AI Models for Predictive Control with Partial Observations for Space Based Applications

Thomas Georges, Manon Lagarde, Lucas Li

Friday December 13, 2024

The massive drop of the cost to launch an object into orbit has paved the way for massive constellations made of thousands of satellites for applications such as communications, earth observation, space logistics for both civilian and defense customers.

The near exponential increase of objects in orbit has made relevant multiple topics, including the need for generate precise and adaptative trajectories for multiple use-cases. One of the most important application is to ensure that a chaser satellite converges and ultimately docks to a target spacecraft to perform on orbit servicing (the setvices can be refueling, upgrading, data transfer, relocating to another orbit etc...). This problem is called the Autonomous Rendezvous Proximity Operations and Docking (ARPOD).

Usual techniques involve feedback control methods, aptly named "Model Predictive Control", which need to split the manoeuvre into three distinct phases, as shown in [**?**]. Some recent studies have been performed to see if trajectories generated through Reinforcement Learning could help yield better results [**?**]. Before diving deeper into the methods employed, we need to define the state space and the dynamics involved

Spacecraft Relative Motion Dynamics

We have two bodies : one target spacecraft and one chaser spacecraft. The dynamics between them are non-linear, but when the chaser is relatively close to the target, they can be approximated by a linear form. All coordinated below will be in what we call the Local-Vertical/Local-Horizontal (LVLH) frame with reference fixed on the center of gravity of the target spacecraft. The states

$$x := \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\top \in \mathbb{R}^6$$

include the spacecraft's position and velocity. The state x denotes the position in the radial direction (from Earth), also known as the cross-track direction, y denotes the position in the in-track direction, and z denotes the position in the cross-track direction that completes the right-hand coordinate system with x and y.

We consider six continuously variable thrusters that provide thrust in the positive and negative direction of each dimension and define the thrust actuation control inputs as

$$u := \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^\top \in \mathbb{R}^3.$$

R denotes the orbit radius of the target spacecraft, $n = \sqrt{\mu/R^3}$ is the angular speed, or "mean motion," of the target through its orbit, and μ is the Earth's gravitational constant.

Here are the equations that define the dynamics of this problem (also called the Hill-CLohessy Wiltshire equations)

$$\begin{aligned} \ddot{x} - 3n^2 x - 2n\dot{y} &= u_x, \\ \ddot{y} + 2n\dot{x} &= u_y, \\ \ddot{z} + n^2 z &= u_z. \end{aligned} \tag{2}$$

We assume that the chaser spacecraft has six thrusters capable of continuously variable thrust, so the three control inputs are continuously variable up to a maximum thrust of $\bar{u}$. Thus, u_x, u_y, and u_z can all take values within the set $[-\bar{u}, \bar{u}]$. Our action space is thus all the thrust vectors that we can input to the agent.

In the environment, the agent's next observation is calculated using the dynamics (**??**) and the current control action a_t and current observation s_t. Every new observation is used to give reward feedback according to a reward function. The agent learns to map state observations to control inputs that transition to states that maximize the reward. The force F_t that each thruster applies is bounded within $[-\bar{F}, \bar{F}]$, and the mass of the chaser m is assumed to stay constant. Thus, the control input u_t is constrained such that $u_t = F_t/m \in [-\bar{F}/m, \bar{F}/m]$.

Because we want the chaser spacecraft to stay within a line-of-sight (LoS) region, we incorporate that constraint into the reward function. We denote ρ_t as the position vector of the chaser relative to the target at time t, β as a vector that points 800 meters out from the target's docking port in the y direction, and φ denotes the angle of the LoS cone. Then the LoS region is given as

$$\text{LoS} = \left\{ \rho_t : \frac{\rho_t \cdot \beta}{\|\rho_t\| \|\beta\|} \geq \cos\left(\frac{\varphi}{2}\right) \wedge [y \geq y_d] \right\},$$

where y_d is the y-position of the docking port.

Given the target/docking state x_d, we define $d_t = x_d - x_t$ as the element-wise difference between the docking state and the current state. The reward generally captures the loss and is a function of the chaser spacecraft's state relative to the target/docking state and the LoS region. Thus, we define the reward as

$$r_t = \begin{cases} d_t^\top Q_x d_t + u_t^\top Q_u u_t - (\rho_1 - \|d_t\|_2)^2 & \text{if not LoS and } \|d_t\|_2 > \rho_1, \\ d_t^\top Q_x d_t + u_t^\top Q_u u_t - (\rho_2 - \|d_t\|_2)^2 & \text{if not LoS and } \|d_t\|_2 < \rho_2, \\ d_t^\top Q_x d_t + u_t^\top Q_u u_t + (\rho_2 - \|d_t\|_2)^2 & \text{if LoS and } \|d_t\|_2 < \rho_2, \\ d_t^\top Q_x d_t + u_t^\top Q_u u_t + (\rho_2 - \|d_t\|_2)^2 + (20\rho_3 - \|d_t\|_2)^2 & \text{if LoS and } \|d_t\|_2 < \rho_3, \end{cases}$$

where Q_x and Q_u are diagonal weighting matrices and ρ_1, ρ_2, and ρ_3 are scalars that define the conditions for changing the weights of the reward function.

Potential approach for a solution

The agent's dynamics are solved using PPO, a policy gradient method. This algorithm is particularly adapted to our problem because unlike DRQN it is built to handle a continuous action space, by parameterizing the policy with a distribution (usually Gaussian) over the action space.

In the studies being done, the chaser is assumed to have full observation over the state, but in real conditions, it relies on a set of sensors which gives incomplete data. Therefore, our goal would be to adapt the problem by making it into a POMDP.

We plan to do that by following the procedure from [**?**] in which beliefs vectors and state trqnsition matrixes are inferred using Monte Carlo Markov Chain (MCMC) of a Hidden Markov Models.

Then, we will adapt the existing PPO algorithm to make it work on POMDP, using previous works of [**?**], or investigating new methods to solve POMDP like Cross-Entropy Search [**?**], Deep Recurrent Belief Propagation Networks [**?**] or other actor-critic methods [**?**]