2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Multi-robot Cooperative Pursuit via Potential Field-Enhanced Reinforcement Learning

Zheng Zhang, Xiaohan Wang, Qingrui Zhang, and Tianjiang Hu*

Abstract— **It is of great challenge, though promising, to coordinate collective robots for hunting an evader in a decentralized manner purely in light of local observations. In this paper, this challenge is addressed by a novel hybrid cooperative pursuit algorithm that combines reinforcement learning with the artificial potential field method. In the proposed algorithm, decentralized deep reinforcement learning is employed to learn cooperative pursuit policies that are adaptive to dynamic environments. The artificial potential field method is integrated into the learning process as predefined rules to improve the data efficiency and generalization ability. It is shown by numerical simulations that the proposed hybrid design outperforms the pursuit policies either learned from vanilla reinforcement learning or designed by the potential field method. Furthermore, experiments are conducted by transferring the learned pursuit policies into real-world mobile robots. Experimental results demonstrate the feasibility and potential of the proposed algorithm in learning multiple cooperative pursuit strategies.**

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812083

I. Introduction

Multi-robotic cooperation has received extensive research attention in diverse domains due to its efficiency and capability in conducting complex missions [1], e.g., cooperative surveillance [2], search and rescue [3], and air combat [4], etc. As a typical application scenario, the cooperative pursuit aims to coordinate multiple robots for capturing one or multiple evaders [5] in a confined environment with obstacles. One of the fundamental challenges is the cooperation among robots in capturing the evader safely.

A straightforward solution to cooperative pursuit is based on a centralized architecture in which actions of robots are produced all together by one central computation module [6]–[9]. However, centralized methods are troubled with 'the curse of dimensionality', as the computation cost grows exponentially with the increase of the total number of robots. Hence, centralized methods are difficult to scale up to a multi-robot system with a large group size. Centralized methods are also vulnerable to communication failures or delays, as the joint actions computed by the central module are sent to robots in a real-time manner. On the contrary, a decentralized algorithm can distribute the computation burden to all robots in a group, so it is computationally lightweight. In decentralized algorithms, decisions are made based on local observations, which is preferable to large-scale multi-robot systems [10].

In this paper, we propose a decentralized cooperative pursuit algorithm that combines deep reinforcement learning (RL) and the artificial potential field (APF) method. In the proposed design, the last layer of the policy network is designed manually using APF to improve the data efficiency and generalization ability. Since robots are assumed to have homogeneous dynamics, their policy networks share the same structure and parameters to improve the search efficiency [11]. Following the idea in [12], the mean embedding of distributions is introduced to resolve the varying size issue of the local observations in multi-robot settings. Together with a virtual obstacle mechanism, the wall following rules are borrowed from [13], [14] to prevent the robots from being trapped in local minima [15]. The overall contributions of this paper are summarized as follows.

1) A novel decentralized pursuit algorithm is proposed to learn cooperative pursuit strategies efficiently that can leverage the merits of both deep RL and APF.
2) A modified version of wall following rules and a virtual obstacle mechanism are designed to overcome the local minima issues in APF methods.
3) The learned pursuit policy is deployed in real-world robots to show the efficiency of the proposed design.

The rest of this paper is organized as follows. In Section II, the related works are summarized. Preliminaries are provided in Section III. Section IV presents the implementation details of the proposed algorithm. Numerical simulations and experimental results are given in Section V. Finally, conclusions and future works are available in Section VI.

II. Related Works

Many decentralized strategies for cooperative pursuit belong to force-based methods, aiming to design the mathematical form of several forces to guide pursuers' move [16], [17]. For example, Janosov et al. assumed that each pursuer is attracted by the evader and repulsed by their teammates [18]. However, most force-based methods stress more on greedy chasing of evaders and collision avoidance, instead of cooperation. In [19], Escobedo advocated time-depending parameters in the aforementioned formulas are a necessity for cooperation emerging. Muro et al. demonstrated several cooperative strategies, e.g. encirclement, ambushing, and relay running, by designing a variable inter-individual repulsion depending on the distance from the evader [20]. In [21], a surrounding force, which depends on the distance from neighbors, is introduced to evenly distribute pursuers

All authors are with with Machine Intelligence and Collective Robotics (MICRO) Lab, Sun Yat-sen University, Guangzhou 510725, China
zhangzh363@mail2.sysu.edu.cn;
wangxh258@mail2.sysu.edu.cn;
zhangqr9@mail.sysu.edu.cn;
hutj3@mail.sysu.edu.cn

This paper is supported by the National Nature Science Foundation of China under Grant 62103451

on the encirclement to the evader. Although cooperation is achieved to some extent in the above works, it is extremely vulnerable to escape policies and initial conditions. Once the evader becomes more intelligent or the task environment becomes more complex, those cooperative strategies tend to collapse.

Another promising strategy for decentralized cooperative pursuit is via deep RL that learns diverse pursuit strategies by interacting with environments [22]. In comparison with force-based methods, deep RL is possible to obtain various abilities, some of which are difficult to hard-code via explicit rules, in the service of maximizing their rewards [23]. Wang et al. applied MADDPG to learn complicated cooperative strategies to chase an evader, e.g. surrounding and pushing the evader to the wall [24]. Gupta employed TRPO with parameter sharing to learn a pursuit policy that coordinates multiple pursuers to attack the evader simultaneously [11], proving deep RL outperforms Joint Equilibrium search for policies [25]. In [26], De Souza et al. injected domain-specific knowledge into deep RL by a novel reward function, which awards the pursuers when they spread around the evader. As a result, encirclement behaviors emerge to facilitate cooperation between pursuers. However, deep RL is notorious for its sample inefficiency and poor generalization ability [27]. Hence, it is still an open issue to efficiently learn cooperative strategies that can generalize to different situations.

III. Preliminaries

A. Problem Formulation

In this paper, the cooperative pursuit problem is defined for N robots pursuing a faster evader in an environment with obstacles. The evader will be *captured* by a pursuer if the distance between them is less than d_c. The mission is complete, once the evader is captured successfully by all pursuers. The following assumptions are further made.

1) The pursuing robots (pursuers) and evading robot (evader) have the radius of d_p and d_e, respectively;
2) All pursuers are homogeneous and they can only observe neighbours in a distance of d_s as shown in Figure 1.
3) The pursuers and evader move at a constant speed v_p and v_e, respectively, and $v_e > v_p$;
4) The evader position is available to all pursuers.

As shown in Figure 1, the local observation of the i-th pursuer contains the distance from the nearest obstacle $d_{o,i}$, the evader $d_{e,i}$, the teammates $d_{1,i},\ldots,d_{M,i}$, and the corresponding bearing $\phi_{o,i},\phi_{e,i},\phi_{1,i},\ldots,\phi_{M,i}$, where M is the number of observed teammates.

B. Deep Reinforcement Learning

The cooperative pursuit problem can be formulated as a standard Markov Decision Process described by a 6-tuple $(\mathscr{S},\mathscr{A},\mathscr{P},\mathscr{R},\gamma)$, where $\mathscr{S}$ is the state space, $\mathscr{A}$ is the action space, $\mathscr{P}$ is the state-transition model, $\mathscr{R}$ is the reward function, γ is the discount factor. The objective of deep RL is to learn an optimal policy $\pi^*(a|s)$ that maximizes the expectation of accumulated rewards $R_t = \sum_{t=k}^{T} \gamma^{j-t} r_t$, where s is the robot's state (local observation), a is the executed action and r_t is the reward obtained at timestep t.

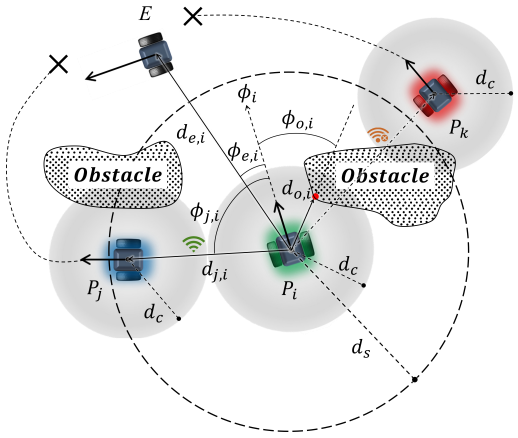


Fig. 1. The observation space of pursuers (E is the evader, while P is the pursuers). The pursuer P_i can not observe P_k due to the long distance in between. The pursuer can perceive the distance and bearing of the nearest point related to obstacles.

In deep Q-Learning, the objective is to maximize an action-value function, namely $Q^{\pi}(s,a)$ [28]. For an optimal policy π^*, there is $Q^{\pi^*}(s,a) = \max_{\pi} \mathbb{E}[R_t|s,a,\pi] = r + \gamma Q^{\pi^*}(s',a')$, where s' and a' are the state and action at the next timestep.

In real applications, the Q-value function will be parameterized by θ. The following loss function will be minimized using a gradient descent method.

$$L_j(\theta) = \frac{1}{2}\big(y_j - Q_j(s,a;\theta)\big)^2$$

where $y_j = \mathbb{E}\big[r + \gamma \max_{a'} Q_j(s',a')|s,a\big]$.

C. Artificial Potential Field

The multi-robot APF method guides robots to a goal position via the combination of attractive, repulsive, and inter-individual forces [29]. In this paper, the attractive force $F_{a,i}$ is a unit vector pointing to the target. The mathematical form of the repulsive force is

$$F_{r,i} = \begin{cases} \eta \frac{\rho_0 - \|x_{o,i} - x_i\|}{\|x_{o,i} - x_i\|^3 \rho_0} \frac{x_i - x_{o,i}}{\|x_{o,i} - x_i\|}, & \text{if } \|x_{o,i} - x_i\| \le \rho_0 \\ 0, & \text{if } \|x_{o,i} - x_i\| > \rho_0 \end{cases}$$

where ρ_0 is the influence range of obstacles and η is a positive scaling factor [17]. The inter-individual force is defined as

$$F_{in,i} = \sum_{\substack{j \neq i \\ j \in P(i)}} \left(0.5 - \frac{\lambda}{\|x_j - x_i\|}\right) \frac{x_j - x_i}{\|x_j - x_i\|} \tag{1}$$

where $P(i)$ is the set of teammates observed by the i-th robot.

The total virtual force is, therefore,

$$F_i = F_{a,i} + F_{r,i} + F_{in,i} \tag{2}$$

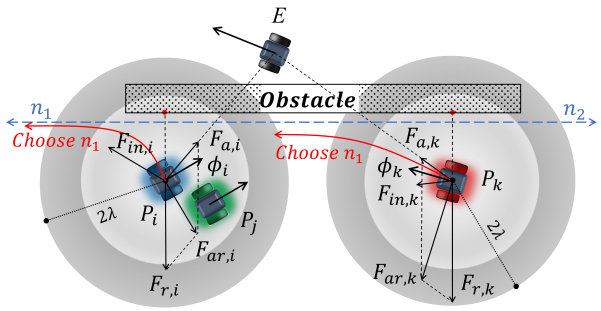


Fig. 2. The wall following rules. When the angle between $F_{ar,i}$ and $F_{a,i}$ exceeds 90°, P_i moves by the wall following rules. Due to the repulsive inter-individual force from P_j, P_i chooses to turn sharply for collision avoidance. On the contrary, P_k moves in the direction of n_1 for a smooth turn.

D. Wall Following

Conventional APF methods are notorious for their local minima issue [30]. In this paper, the local minima issue is mitigated significantly by using wall following rules [13], [14]. As shown in Figure 2, the resultant force of the attractive and repulsive forces is denoted as $F_{ar,i}$. Before each move, the robot checks whether the angle between $F_{ar,i}$ and $F_{a,i}$ exceeds 90°. If so, the robot will move in the direction of either n_1 or n_2, which are vectors perpendicular to $F_{r,i}$. The choice between n_1 and n_2 depends on the current heading ϕ_i and the inter-individual force $F_{in,i}$. If $F_{in,i}$ exceeds a predefined threshold B, the robot will select the vector which forms a smaller angle with $F_{in,i}$, otherwise the vector which forms a smaller angle with ϕ_i will be chosen.

The APF methods might also experience the local minima problem when several robots are all in the proximity of the target. For example, when a robot j has reached the target, it may block the way of another robot i to the target. Analogous to the "robot–obstacle–target" situation in Figure 2, the robot i can not bypass the robot j in the "robot i–robot j–target" situation either. Hence, a virtual obstacle is introduced in the position of any pursuer that has reached the target, which helps following robots go around teammates by the wall following rules.

IV. Adaptive Cooperative Pursuit Algorithm

A. Observation Embedding

In the multi-robot setting, the local observation's dimension of a pursuer would change with respect to time because of the possible varying number of neighbours. To address this issue, we follow the design in [12], which employs a one-layer fully connected neural network to extract high-dimensional features of each neighbour teammate. After that, the mean of all teammates' features is concatenated with the local environment observations $o_{loc,i}$ including $d_{o,i}, \phi_{o,i}, d_{e,i}, \phi_{e,i}$. Since the resultant vector is fixed-length, we can input it directly into the policy network.

B. D3QN-based Adaptive Cooperative Pursuit Algorithm

In the traditional APF method, several parameters are so influential that they determine the behaviors of the pursuers.

Algorithm 1 DACOOP

Input: Maximal episode length T, number of pursuers N, threshold B, policy update frequency T_u, target network update frequency C, replay memory $\mathscr{D}$, action-value network Q with random weights θ, target network Q^- with weights $\theta^- = \theta$

Output: Optimal policy Q

for $episode = 1, 2, ...,$ **do**
  Initialize pursuers' observations $o_1, o_2, ..., o_N$
  for $t = 1, 2, .., T$ **do**
    for $i = 1, 2, .., N$ **do**
      if Any teammate has captured the evader **then**
        Treat it as an obstacle
      end if
      Select (η_i, λ_i) by ε-greedy [28]
      Calculate attractive force $F_{a,i}$, $F_{r,i}$, $F_{in,i}$.
      if $\angle(F_{ar,i}, F_{a,i}) < 90°$ **then**
        $F_i = F_{a,i} + F_{r,i} + F_{in,i}$
      else
        Calculate n_1, n_2 orthogonal to $F_{r,i}$
        if $\|F_{in,i}\| < B$ **then**
          Choose $F_i = n_k$ by ϕ_i
        else
          Choose $F_i = n_k$ by $F_{in,i}$
        end if
      end if
      Move along the direction of F_i
      Observe the reward r_i and new observation o_i'
      Store the transition (o_i, a_i, r_i, o_i') in $\mathscr{D}$
    end for
  end for
  for $update = 1, 2, ...T_u$ **do**
    Sample transitions from $\mathscr{D}$ as [31]
    Calculate loss as [32] and update the network Q
    Every C steps copy θ to θ^-
  end for
end for

Particularly, the scale factor of the repulsive force η adjusts the influence of obstacles while the parameter λ in (1) regulates the tightness of the multi-robot team. Note that λ reflects pursuit strategies to some extent. A larger λ means the pursuers repulse each other to surround the evader while a smaller λ leads to encirclement contraction. In our design, the action space is chosen to be the parameter pair (η, λ), so one has $\pi_\theta(\eta, \lambda | s)$. The APF is thereafter employed to compute the resultant force that determines the reference heading for pursuers.

The action space is discretized into H parameter pairs empirically. Typically, η ranges from 0 (totally ignoring obstacles) to a very large value (highly conservative motions for obstacle avoidance), while λ ranges from a small positive value (moving as a tight group) to some large one (repulsing teammates in the distance).

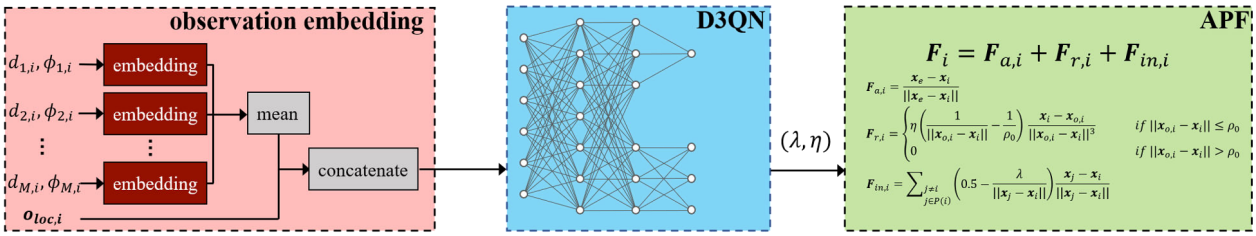


Fig. 3. The deep neural network' structure of DACOOP. Brown blocks: the observation embedding which is trained following the idea in [12]. In this paper, the observation embedding is a one-layer fully connected network with 128 units. The first layer of D3QN network has the same number of units as the observation embedding, after which two streams of fully connected networks are employed. The first stream has 64 units, then outputs Q-value for each valid action, while the second stream has the same size with a state-value output. All the activation functions are ReLU nonlinearities.

C. Reward Design

In our proposed algorithm, each pursuer obtains its own reward at each timestep, which is defined as

$$r = r_{main} + r_{time} + r_{tm} + r_o + r_{pot}$$

where r_{main} is 20 when the pursuer captures the evader and 0 otherwise. The reward r_{time} is employed to encourage pursuers to move smoothly, so it gives a penalty of -5 when the difference of headings between adjacent timesteps exceeds 45°. When the pursuer collides with other teammates, r_{tm} gives a penalty of -20. The pursuer will be punished by r_o if it is too close to obstacles, which is defined as

$$r_o^t = \begin{cases} -20, & \text{if } d_o < d_p \\ -2, & \text{if } d_p \le d_o < 1.5 d_p \\ 0, & \text{otherwise} \end{cases}$$

The function r_{pot} is a potential-based reward with

$$r_{pot} = \gamma\Phi(s) - \Phi(s'')$$

where s is the robot's state [33], and

$$\Phi(s) = \begin{cases} 15, & \text{if } d_e < 0.4 \text{ m} \\ 10, & \text{if } 0.4 \text{ m} \le d_e < 0.6 \text{ m} \\ 5, & \text{if } 0.6 \text{ m} \le d_e < 0.8 \text{ m} \\ 0, & \text{otherwise} \end{cases}$$

Note that the pursuit arena is 3.6 m × 5 m (c.f. Figure 4), so the pursuer is only rewarded by $r_{pot,i}$ when it is close to the evader. Hence, the flanking behavior will not be punished, even if it makes the pursuer temporarily away from the evader.

D. Training Setup

Our algorithm adopts the *centralized training decentralized execution* paradigm with parameter sharing, which implies a shared policy is trained with experiences collected by all pursuers. The robust reinforcement learning algorithm, D3QN [32], is extended to the multi-robot scenario in an independent RL manner [34], which treats the teammates and evader as a part of the external environment. Consequently, the whole pursuit policy is composed of the observation embedding, D3QN network and APF. The proposed algorithm is named DACOOP (D3qn-based Adaptive COOperative Pursuit algorithm) with the network architecture depicted in Fig. 3. The whole training procedure is summarized in Algorithm 1.

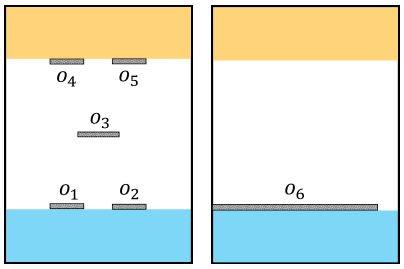


Fig. 4. The cooperative pursuit arenas. We train the cooperative pursuit policy in (a) while verifying the generalization ability in (b). The pursuers are randomly initialized in the blue area while the evader in the yellow area. o_1, o_2, o_3, o_4, o_5, and o_6 are obstacles.

V. Results And Discussion

In this section, simulations and experiments are performed to demonstrate the efficiency of the proposed DACOOP algorithm. In the simulations, the advantages of DACOOP are verified by comparing with an improved APF method and a vanilla RL algorithm. In the experiments, the learned policies by DACOOP are implemented on ground robots to illustrate their efficiency in real-life applications. In both simulations and experiments, the escaping policy of the evader is a modified version of the one in [18].

A. Numerical Simulations

The pursuit arena is shown in Fig. 4(a). The pursuers' speed is 0.3 m/s while the faster evader's speed is 0.4 m/s . The action space of DACOOP has 24 parameter pairs combined with $\eta = 0$, 1.5×10^8, 3×10^8 and $\lambda = 30$, 100, 250, 500, 750, 1000, 2000, 3000. Each episode is run maximally 1000 steps. The Q-value neural network is trained using Adam optimizer until the success rate converges (approximately 7000 episodes) [35]. We update the network 1000 times after each episode, i.e. T_u in Algorithm 1 is 1000. The learning rate is 3×10^{-4}. The discount factor γ is 0.99. The ε-greedy exploration is employed with ε linearly decayed from 1 to 0.01 in 4000 episodes. The DACOOP algorithm is compared with the following methods.

- D3QN: selecting a reference heading from 24 uniformly distributed directions. An extra reward $r_{app,i} = \frac{d''_{e,i} - d_{e,i}}{200}$ is

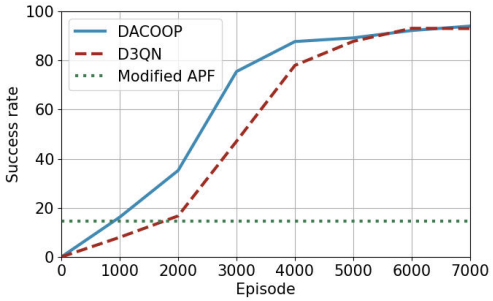


Fig. 5. The success rates of DACOOP, D3QN and the modified APF for every 1000 episodes at training.

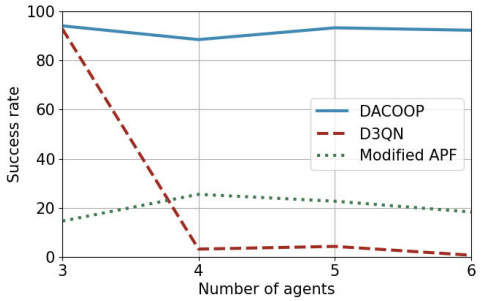


Fig. 6. The success rate with different number of pursuers. All results are averaged over 1000 episodes.

introduced with $d''_{e,i}$ denoting the distance to the evader at the last timestep. Besides the action space and reward function, all settings are the same as DACOOP;

- Modified APF: reducing η and λ from their initial values to 0 with the decrease of the distance to the evader. The best initial values are chosen from the candidate parameter pairs of DACOOP.

1) Sample efficiency: For DACOOP and D3QN, the policies are recorded every 1000 episodes at training. Fig. 5 shows the comparison results of the DACOOP, D3QN and modified APF over 1000 validation episodes. The final success rate of DACOOP is 94%, which is significantly better than the modified APF (14.6%). It verifies the significance of changing parameters for APF methods in the cooperative pursuit scenario. But it is challenging to manually design an optimal rule that adjusts the parameters according to local observations. Both DACOOP and D3QN result in similar success rates at the end of training (93% for D3QN). However, the proposed DACOOP is more data-efficient at learning according to Fig. 5. The main reason is the attractive and repulsive forces produced by the APF layer in DACOOP help pursuers approach the evader and avoid collisions in most scenarios, so many unnecessary mistakes are excluded during exploration. Furthermore, staying away from teammates allows pursuers to cover a larger area and block more possible escape routes of an evader, which is the essence of many cooperative pursuit strategies and easily accomplished by adjusting λ. Hence, high-quality data accelerates the learning process.

2) Generalization: The best immediate pursuit policies trained by DACOOP and D3QN, which achieve the highest success rate in the training environment, are selected to test their generalization performance in different scenarios. For the modified APF, the initial parameter values are the same as those in Fig. 5. As shown in Fig. 6, the policies are tested in scenarios with 3, 4, 5 and 6 pursuers, respectively. The modified APF achieves a steady performance with about 20% success rate in all scenarios. However, the success rate of D3QN decreases dramatically with the increase of the number of pursuers. The main reason is that the empirical mean embedding becomes inaccurate when the size of the multi-robotic system changes, resulting in frequent collisions between pursuers. Although DACOOP suffers from the same issue, the pursuit policy maintains the ability of approaching the evader and avoiding collisions due to the APF layer (c.f. Fig. 3). As a result, DACOOP has a better generalization performance over D3QN.

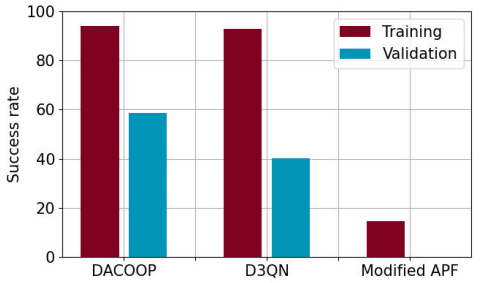


Fig. 7. The success rate in both training and validation environments. The red bars denote the performance in Fig. 4(a) while the blue bars indicate the success rate in Fig. 4(b).

The learned policies are also implemented in different arenas (Fig. 4(b)). To accomplish the mission, pursuers need to pass through the narrow gap in the right of obstacle o_6 one by one. Note this gap is only 0.4 m while those in the training environment are 0.9 m and 0.5 m (the left side of o_1, the right side of o_2 and the middle between o_1 and o_2). As shown in Fig. 7, the success rate of the modified APF is 0 in this validation environment. It implies inappropriate parameters hinder pursuers to pass through a narrower gap. In comparison with D3QN, the pursuit policy trained by DACOOP is less deteriorating in terms of success rate (40% for D3QN and 58% for DACOOP). This is because the wall following rules encourage pursuers to move along obstacles until finding a way out, while D3QN pursuers are always stuck around the obstacle previously with a gap in the training environment.

B. Experiments

Our pursuit policies are applied to differential wheeled robots. Each differential wheeled robot has a high-level computation module for decision making. A low-level PID controller runs onboard to track the heading reference command. An AprilTag is attached to the top of each robot for position and heading measurement by two monochrome cameras as shown in Fig. 8. An image processing workstation

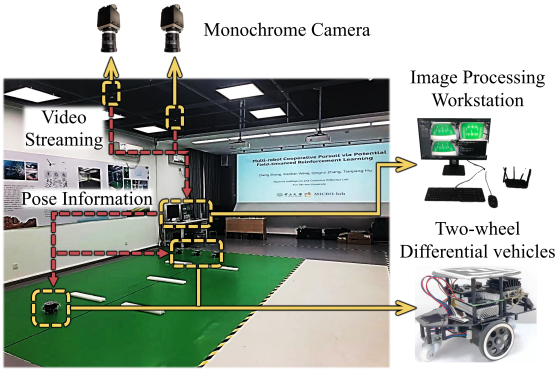
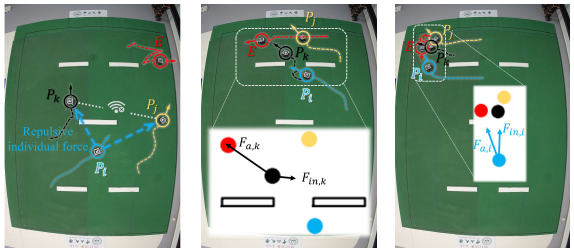
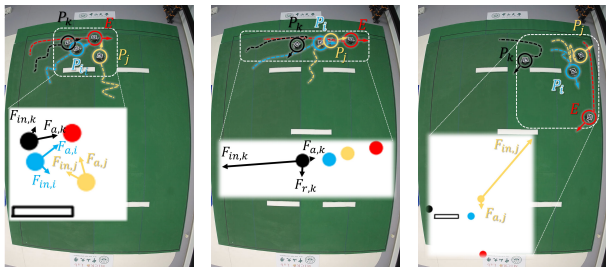


Fig. 8. The multi-ground-robots experiment system.



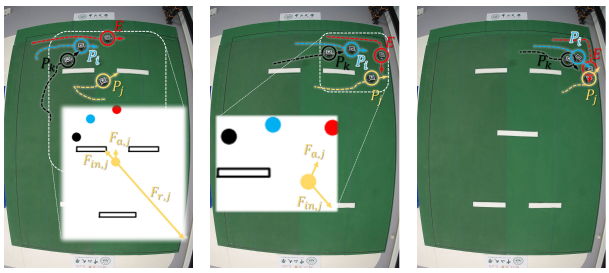
(a) $t = 10.16$ s (b) $t = 20.09$ s (c) $t = 26.17$ s

Fig. 9. The adaptive pursuit policy according to the distance from evader.



(a) $t = 24.05$ s (b) $t = 27.02$ s (c) $t = 33.08$ s

Fig. 10. The adaptive pursuit policy according to the current task. Note that the attractive force is always a unit vector and all the forces are scaled to fit the figure's size.



(a) $t = 20.13$ s (b) $t = 25.08$ s (c) $t = 27.20$ s

Fig. 11. The adaptive pursuit strategy according to the relative position in the team.

receives the video streaming from the monochrome cameras and transmits the available information to each robot at 15 Hz. The whole experiment system is shown as Fig. 8. In the rest of this section, three sets of snapshots from different episodes are presented to demonstrate the strategies learned via DACOOP.

In the first set of snapshots, when the pursuers are far away from the evader, the pursuer P_k is on the leftmost side of the team while P_j is on the rightmost (Fig. 9(a)). They can not detect each other as they are out of the sensing range. However, both of them can observe P_i who is in the rear of the team. Therefore, a large λ is selected to make the lateral pursuers move away from the group center, leading to the encirclement of the evader. When the pursuers are close to the evader, P_k and P_j approach the evader vibrantly from different paths by choosing a small λ (Fig. 9(b)). After that, the evader is captured successfully in Fig. 9(c). This case shows the learned pursuit policy is adaptive to the distance from evader.

The second set of snapshots is shown in Fig. 10. In Fig. 10(a), P_k and P_i chase the evader from the left, while P_j flanks it from below. Since the evader is faster than pursuers, it escapes from the encirclement successfully. All pursuers are thereafter behind the evader and struggle to capture it in a short time. Due to the short distance between pursuers, P_k and P_j select a large λ to move away from the overly close teammates for collision avoidance (Fig. 10(b) and (c)). This case shows that the policy trained by DACOOP is adaptive to the current task. If there is a great chance to capture the evader, pursuers will take a chance to aggregate. If the current task is to avoid collisions and wait for the next opportunity, they will repulse each other to keep a safe distance.

In the third set of snapshots shown in Figure 11, P_j could have approached the evader from the gap between obstacles, but instead it moves to the right of the map to flank the evader by selecting a large η (Fig. 11(a)-(b)). Similarly, P_j in Fig. 11(b) could choose to approach the evader directly, but it continues moving to the right to cut off the evader's escaping route. Note that P_k and P_i tend to chase the evader directly while P_j flanks it. This case demonstrates the learned policy is adaptive to the relative position in the team. Although pursuers share the same pursuit policy, they can play different roles spontaneously according to their local observations.

VI. Concluding Remarks

A novel decentralized cooperative pursuit algorithm called DACOOP is presented in this paper by integrating deep RL with APF. The DACOOP algorithm is able to fuse advantages from both learning-based methods and rule-based designs to improve the pursuit performance significantly. Simulation results showed that DACOOP achieves a higher success rate in the cooperative pursuit problem than APF. In addition, it also outperforms D3QN in terms of data efficiency and generalization. The learned policy was deployed in real-world differential wheeled robots using a direct sim-to-real transfer. We demonstrated that the learned policy was adaptive to different situations. Since the pursuers can not observe the evader all the time in the real world, an efficient search of the arena without a target signal will be our future work.

REFERENCES

[1] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[2] F. Castanedo, J. García, M. A. Patricio, and J. M. Molina, "Data fusion to improve trajectory tracking in a cooperative surveillance multi-agent architecture," *Information Fusion*, vol. 11, no. 3, pp. 243–255, 2010.

[3] M. K. Allouche and A. Boukhtouta, "Multi-agent coordination by temporal plan fusion: Application to combat search and rescue," *Information Fusion*, vol. 11, no. 3, pp. 220–232, 2010.

[4] Z. Sun, H. Piao, Z. Yang, Y. Zhao, G. Zhan, D. Zhou, G. Meng, H. Chen, X. Chen, B. Qu *et al.*, "Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play," *Engineering Applications of Artificial Intelligence*, vol. 98, p. 104112, 2021.

[5] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous robots*, vol. 31, no. 4, pp. 299–316, 2011.

[6] S. M. LaValle and J. E. Hinrichsen, "Visibility-based pursuit-evasion: The case of curved environments," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 196–202, 2001.

[7] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[8] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin, "Cooperative pursuit with voronoi partitions," *Automatica*, vol. 72, pp. 64–72, 2016.

[9] K. Shah and M. Schwager, "Multi-agent cooperative pursuit-evasion strategies under uncertainty," in *Distributed Autonomous Robotic Systems*. Springer, 2019, pp. 451–468.

[10] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[11] 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.

[12] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.

[13] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[14] X. Yun and K.-C. Tan, "A wall-following method for escaping local minima in potential field based motion planning," in *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*. IEEE, 1997, pp. 421–426.

[15] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous robots*, vol. 13, no. 3, pp. 207–222, 2002.

[16] L. Angelani, "Collective predation and escape strategies," *Physical review letters*, vol. 109, no. 11, p. 118104, 2012.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[18] M. Janosov, C. Virágh, G. Vásárhelyi, and T. Vicsek, "Group chasing tactics: how to catch a faster prey," *New Journal of Physics*, vol. 19, no. 5, p. 053003, 2017.

[19] R. Escobedo, C. Muro, L. Spector, and R. Coppinger, "Group size, individual role differentiation and effectiveness of cooperation in a homogeneous group of hunters," *Journal of the Royal Society Interface*, vol. 11, no. 95, p. 20140204, 2014.

[20] C. Muro, R. Escobedo, L. Spector, and R. Coppinger, "Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations," *Behavioural processes*, vol. 88, no. 3, pp. 192–197, 2011.

[21] X. Fang, C. Wang, L. Xie, and J. Chen, "Cooperative pursuit with multi-pursuer and one faster free-moving evader," *IEEE transactions on cybernetics*, 2020.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, p. 103535, 2021.

[24] Y. Wang, L. Dong, and C. Sun, "Cooperative control for multi-player pursuit-evasion games with reinforcement learning," *Neurocomputing*, vol. 412, pp. 101–114, 2020.

[25] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella, "Taming decentralized pomdps: Towards efficient policy computation for multiagent settings," in *IJCAI*, vol. 3. Citeseer, 2003, pp. 705–711.

[26] C. de Souza, R. Newbury, A. Cosgun, P. Castillo, B. Vidolov, and D. Kulić, "Decentralized multi-agent pursuit using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4552–4559, 2021.

[27] A. Irpan, "Deep reinforcement learning doesn't work yet," https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 1398–1404.

[30] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[32] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[33] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999, pp. 278–287.

[34] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.