2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

Large Scale Pursuit-Evasion Under Collision Avoidance Using Deep Reinforcement Learning

Helei Yang[1], Peng Ge[1], Junjie Cao[1,*], Yifan Yang[1], and Yong Liu[1,*]

Abstract— **This paper examines a pursuit-evasion game (PEG) involving multiple pursuers and evaders. The decentralized pursuers aim to collaborate to capture the faster evaders while avoiding collisions. The policies of all agents are learning-based and are subjected to kinematic constraints that are specific to unicycles. To address the challenge of high dimensionality encountered in large-scale scenarios, we propose a state processing method named Mix-Attention, which is based on Self-Attention. This method effectively mitigates the curse of dimensionality. The simulation results provided in this study demonstrate that the combination of Mix-Attention and Independent Proximal Policy Optimization (IPPO) surpasses alternative approaches when solving the multi-pursuer multi-evader PEG, particularly as the number of entities increases. Moreover, the trained policies showcase their ability to adapt to scenarios involving varying numbers of agents and obstacles without requiring retraining. This adaptability showcases their transferability and robustness. Finally, our proposed approach has been validated through physical experiments conducted with six robots.**

I. INTRODUCTION

The Pursuit-Evasion problem holds significant potential for applications in both civilian [1] and military [2]–[4] domains. A growing body of literature recognizes the multi-agent PEG since the cooperation of autonomous decision-making pursuers can increase the task success rate. While traditional control theories and optimization-based methods have proven effective in scenarios with multiple pursuers and a single evader [5]–[7], they encounter difficulties when modeling complex real-world environments.

Drawing upon the technique of learning from agents' interactions with the environment, Multi-Agent Deep Reinforcement Learning (MADRL) has been successfully applied to the PEG. However, several drawbacks still persist in current works: 1) Single Evader [8]–[11]. Advanced assignment and collaboration strategies of pursuers are crucial for environments that involve multiple evaders. 2) Reliance on rule-based or pre-trained evasion policies. Such policies adopted by the evaders can introduce the risk of overfitting in the pursuers' policy. 3) Insufficient emphasis on collision avoidance [8]–[11]. Insufficient attention has been given in previous studies to the exploration of collision avoidance between robots and obstacles as a necessary constraint, resulting in the game continuing even after a collision occurs. 4) Limited generalization and scalability. Most previous works require retraining specific policies to accommodate varying numbers of pursuers or evaders due to the fixed dimension of the input.

This paper investigates a decentralized scenario involving slower pursuers and faster evaders. Both pursuers and evaders adhere to unicycle kinematic constraints. It is assumed that the states of all agents are observable, and obstacles are distributed randomly throughout the environment. When an agent collides with an obstacle, it becomes immobile and loses its ability to interact with the environment. Successful pursuit is defined as the pursuer and evader within a predefined capture radius. The proposed scenarios in this study introduce new challenges to the algorithm, including target assignment, obstacle avoidance, and target guidance.

To address the limitation of low generalization and scalability caused by the fixed input dimension, we introduce a state processing technique called Mix-Attention based on Self-Attention. This method provides a compact feature representation for MADRL and is non-parametric in the number of agents and obstacles. We combine Mix-Attention with the widely-used multi-agent reinforcement learning algorithm IPPO [12] to handle the scenario involving multiple pursuers and evaders. Taking inspiration from [13], we train the pursuit and evasion policies synchronously to facilitate the complex co-evolution of policies.

In our simulation experiments, we evaluate the performance of our approach in various scenarios with different numbers of entities. We measure the success rate, travel distance, pursuit cost, and per-evader cost. Additionally, we assess the generalization capability of the trained policy by testing it in environments with different numbers of entities from the training setup. The experimental results demonstrate that our method significantly outperforms other approaches.

The main contributions can be summarized as follows:

1) The paper considers collision avoidance and motion constraints within a decentralized multi-pursuer multi-evader pursuit-evasion game.
2) A novel state processing method named Mix-Attention is introduced and combined with IPPO. The approach achieves lower training costs, better performance, and improved scalability compared to Bi-RNN and Mean-Embedding.
3) The effectiveness of the learned policies is demonstrated through a physical experiment involving four autonomous robots pursuing two targets, showcasing successful policy transfer to the real world.

The remaining sections of the paper are organized as follows: Section II provides a review of relevant works. Our

[1]Helei Yang, Peng Ge, Junjie Cao, Yifan Yang and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China.

*Both Yong Liu and Junjie Cao are the corresponding authors, emails: yongliu@iipc.zju.edu.cn, cjunjie@zju.edu.cn

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341975

approach is presented in Section III. Section IV offers details, results, and analysis of the simulation experiments. Section V elaborates on the implementation of the physical experiment. Finally, conclusions are drawn in Section VI.

II. Related Work

A. Non-learning methods in Pursuit-Evasion

Many non-learning approaches to the pursuit-evasion problem are rooted in differential game theory [14], graph theory, biological behavior theory, or probabilistic game theory.

Differential game theory explores the multi-pursuer single-evader PEG [5], [6] and the single-pursuer multi-evader PEG [15], [16] by considering continuous states and actions of the agents, with the state variables evolving through differential equations. Some works adopt a discrete graph representation of the environment [17]–[19] and employ graph theory to analyze it. Additionally, heuristic solutions inspired by biological perspectives [20]–[22] utilize computational simulations to study emergent behavior in groups. Certain solutions [23], [24] frame the problem within the theoretical framework of probabilistic games.

Most of the aforementioned works make assumptions about holonomic agents and known environmental dynamics. However, challenges arise when dealing with scenarios involving a growing number of entities, motion constraints, and randomly located obstacles, making it difficult to model complex environments and establish suitable objectives.

B. Deep Reinforcement Learning in Pursuit-Evasion

The majority of works in deep reinforcement learning (DRL) have focused on scenarios with multiple pursuers and a single evader. In [8], the Twin Delayed Deep Deterministic Policy Gradient (TD3) [25] algorithm was employed to train a decentralized, shared policy for homogeneous pursuers under unicycle kinematic constraints. In [9], a communication protocol based on learning was proposed for communication among pursuers. In [10], the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [26] algorithm was employed to train the pursuit policy for the task with multiple pursuers and a single evader.

The large-scale decentralized PEG has received minimal attention. In a recent work [27], a group pursuit-evasion strategy for large-scale multi-player scenarios was proposed. The strategy is based on Mean Field Games theory [28] and an actor-critic framework. A successful capture was defined as the average positions of the pursuers matching those of the evaders.

In most of the aforementioned works, evasion policies are either rule-based or pre-trained, potentially leading to overfitting risks when training the pursuit policy. Additionally, collision avoidance, which is imperative in real-world scenarios, has been neglected.

C. Policy Generalization and Scalability

Due to the fixed-dimensional input requirement of policy networks, most models are trained for a specific number of agents, which limits their generalization and scalability. Moreover, as the number of agents and obstacles increases, the length of observation data increases as well. Simply flattening or concatenating these observations as fixed-dimensional inputs would result in a dimension explosion and the well-known curse of dimensions.

In the work [29], a state representation method for multi-agent deep reinforcement learning is proposed. This method utilizes the mean-embedding of distributions to tackle the curse of dimensions. The agents are treated as samples, and their empirical mean-embedding is used as input for a decentralized policy. The researchers combine this representation method with Trust Region Policy Optimization (TRPO) [30] and validate it within the PEG.

To improve the generalization and scalability of the policy, the work [31] combines Bi-RNN and Deep Deterministic Policy Gradient (DDPG) [32]. This combination allows new agents to join the game without requiring the entire system to be retrained. However, it is worth noting that these environments do not include obstacles, and collision avoidance between agents is not considered in their approach.

III. Deep Reinforcement Learning for Pursuit-Evasion

We define each agent's task as a Markov Decision Process (MDP), which is represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ represents the set of actions, $\mathcal{P}$ refers to an unknown transition probability model, $\mathcal{R}$ represents the reward model, and $\gamma \in (0, 1]$ is a discount factor. At each time step t, the agent, located in state s_t, takes action a_t based on the policy $\pi_\theta (a_t|s_t)$. Subsequently, the agent receives a reward r_t and transitions to the next state s_{t+1}. The policy $\pi_\theta (a_t|s_t)$ is parameterized by θ, and the objective of DRL is to maximize the discounted accumulated reward $J(\theta) = \mathbb{E}_{a^t, s^t} [\sum_t \gamma^t R(s^t, a^t)]$. To enhance training efficiency, we utilize the parameter-sharing technique [33] because all pursuers and evaders are homogeneous. Specifically, each pursuer uses a shared policy $\pi_{\theta_p} (a_t|s_t)$ parametrized by θ_p, and each evader uses a shared policy $\pi_{\theta_e} (a_t|s_t)$ parametrized by θ_e. The policies $\pi_{\theta_p} (a_t|s_t)$ and $\pi_{\theta_e} (a_t|s_t)$ are trained using the experiences of all pursuers and all evaders separately.

A. The Pursuit-Evasion Scenario

We consider a circular arena with a radius of R_{arena} that contains multiple pursuers, evaders, and randomly placed obstacles. The pursuers are slower but have a numerical advantage over the evaders, who are faster but fewer in number. All agents have complete knowledge of the environment. The objective of the pursuers is to cooperatively capture the evaders in the shortest possible time, while the goal of the evaders is to avoid capture by the pursuers. A successful pursuit is defined as the distance between a pursuer and an evader being less than a specified capture radius ϵ, i.e., $\|x_e(t) - x_p(t)\| \leq \epsilon$, where $x_e(t)$ and $x_p(t)$ are the positions of the evader and the pursuer at time t, respectively. Here, $\|\cdot\|$ represents the Euclidean norm and $\epsilon > 0$. Notably,

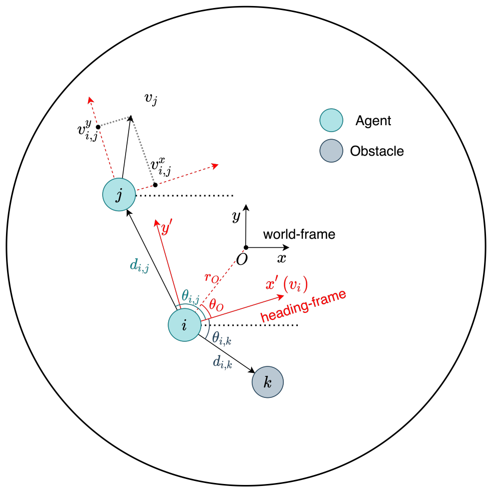


Fig. 1: The state normalization employing the heading-frame.

our approach differs from previous works in that we consider collision avoidance as a mandatory constraint (relevant, for instance, in situations involving the capture of rogue boats at sea, which necessitates avoiding collisions with islands and reefs). If an agent collides with walls, obstacles, or other agents, it becomes immobilized and can no longer interact with the environment. Furthermore, both pursuers and evaders are subject to the unicycle kinematic constraints. These constraints are represented by the following equations:

$$\dot{x} = v \cos \psi \tag{1a}$$

$$\dot{y} = v \sin \psi \tag{1b}$$

$$\dot{\psi} = \omega \tag{1c}$$

where (x, y) denotes the position of the agent, ψ represents the heading angle, v denotes the linear velocity, and ω represents the angular velocity. The controllable variables for both pursuers and evaders are the linear velocity v with limits $0 \leq v \leq v_{max}$ and the heading angular rate ω with limits $-\omega_{max} \leq \omega \leq \omega_{max}$. In our scenario, we adopt a discrete action space to emphasize the motion strategies of pursuit, evasion, and collision avoidance rather than precise movement actions.

B. State & Action Space

State normalization can enhance the efficiency and stability of reinforcement learning raining. However, in the multi-pursuer multi-evader scenario, the ego-frame [34], [35], which is commonly used in single-goal tasks like multi-agent navigation and multi-pursuer single-evader tasks, is not suitable due to the presence of multiple targets. To address this issue, we employ the heading-frame instead of the ego-frame. In the heading-frame, the origin of each agent's heading frame corresponds to its position in the fixed world frame, and the X-axis of the agent's heading-frame represents its heading direction.

The state normalization method, depicted in Fig. 1, employs the heading-frame. The state of each agent i is composed of three components: own state, other agents' state, and obstacles state. First, the own state is described as $[g_i, r_o, \theta_o, v_i]$, where g_i represents a flag indicating whether agent i is involved in a collision, r_o and θ_o denote the polar representation of the world frame origin in the heading-frame of agent i, and v_i denotes the linear velocity of agent i. Second, for each other agent $j \neq i$, their state is given by $\left[g_j, d_{i,j}, \theta_{i,j}, v_{i,j}^x, v_{i,j}^y\right]$. Here, g_j is a flag indicating whether agent j is engaged in a collision, $d_{i,j}$ represents the distance between agent i and agent j, $\theta_{i,j}$ is the polar angle of agent j in the heading-frame of agent i, and $v_{i,j}^x$ and $v_{i,j}^y$ denote the projections of the velocity vector along the x and y axes of the heading-frame of agent i. Third, the state of each obstacle k is described as $[d_{i,k}, \theta_{i,k}]$, where $d_{i,k}$ represents the distance between agent i and obstacle k, and $\theta_{i,k}$ denotes the polar angle of obstacle k in the heading-frame of agent i.

In our scenario, we utilize a discrete action space, allowing for control over both linear velocity (v) and angular rate (ω). The linear velocity ranges from 0 to v_{max}, while the angular rate ranges from $-\omega_{max}$ to ω_{max}. Specifically, we assume that the discrete linear velocity has a cardinality of N_v, and the discrete angular rate has a cardinality of N_ω. As a result, the discrete action space has a total cardinality of $N_v \times N_\omega$. Each combination action, denoted as $[v, \omega]$, can be uniquely represented by its corresponding index through the following equations:

$$v = \frac{v_{max}}{N_v - 1}\left(n_v - 1\right), \text{where } n_v \in [1, N_v] \tag{2a}$$

$$\omega = -\omega_{max} + \frac{2\omega_{max}}{N_\omega - 1}\left(n_\omega - 1\right), \text{where } n_\omega \in [1, N_\omega] \tag{2b}$$

C. Reward Structure

Assuming there are m agents and n obstacles in the circular arena. At each step, each agent takes action based on its policy and receives an individual reward. We design a compound reward function that considers distances to meet the requirements of the complex task. Specifically, for agent i taking action, we obtain its distance from the wall (d_w^i), its distance set from all obstacles ($D_o^i = \{d_o^{(i,1)}, d_o^{(i,2)}, \ldots, d_o^{(i,n)}\}$), its distance set from its teammates ($D_t^i = \{d_t^{(i,1)}, d_t^{(i,2)}, \ldots, d_t^{(i,k-1)}\}$), and its distance set from opponents ($D_{op}^i = \{d_{op}^{(i,1)}, d_{op}^{(i,2)}, \ldots, d_{op}^{(i,m-k)}\}$).

We define a common reward function for both pursuer and evader to avoid collisions:

$$f_c(x) = \begin{cases} r_c, & \text{if } x \leq 0 \\ r_c \cdot \frac{d_{danger} - x}{d_{danger} + x}, & \text{if } 0 < x \leq d_{danger} \\ 0, & \text{otherwise.} \end{cases}$$

where x is the distance between entities, r_c is the negative reward for collisions, and d_{danger} is a constant. Once $x < d_{danger}$, $f_c(x)$ becomes negative.

Another aspect of the reward is to encourage pursuit and evasion behaviors.

The reward of pursuer i to encourage pursuit behavior is:

$$\sum_{j=1}^{m-k} f_p(p_i, e_j)$$

where $f_p(p_i, e_j)$ is:

$$f_p(p_i, e_j) = \begin{cases} \frac{10}{\max\left(d_{op}^{(i,j)}, d_{cap}\right)}, & \text{if } e_j \text{ is not captured} \\ r_s, & \text{if } e_j \text{ is captured by } p_i \\ f_c\left(d_{op}^{(i,j)}\right), & \text{otherwise.} \end{cases}$$

Here, p_i is pursuer i, e_j is evader j, d_{cap} is the capture distance, and r_s is the reward for successful capture.

In time step t, the total reward of pursuer i is given by:

$$r_t^{(i)} = f_c\left(d_\omega^i\right) + \sum_{d^i \in D_o^i \cup D_t^i} f_c\left(d^i\right) + \sum_{j=1}^{m-k} f_p(p_i, e_j) \quad (3)$$

The reward of evader j to encourage evasion behavior is:

$$f_e(e_j) = \begin{cases} r_e, & \text{if } e_j \text{ is not in a collision or captured} \\ 0, & \text{otherwise.} \end{cases}$$

where r_e is a positive reward to encourage evasion behavior.

The total reward of evader j in time step t is given by:

$$r_t^{(j)} = f_c(d_\omega^j) + \sum_{d^j \in D_o^j \cup D_t^j \cup D_{op}^j} f_c(d^j) + f_e(e_j) \quad (4)$$

D. Action Mask Module

The presence of randomly located obstacles in the environment necessitates that agents take into account the possibility of colliding with walls or obstacles when selecting actions. Given that pursuers and evaders have discrete action spaces and walls/obstacles remain static, we utilize an action mask module in our experiments.

The fundamental concept underlying the action mask module is to mask actions that would lead to collisions with walls or obstacles. Importantly, the action mask module does not consider actions that could result in collisions with teammates, as agents are dynamic entities in the environment. Instead, these actions are penalized with negative rewards to foster the learning of collision avoidance strategies with teammates. Furthermore, in certain cases where an agent is positioned in close proximity to an obstacle, considering kinematic constraints, all actions within the discrete action space would inevitably lead to a collision. To mitigate this situation, we integrate a penalty into our reward design for being too close to walls/obstacles.

E. State Processing

The state processing structure is illustrated in Fig. 2 and referred to as Mix-Attention. Mix-Attention leverages self-attention [36] to facilitate the integration of information from all entities during state processing. It involves defining a projection function based on neural networks for each entity type. By applying the corresponding projection function, each agent's original state is transformed into query, key, and value components.

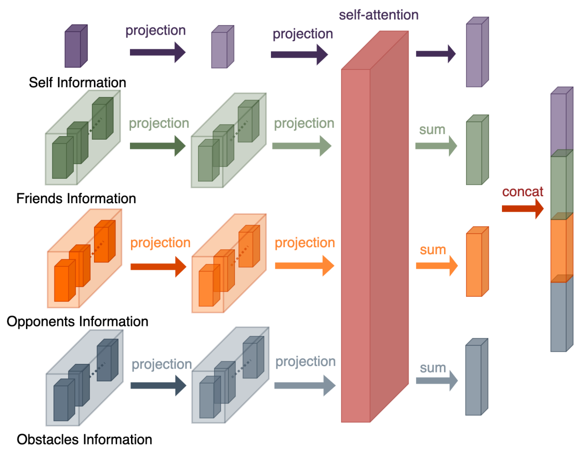


Fig. 2: The state processing process of Mix-Attention.

The state processing network is trained end-to-end through reinforcement learning. The use of projections offers several advantages: first, it ensures that the resulting embeddings remain unaffected by the permutation of entities, enabling consistent representation across different entity orderings. Second, the scale of projection function parameters is independent of the scale of entities, thereby addressing challenges associated with large system sizes and preventing inefficiencies in the learning process [37].

IV. Simulation Experiments

In this section, we conduct extensive experiments to compare the performance of our approach with the three baseline methods outlined in Section IV-A. To ensure a fair comparison of different feature representation networks, we utilize the same configuration for training and evaluating each method. Given that the policies of the evaders and pursuers are trained synchronously, facilitating their co-evolution, we introduce a cross-confrontation method in Section IV-B to assess the performance of the different methods. In Section IV-C, we present the evaluation metrics employed in our simulation experiments. Subsequently, we analyze the performance of the different methods from both the pursuer and evader perspectives, considering various scales of entities in Section IV-D. Section IV-E evaluates the generalization capability of policies trained using different methods, examining the scaling up of the number of entities without the need for retraining new policies. Lastly, in Section IV-F, we analyze the emergent group behavior of the multi-agent system.

The experimental environment configuration is displayed in Table I. The pursuer's and evader's linear velocities are discretized into three and five options, respectively. Similarly, the pursuer's and evader's heading angular velocities are discretized into 13 and 19 options, respectively.

A. Baseline Methods

We implemented three state processing methods combined with IPPO as our baseline: MLP, Bi-RNN [31], and Mean-Embedding [29]. MLP directly concatenates observations

TABLE I: Environment configuration

Item	Variable	Value
Simulation	Timestep	0.2 s
	Max Iteration Time Per Episode	200 s
Environment	Shape	Circle
	Radius	5 m
Obstacle	Shape	Circle
	Minimum Radius	0.3 m
	Maximum Radius	0.5 m
Pursuer	Shape	Circle
	Radius	0.1 m
	Maximum Linear Velocity	0.2 m/s
	Maximum Heading Angular Velocity	$\pi/3$ rad/s
Evader	Shape	Circle
	Radius	0.15 m
	Maximum Linear Velocity	0.4 m/s
	Maximum Heading Angular Velocity	$\pi/2$ rad/s

TABLE II: Results of 16 pursuers, 4 evaders, and 4 obstacles

Evader's Policy	Pursuer's Policy	Average Success Rate	Average Travel distance	Average Pursuit Cost	Average Per-Evader Cost
$E^{(16,4,4)}_{\text{MLP}}$	$P^{(16,4,4)}_{\text{Bi-RNN}}$	0.995	3.948	4.32	1.085
	$P^{(16,4,4)}_{\text{MEAN}}$	0.998	**2.954**	**3.89**	**0.975**
	$P^{(16,4,4)}_{\text{MIX}}$	**1.000**	3.527	4.35	1.088
$E^{(16,4,4)}_{\text{Bi-RNN}}$	$P^{(16,4,4)}_{\text{MLP}}$	0.565	19.805	7.05	3.119
	$P^{(16,4,4)}_{\text{MEAN}}$	0.470	**18.477**	**3.71**	1.973
	$P^{(16,4,4)}_{\text{MIX}}$	**0.863**	20.694	5.18	**1.501**
$E^{(16,4,4)}_{\text{MEAN}}$	$P^{(16,4,4)}_{\text{MLP}}$	0.193	**22.096**	5.32	6.909
	$P^{(16,4,4)}_{\text{Bi-RNN}}$	0.648	29.808	5.26	2.031
	$P^{(16,4,4)}_{\text{MIX}}$	**0.748**	27.832	**4.42**	**1.478**
$E^{(16,4,4)}_{\text{MIX}}$	$P^{(16,4,4)}_{\text{MLP}}$	0.638	**16.641**	8.56	3.357
	$P^{(16,4,4)}_{\text{Bi-RNN}}$	**0.733**	27.303	5.44	1.857
	$P^{(16,4,4)}_{\text{MEAN}}$	0.503	19.276	**3.52**	**1.751**

into a fixed dimension; Bi-RNN takes the features of the entities as the sequence input of the network; and Mean-Embedding treats the entities as samples and uses the empirical embedding as the feature representation.

Since the works mentioned above consider single-evader and assume no obstacles in the environment, we adapted their approach based on their main idea to fit our setting. Specifically, we designed an embedding module based on Bi-RNN or Mean-Embedding for each type of entity. Because non-permutation-invariant neural networks such as MLP and Bi-RNN can lead to inefficiencies in learning when operating on sets [37], we sorted the observed agent information by its type and the relative distance in order to make training more efficient and comparisons fairer.

B. Policy Cross-Confrontation

Given that the pursuit and evasion policies are learning-based and trained synchronously, a key concern arises regarding how to evaluate their performance. It becomes challenging to differentiate whether a successful capture is a result of the pursuer's strong performance or the evader's poor performance.

To address this issue, we employ the concept of cross-confrontation in our experiments. Firstly, we utilize four distinct methods (MLP, Mean-Embedding, Bi-RNN, and Mix-Attention) as state processing structures. We train simultaneous policies of pursuit and evasion using IPPO, facilitating the co-evolution of policies. By conducting training for 10 million time steps, we obtain a pursuit policy and an evasion policy for each co-evolution.

Subsequently, we evaluate the efficacy of the evolved strategies by cross-comparing the pursuit and evasion policies derived from different co-evolutions. In these cross-versus scenarios, the opponent's policy differs from the one employed during training. For the sake of convenience, we denote the policies obtained using the method $method$ and trained with m pursuers, n evaders, and k obstacles as $P^{(m,n,k)}_{method}$ or $E^{(m,n,k)}_{method}$, where P and E represent the pursuit and evasion policies, respectively.

C. Evaluation Metrics

We perform 100 trials for each comparison and utilize four evaluation indices: Success Rate, Travel Distance, Pursuit Cost, and Per-Evader Cost, to assess the performance of the policies. Let m represent the number of pursuers and n the number of evaders in the environment. In a trial where m' pursuers and n' evaders collide, the Success Rate is determined by the proportion n'/n, indicating the number of incapacitated evaders after a simulation trial.

The Travel Distance is computed as the average distance traveled by all pursuers, which is obtained by summing the travel distances of each pursuer (d_i) from 1 to m, and dividing it by m. Meanwhile, the Pursuit Cost signifies the number of incapacitated pursuers after a trial, which corresponds to m'. Additionally, the Per-Evader Cost represents the average number of pursuers required to successfully capture an evader. It is calculated by dividing n' by m'.

D. Effect of the Scale of Entities

In this section, we aim to assess how the scale of entities impacts the performance of different methods through policy cross-confrontation across three entity scales. The results of policy cross-confrontation for each environment, distinguished by their entity scales, are presented in Table II, III, and IV.

Regarding the pursuit policy, Mean-Embedding and Mix-Attention demonstrate a significant advantage in terms of Average Success Rate compared to MLP and Bi-RNN. The pursuit policy utilizing Mix-Attention surpasses other pursuit policies with respect to the Average Per-Evader Cost performance index. This implies that it achieves a considerable Average Success Rate while incurring lower pursuit costs than other state processing structures.

On the other hand, the evasion policy exhibits under-performance when using MLP and Bi-RNN. Although the evasion policy employing Mean-Embedding outperforms the Mix-Attention policy in environments with smaller entity scales, as the entity scale increases, the evasion policy using Mix-Attention dominates the Mean-Embedding policy across nearly all performance indexes.

TABLE III: Results of 20 pursuers, 5 evaders, and 8 obstacles

Evader's Policy	Pursuer's Policy	Average Success Rate	Average Travel distance	Average Pursuit Cost	Average Per-Evader Cost
$E_{\text{MLP}}^{(20,5,8)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.998	**2.743**	6.30	1.263
	$P_{\text{MEAN}}^{(20,5,8)}$	0.998	3.068	5.26	1.054
	$P_{\text{MIX}}^{(20,5,8)}$	**1.000**	3.637	**4.96**	**0.992**
$E_{\text{Bi-RNN}}^{(20,5,8)}$	$P_{\text{MLP}}^{(20,5,8)}$	0.708	14.526	8.34	2.356
	$P_{\text{MEAN}}^{(20,5,8)}$	**1.000**	**9.690**	5.68	1.136
	$P_{\text{MIX}}^{(20,5,8)}$	0.992	12.271	**5.38**	**1.085**
$E_{\text{MEAN}}^{(20,5,8)}$	$P_{\text{MLP}}^{(20,5,8)}$	0.440	**17.513**	7.31	3.323
	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.760	21.614	7.61	2.003
	$P_{\text{MIX}}^{(20,5,8)}$	**0.932**	20.138	**5.56**	**1.193**
$E_{\text{MIX}}^{(20,5,8)}$	$P_{\text{MLP}}^{(20,5,8)}$	0.542	**15.306**	9.15	3.376
	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.802	20.432	9.12	2.274
	$P_{\text{MEAN}}^{(20,5,8)}$	**0.932**	21.375	**6.11**	**1.311**

TABLE IV: Results of 24 pursuers, 6 evaders, and 12 obstacles

Evader's Policy	Pursuer's Policy	Average Success Rate	Average Travel distance	Average Pursuit Cost	Average Per-Evader Cost
$E_{\text{MLP}}^{(24,6,12)}$	$P_{\text{Bi-RNN}}^{(24,6,12)}$	0.995	3.190	6.72	1.126
	$P_{\text{MEAN}}^{(24,6,12)}$	**1.000**	2.779	6.16	1.027
	$P_{\text{MIX}}^{(24,6,12)}$	**1.000**	**2.570**	**5.93**	**0.988**
$E_{\text{Bi-RNN}}^{(24,6,12)}$	$P_{\text{MLP}}^{(24,6,12)}$	0.867	6.727	11.50	2.211
	$P_{\text{MEAN}}^{(24,6,12)}$	**1.000**	5.725	6.68	1.113
	$P_{\text{MIX}}^{(24,6,12)}$	**1.000**	**4.191**	**5.96**	**0.993**
$E_{\text{MEAN}}^{(24,6,12)}$	$P_{\text{MLP}}^{(24,6,12)}$	0.378	7.939	8.20	3.616
	$P_{\text{Bi-RNN}}^{(24,6,12)}$	0.943	14.928	8.20	1.449
	$P_{\text{MIX}}^{(24,6,12)}$	**1.000**	**7.745**	**6.49**	**1.082**
$E_{\text{MIX}}^{(24,6,12)}$	$P_{\text{MLP}}^{(24,6,12)}$	0.502	**7.365**	11.05	3.669
	$P_{\text{Bi-RNN}}^{(24,6,12)}$	0.912	16.275	8.59	1.570
	$P_{\text{MEAN}}^{(24,6,12)}$	**0.985**	14.509	**8.05**	**1.362**

E. Generalization and Scalability

In this section, we assess the generalization and scalability of the policies by directly transferring them to test environments where the opponent's policy and the number of entities (pursuers, evaders, and obstacles) differ from the environment used for training. Specifically, we transfer the pursuit policies trained through different co-evolutions in an environment with 20 chasers, 5 evaders, and 8 obstacles to environments with fewer entities (16 pursuers, 4 evaders, and 4 obstacles) and more entities (24 pursuers, 6 evaders, and 12 obstacles) without retraining. The results of these experiments are shown in Table V and VI.

Comparing the results in Table II and V, we observe that the pursuit policies trained in environments with more entities than the test environment perform well in the test environment, even outperforming specifically trained pursuit policies with the same number of entities as the test environment. This may be attributed to the parameter-sharing mechanism, which can utilize more parallel empirical data to train the policy in environments with more entities, resulting in faster convergence. However, when comparing the results in Table IV and VI, we find that directly scaling up the

TABLE V: The pursuit policy transfer to the environment with fewer entities

Evader's Policy	Pursuer's Policy	Average Success Rate	Average Travel distance	Average Pursuit Cost	Average Per-Evader Cost
$E_{\text{MLP}}^{(16,4,4)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.998	3.482	4.86	1.218
	$P_{\text{MEAN}}^{(20,5,8)}$	**1.000**	3.116	3.94	0.985
	$P_{\text{MIX}}^{(20,5,8)}$	**1.000**	**3.003**	**3.74**	**0.935**
$E_{\text{Bi-RNN}}^{(16,4,4)}$	$P_{\text{MEAN}}^{(20,5,8)}$	0.835	20.949	4.39	1.314
	$P_{\text{MIX}}^{(20,5,8)}$	**0.915**	**19.335**	**4.04**	**1.104**
$E_{\text{MEAN}}^{(16,4,4)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.390	27.899	3.59	2.301
	$P_{\text{MIX}}^{(20,5,8)}$	**0.868**	**22.598**	**3.71**	**1.069**
$E_{\text{MIX}}^{(16,4,4)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.788	**21.943**	6.52	2.070
	$P_{\text{MEAN}}^{(20,5,8)}$	**0.893**	22.985	**4.53**	**1.269**

TABLE VI: The pursuit policy transfer to the environment with more entities

Evader's Policy	Pursuer's Policy	Average Success Rate	Average Travel distance	Average Pursuit Cost	Average Per-Evader Cost
$E_{\text{MLP}}^{(24,6,12)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.998	3.263	7.92	1.322
	$P_{\text{MEAN}}^{(20,5,8)}$	**1.000**	**2.899**	6.24	1.040
	$P_{\text{MIX}}^{(20,5,8)}$	0.988	6.881	**5.79**	**0.976**
$E_{\text{Bi-RNN}}^{(24,6,12)}$	$P_{\text{MEAN}}^{(20,5,8)}$	**1.000**	**7.523**	7.08	1.180
	$P_{\text{MIX}}^{(20,5,8)}$	0.997	13.058	**6.36**	**1.064**
$E_{\text{MEAN}}^{(24,6,12)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.823	**15.803**	10.83	2.192
	$P_{\text{MIX}}^{(20,5,8)}$	**0.940**	20.449	**6.78**	**1.202**
$E_{\text{MIX}}^{(24,6,12)}$	$P_{\text{Bi-RNN}}^{(20,5,8)}$	0.738	**18.743**	11.04	2.492
	$P_{\text{MEAN}}^{(20,5,8)}$	**0.900**	20.796	**8.12**	**1.504**

number of entities without retraining the pursuit policies leads to performance deterioration across all indicators, in contrast to specifically trained policies in environments with the same number of entities as the test environment. In the presence of a change in the scale of entities, the pursuit policies using Mean-Embedding or Mix-Attention exhibit more stable performance compared to the pursuit policy using Bi-RNN, while the policy using MLP fails to adapt to the scale change due to the fixed-dimension input.

F. Analysis of Emergent Behavior

During the co-evolution of pursuit and evasion policies, we have observed interesting emergent group behaviors that appear to be crucial for successful captures.

Fig. 3a illustrates a highly cooperative behavior exhibited by the pursuers. The three pursuers on the right side of the box are actively attempting to drive the evader to the left, while the two pursuers on the left side of the box strategically position themselves to block the potential escape routes through the obstacles.

Fig. 3b demonstrates the pursuers' ability to switch targets. The pursuer on the right side of the box has already collided with an evader, prompting the other pursuers to redirect their pursuit towards different evaders to avoid wasteful pursuit costs.

Fig. 3c we observe the pursuers' tendency to trust their teammates. Despite the presence of an uncaptured evader in

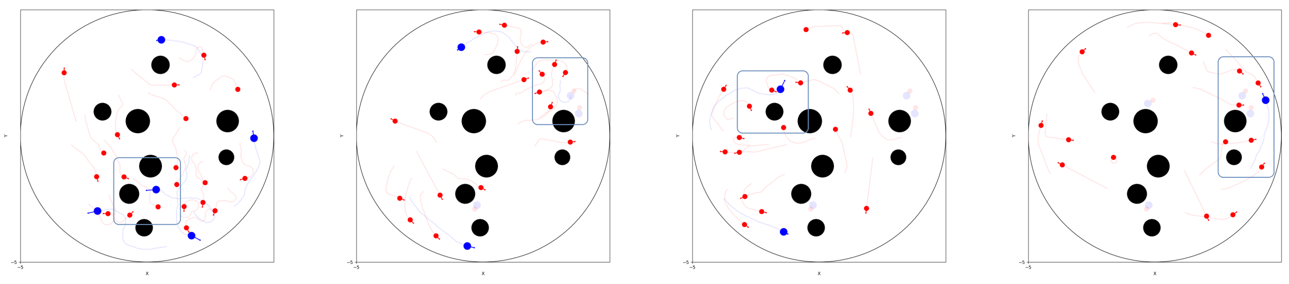
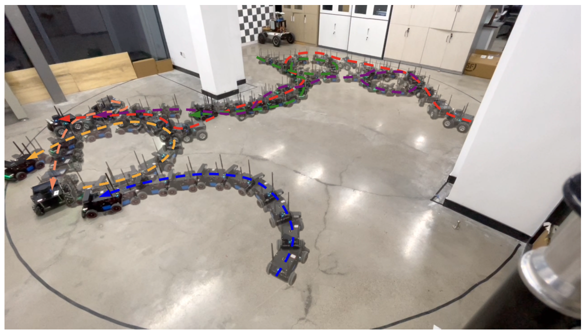
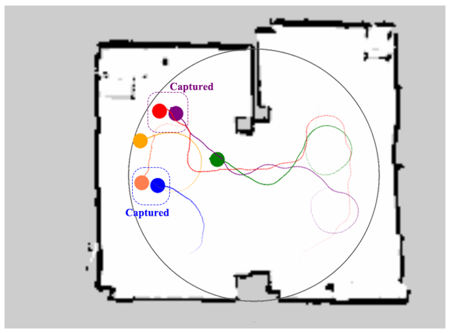


Fig. 3: The learned emergent group behaviors during the co-evolution of pursuit and evasion policies. (a) The highly cooperative behavior of the pursuers. (b) The target-switching behavior of the pursuers. (c) The pursuers' trusting behavior. (d) The behavior of exploiting the obstacles.


(a)



(b)

Fig. 4: The multi-pursuer multi-evader PEG in the real world. (a) The trajectories of all robots in a successful pursuit. (b) The trajectories of all robots showed in the constructed map.

the box, the pursuer on the left side of the box has decided to move on and capture other evaders. This decision is based on the belief that the evader in the box can be successfully captured by other teammates within a few time steps.

Fig. 3d showcases the pursuers' skill in exploiting obstacles. The three pursuers in the lower part of the box collaborate to drive the evader towards the upper part, where the presence of obstacles and the restricted movement space between the pursuers and the evaders limit the options for evasive maneuvers.

V. Demonstration on Physical World

The experiment was conducted in a real-world setting using Ubuntu 18.04 with ROS Melodic. The experiment took place in an indoor environment with a radius of approximately 3 meters. Our approach was demonstrated in a scenario where four pursuers were chasing two evaders. The pursuers were represented by Ackerman robots, while the evaders consisted of a differential chassis robot and a Mecanum wheels robot. Communication between all robots was established through a WiFi module connected to a router.

To facilitate sim-to-real transfer, we trained the pursuit and evasion policies in a simulation environment described in Section III-A. The state spaces and action spaces for the pursuers and evaders were detailed in Section III-B. The linear velocity and angular velocity of each robot were determined by the integer output of the policy network. It is important to note that due to the Ackerman chassis motion constraint on the pursuer robots, we excluded actions that had a turning radius smaller than the minimum turning radius when sampling from the policy network.

We constructed the indoor environment map using the Cartographer algorithm [38]. Each robot utilized the Adaptive Monte Carlo Localization (AMCL) algorithm to determine its position on the map and published its location and pose topic. All the required information for our algorithm was obtained from the robots' onboard sensors. The behavior of each robot was computed on a local computer and transmitted wirelessly to the robot at a rate of 5 Hz. Although our setup involved a centralized system with the local computer and robots, our methodology can still be adapted for a decentralized system if the onboard processors on each robot can be utilized for neural network inference. Fig. 4a showcases a successful decentralized pursuit-evasion demonstration, with trajectories of the robots drawn on the constructed map. The resulting trajectories can be observed in Fig. 4b.

VI. Conclusion

We propose a decentralized approach based on MADRL to address the pursuit-evasion problem involving multiple pursuers and evaders while considering collision avoidance. The learning-based policies of pursuers and evaders are trained synchronously to achieve co-evolution. Each agent, adhering to unicycle kinematics, independently determines its own action. Simulation results demonstrate that our approach, which incorporates the Mix-Attention network as the feature processor, offers superior generalization in both pursuit and evasion policies as the number of entities increases. Comparatively, it outperforms the mean-embedding network and the bidirectional RNN network commonly employed in recent MADRL-based approaches. Furthermore, we validate the feasibility of our algorithm through real-world experiments involving up to six robots with motion constraints.

References

[1] D. W. Oyler, P. T. Kabamba, and A. R. Girard, "Pursuit–evasion games in the presence of obstacles," *Automatica*, vol. 65, pp. 1–11, 2016.

[2] V. Turetsky and T. Shima, "Target evasion from a missile performing multiple switches in guidance law," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 10, pp. 2364–2373, 2016.

[3] V. Turetsky and J. Shinar, "Missile guidance laws based on pursuit–evasion game formulations," *Automatica*, vol. 39, no. 4, pp. 607–618, 2003.

[4] B. Vlahov, E. Squires, L. Strickland, and C. Pippin, "On developing a uav pursuit-evasion policy using reinforcement learning," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 859–864.

[5] A. Von Moll, D. W. Casbeer, E. Garcia, and D. Milutinović, "Pursuit-evasion of an evader by multiple pursuers," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 133–142.

[6] 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.

[7] W. Sun, P. Tsiotras, T. Lolla, D. N. Subramani, and P. F. Lermusiaux, "Multiple-pursuer/one-evader pursuit–evasion game in dynamic flow-fields," *Journal of guidance, control, and dynamics*, vol. 40, no. 7, pp. 1627–1637, 2017.

[8] 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.

[9] 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.

[10] G. Singh, D. M. Lofaro, and D. Sofge, "Pursuit-evasion with decentralized robotic swarm in continuous state space and action space via deep reinforcement learning." in *ICAART (1)*, 2020, pp. 226–233.

[11] A. Brandenburger, F. Hoffmann, and A. Charlish, "Co-training an observer and an evading target," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. IEEE, 2021, pp. 1–8.

[12] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the starcraft multi-agent challenge?" *arXiv preprint arXiv:2011.09533*, 2020.

[13] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

[14] A. Friedman, *Differential games*. Courier Corporation, 2013.

[15] S.-Y. Liu, Z. Zhou, C. Tomlin, and K. Hedrick, "Evasion as a team against a faster pursuer," in *2013 American control conference*. IEEE, 2013, pp. 5368–5373.

[16] A. A. Chikrii and S. Kalashnikova, "Pursuit of a group of evaders by a single controlled object," *Cybernetics*, vol. 23, no. 4, pp. 437–445, 1987.

[17] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and applications of graphs*. Springer, 1978, pp. 426–441.

[18] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 32–47, 2009.

[19] G. Ibragimov and S. Luckraz, "On a characterization of evasion strategies for pursuit-evasion games on graphs," *Journal of Optimization Theory and Applications*, vol. 175, no. 2, pp. 590–596, 2017.

[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] W. Li, "A dynamics perspective of pursuit-evasion: Capturing and escaping when the pursuer runs faster than the agile evader," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 451–457, 2016.

[22] J. Wang, G. Li, L. Liang, C. Wang, and F. Deng, "Pursuit-evasion games of multiple cooperative pursuers and an evader: A biological-inspired perspective," *Communications in Nonlinear Science and Numerical Simulation*, vol. 110, p. 106386, 2022.

[23] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 662–669, 2002.

[24] F. Shkurti, N. Kakodkar, and G. Dudek, "Model-based probabilistic pursuit via inverse reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7804–7811.

[25] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[26] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[27] Z. Zhou and H. Xu, "Decentralized optimal large scale multi-player pursuit-evasion strategies: A mean field game approach with reinforcement learning," *Neurocomputing*, vol. 484, pp. 46–58, 2022.

[28] J.-M. Lasry and P.-L. Lions, "Mean field games," *Japanese journal of mathematics*, vol. 2, no. 1, pp. 229–260, 2007.

[29] 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.

[30] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[31] L. Xu, B. Hu, Z. Guan, X. Cheng, T. Li, and J. Xiao, "Multi-agent deep reinforcement learning for pursuit-evasion game scalability," in *Chinese Intelligent Systems Conference*. Springer, 2019, pp. 658–669.

[32] 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.

[33] 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.

[34] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[35] ——, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[37] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[38] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.