

# Agile Interception of a Flying Target using Competitive Reinforcement Learning

Timothée Gavin  
IAS

Thales LAS

Rungis, France

timothee.gavin@thalesgroup.com

Simon Lacroix  
RIS

LAAS CNRS

Toulouse, France

simon.lacroix@laas.fr

Murat Bronz  
Dynamic Systems, OPTIM

Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse

Toulouse, France

murat.bronz@enac.fr

**Abstract**—This article presents a solution to intercept an agile drone by another agile drone carrying a catching net. We formulate the interception as a Competitive Reinforcement Learning problem, where the interceptor and the target drone are controlled by separate policies trained with Proximal Policy Optimization (PPO). We introduce a high-fidelity simulation environment that integrates a realistic quadrotor dynamics model and a low-level control architecture implemented in JAX, which allows for fast parallelized execution on GPUs. We train the agents using low-level control, collective thrust and body rates, to achieve agile flights both for the interceptor and the target. We compare the performance of the trained policies in terms of catch rate, time to catch, and crash rate, against common heuristic baselines and show that our solution outperforms these baselines for interception of agile targets. Finally, we demonstrate the performance of the trained policies in a scaled real-world scenario using agile drones inside an indoor flight arena.

**Index Terms**—Reinforcement Learning, Multi-Agent Systems, Interception, Agile Flight

## I. INTRODUCTION

The interception of agile aerial targets using autonomous drones is a challenging and increasingly relevant problem in robotics and security. The increasing presence of unmanned aerial vehicles (UAVs) in unauthorized, restricted airspaces poses significant safety and security risks and has spurred interest in developing effective interception strategies [1]. In particular, scenarios such as airspace protection, infrastructure security, and event safety require the ability to capture or neutralize unauthorized drones with high precision and minimal collateral risk. Deploying interceptor drones equipped with nets is a promising approach, but it demands advanced control capabilities to match or exceed the agility of evasive targets.

Traditional interception methods often rely on accurate models, preplanned strategies, or predictable target behaviour [2]. However, modern quadrotor drones can perform highly dynamic manoeuvres, and will actively evade capture, rendering their trajectories unpredictable and challenging the effectiveness of classical methods [3].

Recent advances in deep reinforcement learning (RL) have demonstrated the potential to learn complex, high-dimensional control policies for drones directly from interaction with the environment. In particular in drone racing, RL-trained policies have achieved superhuman performance in highly dynamic and agile flight tasks [4]. However, the drone racing problem

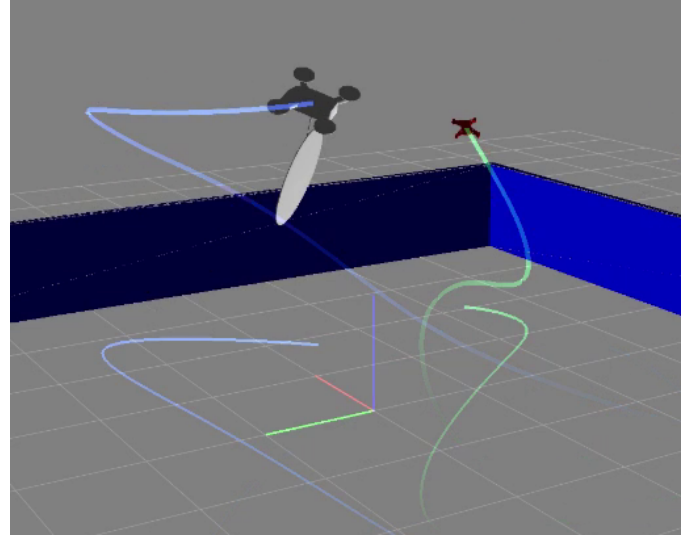


Fig. 1: A competitive reinforcement learning approach to train both a pursuer and an evader drone for agile interception tasks. Both agents learn low-level control policies that enable them to perform dynamic maneuvers in a high-fidelity simulation environment.

typically involve navigating static or slowly moving gates, while interception requires reacting to an adversarial agent that actively attempts to evade capture.

Competitive Multi-Agent RL (MARL) have shown outstanding results in adversarial settings, such as games [5], [6]. In this work, we formulate the agile interception problem as a competitive multi-agent RL task, where both the interceptor (pursuer) and the target (evader) are controlled by independent policies trained using Proximal Policy Optimization (PPO) in a co-evolution framework. Our approach integrates a high-fidelity quadrotor dynamics model, enabling both agents to learn agile, physically realistic manoeuvres from low-level control inputs. Through extensive simulation and real-world experiments, we show that RL training leads to robust and adaptive interception and evading strategies, outperforming heuristic control approaches.

The main contributions of this paper are:

- A competitive MARL framework for agile drone interception, with both pursuer and evader learning from low-level control.
- Integration of a realistic quadrotor dynamics model to enable physically realistic and agile flight behaviors.
- Empirical evaluation demonstrating superior performance over standard baselines in simulation.

The remainder of the paper is organized as follows. Section II reviews related work on interception and agile flight. Section IV and III detail our agile flight simulation environment and our training methodology. Section V presents experimental results and comparisons. Section VI and VII conclude and discuss future directions.

## II. RELATED-WORK ON THE AGILE INTERCEPTION PROBLEM

### A. Agile flight

Agile flights in multi-rotors drones are typically characterized by the ability to perform large-angle manoeuvres, sustain high linear and angular accelerations, maintain precise control near dynamic limits and do so reliably in real-time, often in complex and cluttered environments. Traditionally, achieving such agility relied on trajectory optimization coupled with controllers like Model Predictive Control (MPC), often requiring pre-planned paths and accurate system models [7]. However, these methods can be brittle when faced with unexpected disturbances of real-world flights. Reinforcement Learning (RL) has emerged as a powerful alternative, enabling the learning of complex, non-linear control policies directly from interaction. Research, such as work from [4] and [8], has demonstrated RL's capability to achieve highly dynamic and agile flights for quadrotors, pushing the boundaries of autonomous aerial manoeuvring beyond what traditional methods could easily achieve, particularly in tasks requiring aggressive, near-limits flight.

### B. Interception

Traditional interception methods use heuristic or optimal control methods that often rely on accurate models, pre-planned strategies, or predictable target behaviour. [2]. These approaches have been historically designed for the control of missiles and the interception of fixed-wing manned aircraft. Optimal control methods require an accurate model of the pursuer and the evader to compute interception trajectories [9]. Such model may not be available or may be too computationally expensive for real-time adaptation against unpredictable targets. Heuristic guidance laws, such as Proportional Navigation (PN) or Pure Pursuit, offer computationally simpler alternatives and are widely used in missile guidance. However, these methods often assume relatively simple target manoeuvres and can struggle against highly agile or adversarial evaders. Among recent works, [10] have proposed heuristic methods for drone interception of agile manoeuvring targets, but these still assume a predictable target model. More recently, learning-based solutions, particularly MARL have shown promise for developing complex control

policies in adversarial settings. MARL has been explored for pursuit-evasion games in various contexts, including simulated environments like Multi-Agent Particle Environments (MPE) [11] and initiatives like the DARPA AlphaDogfight Trials, demonstrating the potential to learn sophisticated tactics [12]. For quadrotors, [13] present a RL approach for quadrotor interception in an urban environment, and [14] uses RL to give low-level commands for interception, however both these works consider only the pursuit side, assuming fixed evader behaviours. The closest work to ours we found is [15] which uses RL to train both the pursuer and the evader in a co-evolution framework. However, like most RL approaches [13], they use high-level control inputs (e.g., velocity commands) and simplify the dynamics of the quadrotors, which limits the agility of the learned behaviours. Overall, while RL has demonstrated potential in interception tasks, existing work either ignores the adversarial aspect of a learning evader, or often lacks the integration of highly dynamic capabilities in both the pursuer and the evader.

### C. Our approach

Building upon the challenges highlighted in agile flight and interception, our approach directly addresses the need for highly dynamic capabilities in both the pursuer and the evader. We recognize that interception is fundamentally a dynamic, adversarial interaction requiring controllers that can operate effectively near the physical limits of the hardware. While RL has demonstrated remarkable success in achieving agile flight and has been applied to interception problems, to our knowledge, none have focused on training both an agile pursuer and an agile evader using low-level commands within a competitive RL framework. Our work fills this gap by formulating the problem as a competitive multi-agent RL task where both agents learn physically plausible, agile manoeuvres through interaction in a realistic environment, to foster robust and adaptive strategies.

## III. AGILE FLIGHT SIMULATION ENVIRONMENT

We use a high-fidelity simulator of the quadrotor dynamics. The simulator models air-drag, the low-level control architecture, the motor speeds, and the transmission delays. This quadrotor model was taken from [16] which include a low-level control architecture taking mass-normalized collective thrust and body rates as inputs. We also implemented a high-level controller: an SE(3) controller [17], following the implementation from [18]. This controller, combined with the low-level quadrotor model, allow us to give alternate high-level commands to the quadrotors, such as position, velocity, or acceleration commands. The control architecture is illustrated in Figure 2. Our simulator also computes the collision between quadrotors, the elements of the arena and the net carried by the pursuers. This fidelity facilitates the transfer of policies trained in simulation to the real world.

The simulation framework is entirely written using JAX [19]. This Python library allows the code to be just-in-time compiled and lowered to GPU during runtime, resulting in

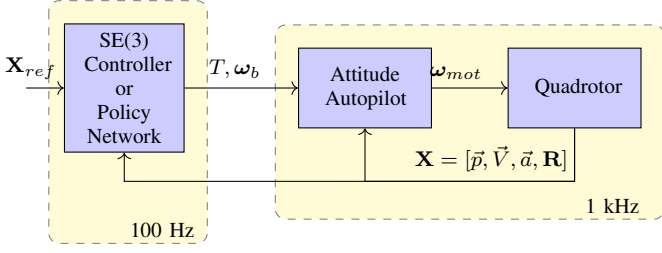


Fig. 2: Control architecture used for the quadrotor dynamics simulation.

fast execution times of up to millions of steps per second by leveraging the parallelization capabilities of GPUs.

#### IV. INTERCEPTION OF AN AGILE TARGET USING REINFORCEMENT LEARNING

Drone neutralisation methods are classified as kinetic or non-kinetic [1]. Non-kinetic approaches, such as jamming or spoofing, are ineffective against autonomous drones. Kinetic methods, including projectiles or collisions, and electromagnetic weapons, risk causing uncontrolled crashes and debris. Using a net, towed or projected by the pursuer, avoids these issues by safely capturing the target; here, we consider a net taut to the pursuer and released upon capture.

##### A. Reinforcement Learning

Reinforcement Learning is a type of machine learning for sequential decision-making. In a *rollout* phase, an *agent* interacts with an uncertain *environment* which provides it with a *partial observations* of its *state*, takes a series of *actions* following a *policy* and receives a scalar feedback in the form of *rewards*. These sequences of observe-act-reward, repeated over time, form the *rollouts*. The collected rollouts are then used to update the policy in a learning phase, which will then be employed in the rollout phase of the next training iteration. The goal of the agent is to learn a policy that maximizes the expected cumulative reward over time.

Multi-Agent Reinforcement Learning (MARL) extends RL to scenarios with multiple agents interacting in a shared environment. MARL suffers from the curse of dimensionality and non-stationarity, as the environment dynamics change as other agents learn and adapt their policies. Recent works in MARL adopted centralized training with decentralized execution (CTDE) [11], where agents have access to global information during training but operate based on local observations during execution. In competitive settings, this alleviates non-stationarity by allowing the agents to access the state and actions of their opponents during training.

##### B. Pursuit-evasion problem

We study a pursuit-evasion scenario with two quadrotors, a *pursuer* and an *evader*, operating in an obstacle-free rectangular arena of size  $L \times L \times H$ . At the beginning of each episode, the agents' initial positions are drawn uniformly at random inside the arena. The pursuer seeks to capture the

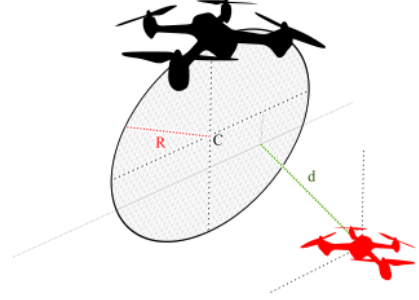


Fig. 3: Schematic of the interception problem. Capture happens when the distance  $d$  between the evader's centre comes within a capture distance of the pursuer's net.

evader as quickly as possible, whereas the evader tries to evade capture. Capture occurs when the evader's centre comes within a *capture distance* of the pursuer's rigid, circular net of radius  $R$ , which is mounted on the pursuer and aligned with its body frame, and represented in Figure 3. Because a single pursuer cannot intercept a faster evader alone, we assume the pursuer's and the evader's manoeuvring capabilities are identical. Target detection, state estimation, and trajectory prediction are not addressed in this work.

Both the evader and the pursuer are forbidden to exit the boundaries of the arena. In this setting the evader can quickly learn to fly close to the arena walls to stay safe, exploiting the pursuer's fear to avoid boundary violations. To discourage this behaviour and promote agile evasive flight in the central region, a narrow buffer zone is added adjacent to every wall; only the evader is penalized for entering this zone and thus, the evader is constrained in a smaller volume in the centre of the arena.

1) *Observation, actions, and rewards:* At time step  $t$ , each agent's  $i \in (\text{pursuer}, \text{evader})$  observation  $\mathbf{o}_i$  is composed of the following elements: self-state observation  $\mathbf{o}_i^{\text{self}}$ , observation of the opponent  $\mathbf{o}_i^{\text{opp}}$ , and the observation of the arena bounds and the ground  $\mathbf{o}_i^{\text{env}}$ . The self-state observation is  $\mathbf{o}_i^{\text{self}} = [\mathbf{v}_i, \text{vec}(\mathbf{R}_i)]$  containing the agent's linear velocity  $\mathbf{v}_i$ , and its rotation matrix  $\mathbf{R}_i$ , with  $\text{vec}(\cdot)$  being the flattening function. The observation of the opponent is  $\mathbf{o}_i^{\text{opp}} = [\mathbf{p}_o - \mathbf{p}_i, \mathbf{v}_o - \mathbf{v}_i]$  containing the position and velocity of the opponent expressed relative to the agent in world coordinates. The observation of the arena bounds and the ground is  $\mathbf{o}_i^{\text{env}} = [\text{norm}(\mathbf{p}_o - \mathbf{p}_i)]_{o \in \text{bounds} + \text{ground}}$  and is composed of the Euclidean distances from the agent to each arena boundary and to the ground. We normalize the observations before feeding them to the neural network. Relative positions are normalized by the maximum range of view  $\mathbf{k}_{\mathbf{p}_i}$ , and velocities are normalized by a maximum velocity parameter for each agent  $\mathbf{k}_{\mathbf{v}_i}$ .

The control policies are trained using Proximal Policy Optimization (PPO) [20]. This Actor-Critic method uses two neural networks for each agent: a policy network and a value network.

The policy network produces an action  $\mathbf{a}_i$  for each agent,

which is a vector of body rates  $\mathbf{a}_i^\omega$  and a collective thrust  $\mathbf{a}_i^{th}$ .

The value networks are only used during training time and have access to privileged information about the opponent's state, which is not available to the policy network. This alleviates the non-stationarity of the environment due to the simultaneous learning of both agents [11]. The input of the value network of each agent is the concatenation of the position, velocity, and rotation matrix of each agent, as well as the action taken by the opponent at this time step. This input is normalized before being fed to the neural network.

The reward of the pursuer  $r^P$  and the evader  $r^E$  are given by:

$$\begin{aligned} r^P &= r^{\text{catch}} - r^{\text{dist}} - r^{\text{coll}} - r^{\text{fail}} - r^{\text{cmd}}, \\ r^E &= -r^{\text{catch}} + r^{\text{dist}} - r^{\text{coll}} - r^{\text{fail}} - r^{\text{cmd}} - r^{\text{bnd}}. \end{aligned}$$

in which  $r^{\text{catch}}$  rewards the pursuer for catching the evader,  $r^{\text{dist}}$  penalizes the pursuer for being far from the evader,  $r^{\text{fail}}$  penalizes any agent for crashing or going out of bounds,  $r^{\text{coll}}$  penalizes any agent for colliding with the body of their opponent, and  $r^{\text{cmd}}$  discourages dynamically infeasible commands. Instead of terminating the episode upon collision between agents, we apply a soft continuous penalty  $r^{\text{coll}}$  to both agents, allowing for gradual learning of collision avoidance while maintaining focus on the primary tasks of pursuit and evasion. We still terminate the episode if any agent crashes on the ground or goes out of bounds and apply a hard penalty  $r^{\text{fail}}$ . However, neither the evader nor the pursuer receive a reward when the opponent reaches a failure state to promote actual pursuit-evasion behaviours rather than forcing the opponent to crash. Additionally, we add  $r^{\text{bnd}}$  to the evader's reward function, which penalizes it for approaching the arena bounds.

Specifically, the reward terms are:

$$\begin{aligned} r^{\text{catch}} &= \lambda_{\text{catch}} \cdot \mathbf{1}_{\text{catch}}, & r^{\text{dist}} &= \lambda_{\text{dist}} \cdot \|\mathbf{p}_e - \mathbf{c}^{\text{net}}\|_2, \\ r^{\text{coll}} &= \lambda_{\text{coll}} \mathbf{1}_{\text{contact}}, & r^{\text{fail}} &= \lambda_{\text{fail}} \mathbf{1}_{\text{fail}}, \\ r^{\text{cmd}} &= \lambda_{\text{cmd}} \|\mathbf{a}^\omega\|, & r^{\text{bnd}} &= \phi_{\text{bnd}}(d^{\text{bnd}}), \end{aligned}$$

in which the indicator functions return 1 when their condition is met:  $\mathbf{1}_{\text{catch}}$  when catching the evader,  $\mathbf{1}_{\text{contact}}$  for inter-agent contact,  $\mathbf{1}_{\text{fail}}$  for reaching a failure state because of a ground crash or leaving the arena bounds.  $\mathbf{c}^{\text{net}}$  is the pursuer's catching net-centre position, and  $\mathbf{a}^\omega$  are the commanded body rates.  $\phi_{\text{bnd}}$  is a function that penalizes the evader for approaching the arena bounds, triggering under a set threshold and growing exponentially the shorter the distance to the arena bounds  $d^{\text{bnd}}$ .  $\lambda_{\text{catch}}$ ,  $\lambda_{\text{dist}}$ ,  $\lambda_{\text{coll}}$ ,  $\lambda_{\text{term}}$ ,  $\lambda_{\text{cmd}}$  are positive hyperparameters that balance the different reward terms and have been tuned to obtain the desired behaviour and listed in Table I.

TABLE I: Reward coefficients.

Coefficient	Value	Coefficient	Value
$\lambda_{\text{catch}}$	10.0	$\lambda_{\text{coll}}$	0.1
$\lambda_{\text{dist}}$	0.001	$\lambda_{\text{fail}}$	30.0
$\lambda_{\text{cmd}}$	2e-04	$\lambda_{\text{bnd}}$	1.0

### C. Training details

Rollouts are generated in parallel across 1024 environments. Episodes start from uniformly sampled initial positions in the  $L \times L \times H$  arena; no domain-randomisation of the platform dynamics is applied. Episodes last up to  $T = 10$  s (1000 time steps) unless terminated earlier due to capture, crash, or arena exit.

Each policy network is a two-layer multilayer perceptron with 256 ReLU units per hidden layer. The output layer produces the mean and standard-deviation of a multivariate Gaussian, followed by a  $\tanh$  squashing to obtain bounded continuous actions. The value networks mirrors this architecture but ends with a linear output.

The entire pipeline is written in Python using JAX [19], enabling just-in-time compilation and parallelized execution. Running on a single machine equipped with an NVIDIA RTX 4090 (24 GB VRAM), an AMD Ryzen 9 7950X3D (16 cores, 4.2 GHz) and 128 GB RAM, the system collects and processes approximately  $3.5 \times 10^5$  environment steps per second. We train for a total of  $2 \times 10^9$  environment steps, corresponding to roughly 1h35 of wall-clock training time. Training hyperparameters are listed in Table II.

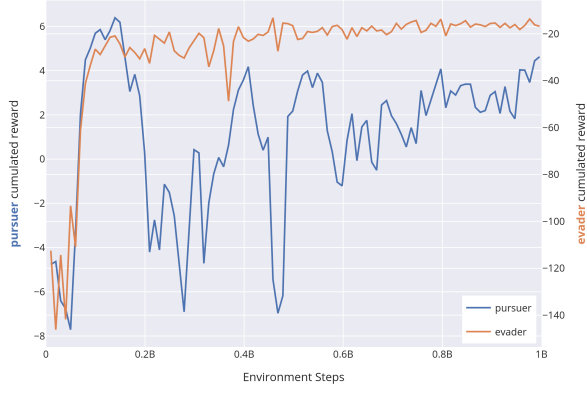
## V. EXPERIMENTAL RESULTS

### A. Training Results

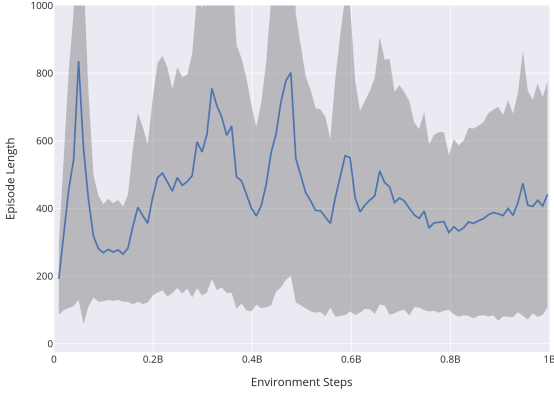
Figure 4 compares the learning curves of the pursuer and the evader. The pursuer cumulated return initially rises as the drone learns to fly and avoid crashes and the first interception happens. Because the evader's reward contains an additional boundary term, its learning progress is intrinsically slower; it does not reach high-speed flight as early as the pursuer. The pursuer therefore overfits to an increasingly predictable evader. However, as the evader also learns to fly and evade, the pursuer's return decreases drastically as it can no longer catch the evader. This also translates into the average episode length which first increases as both agents learn to hover and avoid crashes, but soon falls sharply as the pursuer discovers a quick capture strategy. Eventually, the pursuer finds a new strategy to catch the evader again, the average episode length

TABLE II: Training hyperparameters.

Hyperparameter	Value
Number of parallel environments (per agent)	1024
Rollout length	128
Learning rate	$5 \times 10^{-4}$
Discount factor	0.99
Number of PPO epochs per training data batch	15
Number of minibatches per PPO epoch	1
Discount factor	0.99
Lambda value for GAE computation	0.95
Clipping value for PPO updates	0.2
Entropy	0.01
Critic weight in loss function	0.5
Maximum norm of the gradients for a weight update	0.5
Decay learning rates	False
Total number of training steps	$4 \times 10^9$



(a) Average cumulated reward over training.



(b) Average episode length.

Fig. 4: Comparison of learning curves and average episode length.

decreases and the return of the pursuer increases as it learns to catch the evader more consistently. This behaviour is typical of co-evolutionary learning [21], and happens multiple times during the training as both agent cycle through periods of adaptation and counter-adaptation. Both curves converge to a near-stationary value, suggesting that the joint policy profile is approaching a Nash equilibrium.

### B. Evaluation in Simulation

We compare the performances of the trained policies with baseline heuristic methods. For the pursuer, Pure-Pursuit (PP), a classical interception strategy where the pursuer follows a straight line towards the position of the evader, and Fast-Response Proportional Navigation [10] which is an evolution of Proportional Navigation for manoeuvring multi-rotors. For the evader, a hovering strategy where the evader tries to maintain a fixed position in space, and an Artificial Potential Field strategy where the evader is repelled by the pursuer and the boundaries of the arena. We use the potential field formulation from [22]. In the sake of comparison, these heuristic methods only access the position and velocity of the opponent agent, as our RL policies only access this information. How to estimate the state of the evader in high-speed manoeuvring flights is not

TABLE III: Performances of the pursuer and the evader in a 40x40x14m (Large) and a 8x8x5m (Small) arena.

Pursuer mode	Evader Mode					
	Small arena			Large arena		
PP	Hov.	APF	DRL	Hov.	APF	DRL
<b>Catch Rate (%)</b>	96.4	19.5	58.3	<b>100</b>	15.9	24.6
<b>Evade Rate (%)</b>	0.0	47.9	<b>40.0</b>	0.0	66.6	<b>74.6</b>
of which timeout	0.0	0.0	<b>0.8</b>	0.0	0.0	<b>1.1</b>
<b>Crash rates (%)</b>						
Pursuer	0.0	47.9	39.3	<b>0.0</b>	53.2	73.5
Evader	-	32.0	<b>0.6</b>	-	25.9	<b>0.3</b>
Double	3.6	0.6	1.0	<b>0.0</b>	0.3	0.5
<b>Time to Catch (s)</b>						
Mean	2.05	8.29	5.29	<b>6.65</b>	8.72	8.32
Std	1.66	3.48	4.05	3.90	2.94	3.11
<b>FRPN [10]</b>						
<b>Catch Rate (%)</b>	<b>97.4</b>	19.6	37.7	97.5	<b>68.8</b>	49.2
<b>Evade Rate (%)</b>	0.1	43.1	<b>59.9</b>	0.0	1.0	<b>47.3</b>
of which timeout	0.0	1.2	<b>0.1</b>	0.0	0.0	<b>20.2</b>
<b>Crash rates (%)</b>						
Pursuer	<b>0.1</b>	42.0	59.8	0.0	<b>1.0</b>	27.1
Evader	-	36.4	<b>1.3</b>	-	30.2	<b>3.2</b>
Double	<b>2.5</b>	0.8	1.1	2.5	<b>0.0</b>	0.3
<b>Time to Catch (s)</b>						
Mean	<b>2.03</b>	8.49	6.88	<b>2.70</b>	<b>4.97</b>	6.72
Std	1.33	3.15	4.03	1.29	3.44	3.67
<b>DRL (Ours)</b>						
<b>Catch Rate (%)</b>	90.7	<b>71.8</b>	<b>78.8</b>	20.7	34.0	<b>66.5</b>
<b>Evade Rate (%)</b>	6.6	6.9	<b>16.5</b>	<b>75.6</b>	50.4	31.9
of which timeout	1.0	0.3	<b>2.3</b>	<b>74.0</b>	8.1	14.7
<b>Crash rates (%)</b>						
Pursuer	5.6	<b>6.6</b>	<b>14.2</b>	1.6	42.3	<b>17.2</b>
Evader	-	20.3	<b>4.1</b>	-	15.4	<b>1.5</b>
Double	0.4	<b>1.0</b>	<b>0.6</b>	3.7	0.2	<b>0.1</b>
<b>Time to Catch (s)</b>						
Mean	2.62	<b>4.18</b>	<b>3.78</b>	8.96	7.61	<b>6.62</b>
Std	2.76	3.89	3.60	2.46	3.55	3.34

Hov.: Hovering, APF: Artificial Potential Field

text in blue : best pursuer against this column's evader

text in orange : best evader against this row's pursuer

in the scope of this paper. The heuristic baselines give velocity or acceleration commands that are then converted to body rates and collective thrust using the SE(3) controller described in Section III.

The main comparison metrics are the catch rate of the pursuer, the evade rate of the evader, the time to catch and the crash rate. The catch rate is the percentage of episodes where the pursuer successfully catches the evader before a timeout of 10 seconds. The evade rate is the percentage of episodes in which the evader avoids capture for 10 seconds or the pursuer crashes. We also identify three different crash rates: pursuer crash rate and evader crash rate are the percentage of episodes where either the pursuer or the evader crashes alone, and double-crash rate is the percentage of episodes where both agents crash simultaneously. Finally, the time to catch is the time taken by the pursuer to catch the evader.

Time-to-catch is naturally biased towards lower values as it only consider successful catches, thus a weaker pursuer can appear to have a better time to catch as it would only succeed in catching the easiest targets without crashing. To alleviate



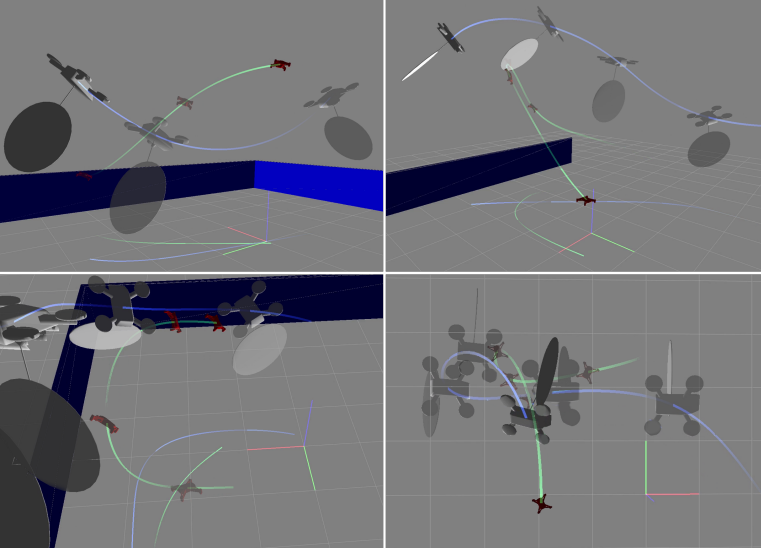


Fig. 5: Evasive manoeuvres: from top left to bottom right, the evader (green) performs a vertical escape, a dive, a sharp turn, and a sudden stop followed by a feint.

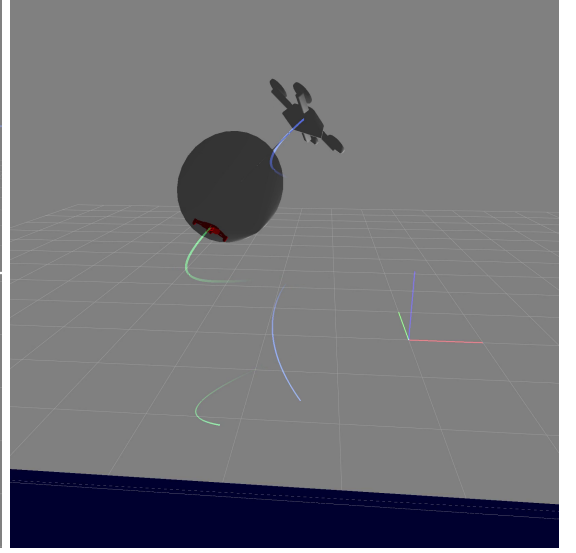


Fig. 6: a high roll angle catch: the pursuer (blue) intercepts the evader (green) with a roll angle of more than 45 degrees.

this issue, we use a right-censored metric for the time to catch: if the episode ends because of a crash or a timeout, the time to catch is considered to be of 10 seconds.

We evaluate the performances of our strategies in two different settings. First in a large arena of size  $40 \times 40 \times 14$  meters, with the evader constrained in a smaller volume of size  $20 \times 20 \times 4$  meters in the centre of the arena. In this setting, the agents can reach higher speeds and perform long-range manoeuvres with low risk of crashing into the boundaries. This is to the advantage of the heuristic pursuer baselines, which do not account for the presences of boundaries. Then in a smaller arena of size  $8 \times 8 \times 5$  meters, with the evader constrained in a volume of size  $6 \times 6 \times 4$  meters in the centre of the arena, closer to indoor voliere flight conditions. In this setting, the agents are more constrained by the boundaries and have to perform tighter manoeuvres. For each setting, a specific pursuer and an evader model was trained for this specific arena size. For each combination of pursuer and evader strategies, we run 10,000 episodes and report the averaged metrics in Table III.

The learned evader outperforms the *moving* heuristic evaders in all settings, achieving a higher evade rate and lower crash rate against all pursuers. The learned evader is particularly effective against the heuristic pursuers, which have a high crash rate when facing agile manoeuvres. Against FRPN in the larger arena where it is less crash-prone, half of the successful evasions are due to timeouts, showing that the learned evader can consistently avoid capture for the full duration of the episode. This shows that we successfully trained an agile evader that can exploit the full 3D space to avoid capture while avoiding crashes.

All the pursuer heuristic baselines performances drop when facing the agile learned evaders. Their crash rate is high, as it does not take into account for the presence of boundaries. This

effect is exacerbated in the smaller arena. In comparison, the learned pursuer that was trained to avoid crashes shows a much lower crash rate against the agile learned evader. The learned pursuer has also the highest catch rate and lowest time to catch against the agile learned evader, showing that it has learned effective interception strategies against agile manoeuvres while respecting arena boundaries and minimizing crashes. The low time-to-catch shows that the learned pursuer succeeds against the hardest-to-catch, most time-consuming evaders where the other methods fail or crash.

However, the performances of the learned pursuers drop significantly when facing the heuristic evaders, especially the hovering one. While the learned pursuer still display a low crash rate, its catch rate is much lower than the heuristic pursuers in the larger arena. This suggests that the learned pursuer has overfitted to the strategies of the agile learned evader encountered during training, and fails to find effective interception strategies against less agile evaders. This is especially true against the hovering evader. It seems to be a very easy target to catch, but this situation was likely not encountered during training, as the evader was always trying to escape. Moreover, the pursuer's observation do not encode history information, and cannot infer that the evader is stationary from its current observed state (position, velocity). As a result, it did not learn to exploit the lack of movement to optimize its interception strategy. In fact, it must expect it to flee at any moment. This is less pronounced in the smaller arena, where the boundaries further constrain the evader's movements. It is likely that the learned pursuer encountered more trajectories where the evader was close to stationary during training, allowing it to learn some interception strategies against this type of target. As a result, the learned pursuer still achieves a higher catch rate and faster time to catch than the heuristic pursuers against

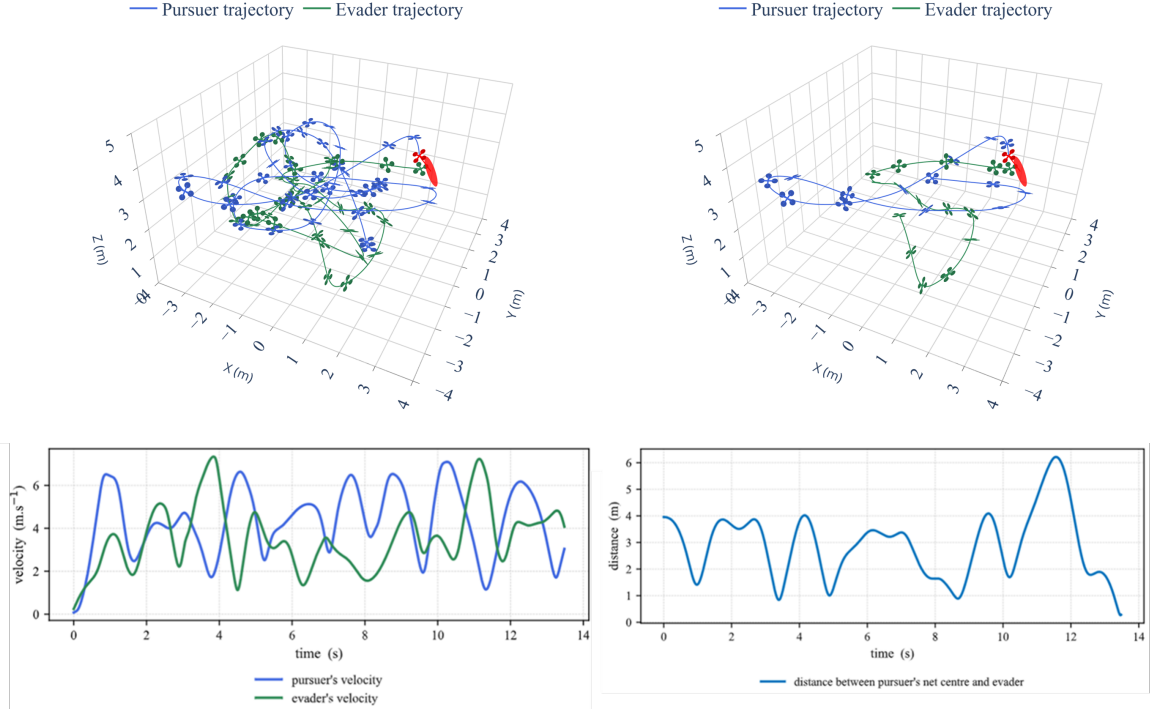


Fig. 7: Simulation results: A full pursuit-evasion episode using the trained policies. The trajectory on the right is a shortened version of the full episode for more visibility. We can see the pursuer first missing the evader in a first attempt, then successfully catching it after a second attempt.

all moving evaders in the smaller arena, primarily due to its ability to avoid boundaries and crashes.

### C. Qualitative Results in Simulation

In this section, we present qualitative results of our trained policies in the small arena setting ( $8 \times 8 \times 5$  m).

As shown in Figure 5, the evader learned a diverse set of agile evasive manoeuvres, including high accelerations, high velocity flights, sudden stops, sharp turns, vertical movements, and feints. In response, the pursuer learned to anticipate these manoeuvres. We observed the pursuer catching the evader with very high roll and pitch angles ( $>45$  degrees) (Figure 6). Despite being not specifically enforced by the reward function to turn the heading towards the evader, the pursuer learned to do so in order to maximize the surface of the catching net facing the evader, which increases the chances of a successful capture.

It also learned to catch the evader using both sides of the catching net, to intercept an evader that went behind it without turning around.

One trajectory obtained is shown in Figure 7. Both the pursuer and the evader display agile manoeuvres in a very restricted arena, with velocities of up to  $\sim 7.5$  m/s. The pursuer (in blue) is able to catch the evader (in green) after 13 seconds of intense chase, showcasing the ability of the learned policies to sustain high intensity flights while avoiding crashes.

### D. Real-World Demonstration

We demonstrated the trained policies in a real-world scenario in our indoor flight arena of size  $8 \times 8 \times 5$  meters. The policies have been directly transferred from simulation to reality without any additional fine-tuning or adaptation. For this flight, the evader was simulated on a ground station computer, while the pursuer was flying a real quadcopter equipped with a Betaflight [23] flight controller. The flight logs were recorded, including and action commands, and analysed afterward to identify successful catches and the collisions between the pursuer and the evader. The state of the real drone is estimated using a motion capture system (OptiTrack) that provides accurate position and orientation data at 200 Hz and transferred to the simulation. The neural network policy was executed remotely on the ground station computer and the outputted control commands transferred to the drones via an RF link at 100 Hz. We adopted this Hardware-In-The-Loop setup to ensure safety during the flights, but it is not a limitation of our approach as the trained policies can be executed on-board in a decentralized way.

The pursuer managed to fly without crashing or exiting the arena during 28 seconds, and successfully caught the evader 7 times during this period. A portion of the recorded trajectory is shown in Figure 8.

The flight logs were analysed to identify the error between the simulation and the real-world execution. At each time step, we computed the error between the expected next state from the simulation and the actual next state recorded from the

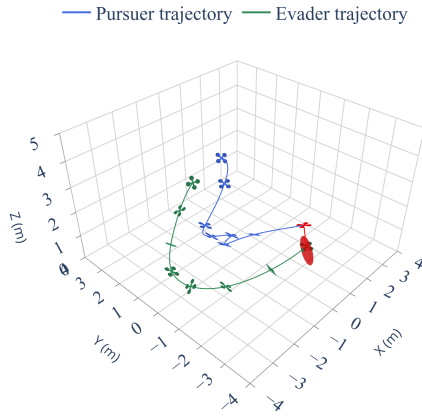


Fig. 8: A portion of the real-world flight trajectory. The pursuer (blue) successfully caught the evader (green).

real-world flight with the actions given to the policy network. With a period between time steps at 100Hz of 0.01 seconds, the position RMSE is 0.009 m on the xy plane and 0.003 m on the z axis, while the velocity RMSE is 0.070 m/s on the xy plane and 0.040 m/s on the z axis.

## VI. CONCLUSION

In this work, we addressed the challenging problem of intercepting an agile aerial target using a pursuer drone equipped with a catching net. We formulated this task as a competitive multi-agent reinforcement learning problem, training independent policies for both the pursuer and the evader using PPO with low-level control inputs (collective thrust and body rates). A key element of our approach was the integration of a high-fidelity quadrotor dynamics model and a multi-agent reinforcement learning framework for the training of both the pursuer and the evader.

Our simulation results demonstrated that the trained policies outperformed classical heuristic baselines in simulated interception tasks of agile evader, achieving higher catch rates and demonstrating greater robustness against crashes, particularly when facing agile, learned opponents. Furthermore, the development of our simulation environment entirely within the JAX framework proved crucial, enabling massively parallelized execution and drastically accelerating the training process, which made extensive RL training computationally feasible.

While comprehensive quantitative evaluation in the physical world remains challenging, we successfully demonstrated the learned policies on agile quadrotors in our indoor flight arena, validating the potential for zero-shot sim-to-real transfer and showcasing the practical applicability of our approach.

Overall, this research highlights the effectiveness of Multi-Agent Competitive Reinforcement Learning for generating highly agile and reactive control policies for complex robotic interaction tasks like drone interception.

## VII. LIMITATIONS

While our approach demonstrates promising results for agile drone interception, several limitations should be acknowledged.

First, the sim-to-real gap remains a significant challenge. Although our simulation uses quadrotor dynamics identified from real flight data, trained policies remain sensitive to mismatches between the simulated model and actual hardware. Uncertainties in parameters like mass, inertia, or motor limits can degrade real-world performance. Incorporating domain randomization during training could improve robustness to such discrepancies.

Second, we observed instances where agents appeared to overfit to certain interaction patterns. When opponents deviated from typical behaviors (e.g., flying erratically or hovering), agents sometimes responded suboptimally or failed (e.g., crashing), rather than robustly pursuing their objectives. When training only against the latest version of the opponent, agents are prone to forget how to deal with previously encountered strategies, and it limits the generalization capabilities of the learned policies. Expanding the diversity of opponent strategies met during training, for example via Self-Play or Population-Based Training [6], could mitigate overfitting and improve generalization.

Third, we assume perfect state information for both agents during training and execution. Incorporating realistic sensor models and handling partial observability are important for real-world deployment, where robust perception of the target in high-velocity flights and state estimation are required but were not addressed in this study.

Fourth, the current study focuses on a one-vs-one "dog-fight" scenario within a bounded, obstacle-free arena. Extending the approach to handle multiple pursuers and/or evaders, operate in cluttered environments, or address different objectives like area defense requires further investigation.

Finally, while we demonstrated feasibility in real-world flights, the quantitative evaluation was primarily conducted in simulation. A more extensive real-world experimental campaign would be necessary to rigorously quantify performance metrics like catch rate and time-to-catch under physical conditions.

## REFERENCES

- [1] S. Park, H. T. Kim, S. Lee, H. Joo, and H. Kim, "Survey on Anti-Drone Systems: Components, Designs, and Challenges," *IEEE Access*, vol. 9, pp. 42 635–42 659, 2021.
- [2] R. Yanushevsky, *Modern Missile Guidance*, 2nd ed. Boca Raton: CRC Press, Sep. 2018.
- [3] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, Nov. 2011. [Online]. Available: <https://doi.org/10.1007/s10514-011-9241-4>
- [4] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023. [Online]. Available: <https://doi.org/10.1038/s41586-023-06419-4>
- [5] "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," Dec. 2017, arXiv:1712.01815 [cs]. [Online]. Available: <http://arxiv.org/abs/1712.01815>



- [6] “Dota 2 with Large Scale Deep Reinforcement Learning,” Dec. 2019, arXiv:1912.06680 [cs]. [Online]. Available: <http://arxiv.org/abs/1912.06680>
- [7] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.
- [8] X. Wang, J. Zhou, Y. Feng, J. Mei, J. Chen, and S. Li, “Dashing for the golden snitch: Multi-drone time-optimal motion planning with multi-agent reinforcement learning,” *arXiv preprint arXiv:2409.16720*, 2024. [Online]. Available: <https://arxiv.org/abs/2409.16720>
- [9] M. Geisert and N. Mansard, “Trajectory generation for quadrotor based systems using numerical optimal control,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 2958–2964.
- [10] M. Pliska, M. Vrba, T. Báča, and M. Saska, “Towards safe mid-air drone interception: Strategies for tracking & capture,” *IEEE Robotics and Automation Letters*, 2024.
- [11] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.
- [12] A. P. Pope, J. S. Ide, D. Mićović, H. Diaz, J. C. Twedt, K. Alcedo, T. T. Walker, D. Rosenbluth, L. Ritholtz, and D. Javorsek, “Hierarchical reinforcement learning for air combat at darpa’s alphasight trials,” *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 6, pp. 1371–1385, 2023.
- [13] R. Zhang, Q. Zong, X. Zhang, L. Dou, and B. Tian, “Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7900–7909, Oct. 2023. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9712866>
- [14] J. Chen, C. Yu, G. Li, W. Tang, S. Ji, X. Yang, B. Xu, H. Yang, and Y. Wang, “Online planning for multi-uav pursuit-evasion in unknown environments using deep reinforcement learning,” 2024.
- [15] J. Xiao and M. Feroskhan, “Learning Multi-Pursuit Evasion for Safe Targeted Navigation of Drones,” *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 12, pp. 6210–6224, Dec. 2024, arXiv:2304.03443 [cs]. [Online]. Available: <http://arxiv.org/abs/2304.03443>
- [16] J. Heeg, Y. Song, and D. Scaramuzza, “Learning quadrotor control from visual features using differentiable simulation,” *arXiv preprint arXiv:2410.15979*, 2024.
- [17] T. Lee, M. Leok, and N. H. McClamroch, “Geometric tracking control of a quadrotor uav on se (3),” in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.
- [18] S. Folk, J. Paulos, and V. Kumar, “Rotorpy: A python-based multirotor simulator with aerodynamics for education and research,” *arXiv preprint arXiv:2306.04485*, 2023.
- [19] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018.
- [20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Tech. Rep., Aug. 2017.
- [21] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, “Emergent complexity via multi-agent competition,” *CoRR*, vol. abs/1710.03748, 2017. [Online]. Available: <http://arxiv.org/abs/1710.03748>
- [22] Z. Zhang, D. Zhang, Q. Zhang, W. Pan, and T. Hu, “Dacoop-a: Decentralized adaptive cooperative pursuit via attention,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.15699>
- [23] “The betafight open source flight controller firmware project.” 2022. [Online]. Available: <https://github.com/betaflight/betaflight>