IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 June 2025, accepted 1 August 2025, date of publication 11 August 2025, date of current version 19 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3597199

RESEARCH ARTICLE

Deep Reinforcement Learning Agents With Collective Situational Awareness for Beyond Visual Range Air Combat

JOAO P. A. DANTAS[1,2,*], **FELIPE L. L. MEDEIROS**[2,*], **ADRISSON R. SAMERSLA**[2,*], **PEDRO L. R. BOTELHO**[1], **VITOR C. F. GOMES**[2], **SAMARA R. SILVA**[2], **YURI D. FERREIRA**[2], **ALESSANDRO O. ARANTES**[2], **MARCIA R. C. AQUINO**[2], **AND MARCOS R. O. A. MAXIMO**[1]

[1]Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo 12288-900, Brazil
[2]Instituto de Estudos Avançados, São José dos Campos, São Paulo 12228-001, Brazil

Corresponding author: Joao P. A. Dantas (jpdantas@ita.br)

This work was supported in part by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, under Grant ROR Identifier: 00x0ma614; and in part by the Funding Authority for Studies and Projects (FINEP) under Finance Code 01.20.0196.00/2824/20. The work of Marcos R. O. A. Maximo was supported in part by the National Research Council of Brazil (CNPq) under Grant 307525/2022-8.

*Joao P. A. Dantas, Felipe L. L. Medeiros, and Adrisson R. Samersla contributed equally to this work.

ABSTRACT This work explores Beyond Visual Range (BVR) air combat simulations, focusing on two-versus-two scenarios involving autonomous agents. The engagement phase in BVR combat presents complex and unpredictable situations, as it is difficult to anticipate the behavior of opposing aircraft and the outcomes of tactical decisions, especially in multi-agent settings. A promising approach is the use of Deep Reinforcement Learning (DRL), which enables agents to learn from dynamic environments. According to fighter pilots, collective situational awareness, defined as understanding the spatial distribution and orientation of allies and opponents, is essential for executing coordinated tactical maneuvers. The main contribution of this work is AsaGym, a library for developing and training DRL-based fighter agents in BVR scenarios. A case study demonstrates its use, applying a reward function that promotes coordination based on collective situational awareness, and compares different DRL algorithms to assess their ability to foster cooperative behavior. The results highlight DRL's potential to address the complexities of modern air combat and support the development of more adaptive and effective tactics in multi-agent BVR scenarios.

INDEX TERMS Artificial intelligence, autonomous agents, beyond visual range air combat, deep reinforcement learning, simulation.

I. INTRODUCTION

Air combat is a complex and dynamic scenario where skilled pilots make quick decisions to gain a tactical advantage over their opponents [1]. Beyond Visual Range (BVR) air combat, in particular, involves engagements taking place at distances where pilots cannot see the enemy aircraft [2], [3]. While some air combat still occurs within visual range (WVR), most engagements start in BVR. This phase is often the most important, as it can give advantages or create difficulties for the later stages of combat. The main challenge for pilots is the planning of maneuvers, which shows their ability to think tactically and decide the outcome of the fight [4].

Computer simulations of BVR air combat can recreate many different situations, helping to test new tactics, sensors, and weapons [5]. One of the hardest parts of these simulations is mimicking the complex behaviors of pilots during all stages of combat. These decisions include adapting to new situations, coordinating with allies to execute strategies, and timing missile launches effectively.

This work explores the learning of BVR engagement maneuvers by autonomous agents. Engagement involves maneuvering the aircraft to gain an advantage over the opponent, that is, to position the enemy within the effective

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

range of the aircraft's own missiles, known as the Weapon Engagement Zone (WEZ), while staying outside the opponent's WEZ [6], [7]. The engagement stage becomes even more complex when there is more than one opponent.

A promising solution to this problem is the use of Reinforcement Learning (RL), which allows autonomous agents to learn from challenging experiences. RL is a machine learning method in which an autonomous agent learns to make better decisions by interacting with its environment. The agent receives rewards or penalties for its actions and adjusts its strategy to maximize the rewards over time [8]. Deep Reinforcement Learning (DRL) is a more advanced form of RL that uses deep neural networks to manage complex environments, enabling agents to make decisions in dynamic and uncertain conditions, such as BVR air combat [9].

In this context, existing simulation environments for air combat often lack modularity, support for multi-agent DRL experimentation, or mechanisms to incorporate operational insights such as spatial coordination between allied and enemy aircraft.

Therefore, the main contribution of this work is:

- Development of AsaGym, a library for simulating and training DRL-based autonomous fighter agents in BVR air combat, with a case study demonstrating its use with a reward design that promotes coordination and situational awareness.

In addition, we provide the following specific contributions:

- Design of a task-oriented reward function that encourages cooperative agent behavior based on spatial relationships among allies and opponents.
- Incorporation of operational knowledge from Brazilian Air Force (FAB) fighter pilots, serving as Subject Matter Experts (SMEs), who emphasized the importance of spatial awareness for coordinated maneuvers in BVR combat.
- Comparative evaluation of four state-of-the-art DRL algorithms – Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), Twin Delayed Deep Deterministic policy gradient (TD3), and Advantage Actor-Critic (A2C) – applied to the engagement phase in a simulated BVR air combat scenario.

The remainder of this work is organized as follows. Section II presents an overview of related work, highlighting previous research on DRL applications in BVR air combat simulations. Section III details the proposed methodology, including the design of the DRL models used to represent the engagement phase of BVR combat, and the experimental setup used for training and evaluation. The results and analysis of the conducted experiments are discussed in Section IV, providing insights into the agent's performance across different scenarios. Finally, Section V summarizes the key findings and outlines potential directions for future work.

II. RELATED WORK

RL has been widely used to model the engagement stage in constructive BVR air combat simulations. Most studies employ two types of rewards: intermediate and final. The intermediate reward is provided during training whenever the agent makes a decision, reflecting its current tactical advantage or disadvantage relative to opponents. The final reward is assigned at the end of the training episode, often based on the difference between the number of surviving friendly and opposing agents. A positive value indicates a favorable outcome, while a negative value indicates an unfavorable one. In this context, a group of multiple aircraft is referred to as a swarm.

This section focuses on behavior modeling with DRL, particularly in terms of reward design; for a more detailed review of simulation frameworks, we refer the reader to [10]. A review of references that applied RL to model BVR air combat behaviors is summarized in Table 1. The second column indicates the BVR air combat setup, such as one-versus-one (1v1), two-versus-two (2v2), or n-versus-one (nv1). The third column lists the RL methods used in each study. The fourth and fifth columns indicate whether the works employed functions to compute intermediate and final rewards, respectively.

In [11], an intermediate reward function was used based on angles between the aircraft, the distance relative to the WEZ, radar range from one aircraft's perspective, and altitude differences.

In [12], an intermediate reward considered angles between the aircraft, altitude relative to a safe threshold, and speed in relation to the minimum required to avoid a stall. The final reward penalized the agent if it was eliminated by the opponent at the end of the episode.

Reference [13] proposed an intermediate reward that assigned positive or negative values based on events such as missile launches, stall situations, radar tracking (both active and passive), missile warnings, and evasion maneuvers. The final reward reflected outcomes like eliminating the opponent, being eliminated, collisions, and victory or defeat conditions.

In [4], an intermediate reward function was employed, considering factors such as the distance from the simulation boundaries, angles and distances between the aircraft, and the likelihood of eliminating the opponent or being eliminated.

Reference [14] proposed a final reward function focused solely on the final outcome, calculated as the difference in the number of surviving aircraft between the opposing swarms.

The study presented in [15] employed an intermediate reward function based on the angles and distances between the aircraft.

In [16], the authors introduced a two-stage maneuver control system: the first stage focused on swarm control, using metrics such as separation distance (for dispersion) and grouping distance (for cohesion); while the second stage handled engagement using reinforcement learning.

TABLE 1. Review of references that applied RL to model BVR air combat behaviors.

Reference	Air Combat Setup	RL Methods	Intermediate Rewards	Final Rewards
[11]	1v1	Improved Q-Network (IQN)	✓	
[12]	1v1	TD3	✓	✓
[13]	1v1	Key Air Combat Event Reward Shaping (KAERS) and PPO	✓	✓
[4]	1v1	Long Short-Term Memory (LSTM) and Deep Q-Network (DQN)	✓	
[14]	6v6	Neuroevolution-based Simulation Optimization (NSO)		✓
[15]	1v1	Double Deep Q-Network (DDQN)	✓	
[16]	nv1	Deep Deterministic Policy Gradient (DDPG)	✓	
[17]	2v1	DQN	✓	✓
[18]	1v1	Generative Adversarial Imitation Learning (GAIL) and Multi-Agent Deep Deterministic Policy Gradients (MADDPG)	✓	
[19]	1v1	SAC and Parallel Self-Play (PSP)	✓	
[20]	4v4	Deep Relationship Graph Reinforcement Learning (DRGRL)	✓	✓
[21]	2v1	Multi-Agent Proximal Policy Optimization (MAPPO)	✓	✓
[22]	1v1	Monte Carlo Tree Search (MCTS) and Deep Neural Network (DNN)		✓
[23]	1v1	Asynchronous Advantage Actor-Critic (A3C)	✓	✓
[24]	1v1	PPO	✓	✓
[25]	1v1	Dueling Double Deep Q-Network (D3QN)	✓	✓
[26]	1v1	Expert-Soft Actor-Critic (E-SAC)	✓	
[27]	1v1	PPO		✓
[28]	2v2	Advantage Highlight Multi-Agent Proximal Policy Optimization (AHMAPPO)	✓	
[29]	1v1	Three-level Hierarchical decision framework embedding Expert knowledge (H3E)	✓	✓
[30]	1v1	Self-play DRL	✓	✓

Although the scenario involved a swarm versus a single opponent, the intermediate reward function evaluated each agent individually based on distances and angles relative to the opponent.

In [17], the intermediate reward was based on angles and distances between the agent and the opponent. The final reward assigned a positive value if the agent eliminated the opponent and a negative value (penalty) if the agent was eliminated.

The intermediate reward proposed in [18] considered factors like angles, altitude differences, and speed variations between the agent and the opponent.

In [19], the intermediate reward function was based on angles and distances between the agent and the opponent.

The intermediate reward function proposed in [20] considered events such as radar tracking (active and passive), missile firing, and missile evasion. The final reward was defined as the difference between the number of surviving allies and opponents at the end of the episode.

Similar to other swarm-based studies, the intermediate reward function proposed in [21] considered angles, distances, speed, and altitude differences relative to the opponent. The final reward reflected events such as successful tracking, missile engagements, and elimination outcomes.

In [22], the final reward function assigned a value of 1 if the agent eliminated the opponent without being eliminated, -1 if the agent was eliminated, and 0 in the case of no eliminations or mutual elimination.

In [23], the authors designed an intermediate reward considering altitude, speed, distance, and angles between the agent and the opponent. The final reward assigned positive values for successful eliminations and negative values for defeats or ties.

Both intermediate and final rewards were explored in [24]. The intermediate reward considered missile launches, angle relations, distances, and altitude differences, while the final reward assigned positive values for victories and negative values for defeats.

Reference [25] applied a reward function that combined an intermediate reward, focused on distances and angles between the agent and the opponent, and a final reward, with positive values for eliminating the opponent, negative values for being eliminated, and zero for ties.

The intermediate reward function used in [26] was based on angles and distances between the agent and the opponent.

DRL was specifically applied in [27] to model defensive maneuvers against incoming missiles in 1v1 BVR air combat. The reward was based on the effectiveness of defensive actions, assigning positive values for successful evasion (based on the distance from the missile) and zero if the agent was hit.

A sparse intermediate reward function was employed in [28] based on events like radar tracking and missile launches by individual aircraft. However, it did not account for the spatial distribution or formation orientation of aircraft within the swarms.

In [29], intermediate reward was used for actions such as missile firing, radar tracking, evading enemy tracking, and missile avoidance. The final reward assigned a value of 1 for eliminating the opponent or surviving a specific duration and -1 if the agent was eliminated.

Reference [30] employed an intermediate reward function based on missile usage, distances, angles, and altitude differences. The final reward considered missile effectiveness, collision events, boundary violations, and navigation performance.

III. METHODOLOGY

This section presents the methodology for developing and evaluating our RL environment and agents, focusing on the AsaGym library, which supports fighter agent training in air combat scenarios within the Aerospace Simulation Environment (ASA) framework [31], [32]. It covers the integration of key modules, the design of observation and action spaces, the implementation of the reward function, and episode termination criteria. Additionally, it presents the scenario configurations, initialization sampling methods, and custom wrappers that enhance training flexibility. The methodology also includes rendering methods for validation and debugging, the training setup with selected algorithms and hyperparameters, and the computational infrastructure. Finally, evaluation metrics are introduced to determine the most effective algorithm choices based on performance indicators such as episode rewards, episode lengths, and total training time.

A. ASAGYM

The AsaGym computational library is essential to our methodology, providing a custom environment built on top of the Gymnasium framework specifically for training fighter agents using DRL. This subsection details the development and integration of AsaGym within ASA. AsaGym extends the capabilities of Gymnasium by introducing specialized modules that facilitate the training of Artificial Intelligence (AI) agents in complex air combat scenarios.

1) LIBRARY DEVELOPMENT

The open-source Python module Gymnasium [33] has emerged as a significant tool for RL researchers, replacing the widely used OpenAI Gym [34]. Gymnasium builds upon the foundation laid by OpenAI Gym, offering a robust and flexible framework for developing and comparing RL algorithms.

Gymnasium provides a comprehensive set of pre-built environments that are fully compatible with its API [33]. These environments cover a wide range of RL tasks, from classic control problems to more complex scenarios involving robotics, games, and simulations [34]. By standardizing the interface for these environments, Gymnasium ensures that researchers can seamlessly switch between different tasks without needing to modify their underlying RL algorithms [33].

One of the key advantages of Gymnasium is its ability to streamline the development process of RL algorithms. Researchers can leverage the pre-built environments to quickly test and refine their algorithms, focusing on improving performance and exploring new techniques. This modularity and ease of use significantly reduce the overhead associated with setting up and managing different RL tasks [33].

A Python package for the ASA framework, the AsaGym, was created to employ RL training for fighter agents. This package includes a subclass of the Gymnasium environment class to facilitate the training of new AI agents with RL. The library also offers unique wrappers to help in training setup configuration and methods for registering special environments to be recognized by the Gymnasium.

We utilize the Stable-Baselines3 Python module, a collection of RL algorithm implementations with a common interface, to test and verify the AsaGym library. These algorithms adhere to a standardized structure, using a unified interface that includes functions such as `train`, `save`, `load`, and `predict` [35]. Each algorithm has its limitations, especially regarding support for discrete or continuous action spaces. Additionally, the Stable-Baselines3 package provides the capability to multiprocess multiple separate contexts into a single environment, accelerating the training process for supported algorithms.

2) MODULES INTEGRATION

This work aims to use the ASA framework modules previously created in the background to construct a Python package that subclasses the Gymnasium main class. The modules utilized will be the Tacview recorder for rendering the environment, a specific AsaNode for processing the simulation steps and interacting with AsaGym, and the dynamically loaded models.

The set of BVR air combat behaviors of a fighter agent in the ASA framework is modeled through a Behavior Tree (BT). A simplified version of the fighter's BT is presented in Figure 1. BT is a formal language for the graphical and hierarchical representation of logical processes, i.e., behaviors [36]. Fighter's BT is composed of six main branchs: five subtrees that handle collision situations, threat reactions (break, crank, etc), situations to return to base (bingo fuel, damage, etc), combat air patrol, and navigation in formation; and a branch that handles three specific BVR phases: engagement; missile shooting; and missile supporting. The condition node of this branch checks whether the autonomous agent has a target, which is a requirement for the activation of one of the three mentioned phases. A detailed description of the engagement phase can be found in [37]. It is important to mention that subtrees encapsulate complex behaviors composed of a combination of many other nodes. Due to the large size of the fighter's behavior tree, subtrees were used to make this tree easier to visualize in the article.

Initially, the engagement node of the fighter's BT was modified to respond to commands from an external software component rather than execute a predefined function. The goal is to train a specific behavior of the flying agent to locate a suitable spot to take a shot in a BVR conflict, as discussed in the following section. All other behaviors will still be dictated by the BT nodes already implemented. An ASA extension, the External Processor, was created to properly handle Input/Output (IO) messages. It initiates a high-performance asynchronous communication socket called ZeroMQ (ZMQ) and uses it to convey messages serialized using the Protobuf

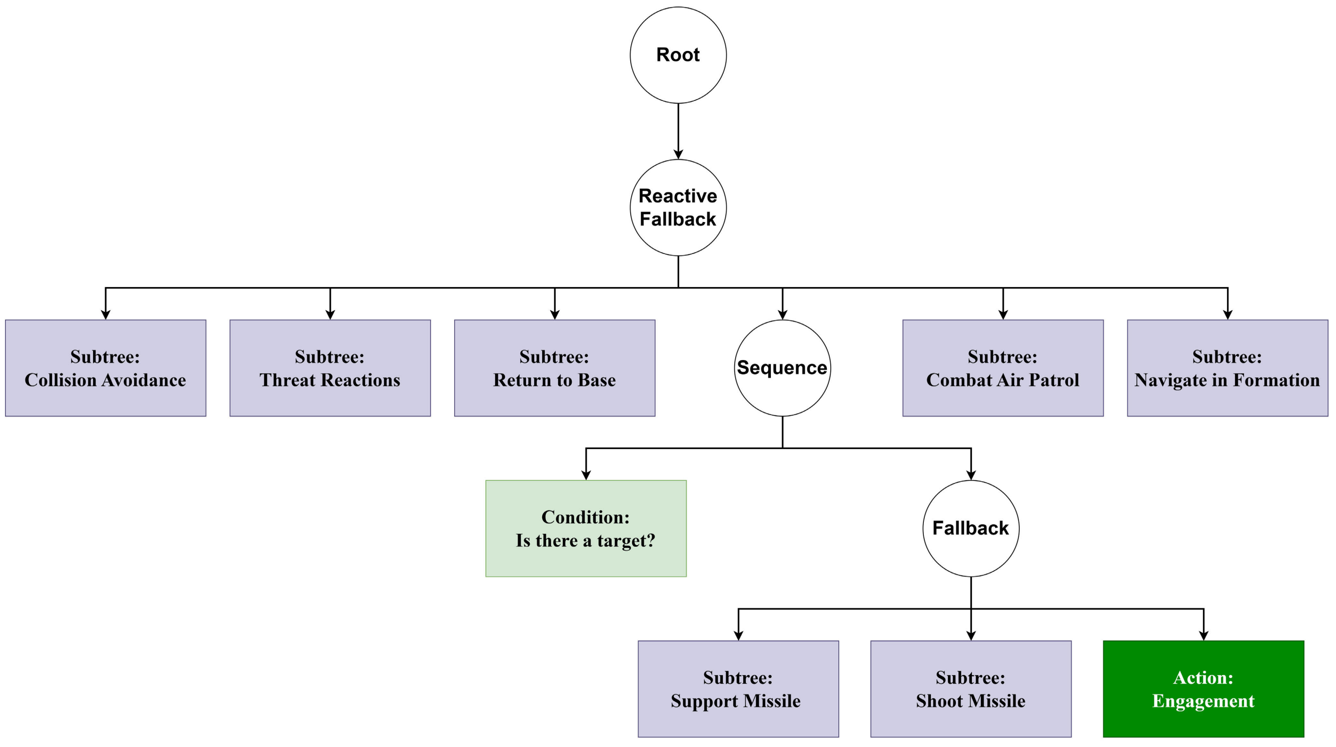

Root
Reactive Fallback
Subtree: Collision Avoidance
Subtree: Threat Reactions
Subtree: Return to Base
Sequence
Subtree: Combat Air Patrol
Subtree: Navigate in Formation
Condition: Is there a target?
Fallback
Subtree: Support Missile
Subtree: Shoot Missile
Action: Engagement

FIGURE 1. Simplified BT of the fighter agent.

library [38]. We customized the AsaNode with an RL server capable of answering many queries required by the Gymnasium Application Programming Interface (API). This server established a second ZMQ connection, which AsaGym will connect to for message exchange with our package. Additionally, the RL server starts, resets, and ends the node's execution.

3) COMMUNICATION AND SEQUENCE DIAGRAM

The communication between the AsaGym library and the related objects is depicted in a Unified Modeling Language (UML) sequence diagram for the `step` method in Figure 2. This diagram visually represents the interactions and message flow between the various components involved in the RL training process. In the diagram, closed arrowheads imply synchronous messages, indicating that the sender waits for a response before proceeding. In contrast, open arrowheads denote asynchronous messages, where the sender continues its process without waiting for a response.

Dashed lines in the diagram represent responses, illustrating the return flow of information from the receiver back to the sender. Filled lines indicate method calls, showing the initiation of actions or data transfers between components. The specific methods used to write and read at the ZMQ connection are `zmq_send` and `zmq_recv`, respectively. These methods facilitate high-performance communication by efficiently managing the transmission and reception of serialized messages within the network.

Using the same concept, message-exchanging mechanisms for setting up, clearing, and shutting down the environment were implemented. These mechanisms ensure seamless RL training session initialization, maintenance, and termination.

As presented in Figure 3, the engagement node of the BT communicates cyclically with a DRL model in AsaGym via the External Processor. The engagement node sends the agent's state to the DRL model, which processes this state to generate an observation. This observation serves as the input to the deep neural network of the DRL model, which then estimates an action. The estimated action is sent back to the engagement node, which executes it through the fighter agent. After performing the action, the agent reaches a new state that is sent back to the DRL model, where it is processed into a new observation.

If the new state benefits the agent, a positive reward is assigned. If the state is unfavorable, a negative reward is given. During training, the RL model adjusts the weights of its neural network based on these rewards, refining its policy to maximize future rewards. A detailed explanation of state, observation, action, and reward is given in this section.

B. SCENARIO DESCRIPTION

In this study, we consider simulations of three-dimensional BVR air combat scenarios. Each fighter agent is a computational model of the F-16 aircraft in the ASA framework, and is equipped with: datalink; six active BVR missiles with seekers; a radar; and a Radar Warning Receiver (RWR) [10].

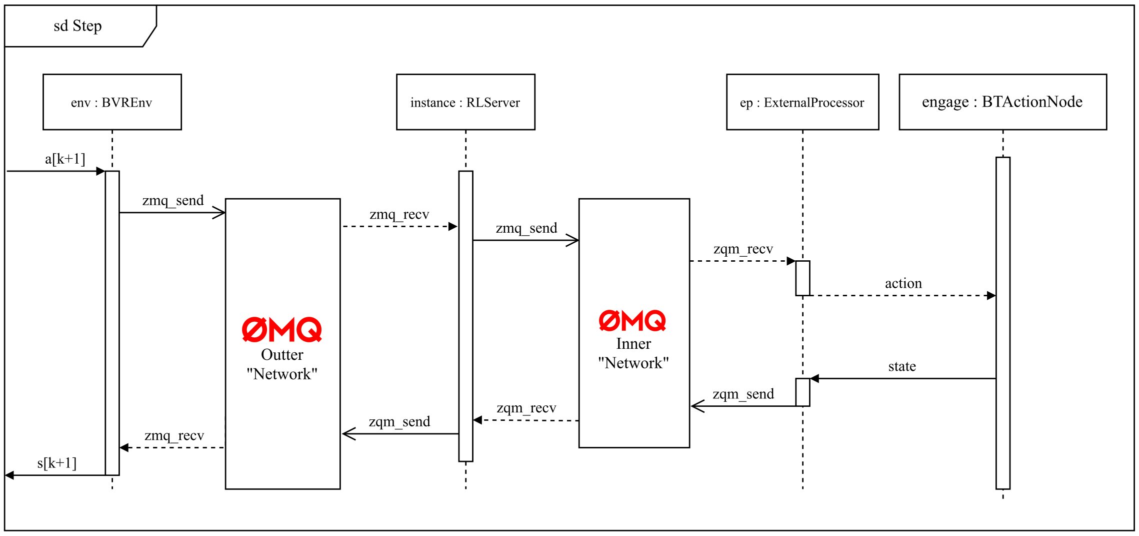


FIGURE 2. UML sequence diagram illustrating the messaging interactions in the *step* method of the AsaGym library using ZMQ for high-performance communication.

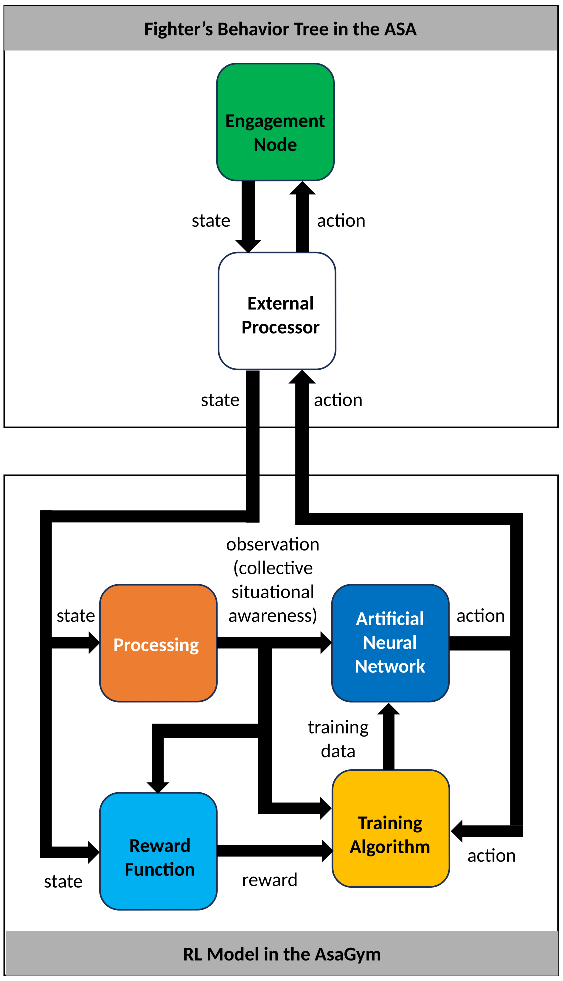


FIGURE 3. RL model in AsaGym integrated with the fighter's BT in ASA, showing the interactions between situational awareness, the reward function, and decision-making processes.

Each fighter agent has a BT that emulates BVR air combat behaviors. The training of the DRL model involves engaging a team of two fighters against an opposing team of two aircraft in BVR air combat. In this type of combat, pilots must position themselves strategically to gain an offensive advantage while avoiding enemy tracking [39].

This study explores a more complex multi-agent environment by introducing 2v2 BVR air combat. In this scenario, two allied aircraft engage two enemy aircraft, requiring coordination and teamwork to achieve tactical objectives.

Two distinct setups are considered, as illustrated in Figure 4. In **Setup 1**, the blue team consists of two agents: the Blue Leader, whose engagement node is controlled by a DRL model, and the Blue Wingman, whose engagement node follows engagement rules based on operational expert knowledge. They face an opposing red team composed of two enemies, both of which use engagement rules identical to those controlling the Blue Wingman. This setup challenges the Blue Leader to coordinate with its wingman to effectively engage a reactive opponent.

In **Setup 2**, both blue team agents, Blue Leader and Blue Wingman, have engagement nodes controlled by the same DRL model. Each agent has its own state, but receives a shared observation (collective situational awareness) that includes information about both allies and opponents. This setup allows the use of a single shared policy while providing each agent with a collective view of the scenario. It supports the learning of cooperative behavior and helps evaluate how a unified policy improves team coordination and performance against the red team. Each fighter agent of the red team follows the same engagement rules as in the first setup.

By expanding the scenario to include multi-agent interactions, this study enhances the robustness and versatility of the models, preparing them for combat situations. In both setups,

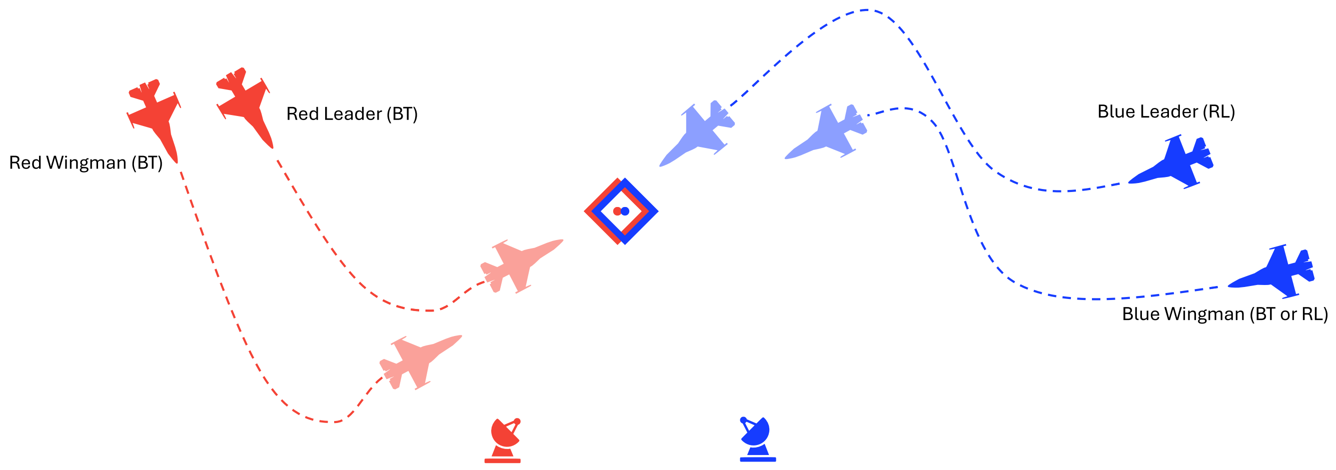


FIGURE 4. Diagram illustrating the positions and roles of the Blue and Red teams in the exercise. In Setup 1, the Blue Leader has an engagement node controlled by a DRL model, while the Blue Wingman has an engagement node controlled by engagement rules. In Setup 2, both the Blue Leader and Blue Wingman have engagement nodes controlled by a single DRL model. Each team is responsible for defending a common point of interest and is equipped with ground radar to enhance enemy detection.

TABLE 2. State and observation components for blue and red teams.

Category	Subcategory	Components
State	Blue Leader	Latitude, Longitude, Altitude, Heading, and Airspeed
	Blue Wingman	
	Red Leader	
	Red Wingman	
Observation	(Blue Leader , Red Leader)	Magnitude Difference of Relative Azimuths, Slant Distance, Airspeed Difference
	(Blue Leader , Red Wingman)	
	(Blue Wingman , Red Leader)	
	(Blue Wingman , Red Wingman)	
	(Blue Leader , Blue Wingman)	

each team is tasked with defending a point of interest and is equipped with ground radar to improve enemy detection.

C. LEARNING FRAMEWORK

This subsection describes the learning framework adopted in this study. It is organized into three parts: the observation space provided to the agents, the set of available actions, and the reward function designed to guide the learning process.

1) OBSERVATION

Observation is the input of the artificial neural network of the DRL model, as we can verify in Figure 3. In this work, observation is the collective situational awareness between the Blue and Red teams, as summarized in Table 2. This collective situational awareness is obtained through the processing of the estimated state of each agent.

The use of such collective situational awareness, which is based on angular relations, slant distances and airspeed differences, plays an important role in training the DRL model. It makes the DRL model invariant to the specifications of positions and velocities. To enhance the learning process, all observations are normalized within predefined ranges, as this is a common practice in RL training that helps improve training efficiency and convergence.

2) ACTION

The action is designed to capture essential flight control elements, allowing the agent to maneuver effectively during BVR engagements. It includes key flight parameters that influence combat performance, such as **heading** and **airspeed**. Unlike real-world pilots, who have full control over their aircraft, RL algorithms often benefit from a simplified action structure. In the ASA framework, the actions do not represent direct control commands for the aircraft but rather a desired attitude that the system aims to achieve.

The elements included in the action are:

- **Heading**: The direction in which the aircraft's nose is pointing, expressed in degrees, ranging from `−180` to `180`.
- **Airspeed**: The aircraft's speed along its flight path.

The action is structured as a dictionary, where each action parameter is represented as a constrained continuous value. The heading action allows for full directional control, while the airspeed action lets the agent adjust velocity to optimize engagement strategies. The decision to use a continuous action space improves maneuver precision compared to discrete alternatives. This setup enables the agent to explore a broader range of tactical options and adapt more effectively to different combat scenarios.

3) REWARD

Before defining the reward function used in this study, we conducted preliminary tests with alternative formulations inspired by existing literature. These initial experiments focused on individualistic reward strategies, such as encouraging agents to minimize their distance to opponents or maximize angular advantage individually. However, such strategies often resulted in undesired behaviors, including agents pursuing conflicting goals, clustering in suboptimal areas, or failing to coordinate effectively with teammates. These observations motivated the adoption of a shared reward formulation that promotes collective behaviors and situational awareness across the team.

The proposed reward function is designed to encourage agents to adopt collectively advantageous tactical behaviors in BVR air combat. The definitions of advantageous and disadvantageous situations in BVR scenarios were informed by the experience of fighter pilots.

To achieve these behaviors, we propose a reward function that incorporates collective situational awareness among agents on both the Blue and Red teams. Specifically, the reward is composed of two primary components: **Relative Azimuth Reward** and **Velocity Reward**. Each of these components is designed to promote different aspects of engagement strategy, as detailed below.

a: RELATIVE AZIMUTH REWARD

The idea behind this metric comes from a fundamental aspect of air combat: a pilot always tries to keep their aircraft pointed in a tactically favorable direction relative to the enemy. In practice, this means constantly adjusting position and heading to face the opponent, making it easier to react, pursue, or defend. This reward is important for achieving favorable positioning, as maintaining alignment with the target enhances maneuverability and increases the effectiveness of weapon deployment.

It evaluates the angular alignment between the agents of both teams by considering their relative azimuth values. The relative azimuth of an agent is calculated as the difference between the bearing to the target and its own heading. Since each agent has its own frame of reference, the relative azimuth from the perspective of one agent may differ from that of the opposing agent. The reward function is designed to minimize differences in orientation between allied and enemy agents, encouraging the Blue team to maintain an optimal orientation toward the Red team. The calculated values are normalized using the maximum allowed azimuth difference to ensure they fall within the range $[-1.0, 1.0]$. This reward is important for achieving favorable positioning, as maintaining alignment with the target enhances maneuverability and increases the effectiveness of weapon deployment. The relative azimuth reward is defined by:

$$reward_{ra} = \sum_{i=1}^{n_B} \sum_{j=1}^{n_R} \frac{\left|ra(R_j, B_i)\right| - \left|ra(B_i, R_j)\right|}{n_B n_R \, ra_{\max}} \tag{1}$$

where n_B and n_R are the number of agents of the Blue and Red teams, respectively; B_1 represents the Blue Leader; B_2 represents the Blue Wingman; R_1 represents the Red Leader; R_2 represents the Red Wingman; $ra(A, B) \in [-180^\circ, 180^\circ]$ represents the relative azimuth (in degrees) of agent A to agent B, which is computed as:

$$ra(A, B) = \text{bearing}(B \to A) - \text{heading}(A) \tag{2}$$

where $ra_{\max} = 180^\circ$ is the maximum value of the relative azimuth.

b: VELOCITY REWARD

This metric is inspired by how pilots control their speed in relation to an enemy during an engagement. In a real combat scenario, a pilot needs to approach the opponent at a speed that allows for effective action but also keeps a safe and manageable distance. If the agent stays too far, it risks losing the advantage; if it gets too close too quickly, it may compromise safety or tactical position. This component ensures that the agent does not remain too far from the target, which could compromise combat effectiveness.

Let B_i be an agent of the Blue team and R'_i the nearest agent of the Red team in relation to B_i. Let $v(B_i)$ and $v(R'_i)$ denote the airspeeds of B_i and R'_i, respectively, where airspeed is a scalar representing the magnitude of the agent's velocity. The velocity factor $F_v(B_i, R'_i)$ is computed based on the airspeeds of these agents, and $dv_{\max}$ is the maximum allowed velocity difference. The velocity reward encourages a single agent to close the distance with the nearest target while maintaining a tactically safe range. It is calculated based on the slant distance between the agent and the target, which is derived from their respective positions and altitudes. The reward is normalized to fall within the range [0.0, 1.0], with higher values awarded for shorter distances within an acceptable operational envelope. This component ensures that the agent does not remain too far from the target, which could compromise combat effectiveness. The velocity reward is defined by:

$$reward_v = \sum_{i=1}^{n_B} \frac{F_v(B_i, R'_i)}{n_B \, ra_{\max}} \left(|ra(R'_i, B_i)| - |ra(B_i, R'_i)| \right), \tag{3}$$

which applies only if:

$$\frac{|ra(R'_i, B_i)| - |ra(B_i, R'_i)|}{ra_{\max}} > 0.0, \tag{4}$$

otherwise:

$$reward_v = 0. \tag{5}$$

The velocity factor $F_v(B_i, R'_i)$ is given by:

$$F_v(B_i, R'_i) = \begin{cases} 0, & \text{if } \Delta v < 0 \\ 1, & \text{if } \Delta v > 1 \\ \Delta v, & \text{otherwise} \end{cases} \tag{6}$$

where:

$$\Delta v = 1 - \frac{dv_{\max} - (v(B_i) - v(R'_i))}{dv_{\max}}. \tag{7}$$

The total reward is computed as the sum of the normalized relative azimuth reward and velocity reward, ensuring a balanced evaluation of both positioning and engagement distance. The reward function dynamically adapts to the scenario by accounting for real-time changes in the relative positions and headings of the agents and targets.

D. INITIALIZATION SAMPLING

The scenario setup defines the initial conditions for both friendly and enemy agents, allowing for variability in positioning and airspeed. The initialization process is implemented through a custom function, which sets the starting parameters for the agents based on a fixed reference point. The Red Leader serves as the reference for positioning the Blue Leader. The initial latitude and longitude of the Red Leader are fixed, with a constant heading of 180° and airspeed of 450 knots.

The Blue Leader is always initialized directly in front of the Red Leader, ensuring that the two agents start head-on. The distance between them is randomly selected within a range of 20 to 60 nautical miles. The Blue Leader's airspeed is assigned with a random value within [250,650] knots.

To determine the initial positions, the ground distance, which is the horizontal separation between agents measured along the Earth's surface, and the bearing, which is the direction from one agent to another measured in degrees clockwise from true north, are used to compute the Blue Leader's latitude and longitude. The bearing is set to match the Red Leader's heading, ensuring that both agents are aligned head-on, while the Blue Leader's heading is fixed at 0°. Both agents start the engagement at an altitude of 25,000 feet, simulating a realistic operational environment for BVR air combat.

Once the initial state is set, the formation pattern for the blue team is configured. In this formation pattern, the Blue Wingman is positioned 180 degrees relative to the Blue Leader at a distance of approximately 1.6 nautical miles. The Blue Wingman is also assigned the same heading as the Blue Leader. The Red Wingman is positioned alongside the Red Leader, following a similar formation strategy. The complete initialization sampling process is summarized in Algorithm 1. A visualization of this initialization process is presented in Figure 5.

Algorithm 1 Initialization Sampling for 2v2 BVR Air Combat

```
1: Set fixed initial position, heading (180°), airspeed (450 knots), and altitude (25,000 feet) for Red Leader
2: Determine random distance from the Red Leader within [20, 60] NM to set the Blue Leader's position directly in front
3: Compute the Blue Leader's latitude and longitude using geodesic calculations with bearing equal to the Red Leader's heading
4: Set the Blue Leader's heading to a fixed value (0°)
5: Assign the Blue Leader's altitude as 25,000 feet
6: Assign the Blue Leader's airspeed with a random value within [250,650] knots.
7: Set the Blue Wingman 180 degrees relative to Blue Leader at a distance of 1.6 NM, maintaining the same heading
8: Set the Red Wingman alongside the Red Leader in a similar formation
9: Return the initial configuration with positions, headings, airspeeds, and formation details
```

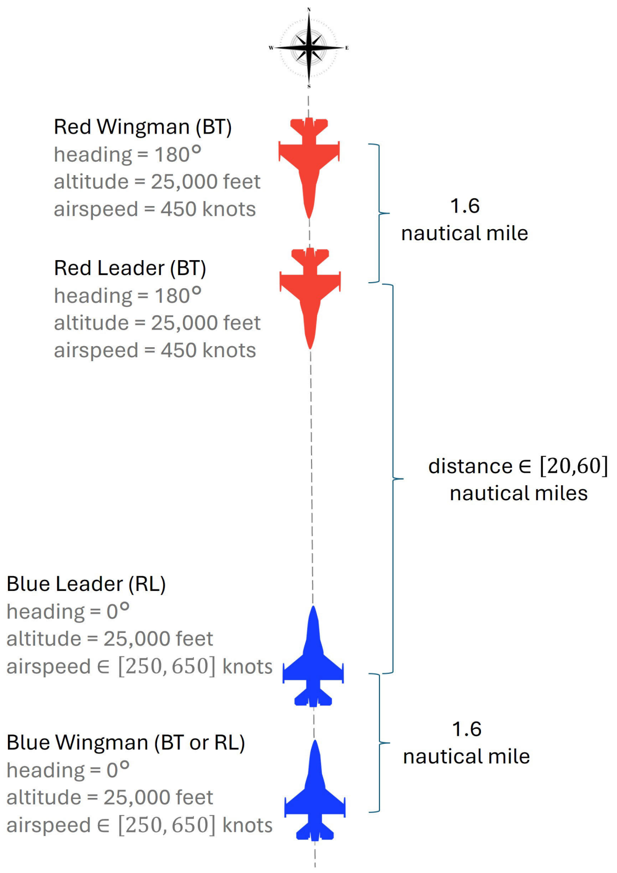


FIGURE 5. Visualization of the initialization sampling process.

E. EPISODE TERMINATION CRITERIA

The criteria for episode termination are important in training agents in the 2v2 BVR air combat environment, as they facilitate the training of specific aspects of combat, such as engagement scenarios, ensuring that engagements are tactically meaningful.

An episode ends when one of these conditions is met:

- **Aircraft Elimination**: The episode will conclude if at least one aircraft from either the blue or red team is eliminated. This condition ensures the integrity of the combat scenario, as the elimination of an aircraft would alter the dynamics of the engagement, effectively making it no longer a 2v2 scenario.
- **Priority-Based Termination Condition**: An episode will terminate if a BT node with a priority level greater than the priority of the engagement node is activated.

By defining structured termination criteria, the training environment ensures that each episode concludes with meaningful tactical outcomes. These criteria not only provide agents with well-defined objectives but also enable focused evaluation of specific behaviors and decision-making processes.

F. RENDERING METHODS

For validation and debugging purposes, visualizing the evolution of the agent's behavior is a key component of the development process. A practical approach to achieve this is through the use of the commercial software Tacview [40], which provides 3D visualizations of air combat scenarios. The ASA framework integrates seamlessly with Tacview by exposing all model attributes to a local port, enabling real-time rendering of the scenario. However, Tacview's streaming mode expects a continuous flow of temporal data, whereas AsaGym resets the environment's time at each episode. This episodic reset creates time discontinuities that Tacview cannot properly handle due to its serialized data collection, preventing its direct use for real-time visualization during training.

To address this limitation, each episode is saved as a separate `.acmi` file, Tacview's native format for recordings. This approach allows developers to review individual episodes offline, enabling a detailed post-training analysis of the agent's behavior and decision-making. By storing episodes in this manner, it becomes possible to thoroughly examine specific maneuvers and engagement strategies without interfering with the training process. The use of Tacview extends beyond basic debugging by providing a framework for performance evaluation. Its intuitive interface and robust visualization tools make it an invaluable resource for refining reinforcement learning models and ensuring that agents are prepared for realistic air combat scenarios.

G. WRAPPERS

The Gymnasium documentation [33] states that wrappers provide a simple and flexible way to modify existing environments without changing the core code. In AsaGym, wrappers keep the main environment intact while allowing customization to meet the specific needs of air combat training. This section explains the custom wrappers developed for the AsaGym library, expanding the features of the Gymnasium framework.

1) FLATTEN ACTION

The `FlattenAction` wrapper simplifies the multi-dimensional action space of the environment by converting it into a flat, continuous vector representation. This transformation helps the agent by reducing complexity while preserving essential control information. The `FlattenAction` wrapper leverages the `ActionWrapper` from the Gymnasium framework to create a simplified interface for action processing. This allows RL algorithms to output a vector of continuous values, which the wrapper then maps to the appropriate control inputs for the environment.

2) SKIP FRAME

A wrapper that maintains the same action for multiple frames has been implemented to train a more stable agent and emphasize the impact of each action. This technique was first introduced by DQN [41] to improve learning stability by reducing temporal aliasing and increasing the effective action duration. It is particularly effective in environments like AsaGym, where the ASA simulation processes commands every 100 milliseconds. By skipping frames, the wrapper ensures a stronger temporal influence on actions by accumulating all intermediate rewards and returning a cumulative reward at the end of the skipped frame block. The returned observation reflects the most recent state of the environment, ensuring accurate decision-making.

The `SkipFrameWrapper` takes a parameter defining the number of frames to be skipped. In this work, we are skipping 49 frames, which extends the interval between action decisions to approximately 5 seconds. This duration was chosen to prevent abrupt changes in actions at high frequencies, allowing sufficient time for the agent to execute a selected maneuver before making a new decision.

As previously described in Subsection III-C2, the actions in AsaGym do not represent direct control commands for the aircraft but rather a desired attitude that the system aims to achieve. Because of this, the low sampling rate introduced by the frame-skipping mechanism does not hinder control precision. Instead, it allows the agent to focus on meaningful tactical decisions rather than frequent low-level adjustments as the system smoothly transitions toward the commanded state.

H. TRAINING SETUP

For training the agents in the proposed scenarios, we employed four DRL algorithms: PPO [42], A2C [43], SAC [44], and TD3 [45]. These algorithms were implemented using the Stable-Baselines3 framework [35].

The selection of these algorithms was based on balancing *on-policy* and *off-policy* learning methods, as well as their ability to handle continuous action spaces. Table 3 summarizes the main characteristics of each algorithm.

The default hyperparameter values from Stable-Baselines3 were used, with adjustments to the step-related parameters and `batch_size`. Specifically, for PPO and A2C, the

TABLE 3. Comparison of selected RL algorithms.

Algorithm	Policy Type	Exploration Strategy	Key Characteristics
PPO	On-policy	Stochastic (policy noise)	Uses a clipped objective to prevent large policy updates, improving training stability. Efficient for large-scale training with parallel environments.
A2C	On-policy	Stochastic (policy noise)	Actor-critic method where the policy is guided by value estimates. More sample efficient than vanilla policy gradient but lacks a mechanism to limit updates, making it more sensitive to hyperparameters.
SAC	Off-policy	Stochastic (entropy maximization)	Encourages exploration through entropy regularization, improving stability in continuous action spaces. Uses experience replay and separate Q-value and policy networks.
TD3	Off-policy	Deterministic (target policy smoothing)	Addresses overestimation bias in DDPG with twin Q-value networks, delayed policy updates, and target smoothing. Improves stability and consistency in continuous action spaces, making it effective for real-world applications.

`n_steps` parameter was set to 128, while for SAC and TD3, the `train_freq` parameter was set to 128. These particular values were chosen through a random search process over traditional powers of 2 (e.g., 32, 64, 128, 256). This choice influences the trade-off between sample efficiency and update frequency. Higher values for step-related parameters allow the agent to accumulate more experience before updating, which helps stabilize policy learning by reducing gradient noise. However, it also delays policy updates, potentially slowing down adaptation to new situations. Smaller `batch_size` values lead to more frequent updates, while larger values help smooth the gradient updates at the cost of increased computational demand. Table 4 summarizes the selected hyperparameters and their roles.

Each algorithm was trained five times with different random seeds in both setups, Setup 1 (RL + BT) and Setup 2 (RL + RL), to ensure statistical robustness while maintaining computational feasibility. This choice follows common practice in RL, where a small number of seeds, typically five, is widely adopted to report average performance with acceptable variance [44], [46], [47].

Training was conducted for 500,000 steps per seed, using 16 parallel environments to accelerate learning. Model checkpoints were saved every 100,000 steps for evaluation and debugging, while performance metrics such as episode rewards, loss values, and training time were logged using TensorBoard. All experiments were run on a system with an AMD Ryzen 9 5900X processor (12 cores, 24 threads, 3.70 GHz) and 64 GB of RAM, enabling efficient parallel execution of training environments.

I. EVALUATION METRICS

To assess the performance of the trained agents in the air combat scenarios, four primary evaluation metrics were considered: average episode reward; mean episode length; and total training time for each algorithm.

- **Average Episode Reward:** This metric measures the cumulative reward achieved by the agent over an episode, averaged across multiple runs. The moving average is computed based on the number of episodes completed within each policy update. Since PPO is trained with $n\_steps = 128$ and $num\_envs = 16$, each update aggregates data from $128 \times 16 = 2048$ environment steps. The number of completed episodes within this window determines the averaging scope. A higher average reward indicates improved decision-making and maneuver execution.
- **Mean Episode Length:** The mean episode length represents the average number of steps the agent takes before an episode terminates. This metric provides insight into the agent's survivability and engagement effectiveness. Shorter episodes may indicate either successful engagements or premature termination due to poor tactical positioning, while longer episodes may suggest prolonged engagements with effective maneuvering.
- **Total Training Time:** This metric captures the wall-clock time to train the agents using each algorithm and is reported in hours. As all experiments were run on the same machine, differences in training time reflect algorithmic efficiency rather than hardware differences.

IV. RESULTS AND DISCUSSION

This section presents the results obtained from training the RL agents in the two proposed setups. The evaluation focuses on three key metrics: average episode reward, mean episode length, and total training time. For all reported results, both the mean and standard deviation were computed over the five independent training runs. In the plots for average episode reward and mean episode length, the solid line represents the mean value, while the shaded region indicates the standard deviation. Additionally, qualitative analyses of the agent's

TABLE 4. Selected hyperparameters for each RL algorithm. The remaining parameters follow the default values from Stable-Baselines3.

Algorithm	train_freq (Off-Policy)	n_steps (On-Policy)	batch_size	Explanation
PPO	–	128	128	Collects 128 steps per environment before updating the policy and optimizes in minibatches to improve training stability.
A2C	–	128	–	Collects 128 steps per environment before updating the policy in a single batch, using a synchronous update mechanism.
SAC	128	–	128	Updates every 128 steps using experience replay and entropy regularization to improve exploration.
TD3	128	–	128	Updates every 128 steps using a replay buffer and twin critics to reduce overestimation bias in Q-values.

behavior in both configurations are provided to illustrate the tactical maneuvers learned.

A. REWARD AND EPISODE LENGTH ANALYSIS

The following figures illustrate the learning progress of each algorithm over the training steps for both configurations proposed.

Figure 6 shows the evolution of the average episode reward in Setup 1. PPO demonstrates the highest reward convergence, followed by SAC, which stabilizes at a lower reward. TD3 exhibits instability, with a decline after an initial increase, which likely occurred because the agent made some random favorable choices early in training. A2C struggles to improve significantly, maintaining the lowest reward throughout training.

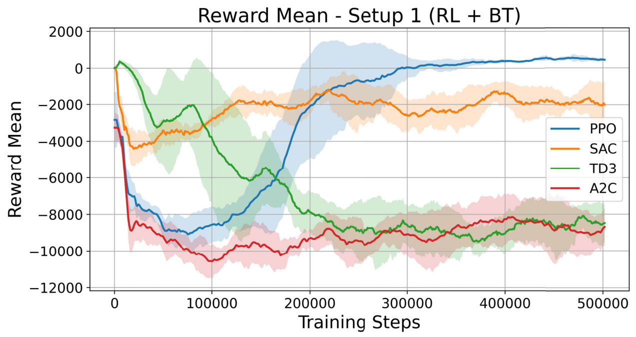


FIGURE 6. Average episode reward for PPO, SAC, TD3, and A2C in Setup 1 (RL + BT).

Figure 7 presents the reward evolution in Setup 2. Interestingly, PPO reaches its stability region even faster than in Setup 1, with a slightly higher average reward, demonstrating strong adaptability to the more complex environment. In contrast, the other algorithms show worse performance compared to Setup 1, likely due to the increased difficulty of controlling two agents simultaneously. SAC performs better than TD3 and A2C but remains below PPO in terms of average reward and takes longer to stabilize. TD3, which already showed instability in Setup 1, performs even

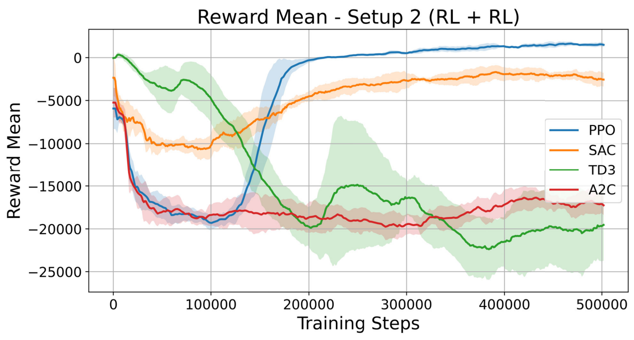


FIGURE 7. Average episode reward for PPO, SAC, TD3, and A2C in Setup 2 (RL + RL).

worse in Setup 2, exhibiting strong fluctuations and failing to improve. A2C remains the weakest, showing stable behavior but with consistently low performance and no meaningful progress.

Previous studies have shown that PPO and SAC frequently outperform other methods in continuous control tasks [42], [44]. Given this, it was expected that PPO and SAC would perform better than TD3 and A2C in this study.

Figure 8 presents the mean episode length over training for Setup 1. PPO and SAC stabilize with shorter episode

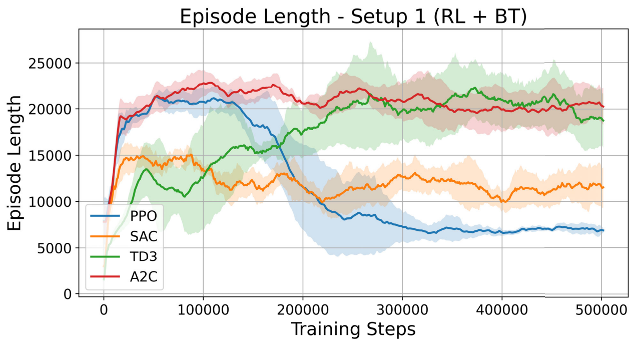


FIGURE 8. Mean episode length for PPO, SAC, TD3, and A2C in Setup 1 (RL + BT).

durations, indicating faster and more decisive engagements. TD3 shows a slower improvement compared to A2C, but its episode length gradually increases, reaching values slightly below A2C. A2C maintains high episode lengths throughout training, reinforcing its weaker performance.

Figure 9 illustrates the episode length trends for Setup 2. Overall, the episode lengths increased compared to Setup 1, which was expected given the higher complexity of this scenario where two agents must be controlled simultaneously. PPO still stabilizes at shorter episode lengths, but the gap between PPO and SAC has decreased, possibly because the more dynamic environment reduces PPO's relative advantage in quickly reaching efficient strategies, allowing SAC to perform closer to PPO.

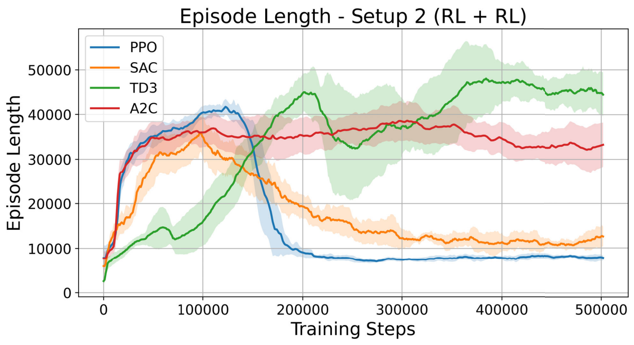


FIGURE 9. Mean episode length for PPO, SAC, TD3, and A2C in Setup 2 (RL + RL).

Notably, the difference between TD3 and A2C has increased, with TD3 now exhibiting longer episode durations than A2C. This shift may suggest that TD3 struggles more with policy convergence in the presence of another learning agent. On the other hand, A2C, despite its generally weak performance, maintains more consistent episode lengths, likely because its on-policy nature limits drastic fluctuations even if it fails to find optimal strategies.

B. TRAINING TIME COMPARISON

The total training time for each RL algorithm was measured in hours under both configurations. Table 5 summarizes the training duration required for PPO, SAC, TD3, and A2C algorithms. These values reflect the computational efficiency of each approach and provide a basis for assessing the trade-off between training speed and model performance.

In Setup 1 (RL + BT), the PPO algorithm required the longest training time, averaging 6.77 ± 0.57 hours, followed by SAC with 5.72 ± 0.26 hours. Despite its higher computational cost, PPO achieved the best performance, indicating that its longer training time is justified by superior results. SAC, while faster than PPO, did not reach the same performance level, suggesting that the slight reduction in training time comes with a trade-off in effectiveness. TD3 and A2C were the most time-efficient algorithms, with training times of 4.74 ± 0.24 and 4.10 ± 0.25 hours, respectively, but their performance was inferior to both PPO and SAC.

TABLE 5. Training time comparison for algorithms in different setups.

Setup	Algorithm	Training Time (hours)
Setup 1 (RL + BT)	PPO	6.77 ± 0.57
	SAC	5.72 ± 0.26
	TD3	4.74 ± 0.24
	A2C	4.10 ± 0.25
Setup 2 (RL + RL)	PPO	6.30 ± 0.16
	SAC	5.02 ± 0.31
	TD3	3.45 ± 0.10
	A2C	3.10 ± 0.14

In Setup 2 (RL + RL), training times decreased across all algorithms. PPO's training time was reduced to 6.30 ± 0.16 hours, maintaining its performance advantage. SAC also saw an improvement in efficiency, with a training time of 5.02 ± 0.31 hours, though it remained less effective than PPO. TD3 and A2C continued to demonstrate the shortest training times, 3.45 ± 0.10 and 3.10 ± 0.14 hours, respectively, reinforcing their computational efficiency.

Despite Setup 2 being more complex, as it required the control of two agents through RL, the training times were consistently lower compared to Setup 1. This result may be attributed to the fact that training two agents simultaneously allowed the algorithms to collect twice as much experience per update, accelerating convergence. This suggests that the increased complexity in agent coordination did not necessarily lead to higher computational demands, possibly due to the shared learning process between the agents.

C. AGENT BEHAVIOR ANALYSIS

To better understand the learned behaviors, qualitative assessments were conducted by visualizing the agents' actions in different combat scenarios. The analysis focused on the PPO algorithm, which demonstrated the best performance among the evaluated methods. This evaluation was supported by SMEs, consisting of BVR fighter pilots from FAB, who provided insights into tactical maneuvers, positioning strategies, and decision patterns typically used in real-world BVR air combat. These insights guided the design of the reward function and served as a reference for interpreting the agent's behavior.

Figure 10 presents a combined image made up of three subfigures: (a), (b), and (c), respectively, illustrating the initial situation and agent behaviors in two distinct setups: Setup 1 (RL + BT) and Setup 2 (RL + RL). In these visualizations, blue and red aircraft represent the Blue and Red teams, respectively. The cones extending forward from each aircraft represent radar coverage. These cones indicate the area within which each aircraft is able to detect and track enemy targets, reflecting the sensor's range and field of view used for building situational awareness.

The initial situation at the start of the episode provides a common context for both setups. In Setup 1 (RL + BT), the PPO-controlled agent coordinates effectively with a BT-controlled wingman. Setup 2 (RL + RL) features two PPO-controlled agents exhibiting adaptive behaviors.

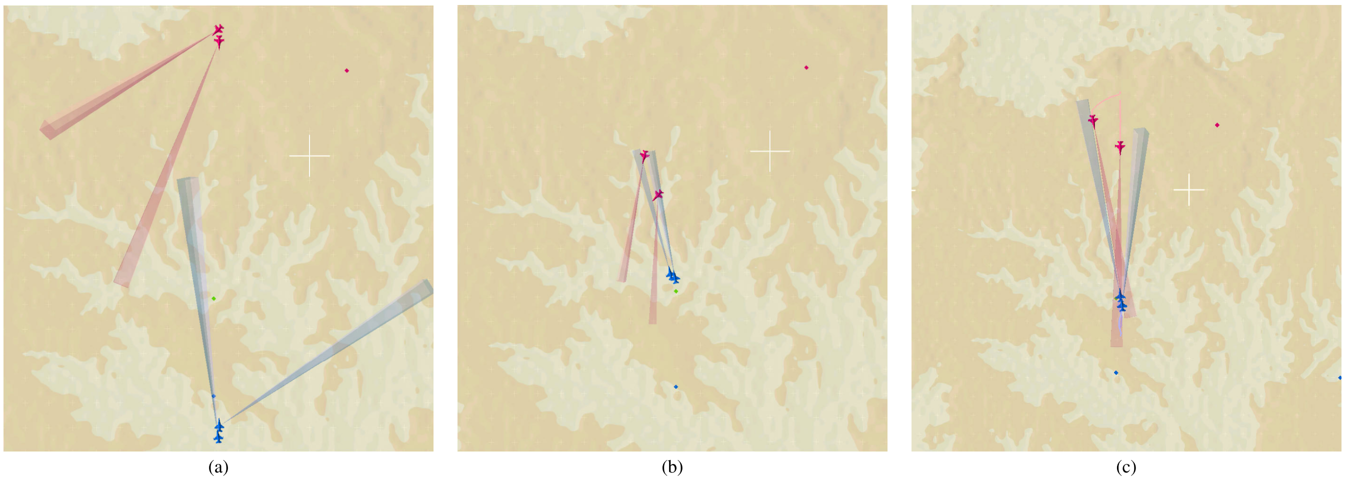


FIGURE 10. Agent behavior in: (a) Initial engagement scenario, (b) Final engagement for Setup 1 (RL + BT), and (c) Final engagement for Setup 2 (RL + RL).

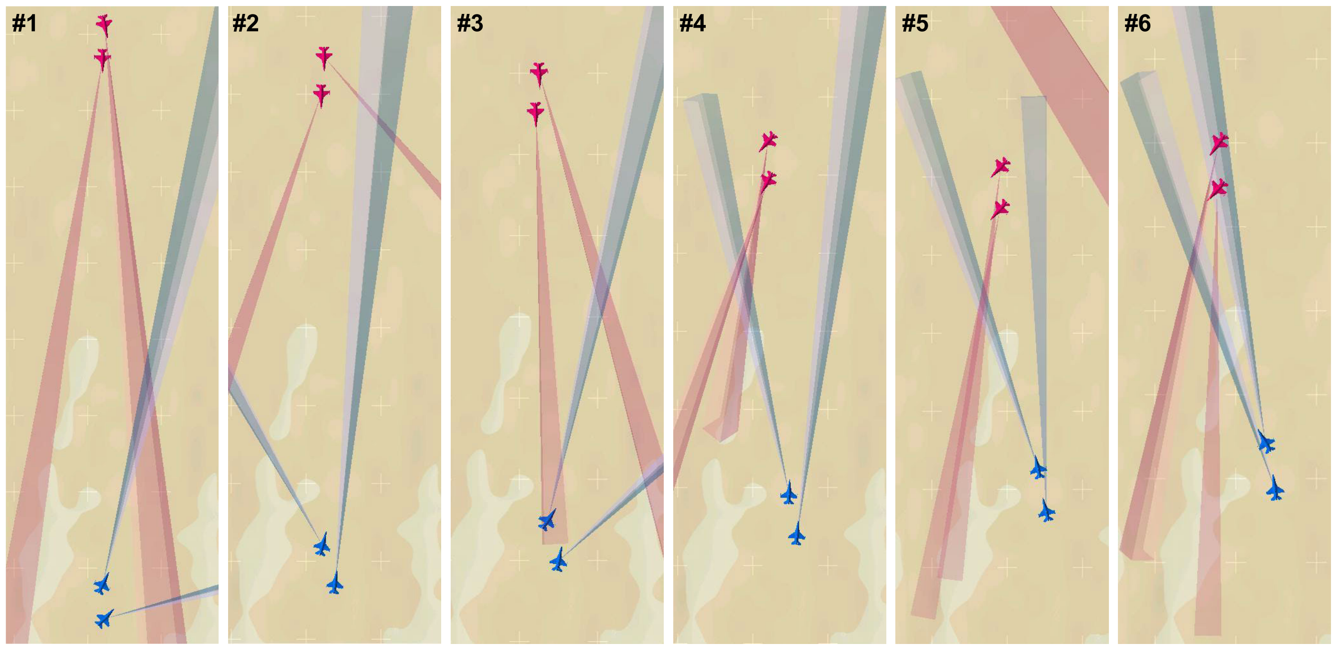


FIGURE 11. Sequence of six frames from a typical engagement in Setup 2 (RL + RL). The images (#1 to #6) illustrate the evolution of the scenario, showing the adaptive maneuvers and coordination between the PPO-controlled agents as they react to enemy actions and work together to achieve positional advantage.

Despite the different control schemes, both setups result in coordinated actions and successful target engagement, highlighting the effectiveness of the proposed reward-driven training. Agents in Setup 2 showed greater adaptability, dynamically adjusting their strategies in response to enemy maneuvers, while Setup 1 benefited from the predictable support provided by the BT-controlled wingman.

Additionally, to better illustrate the agent's behavior evolution during the engagement, Figure 11 presents a sequence of six frames sampled from a representative episode in Setup 2 (RL + RL). This sequence visually captures the agents' coordinated maneuvers and adaptive positioning as the engagement progresses. In these frames, it is possible to observe how the agents continually adjust their headings and positions in response to both enemy actions and the movements of their ally. The progression highlights moments of mutual support, attempts to outmaneuver opponents, and the use of positioning to maintain tactical advantage. By presenting these key snapshots, the figure offers a clear, step-by-step visualization of the decision-making processes and dynamic interactions that characterize multi-agent behavior in BVR air combat scenarios.

It is important to highlight that the use of an intermediate reward based on collective situational awareness allowed each agent to learn behaviors that led to situations of collective advantage. In other words, each agent made a combat decision that was favorable not only to the agent itself but also to the ally. These findings reflect common tactical principles in real 2v2 BVR air combat, such as mutual support, coordinated maneuvers, and adaptive targeting. According to feedback from SMEs, the ability of RL agents to create and maintain positional advantage while considering the ally's actions is similar to behaviors seen during real pilot training. This strengthens the case for using such agents in future decision support systems and human-autonomy teaming [48], as it shows that these agents can internalize and apply concepts of teamwork and collective awareness in operational contexts.

D. EVALUATION SUMMARY

The evaluation highlights the superior performance of PPO across both setups, achieving the highest average rewards and shortest episode lengths, though at the cost of longer training times. SAC demonstrated stable learning but with slower convergence and lower rewards compared to PPO. TD3 and A2C exhibited inconsistent performance, with TD3 struggling in complex scenarios and A2C showing limited learning progress.

The qualitative behavior analysis was conducted specifically on PPO, given its superior performance, with the support of SMEs. This analysis confirmed that, in both training setups, the agents were able to effectively engage enemy aircraft by employing coordinated tactics and demonstrating evidence of collective situational awareness throughout the engagements, including adaptive positioning, mutual coverage, and synchronized maneuvers. These observations were consistent across multiple episodes, further validating the reliability of the learned behaviors in dynamic combat scenarios.

V. CONCLUSION AND FUTURE WORK

This work addressed the use of DRL to model the engagement phase of BVR air combat, considering the collective situational awareness of both allies and opponents.

A key contribution was the development of the AsaGym computational library, which allows the development and training of DRL models for BVR air combat autonomous agents, i.e., fighter agents. Then, using AsaGym, the study analyzed the effectiveness of different DRL algorithms in a multi-agent air combat scenario and assessed their tactical capabilities based on operational knowledge from experts.

The results showed that PPO achieved the highest performance, demonstrating superior adaptability and decision-making, while SAC provided stable learning with slightly lower efficiency. TD3 and A2C faced more significant challenges in maintaining consistent performance. Training time analysis indicated that PPO required the most computational resources but delivered the best results, while TD3 and A2C trained faster but with limited effectiveness.

A qualitative assessment conducted by SMEs through visual analysis confirmed that agents with the engagement node controlled by a DRL model exhibited adaptive and cooperative tactics, such as synchronized offensive maneuvers and effective positioning strategies. In contrast, the setup, considering a leader with the engagement node controlled by a DRL model and a wingman with the engagement node controlled only by predetermined engagement rules, showed more rigid and predictable behaviors, reflecting the structured nature of BT-based decision-making.

Future work can address several limitations of this study. Introducing altitude variations would create more complex three-dimensional combat scenarios, enhancing the agents' maneuvering capabilities. Expanding to larger formations, such as n-versus-m, could evaluate the scalability of the approach and coordination among multiple agents. Decentralizing control by assigning independent policies to each agent, instead of a shared policy, may improve adaptability in dynamic situations. Additionally, relying solely on non-adaptive behavior tree-controlled opponents may cause overfitting; incorporating adaptive adversaries or leveraging self-play mechanisms would challenge the agents to develop more robust tactics. We also plan to explore adversarial reinforcement learning and self-play learning in larger formations such as 3v3 and 4v4. Involving human pilots in simulated missions could improve the agents' strategies by adding realistic feedback and enabling the study of real-time cooperation between humans and autonomous agents.

Furthermore, future research could explore replacing other behavior tree nodes beyond engagement, such as defensive and offensive maneuvers, allowing RL-based agents to handle a broader range of tactical decisions. Involving human pilots in simulated missions could improve the agents' strategies by adding realistic feedback and enabling the study of real-time cooperation between humans and autonomous agents.

Another possible direction for future research is the evaluation and comparison of different reward functions for multi-agent air combat. While this study focused on one specific formulation designed to encourage collective situational awareness, a deeper analysis of alternative reward functions and their effects on coordination strategies may be explored in future work.

The findings highlight the potential of DRL to improve air combat simulations by allowing autonomous agents to learn and perform complex collective tactical maneuvers. The research showed that it is possible to use RL to train intelligent fighter agents that can adapt to dynamic scenarios and coordinate effectively in multi-agent settings. This work has the potential to develop autonomous agents that can operate with human pilots, supporting better teamwork between humans and machines in tactical missions.

VI. SOURCE CODE

The source code used in this research is available in the AsaGym repository at https://github.com/ASA-Simulation/asa-gym. This repository provides an open-source version of the AsaGym environment, including its main modules, wrappers, and integration components, offering insights into the structure used for training and evaluating RL agents in air combat scenarios. The ASA framework itself, which runs the simulations, is not publicly available due to access restrictions. However, while researchers will not be able to run the environment, the released code can still serve as a reference for those working on RL applications in air combat.

REFERENCES

[1] Z. Yang, Z.-X. Sun, H.-Y. Piao, J.-C. Huang, D.-Y. Zhou, and Z. Ren, "Online hierarchical recognition method for target tactical intention in beyond-visual-range air combat," *Defence Technol.*, vol. 18, no. 8, pp. 1349–1361, Aug. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214914722000253

[2] L. C. P. Higby and U. Col, "Promise and reality: Beyond visual range (BVR) air-to-air combat," *Air War College (AWC) Electives Program: Air Power Theory, Doctrine, Strategy*, vol. 30, Jan. 2005.

[3] J. P. A. Dantas, M. R. O. A. Maximo, A. N. Costa, D. Geraldo, and T. Yoneyama, "Machine learning to improve situational awareness in beyond visual range air combat," *IEEE Latin Amer. Trans.*, vol. 20, no. 8, pp. 2039–2045, Aug. 2022. [Online]. Available: https://latamt.ieeer9.org/index.php/transactions/article/view/6530

[4] D. Hu, R. Yang, J. Zuo, Z. Zhang, J. Wu, and Y. Wang, "Application of deep reinforcement learning in maneuver planning of beyond-visual-range air combat," *IEEE Access*, vol. 9, pp. 32282–32297, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9358136

[5] J. P. A. Dantas, A. N. Costa, F. L. L. Medeiros, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Supervised machine learning for effective missile launch based on beyond visual range air combat simulations," in *Proc. Winter Simulation Conf. (WSC)*, Singapore, Dec. 2022, pp. 1990–2001.

[6] J. P. Dantas, A. N. Costa, D. Geraldo, M. R. Maximo, and T. Yoneyama, "PoKER: A probability of kill estimation rate model for air-to-air missiles using machine learning on stochastic targets," *J. Defense Model. Simul., Appl., Methodol., Technol.*, vol. 2025, Jan. 2025, Art. no. 15485129241309675, doi: 10.1177/15485129241309675.

[7] J. P. A. Dantas, A. N. Costa, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Weapon engagement zone maximum launch range estimation using a deep neural network," in *Intelligent Systems*, A. Britto and K. Valdivia Delgado, Eds., Cham, Switzerland: Springer, 2021, pp. 193–207.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] A. N. Costa, J. P. A. Dantas, E. Scukins, F. L. L. Medeiros, and P. Ögren, "Simulation and machine learning in beyond visual range air combat: A survey," *IEEE Access*, vol. 13, pp. 76755–76774, 2025.

[11] Y. Weilin, D. Wei, P. Shuangchun, X. Yu, and P. Liang, "Decision-making of one-on-one beyond-visual-range air combat based on improved Q-network," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Changchun, China, Aug. 2018, pp. 809–815. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8484466

[12] X. Qiu, Z. Yao, F. Tan, Z. Zhu, and J.-G. Lu, "One-to-one air-combat maneuver strategy based on improved TD3 algorithm," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2020, pp. 5719–5725.

[13] H. Piao, Z. Sun, G. Meng, H. Chen, B. Qu, K. Lang, Y. Sun, S. Yang, and X. Peng, "Beyond-visual-range air combat tactics auto-generation by reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[14] S. Soleyman, F. Hung, D. Khosla, Y. Chen, J. Fadaie, and N. Naderializadeh, "Multi-agent autonomous battle management using deep neuroevolution," *Proc. SPIE*, vol. 11758, Apr. 2021, Art. no. 117580C.

[15] T. Zhang, Y. Wang, M. Sun, and Z. Chen, "Air combat maneuver decision based on deep reinforcement learning with auxiliary reward," *Neural Comput. Appl.*, vol. 36, no. 21, pp. 13341–13356, Apr. 2024, doi: 10.1007/s00521-024-09720-z.

[16] L. Wang, J. Hu, Z. Xu, and C. Zhao, "Autonomous maneuver strategy of swarm air combat based on DDPG," *Auto. Intell. Syst.*, vol. 1, no. 1, p. 15, Dec. 2021, doi: 10.1007/s43684-021-00013-z.

[17] J. Hu, L. Wang, T. Hu, C. Guo, and Y. Wang, "Autonomous maneuver decision making of dual-UAV cooperative air combat based on deep reinforcement learning," *Electronics*, vol. 11, no. 3, p. 467, Feb. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/3/467

[18] L. Wang and H. Wei, "Research on autonomous decision-making of UCAV based on deep reinforcement learning," in *Proc. 3rd Inf. Commun. Technol. Conf. (ICTC)*, May 2022, pp. 122–126.

[19] B. Li, J. Huang, S. Bai, Z. Gan, S. Liang, N. Evgeny, and S. Yao, "Autonomous air combat decision-making of UAV based on parallel self-play reinforcement learning," *CAAI Trans. Intell. Technol.*, vol. 8, no. 1, pp. 64–81, Jun. 2022. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/cit2.12109

[20] Y. Han, H. Piao, Y. Hou, Y. Sun, Z. Sun, D. Zhou, S. Yang, X. Peng, and S. Fan, "Deep relationship graph reinforcement learning for multi-aircraft air combat," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Padua, Italy, Jul. 2022, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9892208

[21] X. Liu, Y. Yin, Y. Su, and R. Ming, "A multi-UCAV cooperative decision-making method based on an MAPPO algorithm for beyond-visual-range air combat," *Aerospace*, vol. 9, no. 10, p. 563, Sep. 2022. [Online]. Available: https://www.mdpi.com/2226-4310/9/10/563

[22] H. Zhang, H. Zhou, Y. Wei, and C. Huang, "Autonomous maneuver decision-making method based on reinforcement learning and Monte Carlo tree search," *Frontiers Neurorobot.*, vol. 16, Oct. 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2022.996412

[23] Z. Fan, Y. Xu, Y. Kang, and D. Luo, "Air combat maneuver decision method based on A3C deep reinforcement learning," *Machines*, vol. 10, no. 11, pp. 1–18, Nov. 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/11/1033

[24] H. Zhang, Y. Wei, H. Zhou, and C. Huang, "Maneuver decision-making for autonomous air combat based on FRE-PPO," *Appl. Sci.*, vol. 12, no. 20, p. 10230, Oct. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/20/10230

[25] Y. Jiang, J. Yu, and Q. Li, "A novel decision-making algorithm for beyond visual range air combat based on deep reinforcement learning," in *Proc. 37th Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Kunming, China, Nov. 2022, pp. 516–521. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10023870

[26] B. Li, S. Bai, S. Liang, R. Ma, E. Neretin, and J. Huang, "Manoeuvre decision-making of unmanned aerial vehicles in air combat based on an expert actor-based soft actor critic algorithm," *CAAI Trans. Intell. Technol.*, vol. 8, no. 4, pp. 1608–1619, Mar. 2023, doi: 10.1049/cit2.12195.

[27] E. Scukins, M. Klein, and P. Ögren, "Enhancing situation awareness in beyond visual range air combat with reinforcement learning-based decision support," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Warsaw, Poland, Jun. 2023, pp. 56–62. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10156497

[28] Y. Zhou, F. Yang, C. Zhang, S. Li, and Y. Wang, "Cooperative decision-making algorithm with efficient convergence for UCAV formation in beyond-visual-range air combat based on multi-agent reinforcement learning," *Chin. J. Aeronaut.*, vol. 37, no. 8, pp. 311–328, Aug. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936124001389

[29] C. Qian, X. Zhang, L. Li, M. Zhao, and Y. Fang, "H3E: Learning air combat with a three-level hierarchical framework embedding expert knowledge," *Expert Syst. Appl.*, vol. 245, Jul. 2024, Art. no. 123084. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423035868

[30] L. Zhao, C. Xing, and X. Yang, "Research on beyond-visual-range air combat decision making based on improved self-play deep reinforcement learning," in *Proc. 7th Int. Conf. Mach. Learn. Natural Lang. Process. (MLNLP)*, Oct. 2024, pp. 1–7.

[31] J. P. A. Dantas, A. N. Costa, V. C. F. Gomes, A. R. Kuroswiski, F. L. L. Medeiros, and D. Geraldo, "ASA: A simulation environment for evaluating military operational scenarios," 2022, *arXiv:2207.12084*.

[32] J. P. A. Dantas, D. Geraldo, A. N. Costa, M. R. O. A. Maximo, and T. Yoneyama, "ASA-SimaaS: Advancing digital transformation through simulation services in the Brazilian air force," in *Proc. Simpósio de Aplicações Operacionais em Áreas de Defesa*, Sao Jose dos Campos, Brazil, Sep. 2023, p. 6. [Online]. Available: https://www.sige.ita.br/edicoes-anteriores/2023/st/235455_1.pdf

[33] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. D. Cola, T. Deleu, M. Goulao, A. Kallinteris, M. Krimmel, R. Perez-Vicente, A. Pierre, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, Mar. 2023, "Gymnasium," [Online]. Available: https://zenodo.org/record/8127025

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*.

[35] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. (2019). *Stable Baselines3*. Accessed: Apr. 15, 2023. [Online]. Available: https://github.com/DLR-RM/stable-baselines3

[36] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Boca Raton, FL, USA: CRC Press, 2020.

[37] J. P. A. Dantas, A. N. Costa, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Engagement decision support for beyond visual range air combat," in *Proc. Latin Amer. Robot. Symp. (LARS), Brazilian Symp. Robot. (SBR), Workshop Robot. Educ. (WRE)*, Natal, Brazil, Oct. 2021, pp. 96–101.

[38] Google. (2023). *Protocol Buffers Documentation*. [Online]. Available: https://protobuf.dev

[39] J. P. A. Dantas, M. R. O. A. Maximo, and T. Yoneyama, "Autonomous agent for beyond visual range air combat: A deep reinforcement learning approach," in *Proc. ACM SIGSIM Conf. Princ. Adv. Discrete Simulation*, New York, NY, USA, Jun. 2023, pp. 48–49, doi: 10.1145/3573900.3593631.

[40] R. Software. (2024). *Tacview*. [Online]. Available: https://www.tacview.net

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[43] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, Jan. 2016, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html

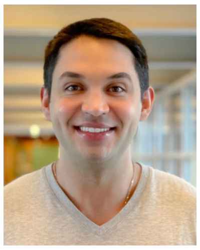
[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jan. 2018, pp. 1861–1870.

[45] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jan. 2018, pp. 1587–1596.

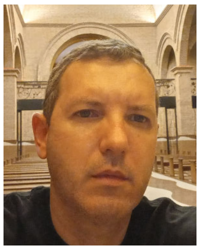
[46] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Jan. 2017, pp. 3207–3214.

[47] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. J. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, Nov. 2019.

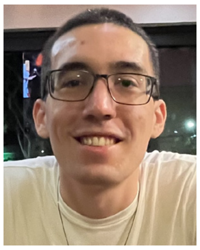
[48] J. P. A. Dantas, M. R. O. A. Maximo, and T. Yoneyama, "Loyal wingman assessment: Social navigation for human-autonomous collaboration in simulated air combat," in *Proc. 38th ACM SIGSIM Conf. Princ. Adv. Discrete Simulation*, New York, NY, USA, Jun. 2024, pp. 61–62, doi: 10.1145/3615979.3662149.



JOAO P. A. DANTAS received the B.Sc. degree in mechanical-aeronautical engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2015, and the M.Sc. degree from the Graduate Program in Electronic and Computer Engineering, ITA, in 2019, where he is currently pursuing the Ph.D. degree. In 2015, he participated in a year-long exchange program at Stony Brook University, USA. In 2022, he was a Visiting Researcher with the AirLab, Robotics Institute, Carnegie Mellon University. He is a Researcher with the Institute for Advanced Studies, Brazilian Air Force.

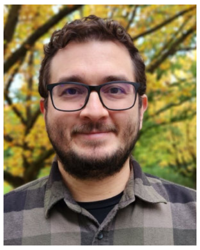


FELIPE L. L. MEDEIROS received the B.Sc. degree in computer science from the Federal University of Ouro Preto, Ouro Preto, Brazil, in 1999, and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research, São José dos Campos, Brazil, in 2002 and 2012, respectively. He has been working in the field of artificial intelligence, with an emphasis on metaheuristics, autonomous agents, and simulation.



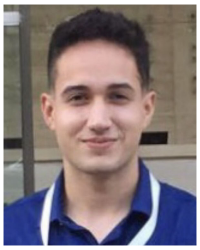
ADRISSON R. SAMERSLA received the B.Sc. degree in computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2021, and the M.Sc. degree from the Graduate Program in Electronic and Computer Engineering, ITA, in 2022. He is currently a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. His research interests include high-performance computing, machine learning, robotics, and simulation.

PEDRO L. R. BOTELHO received the B.Sc. degree in electronic engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2023. He works as an Electronic Engineer at the Brazilian Air Force and collaborates with the Institute for Advanced Studies. His research interests include reinforcement learning, robotics, multimodal learning, and simulation.



VITOR C. F. GOMES received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research (INPE), Brazil, in 2009, 2012, and 2023, respectively. Currently, he is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. His research interests include distributed computing, geospatial big data, and data science.

SAMARA R. SILVA received the B.Sc. degree in aeronautical science from Brazilian Air Force Academy (AFA), in 2013, and the B.Sc. degree in computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2022, where she is currently pursuing the M.Sc. degree. She is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. Her research interests include data science, artificial intelligence, machine learning, and simulation.



YURI D. FERREIRA received the B.Sc. degree in aerospace engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2021. Currently, he is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. His research interest includes reinforcement learning and its applications.

ALESSANDRO O. ARANTES received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research (INPE), Brazil, in 2000, 2008, and 2016, respectively. Currently, he is a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. His research interests include applied computing, software testing, and data science.

MARCIA R. C. AQUINO received the B.Sc. degree in computer science from the Federal University of Juiz de Fora, Brazil, in 1997, and the M.Sc. degree in applied computing from the National Institute for Space Research, São Josédos Campos, Brazil, in 2005. She is currently a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. Her research interests include data science, artificial intelligence, machine learning, simulation, and project management.

MARCOS R. O. A. MAXIMO received the B.Sc. degree (Hons. and summa cum laude) in computer engineering and the M.Sc. and Ph.D. degrees in electronic and computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2012, 2015, and 2017, respectively. He is currently a Professor with ITA, where he is also a member of the Autonomous Computational Systems Laboratory (LAB-SCA) and leads the Robotics Competition Team: ITAndroids. He is especially interested in humanoid robotics. His research interests include mobile robotics, dynamical systems control, and artificial intelligence.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614