Reinforcement Learning for Spacecraft Guidance

Thales Alenia Space

A. Bacot, M. Chau, and A. Salmona

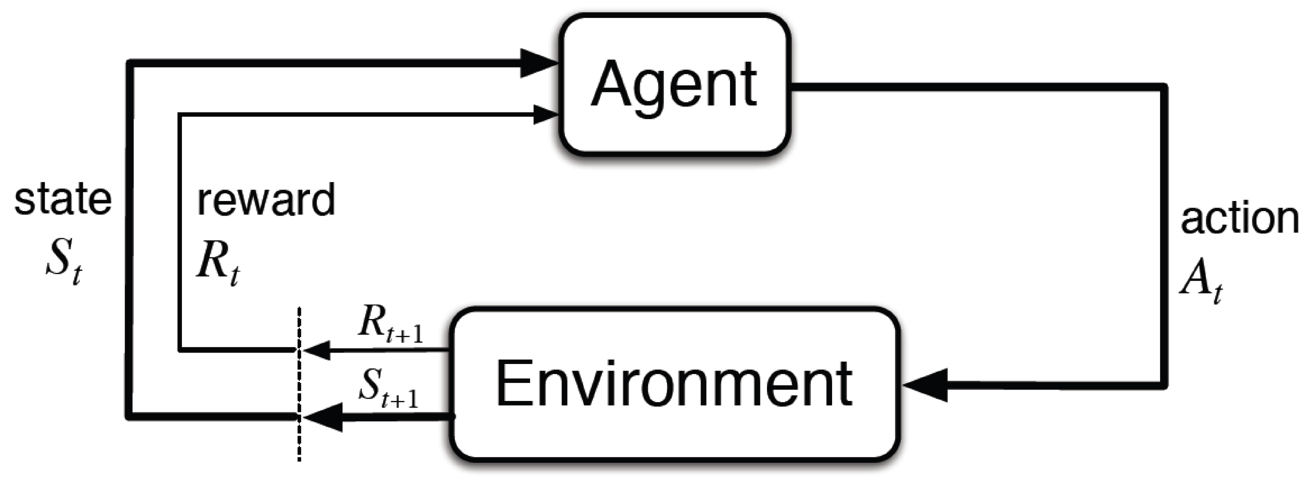
École des Ponts ParisTech, Champs-sur-Marne, France

Abstract

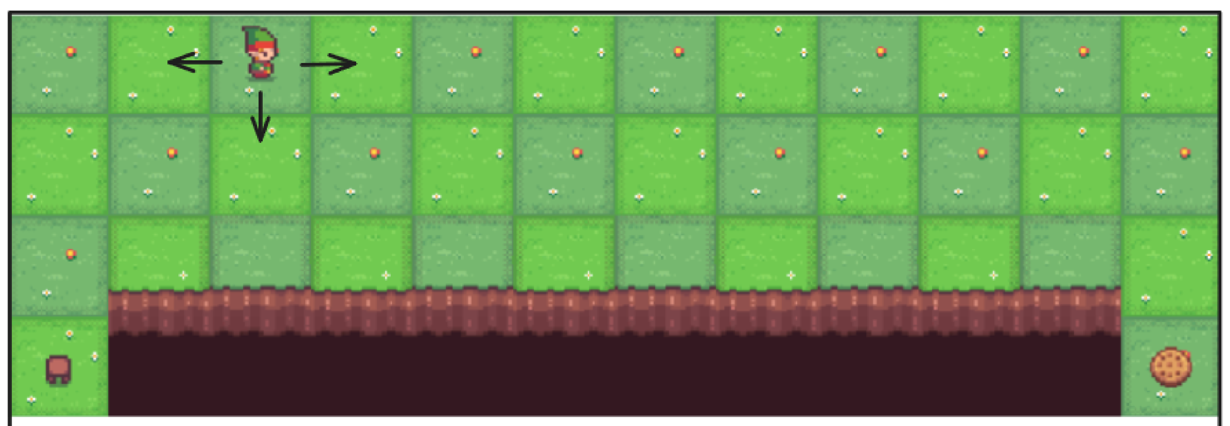
This project explores the application of a reinforcement learning (RL) algorithm to control a satellite in geostationary orbit for autonomous rendez-vous maneuvers with a target using a low propulsion motor. Traditional methods for such maneuvers rely on pre-programmed control strategies, which can be computationally expensive and inflexible. The success of this project will be measured by the ability of the trained RL agent to autonomously guide the simulated satellite to rendez-vous with the target within specified time, accuract and resource constraints. If successful, this approach has the potential to revolutionize on-orbit operations for geostationary satellites, enabling more efficient and adaptable maneuvers for tasks like inspection, servicing, and active debris removal.

Introduction to Reinforcement Learning

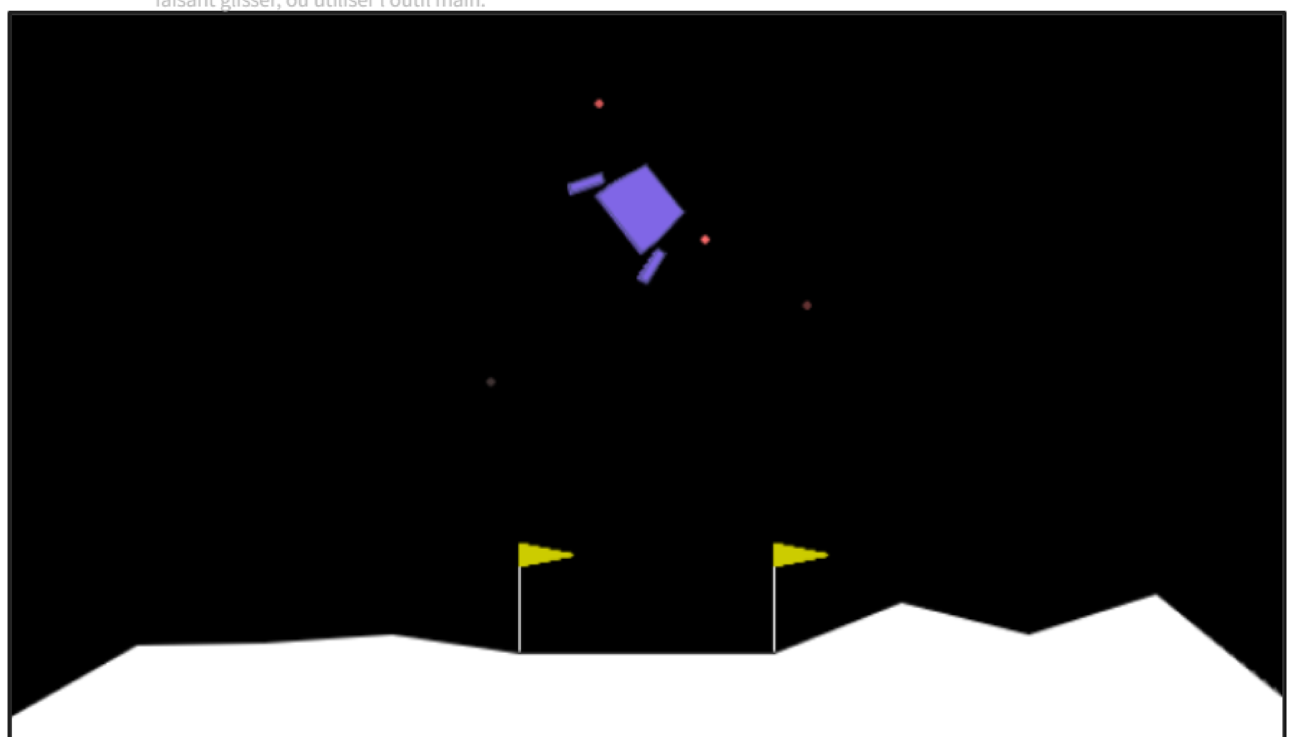
Reinforcement learning is a type of machine learning where an agent learns through trial and error in an environment. It receives rewards for actions that move it closer to a goal and avoids penalties for those that don't. The main framework for RL is the python library `gymnasium` that provides classic environments and tools to visualize training and test phases. The core of the project involves implementing a sophisticated RL algorithm on a simulated environment replicating the dynamics of geostationary satellites. The algorithm will receive the current state of the satellite (position, velocity and orientation) as input and will determine the optimal thrust direction and magnitude for the electric propulsion motor to achieve a safe and efficient approach towards the target. The RL agent will learn through trial and error within the simulation, receiving rewards for actions that bring it closer to the target while considering factors like energy consumption and arrival position accuracy.



2 simple examples of RL resolved problems



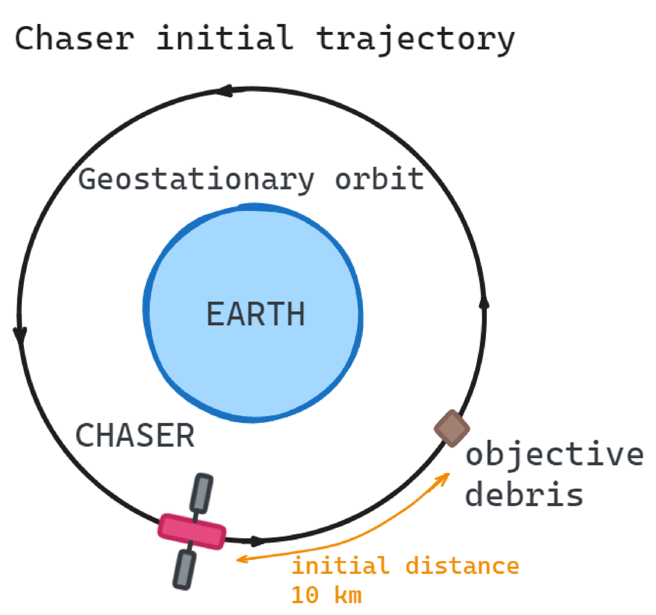
```
Cliff Walking
goal : walk from the stool to the cookie
env : 4*12 grid
actions : walk left, right, forward or backward
reward : - 1 for each case exept the cookie one (0)
         - 300 for the cliff's squares
```



```
Lunar Lander
goal : land between the flags
env : 2D discretised plan with lunar gravity
actions : propulsion up, right or left
reward : quite complex, takes in account speed, angle,
         altitude, propulsion and ground contact
```

Spacecraft Guidance problem and Dynamics

Several sensors provide access to the position $\mathbf{r}$, speed $\mathbf{v}$ and angular speed ω, and rotational quaternion $\mathbf{q}$ of the satellite. The state of the system is therefore described by the vector $\mathbf{s} = \left(\mathbf{r}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \omega^\top\right)^\top$. The satellite is equipped with a thruster at the rear and a reaction wheel. The system control vector is therefore $\mathbf{u} = \left(\mathbf{F}^\top, \mathbf{L}^\top\right)^\top$, where $\mathbf{F}$ is the reactor thrust force and $\mathbf{L}$ is the torque exerted by the reaction wheel.



Thanks to the Clohessy - Wiltshire equations, the system dynamics are of the following form :

$$\begin{cases} \ddot{x} = 3n^2x + 2n\dot{y} + \frac{F_x}{m} \\ \ddot{y} = -2n\dot{x} + \frac{F_y}{m} \\ \ddot{z} = -n^2z + \frac{F_z}{m} \\ \dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}\omega \\ \dot{\omega} = \mathbf{J}^{-1}\left(\mathbf{L} - \omega \times \mathbf{J}\omega\right) \end{cases}$$

The main goal is to find an optimal control policy in the sense of minimizing the speed variation $\int_0^T \|\mathbf{F}(s)\|\, ds$, where $T \leq 1$ week. We have additional observability constraints and limitations on thrust force, since the satellite is powered by a low propulsion engine.

How to treat Spacecraft Guidance with RL

- Implement the custom `gymnasium` environment - this consists in discretizing the Clohessy-Wiltshire equations and choosing the reward function (which is one of the main stakes)
- Code the "step" function to actualize the the environment's state after each action
- Choose and implement a sophisticated algorithm (Deep Q Learning, see DDPG and PPO in references) to find the optimal control policy
- Optional : code a visualization of each episode, to see how the agent learns
- Train the model and potentially change the reward or the optimization algorithm until reaching a satisfying result

References

- **Reinforcment Learning, an introduction** , Richard S. Sutton, Andrew G. Barto, 2020
- **Proximal Policy Optimization Algorithms**, John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, 2017, https://arxiv.org/abs/1707.06347
- **Continuous control with deep reinforcment learning**, Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, 2010, https://arxiv.org/abs/1707.06347