Agent Coordination in Air Combat Simulation using Multi-Agent Deep Reinforcement Learning

1st Johan Källström
Dept. of Computer and Information Science
Linköping University
Linköping, Sweden
johan.kallstrom@liu.se

2nd Fredrik Heintz
Dept. of Computer and Information Science
Linköping University
Linköping, Sweden
fredrik.heintz@liu.se

***Abstract*—Simulation-based training has the potential to significantly improve training value in the air combat domain. However, synthetic opponents must be controlled by high-quality behavior models, in order to exhibit human-like behavior. Building such models by hand is recognized as a very challenging task. In this work, we study how multi-agent deep reinforcement learning can be used to construct behavior models for synthetic pilots in air combat simulation. We empirically evaluate a number of approaches in two air combat scenarios, and demonstrate that curriculum learning is a promising approach for handling the high-dimensional state space of the air combat domain, and that multi-objective learning can produce synthetic agents with diverse characteristics, which can stimulate human pilots in training.**

***Index Terms*—agent-based modeling, intelligent agents, machine learning, multi-agent systems**

I. Introduction

Conducting air combat training using only real aircraft is difficult, because of the high costs of flying, air space regulations, and limited availability of platforms representative of those used by opposing forces. Instead, simulations can be used to replace some human players with synthetic, computer controlled entities. This can lower the costs of training, reduce the dependency on human training providers (see Fig. 1), and improve training value [1]. Ideally, the opponents of trainee pilots should all be synthetic entities, so that role-players and real aircraft are not required to support training. However, to achieve a high training value synthetic opponents must be controlled by high-quality behavior models, and exhibit human-like behavior. Building such models by hand is recognized as a very challenging task [2], [3].

In recent years, the performance of reinforcement learning algorithms has improved rapidly. By combining reinforcement learning with deep learning it has become possible to achieve impressive results in complicated control tasks [4]–[6], classic board games [7]–[9], and challenging real-time, multi-player computer games [10], [11]. This leads us to believe that reinforcement learning could also be a viable option for constructing behavior models for synthetic agents in air combat simulation. With such an approach, users of training systems would not need to explicitly program the behavior of agents, but could instead simply specify their required goals and characteristics. However, not many studies have yet been performed to evaluate the performance of the latest approaches for multi-agent learning in the air combat domain.

This work was partially supported by the Swedish Governmental Agency for Innovation Systems (NFFP7/2017-04885), and the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

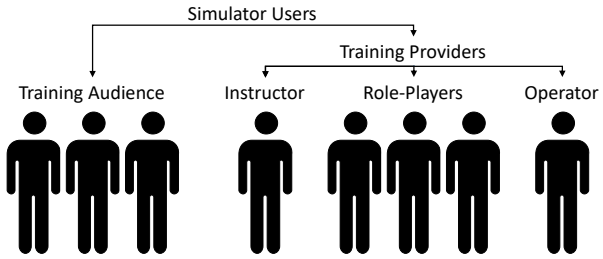


Fig. 1. Users of air combat training systems. By constructing smarter synthetic agents, the need for human training providers can be reduced.

In this work, we study how multi-agent deep reinforcement learning can be used to learn coordination in air combat simulation. Coordination of multiple agents is important in the air combat domain, since pilots never fly alone. Our contributions can be summarized as follows:

- Firstly, we discuss use cases, design principles, and challenges for reinforcement learning algorithms in the domain of air combat simulation intended for training of pilots
- Secondly, we perform an extensive empirical evaluation of approaches that could help realize the identified use cases, using a high-fidelity simulation engine

Specifically, we study two challenges to learning algorithms in air combat simulation scenarios: *1) Learning with sparse rewards*, and *2) Creating agents with adjustable behavior*. Our experiments demonstrate that curriculum learning can facilitate learning with sparse rewards in the high-dimensional state space of air combat, while multi-objective learning can produce agents with diverse behavioral characteristics, which can stimulate pilots in training.

II. Related Work

Due to the challenges associated with building behavior models for air combat simulation by hand, there has been much interest in learning approaches. Since the availability of data from real air combat, as well as data from simulated air combat with manned aircraft and simulators, is limited, approaches that can learn autonomously from interacting with a simulator (without human supervision) are of particular interest.

Bugajska et al. used an evolutionary machine learning algorithm to model the reactive aspects of agents' tasks, allowing the system to learn from experience [12]. The machine learning algorithm was merged with a cognitive model, which controlled higher level tasks like planning and strategy assessment, and also constrained the action space of the reactive part to improve the realism of the behavior.

Yao et al. combined Grammatical Evolution (GE) and modular Behavior Trees (BT) [13] to develop adaptive human behavior models [14]. The BTs were initially encoded with expert knowledge provided by subject matter experts (SMEs), and then evolve through GE during simulation. The efficiency of the method was studied through simulation of 1-vs-1 Beyond Visual Range (BVR) scenarios.

Teng et al. studied the use of self-organizing neural networks to provide adaptive agents that could learn air combat maneuvering strategies for 1-vs-1 dogfight scenarios [15], [16]. The doctrine used to drive a non-adaptive agent was extracted, and used to define the state and action space of the adaptive agent. The studied techniques allowed the adaptive agent to out-maneuver an agent controlled by hand crafted behaviors.

Toubman used Dynamic Scripting (DS), a reinforcement learning technique originally developed for providing intelligent computer controlled opponents in computer games, to generate behavior for agents [17]. An advantage of DS over, e.g., neural networks is that DS can produce behavior models with greater transparency.

An important milestone in deep reinforcement learning in general was the Deep Q Network (DQN) algorithm [18]. This algorithm uses Convolutional Neural Networks (CNN) to approximate the Q function of a Markov Decision Process (MDP), allowing an agent to play classic video games from pixel input. Since then, deep reinforcement learning has become the state-of-the-art approach for learning sequential decision making in complex domains. Some of the developed techniques have been studied in the context of air combat, but still in quite simple scenarios, see e.g., [19]–[21].

In previous work related to learning in the air combat domain, much attention has been given to 1-vs-1 scenarios, and often with only one learning agent. This is highly limiting, since pilots never fly alone, but instead act in groups of at least two aircraft. Our work in this paper is intended as a step towards shifting focus from single-agent to multi-agent learning. We address two open challenges identified in [21]: Multi-agent learning with sparse rewards, which are common in air combat scenarios, and combined multi-objective and multi-agent learning to produce agents with diverse characteristics.

III. Preliminaries

A. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning, which studies agents that learn by interacting with their environment [22]. Reinforcement learning problems can be modelled as Markov Decision Processes, which are defined as tuples (S, A, T, R, γ), specifying:

- S: The set of states of the process
- A: The set of actions of the process
- T: The transition dynamics of the process
- R: The reward function of the process
- γ: The discount factor indicating the importance of immediate and future rewards respectively

The agent interacts with its environment by selecting actions according to its policy ($a_t = \pi(s_t)$), and observes the resulting environment state (s_{t+1}) and the received reward (r_{t+1}). The objective of the agent is to maximize its future expected return:

$$V_\pi(s) = E[R_t|s_0 = s] = E[\sum_{t=0}^{\infty} \gamma^t r_t|s_0 = s] \tag{1}$$

$V_\pi(s)$ is the state value function, which specifies the value of being in state s and then following the policy π. We can also define a state-action value function Q, which specifies the value of taking action a in state s and then following policy π:

$$Q_\pi(s, a) = E[\sum_{t=0}^{\infty} \gamma^t r_t|s_0 = s, a_0 = a] \tag{2}$$

Reinforcement learning algorithms can be divided into value function learning, policy learning, and actor-critic learning. Value function learning aims to find a value function, e.g., the Q function, and then use it to guide action selection. Policy learning aims to learn a policy directly, without first learning a value function. Actor-critic learning learns a value function (the critic), and then uses it to direct updates of a policy (the actor).

In simple environments, the agent's policy can be represented by a table. In more complex environments, e.g., environments with continuous states and actions, an approximation must be used, e.g., a neural network. One example of an actor-critic algorithm for deep reinforcement learning is the Deep Deterministic Policy Gradient (DDPG) algorithm [4], which can learn policies for environments with continuous actions.

In complex environments it may be challenging to explore and find a good policy. One technique for addressing this challenge is curriculum learning [23]. In curriculum learning the problem is broken down into a sequence of tasks of increasing difficulty. Hopefully, the experience gained in simpler tasks can help the agent learn more efficiently when faced with more difficult tasks later in the learning process. One example of curriculum learning in goal-oriented environments, i.e., environments where it is desired to reach a certain goal state, is to use start states increasingly far from the goal state [24].

B. Multi-Agent Reinforcement Learning

Reinforcement learning can be extended to environments with multiple learning agents. Such environments can be fully cooperative, when agents are trying to optimize a shared reward, or fully competitive when agents have opposite goals. Mixed cooperative and competitive settings are also possible [25]. When several agents learn concurrently, the environment may become non-stationary from a single agent's point of view, i.e., the environment dynamics change over time. Parts of the environment state may also not be observable, e.g., internal states of other agents. This makes it challenging to learn multi-agent coordination in complex environments.

In recent years, there has been an increasing interest in multi-agent deep reinforcement learning, and development of efficient algorithms for learning in multi-agent environments [26]. One proposed approach is actor-critic learning with centralized learning and decentralized execution. One such algorithm is the Multi-Agent Deep Deterministic Policy Gradient algorithm (MADDPG) [27], in which each agent is assigned a centralized critic to guide updates of the agent's policy. During training, the critic has access to the observations and actions of all agents in the system, while at test time each agent selects actions based on its local observations alone.

C. Multi-Objective Reinforcement Learning

Most reinforcement learning algorithms try to maximize the return for a scalar reward signal. However, many real-world problems deal with multiple, possibly conflicting objectives, which may not be easily expressed as a scalar reward. In multi-objective reinforcement learning (MORL) [28], the Markov Decision Process is extended to a Multi-Objective Markov Decision Process (MOMDP), which has a vector-valued reward signal, resulting in vector-valued value functions. Each element in the reward vector represents the reward received for one of the objectives. To compare the performance of policies, a scalarization function is used to convert the vector value function ($\mathbf{V}_\pi(s)$) to a scalar, e.g., by calculating the weighted sum of the objective values:

$$V_\pi^{\mathbf{w}}(s) = f(\mathbf{V}_\pi(s), \mathbf{w}) = \sum_{i=1}^{n} v_i(s) w_i \tag{3}$$

In [28], three use cases for multi-objective reinforcement learning are presented:

- **The unknown priorities scenario**: The priorities among objectives are not known at training time, so the scalarisation of rewards must be delayed to test time
- **The decision support scenario**: The priorities among objectives are known at training time, but it is difficult to express them in the form of a reward function, so instead we want to train a set of policies and study their behavior before deciding which one to use
- **The known priorities scenario**: The priorities among objectives are known at training time, but it is difficult or infeasible to solve the problem without using explicit multi-objective methods

D. Reinforcement Learning in Air Combat Simulation

Based on interviews with experienced pilots, we have identified three major use cases for reinforcement learning in the domain of air combat training.

- **UC1 - Simplified content production**: By using reinforcement learning, it could become easier to produce simulation contents. Instead of explicitly programming synthetic entities, instructors could use a high-level language to define the behavior or goals of agents. By providing a detailed specification, agents can be made to act according to a specific doctrine. One way of providing such a specification is by expert demonstration, i.e., having an expert human pilot perform a task, and then using the recorded data for training of synthetic agents.
- **UC2 - More advanced synthetic entities**: By using reinforcement learning, it could become possible to create more advanced synthetic entities. These entities could challenge experienced pilots, help them improve their skills, and possibly discover flaws in human developed tactics. Thus, they could reduce the need for human role-players to participate in training sessions.
- **UC3 - More diverse and adaptive synthetic entities**: By using reinforcement learning, it could become possible to create synthetic entities with diverse behavior, adapted to current training needs, e.g., the proficiency level of trainees. The behavior of such entities could be selected by an instructor, or (preferably) the entities themselves could adjust their behavior based on observations and inferred training needs.

The characteristics of each use case will affect the design of the reward system and the training process. For use case 1, we want to design reward systems that incorporate domain knowledge about opponents' behavior, which will typically lead to dense reward systems, giving frequent feedback to the agent. For use case 2, we want to use highly abstract and sparse reward systems, as to not introduce a bias in the agent's behavior, which may prevent the agent from finding an optimal policy. For use case 3, we want algorithms that produce agents that can pursue multiple different objectives after training, e.g., by using parameterized policies and exposing the agent to diverse environments during the learning process.

Learning sequential decision making in the air combat training domain is challenging. In typical scenarios there are many interacting agents, which are competing in teams, using complex systems. There is only partial observability, because of performance limitations of sensors and data links, and possibly effects of electronic warfare. The policies learned by agents while interacting with other agents must be robust, so that they can interact effectively with humans in training sessions. Finally, to appear realistic and to be suitable for multiple simulation scenarios, synthetic agents must have diverse behavior, and be able to prioritize among multiple conflicting objectives, e.g., tactical mission goals, resource consumption and safety, with priorities possibly varying over time.

IV. Method

A. Evaluation Scenarios

To study the performance of multi-agent deep reinforcement learning in air combat simulation we define two simulation scenarios: *Aerial Reconnaissance with Sparse Rewards* and *Airstrike with Adjustable Risk Taking*. These scenarios address the challenges of learning with infrequent feedback in complex environments (related to **UC2**), and designing dense reward systems for learning agents, whose behavior can then be affected after training time to improve diversity in the simulation (related to **UC1** and **UC3**).

In the aerial reconnaissance scenario, two agents should visit three Points of Interest (POI) as quickly as possible. The search area has radius 20 km, and the agents spawn in random positions on its perimeter, while POIs spawn in random positions within the perimeter. To solve the problem efficiently, the agents must coordinate their actions. The agents are only rewarded for visiting the points of interest, which in combination with the size of the search area makes learning challenging. Initial positions of aircraft (black stars) and POIs (green circles) in an episode are shown in Fig. 3a.

In the airstrike scenario, a strike aircraft should navigate to a target location protected by an air defense system. A second, escort aircraft, is equipped with a jamming pod. This aircraft can align with the first aircraft and use its jammer to reduce the ability of the air defense's radar to position the two approaching aircraft (see Fig. 2). In this work, we limit ourselves to studying movement and geometry in the horizontal plane, but for maximum efficiency the target and the strike aircraft should be aligned on the jammer Line-of-Sight (LOS) in 3D space. In each episode, the target location is initialized in a random position, while the two aircraft are initialized in random positions with random heading, within two rectangular areas. To produce agents with diverse characteristics, we would like to be able to adjust the level of risk taking of the agents after training. Rectangular spawn areas of target in blue and aircraft in green, as well as initial positions of aircraft (black stars) and target (black circle) in one episode are shown in Fig. 3b. The range of the air defense system co-located with the target is shown by the red circle.

B. Common Experiment Design

The simulation engine used is a high-fidelity simulator, which is part of an operational air combat training system. We use the MADDPG algorithm to train the agents in both scenarios. We use a learning rate of $\alpha = 10^{-2}$, and train for 50000 episodes using the Adam optimizer [29]. Episodes are limited to a maximum of 300 time steps, and each time step is 1 second long. All simulation results are averaged over five runs with different random seeds. The policy is represented by a multilayer perceptron (MLP), with 2 hidden layers, each with 64 neurons and the ReLU activation function. In the observation space of the agents, positions are given in a body-fixed coordinate system, while headings are given relative true north, as illustrated in Fig. 3c.

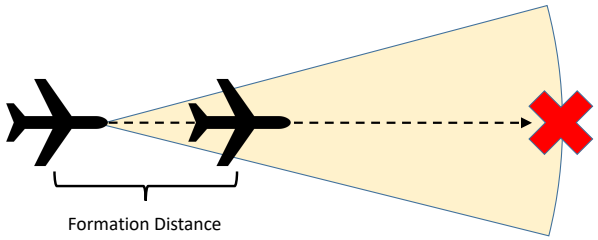


Fig. 2. Alignment of strike and escort aircraft towards target.

All elements of the observation vector o_t are normalized by their expected maximum value. The complete observation space of each agent, which is the input to the neural network representing the agent's policy, is the set of observations from the last four time steps:

$$O_t = \begin{bmatrix} o_t \\ o_{t-1} \\ o_{t-2} \\ o_{t-3} \end{bmatrix} \tag{4}$$

C. Scenario Specific Experiment Design

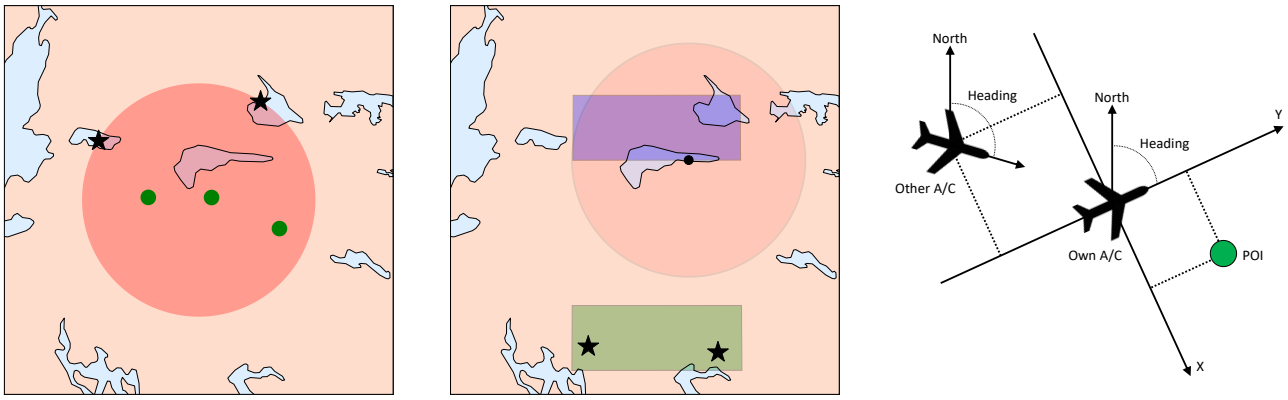
1) Aerial Reconnaissance with Sparse Rewards: In this scenario, we want the two agents to visit the three POIs as quickly as possible. It is not obvious how to construct a dense reward system that effectively represents this goal. Instead, we use a sparse reward signal for each agent defined as:

$$r_t = 50.0 \cdot n_{POI}(t) - 1.0 \tag{5}$$

Here $n_{POI}(t)$ is the number of POIs visited (within 1 km) by the agent in the time step. The shared negative reward of -1.0 in each time step is intended to motivate the agents to cooperate, and complete the episode as quickly as possible.

To help the agents learn in spite of the sparse rewards and large search area, we use curriculum learning. Inspired by [24], we construct learning curricula where the agents are exposed to search areas with increasingly large radius. Before each episode of training, the radius of the search area is sampled from a uniform distribution over the interval $r_i = [r_{min}, r_{max}]$. We study three different learning curricula, each of which updates r_i three times during training, according to the number of completed episodes specified in $cur_{switch} = [n_{ep_1}, n_{ep_2}, n_{ep_3}]$, with values according to Tab.I. With these learning curricula we intend to investigate how the fraction of small and large search areas used during training affects the final performance of the agents. Compared to Curriculum 1, Curriculum 2 trains for a longer time on medium sized search areas. In contrast to the other two curricula, Curriculum 3 keeps the minimum radius of the search are at 5 km throughout training.

In each time step an agent observes the observation vector o_t, which contains the agent's own heading, the other agent's position and heading, the positions of the POIs, and information about which POIs have been visited by either of the agents (indicated by 0 for visited, and 1 for not visited).



(a) Evaluation Scenario 1. (b) Evaluation Scenario 2. (c) Coordinate system of observations.

Fig. 3. Evaluation scenarios and coordinate system of observations.

TABLE I
CURRICULA USED IN EVALUATION SCENARIO 1.

Curriculum Type	**Curriculum Switch Schedule and Search Area Radius Intervals**				
	Switches	r_0	r_1	r_2	r_3
Cur. 1	$[0, 10k, 20k, 30k]$	$[5\ km, 5\ km]$	$[5\ km, 10\ km]$	$[10\ km, 15\ km]$	$[10\ km, 20\ km]$
Cur. 2	$[0, 10k, 30k, 45k]$	$[5\ km, 5\ km]$	$[5\ km, 10\ km]$	$[10\ km, 15\ km]$	$[10\ km, 20\ km]$
Cur. 3	$[0, 10k, 25k, 40k]$	$[5\ km, 5\ km]$	$[5\ km, 10\ km]$	$[5\ km, 15\ km]$	$[5\ km, 20\ km]$

Each agent has a continuous action space, which allows it to turn left or right with a load factor in the interval $[2, 4]$ g.

We train the agents using a discount factor of $\gamma = 0.95$. Episodes end if all POIs have been visited within a range of 1 km. We evaluate the approach by studying the trained agents' performance for the three different learning curricula in 1000 simulations. For comparison, we also train agents without learning curricula, on environments with radii of 5 km, 10 km, and 20 km. After training, we evaluate these agents' performance on the training task as well as the original goal task, i.e., the search area with radius 20 km. We also evaluate the performance of the curriculum learning agents on the search area with radius 5 km.

2) Airstrike with Adjustable Risk Taking: In this scenario, we want the agents to learn how to prioritize between two conflicting objectives: Time and Safety. We implement the environment as an MOMDP, with a vector reward signal for each agent defined as:

$$\mathbf{r}_t = [r_{safe}(t), r_{time}(t), r_{tgt}(t)] \tag{6}$$

Here r_{safe} and r_{time} are the rewards given for the Safety and Time objectives respectively. The Safety objective is achieved by flying in close formation and aligning towards the target, to enable the escort aircraft to use its jammer to reduce the air defense's ability to position the approaching aircraft. The Time objective is achieved by reaching the target as quickly as possible. The two objectives are captured by the following reward design:

$$r_{safe}(t) = r_{form}(t) + r_{align}(t) \tag{7}$$

$$r_{time}(t) = -1.0 \tag{8}$$

In addition to the two major objectives, we also use a potential-based reward shaping [30] element in the reward vector (r_{tgt}), to make learning more efficient. This element rewards the agents for reducing the distance to the target (d_{tgt}), and thus guides the exploration of the agents:

$$r_{tgt}(t) = d_{tgt}(t-1) - d_{tgt}(t) \tag{9}$$

To encourage the agents to fly in close formation, they are given a penalty proportional to the distance between the escort aircraft and a formation reference point relative the strike aircraft:

$$d_{form}(t) = \|p_{escort}(t) - p_{ref}(t)\| \tag{10}$$

$$r_{form}(t) = -0.2 \cdot d_{form}(t) \tag{11}$$

Here p_{escort} and p_{ref} are the positions of the escort aircraft and the formation reference point respectively, d_{form} is the distance between the escort and its reference point, and r_{form} is the reward component given for flying in close formation.

To encourage the agents to align towards the target, they are given a reward dependent on the alignment error, when the positions of the strike aircraft and the target (p_{strike} and p_{target}) are both within the Field-of-View (FOV) of the escort aircraft's jammer:

$$jam_{\delta}(t) = jam_{\delta_s}(t) + jam_{\delta_{tgt}}(t) \tag{12}$$

$$r_{jam}(t) = 0.5 \cdot (jam_{fov}(t) - jam_{\delta}(t))/jam_{fov}(t) \quad (13)$$

$$r_{align}(t) = \begin{cases} r_{jam}(t) & \text{if } p_{strike} \text{ and } p_{tgt} \text{ in } jam_{fov} \\ 0 & \text{ELSE} \end{cases} \quad (14)$$

Here jam_{fov} is the FOV of the escort aircraft's jammer, jam_{δ_s} and $jam_{\delta_{tgt}}$ are the absolute angular alignment errors between the jammer center line and the positions of the strike aircraft and target respectively, and $r_{jam}(t)$ is a reward component given for using the jammer effectively. $r_{align}(t)$ is the reward component given for aligning towards the target.

In these experiments, the formation reference point for the escort aircraft was placed 5 km behind the strike aircraft, on a line passing through the position of the strike aircraft and the position of the target. The jammer FOV was set to 60 degrees. The scale factors in the reward components were selected to give the components comparable magnitude for this scenario, i.e., no component should completely dominate the others.

To specify the priorities of the Time and Safety objectives, we use the weight θ. We then use the following scalarization function to convert the vector reward to a scalar (with $\theta \in [0, 1]$):

$$r_t = \theta \cdot r_{time} + (1 - \theta) \cdot r_{safe} + r_{tgt} \quad (15)$$

The resulting scalar reward r_t is then used as input to MADDPG for training of the agent's policy. θ is included in the input to the agent's policy, so that it can be used to adjust the agent's behavior after training, i.e., we want the agent to learn how the value of θ affects its reward. To allow the agents to learn how to prioritize between Time and Safety, we sample θ from a uniform distribution over the interval $[0.2, 0.8]$ before each episode of training.

In each time step an agent observes the observation vector o_t, which contains its own heading and speed, the position, heading and speed of the other agent, the position of the formation reference point, the position of the target, and the preference between Time and Safety objectives (θ).

We use a tuple action space:

$$A = A_{goal} \times A_{thrust} \quad (16)$$

A_{goal} is a discrete, two element action space, which allows the agents to move towards each other (the strike aircraft moving towards its escort, the escort moving towards its reference point), or to move towards the target. A_{thrust} is a continuous action space, which allows the agents to set the commanded speed in the interval $[0.4, 1.2]$ Mach.

We train the agents using a discount factor of $\gamma = 1.00$. Episodes end if the strike aircraft comes within 10 km of the target. We evaluate the approach by studying the trained agents' behavior for three different values of θ: $\{0.2, 0.5, 0.8\}$, in 1000 simulations. For comparison, we train and evaluate the same type of agents using the single-agent reinforcement learning algorithm DDPG.

TABLE II
PERFORMANCE IN EVALUATION SCENARIO 1.

Curriculum Type	**Radius of Search Area**		
	$r = 5\ km$	$r = 10\ km$	$r = 20\ km$
Fix. 5 km	58.3 ± 15.7	–	-454.1 ± 165.4
Fix. 10 km	–	-252.1 ± 141.2	-403.7 ± 167.6
Fix. 20 km	–	–	-587.0 ± 26.4
Cur. 1	80.8 ± 56.7	–	**-146.1 ± 136.1**
Cur. 2	60.8 ± 105.0	–	-184.5 ± 156.1
Cur. 3	**81.4 ± 51.0**	–	-177.0 ± 156.2

V. RESULTS

A. Aerial Reconnaissance with Sparse Rewards

The training progress of agents with and without curriculum learning is shown in Fig. 4, as mean and standard deviation of the collected reward of both agents.

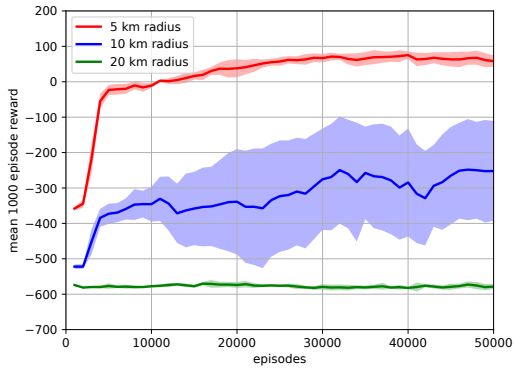
We can see that agents learning without curriculum in the small search area with radius 5 km improve quickly, and then stabilize after about 30000 episodes. Agents learning without curriculum in the medium sized search area with radius 10 km also improve their performance, but display very high variance. This is likely due to the randomness of the exploration process. Finally, the agents learning without curriculum on the goal task (a search area with radius 20 km) do not display any improvement in performance during training.

All the curriculum learning agents improve their performance quickly. They also keep learning after the curriculum switches, but start to plateau towards the end of training. The agents learning with Curriculum 1 seem to make the fastest improvements towards the end of training, after a period of high variance after about 40000 episodes.

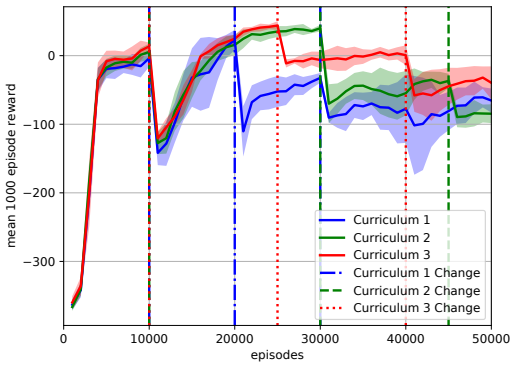
The results of the benchmark simulations for 1000 time steps are shown in Tab. II. We can see that the agents using Curriculum 1 have the best performance on the goal task. This is the curriculum that progresses the quickest to more complicated tasks, and also quickly stops training on the easiest task. Agents learning without a curriculum perform very poorly on the goal task, including the agents that showed strong progress when trained on the simplest task, with a search area with radius of 5 km, who are in fact beaten by the agents that were trained on a search area with radius 10 km. This demonstrates that transfer of learning from simple to complex tasks should not be assumed. It is likely that the agents trained on the smallest search area failed to learn efficient coordination, since it is less important in simple domains. It is also interesting to note that all curriculum learning agents receive higher mean reward than the fixed environment agents on the simplest task. That is, training on harder tasks seems to improve performance on simpler tasks as well.

B. Airstrike with Adjustable Risk Taking

The training progress of agents trained with MADDPG and DDPG is shown in Fig. 5a, as mean and standard deviation of the collected reward of both agents.

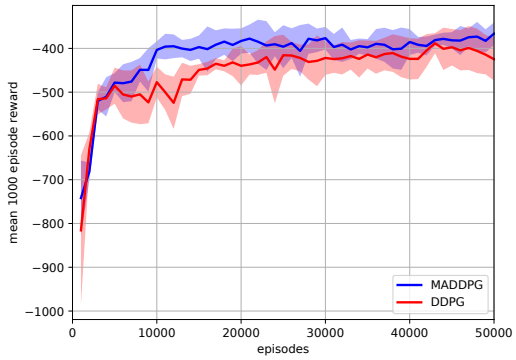


(a) Training progress without curriculum.

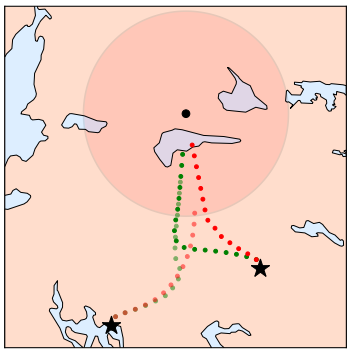


(b) Training progress with curriculum.

Fig. 4. Training progress for Evaluation Scenario 1.



(a) Training progress for MADDPG and DDPG.



(b) Trajectories for safe and fast agents.

Fig. 5. Training progress and emergent behavior for Evaluation Scenario 2.

We can see that the agents trained with MADDPG improve quickly and start to plateau already after 10000 episodes, while the agents trained with DDPG need more than 40000 episodes to reach similar performance, i.e., using a dedicated multi-agent learning algorithm in addition to the shared reward of the agents significantly improves performance. The results of the DDPG agents also have higher variance, and the mean rewards dip towards the end of training, which may be explained by the non-stationarity resulting from multiple agents learning and updating their policies within the same system.

The results of simulations with three different values of θ for 1000 time steps are shown in Tab. III. We can see that the qualitative results are as desired for the MADDPG agents: When θ is small, priority is given to the Safety objective, when θ increases, more priority is given to the Time objective. This effect is present for the DDPG agents as well, but it is not as prominent.

Fig. 5b shows the emergent behavior of agents after training in an example episode. Trajectories for safe agents ($\theta = 0.2$) are shown in green and trajectories for fast agents ($\theta = 0.8$) are shown in red. The escort aircraft starts in the bottom left corner for this episode. Safe agents align before entering the range of the air defense system (taking 179 seconds to finish the episode), while fast agents enter the risk area before aligning properly (taking 141 seconds to finish the episode).

VI. Conclusion

In this paper, we studied applications of multi-agent deep reinforcement learning in the context of air combat simulation intended for pilot training. We demonstrated that curriculum learning is a promising approach for handling the complexity of the air combat domain. With suitably designed learning curricula, challenging tasks can be solved efficiently. We also demonstrated that multi-objective learning is a promising approach for creating agents with diverse characteristics, whose behavior can also be adjusted after learning has completed.

The studied methods can help construct smarter synthetic pilots, adapted to current training needs, and thus reduce the need for human role-players to participate in training sessions. This would improve training value and availability of training.

For complex scenarios, constructing and tuning reward systems and learning curricula by hand can be challenging and time-consuming. In future work, we would like to investigate efficient methods for defining dense reward systems, e.g., learning from demonstration, to simplify this process for non-expert users. We would also like to investigate methods for automated curriculum generation, e.g., by monitoring the performance of learning agents and adapting the simulation environment accordingly. Finally, we would like to proceed to study more complex scenarios, and evaluate human-agent interaction in experiments with manned simulators.

TABLE III
PERFORMANCE IN EVALUATION SCENARIO 2.

Algorithm	**Safety**			**Time**		
Type	$\theta = 0.2$	$\theta = 0.5$	$\theta = 0.8$	$\theta = 0.2$	$\theta = 0.5$	$\theta = 0.8$
DDPG	-507.2 ± 348.6	-525.1 ± 342.3	-561.6 ± 372.3	-346.6 ± 68.5	-338.0 ± 62.8	-337.7 ± 64.7
MADDPG	-358.1 ± 271.3	-399.5 ± 272.5	-485.1 ± 311.6	-396.4 ± 76.8	-349.6 ± 57.4	-339.0 ± 55.9

REFERENCES

[1] J. Roessingh and G. Verhaaf, "Training effectiveness of embedded training in a (multi-) fighter environment," NATIONAL AEROSPACE LAB AMSTERDAM (NETHERLANDS), Tech. Rep., 2009.

[2] J. Thorpe, "Trends in modeling, simulation, & gaming: Personal observations about the past thirty years and speculation about the next ten," in *Interservice/Industry training, simulation, and education conference (I/ITSEC)*, 2010.

[3] T. van den Berg, N. de Reus, and J. Voogd, *LVC Architecture study*. Simulation Interoperability Standards Organization (SISO), 2011.

[4] 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," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[6] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[10] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[11] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[12] M. D. Bugajska, A. C. Schultz, J. G. Trafton, S. Gittens, and F. Mintz, "Building adaptive computer-generated forces: The effect of increasing task reactivity on human and machine control abilities," NAVAL RESEARCH LAB WASHINGTON DC CENTER FOR APPLIED RESEARCH IN ARTIFICIAL INTELLIGENCE, Tech. Rep., 2001.

[13] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.

[14] J. Yao, Q. Huang, and W. Wang, "Adaptive human behavior modeling for air combat simulation," in *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, 2015, pp. 100–103.

[15] T.-H. Teng, A.-H. Tan, Y.-S. Tan, and A. Yeo, "Self-organizing neural networks for learning air combat maneuvers," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–8.

[16] T.-H. Teng, A.-H. Tan, and L.-N. Teow, "Adaptive computer-generated forces for simulator-based training," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7341–7353, 2013.

[17] A. Toubman, "Calculated moves: Generating air combat behaviour," Ph.D. dissertation, Leiden University, 2020.

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

[19] R. Rijken and A. Toubman, "The future of autonomous air combat behavior," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016, pp. 3089–3094.

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

[21] J. Källström and F. Heintz, "Multi-agent multi-objective deep reinforcement learning for efficient and effective pilot training," in *FT2019. Proceedings of the 10th Aerospace Technology Congress, October 8-9, 2019, Stockholm, Sweden*, 2019, pp. 101–111.

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

[23] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[24] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on Robot Learning*, 2017, pp. 482–495.

[25] L. Bu, R. Babu, B. De Schutter *et al.*, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[26] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[27] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[28] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

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

[30] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.