

Automatic Piloting of Drones by Reinforcement Learning

SENECHAL Morgan

M1 Big Data & Machine Learning | EFREI Paris

Mai 2024

Introduction

Unmanned aerial vehicles (UAVs) play a key role in varied applications such as search, rescue and surveillance. However, their autonomous navigation in unpredictable environments remains a major challenge. To overcome this difficulty, a new Deep Reinforcement Learning (DRL) algorithm, named Robust-DDPG, was developed. It uses an actor-critic approach to simultaneously manage speed, orientation and reduce collision risks, enabling efficient navigation in dynamic environments without prior mapping.

Prerequisites

Reinforcement learning (RL)



Motion control of a UAV is formalized via reinforcement learning, where the UAV learns optimal control policies by interacting with a dynamic environment characterized by changing conditions and unexpected events.

Drone kinematics

This equation describes the motion of a UAV in terms of its planar position and flight angles, while accounting for disturbances, as it makes coordinated turns at a constant altitude.

$$\frac{d}{dt} \begin{pmatrix} x_u \\ y_u \\ \psi_u \\ \phi_u \end{pmatrix} = \begin{pmatrix} v_u \cos \psi_u + \eta_x \\ v_u \sin \psi_u + \eta_y \\ -(g/v_u) \tan \phi_u + \eta_\psi \\ f(\phi_u, a_\phi) \end{pmatrix}$$

x_u, y_u : Cartesian planar position of the UAV

g : acceleration due to gravity.

ψ_u, ϕ_u : heading angle and roll angle

v_u : linear speed of the UAV.

$\eta_x, \eta_y, \eta_\psi$: disturbance terms due to speed and rate of change of heading

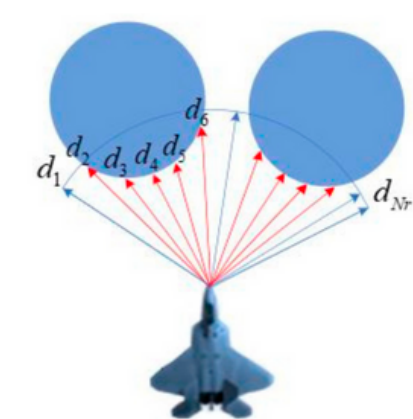
$$\begin{cases} v_{u,t} = (1 - \lambda_v) v_{u,t-1} + \lambda_v (1 - a_{v,t}) v_{u,max} \\ \phi_{u,t} = (1 - \lambda_\phi) \phi_{u,t-1} + \lambda_\phi a_{\phi,t} \phi_{u,max} \end{cases}$$

$v_{u,t}$: Speed control update

$\phi_{u,t}$: Roll command update

Methods

State and Action



State(s): The state of the system is a vector of information that includes planar position (x_u, y_u), planar velocity (x'_u, y'_u), heading angles ψ_u and roll angles ϕ_u , as well as the distances d_i detected by the LiDAR for each ray N_r , which measure the relative distance to surrounding threats. Additionally, the state includes the target position (x_T, y_T), transmitted periodically to the UAV.

Action (a): The action is a vector that includes the collision probability a_v to control the UAV speed and the signal a_ϕ to control the roll angle. a_v varies between 0 and 1, where 1 represents the maximum collision probability, and a_ϕ varies between -1 and 1, where the values determine the direction and amount of roll.

Reward

Reward function:

$$r(s, a) = \mu_1 r_A + \mu_2 r_B + \mu_3 r_C + \mu_4 r_I$$

$\mu_1, \mu_2, \mu_3, \mu_4$: weight the importance of the rewards: sum = 1.

$$r_A = D_{ut}^{pre} - D_{ut}^{cur}$$

$$r_B = (v_u / v_{u,max}) \times \cos \Delta \psi$$

$$r_C = -\Delta \psi / 4$$

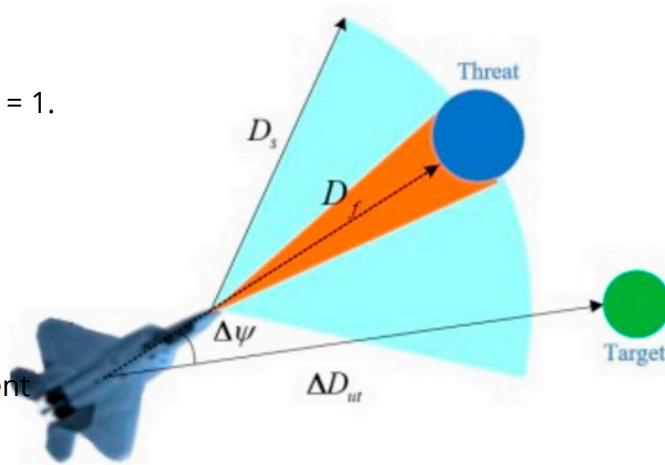
$$r_D = (v_u / v_{u,max}) \times (D_f / D_s - 1)$$

rA: approach towards the target

rB: speed towards the target taking into account alignment

rC: penalty for movement away from the target

rD: penalty for approaching potential threats



Future Rewards & Objective Function

Objective function (J_π):

$$J_\pi = \mathbb{E}_{s_i \sim p, a_i \sim \pi} \left[\sum_{i=0}^H \gamma^i r(s_i, a_i) \right]$$

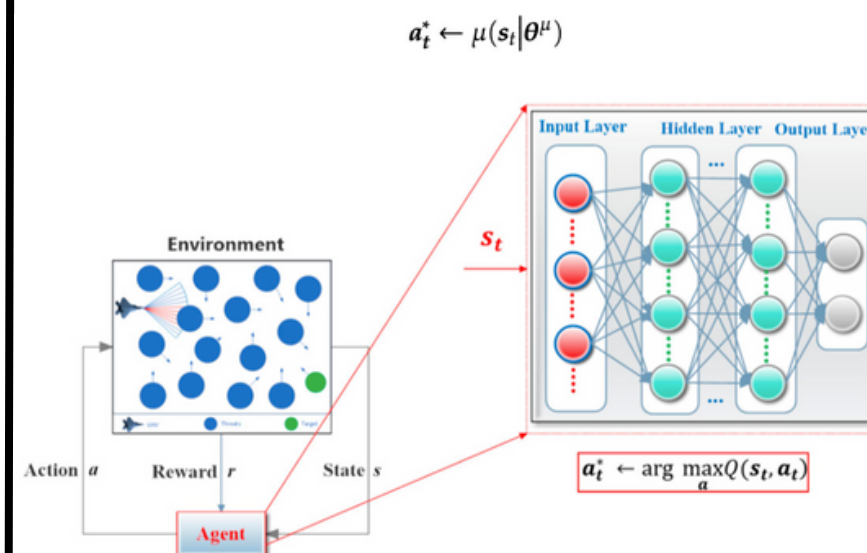
Policy π is optimized to maximize the weighted sum of future rewards over horizon H , with weighting given by a discount factor γ .

Q-Value : $Q_\pi(s_i, a_i)$

$$\begin{aligned} &= \mathbb{E}_{s_i \sim p, a_i \sim \pi} \left[\sum_{i=t}^H \gamma^{i-t} r(s_i, a_i) | s_t, a_t \right] \\ &= \mathbb{E}_{s_{t+1} \sim p} \left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi} [Q_\pi(s_{t+1}, a_{t+1})] \right] \end{aligned}$$

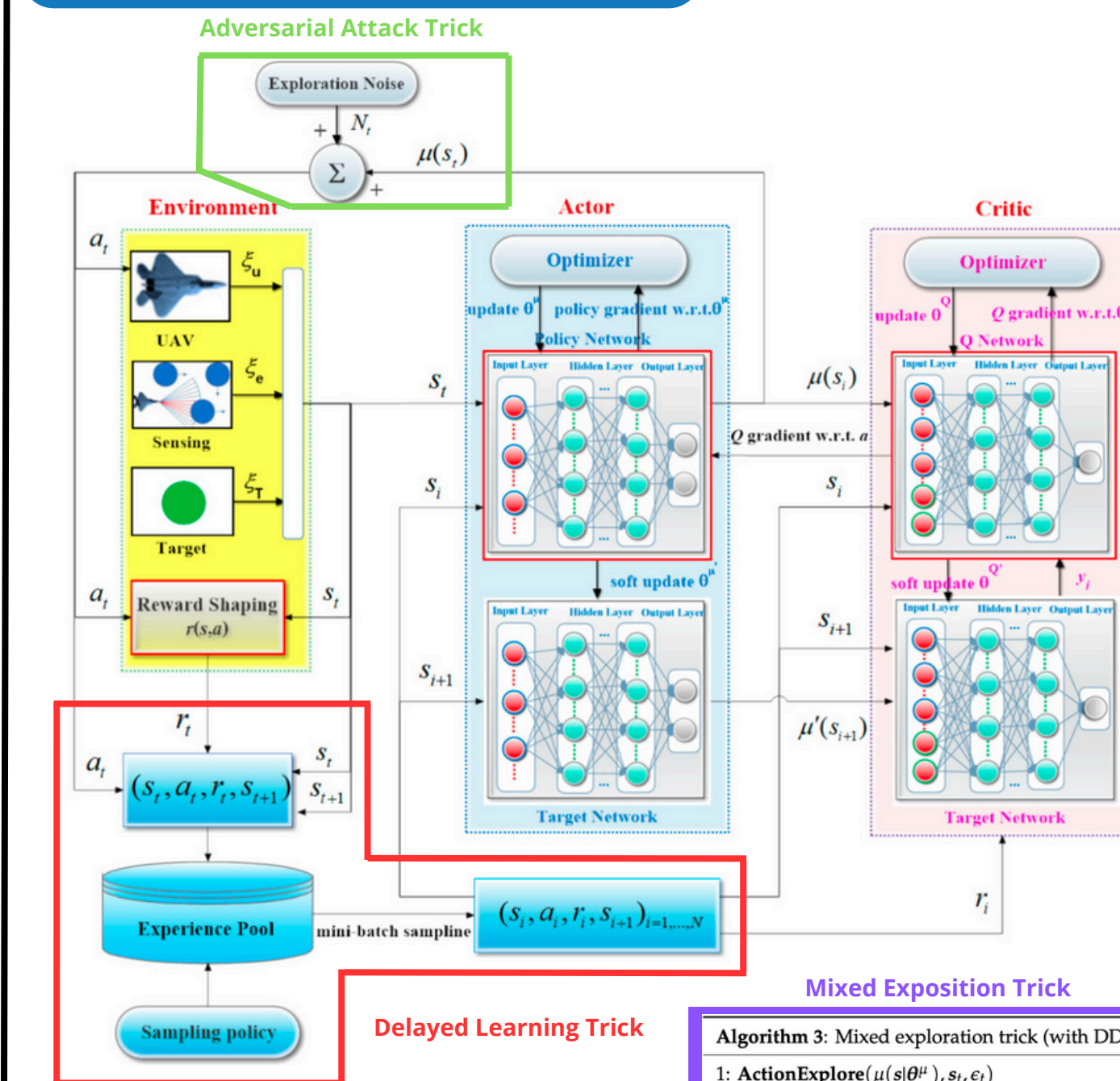
For a state-action pair (s, a), the Q-value is the measure of the quality of action a in state s under policy π . It is defined as the expected sum of future rewards discounted, starting with state s and action a and then following policy π .

Using a Deep Neural Network:



The optimal action at $*$ is identified by maximizing the Q-value for a specific state. To approximately determine the best action, a deep neural network $f_\theta(s)$ directly links a continuous state to its optimal action. The network is trained to estimate the parameters θ^* guiding the control policy.

Robust Deep deterministic policy Gradient



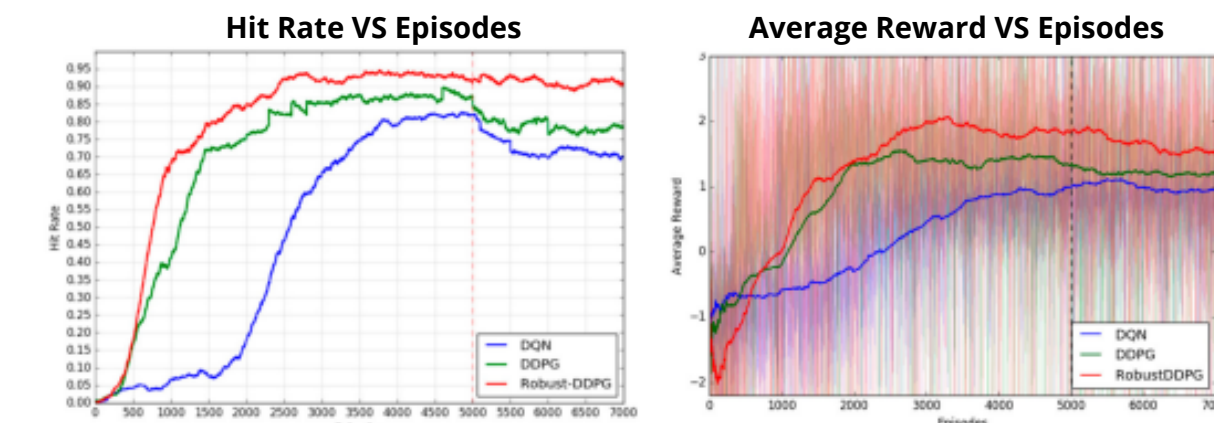
Function which randomly chooses, depending on the ϵ parameter, between a uniform action or one influenced by noise OR. The action is either uniformly selected between [a_{low}, a_{high}], or adjusted by noise OR and clipped within the allowed limits.

Mixed Exposition Trick

Algorithm 3: Mixed exploration trick (with DDPG)

```
1: ActionExplore( $\mu(s|\theta^\mu), s_t, \epsilon_t$ )
2: if rand <  $\epsilon_t$  do
3:    $a_t = \text{Uniform}(a_{low}, a_{high})$ 
4: else do
5:    $a_t = \mu(s_t|\theta^\mu) + \text{OU}(u, \sigma)$ 
6:    $a_t = \text{clip}(a_t, a_{low}, a_{high})$ 
7: end if
8: return  $a_t$ 
```

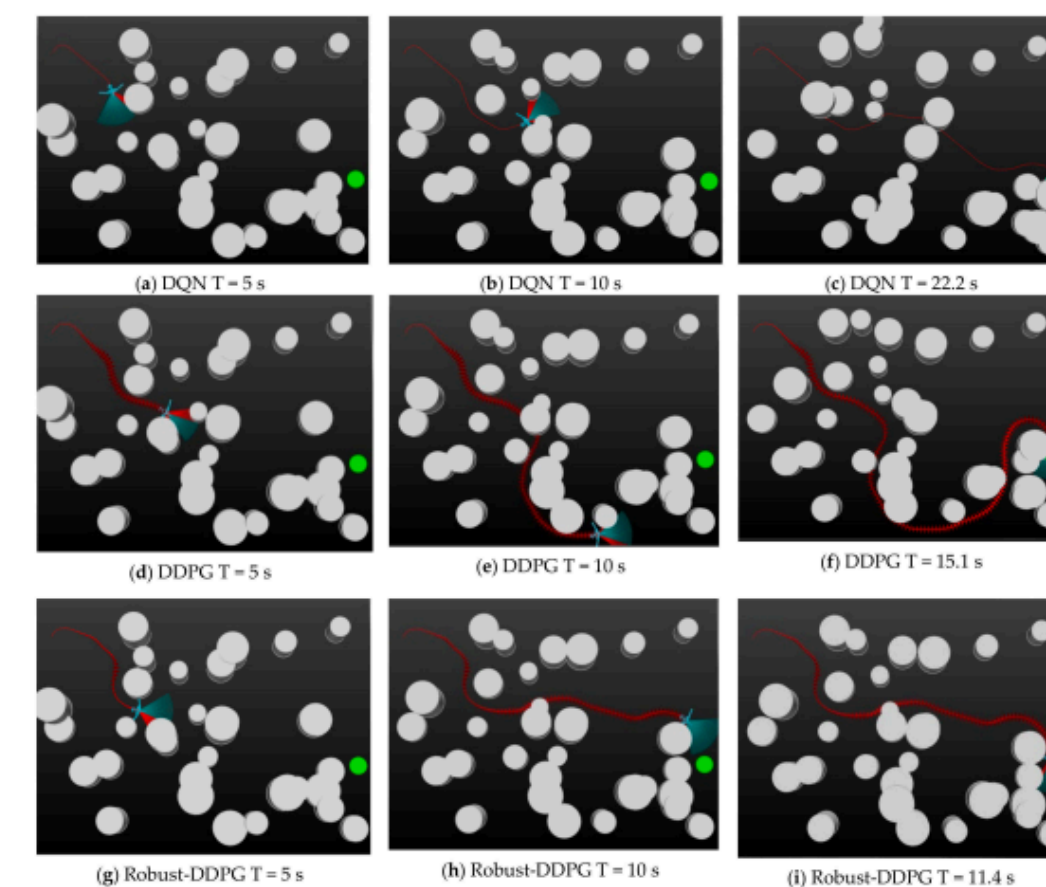
Results



Set of results on several algorithms

	Learning Stage			Exploiting Stage		
	Hit Rate	Crash Rate	Lost Rate	Hit Rate	Crash Rate	Lost Rate
DQN	82.4%	15.1%	2.5%	70.2%	22.1%	7.7%
DDPG	88.7%	9.8%	1.5%	78.9%	17.9%	3.2%
Robust-DDPG	92.6%	6.8%	0.6%	91.0%	7.4%	1.6%

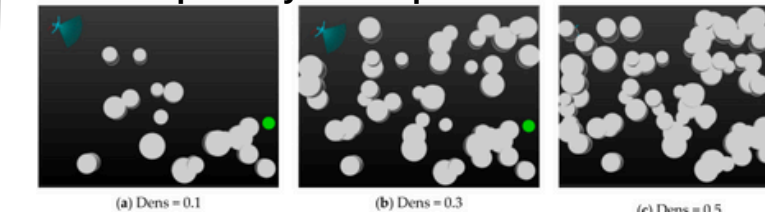
Test



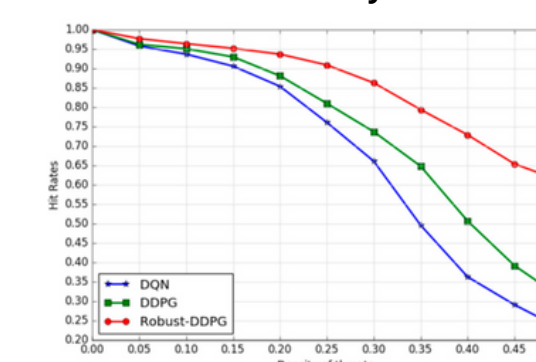
Set of results on several algorithms

Agent	Flight Time	Path Length
DQN	22.2 s	463.2 m
DDPG	15.1 s	651.1 m
Robust-DDPG	11.4 s	443.5 m

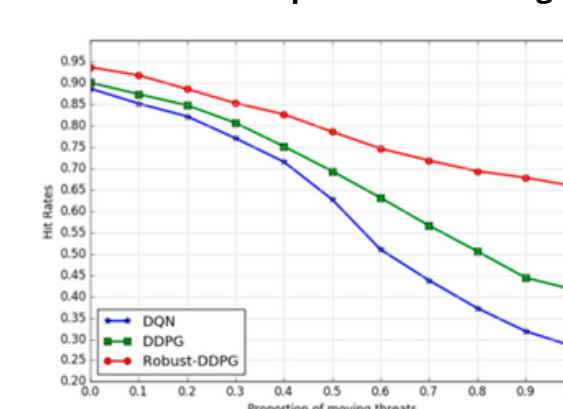
Adaptability to Complex Environments



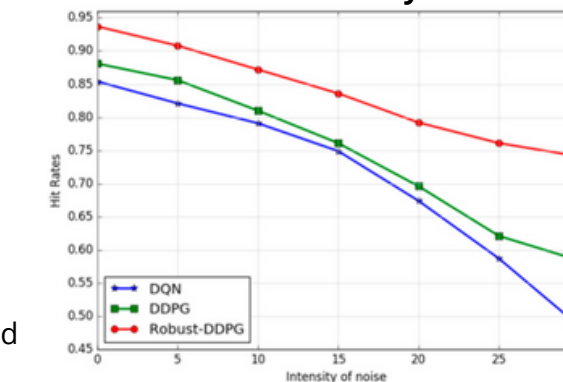
Hit Rates VS Density of threats



Hit Rates VS Proportion of moving threats



Hit Rates VS Intensity of noise



Sources

Article 1: HAL open science : véhicule autonome avec RL

Article 2 : HAL open science : Machine Learning UAV-Assisted

Article 3: HAL open science : drones à base RL

Article 4 : Remote Sensing : Robust-DDPG