# Asymptotically time-optimal smooth trajectory planning in dynamic environments

Hang Zhou
*School of Aeronautics and Astronautics*
*Zhejiang University*
Hangzhou, China
zhou_hang@zju.edu.cn

Tao Meng
*School of Aeronautics and Astronautics*
*Zhejiang University*
Hangzhou, China
mengtao@zju.edu.cn

***Abstract*—In this paper we proposed an algorithm for smooth trajectory generation in complex environments with dynamic obstacles and velocity constraints. The proposed algorithm Tube Space-Time RRT* (TubeST-RRT*) is combined with the improved reformulation dynamic coordinate minimum snap (RDCMS) to generate smooth collision-free trajectories with asymptotic time Optimal.First, sample the space-time state space to obtain the time information for each node to complete the avoidance of moving obstacles.Then, to address the issue of non-smooth paths in ST-RRT*, generate a dynamic Tube for each node and combine it with the improved minisnap to create a smooth, collision-free trajectory Finally,Simulations in complex environments demonstrate the effectiveness of our proposed algorithm.**

***Index Terms*—Tube Space-Time RRT*; asymptotic time Optimal; smooth collision-free trajectories**

## I. Introduction

Trajectory planning is a fundamental challenge in robotics [1], as obstacles in the real world often change over time. Applications such as robotics and autonomous driving typically require a smooth, collision-free trajectory. Assuming the obstacle trajectories are known a priori, the problem can be modeled as navigation in a dynamic environment. Mathematically, this is expressed as planning through a space-time state space [2].

Planning in dynamic environments has been studied for a long time and has yielded significant research results. These results can be broadly categorized into two approaches. For example, RRTX [3] and Real-time RRT* [4] require fast replanning when previously computed paths become invalid during execution. However, as the dimensionality increases, the replanning time becomes longer, making it difficult to respond to moving obstacles. Risk-RRT [5] combines predictions of obstacle movements and computes partial motion paths to keep the collision probability below a given threshold. However, since only partial paths are returned, frequent replanning is still necessary.Another approach assumes that the trajectories of moving obstacles are unknown, while another assumes that the trajectories of the obstacles are completely known. For example, Time-Based RRT [6] extends the configuration state space through the time dimension and unidirectionally plans to a set of known target states. However, it requires the assumption of knowing the specific time for each target configuration. Additionally, due to the random sampling nature of RRT, the resulting trajectories are usually not smooth.

In this paper, we proposes a Tube-based space-time sampling planning algorithm, TubeST-RRT*, and an improved reformulation dynamic coordinate minimum snap (RDCMS) method. Through a two-step method of planning and optimizing, it achieves asymptotic time-optimal collision-free trajectory planning in dynamic environments, while ensuring high trajectory smoothness to facilitate easier tracking control. The TubeST-RRT* algorithm adds a time dimension to the configuration space so that each node contains time information, allowing it to respond to time-varying obstacles. Additionally, the improved RDCMS method combines with each node's tube information and time information to generate smooth, collision-free trajectories. Finally, comparative simulations verify the effectiveness of the algorithm

The rest of this paper is organized as follows: At Section II, explains the problem of time-optimal planning in dynamic environments. In Section III,presents the main results of this paper, including the principles and steps of the Tube ST-RRT* planning algorithm and the RDCMS trajectory optimization algorithm. Section IV provides simulation results and comparative experiments to demonstrate the feasibility and superiority of the proposed method. Finally, Section V summarizes the paper.

## II. Problem Statement

Consider the time-space planning problem with a tube, where the space-time state space is defined as $\mathcal{Q} = \mathcal{X} \times \mathcal{T} \times \mathcal{B}$, where $\mathcal{X} \subset \mathbb{R}^n$ is the configuration state space, $\mathcal{T}$ is the time state space, $\mathcal{B}$ is the size of the tube, and $n$ is the dimension of the configuration space state. Let $\mathcal{Q}_{free} \subset \mathcal{Q}$ be the collision-free subset of states, $Q_{start}$ be the initial state, and $\mathcal{X}_{goal}$ be the goal region. In the following work, it is assumed that the trajectories of obstacles are completely known. Therefore, the planning problem can be described as finding a path solution that starts from the initial state $q_{start}$, reaches the goal region or target state $x_{goal} \in \mathcal{X}_{goal}$. The goal is to find a state-space sequence with the tube size as large as possible and the time taken as short as possible while avoiding time-varying obstacles.

## III. Main Results

### A. Path Node Generation Based on TubeST-RRT*

The TubeST-RRT* algorithm is an enhanced version of the RRT* Connect [7] algorithm , with modifications to the sampling function, steering function, nearest neighbor function, and rewiring function. Since the nodes contain time information, a motion check function can be introduced to detect velocity constraints. Additionally, an extra tube dimension is added to the state space to ensure a minimum distance between sampling nodes and obstacles. By considering the time-varying characteristics of obstacles and incorporating the tube into the nodes, the algorithm ensures that the generated trajectory remains safe and collision-free throughout 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 $T_{tree}$. To address planning problems in environments with time-varying obstacles, the TubeST-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 tube 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 TubeST-RRT* are provided in Algorithms 1. The algorithm inputs, in addition to $\mathcal{Q}, q_{start}, \mathcal{X}goal, d$, 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 outlines the overall framework of TubeST-RRT*. The general procedure of the algorithm is as follows:

First, the algorithm finds the minimum collision-free radius $r_{init}$ for the starting point and initializes the parameters. In each iteration, boundary parameters are updated. Then, with a probability $p_{goal}$, it decides whether to sample a new goal or sample the endpoint to obtain the sampling point $q_{rand}$. Next, it finds the nearest point $q_{near}$ to the sampling point and extends between $q_{near}$ and $q_{rand}$ to obtain $q_{new}$. At $q_{new}$, the maximum collision-free spherical radius $R$ is determined. If the path from $q_{near}$ to $q_{new}$ is collision-free and satisfies the velocity constraints, the new state $q_{new}$ is added to the current tree $T_a$, and an attempt is made to connect $q_{new}$ to the other tree $T_b$. If the connection is successful, the solution is updated. Finally, $T_a$ and $T_b$ are swapped, and the next iteration begins. This process is repeated until the termination condition $ptc$ is met.

As Fig.(1) shows,the blue area represents dynamic obstacles, which move from left to right over time. The orange

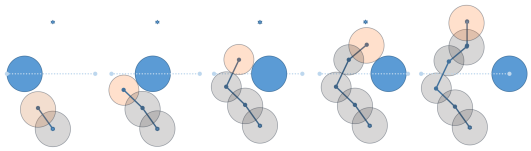

Fig. 1: llustration of the Expansion Method in TubeST-RRT

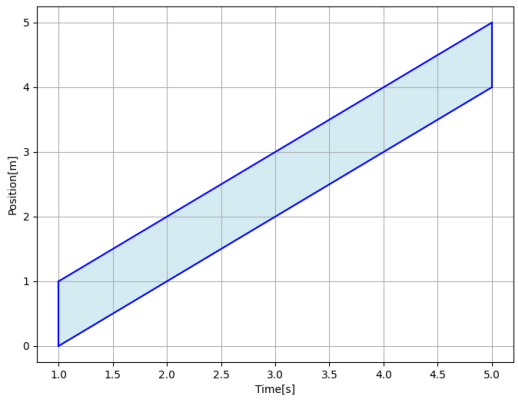


Fig. 2: union of the start velocity cone and the target velocity cone

circle indicates the maximum collision-free circle generated at the current node. The algorithm ultimately generates a path where each node has a collision-free circle, ensuring collision avoidance and providing a foundation for subsequent trajectory optimization.

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

- TubeST-RRT* generates a series of intersecting spheres with radius $R_i$ with time information, The interior of all the spheres combines to form a collision-free flight corridor, ensuring that the trajectory generated by minisnap is collision-free.
- Improved Conditional Sampling. Any state that can be part of the solution path must have a finite distance $d$ from the starting point and must be in contact with at least one target state. Due to velocity constraints, only states within the intersection of the start velocity region and the target velocity region satisfy this requirement shows in Fig.(2). Therefore, similar to Informed RRT*, we sample only from the region where solutions can be generated. Ideally, sampling should occur directly from the union of the start velocity region and the target velocity region.

### B. Reformulation Dynamic Coordinate Minimum Snap Trajectory Optimization

For the state sequences generated by TubeST-RRT*, we use the proposed RDCMS trajectory optimization to smooth the trajectories. In traditional minimum snap trajectory optimization, the optimized trajectory needs to pass through intermediate path nodes. However, since our TubeST-RRT*

**Algorithm 1** TubeST-RRT*

**Input:** $\mathcal{Q}, q_{start}, x_{goal}, p_{goal}, range\_d, Param, ptc$
**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**
  $q_{rand} \leftarrow SampleConditionally(s_{start}, \mathcal{Q}, B, d)$
  $r_{rand} \leftarrow FindMaxRadius(q_{rand})$
  $s_{rand} \leftarrow \{q_{rand}, r_{rand}\}$
  $s_{nearsst} \leftarrow Nearset(s_{rand}, T_a)$
  $s_{new} \leftarrow TubeSTSteer(s_{nearest}, s_{rand}, d)$
  **if** $MotionCheck(s_{nearsst}, s_{new})$ **then**
    $RewireTree(T_a, x_{new})$
    **if** $Connect(T_b, x_{new}, d) = Reached$ **then**
      $solution \leftarrow UpdateSolution(x_{new})$
      $t_{\max} \leftarrow CostPath(solution)$
      $PruneTrees(t_{\max}, T_a, T_b)$
    **end if**
  **end if**
  $Swap(T_a, T_b)$
**end while**
**return** $Solution$

incorporates collision detection within the spherical regions, it 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 ($a$). Inequality constraints can be added to $p$ to ensure it stays within the tube area generated by the planner. This ensures that the optimized trajectory is collision-free. Input as

$$\boldsymbol{S} = [\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_N] \tag{1}$$

where,$\boldsymbol{s}_i = [\boldsymbol{p}_i, \boldsymbol{v}_i, t_i, r_i]$ A quintic polynomial is used to fit a uniaxial trajectory.

$$\boldsymbol{p}(t) = p_0 t^0 + p_1 t^1 + \cdots + p_5 t^5 \tag{2}$$

For the $N+1$ points generated by the planner (including the start point and the end point), it can be assumed that there are $N$ segments of the trajectory, where each segment of the trajectory is a higher-order polynomial, for RDCMS, our cost function is

$$J = J_1 + J_2 + \cdots + J_N = \sum_{n=0}^{M} p_m^\top Q_m p_m \tag{3}$$

where,$J_m = \int_{T_{m-1}}^{T_m} \left(\frac{d^4 p(t)}{dt^4}\right)^2$ However, as the polynomial coefficients do not have practical physical meanings, when N is large, or in some special cases, this optimization problem can easily become numerically unstable, leading to difficulties in solving. To address this issue, a reformulation method was proposed to map the optimization variables to the physically meaningful variables $(p, v, a)$ [8].

$$\boldsymbol{A}_i \boldsymbol{p}_i = \boldsymbol{d}_i, \quad \boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_T \end{bmatrix}_i, \boldsymbol{d}_i = \begin{bmatrix} \boldsymbol{d}_0 \\ \boldsymbol{d}_T \end{bmatrix}_i \tag{4}$$

The mapping relationship between the polynomial coefficients and the motion differential has been obtained.

$$\begin{aligned} \boldsymbol{d}_i &= \boldsymbol{A}_i \boldsymbol{p}_i \\ \boldsymbol{p}_i &= \boldsymbol{A}_i^{-1} \boldsymbol{d}_i \end{aligned} \tag{5}$$

where, $\boldsymbol{d}_i$ is a vector including the derivative values of the start $\boldsymbol{d}_0$ and the end $\boldsymbol{d}_T$ of the $i$ segment. If the specific derivative is not known, a continuity constraint needs to be imposed to ensure that the derivative at the end of the $i$ segment matches the derivative at the beginning of the $(i+1)$ segment:

Replace the coefficient vector of the original cost function through the mapping matrix $\boldsymbol{A}_m$ to obtain the following equation

$$\begin{aligned} J &= \begin{bmatrix} \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_N \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{Q}_0 & 0 & \cdots & \cdots & 0 \\ 0 & \boldsymbol{Q}_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{Q}_{\mathrm{N}} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_N \end{bmatrix} \\ &= \boldsymbol{d}^\top \boldsymbol{A}^{-\top} \boldsymbol{Q} \boldsymbol{A}^{-1} \boldsymbol{d} \end{aligned} \tag{6}$$

Introduce dynamic coorid constraints and continuity constraints for the information of the non-collision circle radius in the nodes calculated by the planner.

For the processing of the continuity constraint, the permutation matrix (permutation matrix) $C$ can be used to introduce the continuity constraint into the cost function.

$$\begin{aligned} J &= \begin{bmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{bmatrix}^\top \boldsymbol{C} \boldsymbol{A}^{-\top} \boldsymbol{Q} \boldsymbol{A}^{-1} \boldsymbol{C}^\top \begin{bmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{FF} & \boldsymbol{R}_{FP} \\ \boldsymbol{R}_{PF} & \boldsymbol{R}_{PP} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{bmatrix} \end{aligned} \tag{7}$$

where, $\boldsymbol{d}_F$ is a fixed variable, which is directly determined by the $p$, $v$, and $a$ of the initial point and the end point here. $\boldsymbol{d}_P$ is a variable that needs to be optimized and solved, so the cost function can be rewritten as follows.

$$J = 2\boldsymbol{d}_F^\top \boldsymbol{R}_{FP} \boldsymbol{d}_P + \boldsymbol{d}_P^\top \boldsymbol{R}_{PP} \boldsymbol{d}_P \tag{8}$$

After changing the optimization variable from the polynomial coefficient to the $p$, $v$, $a$ variables with practical

significance, it becomes very simple to add dymaic coorid constraints and speed and acceleration constraints, and only the state variable constraints need to be added. Limiting the position variable of the intermediate state to a coorid box for optimization can not only further smooth the trajectory, but also effectively prevent the occurrence of the "slewing" phenomenon.Thus, the inequality constraints can be written in the following form

$$\begin{aligned} p_n - coorid_i <& p_n < p_n + coorid_i \\ v_{\min} <& v_n < v_{\max} \\ a_{\min} <& a_n < a_{\max} \\ n = 1, 2, \cdots, N-1 & \end{aligned} \tag{9}$$

where, $coorid_i$ comes from $r_i$ in the state variable of the tubeSTRRT planner.

For the optimization variable $\boldsymbol{d}$.

$$\boldsymbol{d} = \begin{bmatrix} d_0 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} p_0 \\ v_0 \\ a_0 \\ p_1 \\ v_1 \\ a_1 \\ \vdots \\ p_N \\ v_N \\ a_N \end{bmatrix} \tag{10}$$

only has state constraints.

$$\boldsymbol{d}_{Pmin} \leq \boldsymbol{d}_P \leq \boldsymbol{d}_{Pmax} \tag{11}$$

Since we have time information for each node, there is no need to use the time allocation algorithm anymore. Compared to the minimum snap problem without variable conversion, this way not only avoids the problem of numerical stability of polynomial coefficient, but also the solution speed has been greatly improved.

## IV. Simulations

The effectiveness of the proposed planning strategy is verified through numerical simulations. The simulations are divided into two scenarios: a dynamic environment to demonstrate the algorithm's effectiveness in dynamically avoiding global moving obstacles, and a static environment to showcase the superiority of the flight corridor-improved trajectory optimizer

## V. Conclusion

## References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

## References


[1] B. Siciliano and O. Khatib, Springer handbook of robotics. Springer, 2016.

[2] F. Grothe, V. N. Hartmann, A. Orthey and M. Toussaint, "ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 3314-3320

[3] M. Otte and E. Frazzoli, "Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning," International Journal of Robotics Research, vol. 35, no. 7, pp. 797–822, 2016.

[4] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Rt-rrt* a real-time path planning algorithm based on rrt," in ACM SIGGRAPH Conference on Motion in Games, 2015, pp. 113–118.

[5] C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay, "Risk based motion planning and navigation in uncertain dynamic environment," Research Report, Oct. 2010.

[6] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2014, pp. 6745–6750.

[7] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), vol. 2, 2000, pp. 995–1001.

[8] C. Richter, A. Bry and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments" in Robotics research, Springer, pp. 649-666, 2016.