed MFPG with different gradient weights and conduct a case study using a set of trajectories. Each

algorithm is trained with different reward weights/gradient weights separately. For the comparison between the baseline algorithms and the adaptive MFPG, we carry out a set of experiments with different reward weights to construct the reward function, defined as follows:

$$R_{baseline} = \frac{\mathbf{\Phi}_{baseline} \cdot [R^{Nav}, R^{CA}]}{2}, \tag{34}$$

Here, the reward weight $\mathbf{\Phi}_{baseline}$ is also applied to R^{Nav} and R^{CA} in our adaptive MFPG method.

The experimental results between our proposed algorithms and the baselines are shown in Table III. The success rates of the algorithms vary with the changes in the reward weight, reaching its peak when the reward weights are approximately $\mathbf{\Phi} = [1, 3]$. In the meantime, the average duration of successful runs increases as the proportion of the collision-avoidance weight increases with all algorithms, except for the adaptive MFPG. Increasing the weight for collision avoidance results in the policy becoming more cautious about its surroundings, and this leads to increased time consumption. The increase in weight shifts the policy's attention towards safety, while neglecting the navigation task reward. On the other hand, the comparison with other algorithms indicates that our proposed MFPG algorithm has the shortest average duration with stable performance. However, Soft Actor-Critic (SAC) shows a higher capability for navigation. Meanwhile, the performance of the DDPG is not satisfactory, the optimal success rate is 23.5%. The TD3 algorithm exhibits comparable performance to our proposed MFPG and Adaptive MFPG methods, although it requires significantly more computation time and has a greater degree of variance. SAC achieves the highest success rate among all the algorithms. This is due to the fact that, in SAC, the Q value is not only composed of the reward, but also the entropy of the policy. This encourages the policy to explore more and avoids getting stuck in local optima. As a result, SAC is more likely to find the optimal policy. However, according to the proof presented in Section III-B, the MFPG algorithms are expected to exhibit similar performance, which contrasts with the actual result obtained. We attribute this difference to the decoupling effect [46], [47] of the neural network parameters. In theory, it would be natural to assume that the policy gradient from 1) a combined reward-constructed critic network is identical to 2) the summation of gradients computed from separate reward-constructed critics. However, in the second case of gradient summation from separate critics, the policy gradients are computed by separate networks with different parameters. This means the gradients are constructed from distinct representations and each gradient is independent from the influence of the other. Thus, this is different from the gradient computed in the reward-summation manner. While the decoupled agent may not be able to accurately reflect the full state-action value function, it can be sufficient to do so over a subspace relevant to solving the particular task [48].

The variation of Q values during training are plotted in Figure 8. It is evident that the navigation State-Action value of the MFPG-based algorithms shows an increasing trend, while the collision avoidance value decreased and stabilized after 20,000 steps with a small standard deviation. The SAC method converged the fastest after 8,000 steps and stayed below 0. On the other hand, TD3 fluctuated around 0, making it almost impossible to adjust policy performance according to the State-Action value. Meanwhile, the DDPG method increased at a relatively low speed and eventually stabilized at 80 after 43000 steps.

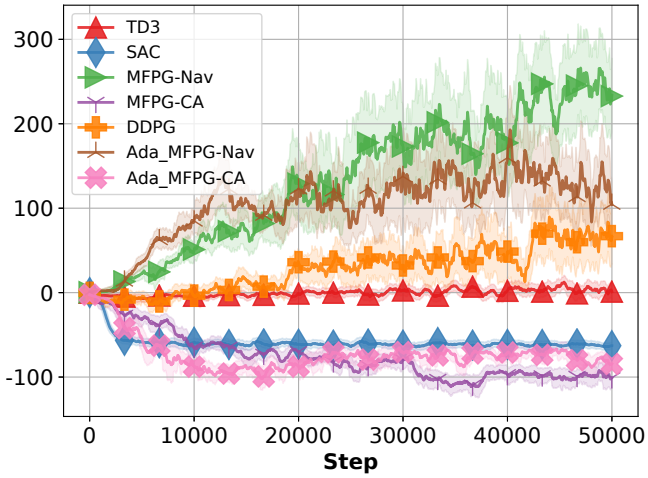


Fig. 8: Q value of each State-Action pair compared with other baselines (reward weight = [1,1]).

In addition, we provide visualization of the state-action value distribution as well as the corresponding actions with respect to a few key waypoints selected according to some given trajectories in challenging environments, as shown in Figures 9 and 10. To the best of our knowledge, this is the first attempt to visualize the distinct state-action values that provide an interpretation of the potential preferences exhibited by the agent within a framework of multi-reward-guided DRL. The test is carried out using MFPG with gradient weights [1, 3], which has the optimal performance among all MFPG-based algorithms. In the figure, we depict the laser scan range (painted in blue), action (black arrow), and the Q value distribution, along with separate action values ranging from -1 rad/s to 1 rad/s. The black arrow represents the action selected by the policy under the given circumstance. It is important to note that we evaluate the State-Action values of each algorithm using pre-defined trajectories. Therefore, the Q value distributions and the actions in the figures only represent the algorithm's preference under the given situation, but do not affect the subsequent actions themselves. Here, we only present one trajectory with a reward weight/gradient weight of [1,1]. For more trajectories and detailed case studies, we encourage readers to refer to the supplementary materials.

At first glance, it is clear that the given trajectory maintains a safe distance between the obstacles. Figure 9 shows that the baseline algorithms generally exhibit lower Q values, reflecting their cautious behaviours in navigating through the narrow gaps between obstacles. Among the seven figures, SAC (Figure 9g) stands out with the most extreme Q value distribution. On the other hand, TD3 (Figure 9e) presents the smallest range of the Q values of [-20, 30].

At waypoint 1, according to the Q value distributions, all

TABLE III: Comparison of each algorithm in terms of Success Rate (SR) and Average Duration (AD) conditioned by different Reward Weights (RW)/Gradient Weights (GW)

RW/GW	[5,1]		[3,1]		[1,1]		[1,3]		[1,5]	
metrics	SR	AD±std	SR	AD±std	SR	AD±std	SR	AD±std	SR	AD±std
MFPG	0.215	**11.536 ±4.08**	0.3525	**20.71±6.31**	0.365	**21.876±9.31**	0.425	26.128±9.97	0.36	26.138±14.4
Ada-MFPG	**0.395**	32.883±18.29	0.27	24.265±25.99	0.3275	29.436±22.46	0.3875	**24.183±16.02**	0.3125	**19.670±16.17**
DDPG	*0*	-	*0.09*	34.557±3.96	*0.235*	37.400±7.16	*0.17*	46.981±14.08	*0.22*	61.991±21.45
TD3	0.2900	*34.829±5.39*	0.3550	*48.264±18.48*	0.3425	*67.023±25.57*	0.4050	*72.219±30.79*	0.2675	*85.299±40.56*
SAC	0.32	*36.921± 7.68*	**0.46**	38.288± 9.43	**0.4775**	40.344±10.38	**0.525**	42.311±12.69	**0.4625**	46.225±15.26

critic networks detected potential risks at the current position, while the baseline algorithms show positive bias towards the robot's left-hand side. Both collision avoidance critic networks of the MFPG-based algorithms identify the risky areas in front of the robot, while the navigation networks show higher values towards the target direction. As a result, the policy selected a left turn. From waypoint 1 to waypoint 4, all policies chose actions that maximize the navigation critic policies.

At waypoint 5, a hairpin turn is used to examine the response of the algorithms. All critic networks detected the robot's tendency to collide with an obstacle. In Figures 9c and 9a, actions that maximize the navigation rewards are located on the right side of the vehicle. However, the policy network chose another action on the opposite side of the desired navigation action, due to the influence of the collision-avoidance critic network. Although the selected action was not dangerous, it could result in a longer traveling distance than its counterpart. However, among the baseline algorithms, only TD3 selected the optimal action of a sharp right turn that moves towards the target location while also avoiding collisions. In contrast, the other two algorithms chose actions that maximized the navigation reward, driving them straight into obstacles and putting them in a hazardous situation. In the subsequent steps, the robot successfully reached the target location. At waypoints 6 and 7, the robot is expected to turn left to reach the destination. The DDPG, SAC, and MFPG algorithms opted to turn left, while TD3 opted to turn right, which would have put the robot in a hazardous situation. In contrast, the Ada-MFPG algorithm chose to proceed forward, guided by the collision avoidance network. The robot finally reached the target destination after waypoints 7 and 8.

Figure 10 depicts a trajectory where the robot collides with obstacles. This trajectory is presented to identify the underlying cause of failure. With waypoint 1, four robots opted to turn left to avoid collisions, except the SAC algorithm chose to proceed straight ahead, which is perilous.

The trajectory's high-risk action commences at waypoint 6. At this point, the robot is expected to be aware of the obstacle in the right front direction of the robot, while turning right will lead to the target location. The trajectories generated exhibit a similar action to that of TD3, as illustrated in Figure 10e.

Based on waypoint 7 presented in Figure 10, it is evident that all seven critic networks hold an optimistic attitude towards the blind alley. This behaviour can be primarily attributed to the laser sensor readings. As per the sensor data, the robot detected a wide area in front of it, while the space on the right side between the obstacles was narrow. Consequently, the robot's policy misjudged the situation, leading to the collision at waypoint 7. After entering the blind alley, the agent tried to escape from the environment by turning right. However, the space within the blind alley is not enough for turning back. Finally, at waypoint 7, the agent crashed into the obstacle.

D. Experiments on Real Robots

In addition, to evaluate the generalization capability of the navigation model, the experiments are also performed on real robots in a real-world environment. Robot navigation in the real world is different from the simulated scenarios. In the real-world environment, the low accuracy of pose estimation and the delay of the commands make the navigation procedures more sophisticated than the simulated scenarios. Therefore, experiments in unseen maps would be essential to examine the validity of the model. In this work, experiments with a single robot and multiple robots were performed to test the navigation ability of the proposed method. Different types of robots are deployed in the test, including two Turtlebot-3 robots and one other customised mobile robot. Due to the limited on-board computational power of the Turtlebot-3 robots, they are connected to remote computers for data processing and control. The other robot is controlled with an onboard computer in the navigation experiments.

In the experiments, ROS runs at a lower update rate of 5Hz. To avoid the poor stability of robot control caused by signal delays, an action filter is used to improve the robustness of the navigation procedure. Firstly, the output of the model is scaled to a new bound using:

$$\hat{a}_t = \frac{2}{3\pi}\arctan(16\pi(s_t)), \tag{35}$$

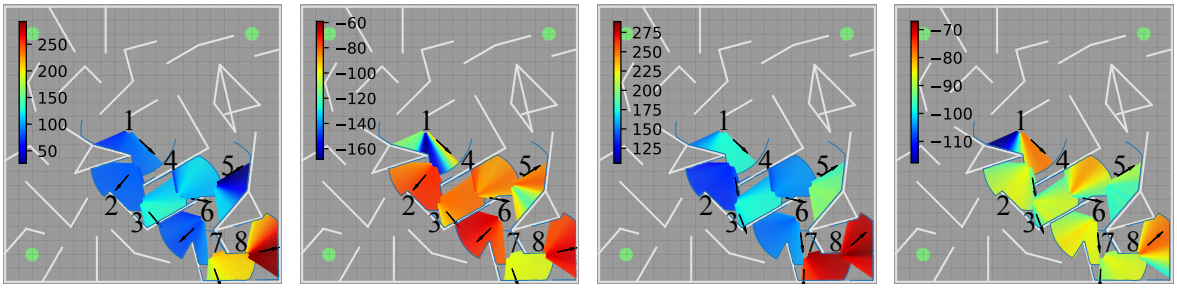
where $\hat{a}_t$ is the scaled output of the policy network $\pi(s)$ at the time step t. The action that the robot finally performed is calculated as follows:

$$a_p = [\hat{a}_t, \hat{a}_{t-1}, \hat{a}_{t-2}] \cdot [0.625I_t, 0.25I_{t-1}, 0.125I_{t-2}]^{\mathsf{T}}. \tag{36}$$

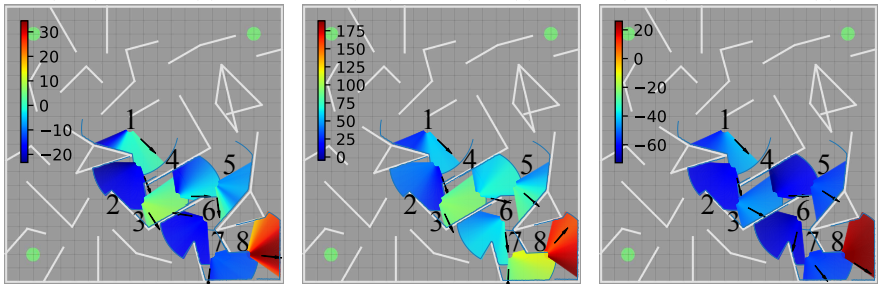
where, $[0.625I_t, 0.25I_{t-1}, 0.125I_{t-2}]$ are the weights for the actions in the previous steps. I_t is an indicator function conditioned by the action a_t and ϑ defined like:

$$I_t = \begin{cases} 1 & a_t \cdot \vartheta \geq 0 \\ 0.7 & else \end{cases}. \tag{37}$$

The indicator function has the effect of smoothing consecutive commands executed on the robot to avoid sudden manoeuvrings and, on the other hand, can undermine those actions that could result in the robot diverging away from its target. Here, ϑ is the orientation of the target at the

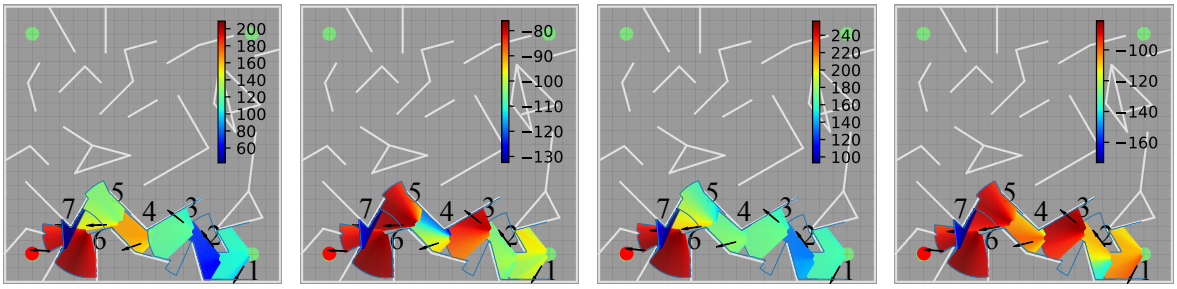


(a) Ada-MFPG (Nav). (b) Ada-MFPG (CA). (c) MFPG (Nav). (d) MFPG (CA).

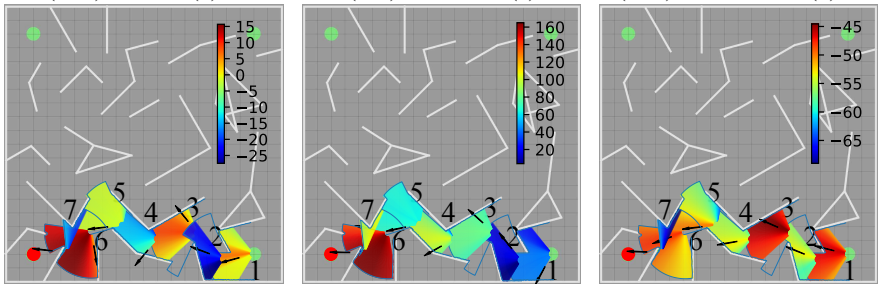


(e) TD3. (f) DDPG. (g) SAC.

Fig. 9: The State-Action value distribution in the complex environment based on the given successful navigation trajectory.



(a) Ada-MFPG (Nav). (b) Ada-MFPG (CA). (c) MFPG (Nav). (d) MFPG (CA).



(e) TD3. (f) DDPG. (g) SAC.

Fig. 10: Action and Q value distribution under the complex environment given a failure navigation trajectory.

current time, and a_t is calculated using the actions at the corresponding previous time steps. When calculating action a_{t+1} at state s_{t+1}, action a_t in Equation 36 is selected.

The control system architecture for multi-robot experiments is shown in Figure 11. Three robots are controlled by separate computers running identical navigation models. The positions of each robot are estimated using the *amcl* package in ROS. The estimated pose (ρ, ϑ) of the robot as well as the laser sensors published with the frequency of 10Hz. Note that one of the robots (robot 3) is controlled by the onboard computer, while the others are controlled by remote PCs via wireless connection, due to the hardware configurations of the work. There is no difference in terms of functionality.

Firstly, the navigation ability of the robots in both static and dynamic environments with moving obstacles is tested using the robot with the onboard computer. The trajectories of the robots in a static environment are shown in Figure 12. Figures 13a and 13b show the snapshot sequences of the robot in two dynamic environments with a human attempting to block the robot.

Furthermore, to verify the navigation ability of the robot to travel a longer distance, we deployed the MFPG with

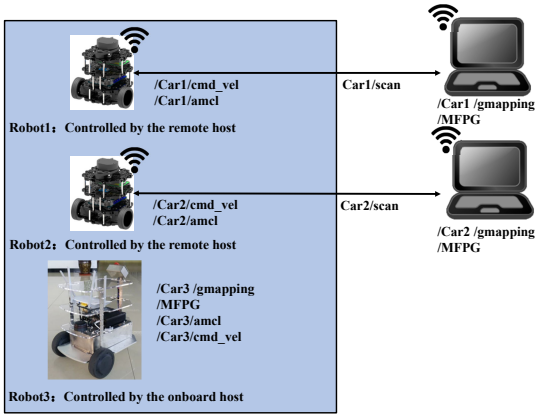
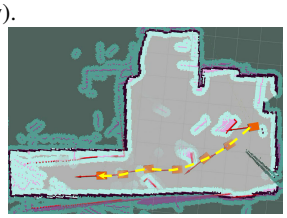


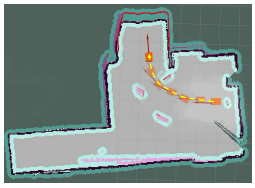
Fig. 11: The control system overview in the multi-robot experiments.

(a) Trajectory 1 in the environment with static obstacles (camera view).

(b) Trajectory 2 in the environment with static obstacles (camera view).



(c) Trajectory 1 in the environment with static obstacles (Rviz).



(d) Trajectory 2 in the environment with static obstacles (Rviz).

Fig. 12: Trajectories of robot navigation.

(a) Slices 1 of the robot in the dynamic obstacles scene.

(b) Slices 2 of the robot in the dynamic obstacles scene.

Fig. 13: Trajectories of robot navigation in dynamic environments.

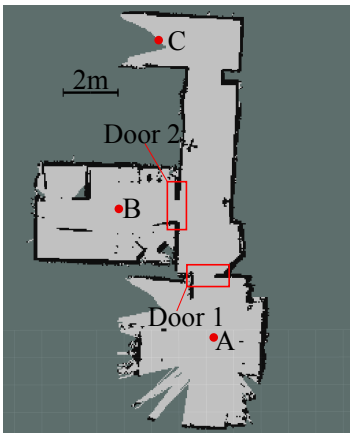


Fig. 14: The map of the experiment on a real robot in the corridor.

gradient weight (1,3) on the real robot and carried out more experiments in a corridor environment. The experimental scene comprises a long corridor with two narrow doors, as displayed in Figure 14. In the map, we selected Three Positions as the candidates for start and target positions, namely A, B, and C. Besides, two doors in the map are also annotated in the figure. The maximum travel distance is 15m from Position A to Position C. We proposed three trajectories to demonstrate the performance of the robot. Due to the page restriction, we only show one trajectory in the paper and encourage the reader to refer to the supplementary materials for more details.

Figure 15 depicts the robot's movement from position A to position B, crossing two narrow doors. The robot began from Step 1 and detected an obstacle on the right-hand side. Accordingly, it decided to move towards the right front. Upon reaching Step 2, the robot detected a dangerous situation from

the laser scan on its left front and reacted by turning right. From Step 3 to Step 4, the robot attempted to pass through the first narrow door and turned towards its target position. After Step 4, the robot headed towards the target location and arrived at Step 6 after passing through the second door. However, it should be noted that, due to the slippery floor, the odometry errors caused a discrepancy between the real trajectory and the poses estimated by Lidar-based SLAM (Simultaneous Localization and Mapping), resulting in inaccurate localization.

In addition, navigation and collision avoidance between the robots are also tested in a larger-scale environment. The trajectories of three robots are shown in Figure 16 in red, green and blue respectively at two timesteps (76 sec and 83 sec). It can be clearly seen that the robots performed well in avoiding collisions with other moving robots as expected.

VI. Conclusion

In this paper, an algorithm named MFPG and its extended version based on DDPG with multiple irreconcilable rewards is proposed for the mapless navigation problem. The method is focused on the management of each Q value network calculated from separate rewards that encourage the robot to move towards the goal, named distance-shortening, and collision avoidance respectively, and reconciling of the policy gradients generated from the Q networks. We further extended the proposed algorithm with an auto-tuning mechanism that allows the policy to adjust the weights of the separate policy gradients according to the standard deviation of the Q values. As the result of the modification, the probability of each transition being selected in the PER algorithm is adjusted via a weight-controlled component to calculate the probability of transition for being selected using two loss functions. Furthermore, we assessed the preferences of each algorithm by visualizing the distributions of state-action values separately across various states.

In order to improve the performance of collision avoidance, a continuous reward offset gain is proposed to form a specific social norm, so as to avoid collisions in the circumstance of vehicles crossing and passing.

Advantages. Experiments have shown that the proposed MFPG requires a shorter duration and higher success rate for passing and crossing in a wide obstacles-free space. In the warehouse simulation environment, MFPG outperformed its prototype algorithm and achieved optimal performance in terms of duration.

Different from combining separate rewards into a single-value reward to construct the Q value network, our proposed approach of using multiple rewards to construct Q value networks separately can also be used to optimize the parameters of the policy network. Additionally, a minor modification is applied to the PER module to reduce the loss value of each Q network effectively. Besides, experiments on the real robots have shown that the algorithm can be transferred into the real world.

Limitations. In this paper, we use only several indicators to compare the performance of the proposed method and baselines in the mapless navigation task. It is worth noting that although the algorithm exhibits a moderate success rate and an optimal average duration, one drawback of the algorithm is the assumption of accurate localization of the agent that will be the most considerable limitation in applying the algorithm to real factories. Moreover, although the convergence of the algorithm has been proved, the stability of the algorithm under complicated scenarios still remains to be examined.

Future work. Besides addressing the limitations of the algorithm, a little similarity can be found between the proposed method and C-MDP-based RL problems, both of which use multiple reward functions and several of which are related to the safety constraints. It would be interesting to extend our approaches to the constrained MDP problems, such as [49]. Another direction from the proposed algorithm is to examine if the approach can be applied to the multi-objective and multi-task problems, while incorporating more performance indicators, such as trajectory smoothness etc [50]. Furthermore, in future, we will consider incorporating trajectory smoothness in designing the social norm, as well as including linear velocity in the action space that can encourage the robot to vary its speed in its motion, in addition to only rotating. Besides, it is interesting to quantitatively evaluate the extent of the decoupling effect of multiple neural networks on the preference of the algorithms [51] [52]. Finally, we will integrate robot self-localisation methods of Visual SLAM or Visual Odometry for real-world deployment. This is part of our immediate future work [53].

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo *et al.*, "Mastering complex control in moba games with deep reinforcement learning." in *AAAI*, 2020, pp. 6672–6679.

[4] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "Decentralized navigation of multiple agents based on orca and model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3446–3451.

[5] P. Cai, H. Wang, Y. Sun, and M. Liu, "DQ-GAT: Towards safe and efficient autonomous driving with deep q-learning and graph attention networks," 2021.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

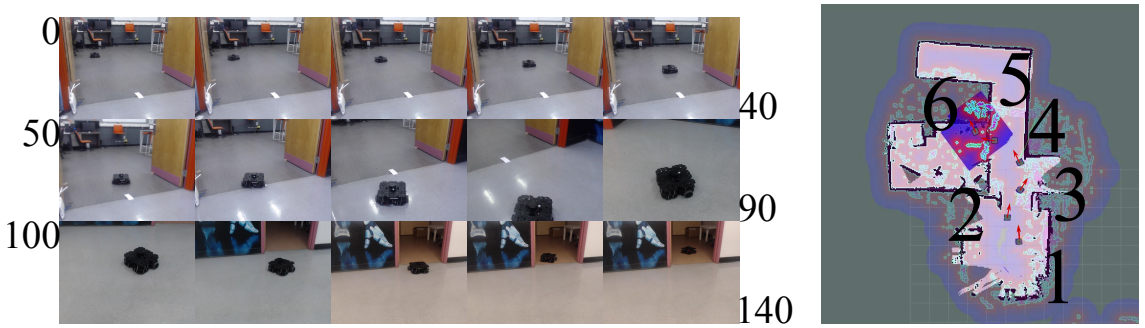
[7] T. Chaffre, J. Moras, A. Chan-Hon-Tong, and J. Marzat, "Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation," in *17th International Conference on Informatics, Automation and Robotics, ICINCO 2020*, 2020, pp. 314–323.

[8] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, p. 31, 2021.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

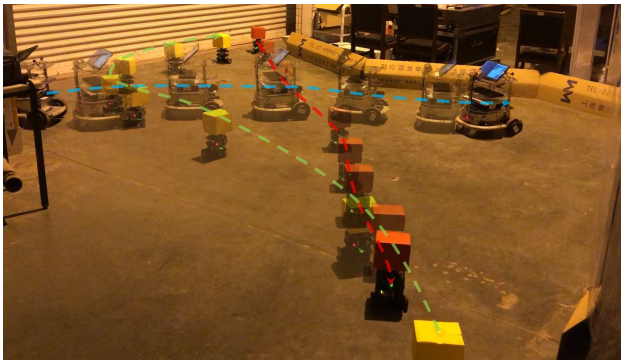
[10] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[11] P. Zhang, C. Wei, B. Cai, and Y. Ouyang, "Mapless navigation for autonomous robots: A deep reinforcement learning approach," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 3141–3146.

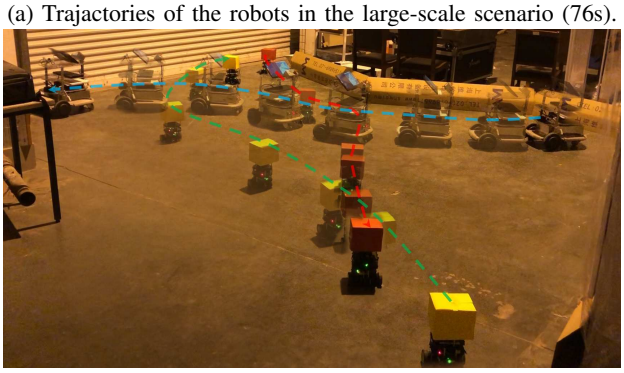


(a) Trajectory 1 view in camera. (b) Trajectory 1 view in rviz.

Fig. 15: Trajectory 1 of real robot navigation in corridor environment.



(a) Trajactories of the robots in the large-scale scenario (76s).



(b) Trajactories of the robots in the large-scale scenario (83s).

Fig. 16: Trajactories of multi-robot navigation with collision avoidance in a large-scale scenario.

[12] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning. in 2017 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1343–1350.

[13] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, p. 0278364920916531, 2020.

[14] C. De La Cruz and R. Carelli, "Dynamic model based formation control and obstacle avoidance of multi-robot systems," *Robotica*, vol. 26, no. 3, pp. 345–356, 2008.

[15] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[16] X. Wang, Z. Yan, and L. Zhong, "Centralized and decentralized methods for multi-robot safe navigation," in *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, 2022, pp. 150–159.

[17] Z. Zhang, R. Han, and J. Pan, "An efficient centralized planner for multiple automated guided vehicles at the crossroad of polynomial curves," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 398–405, 2022.

[18] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[19] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, vol. 1. IEEE, 2000, pp. 256–263.

[20] K. Rana, V. Dasagi, B. Talbot, M. Milford, and N. Sünderhauf, "Multiplicative controller fusion: A hybrid navigation strategy for deployment in unknown environments," in *IEEE International Conference on Intelligent Robots and Systems*, 2020.

[21] M. Innocenti, L. Pollini, and D. Turra, "A fuzzy approach to the guidance of unmanned air vehicles tracking moving targets," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1125–1137, Nov 2008.

[22] Y. Kuwata and J. P. How, "Cooperative distributed robust trajectory optimization using receding horizon milp," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 423–431, 2010.

[23] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[24] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[25] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty." in *AAMAS*, 2012, pp. 147–154.

[26] J. E. Godoy, I. Karamouzas, S. J. Guy, and M. L. Gini, "Implicit coordination in crowded multi-agent navigation." in *AAAI*, 2016, pp. 2487–2493.

[27] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[28] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4693–4700.

[29] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[30] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *European Conference on Computer Vision*. Springer, 2020, pp. 414–430.

[31] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[32] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7262–7269, 2021.

[33] C. Wang, Y. Niu, M. Liu, T. Shi, J. Li, and L. You, "Geomagnetic navigation for auv based on deep reinforcement learning algorithm,"

in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 2571–2575.
[34] X. Guo, Z. Ren, Z. Wu, J. Lai, D. Zeng, and S. Xie, "A deep reinforcement learning based approach for agvs path planning," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 6833–6838.
[35] F. Zeng, C. Wang, and S. S. Ge, "A survey on visual navigation for artificial agents with deep reinforcement learning," *IEEE Access*, vol. 8, pp. 135 426–135 442, 2020.
[36] I. Carlucho, M. De Paula, S. Wang, B. V. Menna, Y. R. Petillot, and G. G. Acosta, "Auv position tracking control using end-to-end deep reinforcement learning," in *OCEANS 2018 MTS/IEEE Charleston*, 2018, pp. 1–8.
[37] X. Cao, C. Sun, and M. Yan, "Target search control of auv in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96 549–96 559, 2019.
[38] B. Kabas, "Autonomous uav navigation via deep reinforcement learning using ppo," in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, 2022, pp. 1–4.
[39] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.
[40] R. B. Grando, J. C. de Jesus, V. A. Kich, A. H. Kolling, P. M. Pinheiro, R. S. Guerra, and P. L. Drews, "Mapless navigation of a hybrid aerial underwater vehicle with deep reinforcement learning through environmental generalization," in *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. IEEE, 2022, pp. 1–6.
[41] R. Chandra, V. Zinage, E. Bakolas, J. Biswas, and P. Stone, "Decentralized multi-robot social navigation in constrained environments via game-theoretic control barrier functions," *arXiv preprint arXiv:2308.10966*, 2023.
[42] L. Marzari, E. Marchesini, and A. Farinelli, "Online safety property collection and refinement for safe deep reinforcement learning in mapless navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7133–7139.
[43] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
[44] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.
[45] Y. Hou and Y. Zhang, "Improving ddpg via prioritized experience replay," *no. May*, 2019.
[46] H. Zhuang, Y. Wang, Q. Liu, and Z. Lin, "Fully decoupled neural network learning using delayed gradients," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 6013–6020, 2022.
[47] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1627–1635.
[48] T. Seyde, P. Werner, W. Schwarting, I. Gilitschenski, M. Riedmiller, D. Rus, and M. Wulfmeier, "Solving continuous control via q-learning," *arXiv preprint arXiv:2210.12566*, 2022.
[49] Y. As, I. N. Usmanova, S. Curi, and A. Krause, "Constrained policy optimization via bayesian world models," *ArXiv*, vol. abs/2201.09802, 2022.
[50] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz *et al.*, "A practical guide to multi-objective reinforcement learning and planning," *arXiv preprint arXiv:2103.09568*, 2021.
[51] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 9870–9879.
[52] R. Raileanu and R. Fergus, "Decoupling value and policy for generalization in reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8787–8798.
[53] F. Lin, C. Wei, R. Grech, and Z. Ji, "VO-safe reinforcement learning for drone navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.