Real-time trajectory planning for free-floating space robot in close range using continuous thrust

Ying Chen, Zhen He and Shunli Li
School of Astronautics, Harbin Institute of Technology, Harbin 150001, China

1 E-mail: lishunli@hit.edu.cn

Abstract. This paper investigates the trajectory planning problem of free-floating space robot in proximity with continuous thrust. Sampling-based motion planning algorithm is viewed as promising method in such high-dimensional, complex scenario. A novel, fast trajectory planning algorithm AGB-RRT* is proposed using linearized orbital dynamics, which is capable of preventing collisions efficiently. Several constraints are considered, such as obstacle avoidance, plume impingement and control feasibility. Two extensions of AGB-RRT* algorithm are made: adaptive sampling strategy and LQR steering method. Finally, the performance of proposed algorithm are tested in 2D planar case and 3D case. Simulation results indicate the real-time feasibility of the proposed AGB-RRT* algorithm and the collision-free trajectories are displayed.

1. Introduction

Autonomous space systems have received widely attention throughout the world in decades. In January 2012, the National Research Council (NRC) released a report named “TA04” assessing robotics, tele-robotics and autonomous systems as high-priority technologies, which would restore “NASA’s technological edge and paving the way for a new era in space” [1]. Meanwhile, close relative guidance and control is viewed as top-class technology of autonomous space systems, because this technology enables many space missions such as on-orbit maintenance, space surveillance and debris removal, etc.

Generally speaking, motion planning problem is, given an initial state and goal state, to find a collision-free time-dependent trajectory of control inputs and system state. With the development of motion planning method, several state-of-the-art techniques for space robot proximity operations have been proposed, such as A star planning [2], artificial potential method [3] and mathematical optimization [4]. These methods exist disadvantages in dealing with collisions or global optimality, as well as computational complexity. Therefore, a more efficient, solvable method is meaningful in this situation.

Sampling-based motion planning (SBP) algorithms, firstly proposed in robotics, solve high dimensional, tightly constrained optimal problem. The core idea of SBP is sampling in the state space, and then connect these samples to form a path to the goal avoiding obstacles. When the number of samples go to infinity, the possibility of find a solution if one exists tends to be deterministic. The most popular and primitive SBP algorithms are Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRTs) [5]. Preliminary attempts have been made on trajectory planning and 6-DOF planning for proximity operations using RRT algorithm in [6-7]. RRT algorithm could not guarantee optimality. To avoid this drawback, [8] presents a two-stage approach using fast solution from RRT
algorithm as an initial guess to gauss pseudospectral method. Further research in [9-10] proposes two modified version of sampling-based algorithms, namely cross entropy RRT* and Fast Marching Tree (FMT*).

Balancing exploration and exploitation in the search tree, a modified adaptive goal-biasing RRT* (AGB-RRT*) algorithm is proposed in this paper that has fast searching rate and good performance in optimality. Then, considering obstacle avoidance and thruster plume impingement, the proposed AGB-RRT* algorithm is applied in close relative trajectory planning.

2. Problem description

Consider the problem of a free-floating space robot autonomously maneuvering toward a target spacecraft that is moving in a prescribed circular orbit in proximity in short time scale. Assume the orbital and attitude information about the target is predicted initially. This paper mainly focus on the development of real-time trajectory planning algorithm that satisfies all the constraints and enable the asymptotically optimal solution.

2.1. System dynamics

First of all, define the coordinate frames as shown in figure 1.

Because proximity operations in the context is finished in relatively short period, perturbations such as Earth oblateness, atmospheric drag and solar radiation pressure could be neglected in translational motion. In this paper, the linearized Clohessy-Wiltshire (C-W) dynamics is implemented to describe relative orbital motion using a continuous thrust strategy.

Define the six dimensional translational state \( x = [\delta x, \delta y, \delta z, \delta \dot{x}, \delta \dot{y}, \delta \dot{z}]^T \) and unit control force \( u \), the linearized relative orbital equations for this scenario as resolved in the \( \hat{r}_t \) frame is given by:

\[
\dot{x}(t) = Ax(t) + Bu(t)
\]

where matrices \( A \) and \( B \) are

\[
A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
3n^2 & 0 & 0 & 2n & 0 & 0 \\
0 & 0 & 0 & -2n & 0 & 0 \\
0 & 0 & -n^2 & 0 & 0 & 0
\end{bmatrix}, \\
B = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Parameter \( n \) denotes the angular velocity of reference spacecraft orbit, can be calculated by Earth gravitational parameter \( \mu \) and reference radius \( r_t \) using \( n = \sqrt{\mu / r_t^3} \).
2.2. Constraints modelling

In spacecraft proximity operations, differential constraint could be viewed as the system dynamics, which is detailed previously. Besides, other constraints such as obstacles, thruster plume and control feasibility also need be carefully dealt with.

2.2.1. Obstacle Avoidance. Keeping the space robot out of the obstacle region is essential in proximity operations. The main obstacle region is the hull of target spacecraft, also known as keep-out-zone. Generally, it’s modelled as a sphere centered at the origin with the radius equals to the maximum characteristic length of target spacecraft. However, for ultra-close relative motion, a sphere region would shrink the solution space as the solar panels are much lengthier. Instead of that, a "sphere + ellipsoid" composite is introduced in this paper and illustrated as figure 2.

Figure 2. Illustration of obstacle region.

2.2.2. Plume impingement. Plume impingement is caused by exhaust gas pressure from thrusters, will lead to dire consequences such as spacecraft tumbling, degradation of optical payload and solar arrays. Due to the uncontrollability of target spacecraft, plume model should be built at the location of every thruster. The influence region of plume is approximately a circular cone with axis along the direction of thrust, half-angle $\alpha$ and range $H$. Assume the direction of thrust is $-\hat{t}_{toj}$, implemented at a fixed point $[x_0, y_0, z_0]^T$, the influence region is represented as:

$$(x - x_0)^2 + (z - z_0)^2 - (y - y_0)^2 \tan^2 \alpha \leq 0, \quad (y_0 - H) \leq y \leq y_0$$

(3)

2.3. Problem formulation

Define the state space $\mathbb{S} \subset \mathbb{R}^d$ as a $d$ dimensional region in $\mathbb{R}^d$, and let $\mathbb{N}_{\text{obs}} \subset \mathbb{N}$ be the obstacle region, such that $\mathbb{N}_{\text{free}}$ denotes the obstacle-free space which is the complementary set of $\mathbb{N}_{\text{obs}}$. Let $x_{init}$ and $x_{goal}$ represent the initial and goal state relative to the target. Finally, $x(t) : \mathbb{R} \rightarrow \mathbb{S}$ is the state trajectory with control sequence $u(t) : \mathbb{R} \rightarrow U$. The optimal spacecraft trajectory planning problem could be formulated in mathematical language as:

$$\begin{align*}
& \text{minimize} \quad J = \int_{t_{\text{min}}}^{t_{\text{max}}} (x^T Q x + u^T R u) dt \\
& \text{s.t.} \quad \dot{x}(t) = Ax(t) + Bu(t) \\
& \quad x(t) \in \mathbb{N}_{\text{free}}, t \in [t_{\text{min}}, t_{\text{max}}] \\
& \quad g\left(\left[\left[ x(t), u(t)\right]\right]\right) \leq 0
\end{align*}$$

(4)

where $Q$ and $R$ are positive definite gain matrices.
3. AGB-RRT* algorithm

3.1. Existing problems to be improved

![Figure 3. Expansion of common random tree.](image)

RRT* is incremental sampling-based planning algorithm, generate a random sample at every iteration and connect to the search tree, then unbiased expand to the unexplored space. Figure 3 describes the expansion of the basic search tree.

**Algorithm 1. Basic RRT***

1. \( G(V, E) \leftarrow \text{Initialization}(x_{\text{init}}) \);
2. **For** \( N \) iterations **do**
   3. \( x_{\text{rand}} \leftarrow \text{Sample}(\mathcal{N}_{\text{free}}) \);
   4. \( V_{\text{near}} \leftarrow \text{Near}(V, x_{\text{rand}}) \);
   5. \( x_{\text{new}} \leftarrow \text{Steer}(V_{\text{near}}, x_{\text{rand}}) \);
   6. **if** Collisionfree=true **then**
      7. \( G(V, E) \leftarrow \text{Rewire}(V_{\text{near}}, V, x_{\text{new}}) \);
   10. **return** \( G(V, E) \)

**Algorithm 2. Adaptive biasing sampling**

1. \( r \leftarrow \text{rand} \);
2. **if** \( r < \sigma \) **then**
   3. \( x_{\text{rand}} \leftarrow x_{\text{rand}} \); \( \text{flag} = 0 \);
   4. **else**
      5. \( x_{\text{rand}} \leftarrow \text{Sample}(\mathcal{N}_{\text{free}}) \); \( \text{flag} = 1 \);
   6. **if** \( \text{flag} = 0 \) **then**
      7. **if** Collisionfree=true **then**
         8. \( \sigma = 2\sigma \);
         9. \( \sigma = \min(\sigma, \alpha) \);
   10. **else**
      11. \( \sigma = \sigma / 2 \);
   12. **return** \( x_{\text{rand}} \)

Basic RRT* algorithm is given as algorithm 1. It’s obvious that the RRT* algorithm has hierarchical structure with local planner \text{Steer}() as the low-level planner and the global planner as the high-level one. The essence of local planning is solving an unconstrained two point boundary value problem (2PBVP). Only edges checked by collision checking could be added into the tree. Collision checking technique does not require the explicit expression of the constraints, which provides advantages over other motion planning methods in dealing with optimization problems in real time.
It’s observed from figure 3 that the search tree is “wandering” in the state space without clear goal, which causes a waste of time in meaningless searching. But purely exploiting known information may result in local optimal solution. In addition, 2PBVP is really difficult in relative orbital motion giving consideration to both computational time and optimality. At last, Near() highly depends on the right selection of metric function. Unfortunately, this is still an open question, usually implemented using Euclidean distance [5]. LaValle [11] pointed out that the solution would be closer to the optimal one when the metric is selected as the cost function. Therefore, an adaptive sampling algorithm is designed to balance exploration and exploitation in this paper. Afterwards, linear quadratic regulator (LQR) method is applied in solving the 2PBVP.

3.2. Adaptive sampling strategy
Different from “breadth first” strategy adopted by basic RRT*, the modified sampling strategy combines heuristic goal biasing with random sampling. The idea behind goal-biasing heuristic is randomly selecting the goal state as the sample to be connected at current iteration. Based on the algorithm from [12], adaptive parameter σ is designed in algorithm 2 to adjust the degree of goal biasing.

When the goal state is accepted then the parameter σ will become greater to increase the possibility to be selected at next iteration. However, when the goal state is rejected by collision checking, σ will become smaller to explore the unknown space better. figure 4 describes the expansion of the modified search tree. It’s noticed that solution is found at early N=500.

3.3. Optimal LQR steering
The functionality of Steer() is solving the 2PBVP and forming the edges of the search tree. Without loss of generality, steering problem is defined as follow [13].

**Definition 1 (Steering).** Given initial state \( x_0 \in \mathbb{R} \), terminal state \( x_f \in \mathbb{R} \), fixed duration \( \Delta t \), incorporating C-W dynamics Steer() returns a state \( z \) that \( z \) minimize the metric function \( \rho(z,x_f) \).

\[
\text{Steer}(x_0,x_f) := \arg\min_{z \in \mathbb{R}} \rho(z,x_f)
\]

**Algorithm 3. LQR steering**

1. \([K,S] \leftarrow \text{LQR}(A,B,Q,R)\);
2. \(x \leftarrow x_0 - x_f; \quad t = 0;\)
3. \(\text{while } t < \Delta t \text{ do}\)
   4. \(u^* \leftarrow -Kx; t = t + \Delta t;\)
   5. \(x \leftarrow \text{ODE45}(@\text{C-W},x,u^*,dt);\)
6. \(z \leftarrow x + x_f;\)
12. \(\text{return } z\)
The simplest solution is a straight line between arbitrary two samples without considering any constraints. However, it’s not suitable in this case with differential dynamics. [14] employs shooting method in the local planner with unacceptable high computational complexity. Therefore, based on LQR-tree presented in [15], this paper employs LQR method to solve unconstrained 2PBVP.

LQR is capable of solving optimal control problem for linear system, just fits the linearized relative orbital dynamics in this context. Given the cost function of the system as:

\[ J = \int_{t_i}^{t_f} (x^T Q x + u^T R u) \, dt \]  

(6)

And the optimal control strategy is

\[ \sigma^*(x) = -Kx \]  

(7)

where LQR gain matrix \( K = R^{-1}BS \). To sum up, \texttt{LQRSteer()} is presented in algorithm 3.

It’s noted that the cost function \( J \) could be used to describe “distance”, vertices with high cost are further to the start vertex. Inspired by this, when searching the \( k \)-nearest neighbour, the metric could be replaced by cost. Hence, \texttt{LQRNAriest} is presented as [15-17] in this paper, which means the vertices in the search tree with lowest cost would be selected as nearest neighbour.

\[ x_{\text{nearest}} = \arg \min_{v \in V} (v-x_{\text{rand}})^T S(v-x_{\text{rand}}) \]  

(8)

4. Numerical simulation

Consider the free-floating space robot maneuvers toward the malfunctioning spacecraft in GEO for inspection and further on-orbit service. Initial orbital and system parameters are shown in table 1.

| Parameter                        | Value | Unit |
|---------------------------------|-------|------|
| Space robot mass \( m \)        | 30    | kg   |
| Orbital reference radius \( r_i \) | 42162 | km   |
| Orbital eccentricity \( e \)    | 0     | -    |
| Orbital inclination \( i \)     | 0     | deg  |
| Orbital true longitude \( l \)  | 125   | deg  |

Table 2 presents the planner gains and relevant parameters. Firstly, the relative orbital state of the space robot is described as \( x \in \mathbb{R}^d, \ x = [-10,-40,0,0]^T \) when \( d=4 \) and \( x = [-10,-40,10,0,0,0]^T \) in 3D case. And then the goal region is set as hyper ball with position tolerance 0.1 m and velocity tolerance 0.01 m/s. Finally, the velocity magnitude upper bound of space robot is 0.33 m/s.

| Parameter                  | Value | Unit | Parameter                  | Value | Unit |
|----------------------------|-------|------|----------------------------|-------|------|
| Step size \( \Delta t \)   | 2     | s    | Range of plume \( H \)     | 16    | m    |
| LQR step \( dt \)          | 0.05  | s    | Number of Samples \( N \)  | 3000  | -    |
| Thrust \( T \)             | 1     | N    | Adaptive parameter \( \sigma \) | 0.1   | -    |
| Half angle of plume \( \theta \) | 10    | deg  | Upper bound of GB \( \alpha \) | 0.5   | -    |

The resultant trajectory in planar case is displayed in figure 5, which is selected randomly. As shown in figure 5, the collision region is filled in black, the bold red line is the trajectory. And the resultant trajectory in 3D case is displayed in figure 6, which is also randomly selected. From these figures, it’s noted that the trajectory is collision-free and sub-optimal, all the results are solved rapidly within about 90 second.
5. Conclusions
This paper presents a modified AGB-RRT* algorithm and applies it to fast trajectory planning for free-floating space robot in close range. The proposed algorithm performs well in balancing exploration and exploitation, and outputs a near optimal solution in short-time-scale. There are mainly two important modifications in proposed algorithm:

1. Adaptive sampling strategy with goal biasing;
2. Local planner with LQR optimal control policy.

Finally, numerical tests are implemented in both planar case and 3D case. Simulation results indicate the effectiveness of AGB-RRT* algorithm. Further research could focus on parameter tuning and 6-DOF motion planning for space robot.

References
[1] Starek J A, Akmee B, Nesnas I A and Pavone M 2016 Lect. Notes Contr. Inf. Sci. 460 1–48
[2] Paluszek M and Thomas S 2005 Infotech@Aerospace (Arlington) vol 1 pp 1–10
[3] Bevilacqua R, Lehmann T and Romano M 2011 Acta Astronaut. 68 1260–1275
[4] Richards A, Schouwenaars T, How J P and E. Feron 2002 J. Guid. Contr. Dynam. 25 755–764
[5] Lavalle S M and Kuffner J J 1999 Proc. Int. Conf. on Robotics and Automation (Tokyo) vol 1 pp 473–479
[6] Garcia I and How J P 2005 Proc. of American Control Conference (Portland) vol 2 pp 889–94
[7] Wang P, Guo J F, Shi X N and Cui N G 2011 Control Decis. 32 741–748
[8] Aoude G S, How J P and Garcia I M 2008 J. Astronaut. Sci. 56 515–544
[9] Kobilarov M and Pellegrino S 2014 J. Guid. Contr. Dynam. 37 566–579
[10] Starek J A, Schmerling E, Maher G D, Barbee B W and Pavone M 2016 J. Guid. Contr. Dynam. 40 418–438
[11] Lavalle M 2006 Planning algorithms (London: Cambridge University Press)
[12] Urmson C and Simmons R 2003 Proc. Int. Conf. on Intelligent Robots and Systems (Pittsburgh) vol 2 pp 1178–1183
[13] Chen Y, He Z, Zhou D, Yu Z and Li S 2018 Acta Astronaut 148 175-185
[14] Jeon J 2015 Sampling-based motion planning algorithms for dynamical systems (Boston: Massachusetts Institute of Technology)
[15] Perez A, Platt R and Konidaris G 2012 Proc. Int. Conf. on Robotics and Automation (Saint Paul) vol 1 pp 2537-2542
[16] Zhu B, Zhang Z, Suo M., Chen Y and Li S 2018 J. Vibration and Contr  (DOI 10.77546318754681)
[17] Zhu B, Zhang Z, Zhou D, Ma J and Li S 2017 Int. J. Syst. Sci. 48 2356-2367