Development and validation of a safe reinforcement learning drone controller

García-Quijano, H.*, Pérez-Muñoz, A.G, López-García, G., Alonso, A.
STRAST group – Universidad Politécnica de Madrid

Objectives

Explore and validate the use of ML techniques on embedded safety systems.

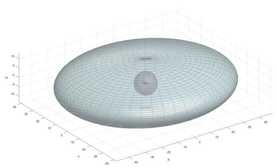
- Develop a neural network (NN) onboard a drone for collision avoidance.
- Generate adequate scenarios (data) for training and testing.
- Validate the neural network:
  - ✓ Functional requirements
  - ✓ Safety requirements: (ECSS-E-ST-40C)
- Time requirements, Bounded resources, Source code verification: Static analysis

Experimental setup

Simulated Scenario

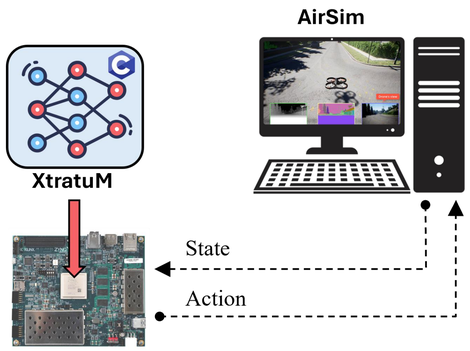
The flight environment was created in Microsoft AirSim.
• **Two UAVs**: The scenario involves "Drone A" (the RL-controlled agent) and "Drone B" (the attacker).
• **Adversary goal**: Drone B is programmed to actively pursue and collide with Drone A.

• **Key zones**:
  • **Vision zone**: Ellipsoid-shaped area defining the observable range of Drone A for detecting Drone B.
  • **Setpoint**: A marked point on the map where Drone A has to fly within 2m.



Execution Platform (Processor-in-the-Loop)

To test performance under hardware constraints, the controller executed on the embedded computer:
• **Hardware**: Ultrascale+ ZCU104.
• **Software**: The controller runs on the XtratuM hypervisor.
• **Communication**: The board connects to the host PC running AirSim via serial line (UART).



Drone controller with RL

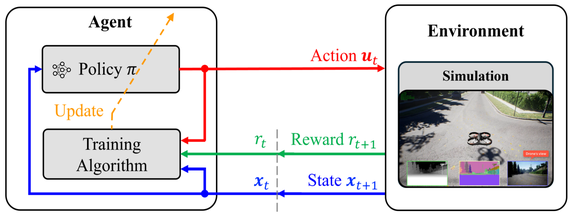
The controller is a Reinforcement Learning (RL) agent trained to follow a trajectory (sequence of setpoints) in an urban environment while avoiding a pursuing attacker drone. We selected the **Soft Actor-Critic (SAC)**, an off-policy agent ideal for continuous action spaces that balances the maximization of rewards and exploration.

Agent Design

1. State Space (What the Agent Observes)
The agent receives a set of 12 continuous values describing the environment:
• **Distance to Current Setpoint**: A 3D vector representing the spatial offset to the target.
• **Drone's Velocity**: The drone's current speed and direction as a 3D vector.
• **Attacker's Relative Position**: A 3D vector to the attacker, if visible. Set to zero if the attacker is outside the "vision zone".
• **Attacker's Relative Velocity**: The attacker's motion relative to the drone, if visible.

2. Action Space (What the Agent Does)
The agent's output is a velocity command:
• **Action**: Continuous 3D velocity vector, where each axis is in the range $[-7{,}7]\ \mathrm{ms}^{-1}$.
• **Control Period**: The agent acts every 0.2 seconds.

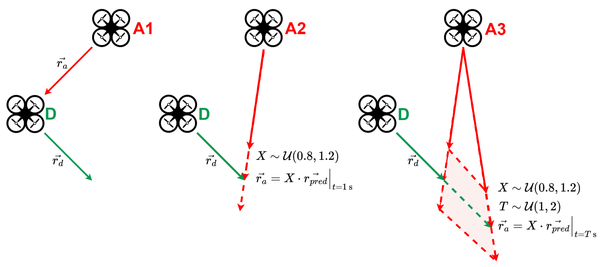
3. Reward Function (How the Agent Learns)
The agent is trained to maximize a cumulative reward that balances multiple objectives.
• **Positive Rewards [+]:**
  •**Setpoint Arrival**: A reward of +5000 is given when reaching a setpoint.
  •**Path Completion**: A +10000 bonus is rewarded for successfully finishing the entire trajectory.
• **Penalties [-]:**
  •**Distance Penalty**: A negative reward proportional to the cube of the distance to the setpoint encourages precision.
  •**Safety Penalty**: A massive penalty of -1000 is applied for collisions or leaving the designated flight area.
  •**Proximity Penalty**: An additional penalty is applied if the distance to an attacker falls below 5 meters.
  •**Time Penalty**: A small penalty of -1 per step incentivizes faster path completion.



Environment Design

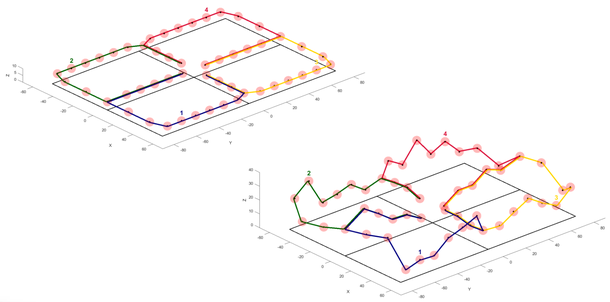
Urban Simulation Map:
The simulation is set in a detailed **urban map** within the **Unreal Engine**, designed to mimic real-world flight challenges with varied building heights and street patterns. The **AirSim** simulator is used to model drone physics and interactions. To guide navigation, **setpoints** are positioned along intended paths, defining a continuous trajectory for the drone to follow.

Attacker Strategies:
To ensure that the RL agent learns robust evasion strategies, three types of **attacker drones** were designed with different pursuit behaviors, creating an unpredictable and challenging environment:
• **A1 (Deterministic Pursuit):** Tracks the defender's current position with a direct, real-time approach.
• **A2 (Predictive Pursuit):** Estimates the defender's future position over a fixed 1-second horizon, scaled by a random factor to introduce variability.
• **A3 (Stochastic Pursuit):** Amplifies unpredictability by using a variable prediction horizon (between 1 and 2 seconds) with random scaling.



Training Process

•**Algorithm**: Soft Actor-Critic (SAC) with automatically adjusting entropy for a balanced exploration-exploitation strategy.
•**Training Steps**: The agent was trained for **400,000 steps**, a duration found to optimize performance without causing overfitting.
•**Environment**: During training, the agent navigated one of four predefined circuits, which was randomly selected at the start of each episode. The attacker's starting position was also randomized to improve the agent's adaptability.
•**Network Architecture**: Both the policy and critic networks used a deep architecture with 4 hidden layers and 512 neurons per layer.



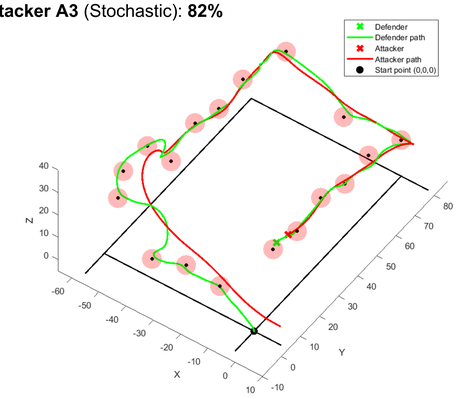
Results

The trained models were validated in a rigorous testing process to evaluate their generalization and robustness in scenarios not seen during training. For each test, the **altitude of every setpoint was randomized** between 6 and 35 meters, forcing the agent to navigate entirely new 3D trajectories.

Final Model Performance

The best-performing agent was trained for 400,000 steps against the simplest deterministic attacker (A1). This model demonstrated outstanding robustness, generalizing effectively to handle the complex and unpredictable behaviors of attackers A2 and A3.

Success Rates over 100 Episodes:
• **Attacker A1** (Deterministic): **87%**
• **Attacker A2** (Predictive): **78%**
• **Attacker A3** (Stochastic): **82%**



Verification and validation

• **Processor-in-the-Loop (PIL)**: The final model was converted to MISRA-C code and then deployed to the Ultrascale+ ZCU104 board running the XtratuM hypervisor.
• **Safety Verification**:
  • **Resources**: Verified bounded usage of CPU and memory.
  • **Timing**: Response time under the 200-ms deadline.
  • **Code**: The static analysis of the code met safety requirements.

Conclusions

• This work demonstrates the development and validation of a safe Reinforcement Learning (RL) drone controller. The final model showed excellent generalization, adapting from simple training to complex and unseen trajectories with an adversary attacker.

• The Processor-in-the-Loop (PIL) validation confirmed that the neural network controller meets strict timing deadlines and has bounded resource usage.

• The results confirm that RL-based neural networks are a promising and adequate solution for developing complex control systems for safety-critical embedded applications like UAVs.

• Future efforts will focus on improving the RL agent's capabilities by training and validating it in more complex scenarios, including real-world flight tests against more advanced attacker strategies.

Real Time Communications Conference and Expo at Illinois Tech
IEEE International Conference
IEEE RTC 2025 - October 7-8, 2025 - IIT, Chicago, IL, USA