Reinforcement Learning based Countermeasures against Intelligent Attacking UAV Swarms

Jennifer Simonjan
Technology Innovation Institute
Abu Dhabi, UAE
jennifer.simonjan@tii.ae

Kseniia Harshina
Lakeside Labs
Klagenfurt, Austria
harshina@lakeside-labs.com

Melanie Schranz
Lakeside Labs
Klagenfurt, Austria
schranz@lakeside-labs.com

***Abstract*—The development of autonomous swarm behavior for UAV swarms has increased significantly in recent years. Many applications such as collective transport, exploration of unknown territory or target search and delivery benefit from the flexibility, scalability and robustness of the swarm approach. Besides new application possibilities, these characteristics might also be used for malicious or dangerous purposes like autonomous target-oriented attacks. To date, research is lacking intelligent countermeasures to intervene in attacking UAV swarms. Typical defense mechanisms employ attack-defense confrontation which increases the risk of collateral damage as drones might fall from the sky. Rather than creating a confrontation, we focus on developing countermeasures to intelligently mislead or delay attacks on a target. Therefore, we explore two multi-agent deep reinforcement learning strategies for defender UAVs to intervene in target-oriented attacks of intelligent UAV swarms. Both strategies are based on the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm and aim at preventing or at least delaying attacks. Via simulations we model and evaluate the performance of both methods and compare it to a baseline approach.**

***Index Terms*—UAV swarms, swarm intelligence, swarm attack, drone defense, anti-drone system, reinforcement learning**

I. Introduction

In the past decade, the concept of swarm intelligence became very popular for unmanned aerial vehicle (UAV) swarm applications. To be able to achieve an intelligent collective behavior, the UAVs need to be equipped with on-board intelligence, communication and sensing capabilities. These capabilities allow the UAVs to coordinate with others, fly in formations and dynamically distribute tasks in a self-organized way. Pursuing common goals while relying solely on local rules makes swarms extremely flexible, scalable and robust, enabling a new era of applications, such as, search and rescue, collective transport or exploration of unknown territory. However, the way to achieve fully autonomous and reliable swarm robotic applications has not been mastered yet [1].

Besides the many advantages that autonomous swarms introduce, they may also be used for malicious or dangerous purposes. Swarms of armed UAVs will be very difficult to stop without physical confrontation, causing even further damage due to crashing and falling UAVs. In the military domain, target-oriented single and multi UAV attacks already gathered a lot of interest in the past years [2], [3].

As machine learning (ML) has been able to solve a variety of problems from countless fields in recent years, we want to explore ML approaches to create novel countermeasures against attacking UAV swarms. In this paper, we consider for the first time in literature a ML-based attacking UAV swarm, and we propose to intervene or delay its attack by inducing defender UAVs. To do so, we consider to use Reinforcement Learning (RL) approaches on both sides, for attackers and defenders. RL is an intuitive approach to the task of intervening in an attacking swarm as it can be easily presented as a game: there are two opposing teams (attacking and defending UAVs) with clearly established goals and winning conditions. For the purpose of keeping the attackers away from the target, we explore and evaluate two different multi-agent deep RL strategies, which are both based on the multi-agent deep deterministic policy gradient (MADDPG) algorithm [4]. The first strategy is based on communication, and thus assumes that defenders have access to the communication channel of attackers (*communication-based defense*). To avoid this requirement, we introduce a second strategy, which aims at physical protection of the target (*dynamic shield defense*). Both methods are evaluated via simulations and compared to each other and a baseline approach.

To the best of our knowledge, this work is the first that explores how to intervene RL-based UAV swarm attacks. In existing works, attacking swarms typically employ heuristics, which are significantly easier to compromise than ML-based behavior. Additionally, our work proposes the first approach towards distracting a UAV swarm from a target with RL-based methods, as existing works focused on direct confrontation of the attacking swarm (e.g., shooting or fighting them actively). To prevent even higher damage, and to make our solution applicable in civil environments, we propose to manipulate the attacking swarm behavior, such that fighting, shooting and crashing is avoided.

The rest of the paper is organized as follows. The related work is discussed in Section II. Section III introduces the methodology, the MADDPG algorithm, the problem scenario and the observation and action spaces of agents. In Section IV, we explain the RL-based behavior of attackers and defenders. The simulation scenario and evaluation results are presented in Section V. Section VI concludes the paper and outlines future work.

II. Related Work

The increasing use of commercially available drones and their growing capabilities threaten the safety of airspace if they are misused. In fact, UAVs are already used maliciously in various ways. Illegal flights at airports, attacks on public institutions or specific individuals are just a few examples (more details can be found in [5]). In the near future, UAV swarm attacks will enable a new way of warfare and will overtake current military technologies. Therefore, we need countermeasures to defend against such attacks. This section provides a brief overview of existing defense mechanisms and counter UAV systems.

Current countermeasures to defend against attacking UAVs include nets [6], signal jammers [7], spoofers [8] or geofencing [9]. However, most of these methods are only applicable in a military environment as they have too much impact on the civilian population or require additional infrastructure. Attempts at misleading a UAV swarm are difficult as parameters like speed, quantity and unpredictability due to local interactions change dynamically. Lykou et al. [10] explain the limitations of these types of defense mechanisms in more detail in their survey.

In general, the literature lacks explorations on how to defeat, mislead or stop UAV swarms from attacking a target. A first study on a mechanism to recognize and suppress the emergent behavior of UAV swarms has been presented by Liu et al. [11]. The authors concluded that comprehensive suppression on the premise of correct recognition of flocking behavior is the best strategy fighting against a swarm emergent behavior. Chen et al. [12] have proposed a fast counter approach to deprive a drone swarm of its coordination and clustering capabilities by splitting it into several unconnected components. To achieve efficient splitting, a genetic algorithm and a particle swarm optimization algorithm have been proposed for searching critical nodes. This approach assumes that defenders can determine critical nodes among the attackers and can disable them to eventually split the topology. This assumption might not be valid in a real-world scenario, especially if the attackers do not rely on clusters. In our previous work [13], we explored how to intervene a UAV swarm by inducing defender UAVs into the swarm with the goal to mislead the swarm's mission. Therefore, we assumed that attackers as well as defenders run the same bio-inspired swarm algorithms, and modified the defenders' objective function and interaction to be able to influence the attackers. However, the more robust the optimization algorithm is, the harder it gets to mislead the swarm. Further, we relied on the strong assumption of defenders being aware of the swarm algorithm used by attackers.

A more flexible direction towards misleading or confronting swarms is to exploit ML. One of the first works that generates policies for swarm systems was introduced by Hüttenrauch et al. [14]. The authors proposed a RL-based solution to generate policies for swarm systems which is based on actor-critic learning. This approach was followed by various RL methods to learn swarm behaviors [15], [16], [17].

A few research groups have started meanwhile to study the dynamic swarm versus swarm combat problem: Xiang et al. [18] have explored the task of UAV swarm confrontation exploiting RL in their recent work. Specifically, they use the MADDPG [4] algorithm together with the rule-coupled method in order to intercept the attacking UAV swarm (defenders shoot the attacking UAVs). The coupling of rules and algorithms gives a winning success rate of 81%. Another recent approach was introduced by Wang et al. [19] who study the UAV swarm confrontation problem with hierarchical multi-agent RL methods. Compared to MADDPG, they claim to improve the performance by 32%. In the domain of RL, there are a few more attempts to tackle the UAV swarm confrontation problem [20], [21]. However, all of the mentioned approaches focus on fighting the attacking swarm, which might cause even higher collateral damage. Therefore, we need novel approaches to defend against UAV swarms in a way that prevents UAVs from crashing and falling.

III. Methodology

In this section, we briefly introduce the RL algorithm which we exploit as a base for the attacker and defender behavior. Afterwards, we outline the problem scenario and describe the action and observation spaces of attackers and defenders.

A. MADDPG Algorithm

The multi-agent deep deterministic policy gradient algorithm (MADDPG) [4] is an actor-critic approach to multi-agent RL problems and extends the DDPG [22] algorithm as follows: (1) it learns policies that only use local information at execution time, (2) it does not assume a differentiable model of the environment dynamics or any particular structure on the communication method between agents, and (3) it is applicable not only to cooperative interaction but to competitive or mixed interaction involving both physical and communicative behavior. MADDPG exploits an actor-critic architecture which means that each agent contains actor and critic networks. The actor network is responsible for the action selection of the agent, which is based on local observations. The critic network is responsible for evaluating the action selected by the actor network. Each agent's critic network needs to access the observations and actions of teammates during the training. Through this implicit information exchange, agents are effectively coordinated. In this way, MADDPG is trained with a global state (centralized training), but only local observations are required for execution (decentralized execution). Figure 1 shows an overview schematic of such a multi-agent actor-critic approach. In our solution, attackers as well as defenders exploit MADDPG to achieve their goals (attackers' try to reach a static target and defenders try to keep them away).

B. Problem Scenario

As mentioned above, the goal of the defenders is to keep the attackers away from the target. In order to do so, we implemented two different RL-based defense strategies. The first one assumes, that defenders have access to the communication

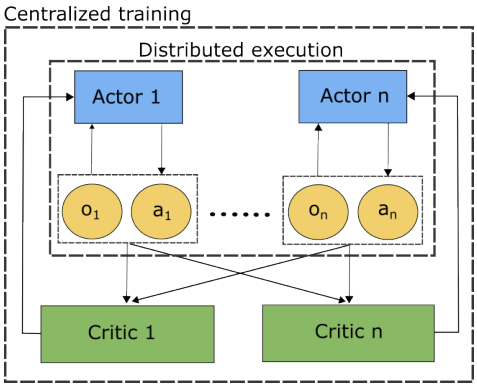


Fig. 1: Schematic of the centralized training and distributed execution in a multi-agent actor-critic approach (adapted from [4]).

channel of the attackers, while the second one aims at physically defending the target. Besides the defenders, also attackers are acting intelligently in their search for the target, exploiting RL as well. To create these two scenarios, we consider the following: A continuous 2D area including a static target M with position $\hat{m} = [x_m, y_m]$ and a fake target F with position $\hat{f} = [x_f, y_f]$, which is only used in the communication-based defense. A swarm of attacking UAVs $a_k \in A, k = 1, \ldots, K$, and a set of defending UAVs $d_l \in D, l = 1, \ldots, L$ move within this area. The set of all UAVs is denoted as $U = A \cup D$. We assume a discrete environment, where time and space are discrete. At every time step t_s, each agent chooses to perform one action of the available action space $\mathcal{S}$, which is explained in the following subsection. We assume that each UAV is equipped with a GPS module and therefore knows its own position. Regarding the notation, positions are always denoted by a circumflex (e.g., $\hat{a}_k$ is the position $[x_k, y_k]$ of attacker a_k). One additional assumption that we did in our first attempt of preventing UAV swarm attacks, is that defenders know the positions of the attackers. This assumption might be met by exploiting additional hardware or sensors to estimate the attackers' positions from the defenders point of views. However, we will relax this assumption in our future work and remodel the reward function that depends on this knowledge.

C. Action and Observation space

To create the actor-critic environment, agents have to have certain observation and action spaces. They choose a certain action from the action space based on what they observe from the environment. In the dynamic shield defense the action and observation spaces are the same for attackers and defenders, while in the communication-based defense they differ as defenders can also send messages to the attackers.

Since the dynamic shield defense only exploits movement and physical interaction between the agents, the action space consists of five movement options in different directions. In the communication-based defense, the defending agents can also send messages, resulting in an additional action on top of the movement actions.

Communication-based approach		
	Attackers	**Defenders**
Action space	move up move down move left move right do nothing	move up move down move left move right do nothing send message
Observation space	position of real target position of other agents message from defenders	position of real target position of fake target Position of other agents

(a)

Dynamic shield approach		
	Attackers	**Defenders**
Action space	move up move down move left move right do nothing	move up move down move left move right do nothing
Observation space	position of the target position of other agents	position of the target position of other agents

(b)

Fig. 2: Illustrations of the simulation environment for the (a) communication-based defense and the (b) dynamic shield defense.

The observation space includes the position of the target, the positions of the other agents and the distance between agent and target. In the communication-based approach, attackers and defenders each have an additional observation: attackers can observe a message from the defenders, and defenders can also observe the position of the fake target.

Figure 2 summarizes the (a) action and (b) observation spaces of attackers and defenders for the communication-based and the dynamic approach.

IV. RL-based Attack and Defense

In general, attackers as well as defenders are modeled to act intelligently applying RL techniques which base the agents' behaviors on certain reward functions r. There are attacker specific rewards r_i^a, defender specific rewards r_i^d and reward functions r_i^u that are used by both. Reward functions either award an agent with a positive reward $+\varrho$ or penalize it with a negative one $-\varrho$. All reward functions are discussed later in this section. In general, the objective of agents is to maximize their rewards. The objective functions $J(a)$ and $J(d)$ (for attackers and defenders respectively) are thus mathematically expressed as follows:

$$J(a) = \max\left(\sum(r_i^a, r_i^u)\right) \tag{1}$$

$$J(d) = \max\left(\sum(r_i^d, r_i^u)\right) \tag{2}$$

In the following sub-sections, we first model the swarm behavior related rewards r_i^u which are exploited by all agents. Afterwards, we discuss the reward functions r_i^a specific to

attackers, followed by the reward functions r_i^d for our two different defense strategies, the communication-based approach and the dynamic shield approach.

A. Basic behavior

Before discussing the strategies of attackers and defenders, we want to introduce general rewards r_i^u, that are executed on all UAVs to serve as basic behavior for all agents. First of all, we do not want UAVs to crash into each other. Therefore, the first reward r_1^u is a typical swarm-behavior reward and is modeled to achieve repulsion between agents:

$$r_1^u(\hat{u}_i) = \begin{cases} -\varrho, & \text{if } \|(\hat{u}_i - \hat{u}_j)\|_2 < \epsilon \mid u \in U \ \wedge \ i \neq j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

If the distance between two agent positions u_i and u_j is smaller than a certain threshold ϵ, the agent gets a negative reward. This reward function is executed on every agent $u \in U$. As UAVs usually exploit certain sensors (e.g., cameras, ultrasound) to measure distance to objects, repulsion can be achieved to any other UAV without the need for communication.

The second general reward r_2^d has been introduced to award the agents for staying within a bounded area around the target. It is formalized as follows:

$$r_2^u(\hat{u}_i, \hat{m}) = \begin{cases} +\varrho, & \text{if } \|(\hat{u}_i - \hat{m})\|_2 < 2\gamma \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

If the agent u_i is within the radius 2γ of the target, it gets a positive reward, otherwise it does not get any reward. These two rewards form the basic behavior of all agents.

B. Attacker behavior

The first attacker specific reward r_1^a ensures that agents stay together in a swarm and is formulated as follows:

$$r_1^a(\hat{a}) = \begin{cases} +\varrho, & \text{if } \exists a_i : \epsilon \leq \|(\hat{a}_i - \hat{a}_j)\|_2 < 2\epsilon \mid a \in A \ \wedge \ i \neq j \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

If the distance to any other attacker $\hat{a}_j$ is between the thresholds ϵ and 2ϵ, the agent $\hat{a}_i$ get a positive reward. The second reward r_2^a is based upon the attacker's proximity to the target. Therefore, they are penalized if they move away from the target as follows:

$$r_2^a(\hat{a}, \hat{m}) = -\min\|(\hat{a}_i - \hat{m})\|_2, \tag{6}$$

where $\hat{a}_i$ are the positions of the attackers and $\hat{m}$ is the position of the target. This function creates a negative reward based on the minimum distance between a attackers and the target. This means, the further the attackers are away from the target, the higher the negative rewards gets. Finally, we define a winning reward to award the agents for winning an episode. This winning reward is granted if any of the attackers reaches the target:

$$r_3^a(\hat{a}, \hat{m}) = \begin{cases} +10\varrho, & \text{if } \exists a_i \in A : \|(\hat{a}_i - \hat{m})\|_2 \leq \gamma \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

where $\hat{a}_i$ are the attacker positions, $\hat{m}$ is the target position and γ is the threshold distance to consider the target as reached.

C. Communication-based defense

The communication-based strategy was developed with the idea of injecting false information into the attacking swarm, and thus misleading it towards a fake target. Therefore, in this scenario, there exists also a fake target F besides the real target M. Defenders know which is the real and which is the fake target, while attackers do not. This defense mechanism assumes that defenders can broadcast messages to infiltrate attackers. Defenders disguise themselves as attacker and broadcast information about the fake target periodically trying to move the attacking swarm towards the fake target. Therefore, rewards are based on the proximity of UAVs to the fake target. All agents rely on local observations. In the communication-based defense, the set of observation differs between attackers and defenders. Attackers' observations include relative distances to both targets and other agents and a message which can be received via the communication channel. The observation space of the defenders includes the same information as the one of the attackers, however, it additionally includes information about which is the real and which is the fake target. The first defender reward r_1^d is based on the distance between defenders and the fake target and is expressed as follows:

$$r_1^d(\hat{d}_i, \hat{f}) = -\min\|(\hat{d}_i - \hat{f})\|_2, \tag{8}$$

where $\hat{d}_i$ is the defender position and $\hat{f}$ is the position of the fake target. This function creates a negative reward based on the minimum distance between defenders and the fake target. This means, the further the defenders are away from the target, the higher the negative rewards gets. We created the exact same reward function for the distance between attackers and the fake target, as the defenders want to mislead them there:

$$r_2^d(\hat{a}_i, \hat{f}) = -\min\|(\hat{a}_i - \hat{f})\|_2, \tag{9}$$

where $\hat{a}_i$ is the position of the defender and $\hat{f}$ is the position of the fake target. Finally, we define two winning rewards granted to defenders for winning an episode. One is achieved if the defenders manage to mislead the attackers to reach the fake goal. It is expressed as follows:

$$r_3^d(\hat{a}, \hat{f}) = \begin{cases} +10\varrho, & \text{if } \exists a_i \in A : \|(\hat{a}_i - \hat{f})\|_2 \leq \gamma \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

where $\hat{a}_i$ is the position of the defender, $\hat{f}$ is the fake target position and γ is the threshold distance to consider the target as reached. The second winning reward is met if the game is over and the attackers did not reach the real target (i.e., did not meet their winning condition). In that case, defenders are awarded if they are all inside the boundary of the $2D$ area:

$$r_4^d(\hat{d}_i, t_s) = \begin{cases} +10\varrho, & \text{if } t_s > T \wedge \forall \hat{d}_i : \text{inside boundary} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where $\hat{d}_i$ are the defender positions, t_s denotes the simulation step and T the total number of simulation steps. This means, if attackers did not manage to reach the real target and defenders managed to stay inside the bounded area, they win.

D. Dynamic shield defense

This strategy was developed in order to avoid the assumption that defenders can access and use the attackers' communication channel to inject false data. In order to prevent the attackers from reaching the target without using any communication, we propose to physically protect the target. To do so, the defenders should create a dynamic shield around the target which keeps attackers away. In this approach the observation space for both, attackers and defenders, is the same. It consists of the relative distances to the target and the relative distances to the other agents. As attackers are assumed to behave like a swarm, they also utilize repulsion (see Equ. 3) which keeps them away from all other UAVs, including defenders, and thus from the target. To protect the target, the first defender reward r_1^d is based on their own proximity to the target. Therefore, we created a reward function which penalizes defenders for being too far from the target:

$$r_1^d(\hat{d}_i, \hat{m}) = -\min\|(\hat{d}_i - \hat{m})\|_2, \tag{12}$$

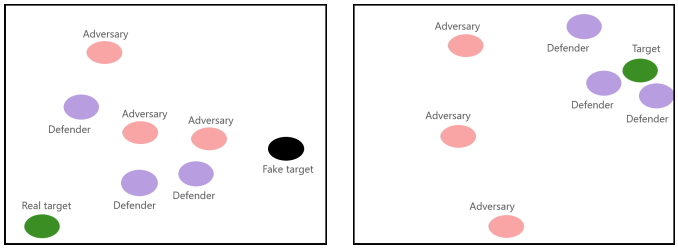
where $\hat{d}_i$ are the defender positions and $\hat{m}$ is the target position. This function creates a negative reward based on the minimum distance between defenders and target. Thus, the further the defenders are away from the target, the higher the negative reward. The second reward function r_2^d is evaluated based on the distance between attackers and the target and is defined as follows:

$$r_2^d(\hat{a}_i, \hat{m}) = \sum_{i=1}^{I} \|(\hat{a}_i - \hat{m})\|_2, \tag{13}$$

with the attacker positions $\hat{a}_i$ and the target position $\hat{m}$. This reward is the sum of the euclidean distances between all attackers and the target, which means that the reward gets higher with larger distances between attackers and target. Finally, we define a winning reward for winning an episode. This winning reward is achieved if the attackers did not manage to reach the target until the end of the game and the defenders managed to stay close around the target (not letting the attackers pass and reach it):

$$r_4^d(\hat{d}_i, \hat{m}, t_s) = \begin{cases} +10\varrho, & \text{if } t_s > T \wedge \|(\hat{d}_i - \hat{m})\|_2 \leq 3\gamma \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

where $\hat{d}_i$ are the defender positions, $\hat{m}$ is the target position, t_s denotes the simulation step and T the total number of simulation steps. As threshold distance we use 3γ, as we want defenders to stay close but not on top of the target. The winning reward is granted if the defenders managed to stay close around the target.



(a) (b)

Fig. 3: Illustrations of the simulation environment for the (a) communication-based defense and the (b) dynamic shield defense.

V. Evaluation

In this section, we discuss the simulation environment and scenarios, and the results of the communication-based defense and the dynamic shield which we compared against a baseline approach.

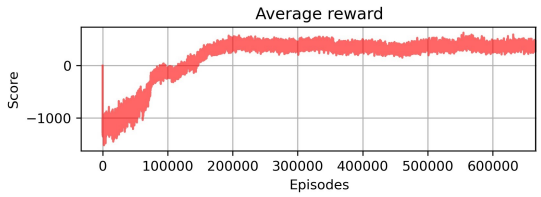
A. Simulation environment

For the simulations, we used the multi-agent particle environment[1] of OpenAI[2], which provides a simple, python-based multi-agent particle world with a continuous observation and discrete action space along with some basic simulated physics. Multi Particle Environments (MPE) [4] are a group of environments with particle agents that also have the capability to communicate with each other by sending messages. Most of these environments contain a landmark and a number of agents to be trained by a RL algorithm. The MPE environments provide a good basis for our problem since they offer the possibility to create cooperative-competitive scenarios.
Our environment models N agents and L landmarks inhabiting a $2D$ area with continuous space and discrete time. Agents can perform physical actions (e.g., move in a certain direction) in the environment and communication actions that get broadcasted to other agents (action and observation spaces of attackers and defenders are detailed in Fig. 2). As explained in [23], the input to the critic network in MADDPG is the concatenation of all agents' local observations, which easily leads to input dimensions that are too high for the critic network. Therefore, MADDPG gets very complex and computationally intensive for large swarms, requiring very long training times. To test our defense algorithms, we thus considered small swarm sizes for six different scenarios:

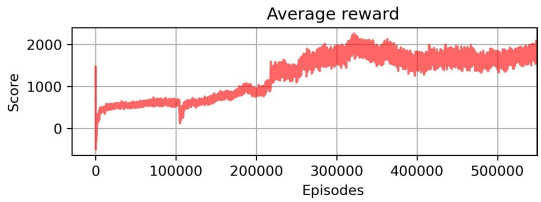
- 2 defenders and 2 attackers (2vs2)
- 2 defenders and 3 attackers (2vs3)
- 3 defenders and 2 attackers (3vs2)
- 3 defenders and 3 attackers (3vs3)
- 3 defenders and 4 attackers (3vs4)
- 4 defenders and 3 attackers (4vs3)

As the target in a real-world scenario is typically known by defenders, we initially spawn the defenders close to the target, while attackers are spawned further away (however, all

[1]https://github.com/openai/multiagent-particle-envs
[2]https://openai.com/



(a)



(b)

Fig. 4: Average reward of all agents in the 3vs3 sceanrio over the number of episodes for the (a) communication-based defense and the (b) dynamic shield defense.

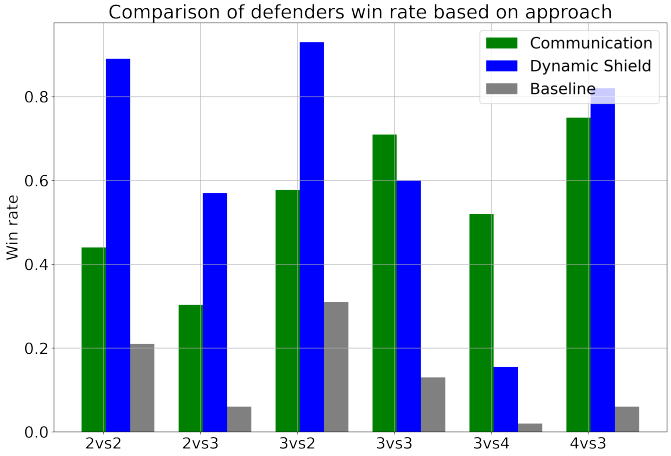


Fig. 5: Comparison of the defenders' winning rate over different scenarios for the two defense and the baseline approach. Green bars depict the winning rate of the communication-based defense, blue bars show the winning rate of the dynamic shield defense and gray bars depict the winning rate of the baseline approach.

are spawned randomly). The hyperparameters of the algorithm are corresponding to the ones of the original MADDPG approach and can be found in [4]. We exploit the Adam optimizer as alternative for the stochastic gradient descent process, for which the learning rate was chosen to be 0.01. Furthermore, we have implemented a baseline approach against which we compare our defense mechanisms. The baseline approach adopts the dynamic shield idea of exploiting physical repulsion, however, defenders move randomly (chosing the next action randomly from the action space). Figure 3 shows two simulation screenshots of (a) the communication-based defense and (b) the dynamic shield defense. As we can see, in the former, there exists a fake target which is used by the defenders to mislead the attackers. In the dynamic shield approach, we can see how defenders place themselves around the target to physically protect it.

B. Results

This section presents the learning rate of the agents as well as the winning probability of defenders in both defense mechanisms and in comparison with the baseline approach. Figure 4 shows the average reward in the learning phase of all agents for the 3vs3 scenario over the number of episodes for (a) the communication-based defense and (b) the dynamic shield defense. The y-axis shows the average of the accumulated rewards presented in Section IV-C and Section IV-D, respectively. In both cases, we can observe that the rewards stay quite stable after approx. 400.000 episodes, so we decided to train the models for 600.000 episodes.

Figure 5 shows the probability that the defenders win for the different scenarios and approaches. The green bars depict the winning probability of the defenders for the communication-based approach, the blue bars show the winning rate for the dynamic shield approach and the gray bars depict the winning rate for the baseline approach.

In the first three scenarios which all consider 2 attackers, the dynamic shield approach outperforms the communication-based approach significantly with win rates of approx. $60 - 90\%$. This observation gets more scattered when increasing the amount of attackers. In general, the performance of all defense mechanisms drops a little with increasing amount of attackers. More importantly, we can observe that the dynamic shield only works well if the number of defenders is equal or larger than the number of defenders. This is due to the fact that the attackers start splitting and attacking the target from all sides, which is difficult to interfere if the defenders are fewer in count. For all scenarios, with equal or larger amount of defenders, our proposed defense mechanisms were able to achieve winning rates of $> 60\%$. The dynamic shield defense presents a more realistic solution to the problem, as it avoids the assumption of communication channel access.

As the MADDPG algorithm is quite complex and was shown to only perform well for small swarm sizes, we limited the amount of agents to a total of 7 in the field. In our future work we plan to introduce algorithm improvements and macro-actions to be able to work with larger swarm sizes. In general, the results look promising and provide a first step towards non-invasive defense mechanisms against attacking UAV swarms.

VI. Conclusion

In this paper, we presented our work on RL-based countermeasures to defend against an intelligent attacking swarm of UAVs. Specifically, we modeled two multi-agent deep RL strategies, where one exploits the attackers' communication channel to inject false data, and the other aims at physically protecting the target by creating a dynamic shield around it. As MADDPG does not perform well with a large swarms, we limited the amount of agents in our games to a total of seven. In future work, we plan to improve the algorithm to test it with larger swarm sizes. Additionally, we will introduce macro-

actions, to reduce the complexity (cp. [19]). Such higher-layer actions could e.g., be: "locate the target", "detect an attacker", "move between attacker and target". Besides the ML-based approach, we are also exploring evolutionary algorithms to get a good comparison between different non-invasive defense methods.

ACKNOWLEDGEMENT

This work received funding by the Military Counter Unmanned Aerial Systems project supported by FFG Forte under contract number 873514 and the security research program KIRAS of the Austrian Federal Ministry of Agriculture, Regions and Tourism (BMLRT) under grant agreement number 879709 (KI-Secure).

REFERENCES

[1] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, 2020.

[2] V. Chamola, P. Kotesh, A. Agarwal, N. Gupta, M. Guizani *et al.*, "A comprehensive review of unmanned aerial vehicle attacks and neutralization techniques," *Ad hoc networks*, vol. 111, p. 102324, 2021.

[3] Y. Hou, X. Liang, L. He, and J. Zhang, "Time-coordinated control for unmanned aerial vehicle swarm cooperative attack on ground-moving target," *IEEE Access*, vol. 7, pp. 106 931–106 940, 2019.

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

[5] 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.

[6] E. R. Goossen and S. D. Martinez, "Catch and snare system for an unmanned aerial vehicle," patentus 8 375 837B2. [Online]. Available: https://patents.google.com/patent/US8375837B2/en

[7] B. R. Van Voorst, "Counter drone system," Sep. 14 2017, uS Patent App. 15/443,143.

[8] D. He, G. Yang, H. Li, S. Chan, Y. Cheng, and N. Guizani, "An effective countermeasure against uav swarm attack," *IEEE Network*, vol. 35, no. 1, pp. 380–385, 2020.

[9] Y. L. Chan *et al.*, "Dynamic geo-fence for drone," *US Appl*, no. 14/951,533, 2017.

[10] G. Lykou, D. Moustakas, and D. Gritzalis, "Defending airports from uas: A survey on cyber-attacks and counter-drone sensing technologies," *Sensors*, vol. 20, no. 12, p. 3537, 2020.

[11] Q. Liu, M. He, D. Xu, N. Ding, and Y. Wang, "A mechanism for recognizing and suppressing the emergent behavior of uav swarm," *Mathematical Problems in Engineering*, 2018.

[12] W. Chen, X. Meng, J. Liu, H. Guo, and B. Mao, "Countering large-scale drone swarm attack by efficient splitting," *IEEE Transactions on Vehicular Technology*, 2022.

[13] J. Simonjan, S. R. Probst, and M. Schranz, "Inducing defenders to mislead an attacking uav swarm," in *Proceedings of the IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2022, pp. 278–283.

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

[15] Y. Wei, X. Nie, M. Hiraga, K. Ohkura, and Z. Car, "Developing end-to-end control policies for robotic swarms using deep q-learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 23, no. 5, pp. 920–927, 2019.

[16] B. G. León and F. Belardinelli, "Extended markov games to learn multiple tasks in multi-agent reinforcement learning," *arXiv preprint arXiv:2002.06000*, 2020.

[17] W. Zhou, Z. Liu, J. Li, X. Xu, and L. Shen, "Multi-target tracking for unmanned aerial vehicle swarms using deep reinforcement learning," *Neurocomputing*, vol. 466, pp. 285–297, 2021.

[18] L. Xiang and T. Xie, "Research on UAV swarm confrontation task based on MADDPG algorithm," in *Proceedings of the 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. IEEE, 2020.

[19] B. Wang, S. Li, X. Gao, and T. Xie, "Uav swarm confrontation using hierarchical multiagent reinforcement learning," *International Journal of Aerospace Engineering*, 2021.

[20] G. Zhang, Y. Li, X. Xu, and H. Dai, "Efficient training techniques for multi-agent reinforcement learning in combat tasks," *IEEE Access*, vol. 7, pp. 109 301–109 310, 2019.

[21] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, and J. Liu, "Uav air combat autonomous maneuver decision based on ddpg algorithm," in *Proceedings of the IEEE 15th international conference on control and automation (ICCA)*. IEEE, 2019, pp. 37–42.

[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] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.