Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research xx (2024) xxx-xxx

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Type the title of your paper, only capitalize first word and proper nouns

Hang Zhou[1], Tao Meng[1,1,]

[a]School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, 310027, China
[b]Zhejiang Key Laboratory of Micro-nano satellite Research, Hangzhou, 310027, China

Abstract

wait for edit test – version Please type your abstract here, and the rest of the text, figures, tables, equations etc. in the main body. Please do not modify LaTeX commands unless you need to modify them and know how to do it.

Keywords: motiom planning; trajectory optimization; time-varying obstacles

1. Introduction

Reconfiguring spacecraft operations is extremely complex, especially when they approach other objects (such as other spacecraft or space debris). These close-proximity operations carry high risks due to potential collisions and the unpredictable behavior of the objects involved. Consequently, a significant amount of research is dedicated to finding effective collision avoidance strategies for spacecraft motion planning. This research explores a variety of algorithms, ranging from traditional astrodynamics methods, such as calculating trajectories from one point to another, to more advanced techniques like model predictive control and artificial potential functions. Although these advanced methods show effectiveness in control, they often struggle with optimization challenges and varying constraints.

In the search for feasible solutions, some researchers have started focusing on sampling-based motion planning techniques, which are widely used in robotics. These techniques are considered capable of providing insights into complex motion planning problems, achieving collision-free path planning in astrodynamics, and adapting to ever-changing and optimization-demanding conditions. The effectiveness of these techniques has been demonstrated in related literature and offers potential methods and directions for safe motion planning of future spacecraft.

2. Problem Formulation

2.1. System Dynamics

Since spacecraft proximity operations involve significant drift, space-dependent external forces, and rapid temporal scale changes, any feasible solution must comply with dynamic constraints; In this article, the linearized motion of a circular reference orbit

*Corresponding author

https://dx.doi.org/10.1016/j.jasr.xxxx.xx.xxx
0273-1177/

with radius r_{ref} about the Earth's gravitational constant μ is represented using the classic CWH equations. This model provides a first-order approximation for tracking the motion of a spacecraft relative to the rotating target coordinate system. The linearized motion equations for this scenario are given in the target's LVLH frame.

$$\delta\ddot{x} - 3n_{\text{ref}}^2\delta x - 2n_{\text{ref}}\delta\dot{y} = \frac{F_{\delta x}}{m} \tag{1a}$$

$$\delta\ddot{y} + 2n_{\text{ref}}\delta\dot{x} = \frac{F_{\delta y}}{m} \tag{1b}$$

$$\delta\ddot{z} + n_{\text{ref}}^2\delta z = \frac{F_{\delta z}}{m} \tag{1c}$$

where, $n_{ref} = \sqrt{\frac{\mu}{r_{ref}^3}}$ is the average angular velocity of the reference orbit, m is the mass of the spacecraft, $\boldsymbol{F} = [F_{\delta x}, F_{\delta y}, F_{\delta z}]^\top$ is the thrust on the spacecraft, and $(\delta x, \delta y, \delta z)$, $(\delta\dot{x}, \delta\dot{y}, \delta\dot{z})$ are the relative position and velocity vectors.

Define the state variable $\boldsymbol{x}$as $[\delta x, \delta y, \delta z, \delta\dot{x}, \delta\dot{y}, \delta\dot{z}]^\top$The control variable $\boldsymbol{u}$is the force exerted on the unit mass $\frac{1}{m}\boldsymbol{F}$, and the CWH equation can be described by a linear time invariant (LTI) system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2}$$

Where the dynamic matrix $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$ are given by the following formula

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n_{ref}^2 & 0 & 0 & 0 & 2n_{ref} & 0 \\ 0 & 0 & 0 & -2n_{ref} & 0 & 0 \\ 0 & 0 & -n_{ref}^2 & 1 & 0 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

2.2. time-varying programming problems

Aiming at the problem of tracking spacecraft maneuvering towards a single target with time-varying obstacles in a well-defined circular orbit and considering the space-time motion planning problem with unbounded arrival time, we aim to minimize the arrival time under given constraints by adding a time latitude to the state space. We get the space-time state space $\mathcal{Q} = \mathcal{X} \times \mathcal{T}$, where $\mathcal{X} \subset \mathbb{R}^3$ Is the configuration state space,$\mathcal{T}$is the time state space, and $\mathcal{Q}$can be unbounded in time. Let $\mathcal{Q}_{free} \in \mathcal{Q}$ be the collision-free subset of the state, and $\mathcal{Q}_{start}$be the initial state, $\mathcal{Q}_{goal} = \mathcal{X}_{goal} \times \mathcal{T}_{goal}$is the target region, and in the following work, the trajectory of the obstacle is assumed to be completely known Therefore, the planning problem can be described as finding a path solution that avoids time-varying obstacles in the shortest possible time given the initial state $q_{start}(t_{start})$, the target region or the target state $x_{goal} \in \mathcal{X}_{goal}$

$$\begin{aligned} &\min \quad \int_{t_{start}}^{t_{goal}} 1dt \\ &s.t. x(t_{init}) = x_{init} \\ &\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ &\boldsymbol{x}(t) \in \mathcal{X}_{free}, t \in [t_{init}, t_{final}] \end{aligned} \tag{4}$$

3. Main result

3.1. Path generation based on TubeST-RRTstar

The TubeST-RRT* algorithm is an improved version of RRT* Connect, modifying the sampling function, steer function, find nearest function and rewiring function, and setting the motion check function for the spacecraft's motion dynamics.Considering the

time-varying information of obstacles and the flight corridor with added nodes, we ensure that the generated trajectory is safe and collision-free during subsequent trajectory optimization.

RRT* Connect algorithm is an asymptotically optimal bidirectional sampling method widely used in robot path planning with obstacle avoidance constraints. The core idea of the RRT method is to randomly sample in the state space to obtain a series of path nodes and directed edges from the parent node to the child node, forming a search tree $\mathcal{T}_{tree}$. To address planning problems in environments with time-varying obstacles, the Tube-ST-RRT* algorithm is proposed. This algorithm first adds a time dimension to the state space and samples the node times to obtain nodes that meet the time-varying constraints. Secondly, during the node sampling process, a radius variable is sampled to ensure that the node is collision-free within a spherical region of that radius. The final result is a sequence of nodes $\boldsymbol{s} = [\boldsymbol{s_0}, \boldsymbol{s}_1 \ldots, \boldsymbol{s}_n]$, each containing information on position, velocity, time, and the radius of the maximum collision-free spherical region.

The algorithmic details of Tube-ST-RRT* are provided in Algorithms **??**. In addition to $\mathcal{S}, q_{start}, \mathcal{X}goal, d$, the planning termination condition *ptc*, the probability of sampling a new goal $p_{goal} \in (0, 1]$, and the time limit $t_{\max} \in (0, \infty]$ are also required. The basic framework is similar to RRT-Connect.

Algorithm 1 TubeST-RRT*

Input: $\mathcal{X}, q_{start}, x_{goal}, p_{goal}, range\_d, Param, ptc$
Output: $Soulution$

$r_{start} \leftarrow FindMaxRadius(s_{start})$
$s_{start} \leftarrow \{q_{start}, r_{start}\}$
$T_a \leftarrow \{s_{start}\}; T_b \leftarrow \emptyset$
$B \leftarrow InitailieBoundVariables(Param)$
while ptc **do**
  $B \leftarrow UpdateGoalRegion(B, Param, t_{\max})$
  if $p_{goal} > rand(0, 1)$ **then**
    $B \leftarrow SampleGoal(s_{start}, x_{goal}, T_{gaol}, B)$
  end if
  $q_{rand} \leftarrow SampleConditionally(s_{start}, \mathcal{X}, B, d)$
  $r_{rand} \leftarrow FindMaxRadius(q_{rand})$
  $s_{rand} \leftarrow \{q_{rand}, r_{rand}\}$
  $s_{nearsst} \leftarrow Nearset(s_{rand}, T_a)$
  $s_{new} \leftarrow TubeSTSteer(s_{nearest}, s_{rand}, d)$
  if $MotionCheck(s_{nearsst}, s_{new})$ **then**
    $B.samplesInBatch+ = 1$
    $B.totalSamples+ = 1$
    $RewireTree(T_a, x_{new})$
    if $Connect(T_b, x_{new}, d) = Reached$ **then**
      $solution \leftarrow UpdateSolution(x_{new})$
      $t_{\max} \leftarrow CostPath(solution)$
      $B.batchProbability \leftarrow 1$
      $PruneTrees(t_{\max}, T_a, T_b)$
    end if
  end if
  $Swap(T_a, T_b)$
end while
return $Solution$

Algorithm 1 outlines the overall framework of TubeST-RRT*. The general procedure of the algorithm is as follows:

Firstly, it seeks the minimum collision-free r_{init} for the starting point and then initializes parameters. In each iteration, firstly update the boundary parameters. Then, with a probability p_{goal}, decide whether to sample a new target or sample the endpoint.

Sample a random state q_{rand}, find q_{near}, and expand q_{new} between q_{near} and q_{rand}. In q_{new}, find the maximum collision-free spherical region. If the path from q_{near} to q_{new} is collision-free and satisfies spacecraft motion dynamic constraints, add the new state x_{new} to the current tree T_a and attempt to connect from x_{new} to the other tree T_b. If the connection is successful, update the solution. Finally, swap T_a and T_b and start the next iteration. Repeat until the termination condition ptc is met.

The main improvements of our proposed TubeST-RRT* algorithm over RRT*-Connect are as follows:

1. Generation of collision-free virtual tubes

Tube-ST-RRT* generates a sequence of intersecting spheres δ_c with radius R_i and time information, as shown in Figure b. From δ_c, a path δ_0 is created, and then boundary points within the sphere intersections are connected to form boundary paths δ_1 and δ_2, as shown in Figure c. This results in a collision-free flight corridor, ensuring that the trajectory generated by minisnap is collision-free.

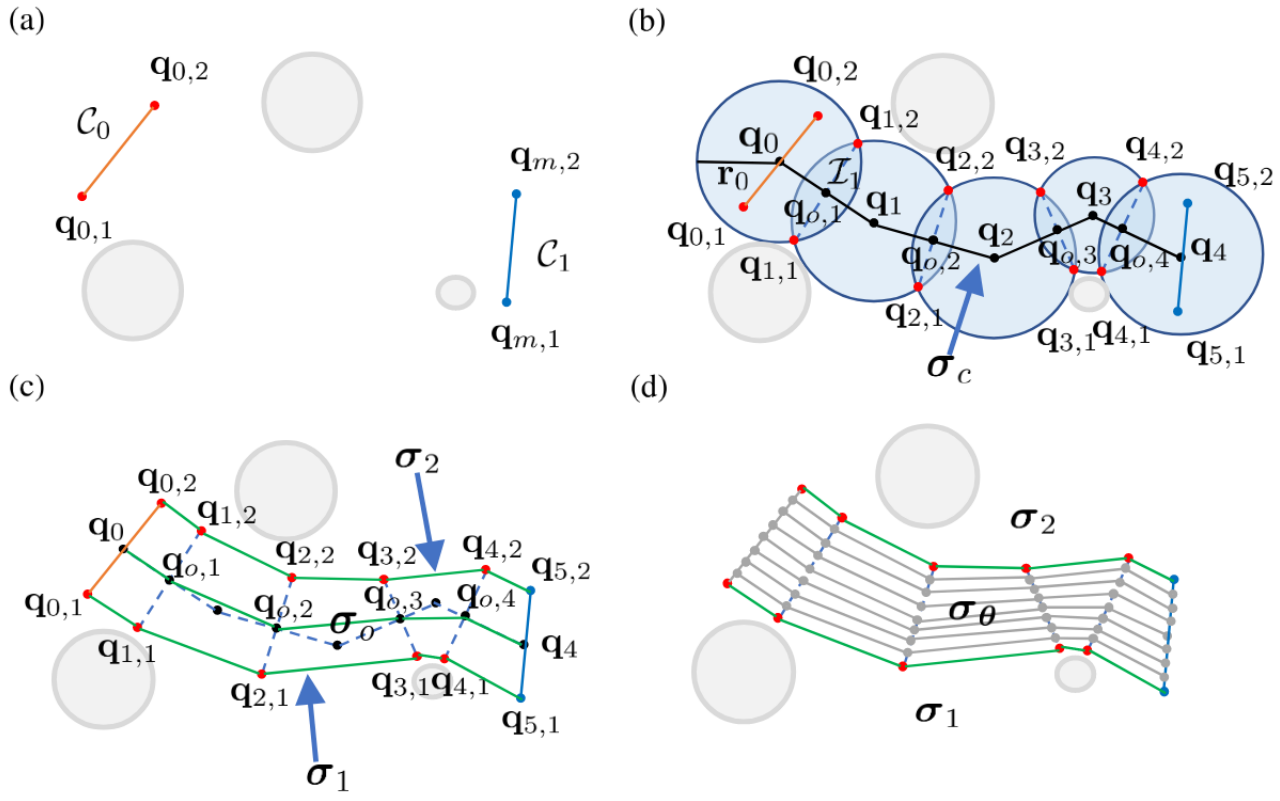

Fig. 1. Diagram of Tube Generation

2.Improved Progressive Goal Region Expansion

Since we have introduced a time dimension, it becomes challenging to generate samples distributed across the entire space if the time space is unbounded. However, applying any time constraint may render the planning problem unsolvable. Therefore, each time a new batch of samples is added, we gradually expand the sampling target area. To achieve this, we introduce several parameters: B.timeRange determines the time boundary, B.batchSize determines the number of samples generated after expansion. When a batch is filled, B.timeRange is increased by P.rangeFactor, and B.batchSize is adjusted accordingly. As the time boundary expands, due to the higher sample density at lower time values from previously generated samples, it becomes increasingly unlikely to find any solution. Hence, we employ weighted sampling to ensure a uniform distribution across the entire space.

3.Improved Conditional Sampling

Any state that can be part of the solution path must have a finite distance d from the starting point and at least one target state. Due to velocity constraints, only states at the intersection of the start cone and the target cone (see Figure 3) satisfy this requirement. Therefore, similar to informed RRT*, we only sample from the region where solutions can be generated. Ideally, sampling would directly occur from the union of the start velocity cone and the target velocity cone.

However, since it's not feasible to explicitly compute the intersection when there are multiple target states, we first uniformly sample a configuration c (Line 2 in Algorithm 5). Then, using c, we sample a feasible time within the possible time range restricted to c. The possible time range depends on s_{start} and previously sampled target states. To achieve a more uniform sampling, we utilize two sets of target states: B.goals and B.newGoals. The time boundaries $t_{lb}$and t_{ub} are obtained from the minimum arrival time from the starting configuration c_start to c (Line 3) and the given maximum valid time.

Algorithm 2 SampleConditionally

Input: s_{start}, Q, B

1: **while do**
2: $q \leftarrow SampleUniform(\mathcal{X})$
3: $t_{\min} \leftarrow t_{start} + LowerBoundArrivalTime(q_{start}, q)$
4: **if** $Rand(0,1) \leq B.batchProbability$ **then**
5: $t_{lb} = t_{\min}$
6: $t_{ub} = MaxValidTime(q, B.goals)$
7: **else**
8: $t^*_{\min} = MaxValidTime(q, B.goals)$
9: $t_{lb} = Max(t_{\min}, t^*_{\min})$
10: $t_{ub} = MaxValidTime(q, B.newgoals)$
11: **end if**
12: **end while**

3.2. reformulation dynamic coorid minimum snap Trajectory Optimization

For the state sequence generated by tubeSTRRT*, trajectory optimization can be applied to make the trajectory smoother. In minimum snap trajectory optimization, the optimized trajectory needs to pass through intermediate path nodes. However, since our tubeSTRRT* incorporates collision detection within the spherical regions, it essentially provides a collision-free flight corridor. Therefore, the trajectory only needs to stay within this corridor to ensure collision-free movement. Traditional minimum snap optimization using polynomial parameters as optimization variables may lead to numerical instability when there are too many nodes. Here, we can use reformulation minimum snap to transform the optimization variables from polynomial coefficients to physically meaningful variables such as position (p), velocity (v), and acceleration (a). Constraints can be added to p to ensure it stays within the circles generated by the planner. This ensures that the optimized trajectory is collision-free.

4. Citations

Citations in the text can be made using

```
\citet{NewmanGirvan2004}
```

for citation in running text like in **?** or using

```
\citep{Vehlowetal2013,NewmanGirvan2004}
```

for citation within parentheses like in (**??**).

Please use the actual `\cite` command in the text. Also, please double-check the `\citep` command.

5. Reference style

You can include the references in the main text file in LaTeXformat. Alternately, you can include a separate bibliography file (refs.bib in this example) and run the following set of commands:

```
==================

pdflatex jasr-template.tex

bibtex jasr-template (no extension in this line!)

pdflatex jasr-template.tex

pdflatex jasr-template.tex

==================
```

6. A sample entry in the bibliography file

```
==================

@ARTICLE{NewmanGirvan2004,
  author  = {Newman, M. E. J. and Girvan, M.},
  title   = {Finding and evaluating community
             structure in networks},
  journal = {Phys. Rev. E.},
  volume  = {69},
  number  = {21},
  year    = {2004},
  pages   = {026113}
}

==================
```

Acknowledgments

Acknowledgments should be inserted at the end of the paper, before the references, not as a footnote to the title. Use the unnumbered Acknowledgements Head style for the Acknowledgments heading.

References

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys. Rev. E.*, *69*(21), 026113.
Vehlow, C., Reinhardt, T., & Weiskopf, D. (2013). Visualizing fuzzy overlapping communities in networks. *IEEE Trans. Vis. Comput. Graph.*, *19*(1), 2486–2495.