Machine Learning with Applications 21 (2025) 100714

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

# A Configurable Intrinsic Curiosity Module for a Testbed for Developing Intelligent Swarm UAVs

Jawad Mahmood *,[1], Muhammad Adil Raja[1], John Loane[1], Fergal McCaffery[1]

*Dundalk Institute of Technology, Co. Louth, Dundalk, A91K584, Ireland*

ARTICLE INFO

*Keywords:*
UAVs
RL
ICM
FlightGear
A3C

ABSTRACT

This paper introduces an Intrinsic Curiosity Module (ICM) based Reinforcement Learning (RL) framework for swarm Unmanned Aerial Vehicles (UAVs) target tracking, leveraging the actor–critic architecture to control the roll, pitch, yaw, and throttle motions of UAVs. A key challenge in RL-based UAV coordination is the delayed reward problem, which hinders effective learning in dynamic environments. Existing UAV testbeds rely primarily on extrinsic rewards and lack mechanisms for adaptive exploration and efficient UAV coordination. To address these limitations, we propose a testbed that integrates an ICM with the Asynchronous Advantage Actor-Critic (A3C) algorithm for tracking UAVs. It incorporates the Self-Reflective Curiosity-Weighted (SRCW) hyperparameter tuning mechanism for the ICM, which adaptively modifies hyperparameters based on the ongoing RL agent's performance. In this testbed, the target UAV is guided by the Advantage Actor-Critic (A2C) model, while a swarm of two tracking UAVs is controlled by using the A3C-ICM approach. The proposed framework facilitates real-time autonomous coordination among UAVs within a simulated environment. This system is developed using the FlightGear flight simulator and the JSBSim Flight Dynamics Model (FDM), which enables dynamic simulations and continuous interaction between UAVs. Experimental results demonstrate that the tracking UAVs can effectively coordinate and maintain precise paths even under complex conditions.

## 1. Introduction

This paper introduces a testbed that is specifically designed for real-time Unmanned Aerial Vehicle (UAV) coordination and control. Two distinct Reinforcement Learning (RL) algorithms are employed to control the target and tracking aircraft, integrated with the JSBSim Flight Dynamics Model (FDM). The target UAV is governed by the Advantage Actor-Critic (A2C) model, while the swarm of tracking UAVs is controlled through the Asynchronous Advantage Actor-Critic (A3C)-Intrinsic Curiosity Module (ICM) model. The use of the A3C model is particularly suited for handling multiple tracking agents due to its asynchronous nature. The target aircraft follows a dynamically generated trajectory by an Neural Network (NN) model trained using the A2C algorithm. Meanwhile, the swarm of tracking UAVs is trained to follow this trajectory in real time. The application of the ICM is particularly beneficial for swarm UAV coordination, where agents must operate autonomously and react to environmental changes in real time. By incorporating the ICM into the A3C architecture, our tracking UAVs improved exploration efficiency and learning consistency. Moreover, the ICM-enhanced approach facilitates robust policy learning, improving both tracking accuracy and system stability. The testbed employed the Self-Reflective Curiosity-Weighted (SRCW) hyperparameter tuning mechanism for the ICM, which dynamically adjusts learning parameters based on agent performance, optimizing the trade-off between exploration and exploitation. This framework significantly enhances the real-time learning capabilities of swarm UAVs in uncertain and complex states, making it a powerful advancement in autonomous aerial systems research.

**The contributions of this research study are:**

1. Demonstrates robust swarm UAV behavior for dynamic target tracking, with synchronized flight and adaptability to complex and evasive trajectories in real-time simulations. The testbed is scalable, allowing multiple UAVs to operate and learn simultaneously.
2. Addresses the delayed reward problem in RL by incorporating an ICM along with SRCW hyperparameter tuning for the ICM, which promotes exploration and more efficient policy learning.

* Corresponding author.
*E-mail addresses:* Jawad.Mahmood@dkit.ie (J. Mahmood), Adil.Raja@dkit.ie (M.A. Raja), John.Loane@dkit.ie (J. Loane), Fergal.McCaffery@dkit.ie (F. McCaffery).

[1] All authors have contributed significantly to the work and agree to its submission.

https://doi.org/10.1016/j.mlwa.2025.100714
Received 21 April 2025; Received in revised form 11 July 2025; Accepted 22 July 2025
Available online 4 August 2025

## 2. Literature review

UAV testbeds serve as vital platforms for simulating, developing, and validating the behavior of autonomous aerial systems under controlled and repeatable conditions. They integrate physical simulation engines, control logic, and communication protocols to replicate real-world missions such as navigation, cooperative tracking, and swarm coordination. Such testbeds are critical for evaluating advanced learning techniques such as RL, enabling the safe and efficient development of scalable autonomous UAV systems.

Yasar, Bridges, Mallapragada, and Horn (2006) developed a simulation-based testbed aimed at facilitating coordination between unmanned aerial and ground vehicles using a Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) framework. The testbed integrates a detailed rotorcraft model of the UH-60 A Black Hawk helicopter alongside ground-based Segway robots that function as Unmanned Ground Vehicless (UGVs). By incorporating discrete-event supervisory control and mathematical modeling techniques, the system supports autonomous functions such as surveillance, exploration, and target tracking. Their experiments demonstrate that distributed control algorithms can effectively support cooperative missions involving both UAVs and UGVs, validated through simulation and hardware-in-the-loop testing.

Michael, Mellinger, Lindsey, and Kumar (2010) proposed a micro-UAV testbed for surveillance and reconnaissance missions. These micro UAVs are small-scale aerial vehicles suited for low-altitude operations. A testbed incorporates a software framework based on a finite-state machine that models a hybrid system architecture. This design facilitates the execution of experiments involving single or multiple quadrotors, with each operational mode incorporating closed-loop control mechanisms. The platform enables the simulation and assessment of various control strategies in a virtual environment before deployment on actual hardware. A high-fidelity dynamic model of quadrotors was also developed, which is capable of executing vertical takeoffs and landings. Nonetheless, a major limitation of the system is its applicability to quadrotor configurations only.

Zhang, Geng, and Fei (2012) proposed a flight simulation testbed that integrates an autopilot system, the FlightGear open-source simulator, and a ground control station. FlightGear plays a central role in this setup, supporting the creation of new aircraft models and terrains while adhering to control standards and offering realistic visualization. The proposed system integrates a ground station, autopilot, and FlightGear, enabling real-time simulation of UAV behavior under different flight conditions. The control laws are tested through User Datagram Protocol (UDP) communication between the autopilot and the simulator. The system successfully demonstrates UAV simulation, reducing the cost and risks of real-world flight tests, while verifying the control logic and enhancing research efficiency. Nevertheless, a major drawback of the proposed system is its limited scalability and the absence of distributed processing capabilities.

Moness, Mostafa, Abdel-Fadeel, Aly, and Al-Shamandy (2012) presented a virtual lab-based testbed that integrates MATLAB with the FlightGear simulator, useful for control engineering. The system utilizes a Cessna 310 aircraft model to demonstrate core and advanced concepts through three structured experiments. The first experiment focuses on simulating aircraft dynamics to apply classical control techniques. The second one explores the behavior of the dynamic UAV for longitudinal and lateral modes through both time and frequency domain analysis. The third experiment involves designing Proportional-Integral-Derivative (PID) controllers for various autopilot functionalities, including pitch, altitude, airspeed, bank angle, and heading regulation. This virtual lab makes use of FlightGear for high-fidelity visualization, JSBSim for accurate flight dynamics simulation, and MATLAB for developing and analyzing control strategies. The overall setup offers a highly interactive and immersive learning platform for researchers.

Sonu and Doshi (2012) presented the Georgia testbed for autonomous control (GaTAC), an open-source and scalable testbed specifically developed to evaluate multi-agent decision-making strategies in realistic environments involving autonomous aerial vehicles. The GaTAC framework is both cost-effective and flexible, integrating seamlessly with the FlightGear simulator to support the testing of various cooperative and competitive policy models. It accommodates both autonomous and manually operated UAVs, facilitating experimentation in complex and uncertain operational contexts. The authors elaborate on the architecture, modular components, and key features of GaTAC, showcasing its use in reconnaissance-based UAV missions. The study underscores the value of GaTAC in enabling the development of scalable, practical solutions for autonomous aerial decision-making systems.

Aschauer, Schirrer, and Kozek (2015) introduced a testbed that combines MATLAB with the FlightGear simulator to facilitate aircraft control and system identification. In this configuration, FlightGear is responsible for visualizing and simulating the flight environment, while MATLAB processes sensor data and calculates actuator commands. Communication between both platforms is established using the UDP, which ensures seamless real-time exchange of flight states and control signals. The flight dynamics are modeled using gray-box techniques, and various controller responses are evaluated using standard input signals like step and ramp functions. This testbed offers flexibility for cross-platform deployment on local networks and can be extended to support other complex environments. The proposed setup is useful for academic research in aerospace control systems.

Zhang, Zhou, and Xu (2015) developed a Hardware-In-the-Loop Simulation (HILS) testbed aimed at enhancing the design, simulation, and validation of UAV flight control systems using the dSPACE platform. This testbed uses MATLAB's Simulink to model UAV dynamics, onboard sensors, and environmental influences, while leveraging dSPACE's real-time simulation tools for rapid and efficient system development. A nonlinear six-degree-of-freedom UAV model was constructed based on standard motion equations, and real-time C code was auto-generated using dSPACE's real-time interface. Flight dynamics data and control commands were exchanged via a UDP module, enabling real-time visualization in FlightGear. Additionally, the testbed integrates the C Interface Library to support customized tools for tracking aircraft paths and monitoring flight status.

Habib, Malik, Rahman, and Raja (2017) introduced a novel Namal Unmanned Aerial Vehicle (NUAV) testbed for the design and simulation of UAVs, providing researchers with the capability to model various operational scenarios. This testbed interfaces with the FlightGear simulator through a dedicated communication module, facilitating the real-time exchange of flight dynamics data. The system features a data reception module that acquires key flight parameters such as velocity, orientation, and spatial coordinates from FlightGear, and responds by transmitting the corresponding control surface commands. The framework incorporates the Non-Dominated Sorting Genetic Algorithm (NSGA)-II algorithm to perform multi-objective optimization and utilizes an Artificial Neural Network (ANN) to construct the control architecture.

Ahmed, Quinones-Grueiro, and Biswas (2022) developed a high-fidelity simulation testbed for fault-tolerant octo-rotor UAV control using RL. This system models the Tarot T-18 UAV using a Gazebo flight simulator, and a Python back-end program was used to evaluate flight control performance under atmospheric disturbances such as wind. The testbed supports simulation of brushless direct current motor failures, both abrupt and gradual, and enables the injection of external force vectors for disturbance modeling. The testbed supports the simulations of multiple UAV geometries, including quadrotor, hexarotor, and octarotor configurations. Experimental trials show stable trajectory tracking in the presence of faults and disturbances, along with analysis of control allocation errors and system performance. The modular Python-Gazebo interface enables fast prototyping and facilitates transfer learning across different UAV platforms.

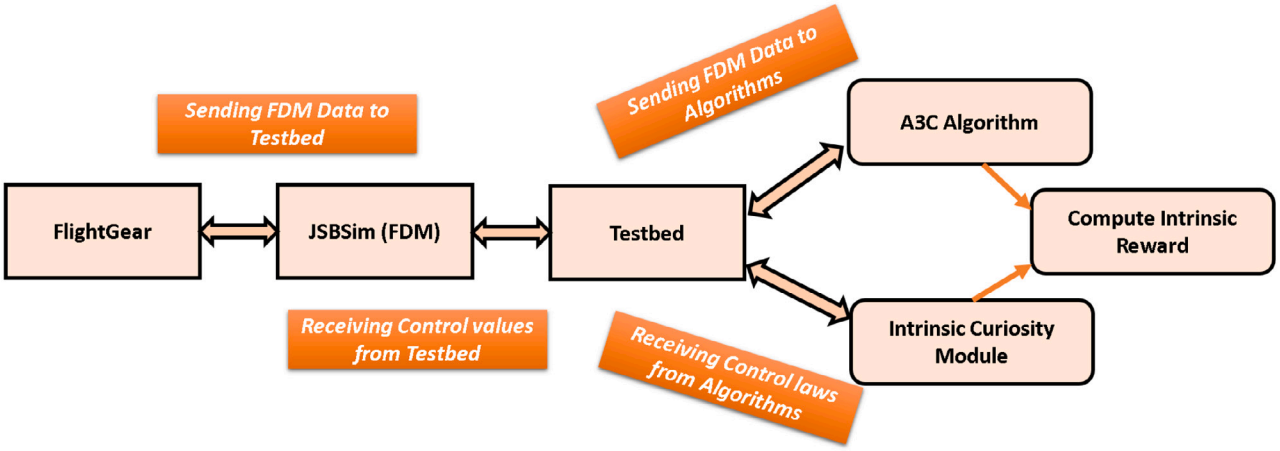


**Fig. 1.** Framework for the implementation of ICM for the testbed.

## 3. The development of the UAV testbed

To enable real-time simulations, the proposed RL algorithms were integrated with the FlightGear flight simulation environment. A major challenge during the development of the testbed was ensuring smooth and reliable communication between FlightGear, the JSBSim FDM, and the RL algorithms. Achieving this integration was critical to maintain synchronized control loops and realistic flight behavior throughout the simulation process. FlightGear served as the primary simulation platform, offering real-time graphical rendering, while JSBSim functioned as the FDM, responsible for simulating realistic flight dynamics. The architecture of the communication module was influenced by design principles from the NUAV and GaTAC testbeds. This module transmits control commands such as elevator, rudder, and aileron adjustments to FlightGear. Concurrently, FlightGear relays updated flight information, including altitude, latitude, and longitude, back to the system via the same communication interface. To ensure efficient and low-latency data transmission, the system utilized the UDP, facilitating bidirectional communication in real-time. This continuous data exchange enables the UAV to dynamically update its control strategies based on real-time environmental feedback, thereby ensuring accurate and adaptive simulation behavior. Fig. 1 illustrates the structural overview of the framework of the testbed, incorporating the A3C-ICM architecture to compute intrinsic reward.

The use of the ICM in UAVs encouraged effective exploration in a complex environment. It helped the agents learn meaningful behaviors even when external rewards were sparse or delayed. It improved the robustness of the policies by guiding UAVs to discover new strategies and explore the environment. Furthermore, it enhanced adaptability in dynamic conditions, making UAVs more autonomous and efficient in real-world missions.

### 3.1. Interfacing FDM with RL algorithms

FlightGear functioned as a simulation environment and utilized a FDM to replicate the flight behavior of the aircraft. Specifically, JSBSim delivered a highly detailed physics-based modeling approach capable of accurately reproducing various aspects of flight, including aerodynamics, propulsion systems, and control surfaces. A communication module was implemented to facilitate data exchange between FlightGear and the RL algorithms.

**Dispatching control instructions:** The RL model generated control outputs such as throttle, rudder, elevator, and aileron commands based on the agent's learned policy and forwarded them to FlightGear through JSBSim.

**Receiving real-time flight data:** In response, FlightGear returned the current state data of the aircraft, such as its coordinates, velocity, and altitude, to the communication module. This process occurs in a continuous feedback loop, allowing the RL models to dynamically adjust their control strategies based on the current flight conditions of the UAVs.

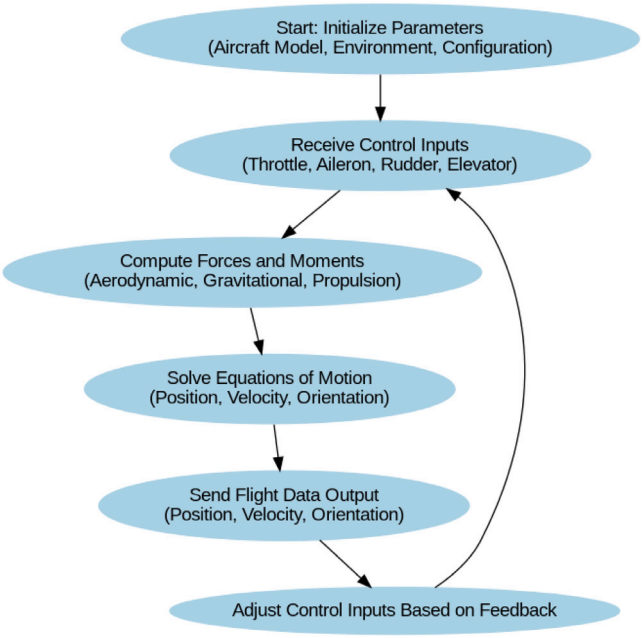


**Fig. 2.** Flowchart illustrating the working mechanism of JSBSim.

As illustrated in Fig. 2, the control commands are relayed from FlightGear to JSBSim, which then processes and simulates the behavior of the aircraft. In return, JSBSim provides comprehensive real-time updates that include spatial location (latitude, longitude, altitude), flight speed and direction, orientation angles (roll, pitch, yaw), engine parameters, and control surface deflections.

### 3.2. Implementation of the intrinsic curiosity module

Curiosity Driven Learning (CDL) is an approach that incentivizes agents to explore their environment by providing internal rewards for encountering novel or unexpected states. Unlike conventional RL methods that depend solely on externally provided rewards (extrinsic rewards), this method introduced intrinsic motivation, allowing the agent to seek and learn from unfamiliar experiences. This mechanism is

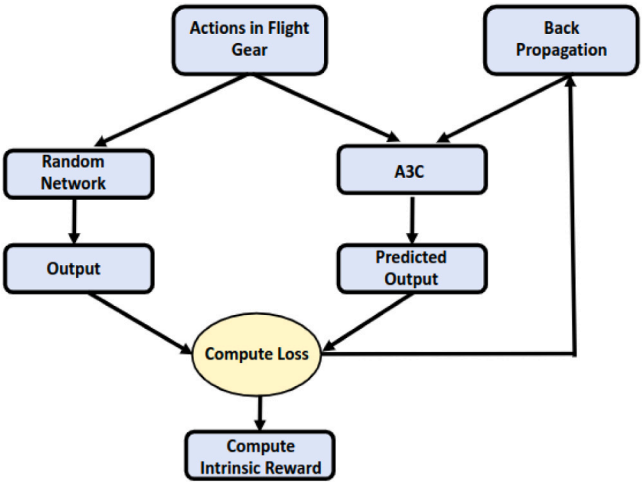


**Fig. 3.** Implementation of ICM.

particularly useful in environments where external rewards are sparse or delayed, as it promotes consistent learning progress. Fig. 3 illustrates the implementation of the ICM, which supports curiosity-driven learning by generating intrinsic motivation based on prediction discrepancies. In UAV target tracking, the ICM helped agents explore effectively when the target is temporarily occluded, enabling re-engagement through learned behaviors. It improved learning in sparse reward settings by providing internal motivation signals. The ICM also supports adaptability to dynamic target paths and helps avoid getting stuck in suboptimal trajectories. This leads to more robust, accurate, and responsive tracking performance in a complex environment.

**Prediction:** The forward dynamics model estimated the next state $\hat{s}_{t+1}$ based on the current state $s_t$ and the action $a_t$, as shown below (Bougie & Ichise, 2020; Colas, Fournier, Chetouani, Sigaud, & Oudeyer, 2019):

$$\hat{s}_{t+1} = f_{\text{forward}}(s_t, a_t) \tag{1}$$

**Intrinsic curiosity:** The intrinsic reward is defined as the prediction error between the next estimated state and the actual next state (Lin, Lai, Chen, Cao, & Wang, 2022; Zhelo, Zhang, Tai, Liu, & Burgard, 2018):

$$r_{\text{intrinsic}} = \|\hat{s}_{t+1} - s_{t+1}\|^2 \tag{2}$$

**Extrinsic Reward:** In addition to intrinsic curiosity signals, agents also receive extrinsic rewards directly from the environment. These rewards are typically task-specific and represent performance feedback provided by the simulation or the environment itself (Li et al., 2019; Wu, Yu, Liao, & Ou, 2024).

$$r_{\text{extrinsic}} = \text{environment reward} \tag{3}$$

**Total Reward:** The overall reward used to train the agent is a combination of intrinsic and extrinsic components, thereby enhancing both exploration and task-specific performance (Li & Gajane, 2023):

$$r_t = r_{\text{extrinsic}} + r_{\text{intrinsic}} \tag{4}$$

We have added the intrinsic reward generated by ICM to the extrinsic reward.

**Final Update:** After integrating the ICM into the A3C architecture, the training process involved computing gradients for both actor and critic networks. These gradients are used to update the respective networks asynchronously. The gradient of the actor network is defined according to Eq. (5) (Zheng et al., 2021):

$$\nabla_\theta L_{\text{actor}} = \frac{\partial L_{\text{actor}}}{\partial \theta} \tag{5}$$

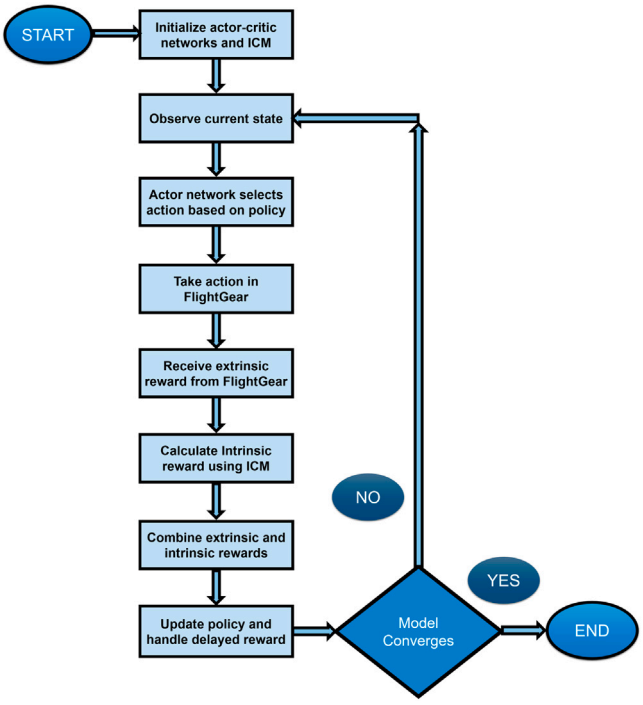


**Fig. 4.** Framework to overcome the problem of delayed reward using A3C-ICM model.

Similarly, the gradient for the critic network is computed according to Eq. (6):

$$\nabla_{\theta_v} L_{\text{critic}} = \frac{\partial L_{\text{critic}}}{\partial \theta_v} \tag{6}$$

In RL, a delayed reward means that the agent, such as a UAV, did not receive immediate feedback after taking an action. Instead, it only gets a reward much later, maybe after completing a task or reaching a certain point. For example, if a UAV is tracking a moving target, it might not receive any reward until it gets close to the target after many steps. During this time, the UAV has no clear idea whether the actions it is taking are good or bad. This makes learning slow and confusing because the UAV cannot immediately connect its earlier movements with the final result. For UAVs, this is a serious problem. They operate in a swarm in continuous environments where the goal (like staying close to a moving target at a constant distance) is critical. Without quick feedback, the UAV may continue flying by following poor strategies, wasting time and fuel, and could cause collisions, and fail to improve its behavior.

To solve this problem, we used the A3C-ICM model, which adds a special type of reward called the intrinsic reward. The UAV does not have to wait for a delayed external reward to learn something useful. The curiosity reward fills in the gaps, keeping the UAV motivated and helping it learn good tracking behaviors much earlier and learn to track moving targets more efficiently, even in complex or changing environments. Fig. 4 shows the framework for overcoming the delayed reward problem.

SRCW hyperparameter tuning is a mechanism to adaptively tune the hyperparameters of the ICM during training, optimizing curiosity-driven exploration. SRCW hyperparameter tuning is an adaptive approach in which the tracking UAV, governed by the A3C-ICM framework, dynamically adjusts its learning parameters based on its performance during training. While pursuing the target, the UAV evaluates its flight dynamics and progress in tracking performance. Through this self-evaluation, the UAV modifies its hyperparameters accordingly. These adjustments include adjusting the intrinsic curiosity, modifying learning rates, and improving decision-making (Wang, Liu et al., 2024).

**Pseudocode: SRCW-Tuning**

```
Pseudocode SRCW-Tuning(agent)
Input:
    Initial Hyperparameters:
        icm_beta in [0.1, 0.6]
        icm_learning_rate in [1e-4, 5e-4]
        entropy_coefficient in [0.005, 0.02]
        gradient_clipping_norm in [20, 40]
        rollout_length (Tmax) in [5, 20]
        inverse_loss_weight in [0.6, 0.9]
        forward_loss_weight in [0.1, 0.4]
        feature_dim in [64, 256]
        curiosity_learning_rate in [3e-5, 5e
            -4]
        reward_normalization in {True, False}
Begin:
    Initialize all hyperparameters randomly
        within specified ranges
    For each training episode do:
        Evaluate:
             Reward = reward(t) - reward(t -
                 1)
             Entropy level of agent's policy
             Gradient norms and training loss
                  trends
             Curiosity reward vs extrinsic
                 reward ratio

        If Reward < threshold and entropy too
             high:
            icm_beta = max(0.1, icm_beta -
                0.05)
            entropy_coefficient =
            max(0.005, entropy_coefficient -
                0.002)

        If Reward < threshold and entropy too
             low:
            icm_beta = min(0.6, icm_beta +
                0.05)
            entropy_coefficient =
            min(0.02, entropy_coefficient +
                0.002)

        If training unstable:
            curiosity_learning_rate =
            curiosity_learning_rate * 0.9
            gradient_clipping_norm =
            min(40, gradient_clipping_norm +
                2)

        If reward improving steadily:
            curiosity_learning_rate =
            max(3e-5, curiosity_learning_rate
                 * 0.95)
            entropy_coefficient =
            entropy_coefficient * 0.95
        Optionally:
            Adjust rollout_length or
                feature_dim
            Maintain num_workers constant or
                increase for stability
        If reward_normalization == True:
            Normalize rewards
        Update agent with new hyperparameters
End
```

Table 1 presents the hyperparameters associated with the ICM, their ranges, and brief descriptions. The hyperparameter ranges in Table 1 are partially derived from previous research papers and partially configured through experimental tuning (Stadie et al., 2020; Sun et al., 2025; Wang, Li et al., 2024; Wang, Liu et al., 2024).

## 4. Experimental validation and discussion of results

This section demonstrates the implementation of these RL models, discusses computation time and training stability, and discusses the effectiveness of the swarm UAVs in tracking a moving target along complex paths.

### 4.1. Implementation of the A2C model

A2C is a synchronous policy-gradient RL algorithm that combines value-based and policy-based approaches. It is designed to optimize decision-making in environments where agents select actions to maximize long-term rewards. A2C consists of an actor and a critic network.

The actor learns a policy $\pi(a|s)$ that maps states to actions, and the critic estimates the value function $V(s)$ to evaluate how good a given state is. The algorithm uses the advantage function to measure how much better an action is compared to the average, defined as:

$$A(s,a) = Q(s,a) - V(s) \tag{7}$$

The actor is updated using this advantage signal to improve the policy, while the critic is updated by minimizing the value estimation error.

The network receives as input the state representation of the environment. For UAVs, this includes position, velocity, and orientation. The hidden layers extract features that are useful both for control and value estimation. At each time step, the agent observes its current state within the environment, including its position, speed, altitude, and orientation. The actor network takes this state as input and produces a probability distribution over possible actions. An action is selected from this distribution and applied to the environment (e.g., change heading or throttle). The environment then returns a reward and the next state based on the action taken. The reward indicates how effective the action was, and the critic network estimates the value of the current state, how much reward the UAV can expect to receive in the future, starting from this state. The algorithm computes the "advantage", which is the difference between the observed reward and the value predicted by the critic. The actor is updated to increase the probability of actions with positive advantage and decrease the probability of bad ones. The critic is updated by minimizing the error between the predicted and actual returns (value loss). Fig. 5 shows the implementation of the A2C algorithm.

### 4.2. Implementation of the A3C algorithm

The A3C algorithm integrates the actor–critic structure with an asynchronous training mechanism to enable efficient RL. The A3C algorithm comprises two primary components: the actor and the critic. The actor is responsible for defining the policy by selecting actions based on the current state of the agent. This policy produces a probability distribution over possible actions. One of the defining qualities of A3C lies in its asynchronous training approach, where multiple workers interact with separate instances of the environment simultaneously. Each worker interacts with its respective part of the environment and periodically calculates gradients from its local experience. These gradients are then applied asynchronously to update the global network. In TensorFlow, this is implemented using $GradientTape$ to compute gradients, and $optimizer.apply.gradients$ is used to apply updates to the shared global network.

**Step-by-Step workflow of asynchronous updates in A3C**: The following steps outline the complete workflow using two workers as an example (Babaeizadeh, Frosio, Tyree, Clemons, & Kautz, 2016; Nahhas, Kharitonov, & Turowski, 2022; Zhou, Wang, Hu, & Deng, 2021):

**Table 1**
SRCW Hyperparameters for ICM (Stadie, Zhang, & Ba, 2020; Sun et al., 2025; Wang, Li et al., 2024; Wang, Liu et al., 2024).

| Hyperparameter | Range | Description |
|---|---|---|
| `icm_beta` | 0.1 to 0.6 | Balances intrinsic (curiosity) and extrinsic rewards |
| `icm_learning_rate` | $1 \times 10^{-4}$ to $5 \times 10^{-4}$ | Learning rate for the ICM's internal forward and inverse models |
| `entropy_coefficient` | 0.005 to 0.02 | Controls randomness in policy to promote exploration |
| `gradient_clipping_norm` | 20 to 40 | Upper limit for gradient norms to stabilize training |
| `rollout_length (Tmax)` | 5 to 20 | Number of steps before the worker sends updates to the global model |
| `inverse_loss_weight` | 0.6 to 0.9 | Weight assigned to inverse model loss in the ICM |
| `forward_loss_weight` | 0.1 to 0.4 | Weight assigned to forward model prediction loss in the ICM |
| `feature_dim` | 64 to 256 | Size of the learned feature representation used in the ICM |
| `curiosity_learning_rate` | $3 \times 10^{-5}$ to $5 \times 10^{-4}$ | Learning rate specifically for curiosity-based updates |
| `reward_normalization` | True/False | Whether to normalize total rewards for stable training |

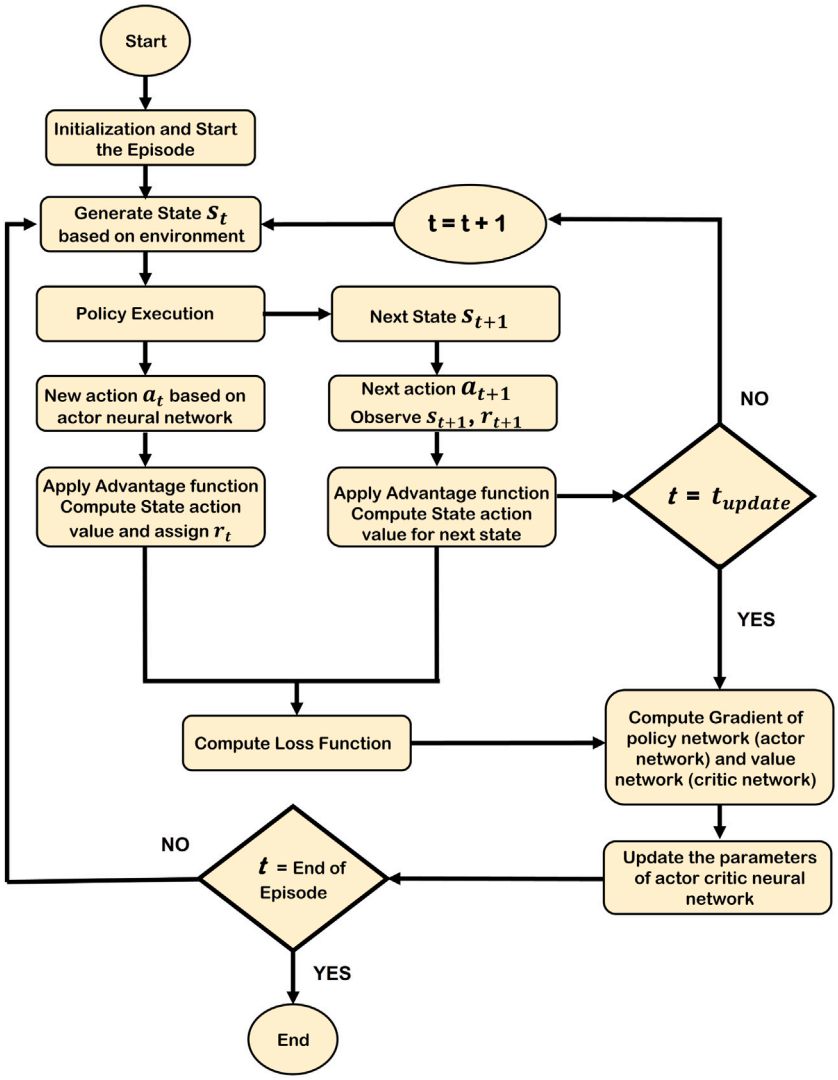


**Fig. 5.** Implementation of A2C algorithm.

1. **Initialization :** A global neural network is created with shared parameters for the actor (policy) and critic (value function). Each worker creates its own local copy to interact with the environment.
2. **Local roll-outs:** Both workers independently observe the current state, choose actions based on their local policy, receive rewards, and store the experiences (state, action, reward, next state) in a local buffer.
3. **Independent experience collection:** Workers proceed for a fixed number of steps or until a terminal state is reached. This process is done entirely independently by each worker, with no synchronization required during rollout.
4. **Gradient computation:** Each worker computes the policy gradient and value loss using its local experience. The advantage function is estimated as:

$$A(s_t, a_t) = R_t - V(s_t) \tag{8}$$

where $R_t$ is the cumulative reward and $V(s_t)$ is the estimated value.
5. **Global update:** Each worker sends its computed gradients to the global network. The global network immediately applies these

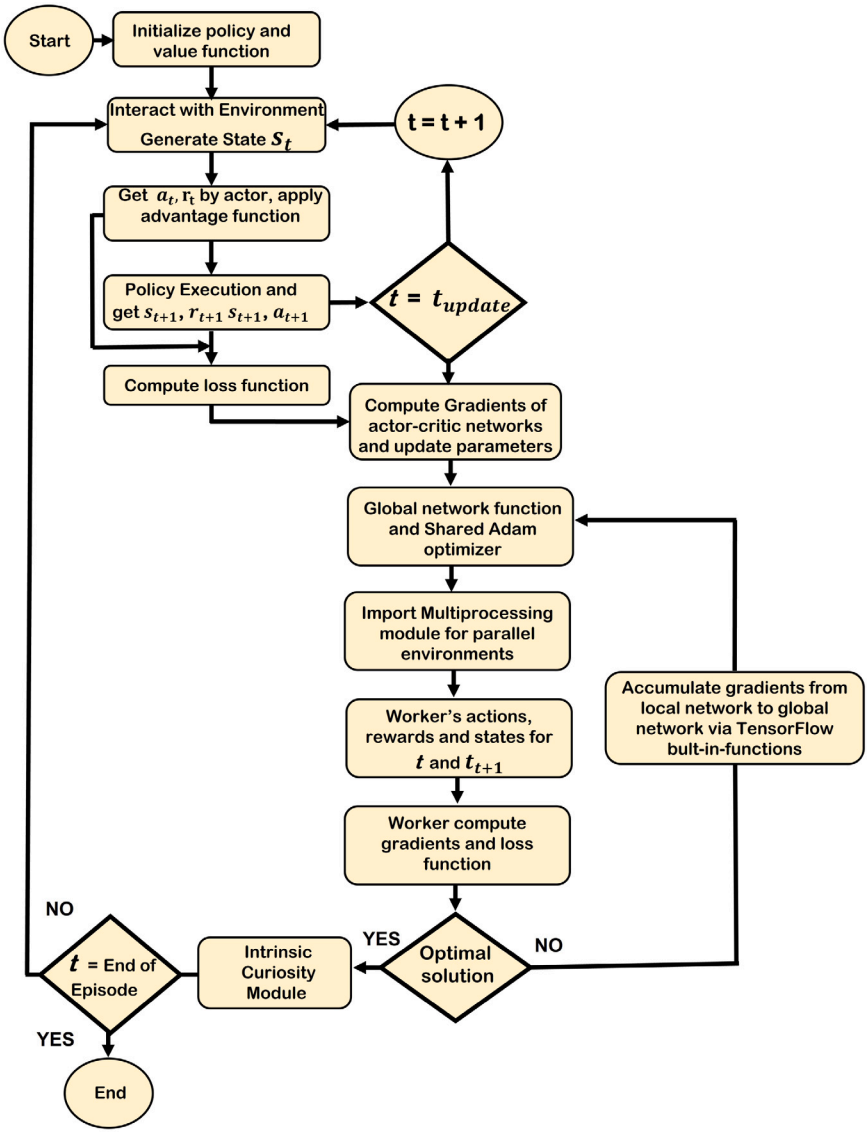


**Fig. 6.** Implementation of A3C model.

updates using an optimizer (e.g., RMSProp or Adam), without waiting for other workers.

6. **Parameter synchronization:** After updating the global network, each worker periodically synchronizes by copying the latest global parameters into its local network to benefit from the improvements learned by others (Nahhas et al., 2022). Fig. 6 shows the implementation of the A3C algorithm.

In UAV applications, A3C allows multiple simulated UAVs to train simultaneously with diverse experiences, accelerating policy development for navigation, tracking, or coordination tasks. It is particularly well-suited for large, continuous environments where traditional single-threaded learning would be too slow or unstable.

The actor loss is a measure of how well the policy network (actor) is learning to choose optimal actions based on the observed state. It is derived from the policy gradient method and encourages the agent to take actions that lead to higher returns (rewards). *Episode length mean* refers to the average number of steps the agent takes in the environment per episode; higher values generally signify better performance. Frames Per Second (FPS) is the step rate that measures the speed of training and reflects how efficiently the system processes the simulation steps. The cumulative reward is the total accumulated reward across iterations and is a direct indicator of the learning progress and performance of the agent. Learning rate is a key hyperparameter that controls how much the model's parameters are updated in response to the gradients calculated during training. Entropy loss is used to encourage exploration by penalizing overly deterministic policies; high entropy loss values promote randomness, while low values indicate convergence. Critic loss quantifies the error between the predicted value of a state and the actual observed return. The critic is responsible for learning the value function, which estimates how good a particular state (or state–action pair) is in terms of expected future rewards. Entropy itself measures the randomness in action selection, with high entropy indicating exploration and low entropy suggesting a more deterministic policy.

In Fig. 7(a), the actor loss initially drops sharply to negative values, then gradually rises to a peak before slowly decreasing. This pattern reflects the agent's early exploration and the gradual stabilization of its policy over time. In Fig. 7(b), the episode length mean increases significantly in the training phase, reaching a peak in the middle of the training iterations, suggesting the agent learns to perform well. In Fig. 7(c), the FPS curve shows that training speed stabilizes early in the process but eventually drops considerably in later stages, likely due to computational overhead. In Fig. 7(d), the cumulative reward

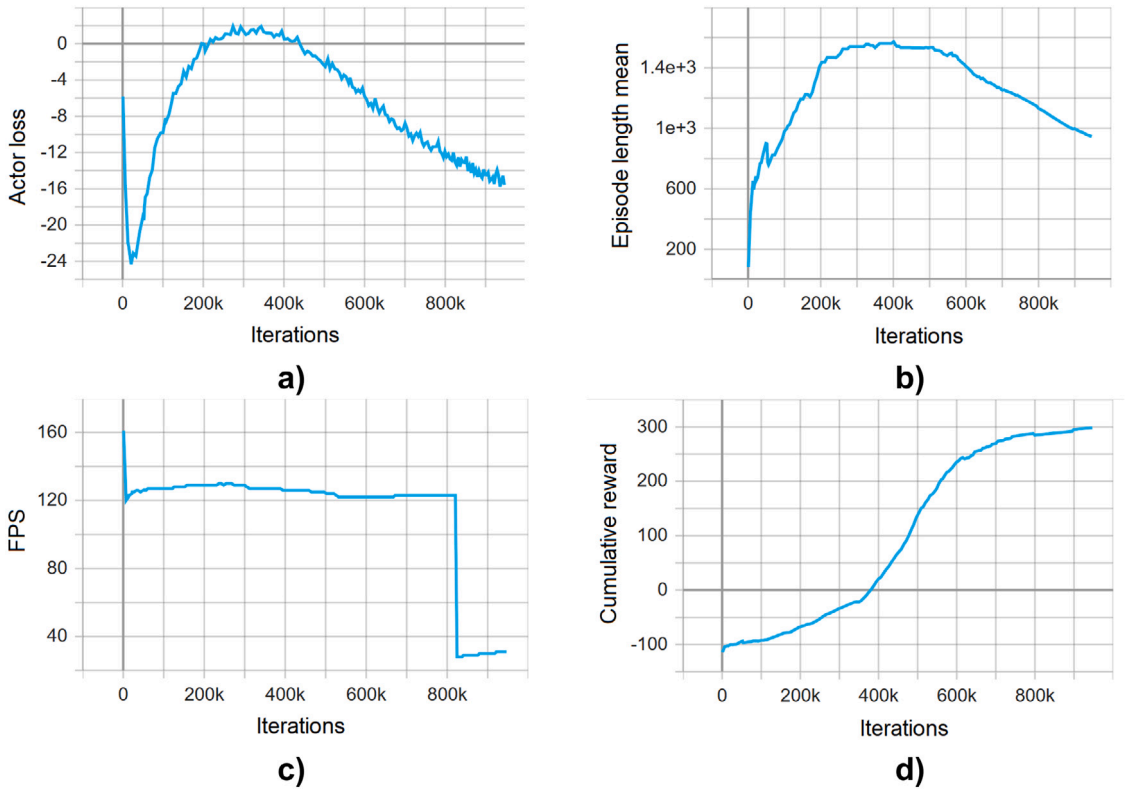


**Fig. 7.** Graphical results for the implementation of A3C algorithm (a) Actor loss, (b) Episode length mean, (c) FPS, (d) Cumulative reward.

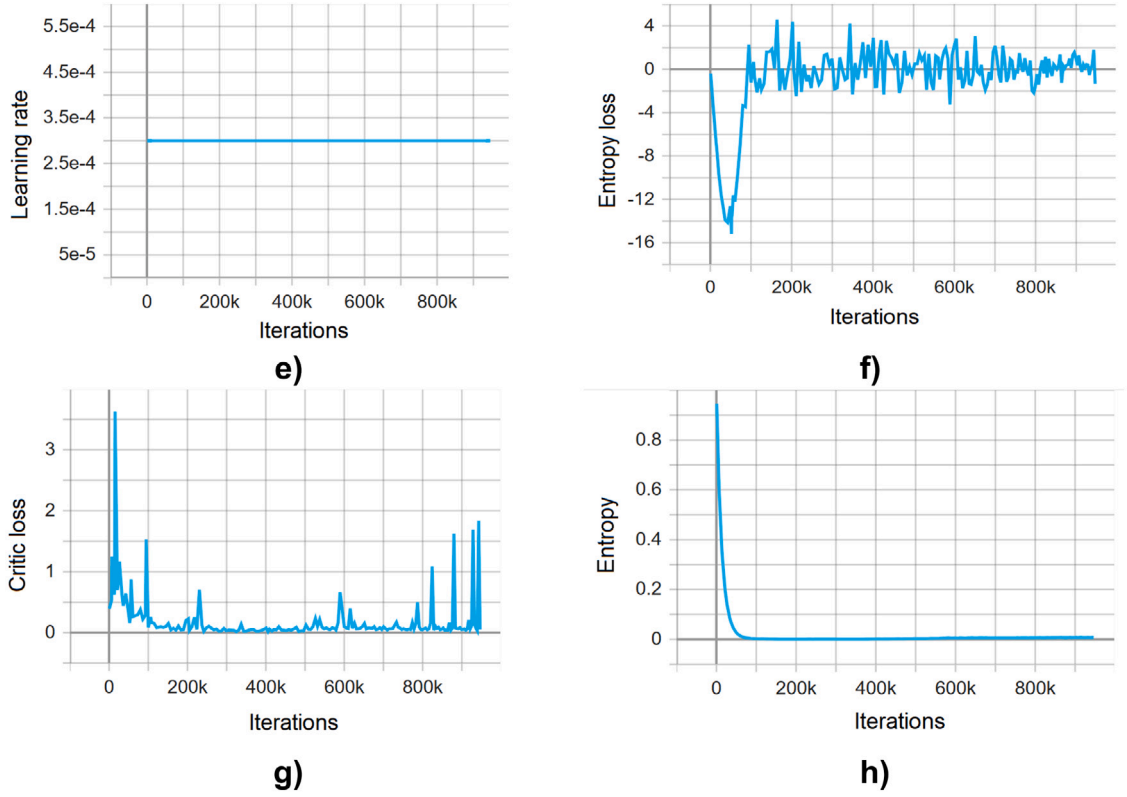


**Fig. 8.** Graphical results for the implementation of A3C algorithm (e) Learning rate (f) entropy loss (g) critic loss (h) entropy.

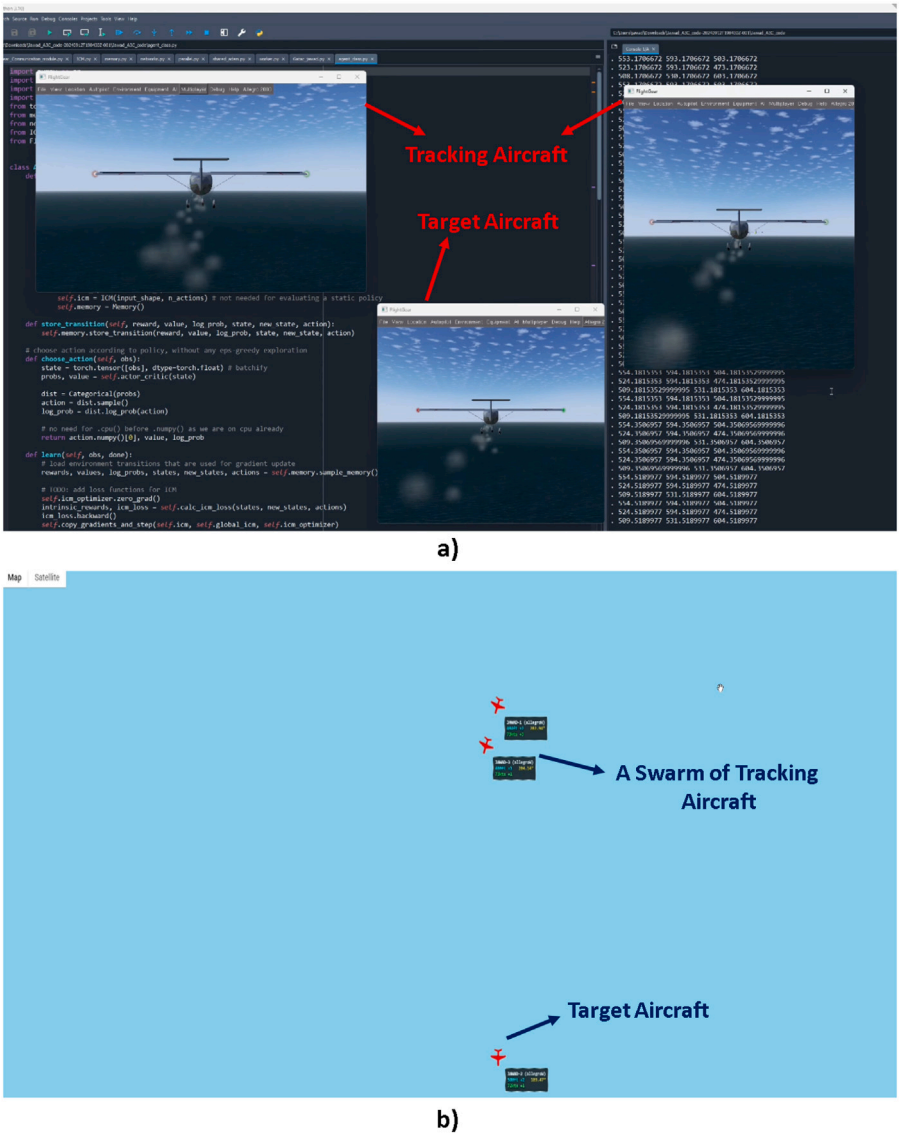


**Fig. 9.** Swarm UAVs to UAV target tracking demonstration on (a) Python editor and (b) FlightGear map.

demonstrates a consistent upward trend, moving from negative values to significantly higher values, indicating successful learning and policy improvement, eventually reaching a plateau that signals convergence.

In Fig. 8(e), the learning rate remains constant throughout the training period. However, the entropy loss in Fig. 8(f) fluctuates heavily at the start, reflecting high randomness and exploration, then gradually settles near zero as the agent becomes more confident and less stochastic in its action selection. The critic loss in Fig. 8(g) displays high variability at the beginning and end of training, with a long stretch of stability in the middle, indicating that the value network becomes reliable during the main training phase. Finally, the entropy graph in Fig. 8(h) shows a sharp decline from high to low values, confirming the transition of the agent's policy from exploratory to deterministic as training progresses.

### *4.3. Swarm UAVs target tracking*

The asynchronous learning architecture of the A3C algorithm makes it highly effective for handling complex and dynamic tasks, such as UAV target tracking. In this study, we utilized the multiplayer feature of FlightGear, which allows real-time interaction among multiple users. This feature supports collaborative or adversarial flight scenarios by allowing pilots to observe and respond to each other's actions within a shared virtual airspace. The target UAV is controlled using the A2C algorithm, enabling it to follow a continuously adaptive trajectory. This setup creates a challenging target for the trackers, as the movement is not predefined but rather evolves based on learned policies. The two tracking UAVs operate under the control of an A3C model integrated with the ICM, which encourages the trackers to explore effectively even when external rewards are sparse or delayed, helping the agents stay engaged in goal-directed behavior by generating internal rewards. This is particularly useful in dynamic path scenarios where the feedback is not immediate. Overall, this approach effectively addresses the delayed reward issue, supports adaptive multi-agent cooperation, and enables robust target tracking in real-time UAV swarm systems. This swarm-based UAV target tracking task involves coordinating multiple tracking agents to follow a single target UAV in real-time. Fig. 9 illustrates the FlightGear map interface used to visualize the real-time execution of the tracking scenario. This visual feedback helps assess tracking accuracy, evaluate swarm behavior, and monitor spatial relationships. Fig. 10 demonstrates different motion patterns of the tracking aircraft as they respond to the manoeuvers of the target UAV.

A three-dimensional trajectory plot is utilized to visualize the flight paths of the target and the tracking UAVs. This visualization employs

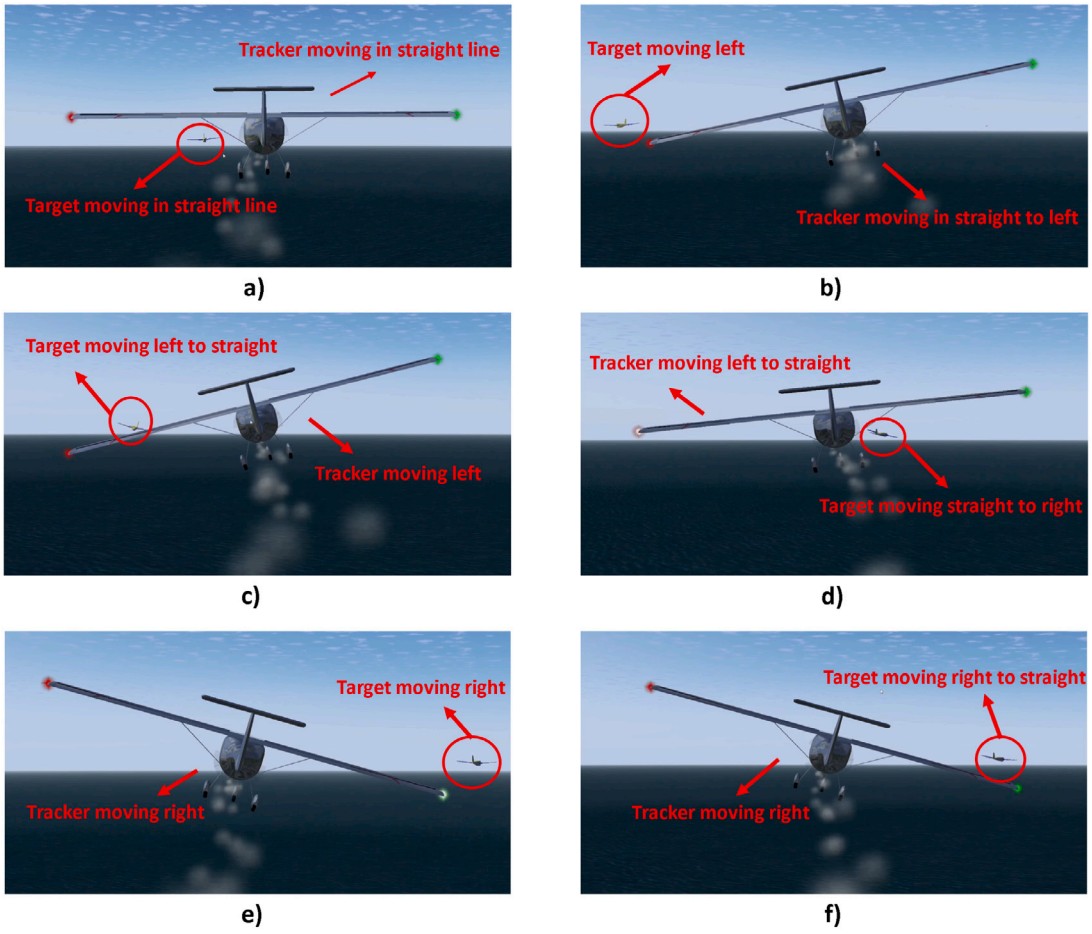


**Fig. 10.** UAV to UAV target tracking (a) Target and tracker moving straight line (b) Target and tracker turning left (c) Target turning right and tracker is turning from left to right (d) Target is turning right and tracker is turning from straight to right (e) Target and tracker are turning right (f) Target is turning straight from right and tracker is turning right.

**Table 2**
Comparison of the testbed from this research with previous studies.

| Testbed | Scalable | Solved Delayed Reward Problem | ICM Based SRCW Hyperparameter Tuning |
|---|---|---|---|
| GaTAC (Sonu & Doshi, 2012) | ✓ | ✗ | ✗ |
| Octo-Rotor RL Testbed (Ahmed et al., 2022) | ✗ | ✗ | ✗ |
| NUAV Testbed (Habib et al., 2017) | ✗ | ✗ | ✗ |
| Micro-UAV Testbed (Michael et al., 2010) | ✗ | ✗ | ✗ |
| RUAV Testbed (Yasar et al., 2006) | ✗ | ✗ | ✗ |
| AutoPilot Testbed (Zhang et al., 2012) | ✗ | ✗ | ✗ |
| Virtual lab Testbed (Moness et al., 2012) | ✗ | ✗ | ✗ |
| dSPACE Testbed (Zhang et al., 2015) | ✗ | ✗ | ✗ |
| MATLAB Co-Simulation(Aschauer et al., 2015) | ✗ | ✗ | ✗ |
| Testbed of this research project | ✓ | ✓ | ✓ |

the X, Y, and Z axes to represent spatial coordinates and altitude, offering a clear and detailed overview of the positional dynamics of the UAVs over time. As illustrated in Fig. 11, the trajectory of the target UAV is highlighted using a distinct color to set it apart from the tracking agents. Each tracking UAV is also assigned a unique color, enabling the visualization of individual flight paths and movement patterns as agents coordinate their efforts to pursue the target.

### *4.4. Demonstration of experimental simulation video for swarm UAVs target tracking in a complex path*

The experimental simulation video was recorded using the iTop screen recorder, with voice-over narration added via Veed software. The resulting video has been uploaded to the Open Science Framework and is available through the corresponding DOI link (Mahmood, 2025). Click on the Files tab to access the MP4 video. The video presents a target tracking scenario that involves one target aircraft and two tracking aircraft.

As shown in Table 2, previous testbeds were developed primarily to evaluate autonomous flight control, coordination, and decision making in real time. However, these systems typically relied on extrinsic rewards and lacked mechanisms to manage delayed reward scenarios. Furthermore, except for GaTAC, existing platforms were not designed for the scalability of UAVs. Delayed rewards remain a significant obstacle in RL, often leading to poor learning performance and unstable policy development. In contrast to earlier testbeds, the testbed in this research incorporates the ICM, effectively addressing the challenge of delayed rewards.

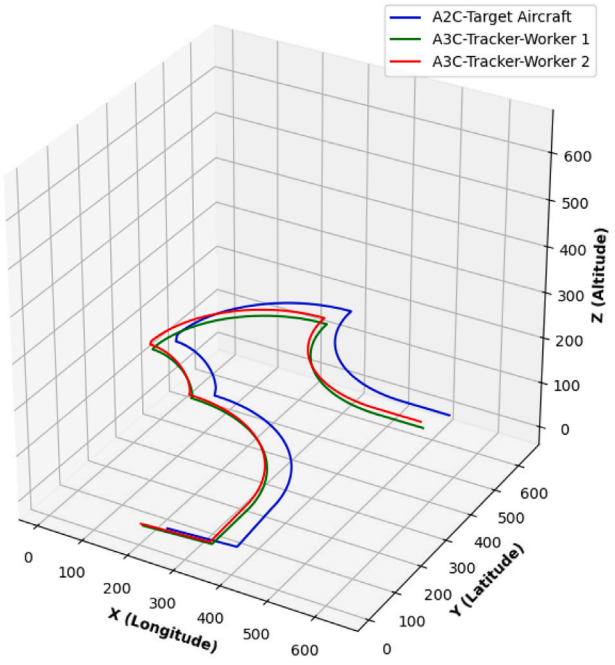


**Fig. 11.** Trajectory plots for target and tracking aircraft.

## 5. Conclusion

This study proposed a RL-based testbed for swarm UAV target tracking, utilizing curiosity-enhanced A2C and A3C models. The framework effectively addressed key limitations of traditional UAV testbeds, specifically, the challenges of delayed rewards and lack of scalability through integration of FlightGear, JSBSim, and state-of-the-art RL techniques. The A2C model was responsible for guiding the target UAV. On the other hand, the A3C algorithm enabled real-time, asynchronous control of multiple tracking UAVs. By incorporating the ICM, the system significantly improved the agent's exploratory behavior, generating internal rewards that mitigated the effects of sparse or delayed extrinsic signals, thus enhancing learning outcomes and policy development. The experimental findings validated the system's capability to accurately track dynamic flight paths, demonstrating coordinated swarm behaviors and adaptability under changing conditions. The inclusion of multiplayer functionality further enriched the simulation environment, allowing realistic testing of cooperative swarm navigation strategies.

## CRediT authorship contribution statement

**Jawad Mahmood:** Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization, Project administration. **Muhammad Adil Raja:** Supervision, Writing – review & editing, Validation, Funding acquisition. **John Loane:** Resources, Writing – review & editing, Validation. **Fergal McCaffery:** Supervision, Writing – review & editing, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This research was funded by the Technological University Transformation Fund (TUTF) of the Higher Education Authority (HEA) of Ireland.

## Data availability

Data will be made available on request.

## References

Ahmed, I., Quinones-Grueiro, M., & Biswas, G. (2022). A high-fidelity simulation testbed for fault-tolerant octo-rotor control using reinforcement learning. In *2022 IEEE/aIAA 41st digital avionics systems conference* (pp. 1–10). IEEE.

Aschauer, G., Schirrer, A., & Kozek, M. (2015). Co-simulation of matlab and flightgear for identification and control of aircraft. *IFAC-PapersOnLine*, *48*(1), 67–72.

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., & Kautz, J. (2016). Reinforcement learning through asynchronous advantage actor-critic on a gpu. arXiv preprint arXiv:1611.06256.

Bougie, N., & Ichise, R. (2020). Skill-based curiosity for intrinsically motivated reinforcement learning. *Machine Learning*, *109*, 493–512.

Colas, C., Fournier, P., Chetouani, M., Sigaud, O., & Oudeyer, P.-Y. (2019). Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning* (pp. 1331–1340). PMLR.

Habib, S., Malik, M., Rahman, S. U., & Raja, M. A. (2017). Nuav-a testbed for developing autonomous unmanned aerial vehicles. In *2017 international conference on communication, computing and digital systems* (pp. 185–192). IEEE.

Li, J., & Gajane, P. (2023). Curiosity-driven exploration in sparse-reward multi-agent reinforcement learning. arXiv preprint arXiv:2302.10825.

Li, B., Lu, T., Li, J., Lu, N., Cai, Y., & Wang, S. (2019). Curiosity-driven exploration for off-policy reinforcement learning methods. In *2019 IEEE international conference on robotics and biomimetics* (pp. 1109–1114). IEEE.

Lin, Z., Lai, J., Chen, X., Cao, L., & Wang, J. (2022). Learning to utilize curiosity: A new approach of automatic curriculum learning for deep RL. *Mathematics*, *10*(14), 2523.

Mahmood, J. (2025). Video demonstration of experimental simulation of target tracking of UAVs based on distributed networking framework. http://dx.doi.org/10.17605/OSF.IO/EGH6J, (Accessed April 2025).

Michael, N., Mellinger, D., Lindsey, Q., & Kumar, V. (2010). The grasp multiple micro-uav testbed. *IEEE Robotics & Automation Magazine*, *17*(3), 56–65.

Moness, M., Mostafa, A. M., Abdel-Fadeel, M. A., Aly, A. I., & Al-Shamandy, A. (2012). Automatic control education using FlightGear and MATLAB based virtual lab. In *The international conference on electrical engineering: Vol. 8*, (pp. 1–15). Military Technical College.

Nahhas, A., Kharitonov, A., & Turowski, K. (2022). Deep reinforcement learning techniques for solving hybrid flow shop scheduling problems: Proximal policy optimization (PPO) and asynchronous advantage actor-critic (A3C).

Sonu, E., & Doshi, P. (2012). Gatac: A scalable and realistic testbed for multiagent decision making. In *AAMAS* (pp. 1507–1508). Citeseer.

Stadie, B., Zhang, L., & Ba, J. (2020). Learning intrinsic rewards as a bi-level optimization problem. In *Conference on uncertainty in artificial intelligence* (pp. 111–120). PMLR.

Sun, H., Chai, Y., Wang, S., Sun, Y., Wu, H., & Wang, H. (2025). Curiosity-driven reinforcement learning from human feedback. arXiv preprint arXiv:2501.11463.

Wang, S., Li, S., Sun, T., Fu, J., Cheng, Q., Ye, J., et al. (2024). Llm can achieve self-regulation via hyperparameter aware generation. arXiv preprint arXiv:2402.11251.

Wang, Y., Liu, C., Li, Y., Amani, S., Zhou, B., & Yang, L. F. (2024). Hyper: Hyperparameter robust efficient exploration in reinforcement learning. arXiv preprint arXiv:2412.03767.

Wu, C., Yu, W., Liao, W., & Ou, Y. (2024). Deep reinforcement learning with intrinsic curiosity module based trajectory tracking control for USV. *Ocean Engineering*, *308*, Article 118342.

Yasar, M., Bridges, D., Mallapragada, G., & Horn, J. (2006). A simulation test bed for coordination of unmanned rotorcraft and ground vehicles. In *AIAA modeling and simulation technologies conference and exhibit* (p. 6263).

Zhang, J., Geng, Q., & Fei, Q. (2012). UAV flight control system modeling and simulation based on FlightGear. In *International conference on automatic control and artificial intelligence* (pp. 2231–2234). IET.

Zhang, H.-M., Zhou, Q., & Xu, G.-Y. (2015). Hardware-in-the-loop simulation platform for UAV based on dSPACE. In *2015 international conference on computational science and engineering* (pp. 450–454). Atlantis Press.

Zhelo, O., Zhang, J., Tai, L., Liu, M., & Burgard, W. (2018). Curiosity-driven exploration for mapless navigation with deep reinforcement learning. arXiv preprint arXiv:1804.00456.

Zheng, L., Chen, J., Wang, J., He, J., Hu, Y., Chen, Y., et al. (2021). Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, *34*, 3757–3769.

Zhou, K., Wang, W., Hu, T., & Deng, K. (2021). Application of improved asynchronous advantage actor critic reinforcement learning model on anomaly detection. *Entropy*, *23*(3), 274.