On-Orbit Optimal Kinodynamic Planning for Low-Thrust Trajectory Maneuvers

Ibrahima Sory Sow, Juan Alvarez-Padilla, Fausto Vega, Nayana Suvarna

Abstract

Designing low-thrust trajectories for spacecraft maneuvers involves significant challenges due to the highly nonlinear dynamics, limited control authority, and extensive duration of trajectories spanning weeks or months. To address this problem, we leverage RRT*, an optimal kinodynamic sampling-based planner, and incorporate a heuristic based on the Linear Quadratic Regulator (LQR) for node extension and distance computations. This heuristic employs the LQR cost-to-go metric to produce paths that are locally optimal and dynamically feasible according to the LQR's quadratic cost, leading to a more effective search within the state space. We demonstrate our algorithm in the case of spacecraft proximity operations with obstacles for varying values of penalty on the control action. Additionally, we assess the actual thrust usage through a custom Time-Varying LQR (TVLQR) to track the generated trajectory, demonstrating the effectiveness of our approach. The project repository is available on the following link: https://github.com/Ibrassow/optimal-sampling-low-thrust

1 Introduction

Space exploration has long captivated humanity, driven by goals such as gathering scientific data about the universe, searching for extraterrestrial life, and establishing human or robotic outposts beyond Earth. Designing spacecraft for these missions presents unique challenges and constraints. A key concern is the efficiency of the spacecraft propulsion system, which not only dictates the range and longevity of the mission but also influences the design and capabilities of the spacecraft. Since fuel resupply is not yet available, optimizing fuel consumption is vital for maneuvering, maintaining orbits, and executing mission-critical tasks.

Traditional propulsion systems use chemical propellants, generating a high amount of thrust over a short burning time, leading designers to consider impulse-based techniques for designing orbital maneuvers [1]. However, this requires carrying a substantial amount of fuel, increasing launch mass and cost. Furthermore, their low-efficiency limits mission flexibility and leads to higher fuel requirements for a given mission profile. In contrast, low-thrust propulsion systems, such as ion engines, have significantly increased in popularity due to their fuel efficiency and finer maneuvering capabilities, and are typically preferred when timing is not a primary concern.

Optimizing low-thrust trajectories for orbital maneuvering presents its set of challenges [2]. The dynamics under low-thrust propulsion are highly nonlinear, complicating trajectory calculations. Such trajectories often last weeks or months, to complete, involving numerous knot points for accurate path representation that can lead to memory issues in naive trajectory planners with intricate cost landscapes. Despite these challenges, kinodynamic sampling-based motion planners have proven effective in finding feasible trajectories [3] with complex constraints. However, using typical Euclidean distances as a metric may not adequately represent the dynamics, leading to inefficient exploration and slow convergence [4]. [5] have demonstrated that Rapidly-exploring Random Trees (RRTs) can explore the state space more efficiently when the distance metric aligns closely with the actual cost-to-go.

In this work, we address the problem of designing low-thrust trajectories for spacecraft during proximity operations with obstacles. We leverage RRT*, an optimal kinodynamic sampling-based planner, incorporating a heuristic based on the Linear Quadratic Regulator (LQR) for node extension and distance computations. The heuristic, adapted from [4], uses the LQR cost-to-go metric for producing locally optimal paths according to the LQR quadratic cost.

2 Approach

2.1 LQR-RRT*

As introduced in class, RRT* is a sampling-based algorithm that continuously samples random points and extends/steers the closest point on the tree toward the current sample. Additionally, it checks and optimizes

the interconnections of the nodes within a local neighborhood to ensure that the paths are as short as possible, which guarantees asymptotic optimality [6]. This is known as the rewiring step.

In our project, we aim to find a dynamically feasible path for a spacecraft. To implement RRT*, two pertinent questions arise: 1) How do we steer the tree to yield dynamically feasible paths?, and 2) How do we define a metric distance that is informative for our problem setup?

To answer both questions, we leverage optimal control theory for linear systems as in [4]. If the system is described by non-linear dynamics, we can linearize them around a state and a control input and then apply linear control theory. The results will indeed be less accurate the farther we are from the linearization point. However, if we consider a low-thrust scenario where the control input is small and also limit the tree extensions to a single timestep, we will remain close to this point, resulting in an informative and decent approximation.

For a given linear system, we can estimate the cost-to-go from the initial to the target state by solving the closed-form Riccati equation (see Equation 1).

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \tag{1}$$

The matrix A represents the state-transition dynamics of the linear system, while S is the solution to the equation, indicating the cost of deviating from the desired state. The matrix B defines how control inputs affect state changes, R represents the cost of using these controls, and Q penalizes deviations from the optimal state.

By solving for S, we can use it for two purposes. First, we can calculate an optimal policy $\pi^*(x)$ to steer the tree (LQRSteer) toward a sample state (see Equations 2 and 3). Where x_0 and u_0 are the linearization point. In this case, we are linearizing around zero controls (u_0) and the current random point x_0. LQR steer consists of integrating the nonlinear dynamics in Equation (6) by a user defined time step.

$$K = R^{-1}B^T(x_0, u_0)^T S \tag{2}$$

$$\pi^*(x) = u_0 - K(x_0 - x) \tag{3}$$

Additionally, the S matrix can be utilized to define a more informative metric of closeness in the state and control space (see Equation 4).

$$\text{CostToGo}(x_1, x_2) = (x_1 - x_2)^T S(x_1 - x_2) \tag{4}$$

Hence, we can use the cost-to-go to find the nearest neighbor (LQRNearest) as in Equation 5. Where V is the set of all vertices of the tree. This formulation can be easily expanded to find the K-nearest neighbors (LQRNear).

$$x_{\text{nearest}} = \arg\min_{v \in V} \text{CostToGo}(v, x_0) \tag{5}$$

Similarly, we can adapt the ChooseParent and Rewire steps to our formulation of distance, as detailed in Algorithms 2 and 3, respectively. Another implementation detail of our project is that we are using a goal-biased strategy for sampling, in which the probability of sampling the goal is set at 0.2. The whole pipeline of LQR-RRT* is presented in Algorithm 1.

Algorithm 1 LQR - RRT*

for $i = 1, \ldots, N$ **do**
  $x_{\text{rand}} \leftarrow$ Sample;
  $x_{\text{nearest}} \leftarrow \text{LQRNearest}(V, x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \text{LQRSteer}(x_{\text{nearest}}, x_{\text{rand}})$;
  $X_{\text{near}} \leftarrow \text{LQRNear}(V, x_{\text{new}})$;
  $x_{\text{min}} \leftarrow \text{ChooseParent}(X_{\text{near}}, x_{\text{new}})$;
  if $\text{CollisionFree}(x_{\text{min}}, x_{\text{new}})$ **then**
    $X \leftarrow X \cup \{x_{\text{new}}\}$;
    $E \leftarrow E \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
    $(V, E) \leftarrow \text{Rewire}((V, E), X_{\text{near}}, x_{\text{new}})$;
  end if
end for
return $G = (V, E)$;

Algorithm 2 ChooseParent

minCost $\leftarrow \infty$; $x_{\text{min}} \leftarrow$ NULL;
for $x_{\text{near}} \in X_{\text{near}}$ **do**
  if Cost(x_{near}) + CostToGo($x_{\text{near}}, x_{\text{new}}$) < minCost **then**
    minCost $\leftarrow$ Cost(x_{near}) + CostToGo($x_{\text{near}}, x_{\text{new}}$);
    $x_{\text{min}} \leftarrow x_{\text{near}}$;
  end if
end for
return x_{min};

Algorithm 3 Rewire

for $x_{\text{near}} \in X_{\text{near}}$ **do**
  if Cost(x_{new}) + CostToGo($x_{\text{new}}, x_{\text{near}}$) < Cost($x_{\text{near}}$) **then**
    if CollisionFree($x_{\text{new}}, x_{\text{near}}$) **then**
      $x_{\text{parent}} \leftarrow$ Parent(x_{near});
      $E \leftarrow E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
      $E \leftarrow E \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
    end if
  end if
end for
return (V, E);

2.2 Dynamics

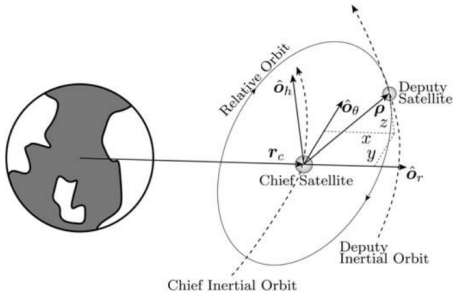

Figure 1: Relative Chief and Deputy Satellite Orbits

We use the nonlinear relative dynamics model to describe the motion of a chaser spacecraft (deputy) with respect to a coordinate system fixed at the target spacecraft (chief) [7]. This relative orbit is shown in Figure 1 and the nonlinear dynamics are shown in (6). In (6), we use a 6 dimensional state $x = [p_x, p_y, p_z, v_x, v_y, v_z]$ and a 3 dimensional control vector $u = [u_x, u_y, u_z]$. The parameter μ is the gravitational parameter for Earth and r_c and r_d are the current orbit radius of the chief orbit and the deputy orbit respectively. The parameter m represents the mass of the spacecraft which we assume to be $100kg$ and $\dot{\theta}$ is the mean motion for the target orbit.

$$\begin{aligned} \ddot{p_x} &= \frac{-\mu}{r_c^3}(r_d + p_x) + \dot{\theta}^2 p_x + 2\dot{\theta}(\dot{p_y} - p_y\frac{\dot{r}_d}{r_d}) + \frac{\mu}{r_d^2} + \frac{u_x}{m} \\ \ddot{p_y} &= \frac{-\mu}{r_c^3}p_y + \dot{\theta}^2 p_y - 2\dot{\theta}(\dot{p_x} - p_x\frac{\dot{r}_d}{r_d}) + \frac{u_y}{m} \\ \ddot{p_z} &= \frac{-\mu}{r_c^3}p_z + \frac{u_z}{m} \end{aligned} \tag{6}$$

2.3 Implementation

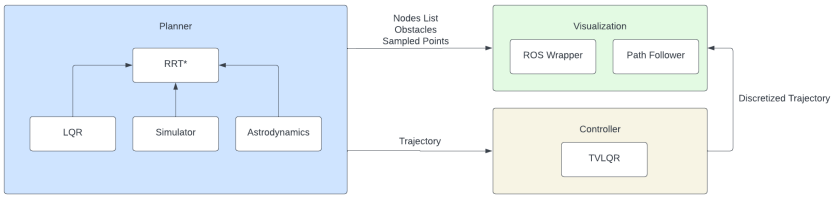

Figure 2: Planner System Diagram

For our implementation, we had three components to the system: the Planner, Controller, and Visualizer. For our planner, we had LQR, Simulator, and Astrodynamics components of the system feed into the RRT* planner as described in Section 2.1 and 2.2. We also implemented a TVLQR controller to discretize the path further as well as a ROS wrapper for visualization.

The visualization node gets a discretized path from the TVLQR controller and various planning components (nodes list, obstacles, and sampled nodes) for testing and debug visualizations. The nodes list was used to visualize tree by simply drawing a line between the x,y,z components of the parent and child pointers for a given node. Obstacles were represented using an asteroid mesh and sampled points were represented as spheres in the environment. Finally, a path follower was developed to show the spacecraft following the trajectory. For every timestep, the agent simply moved to the next waypoint in the discretized trajectory. The orientation for each waypoint was inferred through the node's velocity components. A sample path visualization is shown below in Figure 3

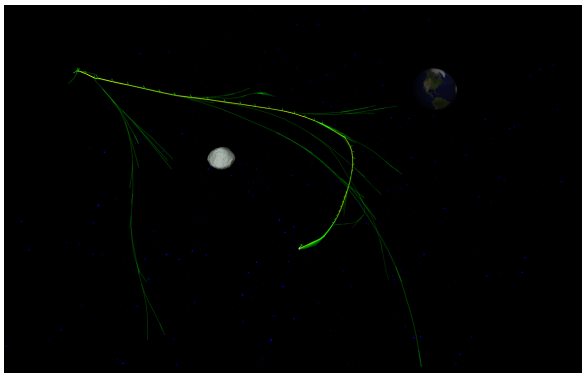

Figure 3: Planner Visualization

3 Experimental Results

dt	Neighborhood Radius	Max. Neighbor	Goal Tolerance	Goal Bias
1.0	1000.0	10.0	5.0	0.25

Table 1: Planning Parameters

The experiment consisted of a chaser spacecraft placed at an arbitrary initial state equal to $x_c = [-12.306, 24.390, 34.307, 0.01, 0.02, -0.015]$ and the goal state at the origin which is the target state in the relative frame. The initial condition has been chosen such that if no control action were applied, the spacecraft would pursue its path along an ellipse around the target. We also placed an asteroid to simulate an obstacle in the path of the chaser spacecraft. The asteroid position was set to $a_c = [-5, 10, 20]$ with a radius of 7 m, and we assume the obstacle as static in the relative coordinate frame which is equivalent to the asteroid having the same motion as the target spacecraft in the inertial frame. We ran three simulations, each with a different R parameter from the LQR cost function to obtain different trajectories. The parameters of the planner such as the simulation timestep dt, neighborhood radius tolerance on the cost to go, the maximum amount of neighbors a node can have, the goal tolerance, and the probability of sampling the goal are summarized in Table 1. The results of the planner for each experiment are shown in Table 2 and Figure 4. The machine used for these experiments was a Dell XPS 13 with a 12th Gen Intel i7 processor and 32 GB of memory.

LQR RRT* Planner Results				
R	Sample size	Tree size	Path size	Solve time
1	486	357	42	18.539 s
10	1353	1044	60	88.453 s
100	2738	1913	175	282.093 s

Table 2: Results of the LQR RRT* Planner

Overall, the R value correlates to the planner run time and solution size. A smaller R allows the planner to find a solution much faster as there is no penalty on the spacecraft control usage. A high R value corresponds to

the low-thrust behavior we desire as the spacecraft makes use of its dynamics to dock with the target spacecraft, as can be seen in the trajectories from Figure 4. With R=100, the trajectory follows more of an elliptical path corresponding to the natural dynamics of the chaser spacecraft. We verify the control usage of following each of these trajectories through a TVLQR controller. A TVLQR controller is a trajectory tracking controller that we use to follow each of the planner output trajectories from Figure 4. We linearize the dynamics about each of the nodes in the trajectory and use the LQR optimal policy from 3) to minimize the LQR quadratic cost between the current spacecraft state and the reference trajectory state. For this TVLQR controller we set Q to 1×10^5 as we want the controller to minimize the deviations from the state trajectory and the R was set to 1 as we do not need to add additional cost to the controls as the reference trajectories the planner outputs already minimize for control usage by varying R in the cost function. We plot the L1 norm of the control (thrust) vector at each timestep for each of the trajectories, and this is shown in Figure 5. As expected, the trajectory with a high R value from the planner generates a low thrust trajectory. However, there is a trade-off in low thrust trajectory maneuvers as it takes the spacecraft more time to reach its goal which may potentially use the same amount of fuel as the high thrust trajectory maneuvers. However, some propulsion systems have a thrust limit, and having a high penalty on R imposes this low thrust behavior that is feasible to run onboard spacecraft. With LQR, it is not guaranteed that the controls satisfy some upper bound constraint, however we plan to implement a model predictive controller that uses the output of the planner as a reference trajectory to include constraints in the problem.

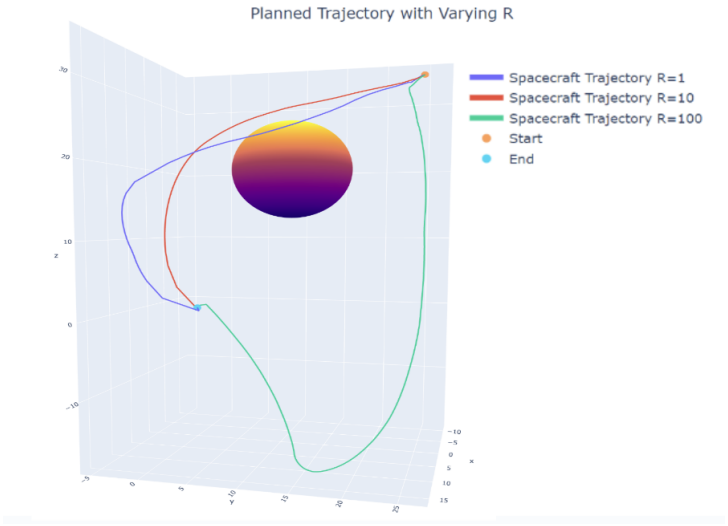

Figure 4: Planner Trajectories with Varying R

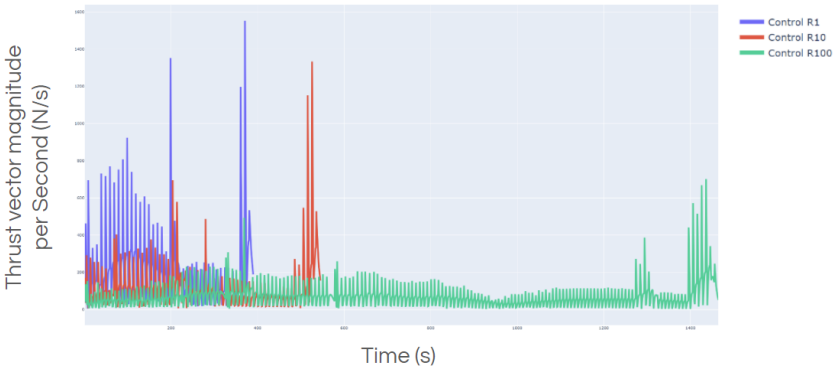

Figure 5: Planner Trajectories with Varying R

4 Conclusion

We have presented an RRT* implementation using LQR as a heuristic for node extension and distance calculation, designed to produce low-thrust trajectories for spacecraft proximity operations involving obstacles. Our

approach showcases easy tuning through the adjustment of Q and R matrices and eliminates the need for trajectory post-processing, attributable to the smoothing effect of the dynamics and the quadratic cost inherent in the LQR policy. To ensure real-system applicability, we introduced a Time-Varying LQR (TVLQR) controller which demonstrated the effects of the tuning parameters on thrust usage as expected. With our clear software architecture, our method offers efficient and rapid implementation capabilities, well-suited for onboard spacecraft systems considering the typically slow translational dynamics of these vehicles. Looking ahead, further developments would involve evaluating the performance of our system with more complex dynamics (e.g. orbital transfers), integrating dynamic obstacles processing, and incorporating attitude control.

References

[1] J. E. Prussing and B. A. Conway, *Orbital Mechanics*, second edition ed. New York: Oxford University Press, 2013.

[2] K. Tracy and Z. Manchester, "Low-thrust trajectory optimization using the kustaanheimo-stiefel transformation," in *AAS/AIAA Space Flight Mechanics Meeting*, Charlotte, NC, 2021.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[4] A. Perez *et al.*, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2537–2542.

[5] P. Cheng and S. LaValle, "Reducing metric sensitivity in randomized trajectory design," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 1, Oct. 2001, pp. 43–48 vol.1. [Online]. Available: https://ieeexplore.ieee.org/document/973334

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 846–853.

[7] H. Schaub and J. L. Junkins, *Analytical mechanics of space systems*. Aiaa, 2003.