Decentralized Multi-Agent Pursuit using Deep Reinforcement Learning

Cristino de Souza Jr[1,2*], Rhys Newbury[3*], Akansel Cosgun[3], Pedro Castillo[1], Boris Vidolov[1], Dana Kulić[3]

***Abstract*—Pursuit-evasion is the problem of capturing mobile targets with one or more pursuers. We use deep reinforcement learning for pursuing an omnidirectional target with multiple, homogeneous agents that are subject to unicycle kinematic constraints. We use shared experience to train a policy for a given number of pursuers, executed independently by each agent at run-time. The training uses curriculum learning, a sweeping-angle ordering to locally represent neighboring agents, and a reward structure that encourages a good formation and combines individual and group rewards. Simulated experiments with a reactive evader and up to eight pursuers show that our learning-based approach outperforms recent reinforcement learning techniques as well as non-holonomic adaptations of classical algorithms. The learned policy is successfully transferred to the real-world in a proof-of-concept demonstration with three motion-constrained pursuer drones.**

I. Introduction

Pursuit-evasion is the problem of capturing targets with one or more pursuers, with applications in robotics such as catching of a rogue drone or a ground target. With multiple-pursuers, decentralized systems are beneficial to avoid single points of failure. Classical algorithms for decentralized multi-agent pursuit [1]–[3] often assume omnidirectional pursuers, derive the local interaction rules from simple geometry and do not learn or adapt to evader behavior. For multi-agent teams consisting of wheeled robots or fixed-wing airplanes, the non-holonomic kinematic constraints on the motion also need to be considered. To our knowledge, decentralized multi-agent pursuit subject to non-holonomic constraints has not been studied extensively by classical approaches in the literature. Deep Reinforcement Learning (DRL) has also been successfully applied to multi-agent pursuit-evasion [4]–[7], however, most approaches to date did not consider real-world limitations such as local measurements and non-holonomic motion constraints and did not offer a thorough analysis of the system on operational metrics.

We propose a DRL approach to multi-agent pursuit. We consider a decentralized scenario in which non-communicating agents independently decide on their own actions based on local information. While our approach applies to any such system, in this paper, we focus on the specific scenario of capturing a finite speed but faster evader with multiple, non-holonomic pursuers in a bounded arena without obstacles. We treat pursuers as homogeneous agents and use shared experience to train a single policy executed independently by each agent at run-time. We use Twin Delayed Deep Deterministic Policy Gradient (TD3) [8], a state-of-the-art DRL algorithm that was successfully applied to other domains [9], [10], with a state representation that encapsulates relative positional information of neighboring agents as well as the target and use a group reward structure that encourages good formations. During training, curriculum learning is applied to start with an easier version of the problem and gradually learn the task with increasing difficulty. In simulation experiments, we compare our approach to three state-of-the-art approaches, two classical methods and a DRL method. They are evaluated in terms of the capture rate and average timesteps to capture. We conduct further analysis on the effect of the number of agents, arena size, as well as using variable linear speed, curriculum learning, and formation score as part of the reward function. The trained policy is demonstrated in a proof-of-concept physical system with three pursuer drones subject to non-holonomic motion constraints.

The organization of this paper is as follows. After reviewing the relevant literature in Sec. II, we define the problem of interest in Sec. III. Our multi-agent DRL method is presented in Sec. IV. We detail the experimental procedure in Sec. V and present simulation results in Sec. VI. Finally, we describe the real-world drone implementation in Sec. VII before concluding with a brief discussion in Sec. VIII.

II. Related Work

A. Multi-agent pursuit

Solutions to the pursuit-evasion problem either assume the 'worst-case' adversary with infinite speed and complete awareness of the pursuers, or average-case behaviors [11]. Although single-agent pursuit-evasion is studied extensively in the literature [11], [12], its extension to multi-agent systems still remains an open problem [13], of interest in biology [13], physics [1], [3], and engineering [2], [14].

Non-learning pursuit methods can be organized into deterministic and heuristic solutions. Deterministic methods attempt to solve the problem with traditional mathematics tools, such as pursuit curves analysis [12] and differential games [2], [14]. Pursuit curve analysis formulates the trajectory of the pursuer analytically using differential equations. The system can then be solved to find the conditions of capture. Although the problem can be stated simply, closed-

[1]Univ. de Tech. de Compiègne, CNRS, Heudiasyc, CS 60319, France cristino.de-souza-junior@hds.utc.fr

[2]Technology Innovation Institute, Abu Dhabi, UAE

Department of Electrical and Computer System Engineering, Monash University, Australia

* Authors contributed equally to this paper.

form analytical solutions are hard to obtain even using simplistic assumptions such as constant velocities and linear trajectories. Differential games formulate pursuit as a game, where the players must optimize an objective function, often the episode duration. However, it is challenging to define an appropriate objective function with increasing problem complexity, such as with a larger number of agents or motion constraints.

Heuristic solutions [1], [3], [15] inspired by behavior-based decentralized approaches [16], use computational simulations that aim to find emergent group behavior based on local observation [16], [17]. Angelani [3] proposed modeling up to a hundred autonomous pursuers as particles based on [16]. Muro [15] proposed small-scale hunter strategies with up to 5 agents, arguing that the behavior observed in wolf-pack hunts can be simulated with simple rules. Janosov [1] re-examined the concepts of the Vicsek particle model [3] in a more realistic scenario considering delays, accelerations, prediction and a faster target. However, these works assume access to information about the evader's position and velocity, which is not directly available from onboard sensors.

The real-world applicability of many of the above approaches is limited due to their assumptions in observation and actuation. [3] and [1] both consider an omnidirectional particle model, which is not directly transferable to many robotic platforms. In contrast, our observation model is expressed relative to each agent, which can be found directly using onboard sensors, such as LiDARs or depth cameras [18]. We also consider a non-holonomic kinematic model, suitable for car-like mobile robots or fixed-wing airplanes.

B. RL In Pursuit

The pursuit-evasion game is a highly studied task in multi-agent RL [19]–[21]. However, most approaches apply only to omnidirectional agents, which cannot be easily transferred to real robotic applications without a loss in performance. Lowe [4] presented an approach for multi-agent RL using an adapted version of an actor-critic algorithm extended to multi-agents. Their approach on the pursuit-evasion game with omnidirectional agents outperformed Deep Deterministic Policy Gradient (DDPG) [22]. Xu [7] considered pursuit-evader games with non-holonomic agents, where new agents can join the game. They adapted Bi-directional Recurrent Neural Networks [23] and DDPG. However, they only consider a situation with 3 and 5 agents. Furthermore, the observation also assumes global information about other agents, limiting the applicability to real-world situations. A few pursuit-evasion works consider non-holonomic constraints [6], [24]. Hüttenrauch [6] studied multi-agent pursuit-evasion systems by considering the agents as interchangeable and the exact number irrelevant. They create a new state representation based on mean embedding of distributions. Their work focuses on scalability and shows that their system can operate with up to fifty agents.

[25]–[27] learn a policy directly from images in a pursuit-evasion scenario with one chaser and one evader. These policies are then successfully transferred to a real-world scenario and show good performance.

C. Curriculum Learning

Curriculum learning [28] is a learning paradigm to help improve speed of convergence and reduce local minima by gradually increasing the complexity of training data. This learning paradigm has been been widely used for RL [29], [30] and deep learning [31], and shown to solve problems which were previously considered intractable [5].

Our work, while borrowing ideas from both classical and learning-based methods, focuses on using DRL to improve the pursuit performance and consider operational metrics such as capture success rate and the average time to capture. Furthermore, we propose a method that is suitable for sim-to-real policy transfer with realistic observation models and non-holonomic constraints. To our knowledge, we are the first to demonstrate a real-world pursuit-evasion implementation with multiple pursuers using a DRL policy.

III. The pursuit-evasion scenario

Our pursuit-evasion problem consists of multiple homogeneous, slower pursuers chasing a single, faster target. The goal for pursuers is to move as a group so that the freedom of the target is constrained to the point where one of the pursuers 'captures' the target in the shortest time possible. We consider a trial successful if, at any point during the trial, the distance between the evader and at least one of the pursuers is less than a given collision radius ($d_{i,T} < d_{cap}$). If the target is not captured within a fixed time period $T_{timeout}$, then a timeout occurs, and the trial is considered unsuccessful. Collisions between pursuers do not result in a failure, however, it is discouraged within the reward function (Sec. IV-C), an important feature for collision avoidance in real-world implementation (Sec. VII). The pursuer motions are subject to non-holonomic kinematic constraints, while the evader is omnidirectional and thus not subject to such constraints. We adopt a unicycle model for each pursuer i:

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

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

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

where (x,y) is the position, ψ is the heading angle, v is the linear velocity and ω is the angular velocity. We will first assume that all agents have a constant v and the only controllable variable is ω with limits $\omega_{min} \leq \omega \leq \omega_{max}$, following the classical formulation approach [1]. Later we will relax this assumption to allow the pursuers to vary both their angular and linear velocities. We approximate the equations as a discrete model. The environment is a circular arena with a radius of R_{arena} and without any obstacles in it. Neither the pursuers nor the target can get out of the arena: if they take an action that would end up outside R_{arena}, their position is updated to be on the nearest arena border.

We assume that the pursuers can differentiate the agents from the target. We further assume that each pursuer is equipped with a sensor that provides the relative position of the target as well as every other pursuer. All sensors provide ground truth data without noise, unaffected by oc-

clusions. The state representation of each agent is detailed in Sec. IV-B.

IV. Deep Reinforcement Learning for Pursuit

We formulate the task for a single pursuer to be a Markov Decision Process (MDP) defined by tuple $\{\mathcal{S}, \mathcal{A}, R, \mathcal{P}, \gamma\}$ where $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$, $r_t \in R$ are state, action and reward observed at time t, $\mathcal{P}$ is an unknown transition probability from s_t to s_{t+1} taking a_t, and γ is a discount factor. The DRL goal is to maximise the sum of future rewards $R = \sum_{t=0}^{T} \gamma^t r_t$, where r_t is provided by the environment at time t. Actions are sampled from a deep neural network policy $a_t \sim \pi_\theta(s_t)$, where a_t is the angular velocity ω of an individual pursuer, which is saturated to be in the interval $[\omega_{min}, \omega_{max}]$.

A. Multi-Agent Deep Reinforcement Learning

As the Deep RL algorithm, we use Twin Delayed Deep Deterministic Policy Gradient approach (TD3) [8], which is an improvement over DDPG [22], designed to reduce the overestimation of the value function. We consider all agents to be homogeneous which allows us to use shared experience to train all agents. This allows the agents to train faster, as well as gathering more information from every step in the environment. All agents are governed with the same policy, however, at each time step the agents use their local observations to individually take actions, resulting in a decentralized system. For each number of agents we train a different policy, which results in a total of n_{max} policies, where n_{max} is the maximum number of pursuers we analyze in this paper. This was because the length of the state representation changes based on the number of pursuers in the game.

B. State Representation

The state of a pursuer i, assuming a total of n pursuers, is given by $s_i = [\psi_i, \dot{\psi}_i, s_{i,T}, s_{i,1}, s_{i,2}, .., s_{i,n-1}]$, where ψ is the heading with respect to a fixed world frame, $s_{i,T}$ is the state of the target relative to pursuer i and $s_{i,j}$ is the state of pursuer j relative to pursuer i. (Time indices are dropped for the sake of clarity). The relative state of the target with respect to pursuer i is $s_{i,T} = [d_{i,j}, \dot{d}_{i,T}, \alpha_{i,T}, \dot{\alpha}_{i,T}]$ and the relative state of pursuer j with respect to pursuer i ($i \neq j$) is $s_{i,j} = [d_{i,j}, \alpha_{i,j}]$, where $d_{i,j}$ is the Euclidean distance between pursuers i and j and $\alpha_{i,j}$ is the heading error defined as the angle between the heading of pursuer i, and the vector between i and j, as shown in in Fig 1. The state representation consists of a total of $2n + 4$ variables, which scales linearly with the number of pursuers n.

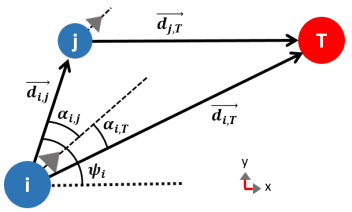

Fig. 1: The state space for each agent i. T denotes the target and j denotes another agent

An important consideration is how the $s_{i,j}$ are ordered in the state representation s_i. A straightforward way would be to assign a unique identifier to each pursuer and always represent them in the same order. However, this leads to inefficiencies in learning as neural networks typically are not permutation-invariant when operating on sets [32]. To illustrate this, consider swapping the poses of two pursuers with everything else being the same. With unique identification ordering, the resulting state would be different than the original, whereas since we have homogeneous agents, the state should not change. To tackle this problem we assign j values for each observation by sorting each other pursuer with respect to their relative angle α.

The observation of the pursuers was designed to be easily applicable on real platforms and is used commonly in conventional single-agent pursuit [12]. The agent observations do not require localization with respect to a global frame, as local observations d and α are not referenced in global coordinates and can be extracted using onboard sensors such as laser scanners or cameras. Recent work [18] demonstrates the feasibility of acquiring measurements such as range and the relative angle between the pursuers, using only embedded sensors in drones. For the heading ψ, a directional sensor would be needed, such as a magnetometer.

C. Reward Structure

At each time step, each agent individually receives a reward designed to incentivize the capture of the evader and encourage a good formation of pursuers. The reward function is:

$$r_i = \begin{cases} r_{captor}, & \text{if } d_{i,T} \leq d_{cap} \\ r_{helper}, & \text{if } d_{j,T} \leq d_{cap}, \exists j \neq i \\ -w_q\, q - w_d\, d_{i,target}, & \text{otherwise} \end{cases}$$

At each step that the target is not captured, each and every agent receives a negative reward that is a weighted linear combination of an individual reward (its distance to the target $d_{i,target}$) and a group reward (q-score [13], which we call the formation score in our work). The formation score is a scalar number in the range $[0, 2]$, which provides a metric for evaluating the fitness of a formation of the pursuers (lower is better). The formation score (q) is defined as:

$$q = \frac{1}{n} \sum_{i=1}^{n} (\hat{\boldsymbol{d}}_{\mathbf{0T}} \cdot \hat{\boldsymbol{d}}_{\mathbf{iT}} + 1) \tag{2}$$

where $\hat{\boldsymbol{d}}_{\mathbf{iT}}$ denotes a unit vector pointing in the direction from agent i and the target, and n is the number of agents. In this equation, the closest agent to the target is defined as agent 0. The formation score encourages agents to spread around the target (i.e., approach the target from different directions) and penalizes the angular proximity between agents. We introduce the formation score when there are at least two agents, as the formation score for one agent is not defined. Early experiments with the formation score showed that when the formation is the only component of the reward, pursuers would only form a good formation but would not make an attempt to capture the evader. To avoid this situation, we penalize the distance to the target, which helps encourage the agents to get close to the evader while being in a good formation. The weights w_q and w_d were

chosen such that when the agents are close to the target, the reward is dominated by the formation score, encouraging good formation. However, when the agents are far from the target, the reward is dominated by the distance to the target, encouraging the agents to move closer to the evader. We analyze the effect of the formation score on pursuit performance in Sec. VI-G.

If the target is captured at a time step, then the pursuer who captures the evader receives the reward r_{captor}, while the rest of the agents receive r_{helper}, such that $r_{captor} > r_{helper}$. This encourages each pursuer to go for the final capture while encouraging collaboration.

D. Curriculum Learning

We apply a curriculum for learning by starting from an easier version of the task and gradually increasing the difficulty until the actual difficulty is achieved. There are two main factors in determining the difficulty of a pursuit-evasion game: the relative speed of the target with respect to the pursuers and the capture radius d_{cap}. We vary the capture radius by starting from a large radius (so it is easier to capture the target), then gradually making it smaller. This encourages agents to not adopt a straightforward chasing tactic at the beginning of learning but to form more sophisticated behaviors, which could be transferred to smaller capture radii. We also experimented with reducing the pursuer speed to reduce the difficulty of the task, however, we found that pursuers mostly learned to follow the evader directly, and it was harder to explore more sophisticated behaviors afterward.

Curriculum learning helps exploration, especially during the early stages of learning, because early, on it helps the pursuers to capture the target, which would take a longer time in the actual, and more difficult scenario. This helps alleviate the sparse reward problem, which is a well-known challenge in DRL [33]. We analyze the effect of curriculum learning on the pursuit performance in Sec. VI-F.

V. Simulation Experiments

We use the following simulation parameters: $T_{timeout} = 500$ iterations, rewards r_{captor}, r_{helper}; the weights w_q and w_d were set to 10, 100, 0.1, 0.002 respectively. The capture radius R_{cap} was set to 30 pixels for testing, although this value was varied as part of curriculum learning. For fixed linear velocity cases, the speed of the pursuers v_p was 10 pixels per timestep, while the target's speed v_T varied from 0 to 20 pixels per timestep. The maximum angular rate ω_{max} was fixed at $\pi/10$ per timestep. The number of pursuers n varied between 1 and 8, initialized at random positions within a circular area with a radius of 100 pixels, while the evader was initialized at a random position between the arena boundary and an inner circle with a radius of 300 pixels. Arena radius $R_{arena} = 430$ pixels, except for the results in Sec VI-C. We assume each pursuer can observe all other pursuers, except for in Sec. VI-E.

A. Evader Behaviors

We implemented two behavior modes for the evader: Fixed Paths and Repulsive. For both, we vary the relative speed of the evader from 0.8 to 2 times the pursuer speed, with a step size of 0.2. We conduct 100 trials for every speed level.

Fixed Paths: We propose a benchmark where the evader follows three predefined paths, as shown in Fig. 2.

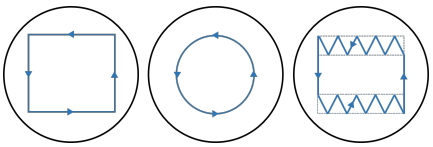

Fig. 2: Paths used by the evader for the fixed paths benchmark.

Repulsive: We use a potential field method with repulsive forces only to find a motion vector. Each pursuer exerts a repulsive force in the direction of the vector between the pursuer and the evader. The arena boundary also exerts a force so that the evader can avoid the wall. These forces decrease proportionally to the distance squared. The resultant vector is calculated by:

$$\vec{v} = \sum_j \left(\frac{\vec{a}_j - \vec{e}}{d_j^2}\right) + \frac{\hat{\gamma} R_{arena} - \vec{e}}{d_w^2} \tag{3}$$

where $\vec{e}$ is the current position of the evader, $\vec{a}_j$ is the position of agent j and d_j is the distance to a_j. γ is the direction of the agent to the closest point on the wall, $\hat{\gamma}$ is the unit vector rotated by an angle of γ and d_w represents the distance of the agent to the wall.

B. Baseline methods

We implemented three baseline methods: Two classical (Janosov [1] and Angelani [3]) and a DRL (Hüttenrauch [6]) method. The approach by Hüttenrauch [6] was trained on our simulation environment using their *communication* set. This observation vector included relative angle, the distance towards the target, and the heading of each of the pursuers. It should be noted that this observation set included more information than our model. Furthermore, this information (orientation of neighbors) would likely require explicit communication in a real-world application since it can not be easily estimated from current embedded sensors. We trained the policy for 4 million timesteps (same as our approach, around 4 times as long as their original paper) without curriculum learning (no curriculum learning was used in their original work) and presented the best simulation results.

We adapted the classical methods to use a non-holonomic model. As these methods are designed for omnidirectional agents, they are not directly comparable to our solution. Therefore, we convert the outputs of these models into the unicycle model using the following equations:

$$\psi_{desired} = \arctan \frac{dy}{dx} \tag{4}$$

$$\omega = K * (\psi - \psi_{desired}) \tag{5}$$

The omnidirectional models have two outputs, dx and dy, which are the velocity in the x and y direction respectively. From this, we find the desired heading $\psi_{desired}$ of the omnidirectional controller. We use a P controller on the error between the desired heading $\psi_{desired}$ and current heading ψ. We tune gain K such that the number of captures is maximized in simulation trials.

VI. Simulation Results

In this section we evaluate our approach against baseline algorithms presented in Sec.V-B. In Sec.VI-A we analyze the performance on a fixed paths benchmark, followed by analysis with a repulsive evader model. We conduct the following analyses on the capture performance: effect of number of pursuers (Sec.VI-B), arena size (Sec.VI-C), relative evader speed (Sec.VI-D), scaling number of agents without retraining (Sec. VI-E), use of curriculum learning (Sec.VI-F), and use of formation score (Sec.VI-G). Finally, we qualitatively describe the emergent behaviour of the multi-agent system in Sec.VI-H.

A. Fixed Paths

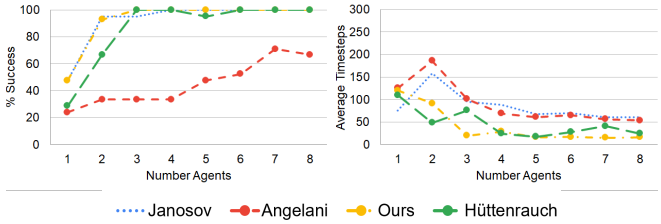

Fig. 3: Success rate and the average timesteps using the fixed paths benchmark, for different number of agents

In these experiments, the evader followed fixed paths, as explained in Sec.V-A. The pursuers were trained on the repulsive evader only but were tested on the fixed paths benchmark. The capture rate and the average number of steps with respect to the number of pursuers are shown in Fig. 3. While Janosov [1] get above 95% on the fixed paths benchmark for $N > 2$, Angelani [3] does not perform well on this benchmark. Both our approach and Hüttenrauch [6] complete the task successfully for $n \geq 3$. The average timesteps to capture for both DRL approaches is significantly lower than the other approaches for the fixed paths benchmark, likely because the classical algorithms attempt to corral the target. In contrast, the DRL based approaches tended to be more aggressive and intercept the evader quickly along the fixed paths.

B. Effect of the Number of Pursuers

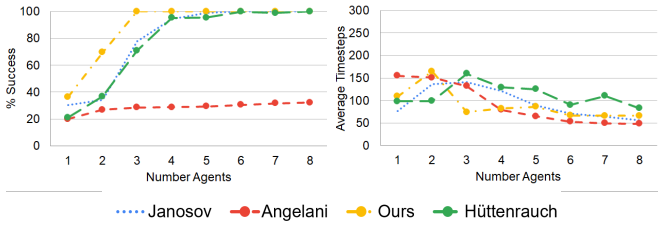

Fig. 4: Success rate and the average timesteps taken for the repulsive evader, for different number of agents.

The success rate and average timesteps to capture the repulsive evader with respect to the number of pursuers are shown in Fig. 4. Our approach outperforms the competing approaches in cases with a lower number of agents in terms of the success rate. Hüttenrauch [6] performs well with a larger number of agents, however, struggles with fewer agents. Janosov [1] completes the task with a success rate above 94% for $n \geq 4$. However, it does not perform well with fewer than four agents. Angelani [3] does not perform well in this benchmark. Furthermore, with one or two agents, all methods showed poor performance, as it is difficult to chase a faster evader with few agents. Our approach takes more time on average to complete the capture compared to [1].

C. Effect of arena size

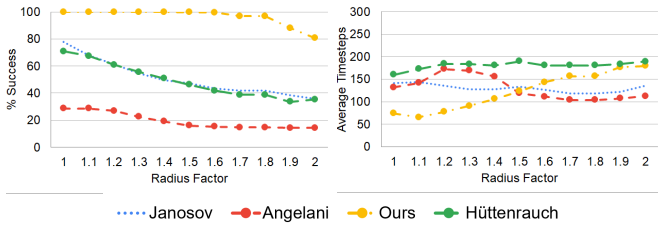

Fig. 5: Success rate and average timesteps to capture with respect to the multiplicative radius factor over R_{arena}. Experiments are conducted with 8 pursuers and the repulsive evader model.

As we showed in Sec. VI.B, with an increasing number of agents, the task decreases in difficulty for a fixed arena size. Therefore, with a larger number of agents, most approaches can perform the task successfully. The pursuit-evasion game is played in an obstacle-free arena, but the agents can use the arena boundaries to constrain the evader movements. In this section, we investigate the effect of larger arena sizes using $n = 3$ agents. We do not retrain our agents on the new arena size but use the model trained on the original radius size r_{area}. As shown in Fig. 5, our approach comfortably outperforms the other approaches in terms of success rate as the arena size increases. This shows that the learned policy can generalize to larger arenas. However, the average number of timesteps to capture for our approach is consistently higher than other approaches. We attribute this result to using only the successful captures to obtain the average number of timesteps: our approach can likely find solutions to more difficult problems at the expense of increased average duration to secure the capture.

D. Effect of Relative Evader Speed

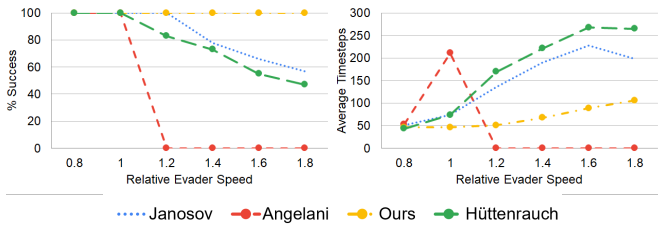

Fig. 6: Success rate and the average number of timesteps to capture is shown with respect to the ratio of evader speed to pursuers' speed. Experiments are run with three agents. Our approach achieves 100% accuracy for all cases in this analysis.

In this section, we examine at the effect of relative evader speed on capture success, for $n = 3$ pursuers. As shown in Fig. 6, while our approach had 100% success rate at all speed levels, all other methods had a drop-off at faster evader speeds. As expected, our approach was the fastest in capturing the target (only considering successful episodes).

E. Scaling number of agents without retraining

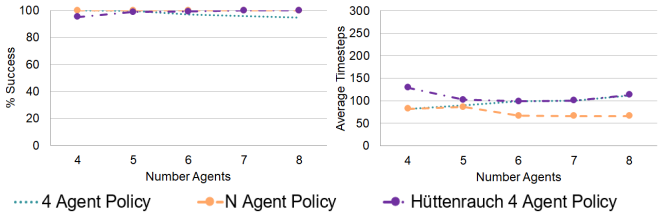

Fig. 7: The success rate and average timesteps on the repulsive evader benchmark, comparing our full approach (trained on the appropriate amount of agents), the approach by Hüttenrauch [6] which was trained on four agents and our approach which was limited to only see the closest four agents during execution.

In this section, we examine the scalability of our approach to an environment with more agents than the network was trained on. For the 4 agent policy in Fig. 7, the agents are trained in the 4 agent setting. At test time, the number of agents increases (as indicated on the horizontal axis), but each agent can only observe their 4 closest neighbors. In contrast, both the N agent policy and Hüttenrauch [6] were given information about all pursuers. The success of our approach decreases with a larger number of agents, by around 5% when there are more agents, as the agents cannot coordinate fully. The other approaches reach 100% for a larger number of agents. The four agent policy requires a larger average number of timesteps to capture the target compared to a more specialized policy, and this performance penalty increases as more agents appear in the environment compared to the number of agents during training. However, the 4-agent policy tends to perform similarly to [6], which was designed for scalability.

F. Effect of Curriculum Learning

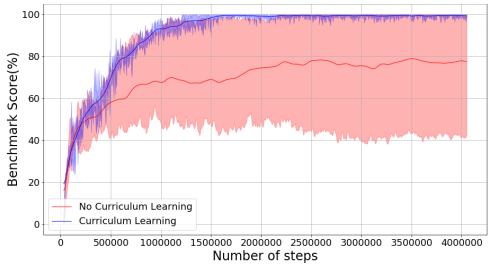

Fig. 8: Comparison of capture success rate with and without curriculum learning, with respect to the number of training steps. With curriculum learning, the benchmark scores are much higher and more consistent.

Fig. 8 compares the effect of using our curriculum learning strategy described in Sec IV-D, for $n = 3$ agents. The network was trained 3 times. At regular intervals, we stop training and evaluate the policy on the repulsive evader benchmark. The results show that curriculum learning is beneficial for capture performance: it converges to about 100% success rate after 1.5 million training steps, whereas without curriculum learning, the average success rate was below 80% even at 4 million training steps. Furthermore, the performance with curriculum learning was much more consistent, as evidenced by the low variance among the three runs.

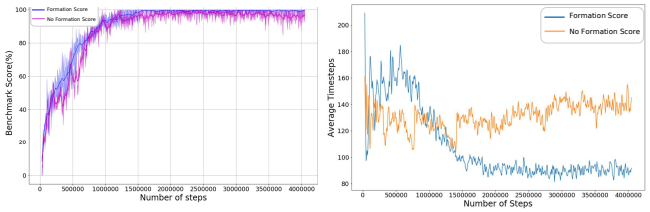

Fig. 9: Using a formation score as a dense reward results in more captures, in less number of timesteps on average.

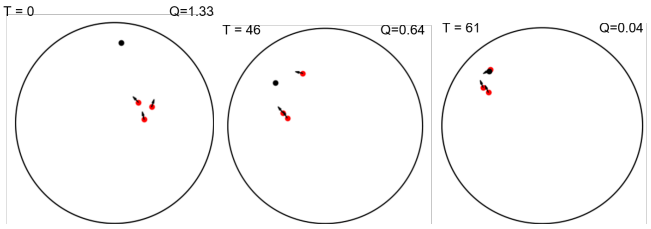

Fig. 10: "Split Up" strategy learned by three pursuers. Timestep (T) and formation scores (Q) are shown at three snapshots. The target is shown as the black circle. The agents start in a random direction (Left), push the agent towards the wall splitting into two groups (Middle) before going for the capture (Right).

G. Effect of Formation Score in Reward Function

As described in Sec. IV-C, we provide a partial reward at every timestep in order to encourage good formations. We analyze the effect of supplying this dense reward component to each agent. Fig. 9 compares the evolution of the capture performance with and without the formation score with respect to the number of training steps. These experiments were conducted with $n = 3$ agents. As shown in the Fig. 9, benchmark scores were slightly higher when the formation score is used as part of the reward. Furthermore, when the formation score is used, the average capture time for successful episodes is decreased.

H. Qualitative Analysis of Emergent Behavior

We observe two interesting learned emergent behaviors that often lead to successful captures: ambushing and splitting up.

Fig.10 shows the splitting up behavior with $n = 3$ agents. This behavior was more common with a smaller number of agents. The agents tend to split up into two groups, trying to push the evader into a wall before attempting to block the two opposite directions. This tactic works well as the evader is backed against the wall and has limited room to escape.

Fig. 11 shows the ambushing behavior with $n = 8$ agents. This behavior was more common with a larger number of agents. The agents tend to form a circle, attempting to move

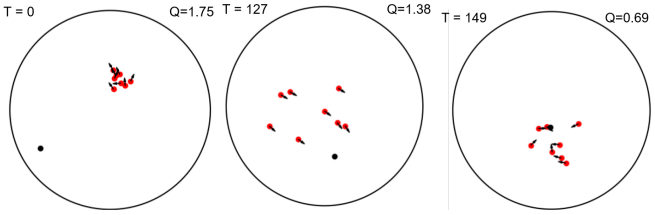

Fig. 11: "Ambush" strategy learned by 8 pursuers. Timestep (T) and formation scores (Q) are shown at three snapshots. The target is shown as the black circle. The agents start in random directions (Left), move as a circle (Middle) ambush the target and capturing it (Right)

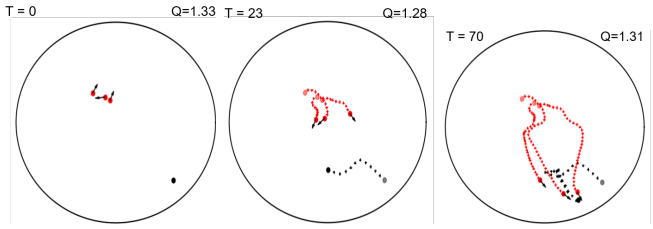

Fig. 12: "Stalking" strategy learned by three pursuers with velocity. Timestep (T) and formation scores (Q) are shown at three snapshots. The target is shown as the black circle. The agents start in random directions (Left), slow down and angle themselves such that they can surround the target (Middle) and capture it (Right).

such that they can surround the evader and then approach from all directions. This seems to be a distinct behavior from the 'Split Up' behavior, as the agents do not use the walls as much, preferring to surround the evader as a pack, similar to pack behaviors observed in Muro's work [15]. When the agents execute this strategy to trap the evader, it is often very difficult for the evader to escape.

I. Variable Linear Velocity

Previous sections considered constant linear speed and variable angular speed for pursuers, primarily because this is an assumption for classical algorithms. We now consider the more general case, where agents can also vary their linear velocity between 0 and v_p. Therefore, we train the network with two outputs: linear and angular velocity. We consider the 3 agent scenario, in which the agents achieve 100% capture rate with and without velocity control, in both the fixed and reactive benchmarks.

The agents often displayed a "Stalking" strategy as illustrated in Fig. 12. With this strategy, the agents move towards the target before slowing down and waiting until the opportunity presents itself to capture the evader. This behavior may have analogs in nature, where pursuers will stalk their prey and position themselves such to maximize the likelihood of attack [34].

VII. Demonstration on Drones

We demonstrate our approach on three autonomous quadcopter drones pursuing a human-controlled target drone. Drones are typically modeled as holonomic vehicles; however, under certain conditions can act as non-holonomic vehicles (e.g. high-velocity maneuvering such as in [35]). The use of drones, classically a holonomic vehicle, will also allow us to compare classical holonomic works (such as [1] and [3]) to our work on the same platform in future work. Furthermore, by constraining the motion, there is an interesting property while considering a frontal camera as a sensor: the drone will move only in the direction of the field of vision, tightly coupling the perception to the movement.

We use direct sim-to-real transfer, where the policies used to control the drone behaviors are trained in the simulation environment described in Sec.III. The input to the actor network was the normalized relative positions of the target and neighbors. The policy output for each agent is a single number, the angular velocity. We artificially constrain the motions of the pursuer drones to emulate a system with 2D agents with the unicycle kinematic model: 1) Each drone is constrained to a fixed height. The pursuer drones are at the same height, however, the evader is constrained to a different altitude, which allows the pursuers to get closer to the evader than if they were at the same altitude. 2) Angular input velocities generated by our approach are converted to input signals for low-level attitude control.

A low-level, non-linear controller runs onboard each drone for tracking velocity reference signals: it takes the requested linear and angular velocities as input and calculates the torque and thrust for the quadcopter. Details for the low-level controller implementation can be found in [36]. The controller is also responsible for stabilizing the quadcopter's altitude and maintaining safety. It implements collision avoidance, constrains the drones to a circular arena of 3m radius, and limits the maximum speed to 1.2m/s.

The experiments took place in an indoor flight arena equipped with a motion capture system, which was used to track the pose of all agents. A centralized motion capture system was used in this implementation due to the ease of prototyping; however, all information needed by our algorithm can be captured using onboard sensors, similar to [18]. Parrot AR Drone 2 was used for all drones. The behavior of each pursuer was calculated on a local computer and transmitted wirelessly to the drone at 20Hz. Details for the hardware implementation can be found in [37]. This setup between drones and a local computer is reminiscent of a centralized system; however, our methodology is also suitable for a decentralized system if onboard processors on each drone can be used for neural network inference. For a decentralized system, each drone would also need to be equipped with a directional sensor such as a magnetometer.

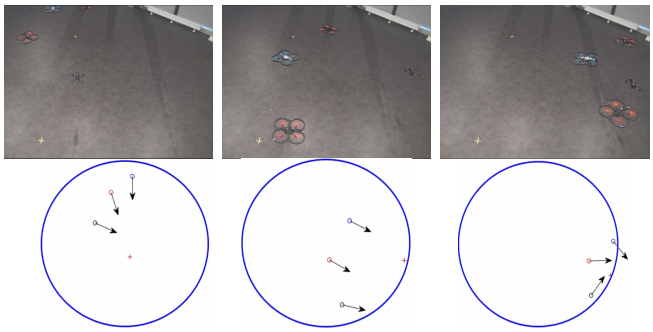

Fig. 13: Snapshots from real-world demonstration with 3 motion-constrained drones. We can see the emergence of the "Split Up" behavior: The pursuers are initially close to each other, then spread toward the target, and finally, regroup by cornering the target.

Snapshots from a successful demonstration can be seen in Fig.13. During the training of the networks, we do not consider the non-linear dynamics of the quadcopter. Direct sim-to-real transfer is possible because DRL policy provides high-level navigation decisions, while a lower-level controller manages the attitude of the drone and assures safe navigation.

VIII. Conclusion

We proposed a DRL approach to multi-agent pursuit with non-holonomic pursuers and an omnidirectional target. We consider a decentralized system where each agent individually decides on its own actions using local observations only. Simulation experiments show that our approach, applied

to non-holonomic agents, outperforms the state-of-the-art in heuristic multi-agent pursuit methods and a recent DRL based approach. Our results show that multi-agent pursuit benefits from curriculum learning and a reward based on agent formation, which we borrowed from the group-pursuit literature. In a demonstration with drones constrained to a fixed height and governed by the unicycle kinematics model, we demonstrate that direct sim-to-real transfer is possible.

A limitation of the current work is the need to train a network for each number of observable agents. This can be partially mitigated by using the same network and fixing the number of observable pursuers, as demonstrated in Section VI-E. Furthermore, it could be interesting to learn fixed-size state representation for neighboring agents by using deep sets [32], mean embeddings (similar to [6]) or making use of Graph Neural Networks [38]. Other interesting directions of future work include exploring scenarios with numerous evaders, integrating smarter evader strategies by using the idea of safe-reachability [39], [40], or implementing the evader as an RL agent and training both the evader and pursuer simultaneously similar to [41].

To enable more realistic applications, we also aim to extend the method from planar to 3D motion in the future. This will require careful consideration of the angular representation to avoid representational singularities. Furthermore, we aim to consider more realistic kinematic and perception models and include other constraints such as more unstructured environments with obstacles and varying arena sizes, a limited field of view, explore sim-to-real transfer further, and real-world applications to non-holonomic robots such as wheeled robots.

REFERENCES

[1] M. Janosov, C. Virágh, G. Vásárhelyi, and T. Vicsek, "Group chasing tactics: how to catch a faster prey," *New Journal of Physics*, (2017).

[2] J. Li, M. Li, Y. Li, L. Dou, and Z. Wang, "Coordinated multi-robot target hunting based on extended cooperative game," in *IEEE ICIA*, 2015.

[3] L. Angelani, "Collective predation and escape strategies," *Phys. Rev. Lett.*, vol. 109, no. 11, 2012.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *NIPS*, 2017.

[5] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *AAMAS*, 2017, pp. 66–83.

[6] M. Hüttenrauch, A. Šošić, and G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *J. of Machine Learning Research*, 2019.

[7] L. Xu, B. Hu, Z. Guan, X. Cheng, T. Li, and J. Xiao, "Multi-agent Deep Reinforcement Learning for Pursuit-Evasion Game Scalability," in *Lect. Notes Electr. Eng.*, 2020.

[8] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv arXiv:1802.09477*, 2018.

[9] J. Matas, S. James, and A. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *CoRL*, 2018.

[10] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *IEEE ITSC*, 2019.

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

[12] N. A. Shneydor, *Missile guidance and pursuit: kinematics, dynamics and control*. Elsevier, 1998.

[13] A. Kamimura and T. Ohira, *Group Chase and Escape: Fusion of Pursuits-Escapes and Collective Motions*. Springer, 2019.

[14] H. Huang, W. Zhang, J. Ding, D. M. Stipanović, and C. J. Tomlin, "Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers," in *CDC-ECC*, 2011.

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

[16] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett*, vol. 75, no. 6, p. 1226, 1995.

[17] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Automat. Contr.*, 2006.

[18] A. Carrio, J. Tordesillas, S. Vemprala, S. Saripalli, P. Campoy, and J. P. How, "Onboard detection and localization of drones using depth maps," *IEEE Access*, vol. 8, pp. 30 480–30 490, 2020.

[19] S. F. Desouky and H. M. Schwartz, "Q(λ)-learning adaptive fuzzy logic controllers for pursuit-evasion differential games," *International Journal of Adaptive Control and Signal Processing*, 2011.

[20] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE SMC Part C*, vol. 28, no. 3, pp. 338–355, 1998.

[21] M. D. Awheda and H. M. Schwartz, "The residual gradient FACL algorithm for differential games," in *CCECE*. IEEE, 2015.

[22] 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.

[23] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process*, 1997.

[24] M. Kothari, J. Manathara, and I. Postlethwaite, "Cooperative multiple pursuers against a single evader," *JINT*, vol. 86, 06 2017.

[25] B. Vlahov, E. Squires, L. Strickland, and C. Pippin, "On developing a uav pursuit-evasion policy using reinforcement learning," in *IEEE ICMLA*, 2018, pp. 859–864.

[26] A. Bonnet and M. A. Akhloufi, "UAV pursuit using reinforcement learning," in *Unmanned Systems Technology*, 2019.

[27] B. Vlahov, E. Squires, L. Strickland, and C. Pippin, "On developing a uav pursuit-evasion policy using reinforcement learning," in *IEEE ICMLA*, 2018, pp. 859–864.

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[29] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic curriculum learning for deep rl: A short survey," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. IJCAI, 7 2020, pp. 4819–4825, survey track.

[30] B. Tidd, N. Hudson, and A. Cosgun, "Guided curriculum learning for walking over complex terrain," *arXiv preprint arXiv:2010.03848*, 2020.

[31] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," *arXiv preprint arXiv:1704.03003*, 2017.

[32] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NIPS*, 2017.

[33] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *IEEE ICRA*, 2018.

[34] P. Stander, "Cooperative hunting in lions: the role of the individual," *Behavioral Ecology and Sociobiology*, vol. 29, pp. 445–454, 2004.

[35] A. M. Elhennawy and M. K. Habib, "Trajectory tracking of a quadcopter flying vehicle using sliding mode control," in *IEEE IECON*, 2017.

[36] G. Sanahuja, P. Castillo, and A. Sanchez, "Stabilization of n integrators in cascade with bounded input with experimental application to a vtol laboratory system," *Int. J. Robust Nonlinear Control*, 2010.

[37] C. De Souza, P. Castillo, R. Lozano, and B. Vidolov, "Enhanced uav pose estimation using a kf: experimental validation," in *ICUAS*, 2018.

[38] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2019.

[39] A. Pierson, Z. Wang, and M. Schwager, "Intercepting rogue robots: An algorithm for capturing multiple evaders with multiple pursuers," *IEEE RA-L*, vol. 2, no. 2, pp. 530–537, 2017.

[40] "Cooperative pursuit with voronoi partitions," *Automatica*, vol. 72, pp. 64 – 72, 2016.

[41] 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*, 2020.