EFFICIENT MULTI-AGENT COOPERATIVE NAVIGATION IN UNKNOWN ENVIRONMENTS WITH INTERLACED DEEP REINFORCEMENT LEARNING

Yue Jin, Yaodong Zhang, Jian Yuan, and Xudong Zhang

Department of Electronic Engineering, Tsinghua University, Beijing, China

ABSTRACT

This work addresses a multi-agent cooperative navigation problem that multiple agents work together in an unknown environment in order to reach different targets without collision and minimize the maximum navigation time they spend. Typical reinforcement learning-based solutions directly model the cooperative navigation policy as a steering policy. However, when each agent does not know which target to head for, this method could prolong convergence time and reduce overall performance. To this end, we model the navigation policy as a combination of a dynamic target selection policy and a collision avoidance policy. Since these two policies are coupled, an interlaced deep reinforcement learning method is proposed to simultaneously learn them. Additionally, a reward function is directly derived from the optimization objective function instead of using a heuristic design method. Extensive experiments demonstrate that the proposed method can converge in a fast way and generate a more efficient navigation policy compared with the state-of-the-art.

***Index Terms*—** Cooperative navigation, deep reinforcement learning, multi-agent control.

1. INTRODUCTION

The multi-agent cooperative navigation problem (MCNP) is a key issue in the field of cooperative multi-agent control, which can be applied to a wealth of practical applications such as autonomous warehouse and logistics, coordinating rescue, coordinating exploration and detection, etc. Given a certain number of targets located in a workspace, the goal of the MCNP is to ensure that agents can arrive at all targets with the minimum time consumption and without collision [1–9].

Many studies have been devoted to MCNPs. They differ in problem models and solutions. In terms of problem models, some studies require each agent to navigate to a target that is pre-allocated to it [1–6]. Several studies enable agents to select targets dynamically during navigation but in non-obstacle environments [7]. In terms of solutions, previous methods can be divided into non-learning methods and learning methods. Non-learning methods generally involve tunable parameters related to the scenario model [2, 9] or resort to simultaneous localization and mapping (SLAM) [10] to generate global maps for target allocation and path planning, but it always needs a centralized global planner [1].

A promising tendency for solving MCNPs is deep reinforcement learning (DRL), which combines deep learning with traditional reinforcement learning (RL) to learn a policy that maximizes the expected long-term rewards by interacting with the environment. Applied to MCNPs, two issues need to be delicately considered: reward function design and policy model. The reward function should meet the demands of action coordination and collision avoidance. The policy model determines the time complexity of the learning process. Most methods adopt heuristic reward function designs, such as setting a global reward function to collectively reward all agents [7, 8] or a local reward function based on each agent's situation [3–5]. In terms of policy models, previous methods directly model the cooperative navigation policy of each agent as a steering policy similar to that in the single-agent single-target navigation. Generally, it lacks effectiveness for solving the MCNPs with unallocated targets in this way.

In this work, we tackle the MCNP in an unknown and complex environment with unallocated targets. We model the navigation policy as a combination of a dynamic target selection policy and a collision avoidance policy. Specifically, this paper makes the following contributions.

- This paper proposes an interlaced DRL (IDRL) method to solve the MCNP and derives a reward function from the MCNP objective function instead of a heuristic design.
- Empirical results demonstrate that IDRL can converge in a fast way and generate a more efficient navigation policy compared with the state-of-the-art.

2. PROBLEM FORMULATION

This paper considers the MCNP in unknown environments with obstacles and unallocated targets as shown in Fig. 1. N self-controlled agents need to arrive at N unallocated targets without collision using as little time as possible. At time t, agent A_i acts according to its observations $\mathbf{o}_i^t = [\mathbf{o}_{ip}^t, \mathbf{o}_{id}^t]$. Specifically, $\mathbf{o}_{ip}^t = [\mathbf{o}_{i,tar_1}^t, ..., \mathbf{o}_{i,tar_n}^t, ..., \mathbf{o}_{i,tar_N}^t, \mathbf{o}_{i,ag_1}^t, ..., \mathbf{o}_{i,ag_{j\neq i}}^t, ..., \mathbf{o}_{i,ag_N}^t]$, where $\mathbf{o}_{i,tar_n}^t$ and $\mathbf{o}_{i,ag_j}^t$ are the relative position coordinates of targets and other agents from the perspective of A_i, respectively. $\mathbf{o}_{id}^t = [d_{i,1}^t, d_{i,2}^t, ..., d_{i,k}^t, ..., d_{i,K}^t]$ is the ranging result of K detection beams between -90 and

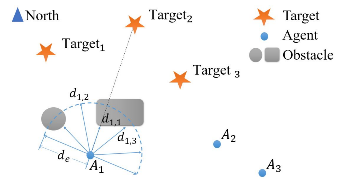

Fig. 1: Illustration of the MCNP. Blue arrows represent some detection beams. The arc dashed line represents the effective detection boundary.

90 degrees [11, 12]. The effective detection range is d_e. For simplicity, we assume that agents move at a fixed speed. Thus, each agent's navigation policy can be formulated as a mapping $\pi_i : \mathbf{o}_i^t \to c_i^t$, where c_i^t denotes A_i's steering angle.

The cooperative navigation terminates when each agent reaches a different target or any agent collides or the time reaches a sufficiently large limit. Uniformly, the cooperative navigation time can be represented as agents' maximum navigation time. Thus, the MCNP can be formulated as:

$$\begin{aligned}
&\min_{\pi_{1:N}} \mathbb{E}\left[T_{max}\left|\mathbf{o}^0_{1:N}, \pi_{1:N}\right.\right] \\
&s.t. \quad I(T_{max}) = 1 \\
&\qquad \|\mathbf{o}^{T_{max}}_{i,ag_j}\|_2 \geq d_r \quad \forall i, j \neq i \\
&\qquad d^{T_{max}}_{i,k} \geq d_r \quad \forall i, k\,.
\end{aligned} \tag{1}$$

$T_{max} = \max\limits_{i\in[1,N]} T_i$, where T_i is A_i's navigation time. $\mathbf{o}^0_{1:N} = (\mathbf{o}^0_1, ..., \mathbf{o}^0_N)$ represents observations at the beginning. $\pi_{1:N} = (\pi_1, ..., \pi_N)$. $I(t)$ is an indicator function. $I(t) = 1$ indicates that at time t each agent arrives at a different target. d_r is the distance threshold for judging the collision. The physical meaning of the constraints is that to accomplish a cooperative navigation task, each agent should arrive at a different target and keep a safe distance from other agents and obstacles.

For solving such a MCNP, we propose a distributed control method. From the perspective of A_i, since it can get the arrival status of all agents and its distances from other agents and obstacles based on $\mathbf{o}^t_{ip}$ and $\mathbf{o}^t_{id}$, its optimization objective function is:

$$\begin{aligned}
&\min_{\pi_i} \mathbb{E}\left[T_{max}\left|\mathbf{o}^0_i, \pi_i\right.\right] \\
&s.t. \quad I(T_{max}) = 1 \\
&\qquad \|\mathbf{o}^{T_{max}}_{i,ag_j}\|_2 \geq d_r \quad \forall j \neq i \\
&\qquad d^{T_{max}}_{i,k} \geq d_r \quad \forall k\,.
\end{aligned} \tag{2}$$

This problem can be verified to be NP-complete. Since it is also a control problem, we design a DRL-based solution.

3. METHOD

3.1. Deep Reinforcement Learning Framework

We model the MCNP as a Markov decision process (MDP). It can be formulated as a tuple (S, A, P, R, γ). S is the state space, A is the action space, P is the state transition probability, R is the reward function defined as: $S \times A \to r \in \mathbb{R}$, and γ is the discount factor. RL can learn a policy to maximize the expectation of long-term discount rewards $E[\sum_t \gamma^t r_t]$, where r_t is the reward at time t. The state-value function $V_\pi(s)$ and the action-value function $Q_\pi(s, a)$ are defined as:

$$V_\pi(s) = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^\tau r_{t+\tau} | s_t = s, \pi\right], \tag{3}$$

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^\tau r_{t+\tau} | s_t = s, a_t = a, \pi\right], \tag{4}$$

where T is the time horizon, s_t is the state at time t and a_t is the action at time t. The optimal policies share the same optimal state-value function $V_*(s) = \max_\pi V_\pi(s)$ and action-value function $Q_*(s, a) = \max_\pi Q_\pi(s, a)$ [13].

DRL combines deep learning with RL. Experience replay [14] and target network [15] are used to stabilize the learning process. Two common DRL algorithms are Deep Q Network (DQN) [16] and deep deterministic policy gradient (DDPG) [17]. DQN can solve the problem with discrete action space. It learns an action-value function by minimizing the loss:

$$L(\theta^\pi) = \mathbb{E}\left[(y_t - Q(s_t, a_t|\theta^\pi))^2\right]. \tag{5}$$

y_t is the target value $y_t = r_t + \gamma \max_a Q'(s_{t+1}, a|\theta^{\pi'})$, where Q' is the target network. DQN uses the ϵ-greedy policy [16] during training and finally generates a greedy policy $a = \max_a Q(s, a)$. DDPG can solve the problem with continuous action space. It learns a deterministic policy $\mu(s|\theta^\mu)$ and a Q-function $Q(s, a|\theta^Q)$ simultaneously to maximize its objective function $J(\theta^\mu) = \mathbb{E}_{s\sim d^\mu}[\sum_t \gamma^t r_t]$, where d^μ is the state distribution. The Q-function is learned to minimize (5) with the target value set as $y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$, where Q' and μ' are target networks. The policy is learned by taking steps in the direction:

$$\nabla_{\theta^\mu} J(\theta^\mu) = \mathbb{E}_{s\sim d^\mu}\left[\nabla_{\theta^\mu}\mu(s|\theta^\mu)\nabla_a Q(s, a|\theta^Q)|_{a=\mu(s|\theta^\mu)}\right]. \tag{6}$$

3.2. Reward function derivation

In this subsection, we generate a reward function from the objective function in (2). Specifically, we use a unit pulse function $\delta(\cdot)$ and two step functions $u(\cdot)$ to mark whether the constraints in (2) are satisfied. For example, if $\delta(I(t) - 1) = 1$, the first constraint is satisfied. Thus, we convert (2) as:

$$\begin{aligned}
\max_{\pi_i} \mathbb{E}\Bigg[\sum_{t=1}^{T_{max}} \Bigg(&-1 + C_1\delta\left(I(t) - 1\right) - C_2 \sum_{j=1, j\neq i}^{N} \\
&u\left(d_r - \|\mathbf{o}^t_{i,ag_j}\|_2\right) - C_3 \sum_{k=1}^{K} u\left(d_r - d^t_{i,k}\right)\Bigg)\Bigg|\mathbf{o}^0_i, \pi_i\Bigg],
\end{aligned} \tag{7}$$

where $C_1, C_2, C_3 > 0$. For ensuring equivalency, C_1 should be great enough to avoid the circumstance that the cooperative navigation process ends with a collision. Compared to the state-value function (3) in DRL, in the condition that $\gamma = 1$, A_i's reward r^t_i at time t can be presented as:

$$\begin{aligned}
r^t_i = -1 + C_1\delta\left(I(t) - 1\right) - C_2 &\sum_{j=1, j\neq i}^{N} \\
u(d_r - \|\mathbf{o}^t_{i,ag_j}\|_2) - C_3 &\sum_{k=1}^{K} u(d_r - d^t_{i,k}),
\end{aligned} \tag{8}$$

which can be expressed as:

$$r_i^t = \begin{cases} -1 - C_2, \text{ if } \exists j \neq i, \|\mathbf{o}_{i,ag_j}^t\| \leq d_r \\ -1 - C_3, \text{ else if } \exists k, d_{i,k}^t \leq d_r \\ -1 + C_1, \text{ else if } I(t) = 1 \\ -1, \text{ otherwise} \end{cases}, \tag{9}$$

3.3. Interlaced DRL for the MCNP

Since agents in this MCNP are homogeneous, they can share a common policy and reward function [3–5,8,18]. Thus, we develop a common navigation policy mapping the observations of each agent to actions. We model the navigation policy as a combination of a dynamic target selection policy and a collision avoidance policy. After selecting a target, the MCNP approximates to a single-agent navigation problem. Moreover, the target selection problem is relatively easy to be solved. Thus, compared with directly modeling the navigation policy as a steering policy, our method can reduce the time complexity of the learning process. Because these two policies are coupled, they cannot be learned separately. We propose an interlaced DRL (IDRL) method to simultaneously learn them.

The dynamic target selection policy maps $\mathbf{o}_{ip}^t$ to the index of a target $a_{ts}^t \in [1, N]$. The collision avoidance policy maps $\mathbf{o}_i^t = [\mathbf{o}_{ip}^t, \mathbf{o}_{id}^t]$ ($\mathbf{o}_{id}^t$ depends on a_{ts}^t) to a steering action a_{ca}^t in a continuous space. Generally, the dynamic target selection policy and the collision avoidance policy work sequentially at each time step. For each agent, the target selection policy works first. Then the agent rotates the center of its field of view to the selected target and obtains the ranging results $\mathbf{o}_{id}^t$ (we set the first element $d_{i,1}^t$ as the ranging result in the target direction). If no obstacles are observed in the selected target direction, i.e. $d_{i,1}^t > d_e$, the agent moves a step towards the selected target. Otherwise, the collision avoidance policy will be activated to output a steering angle relative to the target direction, and the agent moves a step in this direction. Thus, when no obstacles are observed in the selected target direction, the navigation policy is simplified to the target selection policy. Otherwise the navigation policy is a combination of the target selection policy and the collision avoidance policy.

During the training phase, considering that the target selection action space is discrete, we adopt DQN to generate a_t^{ts} using the ϵ-greedy policy based on the value of the Q-function $Q^{ts}(\mathbf{o}_{ip}^t, a_{ts}^t)$. The collision avoidance action space is continuous, thus we adopt DDPG to learn the policy $\mu(\mathbf{o}_i^t)$ and the Q-function $Q^{ca}(\mathbf{o}_i^t, a_{ca}^t)$. Since the target selection policy and the collision avoidance policy are coupled to work and influence each other, we design a unified learning structure to learn $Q^{ts}(\mathbf{o}_{ip}^t, a_{ts}^t)$ and $Q^{ca}(\mathbf{o}_i^t, a_{ca}^t)$. Specifically, according to their sequential relationship, the target value of $Q^{ts}(\mathbf{o}_{ip}^t, a_{ts}^t)$ and $Q^{ca}(\mathbf{o}_i^t, a_{ca}^t)$ are determined by whether obstacles are observed at time $t+1$. As for the target selection policy, if obstacles are observed at the next time step, its performance is determined by the collision avoidance policy. As for the collision avoidance policy, if no obstacles are observed at the next step, its performance is determined by the target selection policy. Consequently, the target value for learning $Q^{ts}(\mathbf{o}_{ip}^t, a_{ts}^t)$ and $Q^{ca}(\mathbf{o}_i^t, a_{ca}^t)$ is defined as:

$$y_i^t = \begin{cases} r_i^t + \gamma \max\limits_{a_{ts}^{t+1}} Q^{ts}(\mathbf{o}_{ip}^{t+1}, a_{ts}^{t+1}), \text{ if } d_{i,1}^{t+1} > d_e \\ r_i^t + \gamma Q^{ca}\left(\mathbf{o}_i^{t+1}, \mu(\mathbf{o}_i^{t+1})\right), \text{ otherwise} \end{cases}. \tag{10}$$

The detailed IDRL is provided in Algorithm 1.

Algorithm 1 Interlaced DRL for the MCNP of N agents

```
Initialize networks $Q^{ts}(\mathbf{o}_{ip}^t, a_{ts}^t|\theta^{ts})$, $Q^{ca}(\mathbf{o}_i^t, a_{ca}^t|\theta^{ca})$ and $\mu(\mathbf{o}_i^t|\theta^{\mu})$
with random weights $\theta^{ts}$, $\theta^{ca}$, $\theta^{\mu}$
Initialize the target networks $Q^{ts'}$, $Q^{ca'}$ and $\mu'$ with weights $\theta^{ts'} \leftarrow \theta^{ts}$,
$\theta^{ca'} \leftarrow \theta^{ca}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer $\mathcal{D}_1$, $\mathcal{D}_2$
for episode= $1, Z$ do:
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\mathbf{o}_{ip}^1$ for each agent
  for $t = 1, T$ do:
    for agent $i = 1, N$ do:
      With probability $\epsilon$ select a random target $a_{i,ts}^t$, otherwise select
      $a_{i,ts}^t = \max_a Q^{ts}(\mathbf{o}_{ip}^t, a|\theta^{ts})$
      Compute the direction angle $\rho_{ts}^t$ of target $a_{i,ts}^t$
      Rotate $\rho_{ts}^t$, detect and receive $\mathbf{o}_{id}^t$
      if $d_{i,1}^t > d_e$:
        Move a step toward target $a_{i,ts}^t$
      else:
        Compute steering angle $a_{i,ca}^t = \mu(\mathbf{o}_i^t|\theta^{\mu}) + \mathcal{N}_i^t$
        Rotate $a_{i,ca}^t$ and move a step
    end for
    for agent $i = 1, N$ do:
      Receive $r_i^t$, $\mathbf{o}_{ip}^{t+1}$
      Select a fictitious target $a_{i,ts}^{t+1} = \max\limits_a Q^{ts'}(\mathbf{o}_{ip}^{t+1}, a|\theta^{ts'})$
      Receive $\mathbf{o}_{id}^{t+1}$ as above (line 13 to 14)
      Store transition $(\mathbf{o}_{ip}^t, a_{i,ts}^t, r_i^t, \mathbf{o}_i^{t+1})$ in $\mathcal{D}_1$
      if $d_{i,1}^t \leq d_e$:
        Store transition $(\mathbf{o}_i^t, a_{i,ca}^t, r_i^t, \mathbf{o}_i^{t+1})$ in $\mathcal{D}_2$
    end for
    Sample a random minibatch of $M$ transitions from $\mathcal{D}_1$
    Sample a random minibatch of $M$ transitions from $\mathcal{D}_2$
    Set $y_i^j$ according to (10) and update $\theta^{ts}$, $\theta^{ca}$ by minimizing:
      $L(\theta^{ts}) = \frac{1}{M}\sum_j (y_i^j - Q^{ts})^2$, $L(\theta^{ca}) = \frac{1}{M}\sum_j (y_i^j - Q^{ca})^2$
    Update $\theta^{\mu}$ according to Equation (6) using sampled gradient:
    $\nabla_{\theta^{\mu}} J \approx \frac{1}{M}\sum_j \nabla_a Q^{ca}(\mathbf{o}, a|\theta^{ca})|_{\mathbf{o}=\mathbf{o}_i^j, a=\mu(\mathbf{o}_i^j)} \nabla_{\theta^{\mu}} \mu(\mathbf{o}|\theta^{\mu})|_{\mathbf{o}=\mathbf{o}_i^j}$
    Update the target networks ($\eta$ is the soft target update rate):
      $\theta^{ts'} \leftarrow \eta\theta^{ts} + (1-\eta)\theta^{ts'}$
      $\theta^{ca'} \leftarrow \eta\theta^{ca} + (1-\eta)\theta^{ca'}$
      $\theta^{\mu'} \leftarrow \eta\theta^{\mu} + (1-\eta)\theta^{\mu'}$
  end for
end for
```

4. EXPERIMENTS

We evaluate the performance of our method in a $30 \times 30m^2$ plane randomly distributed with ten round or square blocks. The size (diameter or side length) of these blocks obeys to uniform distribution $U(1,6)m$. Two target locations and two agent departure locations are randomly generated. The number of detection beams K is set to be 7, and the effective detection range d_e is $4m$. Two agents have the same speed $v = 0.5m/step$. Q^{ts} is constructed as a three-layer network with 300,200,200 units per layer and Q^{ca} is constructed as a three-layer network with 100 units per layer. μ is constructed as a network containing two hidden layers with 100 units per layer and a output layer activated by a tanh function multiplied by $\pi/2$. Additionally: $\gamma = 1$, $C_1 = 50$, $C_2 = C_3 = 2$ and the maximum episode length is 70. We first pre-train a target selection policy in obstacle-free environments. After that, we newly add obstacles and simultaneously train the target selection policy and the collision avoidance policy.

Convergence curves of different methods are shown in Fig. 2. It can be observed that by pre-training the target selection policy for a short time, IDRL converges fast. It demonstrates that the dual policy model is efficient. We compare the performance of IDRL with single-agent DDPG and MADDPG [7]. In single-agent DDPG, we randomly allocate targets and simplify this problem as a single-agent navigation problem. It only learns a collision avoidance policy by DDPG. As shown in Fig. 2, IDRL gains more rewards, because IDRL simultaneously optimizes the target selection policy and the collision avoidance policy. In MADDPG, the agents aim to directly learn a steering policy, which outputs a steering angle that blends the target selection action with the collision avoidance action. Its performance is inferior to that of IDRL and single-agent DDPG. It converges slowly and gains the least rewards. This result illustrates that MADDPG lacks efficiency to solve the MCNP in environments with obstacles and unallocated targets because the coordination among agents is hard to be learned by directly modeling the navigation policy as a steering policy.

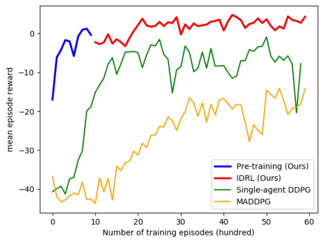

Fig. 2: Convergence curves of single-agent DDPG, MADDPG and IDRL.

To evaluate the policy learned by IDRL, we select two typical scenario samples and depict the navigation trajectories as shown in Fig. 3. The results demonstrate that the agents can avoid obstacles and reach different targets. In Fig. 3 (a), the target selection results are displayed in different colors. They show that agents can select targets dynamically during the navigation process. Specifically, agent 2 first selects target 1 but then turns to target 2 because it observes obstacles and also updates the relative position information of agent 1. Additionally, we observe that these two agents can cooperate to select different targets as the navigation procedure proceeds. In Fig. 3 (b), we compare the policy performance of IDRL with that of MADDPG. The result shows that IDRL generates less tortuous trajectories than MADDPG. The reason is that in IDRL when the agent observes no obstacles in its selected target direction it only needs to go straight to the target. Also in Fig. 3 (b), the difference in navigation time between the two methods can be measured by agents' trajectory lengths. The result shows that the maximum trajectory length of the two agents in IDRL is shorter than that in MADDPG, which indicates that the policy learned by IDRL can take less time to complete a cooperative navigation than MADDPG.

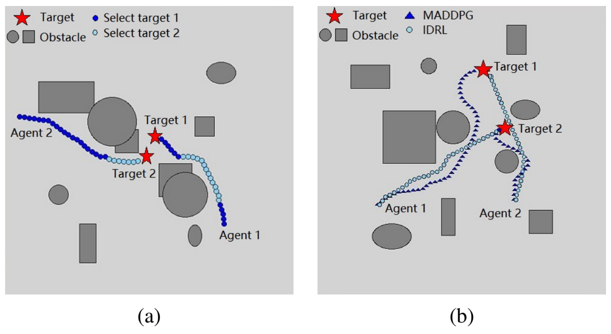

Fig. 3: Navigation trajectories generated by IDRL (Ours) and MADDPG.

To quantitatively measure the performance of different methods, we select two typical metrics: mean arrival rate ("arrival" means that each agent arrives at a different target without collision) and mean maximum navigation time. 1000 scenarios are randomly generated to get the statistical results of the metrics. The results in Table 1 demonstrate that IDRL achieves more than 16% improvement in mean arrival rate and reduces at least 15% mean maximum navigation time compared with single-agent DDPG and MADDPG.

Table 1: Mean arrival rate and mean maximum navigation time.

Obstacle size distribution (diameter or side length)	Learning method	Mean arrival rate	Mean maximum navigation time (s)
$U(1m, 2m)$	IDRL	0.98	41.34
	DDPG	0.82	48.62
	MADDPG	0.56	50.72
$U(3m, 4m)$	IDRL	0.95	41.86
	DDPG	0.76	49.05
	MADDPG	0.49	50.99

5. CONCLUSION

We present an interlaced DRL (IDRL) method which can simultaneously learn a dynamic target selection policy and a collision avoidance policy to solve the MCNP. Additionally, the reward function is directly derived instead of using heuristic design methods. Experimental results demonstrate that IDRL converges faster than traditional DRL methods. The mean arrival rate is increased by more than 16% and the mean maximum navigation time is reduced by more than 15% compared with the typical single-agent DDPG and MADDPG.

6. REFERENCES

[1] R. Reid, A. Cann, C. Meiklejohn, L. Poli, A. Boeing, and T. Braunl, "Cooperative multi-robot navigation, exploration, mapping and object detection with ros," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1083–1088.

[2] Y. Wang, D. Wang, and S. Zhu, "A new navigation function based decentralized control of multi-vehicle systems in unknown environments," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 2, pp. 363–377, August 2017.

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

[4] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE, 2018, pp. 6252–6259.

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

[6] F. S. Melo and M. Veloso, "Learning of coordination: Exploiting sparse interactions in multiagent systems," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 773–780.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[8] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," *arXiv preprint arXiv:1709.06011*, 2017.

[9] X. Li, D. Sun, and J. Yang, "Preserving multirobot connectivity in rendezvous tasks in the presence of obstacles with bounded control input," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2306–2314, January 2013.

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

[11] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 ieee international conference on robotics and automation (icra)*. IEEE, 2017, pp. 1527–1533.

[12] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 31–36.

[13] R. S. Sutton, A. G. Barto, F. Bach *et al.*, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] L. J. Lin, "Reinforcement learning for robots using neural networks," *Ph.d.thesis Carnegie Mellon University*, 1993.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, February 2015.

[16] 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.

[17] 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.

[18] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.