Bayesian Reinforcement Learning for Autonomous Rendezvous and Proximity Operations

Anonymous Author(s)
Affiliation
Address
email

Abstract

To pursue more advanced missions in space, autonomous rendezvous and proximity operations are necessary. Classic methods such as artificial potential function and model predictive control suffer from trapping in local minima, model misspecification and expensive computational burden which prohibit the real world deployment. In this paper, we explore the possibility of reinforcement learning for RPO missions. We introduce an convex optimization subroutine to compute the value function in the deep Bayesian quadrature policy optimization algorithm based on the Lyapunov's method in control theory. The proposed algorithm has been experimentally evaluated on a spacecraft air-bearing test bed and shows impressive and promising performance. The appendix and videos can be found in an anonymous Github repository `https://github.com/brlrpo`.

1 Introduction

Autonomous rendezvous and proximity operations (RPOs) is one of the critical technologies for pursuing a variety of future space missions [17]. There are many different theoretical approaches to solve this problem, such as PD or H_∞ controller [19, 16], artificial potential function(APF) [18, 13], model predictive control (MPC) [9, 1, 25], rapidly exploring random tree [29] and so on. Traditionally, the solution methods for finding optimal trajectories are too computationally burdensome to be performed on board the spacecraft.

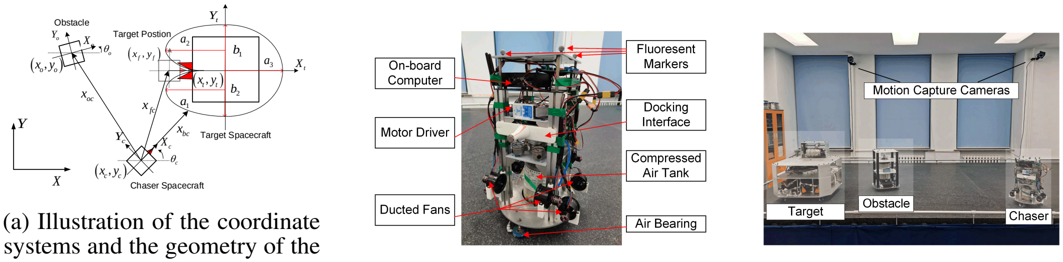
Compared to optimization methods, the computational footprint of analytical methods is small, which meets real-time on-board execution[18]. However, analytical methods do not guarantee overall performance since no cost function is explicitly minimized, and APF-based methods suffer the existence of local minima, which may trap a spacecraft and prevent it from completing the maneuver.

Regardless of the computational burden which fast dynamics systems may solve in the future[24], optimization methods such as MPC might incur significant performance degradation in the presence of model uncertainty. Due to the flexibility of components and dynamic characteristics of the actuator, the accurate dynamic model is extremely difficult to build. This means any proposed method must be evaluated on the air bearing test bed [28] (also see Figure. 1b and 1c).

Many learning-based methods have been widely used in autonomous vehicles, and purpose build processors could improve the computational burden. Typically, RL agent is trained in the simulation and then transferred into the real robot. However, the reality gap would influence the performance of the RL agent [11]. And among these algorithms, the Bayesian version, i.e., Bayesian RL [3, 6, 2, 5, 22], works better in simulated RPO missions than other RL algorithms. Unfortunately, the sim-to-real transfer is unsuccessful.

We substitute the subroutine of the computing value function in the Gaussian process temporal difference procedure [5] with a constrained convex optimization program based on Lyapunov's method in control theory. The algorithm is implemented and modified on the recent deep Bayesian quadrature policy optimization algorithm (BQ) [22], for which the value function is approximated by

Submitted to NeurIPS 2021 4th Robot Learning Workshop: Self-Supervised and Lifelong Learning. Do not distribute.



(a) Illustration of the coordinate systems and the geometry of the target

(b) spacecraft simulator

(c) air bearing test bed

Figure 1: Coordinate configurations and air bearing test bed

a deep Gaussian process [27, 4]. These constraints are related to the energy decreasing condition of Lyapunov stability in control theory. The proposed algorithm has been experimentally evaluated on a spacecraft air-bearing test bed and shows impressive and promising performance.

2 Problem Formulation

In this section, we introduce the proximity maneuver and docking problem and rewrite it into the Markov decision process. Then we explain how to design the reward function in detail.

2.1 Simplified System Dynamics

We consider a planar proximity maneuver and docking problem between a Chaser spacecraft and a Target spacecraft without collision with an obstacle similar to the setup in [18]. As shown in Figure.1a, we consider the following four Cartesian coordinate systems: the inertially fixed coordinate system X, Y, Z; the coordinate system X_c, Y_c, Z_c fixed with the chaser spacecraft, the coordinate system X_t, Y_t, Z_t fixed with the target spacecraft and the coordinate system X_o, Y_o, Z_o of the obstacle. The ideal planar dynamics of the spacecrafts are linear, i.e., $\ddot{x} = \frac{f_x}{m}, \ddot{y} = \frac{f_y}{m}, \ddot{\theta} = \frac{\tau}{I_z}$ or can be rewritten as the state space representation $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, where m and I_z are the spacecraft's mass and moment of inertia, f_x and f_y are the control forces in the respective inertial $x+$ and $y+$direction, τ is the control torque axis along the $z+$direction. And the state vector is denoted by $\mathbf{x} = \left[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}\right]^\top \in \mathbb{R}^6$, the control input is $\mathbf{u} = [f_x, f_y, \tau]^\top \in \mathbb{R}^3$, the corresponding state matrix $\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6}$ and the corresponding control matrix $\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{M}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times3}$, where $\mathbf{M} = \mathrm{diag}\,[1/m, 1/m, 1/I_z]$.

2.2 Mission Goals

At first glance, the goal can be trivially described as moving a (linear) robot from point A to B with obstacle avoidance [12]. For spacecraft RPO missions, the goal could be more complicated. Before proceeding, we firstly introduce the quantities of interest, as shown in Figure. 1a, which is consistent with the spacecraft RPO literature [18, 20]. In this paper, the state vector $\boldsymbol{s} = \{\boldsymbol{x}_{fc}, \boldsymbol{x}_{oc}, \dot{\boldsymbol{x}}_c\} \in \mathcal{S}$, $\mathbf{x}_{fc} = \mathbf{x}_f - \mathbf{x}_c$ is the relative difference between the Chaser coordinate vector $\mathbf{x}_c = [x_c, y_c, \theta_c]^T \in \mathbb{R}^3$ and the desired coordinate vector $\mathbf{x}_f$; $\mathbf{x}_{oc} = \mathbf{x}_o - \mathbf{x}_c$ is the relative difference between the Chaser coordinate vector and the center point of the obstacle; and $\dot{\mathbf{x}}_c = \left[\dot{x}_c, \dot{y}_c, \dot{\theta}_c\right]^T \in \mathbb{R}^3$ is the generalized speed vector of the Chaser. $\mathbf{x}_{bc} = \mathbf{x}_b - \mathbf{x}_c$ is the relative position between the Chaser and the Target boundary constraint position $\mathbf{x}_b$. The action vector $\mathbf{u} = \mathbf{f}_c \in \mathcal{U}$, which is the control force and torque of the Chaser via thrusters.

The goal is to minimize R_{tot}, which is defined as the superposition of three functions, R_a, R_b, R_o, i.e., $R_{\text{tot}} = R_a + R_b + R_o$ where $R_a = \frac{k_a}{2}\mathbf{x}_{fc}^T\mathbf{H}_1\mathbf{x}_{fc}$, $R_b = \frac{k_r}{2}\frac{\mathbf{x}_{fc}^T\mathbf{Q}_b\mathbf{x}_{fc}}{\exp\left[\mathbf{x}_{bc}^T\mathbf{P}_b\mathbf{x}_{bc}-1\right]}$, $R_o = \psi \exp\left[-\frac{\mathbf{x}_{oc}^T\mathbf{N}\mathbf{x}_{oc}}{\sigma}\right]$, and $\mathbf{H}_1, \mathbf{H}_2, \mathbf{Q}_b, \mathbf{P}_b, \mathbf{N}$ is positive definite matrices, and $k_a, k_r, \psi \in \mathbb{R}_+$. Such specification is the same as in [18]. When the Chaser approaches to the Target and docks successfully, $\mathbf{x}_{oc} = \mathbf{x}_f - \mathbf{x}_c$ which means the obstacle repulsive potential functions R_o is a constant number $r_o \in \mathbb{R}_+$. Therefore, we can modify R_{tot} into $R_{\text{tot}} = R_a + R_b + R_o - r_o$ which is always nonnegtive and expected to be zero eventually.

The "shape" of these functions is relatively standard in the spacecraft RPO literature [18, 20], e.g., using (by default) artificial potential field method [18]. The functions are chosen by exploring the gradient of a potential field, which is composed of both attractive and repulsive potentials, to derive the necessary control inputs to reach the desired or goal position [15]. The attractive potential R_a, establishes a global minimum in the workspace $\mathbf{x}_f$ at the desired (terminal) configuration. This function is similar to the cost function is typically defined in the context of model predictive control. The repulsive potential, R_o, creates an area of higher potential in portions of the workspace that should be avoided, such as obstacles or exclusion zones [15, 10, 14]. Two different types of keep-out zones, one for obstacles and one for the Target vehicle, are represented using repulsive potential functions. Keep-out zones centered around an obstacle are represented using a Gaussian function [15, 14]. For this application, the repulsive potential shaping matrix is the identity matrix that corresponds to a circular obstacle keep-out zone. The keep-out zone centered around the Target vehicle was selected to be R_b [30].

2.3 Related works

Two approaches are typically used for spacecraft RPO missions, i.e., APF and MPC. A more detailed description can be found in Appendix A. From the experiment, we found APF works pretty well and is robust in both simulated and air bearing test bed experiments (see Figure.2 and 3). However, the APF methods suffer from the existence of (stable) local minima that may trap a spacecraft and prevent it from completing the maneuver [15, 14, 30, 18]. While for MPC, it works well in simulation but can hardly work in air bearing test bed experiments (see Figure.2). This might be due to the model-misspecification, e.g., the flexible appendages, thruster modulation. These effects are in general, extremely difficult to analyze with customized hardware setup and floating conditions. Meanwhile, both approaches are computationally expensive, which might not be appropriate for spacecraft RPO missions [18].

As an alternative, the RPO mission can be also solved using RL algorithms by setting the reward function as R_{tot} (preliminary on RL can be found in Appendix B.1). This can potentially address the above issues, as well as combining with some sim-to-real transfer techniques [31]. Unfortunately, some of the popular open-source algorithms can hardly converge even in simulations, e.g., TRPO, SAC (training details and results can be founding in Appendix B.3). Interestingly, the Bayesian reinforcement learning can easily converge with impressive performance and sample efficiency [3, 6, 2, 5, 22]. Unfortunately, this only works in simulation, not in the air bearing test bed (see Figure.1c). This motivate us to propose a robust modification of Bayesian reinforcement learning algorithm based on Lyapunov stability in control theory.

3 Bayesian Reinforcement Learning

4 Lyapunov-Based Bayesian Reinforcement Learning

4.1 Lyapunov stability in control theory

Our intuition is that we should enforce more constraints over the state(-action) value function to search the policy in a new value function space. In this article, our intuition is to consider the state value function as the Lyapunov function. A Lyapunov function is utilized to provide a stability guarantee. This tool has been used in the control theory for stability analysis and controller design. The Lyapunov function is a class of positive definite functions $L : \mathcal{S} \to \mathbb{R}_+$. The general idea of exploiting the Lyapunov function is to ensure that the difference (or derivative) of the Lyapunov function along the state trajectory is seminegative definite so that the state goes in the direction of the decreasing value of the Lyapunov function and eventually converges to the origin or a sublevel set of the Lyapunov function. A Lyapunov-based and data-driven method is proposed for analyzing the stability of the system. A more thorough introduction can be found in Appendix C.1 And a data-based stability theorem is given in Appendix C.2 which will be used in the algorithm below.

4.2 Lyapunov-based deep Bayesian quadrature policy optimization algorithm (LBQ)

Now we will translate the theoretical result into a practical Bayesian RL algorithm. Based on the recent deep Bayesian quadrature policy optimization algorithm [23], we substitute the subroutine of computing the value function Q (denoted as Q^L) in the Monte-Carlo Gaussian process (MC-GPTD) with the following convex optimization program:

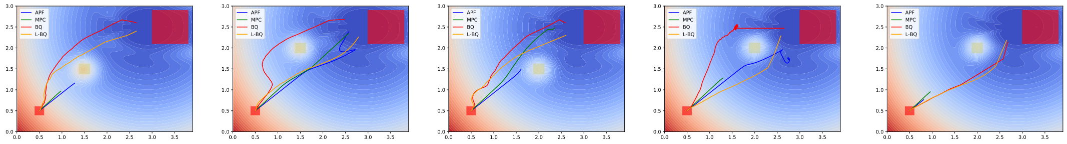
$$\begin{aligned} \min \quad & ||\boldsymbol{R} - \boldsymbol{H}\boldsymbol{Q}^L||_2 \\ \text{s.t} \quad & Q^L(z_{t+1}) - Q^L(z_t) \leq -R_{\text{tot}}(z_t), \\ & Q^L(z_t) \geq 0, \quad \forall t = 0, \dots, T-1 \end{aligned} \tag{1}$$

where $z_t = (s_t, a_t)$ state-action pairs at time instance t. $\boldsymbol{R} = (R(z_0), \dots, R(z_{T-1}))^\top, \boldsymbol{Q}^L = (Q^L(z_0), \dots, Q^L(z_{T-1}))^\top$ and $\boldsymbol{H} = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 & 0 \\ 0 & 1 & -\gamma & \dots & 0 & 0 \\ \dots & \dots & \dots & & \dots & 1 \end{bmatrix} \in \mathbb{R}^{T\times(T+1)}$. More details on the algorithm analysis can be found in Appendix D.

5 Experiment Result

In this section, we respectively evaluated APF, MPC, BQ, and LBQ methods in numeral simulation and the physical simulator, and the results are, respectively, shown in Figure. 2 and Figure. 3. We found that all methods show great performances in numeral simulation, though we found the existence of (stable) local minima that prevents the Chaser from completing the maneuver, which is mentioned in [18]. And the MPC method sometimes could not find the optimized solution.

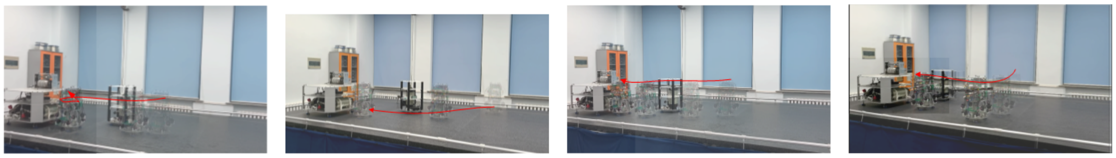
In simulations, the test scenarios are depicted in Figure.2 evaluate the ability of a guidance method to handle various constraints while producing safe trajectories. In cases 1-4, the obstacle positions, respectively, are (1.5, 1.5)m,(1.5, 2)m,(2, 1.5)m and (2, 2)m, while the initial obstacle position is (2, 2)m with a fixed velocity is (-0.1, -0.1)m/s which stresses the replanning capability of the guidance method. Specific parameters of the test scenarios (i.e., obstacle size/location) as well as the real experiment, shown in Table.2 in the Appendix.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5

Figure 2: Simulated experiments

In the air bearing test bed experiment (see Figure. 3), we evaluate the above methods. We find that the MPC method could not work with infeasible convex optimization procedures. For BQ and LBQ methods, the BQ method hardly shows the tendency to avoid the obstacle while the LBQ could achieve the task without collision when the position of the Chaser. Moreover, in the APF method, we find that the Chaser stays at the existence of local minima, which is consistent in the simulation, and the Chaser simulator would collide with the obstacle when the initial position is far from the target position.



(a) APF method for static obstacle

(b) APF method for dynamic obstacle

(c) LBQ-RL method for static obstacle

(d) LBQ-RL method for dynamic obstacle

Figure 3: Air bearing test bed experiments

References

[1] Stefano Di Cairano, Hyeoungjun Park, and Ilya Kolmanovsky. Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering. International Journal of Robust and Nonlinear Control, 22(12):1398–1427, 2012.

[2] Yaakov Engel and Mohammad Ghavamzadeh. Bayesian policy gradient algorithms. Advances in neural information processing systems, 19:457, 2007.

[3] Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In Proceedings of the 22nd international conference on Machine learning, pages 201–208, 2005.

[4] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. arXiv preprint arXiv:1809.11165, 2018.

[5] Mohammad Ghavamzadeh, Yaakov Engel, and Michal Valko. Bayesian policy gradient and actor-critic algorithms. The Journal of Machine Learning Research, 17(1):2319–2371, 2016.

[6] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. arXiv preprint arXiv:1609.04436, 2016.

[7] Wassim M Haddad et al. Finite-time partial stability and stabilization, and optimal feedback control. Journal of the Franklin Institute, 352(6):2329–2357, 2015.

[8] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. IEEE Robotics and Automation Letters, 5(4):6217–6224, 2020.

[9] E. N. Hartley, P. A. Trodden, A. G. Richards, and J. M. Maciejowski. Model predictive control system design and implementation for spacecraft rendezvous. Control Engineering Practice, 20(7):695–713, 2012.

[10] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In Autonomous robot vehicles, pages 396–404. Springer, 1986.

[11] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, pages 119–126, 2010.

[12] Steven M LaValle. Planning algorithms. Cambridge university press, 2006.

[13] I. Lopez and C. R. Mcinnes. Autonomous rendezvous using artificial potential function guidance. Journal of Guidance Control and Dynamics, 18(2):237–241, 1995.

[14] Ismael Lopez and Colin R Mclnnes. Autonomous rendezvous using artificial potential function guidance. Journal of Guidance, Control, and Dynamics, 18(2):237–241, 1995.

[15] Josue Munoz, George Boyarko, and Norman Fitz-Coy. Rapid path-planning options for autonomous proximity operations of spacecraft. In AIAA/AAS Astrodynamics Specialist Conference, page 7667, 2010.

[16] Didier Pinard, Stéphane Reynaud, Patrick Delpy, and Stein E. Strandmoe. Accurate and autonomous navigation for the atv. Aerospace Science and Technology, 11(6):490–498, 2007.

[17] M. B. Quadrelli, L. J. Wood, J. E. Riedel, M. C. Mchenry, and J. A. Cutts. Guidance, navigation, and control technology assessment for future planetary science missions. Journal of Guidance, Control, and Dynamics, 38(7):1165–1186, 2015.

[18] Richard, Zappulla, Hyeongjun, Park, Josep, Virgili-Llop, Marcello, and Romano. Real time autonomous spacecraft proximity maneuvers and docking using an adaptive artificial potential field approach. Control Systems Technology, IEEE Transactions on, 27(6):2598–2605, 2018.

[19] Marcello Romano, David A. Friedman, and Tracy J. Shay. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. Journal of Spacecraft and Rockets, 44(1):164–173, 2007.

[20] Tomasz Rybus. Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. Progress in Aerospace Sciences, 101:31–48, 2018.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pages 1889–1897. PMLR, 2015.

[22] Akella Ravi Tej, Kamyar Azizzadenesheli, Mohammad Ghavamzadeh, Anima Anandkumar, and Yisong Yue. Deep bayesian quadrature policy optimization. arXiv preprint arXiv:2006.15637, 2020.

[23] Akella Ravi Tej, Kamyar Azizzadenesheli, Mohammad Ghavamzadeh, Anima Anandkumar, and Yisong Yue. Deep bayesian quadrature policy optimization. arXiv preprint arXiv:2006.15637, 2020.

[24] R. M. Vignali, F. Borghesan, L. Piroddi, M. Strelec, and M. Prandini. Energy management of a building cooling system with thermal storage: An approximate dynamic programming solution. IEEE Transactions on Automation Science and Engineering, 2017.

[25] Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, Richard Zappulla, and Marcello Romano. Experimental evaluation of model predictive control and inverse dynamics control for spacecraft proximity and docking maneuvers. CEAS Space Journal, 10(1):37–49, 2018.

[26] Vladimir I Vorotnikov. Partial stability and control: The state-of-the-art and development prospects. Automation and Remote Control, 66(4):511–561, 2005.

[27] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In International Conference on Machine Learning, pages 1775–1784. PMLR, 2015.

[28] Ii Zappulla, Richard, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. Journal of Spacecraft and Rockets, 54(4):825–839, 2017.

[29] R Zappulla, Josep Virgili-Llop, and Marcello Romano. Near-optimal real-time spacecraft guidance and control using harmonic potential functions and a modified rrt. In 27th AAS/AIAA Space Flight Mechanics Meeting, San Antonio, TX, 2017.

[30] Richard II Zappulla. Experimental evaluation methodology for spacecraft proximity maneuvers in a dynamic environment. PhD thesis, 2017.

[31] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pages 737–744. IEEE, 2020.

A Related Works

A.1 Artificial potential field

The APF method uses the gradient of a potential field to derive the necessary control to reach the desired position. The total potential function ϕ_{tot} is defined as the superposition of the attractive and all repulsive potential functions

$$\phi_{\text{tot}} = R_a + R_b + R_o \tag{2}$$

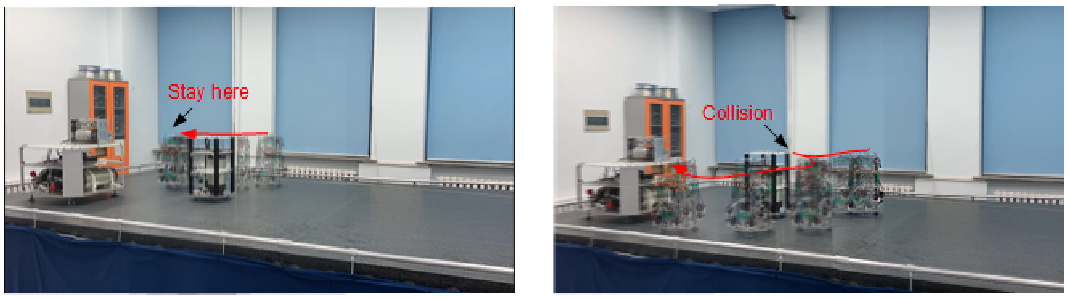
The continuous feedback control law is chosen and defined as[18]

$$\mathbf{u}\left(\mathbf{x}_c, \mathbf{x}_f, \dot{\mathbf{x}}_c\right) = -\mathbf{B}^{-1}\mathbf{K}_a\left(\dot{\mathbf{x}}_c + \nabla_x \phi_{\text{tot}}\right) \tag{3}$$

where $\mathbf{B} = \operatorname{diag}\left[1/m, 1/m, 1/I_z\right]$, $\mathbf{K}_a \in \mathbb{R}^{3\times3}$ is a positive gain matrix, and $\nabla_x \phi_{\text{tot}}$ is given as

$$\begin{aligned}\nabla_x \phi_{\text{tot}} =& k_a \mathbf{H}_1 \mathbf{x}_{fc} - \frac{2\psi}{\sigma} \exp\left[-\frac{\mathbf{x}_{oc}^T \mathbf{N} \mathbf{x}_{oc}}{\sigma}\right] \mathbf{N} \mathbf{x}_{oc} \\ &+ k_r \exp\left[1 - \mathbf{x}_{bc}^T \mathbf{P}_b \mathbf{x}_{bc}\right]\left(\mathbf{Q}_b \mathbf{x}_{fc} - \left(\mathbf{x}_{fc}^T \mathbf{Q}_b \mathbf{x}_{fc}\right) \mathbf{P}_b \mathbf{x}_{bc}\right)\end{aligned} \tag{4}$$

The results of the APF method in simulation and real-world are shown in Figure.2 and Figure.3. Because of the nature shortcoming of the APF method, the Chaser sometimes stays at the local minima and collides with an obstacle with a fast speed as shown in Figure.4. However, the APF method shows great performances in both numeral simulation and experiment, especially in the final docking process.

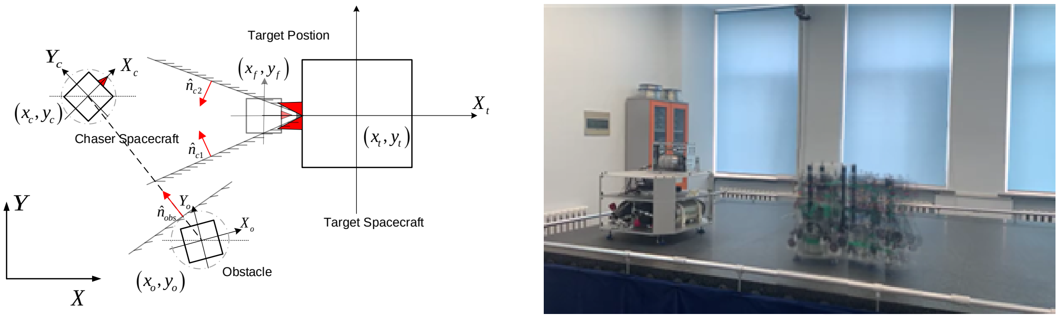


(a) Stay at the local minima

(b) Collides with an obstacle

Figure 4: Failure examples of the APF method

A.2 Model predictive control



(a) Illustration of the geometry of the constrains

(b) Experiment result of the MPC method

Figure 5: Geometry of the constrains and experiment result

The RPOs problem can be seen as a constrained optimization problem that can be solved by MPC. To ensure safety during the maneuver and docking process, we set the obstacle avoidance constraint and docking cone constraints by constructing hyperplanes. When these constraints are activated,

the Chaser is forced to stay within the edges of the cone and collision-free maneuver with obstacle until docking is achieved, as shown in Figure.5a. The MPC problem in discrete form, which only considers the translational motion with the horizon length N, is formed as follows:
Minimize

$$J=\sum_{i=0}^{N}\left(\overline{\mathbf{s}}_{c}(k+i)-\overline{\mathbf{s}}_{f}\right)^{\mathrm{T}}\mathbf{Q}\left(\overline{\mathbf{s}}_{c}(k+i)-\overline{\mathbf{s}}_{f}\right) \tag{5}$$

Subject to

$$\begin{aligned}
&\overline{\mathbf{s}}_{c}(k+1)=\mathbf{A}_{d}\overline{\mathbf{s}}_{c}(k)+\mathbf{B}_{d}\overline{\mathbf{u}}(k)\\
&|f_{x}(k)|\leq u_{\max}\\
&|f_{y}(k)|\leq u_{\max}\\
&\hat{\mathbf{n}}_{\text{obs}}\cdot\overline{\mathbf{x}}_{c}(k)\geq\hat{\mathbf{n}}_{\text{obs}}\cdot\overline{\mathbf{x}}_{o}+d\\
&\hat{\mathbf{n}}_{c1}\cdot\overline{\mathbf{x}}_{c}(k)\geq\hat{\mathbf{n}}_{c1}\cdot\overline{\mathbf{x}}_{f}\\
&\hat{\mathbf{n}}_{c2}\cdot\overline{\mathbf{x}}_{c}(k)\leq\hat{\mathbf{n}}_{c2}\cdot\overline{\mathbf{x}}_{f}
\end{aligned} \tag{6}$$

where $\mathbf{A}_d$, $\mathbf{B}_d \in \mathbb{R}^{4\times 4}$ are the discrete state and control matrices, which can be derived from the translational part of the continuous dynamics in Section2.1 when using a sampling times T_s, and $\overline{\mathbf{s}}_c=[x_c, y_c, \dot{x}_c, \dot{y}_c]^\top$, $\overline{\mathbf{s}}_f=[x_f, y_f, 0, 0]^\top \in \mathbb{R}^4$ are the state vector of the Chaser and the desired state vector. The matrix $\mathbf{Q}\in\mathbb{R}^{4\times 4}$ is the cost function weight on the state, and d is the minimum safe distance between the Chaser and obstacle. The $\hat{\mathbf{n}}_{obs}$, $\hat{\mathbf{n}}_{c1}$, $\hat{\mathbf{n}}_{c2}$ define the normal vectors of the hyperplanes. The $\overline{\mathbf{x}}_c$, $\overline{\mathbf{x}}_o$, $\overline{\mathbf{x}}_f$, $\overline{\mathbf{u}}\in\mathbb{R}^2$ respectively represent the translational part of $\mathbf{x}_c$, $\mathbf{x}_o$, $\mathbf{x}_f$, $\mathbf{u}$, such as $\overline{\mathbf{u}}=[f_x, f_y]^\top$. As mentioned before, only the translational motion is included in the MPC formulation, while the rotational motion is controlled by PID method. The MPC method shows great performance in the numeral simulation shown in Figure.2, but sometimes the system crashes where MPC method couldn't find the optimized solution with $N = 100$. Because of the computing power of our system, MPC method does not work in the air bearing test bed. The Chaser simulator goes to the desired target at about the first 5s and then the system crashes as shown in Figure.5.

B Bayesian Reinforcement Learning

B.1 Preliminaries

Once the Chaser collides with the Target or obstacle, the training episode ends and R is set to a large value as punishment($R = 5000$). While the Chaser reaches the desired range which meets the docking requirement, the training episode ends and R is set to be zero at that step.

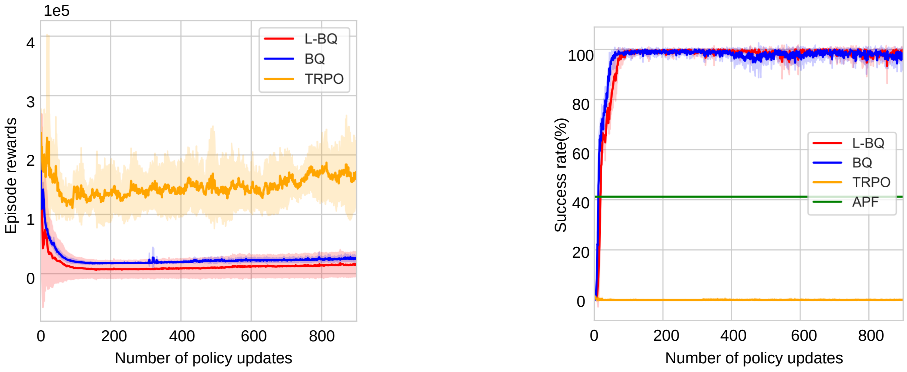
B.2 BQ policy gradient integral

The main idea is that converts numerical integration into a Bayesian inference problem. The first step is to formulate a prior stochastic model over the integrand, which could be a Gaussian process(GP) prior to Q_{π_θ} function. Then GP prior on Q_{π_θ} is conditioned(Bayes rule) on the sample data $\mathcal{D}=\{z_i\}_{i=1}^n$ to obtain the posterior moments $\mathbb{E}\left[Q_{\pi_\theta}(z)\mid\mathcal{D}\right]$ and $\mathcal{C}_Q\left(z_1,z_2\right)=\operatorname{Cov}\left[Q_{\pi_\theta}\left(z_1\right),Q_{\pi_\theta}\left(z_2\right)\mid\mathcal{D}\right]$. In turn, the policy gradient estimate also is a Gaussian distribution $\boldsymbol{L}_{\boldsymbol{\theta}}^{BQ}$ and $\boldsymbol{C}_{\boldsymbol{\theta}}^{BQ}$:

$$\begin{aligned}
\boldsymbol{L}_{\boldsymbol{\theta}}^{BQ}&=\mathbb{E}\left[\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})\mid\mathcal{D}\right]=\int_{\mathcal{Z}}dz\rho^{\pi_\theta}(z)\boldsymbol{u}(z)\mathbb{E}\left[Q_{\pi_{\boldsymbol{\theta}}}(z)\mid\mathcal{D}\right]\\
\boldsymbol{C}_{\boldsymbol{\theta}}^{BQ}&=\operatorname{Cov}\left[\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})\mid\mathcal{D}\right]\\
&=\int_{\mathcal{Z}^2}dz_1dz_2\rho^{\pi_\theta}\left(z_1\right)\rho^{\pi_\theta}\left(z_2\right)\boldsymbol{u}\left(z_1\right)\mathcal{C}_Q\left(z_1,z_2\right)\boldsymbol{u}\left(z_2\right)^\top
\end{aligned} \tag{7}$$

where $\boldsymbol{u}(z)=\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a\mid s)$ is the score function and ρ^{π_θ} is the discounted state-action visitation frequency. The integral in Eq.7 can be solved with a closed-form solution when the GP kernel k is the composition of a state kernel k_s and the Fisher kernel k_f:

$$\begin{gathered}
k\left(z_1,z_2\right)=c_1k_s\left(s_1,s_2\right)+c_2k_f\left(z_1,z_2\right)\\
\text{with } k_f\left(z_1,z_2\right)=\boldsymbol{u}\left(z_1\right)^\top\boldsymbol{G}^{-1}\boldsymbol{u}\left(z_2\right)
\end{gathered} \tag{8}$$



(a) Average episode rewards

(b) Average success rate

Figure 6: The result of RL algorithms during training

where c_1, c_2 are hyperparameters and G is the Fisher information matrix of policy network hyperparameters. Then, the policy gradient is

$$\begin{aligned} \boldsymbol{L}_{\boldsymbol{\theta}}^{BQ} &= c_2 \boldsymbol{U}\left(\boldsymbol{K}+\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{Q} \\ \boldsymbol{C}_{\boldsymbol{\theta}}^{BQ} &= c_2\boldsymbol{G} - c_2^2\boldsymbol{U}\left(\boldsymbol{K}+\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{U}^\top \end{aligned} \tag{9}$$

where $\boldsymbol{U} = [\boldsymbol{u}(z_1), \ldots, \boldsymbol{u}(z_n)]$ is a $\Theta \times n$ dimensional matrix($\Theta$is the number of policy parameters), and $\boldsymbol{Q} = [Q_{\pi_\theta}(z_1), \ldots, Q_{\pi_\theta}(z_n)]$ is an n dimensional vector. $\boldsymbol{Q}$ could be computed using TD(1) method or other methods.

B.3 Reinforcement learning result

We use on-policy RL algorithms, such as TRPO, BQ, LBQ, to solve the RPO mission. Each algorithm is trained for the same amount of policy updates (900 times) over 5 random seeds with optimized hyperparameters which are listed in Table.1. As shown in Figure.6a, we find that BQ and LBQ converge quickly at about 100 iterations and the final episode reward of LBQ is slightly lower than the BQ, while TRPO has not converged eventually at 900 iterations. As shown in Figure.6b, the average success rates of BQ and LBQ both exceed 41.2%, which is the average success rate of APF in 1000 episodes, after about 30 iterations. The average success rate of BQ reaches around 100% after around 90 iterations, while LBQ reaches 100% at around 100 iterations.

Moreover, we also use SAC which is an off-policy RL algorithm, and get a similar result to TRPO. The reasons why some RL algorithms hardly converge at 900 iterations may be as follows: Firstly, the amount of training data is not enough and the strategy would converge after more policy updates. Secondly, the reward function in the RPO mission is special which makes the traditional RL algorithms hardly to solve, while the BQ-based methods are good at it. In this paper, we mainly present the results of RL algorithms, and the real reason why some RL algorithms hardly work needs to be further discussed.

C Lyapunov Stability in Control Theory

C.1 Stability in Control Theory

In RPO mission, we divide the state $s = \{\mathbf{x}_{fc}, \mathbf{x}_{oc}, \dot{\mathbf{x}}_c\}$, into two vectors, s^1 and s^2, where s^1 is composed of elements $\mathbf{x}_{fc}$ and $\dot{\mathbf{x}_c}$ that are aimed at reach the desired position $\mathbf{x}_f$, while s^2 contains $\mathbf{x}_{oc}$ the relationship with the obstacle. Thus, $c(s_t, a_t) = \mathbb{E}_{P(\cdot|s_t,a_t)}\|s_{t+1}^1\|$.

From a control perspective, stabilization is related to the asymptotic stability of the closed-loop system (or error system) under π, i.e., starting from an initial point, the trajectories of the state always converge to the origin or reference trajectory. Let $c_\pi(s_t) \triangleq \mathbb{E}_{a\sim\pi} c(s_t, a_t)$ denote the cost function under the policy π, the definition of stability studied in this paper is given as follows.

Definition 1 *The stochastic system is said to be stable in mean cost if* $\lim_{t\to\infty}\mathbb{E}_{s_t}c_\pi(s_t)=0$ *holds for any initial condition* $s_0\in\{s_0|c_\pi(s_0)\le b\}$. *If b is arbitrarily large then the stochastic system is globally stable in mean cost.*

The above definition is also equivalent to the partial stability [26, 7] when $c(s_t,a_t) = \mathbb{E}_{P(\cdot|s_t,a_t)}\|s^1_{t+1}\|$. Thus the RPO mission can be collectively summarized as finding a policy π such that the closed-loop system is stable in mean cost according to Definition 1.

C.2 Lyapunov stability guarantee

Our approach is to construct/find a Lyapunov function $L:\mathcal{S}\to\mathbb{R}_+$ of which the difference along the state trajectory is negative definite so that the state goes in the direction of decreasing the value of Lyapunov function and eventually converges to the origin. In the following, we show that without a dynamic model, the Lyapunov function could be constructed through sampling and sufficient conditions for a stochastic system to be stable in mean cost are given[8].

Theorem 1 *The stochastic system is stable in mean cost if there exists a function* $L:\mathcal{S}\to\mathbb{R}_+$ *and positive constants* α_1, *such that*

$$\alpha_1 c_\pi\left(s\right)\le L(s)\le\alpha_2 c_\pi\left(s\right) \tag{10}$$

$$\mathbb{E}_{s\sim\mu_\pi}\left(\mathbb{E}_{s'\sim P_\pi}L(s')-L(s)\right)\le-\alpha_3\mathbb{E}_{s\sim\mu_\pi}c_\pi\left(s\right) \tag{11}$$

where

$$\mu_\pi(s)\triangleq\lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N}P(s_t=s|\rho,\pi,t)$$

is the (infinite) sampling distribution.

Inspired by the Theorem 1, we consider the state value function as the Lyapunov function. Once the agent reaches the desired position where $c_\pi\left(\hat{s}\right)=0$, thus the system is (assumed to be) stable in mean cost by Definition.1. We set the target value $Q^L(\hat{s},\hat{a})=0$ and the target value of the whole trajectory $\boldsymbol{Q}^L$ meets the stability constraints in Equation.11. In practice, the target value $\boldsymbol{Q}^L$ could be solved with the convex optimization program (Equation.1) and then replace the $Q_{\pi\theta}(s_t,a_t)$ with $Q^L_{\pi\theta}(s_t,a_t)$ in Section.B.2, where the Lyapunov-based value function is modeled by the Gaussian process.

D Lyapunov Bayesian Reinforcement Learning

Lyapunov Bayesian reinforcement learning uses the proposed Lyapunov-based deep Bayesian quadrature policy optimization algorithm to update its policy gradient under a reinforcement learning algorithm. Following the architecture shown in Figure.7, the actor neural network takes the observed state as input and estimates a normal distribution in the output action-space as output while the critic function is modeled by the Gaussian process computes the value function of the observed state. State feature extractor is another neural network, which could be seen as a part of actor neural network and critic function, to reduce the dimensions of the observed state. The value function computes the n dimensional vector $\boldsymbol{Q}^L$ for the critic function with n samples to update kernel parameters in critic function.

It is highlighted in LBQ policy gradient that we compute $\boldsymbol{Q}^L$ by the value function with a convex optimization program (Equation.1). Then, the kernel parameters in the critic function is updated and the policy gradient estimate $\boldsymbol{L}^{BQ}_{\boldsymbol{\theta}}$ and $\boldsymbol{C}^{BQ}_{\boldsymbol{\theta}}$ could be computed by BQ policy gradient integral(Equation.7). For TRPO algorithm[21], the policy is updated with

$$\boldsymbol{\theta}^{(n+1)}=\boldsymbol{\theta}^{(n)}+\alpha^j\sqrt{\frac{2\delta}{\boldsymbol{L}^{BQ^T}_{\boldsymbol{\theta}}\boldsymbol{G}^{-1}\boldsymbol{L}^{BQ}_{\boldsymbol{\theta}}}}\boldsymbol{G}^{-1}\boldsymbol{L}^{BQ}_{\boldsymbol{\theta}} \tag{12}$$

where $\alpha\in(0,1)$ is the backtracking coefficient, and j is the smallest nonnegative integer such that $\pi_{\theta_{k+1}}$ satisfies the KL constraint and produces a positive surrogate advantage. The policy gradient in Lyapunov Bayesian reinforcement learning is presented in Algorithm.1.

Based on the architecture shown in Figure.7, the actor neural network comprises of a 3-layered fully-connected MLP with 128 hidden units, and a tanh nonlinearity in each layer, and the state

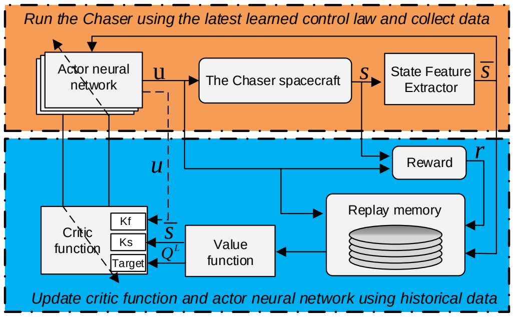


Figure 7: Overview of the Lyapunav Bayesian reinforcement learning

Table 1: Hyperparameter setting in RL

Hyperparameter	Value
Batch size	20000
Discount factor γ	0.98
GAE coefficient τ	0.97
Trust region constrain/step size	0.01
c_1	1
c_2	5×10^{-5}
GP's noise variance σ^2	10^{-4}

feature extractor is another MLP with hidden dimensions 64, 48, and 10, and tanh non-linearity. The critic function models the generalized advantage function using the samples $\boldsymbol{Q}^L$, a fisher kernel and a deep RBF kernel[27]. The critic function is a deep Gaussian process based on GPyTorch[4].

Algorithm 1 LBQ-PG Estimator Subroutine

```
$\mathbf{LBQ-PG}(\boldsymbol{\theta}, n)$
    $\boldsymbol{\theta}$:policy parameters
    $n$:sample size for PG estimation
Collect $n$ state-cation pairs from running the policy $\pi_{\boldsymbol{\theta}}$ in the environment.
Compute value function $Q^L$ for these $n$ samples with a convex optimization (Equation.1).
Update kernel parameters in Critic function described in Section.B.2.
Compute policy gradient estimation $\boldsymbol{L}_{\boldsymbol{\theta}}^{BQ}$ (Equation.9)
Update policy parameters $\boldsymbol{\theta}^{new}$ (Equation.12)
return $\boldsymbol{\theta}^{new}$
```

E Experiment Setup

Given the close proximity and short maneuver duration, an air bearing test bed provides an acceptable approximation of the dynamics, where it is frictionless on a plane[18]. We apply our proposed method in an air bearing test bed, shown in Figure.1b, which includes a Target simulator, a Chaser simulator, an obstacle, and Optitrack motion capture system of which tracking accuracy lower than 1 mm. The specific parameters of the system are as shown in Table.2. The Chaser simulator is a designed autonomous vehicle capable of floatation via air bearings and actuated by a set of eight ducted fans instead of thrusters to provide longer-lasting force and torque. The Chaser simulator is provided with an on-board computer running a real-time operating system(the Intel Core processor N2920, 4g internal storage), fluoresent markers for attitude and position measurement, and a compressed air tank which feeds the air bearings.

The potential parameters are used in both APF-based methods and reward function in RL methods. The parameters in the experiment are shown in Table.3. Because of the gap between the simulation and the real world, the controller which performs well in simulation could not directly be used in the real world. To improve the above problem, We multiply the control command $\mathbf{f}_c$ by a factor l as

Table 2: Specific parameters of the system

Parameter	Value
Chaser Mass, m	$9.6\mathrm{kg}$
Chaser Inertia, I_z	$0.23\mathrm{kgm}^2$
Chaser Control Force, f_{max}	$1\mathrm{N}$
Chaser Control Torque, τ_{max}	$0.2\mathrm{Nm}$
Chaser Width	$0.2\mathrm{m}$
Target Width	$0.8\mathrm{m}$
Obstacle Width	$0.24\mathrm{m}$

Table 3: Specific parameters of the system

Parameter	Value
Sample Time	$0.05\mathrm{s}$
(a_1, a_2, a_3)	$(0.6, 0.6, 0.8)\mathrm{m}$
(b_1, b_2)	$(0.6, 0.6)\mathrm{m}$
k_a, k_r	1.0
$\mathbf{Q}_a$	$\mathrm{diag}([0.025, 0.025, 0.075])\mathrm{s}^{-2}$
$\mathbf{Q}_b$	$\mathrm{diag}([0.0125, 0.0125, 0.175])\mathrm{s}^{-2}$
$\mathbf{P}_b$	$\mathrm{diag}([20, 20, 0])\mathrm{s}^{-2}$
$\mathbf{N}$	$\mathbf{I}_{3\times3}$
σ	$0.125r^2_{\mathrm{obs,i}}\ \mathrm{m}^2$
ψ	0.2

the true control command $\mathbf{f}_{Tc}$ to the Chaser. In the experiment, when the Chaser reaches the desired range, where the distance to the desired position is less than 0.3m, we adopt the APD-based or PD method to dock the Targe.