

# 过驱动飞行器的轨迹规划

刘培焱

学 院：机电工程与自动化学院 专 业：自动化

学 号：180310211 指导教师：陈浩耀教授

2022 年 6 月

# 哈尔滨工业大学深圳校区

## 毕业设计（论文）

题 目 过驱动飞行器的轨迹规划

姓 名 刘培焱

学 号 180310211

学 院 机电工程与自动化学院

专 业 自动化

指 导 教 师 陈浩耀教授

答 辩 日 期 2022 年 06 月 10 日

## 摘要

进入 21 世纪以来，多旋翼无人机领域取得了很大的发展，成为一类成功从实验室走进人们生活的机器人系统。为突破现在市场上主流的欠驱动无人机所存在的瓶颈，近年来人们又研制出了不少种类过驱动多旋翼无人机系统。作为一种全驱动系统，这种无人机可以跟踪 6 自由度轨迹，有效增强了多旋翼无人机的机动性能，拓宽了其应用场景。可以想到，这会让多旋翼飞行器在复杂、拥挤的环境中的自主飞行能力大幅提升，而有效的规划算法是实现这种提升的关键。现有的规划方案主要是基于传统欠驱动飞行器开发的，在过驱动飞行器上鲜有先例。

本文以实现过驱动多旋翼飞行器在复杂环境中的避障规划为目标，为过驱动多旋翼飞行器设计了一个基于优化的全自由度全局轨迹规划器。该规划器根据由一系列凸多面体表示的安全约束，结合飞行器的整体形态以及动力学约束，最终规划出一条由起点到终点的无碰撞最优轨迹。

本文理论部分介绍了规划器前端路径搜索算法以及安全约束生成算法的设计和实现，并且详细讲述了后端轨迹优化算法的原理和设计。其中在后端优化算法部分分别介绍了基于欧拉角和基于四元数的两种姿态规划方案，并在附录中给出了相应公式的推导。

在实验部分中，本文以一种全向六旋翼飞行器机构为平台，针对不同的仿真场景进行了规划实验，以验证和对比基于上述两种姿态规划方案所设计规划器的可行性、有效性和计算效率。随后基于 Gazebo 和 PX4 进行了仿真实验，利用本文设计的规划器成功实现了全向六旋翼飞行器在狭长通道等极端环境中的避障飞行。最后在实物平台上进行实验，验证了规划器的实用性。

**关键词：**过驱动飞行器；轨迹规划；避障，SE(3) 规划

## Abstract

Since entering the 21st century, the field of multi-rotor UAV has achieved unprecedented development and become a kind of robot system that has successfully entered people's life from the laboratory. In order to break through the bottleneck of underactuated UAV in the current market, many kinds of over-actuated multi-rotor UAV systems have been developed in recent years. As a fully actuated system, the UAV can track a 6-DoF trajectory, effectively enhancing the maneuverability of the multi-rotor UAV and broadening its application scenarios. As you can imagine, this would greatly improve the autonomous flight capability of multi-rotor aircraft in complex, crowded environments, and an effective planning algorithm is the key to achieving this improvement. The existing planning schemes are mainly based on traditional underactuated aircraft, and there are few precedents for overactuated aircraft.

In order to realize obstacle avoidance planning of over-actuated multi-rotor aircraft in complex environment, a global trajectory planner with full degree of freedom based on optimization is designed for over-actuated multi-rotor aircraft. According to the safety constraints represented by a series of convex polyhedra, combined with the whole-body shape and dynamics constraints of the aircraft, the planner finally plans an optimal collision free trajectory from the starting point to the end point.

The theoretical part of this thesis introduces the design and implementation of the front-end path search algorithm and the security constraint generation algorithm of the planner, and describes the principle and design of the back-end trajectory optimization algorithm in detail. In the part of the back-end optimization algorithm, two attitude planning schemes based on Euler Angle and quaternion are introduced respectively, and the derivation of corresponding formulas is given in the appendix.

In the experimental part, planning experiments are carried out for different simulation scenarios with an omnidirectional hexarotor aircraft mechanism as the platform to verify and compare the feasibility, effectiveness and computational efficiency of the planners designed based on the above two attitude planning schemes. Then, simulation experiments are carried out based on Gazebo and PX4, and the obstacle avoidance flight of omnidirectional hexarotor aircraft in narrow and long passage and other extreme environments is successfully realized by using the planner designed in this thesis. Finally, experiments are carried out on a physical platform to verify the practicality of the planner.

**Keywords:** over-actuated aircraft, trajectory planning, obstacle avoidance, SE(3) planning

## 目 录

摘要 .....	I
ABSTRACT .....	II
第1章 绪论 .....	1
1.1 课题背景及研究的目的和意义 .....	1
1.1.1 课题背景 .....	1
1.1.2 研究的目的和意义 .....	2
1.2 国内外研究现状 .....	2
1.2.1 欠驱动多旋翼飞行器的避障轨迹规划 .....	3
1.2.2 过驱动多旋翼飞行器 .....	5
1.2.3 过驱动飞行器的轨迹规划 .....	6
1.3 本文的主要研究内容 .....	6
第2章 过驱动多旋翼飞行器的结构设计与动力学分析 .....	8
2.1 引言 .....	8
2.2 过驱动多旋翼飞行器的设计 .....	8
2.2.1 机械结构设计 .....	8
2.2.2 航电系统设计 .....	10
2.3 OmniHex 的动力学分析 .....	11
2.3.1 动力学建模与微分平坦性分析 .....	12
2.4 本章小结 .....	17
第3章 前端路径搜索算法及安全约束生成算法设计 .....	18
3.1 引言 .....	18
3.2 路径搜索算法的设计 .....	18
3.2.1 路径搜索算法概述 .....	18
3.2.2 $SE(3)$ 空间中的 RRT 算法 .....	19
3.2.3 近邻搜索 .....	22
3.2.4 有效性检测 .....	25
3.2.5 算法效果 .....	27
3.3 3D 安全飞行走廊生成算法的设计 .....	28
3.3.1 算法原理 .....	29
3.3.2 效果展示 .....	30
3.4 本章小结 .....	31

<b>第 4 章 后端轨迹生成算法设计 .....</b>	<b>32</b>
4.1 引言 .....	32
4.2 几何约束下多旋翼的轨迹优化原理 .....	32
4.2.1 问题形式 .....	32
4.2.2 最优性条件与 MINCO 轨迹类 .....	34
4.2.3 MINCO 轨迹类中的梯度计算 .....	36
4.2.4 几何约束的消去 .....	37
4.2.5 连续时间约束 $\mathcal{G}$ 的处理 .....	40
4.3 几何约束下过驱动飞行器的 $SE(3)$ 轨迹优化算法设计 .....	41
4.3.1 优化问题构建 .....	42
4.3.2 基于欧拉角姿态表示的过驱动飞行器 $SE(3)$ 轨迹生成 .....	45
4.3.3 基于四元数姿态表示的过驱动飞行器 $SE(3)$ 轨迹生成 .....	46
4.3.4 算法实现 .....	46
4.4 本章小结 .....	47
<b>第 5 章 规划器测试与实验 .....</b>	<b>48</b>
5.1 引言 .....	48
5.2 规划器效果测试 .....	48
5.2.1 随机地图中的轨迹规划 .....	48
5.2.2 狹窄空间中的轨迹规划 .....	51
5.3 仿真避障测试 .....	52
5.4 实物避障测试 .....	52
5.5 本章小结 .....	52
<b>结 论 .....</b>	<b>53</b>
<b>参考文献 .....</b>	<b>54</b>
<b>原创性声明 .....</b>	<b>59</b>
<b>致 谢 .....</b>	<b>60</b>
<b>附录 A 罚函数中正求和项的梯度 .....</b>	<b>61</b>
<b>附录 B 基于欧拉角姿态表示的罚函数梯度 .....</b>	<b>62</b>
<b>附录 C 基于四元数姿态表示的罚函数梯度 .....</b>	<b>63</b>

# 第1章 绪论

## 1.1 课题背景及研究的目的和意义

### 1.1.1 课题背景

近些年来，随着控制科学、感知导航以及人工智能等技术的不断进步，关于多旋翼无人机（multirotor UAV）的设计及应用的领域展现出了空前的繁荣，并取得了长足的发展。多旋翼无人机凭借其简洁的机械结构以及优秀的飞行稳定性，从一众智能机器人中脱颖而出，从实验室里走进了千家万户。在许多地方，如日常生活中的航拍，或是农林、检修和军事行动等特种作业场合中，都有多旋翼无人机的用武之地（图1-1）。



(a) 用于航拍测绘



(b) 用于农林植保



(c) 用于电力巡检



(d) 用于后勤补给

图 1-1 多旋翼无人机的多种应用场景

然而，从机械结构中可以发现，目前所应用的大多数多旋翼无人机的各个旋翼的转轴均互相平行且垂直于机身平面，所以旋翼只能为飞行器提供垂直机身平面的推力。这就使得飞行器成为一个欠驱动系统（under-actuated system）<sup>[1]</sup>，其位置控制和姿态控制是耦合的。具体表现为，如果飞行器需要水平飞行，则必须倾

斜机身才能获得水平方向的分力，以提供水平方向的加速度和克服空气阻力；类似地，飞行器只有在水平姿态下才可以保持悬停。这种特性很大程度上限制了欠驱动多旋翼飞行器的应用场景：在空中作业时，悬停姿态的限制会使飞行器无法克服机载机械臂或机载云台的机械死角；在拥挤的环境中，姿态与推力的耦合也会严重削弱飞行器的避障性能，使环境中的可行区域大幅缩减。

为了改善上述问题，充分发掘多旋翼飞行器的潜力，近年来发展出了多种能将飞机的位置与姿态控制解耦的多旋翼飞行器。这些工作通过改变旋翼几何构型、增加旋翼倾转自由度等方式让旋翼能提供相对机身任意方向的推力和转矩，这赋予了多旋翼飞行器跟踪 6 自由度全状态轨迹的能力，使飞行器成为全驱动系统 (fully actuated system)。这种飞行器能够分别独立地进行平移和旋转运动，故又称为全向飞行器 (omnidirectional aerial vehicle)，这种受控的、自由的刚体运动是传统欠驱动多旋翼飞行器所无法做到的。所谓过驱动飞行器，就是具有冗余控制输入（驱动器）的全驱动飞行器，在少数驱动器故障的情况下依然能够进行控制，增强了飞行器的容错能力。

### 1.1.2 研究的目的和意义

由于上述过驱动多旋翼飞行器的位置和姿态可以独立控制，故可以为其独立规划位置和姿态共 6 个自由度的轨迹，其飞行的灵活性相比于只能跟踪 4 自由度轨迹的传统欠驱动多旋翼飞行器将大幅提高。设想当面对一个狭长笔直、宽度小于自身旋翼间最小距离的通道时，加速度与姿态耦合的欠驱动飞行器显然无法通过；而过驱动飞行器则可以通过控制姿态使机身倾斜以适应通道内的狭小空间，同时控制位置从而实现平稳无碰撞的穿越。这样的优势使得过驱动飞行器在未知环境探索、巷战侦查与打击、灾区废墟搜救等场合势必会发挥出很大的应用价值。要更好地挖掘过驱动多旋翼飞行器的潜力、提高其自主飞行性能，为其设计一套在复杂狭小空间内进行避障轨迹规划的算法是具有重要意义的。

本文以实现过驱动多旋翼飞行器在复杂环境中的避障飞行为目标，对基于过驱动多旋翼飞行器的  $SE(3)$  运动规划算法展开研究，设计了一个考虑飞行器整体形状和动力学约束的全局轨迹规划器，并设计仿真和实物实验验证了所设计规划器的可行性，为相关领域探索了一条可行的道路，且有望后续通过并行计算及对算法进行优化等手段，配合机载感知系统实现实时规划。

## 1.2 国内外研究现状

轨迹规划模块处于无人机软件框架的中间位置，其接收感知模块传来的地图

等上游信息，输出一条安全、平滑的可行轨迹输送给控制模块。规划模块起到了承上启下的作用，可以说是影响无人机自主飞行性能的最重要的一环；另外，在多旋翼飞行器全驱动化方面，目前也有不少种方案可以参考。本节将从欠驱动多旋翼飞行器的避障轨迹规划、过驱动多旋翼飞行器和过驱动多旋翼飞行器的轨迹规划三方面讲述国内外研究现状。

### 1.2.1 欠驱动多旋翼飞行器的避障轨迹规划

多旋翼飞行器的避障轨迹规划这一领域近年来取得了丰富的成果。大多数多旋翼飞行器属于微分平坦系统 (differential flat system)<sup>[2]</sup>，这种系统的运动规划问题可以在保证一定的平滑性的前提下转化为较低维度的优化问题，其平坦输出的轨迹可以表示为分段多项式，这样就能方便地获得轨迹关于时间的各阶导数，进而利用微分平坦关系求得飞行器的各状态变量和控制输入。

2011年，Mellinger等人提出了为四旋翼飞行器生成固定间隔多项式轨迹的方法<sup>[3]</sup>，通过求解一个由加速度的二阶导数（即 snap）的平方积分代价以及线性安全约束构造的二次规划 (Quadratic Programming, QP) 问题，最终生成一条平滑且安全的轨迹。2015年，Bry等人推导出了导数平方积分形式无约束二次规划问题的闭式解<sup>[4]</sup>，并且启发式地在前端 RRT\* 搜索出的可行路径上添加路点直到无碰撞为止，以此来保证轨迹的安全性。2020年，Gao等人将环境中的安全区域表

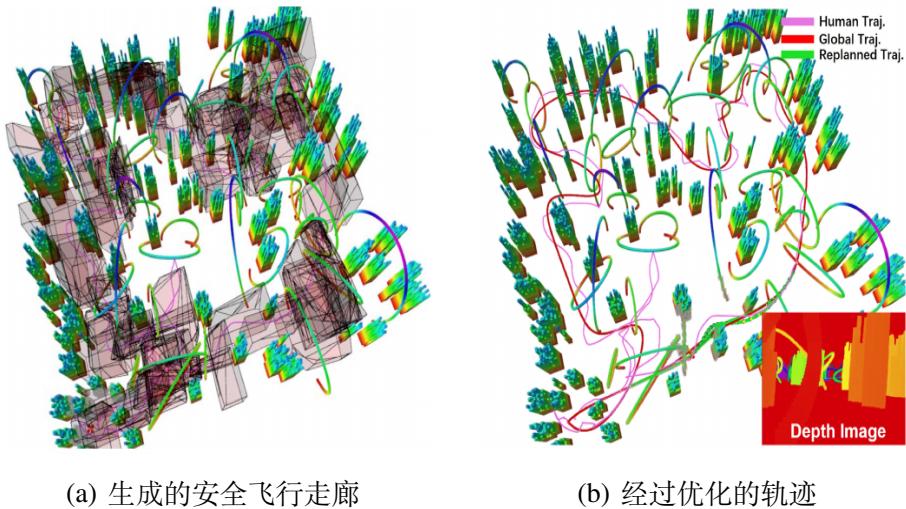


图 1-2 Teach-Repeat-Replan 示意图<sup>[5]</sup>

示为凸多面体<sup>[5]</sup>，一系列凸多面体连接起来构成安全飞行走廊 (safe flight corridor, SFC) (图1-2)，轨迹的安全性由 Bézier 曲线的凸包性质来保证。2021年，Zhou 等人提出了 EGO-Planner，与多数基于梯度的轨迹规划方法不同，EGO-Planner 直接

从障碍物上获得梯度信息，从而免去了对 ESDF 地图的依赖，大大降低了内存消耗和计算量。

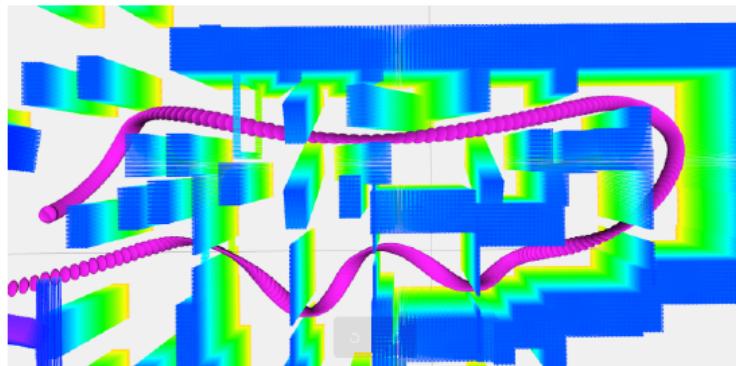


图 1-3 在  $SE(3)$  空间中规划出的轨迹<sup>[6]</sup>

以上工作都能高效地进行轨迹规划，然而它们都是在构型空间（configuration space）中进行规划，并未考虑无人机的几何形状和姿态，具有很大的保守性。在某些比较极端的环境中，存在需要无人机倾斜机身才能飞过的狭窄缝隙，此时传统的  $\mathbb{R}^3$  规划器是难以发挥作用的。要得到如图1-3所示通过改变姿态以实现在较为拥挤的环境中无碰撞的安全轨迹，就需要在  $SE(3)$  空间中进行规划。2018 年，Liu 等人提出了一种基于搜索的  $SE(3)$  规划算法<sup>[7]</sup>，其通过在一定的时间间隔内施加常量控制输入来生成运动基元（motion primitive），配合可行性检测器筛选出与点云无交集的运动基元，最终搜索出一条无碰撞轨迹。不过，该算法存在着组合爆炸（combinatorial explosion）问题：要想获得更高的求解质量，就得提高分辨率，而随着分辨率的增高，计算量会快速增加，带来不可接受的计算时间和内存消耗，故这种算法实用性欠佳。

2022 年，Wang 等人提出了一个几何约束下的多旋翼飞行器轨迹优化框架<sup>[8]</sup>。该框架首先基于多阶段最优控制问题的最优充要条件，构建了名为 MINCO 的一类分段多项式轨迹，这类轨迹以其必须经过的中间点和每段的时间分配作为参数，轨迹的系数矩阵可以根据最优条件以线性的时间和空间复杂度计算出来，这相较于 Bry 等人提出的闭式解<sup>[4]</sup>有了很大改进；使用 MINCO 轨迹类，就可以直接对中间点和时间分配进行优化，同时还能保证轨迹的平滑性。该框架还通过一系列技巧消去了对时间分配的约束以及对中间点的空间约束，并将其余自定义约束软化为目标函数中的惩罚项，最终将整个带约束的轨迹优化问题转化为无约束优化问题，得以高效求解。该框架的提出为  $SE(3)$  规划问题提供了新思路，紧接着 Yang 等人构造了考虑四旋翼飞行器的形状与姿态，并使飞行器整体包含在表示安全区域的凸多面体内的约束形式<sup>[9]</sup>，该约束作为框架中的自定义约束来处理；Han 等人进一步开发了用于无人机竞速的  $SE(3)$  规划器<sup>[6]</sup>，并将其开源<sup>①</sup>。

<sup>①</sup> <https://github.com/ZJU-FAST-Lab/Fast-Racing>

### 1.2.2 过驱动多旋翼飞行器

实现多旋翼飞行器全驱动化的要点在于使驱动器同时且独立地产生相对机体任意方向推力和力矩，而要做到这一点需要改变旋翼的构型（configuration of rotors）。过驱动全向多旋翼飞行器领域近年来正处于持续发展中，这些全向飞行器按照旋翼构型大致可分为两类：固定旋翼型（fixed-rotor）<sup>[10-12]</sup> 和倾转旋翼型（tiltrotor）<sup>[13-15]</sup>。图1-4分别展示了两种构型中具有代表性的一项工作。

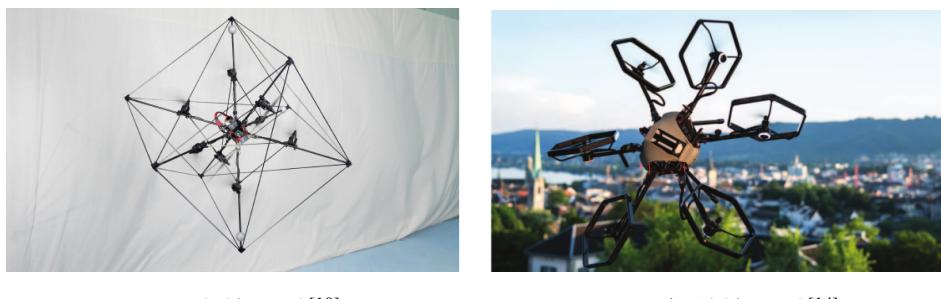
(a) 固定旋翼型<sup>[10]</sup>(b) 可倾转旋翼型<sup>[14]</sup>

图 1-4 两种不同种类旋翼构型的全向多旋翼飞行器示例

固定旋翼的过（全）多旋翼飞行器的机械结构相对简单，飞行器通过改变不同朝向旋翼的转速来控制推力和转矩的大小和方向。如图1-4 (a)所示为 Brescianini 等人于 2016 年发表的一种实现灵活全向飞行的固定旋翼型八旋翼飞行器系统<sup>[10]</sup>，该系统采用 8 个可逆电机-旋翼组合执行器（reversible motor-propeller actuator），故每个执行器都可以产生正推力和负推力，8 个执行器的构型是基于静态力和扭矩分析，采用求解优化问题的方式来设计的，以期最大化飞行器的灵活性并最大限度保证飞行器动力学特性的旋转不变性。不过，固定旋翼型的过驱动飞行器的一个主要缺点在于：这些旋翼通常不会同时直接朝向竖直方向，使得这类飞行器的悬停效率不会很高；并且如果设定旋翼朝向使之更倾向于高效的悬停和更高的有效载荷，就会几乎不可避免地降低产生横向作用力的能力<sup>[12]</sup>。

可倾转旋翼型全向多旋翼飞行器的实现方式多数是为旋翼增加额外的自由度，使其转轴指向可以改变。比较常用的方式是为安装旋翼的机臂添加一个绕其轴的旋转自由度：如图1-5展示了 Markus 等人于 2015 年研制出的一种可倾转旋翼的过驱动四旋翼飞行器的结构<sup>[13]</sup>，该飞行器可以在有限的横滚角和俯仰角下实现悬停；图1-4 (b)展示的是 Kamel 等人于 2018 年开发的一种可倾转旋翼的全向六旋翼飞行器 Voliro<sup>[14]</sup>，该飞行器可以实现任意姿态下的悬停和飞行。这类飞行器通过改变每个旋翼的朝向，实现了更高效的悬停。本课题所使用的仿真及实物实验平台是参考 Voliro 的结构搭建的。

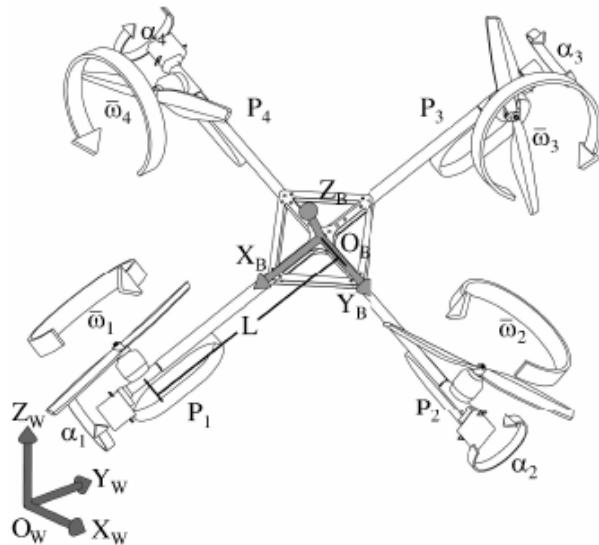


图 1-5 Markus Ryll 等人的可倾转旋翼的全向四旋翼飞行器结构示意图

### 1.2.3 过驱动飞行器的轨迹规划

现有的关于过(全)驱动飞行器的研究主要集中于机械结构设计和飞行控制算法方面,而为过驱动飞行器进行轨迹规划的相关成果并不太多。2018年,Brescianini等人基于运动基元为过驱动飞行器生成了从给定起点到给定终点且满足一定动力学约束的6自由度轨迹<sup>[16]</sup>。同年,Morbidi等人提出了用于双轴倾转旋翼六旋翼飞行器的节能轨迹生成方法<sup>[17]</sup>,通过求解一个显式考虑电机电气模型的优化控制问题得到一条指定边界点的节能轨迹,并做了数值验证。2021年,Pantic等人提出了基于流形网格的运动规划方法<sup>[18]</sup>,该方法将物体表面(surface)建模为三角网格(triangular mesh)并提出原始表面的一个低维参数化表示方法,进一步将原始表面及其低维表示近似为流形(manifold),运用黎曼运动策略(Riemannian Motion Policies,RMPs)构建了一个高效且通用的运动规划框架;该方法目的在于生成一条使飞行器飞向指定表面并沿其飞行,应用于全向飞行器与曲面交互的场景。

## 1.3 本文的主要研究内容

从国内外研究现状中可以看出,目前已有许多成熟的方案能赋予多旋翼无人机跟踪6自由度轨迹、实现全向飞行的能力。在避障轨迹规划方面,以欠驱动多旋翼飞行器为平台的解决方案同样多样且成熟,其中在复杂空间内进行  $SE(3)$  避障规划已经有了效果拔群的成果;然而,在能够进行全向飞行的过驱动多旋翼飞行器平台上进行轨迹规划的成果并不多,其中多数<sup>[16,17]</sup>并未考虑障碍物存在情况下的安全约束,Pantic等人的方法<sup>[18]</sup>也并不太适用于避障。事实上,面向复杂

环境中过驱动飞行器的  $SE(3)$  轨迹规划的解决方案几乎还没有已发表的成果。传统基于欠驱动多旋翼飞行器的轨迹规划方法固然可以应用到过驱动飞行器上，但由于这些方法并不能生成 6 自由度轨迹，所以它们并不能充分发挥过驱动飞行器的避障潜能。

本文重点研究过驱动多旋翼飞行器在复杂环境中的  $SE(3)$  轨迹规划。通过深入研究、对比各种已有的轨迹规划方案，结合过驱动飞行器的不同性质，完成规划算法设计、代码实现、仿真验证及实物验证。本文主要研究内容如下：

- (1) 参考过驱动飞行器系统 Voliro<sup>[14]</sup>，搭建了全向六旋翼飞行器 OmniHex 的仿真和实物模型，并对其动力学模型进行研究和分析。
- (2) 针对前端路径搜索和安全飞行走廊生成的问题，研究已有方案和开源代码，设计  $SE(3)$  空间中的 RRT 算法用于前端路径搜索，设计凸多面体安全飞行走廊生成算法用于空间约束的生成。
- (3) 为实现复杂环境中的 6 自由度  $SE(3)$  轨迹规划，对姿态规划方式及后端轨迹优化问题展开研究，设计了基于欧拉角和基于四元数的两种姿态规划方式，并分别设计对应的安全约束和动力学约束形式，推导对应的罚函数及其梯度。
- (4) 将所设计的算法实现为 C++ 接口，并对其进行整合，实现为基于 ROS<sup>[19]</sup> 的全局规划器，并基于 Gazebo<sup>[20]</sup> 和 PX4<sup>[21]</sup> 进行仿真实验、基于实物平台进行实物实验，以验证其可行性。

## 第2章 过驱动多旋翼飞行器的结构设计与动力学分析

### 2.1 引言

过驱动多旋翼飞行器的结构和动力学模型相较传统欠驱动多旋翼飞行器而言有较大不同，为了科学地为其设计运动规划算法，有必要对其结构和动力学模型进行分析。本章 2.2 节对本课题所使用的过驱动多旋翼平台 OmniHex 的设计进行简要介绍；2.3 节建立 OmniHex 的动力学模型，并分析其微分平坦性质。

### 2.2 过驱动多旋翼飞行器的设计

成功搭建一台过驱动飞行器的要点在于为其设计合理的机械与电子系统，本节就从机械结构设计与航电系统设计两方面对本课题所搭建的 OmniHex 飞行器（图2-1）进行介绍。



图 2-1 OmniHex 飞行器实物示意图

#### 2.2.1 机械结构设计

OmniHex 采用类似 Voliro 的可倾转旋翼构型，图2-2展示了其机械结构外观，并对重要部件做出了标注。OmniHex 的每个旋翼除了可以绕电机轴进行旋转以提供推力外，其指向还可以在舵机的驱动下随着机臂绕机臂轴线旋转，如图2-2 (b)和图2-2 (c)所示，驱动旋翼的无刷电机与机臂固连在一起，所以旋翼与其所在的机臂是同时倾转的；舵机安装于组成机身的两块中心板之间，且个舵机都位于其所驱动

的机臂根部，二者之间由一个舵机-机臂连接件进行桥接，舵机的舵盘旋转时，将动力传递给连接件，连接件再将动力经由固定用的螺栓传递给机臂。为防止机臂倾转造成线缆缠绕，需要连接无刷电机与电子调速器（固定于机身）的三相线在机臂位于倾转零点时预留有一定长度。

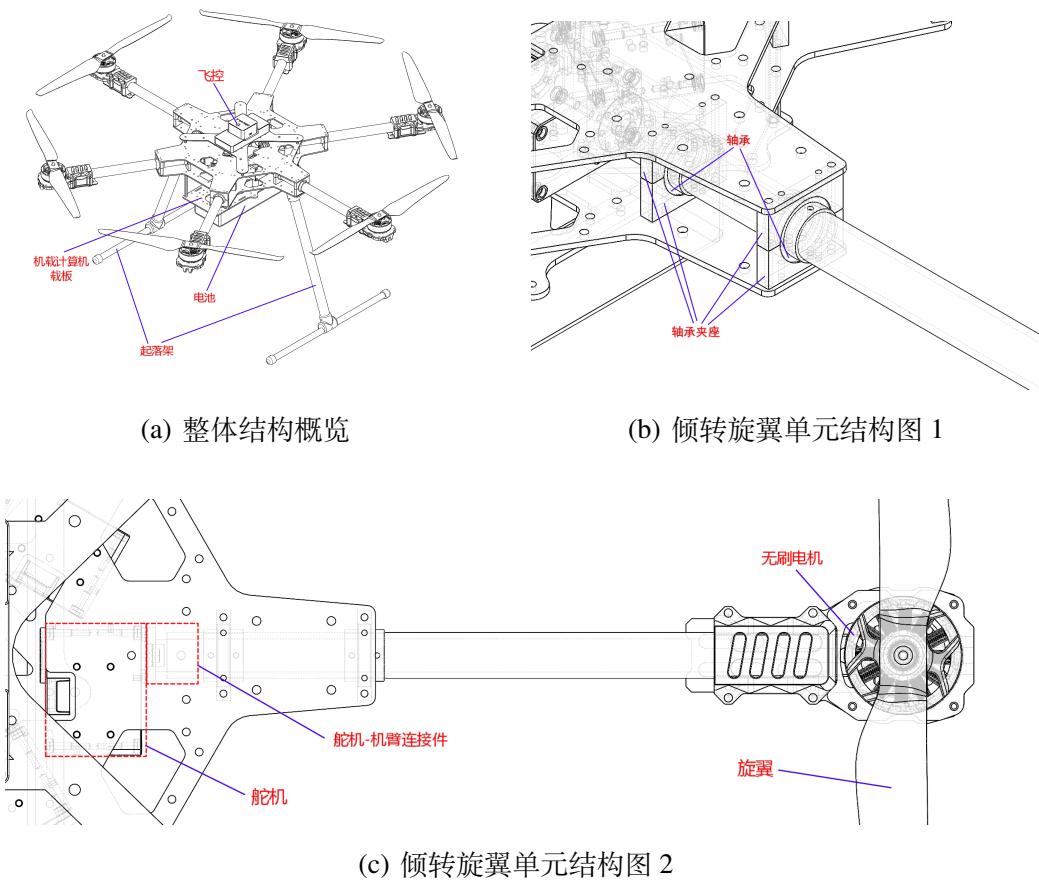


图 2-2 OmniHex 机械结构示意图

每一个旋翼-无刷电机组合，与其所在的机臂及相应舵机一起组成了一个倾转旋翼单元，为飞行器带来了两个机械自由度，OmniHex 上安装了 6 个独立的倾转旋翼单元，故拥有 12 个机械自由度。每个倾转旋翼单元不仅在舵机处与机身相连，为了防止组成机臂的柔软碳管发生弯曲形变，还在每个机臂与机身之间加入了两个由滚珠轴承形成的旋转运动副，每个轴承的位置由一对定制的轴承夹座和一对套在机臂上的阶梯圈所固定，如图2-3所示为安装方式的特写。

在充分考虑动力要求、结构强度等性能指标后，最终确定 OmniHex 主要机械部件的选型（材）如表2-1所示。

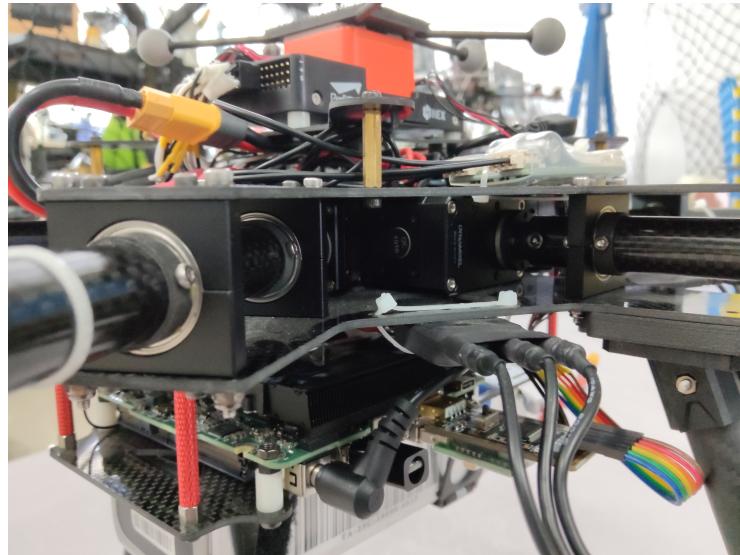


图 2-3 倾转旋翼单元与机身之间的安装方式特写

表 2-1 OmniHex 主要机械部件的选型 (材)

部件	材料 (型号)
舵机	Dynamixel-XH340
无刷电机	
旋翼	EOLO 1350
机臂	碳纤维
中心板及各种载板	碳纤维
轴承夹座	铝合金
舵机-机臂连接件	高性能尼龙
阶梯圈	高性能尼龙

## 2.2.2 航电系统设计

OmniHex 的航电系统由飞行控制器、机载计算机、电子调速器、无线通信设备等组成。

飞行控制器采用 PixHawk CubeOrange 开源飞控（图2-4），运行 PX4-Autopilot 固件<sup>①</sup>，主要承担姿态解算、位置姿态控制及控制分配等工作，可根据输入的控制指令来输出相应的执行器指令；电子调速器根据飞控发送来的 PWM 信号控制无刷电机的转速；无线通信设备为一个遥控器信号接收机，其接收遥控器发来的指令并进行解析，随后发送给飞控；

机载计算机使用 Intel NUC11PAHi7 迷你电脑（图2-5），主要运行基于 ROS 的软件系统，如 Optitrack 动作捕捉系统驱动、Dynamixel 舵机驱动、轨迹生成模块及 OffBoard 控制模块等，其与飞行控制器之间通过串行接口（serial interface）连接，使用 MAVLink 协议和 RTPS 协议进行通信。

① <https://github.com/PX4/PX4-Autopilot>



图 2-4 PixHawk CubeOrange 飞控



图 2-5 Intel NUC11PAHi7 迷你电脑

OmniHex 暂未部署机载的感知系统，目前使用动作捕捉系统的反馈模拟视觉里程计（visual odometry, VO）的数据。机载计算机基于 VRPN<sup>[22]</sup> 与 OptiTrack 动作捕捉系统进行无线通信，并将处理后的信号发送给飞控用于位姿估计。

只有上述各个软硬件模块正常工作、协同配合，才能保证飞行器的稳定飞行。OmniHex 的软硬件框架以及模块间的主要数据流向概括为图2-6。

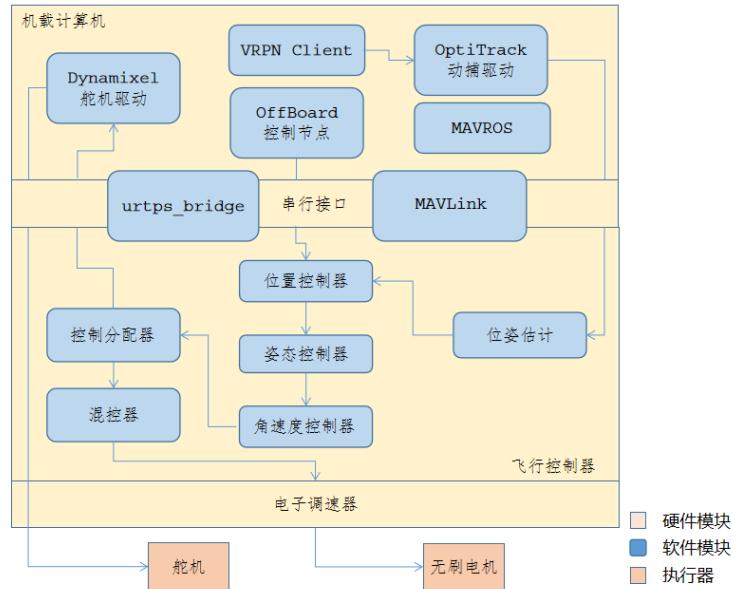


图 2-6 OmniHex 软硬件架构与数据流向简图

### 2.3 OmniHex 的动力学分析

本节首先建立 OmniHex 飞行器的动力学模型，并在动力学模型的基础上分析 OmniHex 的微分平坦性质，给出平坦输出及相应的微分平坦关系。

为简化模型，且不失模型的实用性，如果不作特别说明，本节及本文后续章节的分析均基于以下假设：

- (1) 机体结构近似为刚性且对称的，每个旋翼的推力相对质心的力臂长度均为  $l$ ；
- (2) 每个旋翼产生的推力和反扭矩正比与其转速的平方，且电机驱动旋翼达到目标转速的调整时间忽略不计；
- (3) 各个旋翼所产生的推力和反扭矩相互独立，不考虑空气动力学效应。
- (4) 舵机的动态特性与旋翼转速无关；
- (5) 机体坐标系  $\mathcal{F}_b$  的原点  $O_b$  与飞行器质心重合，且初始时刻机体坐标系轴的指向与世界（惯性）坐标系  $\mathcal{F}_w$  轴的指向一致；

### 2.3.1 动力学建模与微分平坦性分析

#### 2.3.1.1 机体坐标系及基本量规定

在传统欠驱动四旋翼飞行器家族中已经出现了许多种类的结构和形状。从旋翼数量上来看，除了最常见的四旋翼飞行器外，还出现了三旋翼、六旋翼、八旋翼等飞行器，以及共轴双桨类飞行器；从机头相对旋翼的位置来看，又可分为 X 形和十字形等。大多数多旋翼飞行器的旋翼和结构的布局追求对称性，且为抵消静止悬停时空气施加的反扭矩，通常相邻和相对的两个旋翼旋转转速相等、方向相反。图2-7展示了几种常见构型的多旋翼飞行器及其在 PX4 开源飞控<sup>①</sup>中约定的旋翼编号和旋转方向。

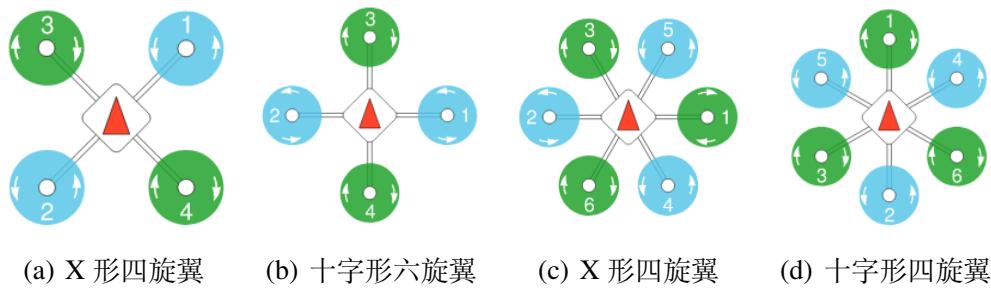


图 2-7 PX4 中几种不同构型的俯视示意图

本课题中 OmniHex 沿用如图2-7 (c)所示的 X 形六旋翼飞行器的旋翼编号和旋转方向。除此之外，设置机体坐标系  $\mathcal{F}_b$  为“北-东-地”（NED）坐标系：机头方向为  $x$  轴，垂直向机腹为  $z$  轴， $y$  轴指向由右手法则确定；机臂倾转时的转轴设定为机臂轴线向外，倾转角  $\alpha_i$  的正方向由右手法则确定。图2-8对上述规定做出了标注。

<sup>①</sup> <https://github.com/PX4/PX4-Autopilot>

### 2.3.1.2 动力学建模

基于前文所述的假设与规定，可以列出机体坐标系下 OmniHex 的刚体动力学 Newton-Euler 方程：

$$\mathbf{f}_b = m(\dot{\mathbf{v}}_b + \boldsymbol{\omega}_b \times \mathbf{v}_b - \mathbf{g}_b) \quad (2-1)$$

$$\boldsymbol{\tau}_b = \boldsymbol{\omega}_b \times \mathbf{J}_b \boldsymbol{\omega}_b + \mathbf{J}_b \dot{\boldsymbol{\omega}}_b \quad (2-2)$$

式中  $m$ ——飞行器的总质量；

$\mathbf{f}_b$ ——旋翼合推力在机体坐标系下的表示；

$\boldsymbol{\tau}_b$ ——旋翼合转矩在机体坐标系下的表示；

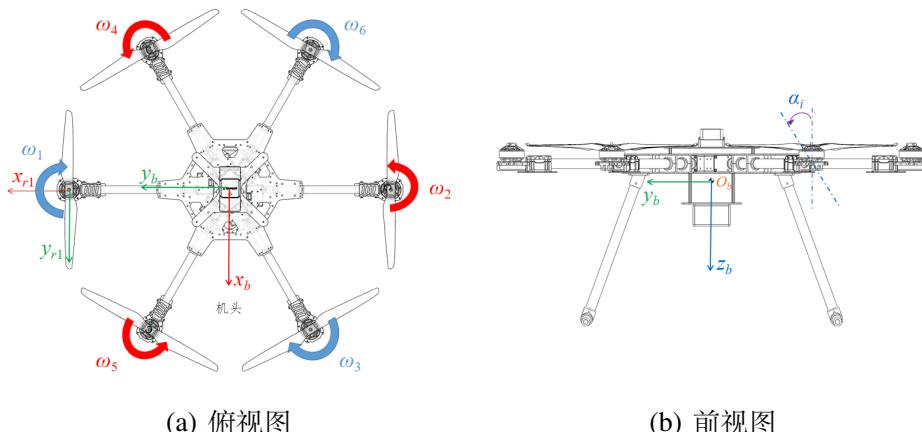
$\mathbf{v}_b$ ——飞行器质心速度在机体坐标系下的表示；

$\boldsymbol{\omega}_b$ ——飞行器相对世界坐标系的角速度在机体坐标系下的表示；

$\mathbf{g}_b$ ——重力加速度在机体坐标系下的表示；

$\mathbf{J}_b$ ——飞行器相对机体坐标系的惯量矩阵；

式 (2-1) 及式 (2-2) 分别完整描述了 OmniHex 的平移和旋转动力学特性（实际上可



(a) 俯视图

(b) 前视图

图 2-8 OmniHex 坐标系及相关量规定示意图

以描述任何刚体），为后续分析和控制策略的设计提供了依据。

根据先前的假设，每个旋翼所产生的推力和反扭矩与转速的平方成正比，记旋翼的推力系数为  $k_F$ 、反扭矩系数为  $k_M$ ，则第  $i$  个旋翼所产生的推力和反扭矩的大小为：

$$f_i = k_F \omega_i^2 \quad (2-3)$$

$$\tau_i = k_M \omega_i^2 \quad (2-4)$$

记平方转速向量和机臂倾转角向量为：

$$\boldsymbol{\Omega} = \left[ \Omega_1 \cdots \Omega_6 \right]^T = \left[ \omega_1^2 \cdots \omega_6^2 \right]^T \in \mathbb{R}^6 \quad (2-5)$$

$$\boldsymbol{\alpha} = \left[ \alpha_1 \cdots \alpha_6 \right]^T \in \mathbb{R}^6 \quad (2-6)$$

取 OmniHex 的控制输入为机体坐标系下的旋翼合推力与和转矩，则平方转速向量与控制输入之间具有如下的映射关系：

$$\mathbf{u} = \left[ \mathbf{f}_b^T \boldsymbol{\tau}_b^T \right]^T = \mathbf{A}_{\alpha} \boldsymbol{\Omega} \quad (2-7)$$

其中  $\mathbf{A}_{\alpha} \in \mathbb{R}^{6 \times 6}$  为分配矩阵，其元素均为  $\sin \alpha_i$  和  $\cos \alpha_i$  的线性组合，由于不考虑倾转舵机的响应时间，又把  $\mathbf{A}_{\alpha}$  称为瞬时分配矩阵<sup>[23]</sup>。

给定期望的控制输入  $\mathbf{u}$ ，要得到对应的执行器指令  $\boldsymbol{\alpha}$  和  $\boldsymbol{\Omega}$ ，需要求解非线性方程组式 (2-7)。一种方法是先确定一组机臂倾转角  $\boldsymbol{\alpha}$ ，这样瞬时分配矩阵  $\mathbf{A}_{\alpha}$  就成为一个已知的非奇异矩阵，进而可解得唯一的转速分配  $\boldsymbol{\Omega}$ ；不过这种做法存在一个较为严重的问题：一旦  $\boldsymbol{\alpha}$  确定，那么  $\boldsymbol{\Omega}$  也随之确定，难以选择合适的  $\boldsymbol{\alpha}$  来保证  $\Omega_i$  的非负约束。为避免这个问题，可以对方程组式 (2-7) 进行变形。首先构造以下中间变量：

$$\zeta_{ci} = \Omega_i \cos \alpha_i, \zeta_{si} = \Omega_i \sin \alpha_i \quad (2-8)$$

$$\boldsymbol{\zeta} = \left[ \zeta_{c1} \ \zeta_{s1} \cdots \zeta_{c6} \ \zeta_{s6} \right]^T \in \mathbb{R}^{12} \quad (2-9)$$

则方程组式 (2-7) 可以变形为如下形式：

$$\mathbf{u} = \mathbf{A} \boldsymbol{\zeta} \quad (2-10)$$

其中  $\mathbf{A} \in \mathbb{R}^{6 \times 12}$  是常量矩阵，仅与  $k_F$ 、 $k_M$  和  $l$  有关，可以通过其 Moore-Penrose 伪逆求得中间变量  $\boldsymbol{\zeta}$ ：

$$\boldsymbol{\zeta} = \mathbf{A}^\dagger \mathbf{u} \quad (2-11)$$

然后相应的  $\boldsymbol{\alpha}$  和  $\boldsymbol{\Omega}$  由下式确定：

$$\alpha_i = \arctan 2(\zeta_{si}, \zeta_{ci}) \quad (2-12)$$

$$\Omega_i = \sqrt{\zeta_{si}^2 + \zeta_{ci}^2} \quad (2-13)$$

### 2.3.1.3 微分平坦性分析

现代控制理论常常使用如下形式的状态空间方程来对一个非线性动态系统进行描述：

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^m \quad (2-14)$$

$$\mathbf{z} = \mathbf{g}(\mathbf{x}), \mathbf{z} \in \mathbb{R}^m \quad (2-15)$$

在无人机系统中常把状态变量取为位姿与速度的组合：

$$\begin{aligned} \mathbf{x} &= [p_x, p_y, p_z, \phi, \theta, \psi, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^T \\ &= [\mathbf{p}^T \ \boldsymbol{\varepsilon}^T \ \mathbf{v}^T \ \boldsymbol{\omega}^T]^T \in \mathbb{R}^{12} \end{aligned} \quad (2-16)$$

即  $n = 12$ 。对于欠驱动多旋翼无人机而言，有  $m = 4$ ；对于 OmniHex 这类全驱动多旋翼无人机而言，则有  $m = 6$ 。

微分平坦的概念最早由 Fliess 等人于 1995 年提出<sup>[24]</sup>，随后便在机器人轨迹规划领域收到了很大的关注。利用微分平坦性质可以很方便地从系统的输出  $\mathbf{z}$  及其有限阶导数还原出系统的状态变量  $\mathbf{x}$  和控制输入  $\mathbf{u}$ ，反过来也可以将对系统状态和输入的约束  $\mathcal{G}_D(\mathbf{x}, \mathbf{u})$  转化为对平坦输出及其有限阶导数的约束  $\mathcal{G}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(s)})$ （图 2-9）。在平坦输出空间中进行轨迹规划，一来可以对轨迹进行有效降维，使问题得到简化，加快求解速度；二来能更容易地使轨迹满足机器人的各种约束，更有利

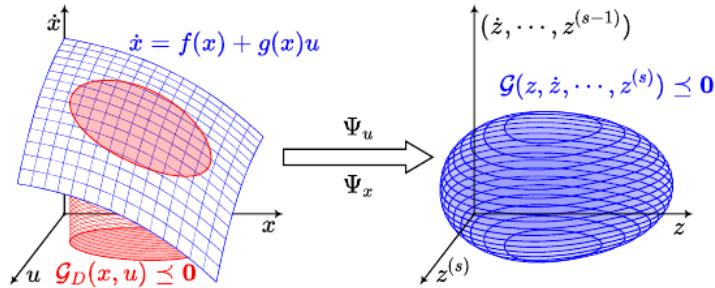


图 2-9 利用微分平坦性质进行约束转化

题得到简化，加快求解速度；二来能更容易地使轨迹满足机器人的各种约束，更有利

Mellinger 等人就已经推导出了欠驱动四旋翼飞行器的微分平坦关系<sup>[3]</sup>，而全驱动飞行器的微分平坦性质还没有较为成熟的研究。

本课题也将在平坦输出空间中为过驱动飞行器 OmniHex 进行轨迹规划，下面将对 OmniHex 的微分平坦性质进行简要分析。首先给出微分平坦的定义：

**定义 2.1（微分平坦）** 考虑式 (2-14) 所描述的非线性系统，若能够找到一组系统输出  $\mathbf{z} \in \mathbb{R}^m$ ，使得系统状态  $\mathbf{x}$  和控制输入  $\mathbf{u}$  都能用  $\mathbf{z}$  及其有限阶导数表示，即：

$$\mathbf{x} = \Psi_{\mathbf{x}}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(s-1)}) \quad (2-17)$$

$$\mathbf{u} = \Psi_{\mathbf{u}}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(s)}) \quad (2-18)$$

则称该系统为微分平坦系统，相应的输出  $\mathbf{z}$  称为平坦输出。

对于 OmniHex 这样的全/过驱动飞行器而言，其平坦输出应为一个 6 维向量。不妨取系统输出为：

$$z = \begin{bmatrix} p^T & \sigma^T \end{bmatrix}^T \in \mathbb{R}^6 \quad (2-19)$$

其中  $\sigma \in \mathbb{R}^3$  是机体姿态参数化为 3 维向量的结果（如欧拉角等），其对应的姿态由一个从  $\mathbb{R}^3$  到  $SO(3)$  空间的二阶及以上连续可微满射确定，若一个姿态对应有多个  $\sigma$  值，则姿态参数化取此满射的一个局部逆映射。考虑旋转矩阵微分与角速度的关系，有

$$\omega^\wedge = \dot{\mathbf{R}}(\sigma) \mathbf{R}^T(\sigma) = \sum_{i=1}^3 \frac{\partial \mathbf{R}(\sigma)}{\partial \sigma_i} \dot{\sigma}_i \mathbf{R}^T(\sigma) \quad (2-20)$$

其中  $\sigma_i \in \mathbb{R}^3$  表示  $\sigma$  的第  $i$  个元素；映射  $(\cdot)^\wedge : \mathbb{R}^3 \mapsto so(3)$  表示取向量 · 对应的反对称矩阵（skew-symmetric matrix），即有关系  $\mathbf{a} \times \mathbf{b} = \mathbf{a}^\wedge \mathbf{b}$ ，同理可以定义其逆映射  $(\cdot)^\vee : so(3) \mapsto \mathbb{R}^3$ ，表示取反对称矩阵对应的向量。这样机体的角速度  $\omega$  也可以由  $\sigma$  及其导数表示：

$$\omega(\sigma, \dot{\sigma}) = \left[ \sum_{i=1}^3 \frac{\partial \mathbf{R}(\sigma)}{\partial \sigma_i} \dot{\sigma}_i \mathbf{R}^T(\sigma) \right]^\vee \quad (2-21)$$

进一步可以表示角加速度：

$$\begin{aligned} (\dot{\omega})^\wedge &= \frac{d\omega^\wedge}{dt} = \frac{d}{dt} \left( \sum_{i=1}^3 \frac{\partial \mathbf{R}(\sigma)}{\partial \sigma_i} \dot{\sigma}_i \mathbf{R}^T(\sigma) \right) \\ &= \sum_{i=1}^3 \left( \sum_{j=1}^3 \frac{\partial^2 \mathbf{R}}{\partial \sigma_i \partial \sigma_j} \ddot{\sigma}_j \dot{\sigma}_i + \frac{\partial \mathbf{R}}{\partial \sigma_i} \ddot{\sigma}_i \right) \mathbf{R}^T + \dot{\mathbf{R}} \dot{\mathbf{R}}^T \end{aligned} \quad (2-22)$$

于是我们得到了状态变量  $x$  关于输出  $z$  及其各阶导数的函数关系：

$$x = \begin{bmatrix} p \\ \varepsilon \\ v \\ \omega \end{bmatrix} = \Psi_x(z, \dot{z}) = \begin{bmatrix} p \\ \varepsilon(\sigma) \\ \dot{p} \\ \left( \sum_{i=1}^3 \frac{\partial \mathbf{R}(\sigma)}{\partial \sigma_i} \dot{\sigma}_i \mathbf{R}^T(\sigma) \right)^\vee \end{bmatrix} \quad (2-23)$$

控制输入  $u$  关于输出  $z$  及其各阶导数的函数关系则可根据式 (2-1) 和式 (2-2)，结合式 (2-21) 及式 (2-22) 确定：

$$u = \begin{bmatrix} f_b \\ \tau_b \end{bmatrix} = \Psi_u(z, \dot{z}, \ddot{z}) = \begin{bmatrix} m \mathbf{R}(\sigma) (\ddot{p} - g) \\ \mathbf{R}(\sigma) (\omega \times \mathbf{J}(\sigma) \omega + \mathbf{J}(\sigma) \dot{\omega}) \end{bmatrix} \quad (2-24)$$

$$\mathbf{J}(\sigma) = \mathbf{R}(\sigma) \mathbf{J}_b \mathbf{R}^T(\sigma) \quad (2-25)$$

至此我们说明了全驱动多旋翼飞行器是微分平坦系统，式(2-19)所选的系统输出 $z$ 为平坦输出。

## 2.4 本章小结

本章简要介绍了 OmniHex 实物系统的设计与搭建，并基于实际系统建立并分析了 OmniHex 的动力学模型，阐明了其是微分平坦系统并给出了平坦输出的形式，为后续选取平坦输出进行 6 自由度  $SE(3)$  轨迹规划提供了依据。

## 第3章 前端路径搜索算法及安全约束生成算法设计

### 3.1 引言

基于优化的多旋翼无人机的轨迹规划过程一般可分为前端和后端两个部分：

(1) 前端路径查找 (front-end path search): 利用地图信息在空间中搜寻出一条可行的无碰撞路径，路径由一系列路径点组成，是对飞行器运动的纯几何描述。因此路径查找工作在离散、低维的空间中。根据需要，前端算法还可以根据搜索出的路径得到四周安全无碰撞区域的几何描述（如前文提到的安全飞行走廊），作为后端优化的约束信息。

(2) 后端轨迹生成 (back-end trajectory generation): 在前端找到的可行路径和安全约束的基础上，根据动力学和运动学约束，通过拟合、优化的方法得到一条具备时间律的连续、平滑且安全的轨迹输入到飞行器的控制器中。因此轨迹生成工作在连续、高维的空间中。

本章将为过驱动飞行器的轨迹规划器设计前端算法。3.2节介绍  $SE(3)$  空间中 RRT 算法的设计与实现；3.3节介绍安全飞行走廊生成算法的设计与实现。

### 3.2 路径搜索算法的设计

为实现 OmniHex 等过驱动多旋翼飞行器的路径搜索，有很多算法可供选择。本节从算法选择、算法原理和算法实现三方面讲述路径搜索的算法设计。

#### 3.2.1 路径搜索算法概述

目前常用的路径搜索算法种类非常多，大体上可以分为基于图搜索的和基于采样的两类。

在基于图搜索的算法中，比较有代表性的有 Dijkstra 算法、基于 Dijkstra 算法改进的 A\* 算法<sup>[25]</sup> 以及基于 A\* 算法改进的跳点搜索 (jump point search, JPS)<sup>[26]</sup>，这些算法同时具有完备性和最优性，即当可行解存在时就一定能找到可行解，并且这个可行解一定是所有可行解里最优的，然而一个比较大的缺点是，这些算法需要按一定的分辨率将搜索空间离散化并存储每一个网格，这就导致其内存占用量会随着搜索空间的维度增加呈指数上升，与此同时计算时间也会快速增加，因此这类算法不适用于  $SE(3)$  空间等高维状态空间中的路径搜索。

与基于图搜索的算法有序地搜索离散化的构型空间不同，基于采样的路径搜

索算法在连续的构型空间中进行随机采样以期找到可行解。如概率路图 (probabilistic roadmap, PRM) 算法<sup>[27]</sup> 使用随机采样得到的构型空间中的可行点构建无碰撞图，并在图中使用图搜索得到图中从起点到终点最短路径；快速探索随机树 (rapidly-exploring random tree, RRT)<sup>[28]</sup> 算法基于单次查询的信息、以起点为根节点增量式地构建树结构，树触及终点后即可回溯出一条从起点到终点的可行路径。PRM 算法和 RRT 算法都是概率完备的<sup>[29,30]</sup>，即如果可行解存在，则它们搜索失败的概率会随着样本数趋向无穷大而衰减到 0。然而，PRM 算法和 RRT 算法都不能保证解的最优性，作为改进，2011 年 Karaman 等人提出了这两种算法的渐进最优版本：PRM\* 和 RRT\*<sup>[31]</sup>，这两种算法可以保证可行解存在时能被找到，且随着样本数趋于无穷而收敛至全局最优。在此基础上又发展出了加快 RRT\* 收敛速率的 Informed-RRT\*<sup>[32]</sup>、适用于高维空间的商空间 RRT 算法 QRRT<sup>[33]</sup> 等。基于采样的算法在内存消耗和计算时间方面相比基于搜索的算法更有优势。

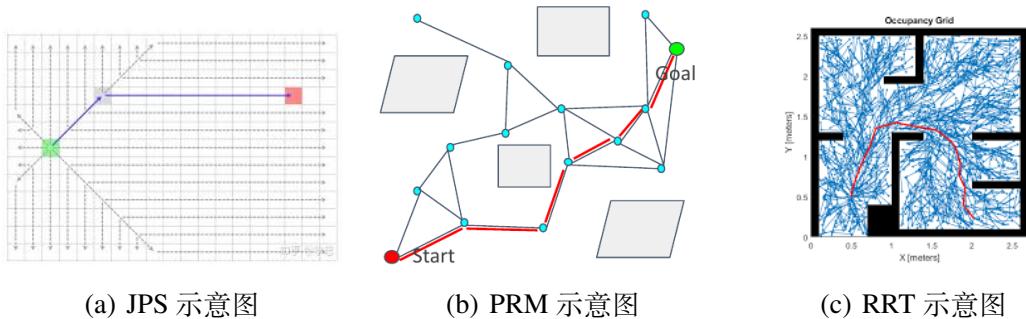


图 3-1 三种典型的路径搜索算法示意图

本课题拟在  $SE(3)$  状态空间中为过驱动多旋翼飞行器搜索出一条可行路径，构型空间维度较高（6 维）且重点关注路径的安全性，并不优先考虑最优性，故选择 RRT 算法作为前端路径搜索算法。

### 3.2.2 $SE(3)$ 空间中的 RRT 算法

#### 3.2.2.1 RRT 算法原理

如3.2.1节所述，RRT 是一种通过随机构建空间填充树来高效搜索非凸高维空间的算法。这棵树是以起始状态为根节点，根据从搜索空间中随机抽取的样本增量式构建的。每次迭代都从搜索空间中采样出一个样本点，随后尝试从已有的树中距离样本点最近的节点向样本点以一定步长作延伸，到达一个新状态并形成一个运动基元，如果该运动基元是有效的（比如全部位于无碰撞空间中，或满足某些其它的约束），则把这个运动基元和这个新状态分别作为边和节点加入树中，这

样整棵树就完成了一次生长。当新状态与目标状态的距离小于一定阈值且与目标状态的连接有效，则完成搜索。

本课题的  $SE(3)$  空间中的 RRT 算法完全遵循这些步骤，具体算法流程见算法3-1。

---

#### 算法 3-1 $SE(3)$ 空间中的 RRT 算法

---

**Input** 地图  $\mathcal{M}$ 、起始状态  $\mathbf{x}_{init} \in SE(3)$ 、目标状态  $\mathbf{x}_{goal} \in SE(3)$ 、最大迭代次数  $n$ 、采样概率  $p$  和步长  $\delta$

**Output** 从  $\mathbf{x}_{init}$  到  $\mathbf{x}_{goal}$  的一条可行路径  $\Gamma$

- 1: 初始化搜索树  $\mathcal{T}$ ，其中只有根节点  $\mathbf{x}_{init}$
- 2: **for**  $i = 1$  to  $n$  **do**
- 3: 获得  $\mathbf{x}_{sample}$ :  $\mathbf{x}_{sample}$  有  $p$  的概率通过在搜索空间中均匀随机采样得到，有  $(1 - p)$  的概率直接取  $\mathbf{x}_{goal}$
- 4: 在搜索树  $\mathcal{T}$  中找到与  $\mathbf{x}_{sample}$  距离最近的节点  $\mathbf{x}_{near}$
- 5: 从  $\mathbf{x}_{near}$  到  $\mathbf{x}_{sample}$  以步长  $\delta$  作插值，得到新状态  $\mathbf{x}_{new}$
- 6: **if** 运动基元  $(\mathbf{x}_{near} \rightarrow \mathbf{x}_{new})$  无碰撞 **then**
- 7: 将  $\mathbf{x}_{new}$  作为新节点加入搜索树  $\mathcal{T}$ ，并设置其父节点为  $\mathbf{x}_{near}$
- 8: **end if**
- 9: **if**  $\mathbf{x}_{new}$  与  $\mathbf{x}_{goal}$  之间的距离小于等于  $\delta$  且运动基元  $(\mathbf{x}_{near} \rightarrow \mathbf{x}_{new})$  无碰撞 **then**
- 10: 将  $\mathbf{x}_{goal}$  加入搜索树  $\mathcal{T}$ ，并设置其父节点为  $\mathbf{x}_{new}$
- 11: 从  $\mathbf{x}_{goal}$  根据父节点回溯出路径  $\Gamma$
- 12: 结束循环
- 13: **end if**
- 14: **end for**

---

### 3.2.2.2 $SE(3)$ 状态空间中的相关操作

从算法3-1中可以看到，在搜索过程中需要对状态空间进行均匀采样，需要计算两个状态之间的距离，在得到新状态以及对运动基元的有效性进行检测时还需要在两个状态之间进行插值。即需要对  $SE(3)$  空间作采样、度量和插值三种操作。 $SE(3)$  状态空间是一种复合的状态空间，由位置子空间  $\mathbb{R}^3$  和姿态子空间  $SO(3)$  构成，即  $SE(3) = \mathbb{R}^3 \times SO(3)$  在进行上述三种操作时可以先分别对两个子空间独立进行考虑，最后再进行整合。

位置子空间是欧几里得空间，在其中进行均匀采样就是分别对三个坐标值进行均匀采样；两个位置之间的距离就是欧氏距离：

$$\text{Dist}_{\mathbb{R}^3}(\mathbf{p}_1, \mathbf{p}_2) = \|\mathbf{p}_1 - \mathbf{p}_2\|_2, \forall \mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^3 \quad (3-1)$$

两个位置向量之间的线性插值如下式所示：

$$\text{Interp}_{\mathbb{R}^3}(\mathbf{p}_1, \mathbf{p}_2; t) = (1 - t)\mathbf{p}_1 + t\mathbf{p}_2, \forall \mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^3, \forall t \in [0, 1] \quad (3-2)$$

位置子空间中上述三种操作都很直接。然而，在非欧的姿态子空间  $SO(3)$  中

进行这些操作就显得不那么直观了。一种比较朴素的做法是把  $SO(3)$  中的姿态用欧拉角表示，然后按  $\mathbb{R}^3$  空间中的方法来进行操作。但是由于欧拉角的奇异性与周期性，使得欧氏空间中的度量方法并不能很好地反映姿态之间的差异。比如，两组数值上差别很大的欧拉角可能表示的是两个相近甚至相同的姿态。所以这种做法缺乏合理性。

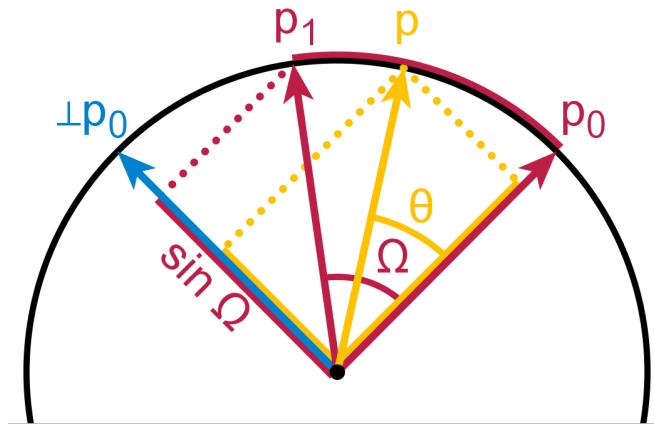


图 3-2 球面线性插值 (slerp) 示意图

本课题设计的 RRT 算法中对  $SO(3)$  空间的处理方式参考 Kuffner 等人于 2004 年提出的方法<sup>[34]</sup>，以上三种操作均基于四元数来进行，这样可以有效避免姿态表示的奇异性与重复性带来的缺点。记表示绕轴  $\nu = [v_1 \ v_2 \ v_3]^T \in \mathbb{R}^3$  旋转角度  $\theta \in \mathbb{R}$  的单位四元数为：

$$\mathbf{Q} = [w \ x \ y \ z]^T = \left[ \cos \frac{\theta}{2} \ v_1 \cos \frac{\theta}{2} \ v_2 \cos \frac{\theta}{2} \ v_3 \cos \frac{\theta}{2} \right]^T \quad (3-3)$$

则将在  $SO(3)$  空间中进行均匀采样可视为在单位四元数超球面上进行均匀采样，其流程如算法 3-2 所示；

---

算法 3-2 生成均匀分布的随机单位四元数

---

**Input** 无

**Output** 均匀随机四元数  $\mathbf{Q} = (w, x, y, z)$

- 1:  $s = uniform(0, 1) // uniform(0,1)$  为生成 0 到 1 之间的均匀分布随机数
  - 2:  $\sigma_1 = \sqrt{1 - s}$
  - 3:  $\sigma_2 = \sqrt{s}$
  - 4:  $\theta_1 = 2\pi * uniform(0, 1)$
  - 5:  $\theta_2 = 2\pi * uniform(0, 1)$
  - 6:  $w = \cos(\theta_2) * \sigma_2$
  - 7:  $x = \sin(\theta_1) * \sigma_1$
  - 8:  $y = \cos(\theta_1) * \sigma_1$
  - 9:  $z = \sin(\theta_2) * \sigma_2$
  - 10: **return**  $(w, x, y, z)$
-

两个  $SO(3)$  状态间的距离定义为其在超球面上的弧长，即

$$\text{Dist}_{SO(3)}(\mathbf{Q}_1, \mathbf{Q}_2) = \arccos(\mathbf{Q}_1^T \mathbf{Q}_2), \forall \mathbf{Q}_1, \mathbf{Q}_2 \in SO(3) \quad (3-4)$$

在两个  $SO(3)$  状态间进行插值可视为在单位四元数超球面上进行球面线性插值 (slerp, 图3-2)，即：

$$\text{Interp}_{SO(3)}(\mathbf{Q}_1, \mathbf{Q}_2; t) = \frac{\sin[(1-t)\Omega]}{\sin \Omega} \mathbf{Q}_1 + \frac{\sin[t\Omega]}{\sin \Omega} \mathbf{Q}_2, \forall \mathbf{Q}_1, \mathbf{Q}_2 \in SO(3), \forall t \in [0, 1] \quad (3-5)$$

现在已经分别确定了  $SE(3)$  中位置子空间和姿态子空间的采样、度量和插值的方法，那么整个  $SE(3)$  空间中的采样和插值操作可以分别对其子空间进行，最后将得到的结果复合即可；两个  $SE(3)$  状态的距离则可以表示如下：

$$\text{Dist}_{SE(3)}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\text{Dist}_{\mathbb{R}^3}^2(\mathbf{p}_1, \mathbf{p}_2) + \text{Dist}_{SO(3)}^2(\mathbf{Q}_1, \mathbf{Q}_2)}, \forall \mathbf{x}_1, \mathbf{x}_2 \in SE(3) \quad (3-6)$$

其中  $\mathbf{p}_i \in \mathbb{R}^3$  和  $\mathbf{Q}_i \in SO(3)$  分别为  $\mathbf{x}_i \in SE(3)$  的位置和姿态分量 ( $i = 1, 2$ )。

### 3.2.3 近邻搜索

RRT 算法的另一个关键步骤是在已有的树中寻找出距离采样点  $\mathbf{x}_{sample}$  最近的节点，这是一个近邻搜索问题，其耗时与待搜索集合中的元素个数  $n$  呈正相关，也就是说，随着 RRT 算法迭代次数的增加，查找最近节点这一步的耗时也会随之增加，所以这一步的效率是影响整个寻路算法效率的最重要因素之一。

近邻搜索最简单的方法就是线性搜索，其具有  $O(n)$  的时间复杂度。显然，当  $n$  很大时线性搜索并不是一种高效的方法。目前有许多用来存储带查找元素的数据结构可以加速搜索，如 Kd 树<sup>[35]</sup> 就是一种用来组织欧氏空间  $\mathbb{R}^k$  中点的数据结构，在一棵平衡的 Kd 树中作最近邻搜索的平均时间复杂度为  $O(\log n)$ ；1995 年 Brin 提出的几何近邻搜索树 (geometric near-neighbor access tree, GNAT)<sup>[36]</sup> 只需要定义好距离函数就可以存储任意种类的数据点，并且有相关实验能说明 GNAT 树在近邻搜索问题上的高效性。

根据本课题的需求，这里选择 GNAT 作为存储非欧空间  $SE(3)$  中点的数据结构。下面对 GNAT 的构建和基于 GNAT 的近邻搜索算法作简要介绍

#### 3.2.3.1 GNAT 数据结构简介

GNAT 是一种基于空间的分层超平面划分 (hierarchical hyperplane partitioning) 对度量空间 (metric space) 进行索引的数据结构。在介绍 GNAT 的核心思想之前，先给出 Dirichlet 域的概念：

**定义 3.1 (Dirichlet 域)** 给定度量空间  $M$ , 以及点集  $P = \{x_1, \dots, x_k\} \subset M$ , 点  $x_i \in P$  的 Dirichlet 域  $\mathbb{D}_{x_i} \subset M$  是  $M$  中所有满足下述条件的点  $x$  的集合:

$$\text{Dist}(x, x_i) \leq \text{Dist}(x_j, x_i), \forall j \in \{1, \dots, k\} \setminus \{i\} \quad (3-7)$$

在 GNAT 树的根节点处, 一系列分割点  $X$  被一种贪心策略从待存储的点集  $S$  中选出, 接着基于这些分割点将度量空间  $M$  划分为一系列 Dirichlet 域  $\mathbb{D}_{x_1}, \dots, \mathbb{D}_{x_k}$ , 剩下的点将根据它们落入的 Dirichlet 域分组, 随后继续对这些分组重复上述步骤, 最后递归地构建出 GNAT。图3-3展示了棵简单的 2 层 GNAT 结构, 图中较大的点代表顶层节点(根节点)的分割点, 较小的点代表子节点(叶节点)的分割点; 较粗的线代表顶层分割点对应区域的边界, 而较细的线则代表底层分割点的。

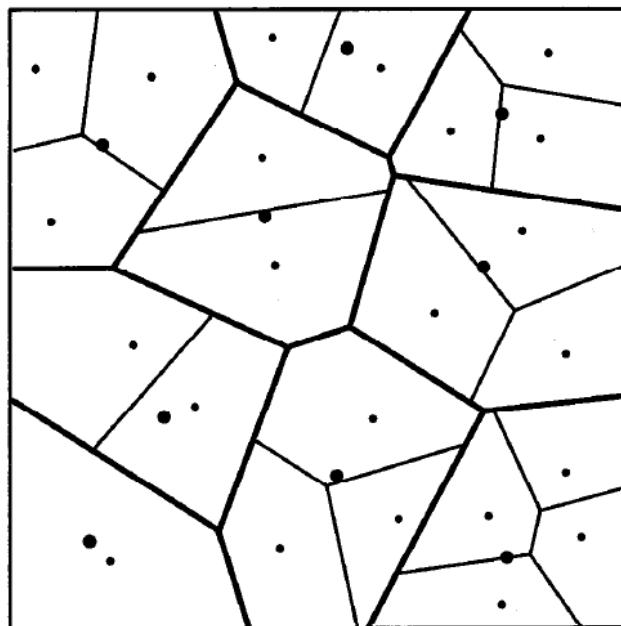


图 3-3 一棵 2 层 GNAT 示意图

GNAT 树的基本构建过程叙述如下:

- (1) 从待组织的数据集中用贪心策略选出相隔足够远的  $k$  个分割点  $x_1, \dots, x_k$ ;
- (2) 将数据集中剩下的点与最近的分割点相关联, 并将与分割点  $x_i$  相关联的点的集合记为  $D_{x_i}$ ;
- (3) 对每对分割点  $(x_i, x_j)$ , 计算如下距离范围:

$$\text{range}(x_i, D_{x_j}) = [\min \text{Dist}_M(x_i, D_{x_j}), \max \text{Dist}_M(x_i, D_{x_j})] \quad (3-8)$$

, 用于搜索时的剪枝操作;

(4) 递归地对每个  $D_{x_i}$  重复上述过程。

### 3.2.3.2 基于 GNAT 的近邻搜索

给定一个点  $x$ , 若要在 GNAT 中搜索出所有与  $x$  的距离在  $r$  之内的所有点 ( $r$ -近邻点), 除了充分利用 GNAT 所表达的数据集的固有几何信息外, 还可以利用距离信息进行剪枝以提高效率。如图3-4所示, 因为  $\text{Dist}(x, p) + r < \min_d(p, D_{p_i})$ , 所以  $D_{p_i}$  中就一定不存在  $x$  的  $r$ -近邻点, 这样  $D_{p_i}$  对应的子节点就可以被剪除。

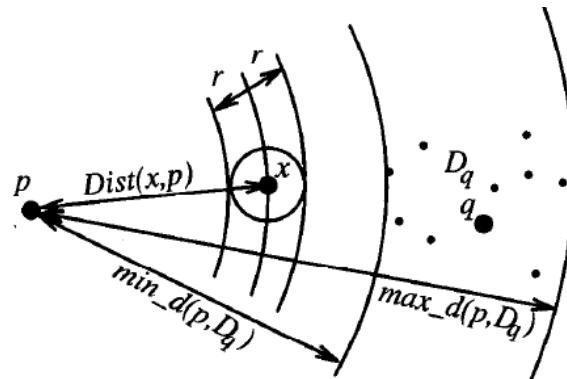


图 3-4 利用距离信息在  $r$ -近邻搜索中进行剪枝的原理图

在 GNAT 中进行  $r$ -近邻搜索的步骤叙述如下:

- (1) 记  $P$  为当前可能包含  $x$  的  $r$ -近邻点的节点 (初始状态下为根节点) 的分割点, 初始时  $P$  包含当前节点所有的分割点;
- (2) 不重复地取一点  $y \in P$ , 若  $\text{Dist}_M(x, y) \leq r$ , 则将  $y$  加入结果中;
- (3) 考察所有  $x_i \in P$ , 如果有  $[\text{Dist}_M(x, y_i) - r, \text{Dist}_M(x, y_i) + r] \cap \text{range}(y, D_{x_i}) = \emptyset$ , 则将  $x_i$  从  $P$  中移除;
- (4) 重复步骤2和3, 直到  $P$  中所有点都试过一遍;
- (5) 对  $P$  中剩下的所有点  $y_i$ , 递归地对  $D_{y_i}$  进行搜索。

如果要得到数据集中距离给定的  $x$  最近的点, 可以先基于 Dirichlet 域找到  $x$  在 GNAT 中所在的叶节点, 随后在叶节点中线性搜索出其中距离  $x$  最近的点, 记这个最小距离为  $r'$ ; 接着搜索出  $x$  的所有  $r'$ -近邻点, 最后再对这些  $r'$ -近邻点进行线性搜索得到精确  $x$  的最近邻。

为说明 GNAT 在近邻搜索上的高效性, 本课题随机生成了  $10^5$  个 Eigen 三维双精度浮点型向量, 分别基于线性搜索和 GNAT 做最近邻搜索。经过数次集重复试验, 每次都生成不同的随机点集, 发现二者耗时均很稳定, 线性搜索平均耗时为 34.6 毫秒, 而 GNAT 平均耗时仅为 0.105 毫秒, 二者相差数百倍, 根据这个结果可以认为基于 GNAT 的近邻搜索之于线性搜索有巨大的优势。

以上介绍了基于 GNAT 的近邻搜索的通用方法，在本课题的应用场景中，对应的度量空间  $M$  就是  $SE(3)$  空间，而 GNAT 所存储的就是当前的搜索树的所有节点。

### 3.2.4 有效性检测

在 RRT 算法中涉及两种有效性检测，一种是判断搜索空间中的单个状态是否有效，另一种是基于单个状态的有效性检测来判断一个运动基元是否有效。

在本课题所设计的  $SE(3)$  空间的 RRT 算法（算法3-1）中，对状态和运动有效性的判断标准是其是否让飞行器与障碍物发生碰撞。因为需要同时考虑飞行器的位置和姿态，且考虑到在狭小空间内实现轨迹规划的最终目标，有必要在前端路径搜索时就考虑飞行器的形状。

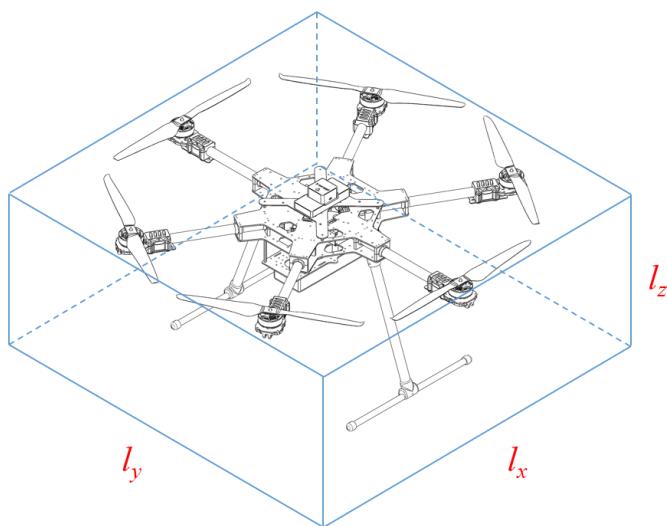


图 3-5 将飞行器的形状近似为长方体

如图3-5所示，这里将飞行器的形状简化为一个以其质心为中心、棱始终与机体坐标系保持一致的长方体。则判断单个  $SE(3)$  状态是否有效，就转化为判断代表飞行器的长方体在这个状态对应的位置和姿态下是否与障碍物无交集，而判断运动基元是否有效则可以在其上离散地取点（需要用到插值），若每个点对应的状态都是有效的，则认为该运动基元有效。因此，本小节着重考虑单个状态的有效性检测。

在本课题所编写的 RRT 算法框架中，借鉴开源运动规划库 OMPL<sup>[37]</sup> 的思想，将有效性检测这一操作抽象为 StateValidityChecker 和 MotionValidityChecker 两个抽象基类，其中有纯虚函数 `isValid()` 作为提供给 RRT 算法主体的有效性检测公共接口，如果要采用不同的有效性检测方法，只需继承 `StateValidityChecker` 或 `MotionValidityChecker` 抽象基类，实现其 `isValid()` 接口即可，这样就实现了 RRT 算法主体与有效性检测的分离，提高了算法的复用性。

本课题根据地图形式的不同实现了基于点云的状态有效性检测和基于八叉树的状态有效性检测两种方法，下面分别作简要说明。

### 3.2.4.1 基于点云的状态有效性检测

基于障碍物点云地图进行状态有效性检测的思路比较简单，只要检查点云中是否有障碍物点落在给定  $SE(3)$  状态下的长方体内即可，有则无效，无则有效。但如果单纯地使用线性搜索就必须要遍历整个点云，这对于稠密的点云来说非常的低效。

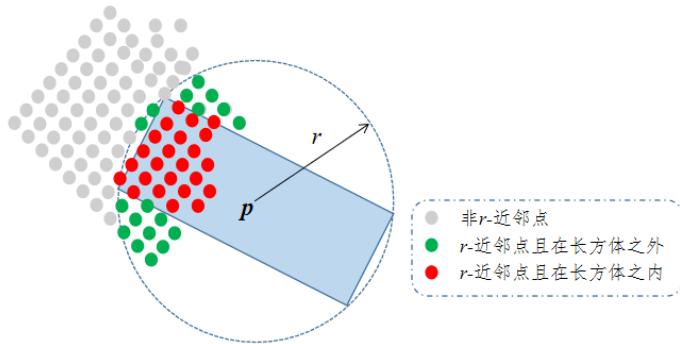


图 3-6 使用近邻搜索对有效性检测进行优化

为解决这个问题，考虑使用 GNAT 数据结构来存储点云，配合近邻搜索来加速有效性检测过程。如图3-6所示，长方体中心的位置为  $p \in \mathbb{R}^3$ ，对  $p$  进行近邻搜索得到其在点云中所有的  $r$ -近邻点，其中  $r = \sqrt{l_x^2 + l_y^2 + l_z^2}/2$  为长方体的外接球半径，然后遍历所有  $r$ -近邻点检查是否有点位于长方体内，具体流程见算法3-3。

下面分析这种方法对有效性检测效率的改进程度。令一次线性搜索有效性检测耗时为  $T_1$ ，经由近邻搜索改进后的一次有效性检测耗时为  $T_2$ ，其中线性遍历检测整个点云耗时  $T_{\text{linear}}$ ，使用 GNAT 获取  $r$ -近邻点耗时  $T_{\text{GNAT}}$ ，线性遍历检测所有  $r$ -近邻点耗时  $T_r$ ，则有：

$$T_1 \approx T_{\text{linear}} \quad (3-9)$$

$$T_2 \approx T_{\text{GNAT}} + T_r \quad (3-10)$$

再令  $T_{\text{GNAT}} = \lambda_1 T_{\text{linear}}$ ， $T_r = \lambda_2 T_{\text{linear}}$ ，那么通常有  $\lambda_1 \ll 1$ ，比如在3.2.3.2节给出的例子中，就有  $\lambda_1 < 1/300$ ； $\lambda_2$  近似为  $r$ -近邻点数量与点云中点的总数量的比值，通常也有  $\lambda_2 \ll 1$ ，故大多数情况下， $\lambda_1 + \lambda_2 \ll 1$ ，于是就有

$$T_2 \approx (\lambda_1 + \lambda_2)T_1 \ll T_1 \quad (3-11)$$

这说明使用 GNAT 近邻搜索使有效性检测效率相比于暴力线性搜索有相当大的改进，实际对比测试也支持这一结论。

**算法 3-3 基于障碍物点云的状态有效性检测**

**Input** 用 GNAT 组织好的障碍物点云  $\mathcal{P}$ , 待检测状态  $x \in SE(3)$ , 长方体尺寸  $(l_x, l_y, l_z)$ , 地图下界  $b_l \in \mathbb{R}^3$  和上界  $b_u \in \mathbb{R}^3$

**Output** 状态  $x$  是否有效

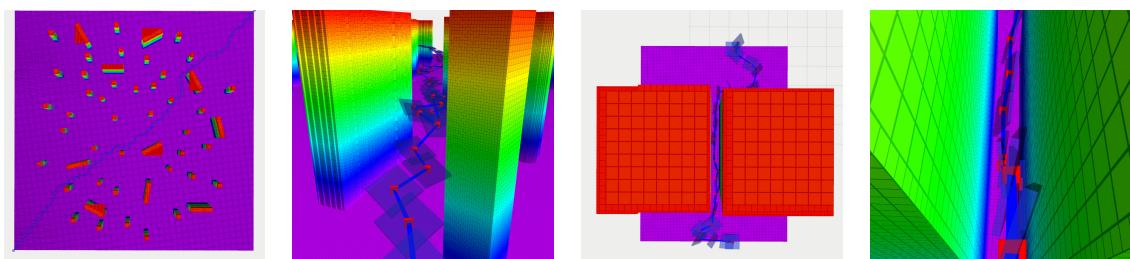
- 1: 计算长方体外接球半径  $r = \sqrt{l_x^2 + l_y^2 + l_z^2}/2$
- 2: 根据尺寸  $(l_x, l_y, l_z)$  和状态  $x$  求出长方体各顶点的坐标  $V = [v_1 \dots v_8]$
- 3: **for**  $i = 1$  to 8 **do**
- 4:     **if**  $v_i$  超出地图上下界 **then**
- 5:         **return** False
- 6:     **end if**
- 7: **end for**
- 8: 在  $\mathcal{P}$  中搜索出长方体中心点  $p$  ( $x$  的位置分量) 的  $r$ -近邻点, 构成点集  $N$
- 9: **for all**  $p_{near} \in N$  **do**
- 10:    **if**  $p_{near}$  落在了长方体内 **then**
- 11:        **return** False
- 12:    **end if**
- 13: **end for**
- 14: **return** True

**3.2.4.2 基于八叉树地图的状态有效性检测**

本课题中基于八叉树地图的状态有效性检测是通过调用 FCL (flexible collision library)<sup>[38]</sup> 实现的。FCL 库提供了与八叉树地图库 OctoMap<sup>[39]</sup> 的接口, 可以很方便地检测不同几何形状 (如代表飞行器的长方体) 在不同位姿下与障碍物的碰撞。

**3.2.5 算法效果**

本小节展示所设计的  $SE(3)$  空间中的 RRT 算法使用基于点云和基于八叉树地图两种有效性检测方法的搜索效果, 并且每种方法都分别在障碍物稀疏及穿越狭窄通道两种场景下进行测试。测试中设置步长为 0.8, 采样概率为 0.9。

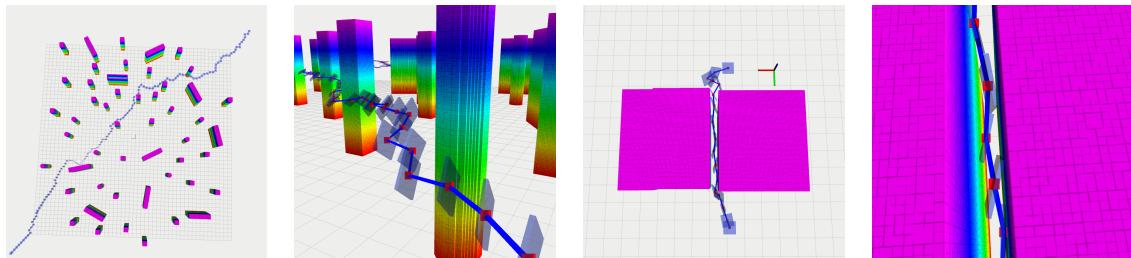


(a) 障碍稀疏 (整体) (b) 障碍稀疏 (局部) (c) 狹窄通道 (整体) (d) 狹窄通道 (局部)

图 3-7  $SE(3)$  空间 RRT 算法效果可视化结果 (基于八叉树地图)

如图3-7所示为使用基于八叉树地图的有效性检测方法时的可视化结果。图中蓝色半透明的长方体可视化表达了路径上每个  $SE(3)$  状态, 可见算法无论在宽松

的环境中还是狭窄的环境中均能成功搜索出一条无碰撞路径，说明有效性检测成功发挥了其作用，图3-7 (c)中路径起点和终点附近的弯曲也表现出了 RRT 算法无法保证解的最优性的性质，后续应用时将通过一定策略对路径点数量进行缩减来改善 RRT 的解中可能存在的这种局部的“舍近求远”现象。



(a) 障碍稀疏（整体） (b) 障碍稀疏（局部） (c) 狹窄通道（整体） (d) 狹窄通道（局部）

图 3-8 SE(3) 空间 RRT 算法效果可视化结果（基于点云地图）

如图3-8所示为使用基于点云的有效性检测方法时的可视化结果，可见得到的解与基于八叉树的方法并无很大区别。

表3-1中列出了测试中所得到的一些数据。可以看出狭窄环境的迭代次数要多于宽松的环境，这是因为狭窄环境中可行区域占比更小，在一定迭代次数内找到解的概率就会更低。而且可以注意到，同样环境和起终点条件下，使用点云方法的求解时间显著高于使用八叉树方法，这是由于这里的点云地图中障碍物内部也被填充，导致点的数量异常庞大，极大地增加了有效性检测的负担；而基于八叉树的方法检测只需要用到障碍物表面信息。如果在生成点云时能够合理地控制点的数量，尽量剔除冗余点，就能使点云法的求解速度大大提高；在后文中所用到的一个  $50\text{m} \times 50\text{m}$  大小、障碍物较为稠密的随机地图中，使用点云法可以在数百毫秒内搜索出一条对角路径。另外，利用 OctoMap 库和 PCL 库<sup>[40]</sup> 可以很方便地在点云和八叉树之间进行转换，可以根据实际需求灵活地选择地图形式。

表 3-1 关于 SE(3) 空间 RRT 算法的一些数据

环境	地图类型	地图尺寸 (m)	起点 (m)	终点 (m)	迭代次数数量级	平均耗时
宽松	八叉树	$60 \times 60 \times 6$	(29, -29, 3)	(-29, 29, 3)	$10^2$	20 毫秒左右
宽松	点云	$60 \times 60 \times 6$	(29, -29, 3)	(-29, 29, 3)	$10^2$	数秒
狭窄	八叉树	$7.5 \times 10 \times 6$	(3, 1, 3)	(3, 9, 3)	$10^3$	200 毫秒以内
狭窄	点云	$7.5 \times 10 \times 6$	(3, 1, 3)	(3, 9, 3)	$10^3$	数十秒

### 3.3 3D 安全飞行走廊生成算法的设计

3D 空间中的安全飞行走廊 (SFC) 一般用来描述无障碍物的空闲空间 (free

space)，一般用一系列球或凸多面体来表示，在无人机的轨迹规划中取得了许多应用<sup>[6,41,42]</sup>。SFC 通常在轨迹规划框架的前端部分根据初始路径生成，然后作为轨迹优化的约束给到后端。目前 SFC 的生成方法也有多种，如 Liu 等人的 RILS 算法<sup>[43]</sup>、Deits 等人的 IRIS 方法<sup>[44]</sup>等。本课题使用前者，下面对其进行简要介绍。

### 3.3.1 算法原理

记障碍物点云为  $O$ ，本算法对  $\mathbb{R}^3$  空间进行操作，只需要  $SE(3)$  路径的位置部分，记无碰撞位置路径为  $P = \langle \mathbf{p}_0 \rightarrow \dots \rightarrow \mathbf{p}_s \rangle$ ，则算法将在每条线段  $L_i = \langle \mathbf{p}_i \rightarrow \mathbf{p}_{i+1} \rangle$  周围生成一个包含无碰撞空间的凸多面体  $P_i$ ，此过程分为两步，简述如下：

#### 3.3.1.1 无碰撞椭球的生成

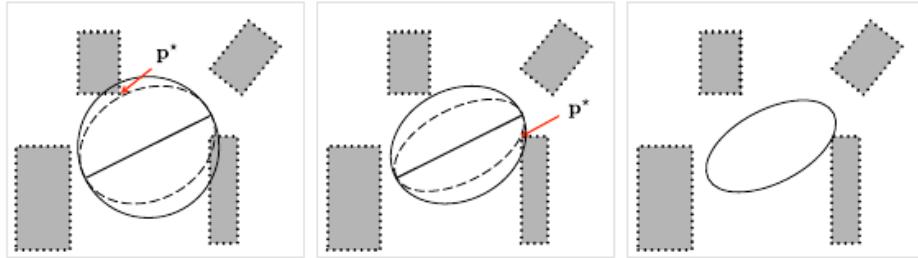


图 3-9 生成无碰撞椭球示意图

一个椭球  $\xi$  可以表示为如下形式：

$$\xi(\mathbf{E}, \mathbf{d}) = \{\mathbf{p} = \mathbf{E}\bar{\mathbf{p}} + \mathbf{d} \mid \|\bar{\mathbf{p}}\| \leq 1\} \quad (3-12)$$

对于一个  $\mathbb{R}^3$  中的椭球来说，变换矩阵  $\mathbf{E}$  可以分解为旋转变换加缩放变换，即：

$$\mathbf{E} = \mathbf{R}^T \mathbf{S} \mathbf{R} = \mathbf{R}^T \text{diag}(a, b, c) \mathbf{R} \quad (3-13)$$

其中  $a, b, c$  分别为椭球坐标系的  $\tilde{x}, \tilde{y}, \tilde{z}$  轴对应的缩放系数。任意一点到椭球的距离定义如式 (3-14)，若  $\rho(\mathbf{p}, \xi) < 1$ ，则  $\mathbf{p}$  位于椭球内部。

$$\rho(\mathbf{p}, \xi(\mathbf{E}, \mathbf{d})) = \|\bar{\mathbf{p}}\| = \|\mathbf{E}^{-1}(\mathbf{p} - \mathbf{d})\| \quad (3-14)$$

求线段  $L$  的无碰撞椭球即是令  $\mathbf{d}$  为线段的中点，固定椭球  $\tilde{x}$  轴与  $L$  重合且  $2a = L$ ，改变  $b$  和  $c$  得到一个尽可能大的、不包含障碍物点的椭球。如图3-9所示，这个过程是一个从初始球体开始迭代收缩的过程，具体流程如算法3-4所示。

#### 3.3.1.2 凸多面体的生成

如图3-10所示，此过程也是通过一步步迭代，对之前生成的无碰撞椭球进行膨

---

**算法 3-4 生成无碰撞椭球**


---

**Input** 线段  $L_i = \langle p_i \rightarrow p_{i+1} \rangle$ , 障碍物点集  $O$

**Output** 无碰撞椭球  $\xi$

- 1: 初始化  $\xi$  为以  $L_i$  为直径的球体, 且以  $p_i p_{i+1}$  为  $\xi$  的  $\tilde{x}$  轴
  - 2:  $O_{inside} \leftarrow \xi.getInside(O)$
  - 3: **while**  $O_{inside} \neq \emptyset$  **do**
  - 4:    $p \leftarrow \min_{q \in O_{inside}} \|E^{-1}(p - d)\|$
  - 5:   保持  $\tilde{x}$  轴不变,  $b = c$ , 调整  $\xi$  使其的边界经过点  $p$
  - 6:    $O_{inside} \leftarrow \xi.getInside(O)$
  - 7: **end while**
  - 8: 此时  $\xi$  接触一障碍物点  $p^*$ , 将其与  $\xi$  的  $\tilde{x}$  轴确定的平面定为  $\xi$  的  $\tilde{x} - \tilde{y}$  平面
  - 9: 沿  $\tilde{z}$  轴调整  $\xi$  使  $c = a$
  - 10:  $O_{inside} \leftarrow \xi.getInside(O)$
  - 11: **while**  $O_{inside} \neq \emptyset$  **do**
  - 12:    $p \leftarrow \min_{q \in O_{inside}} \|E^{-1}(p - d)\|$
  - 13:   沿  $\tilde{z}$  轴调整  $\xi$  使其的边界经过点  $p$
  - 14:    $O_{inside} \leftarrow \xi.getInside(O)$
  - 15: **end while**
  - 16: **return**  $\xi$
- 

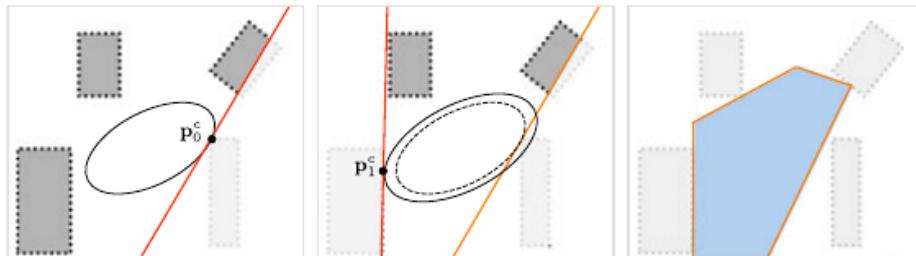


图 3-10 生成无碰撞凸多面体示意图

胀来得到最终的无碰撞凸多面体的, 具体流程见算法3-5。这里将第  $i$  步膨胀得到的半空间 (halfspace) 记作:

$$H_i = \{p \mid a_i^T p < b_i\} \quad (3-15)$$

记最后得到的由  $(m + 1)$  个半空间所确定的凸多面体为:

$$\mathcal{P}(A, b) = \bigcap_{i=0}^m H_i = \{p \mid A^T p < b\}, A \in \mathbb{R}^{3 \times m}, b \in \mathbb{R}^m \quad (3-16)$$

### 3.3.2 效果展示

如图3-11所示为 SFC 生成算法的运行结果的可视化示意图。所使用的地图即是前述 RRT 测试所使用的宽松环境。可以从图中清晰地看到每段路径周围的无碰撞椭球和在这些无碰撞椭球的基础上生成的凸多面体。附近没有障碍物的路径段

**算法 3-5 膨胀椭球生成无碰撞凸多面体**

**Input** 无碰撞椭球  $\xi^0(\mathbf{E}, \mathbf{d})$ , 障碍物点集  $O$

**Output** 凸多面体  $\mathcal{P}(\mathbf{A}, \mathbf{b})$

- 1:  $O_{remain} \leftarrow \xi.getInside(O), j \leftarrow 0$
- 2: **while**  $O_{remain} \neq \emptyset$  **do**
- 3:    $\mathbf{p}_j^c \leftarrow \min_{\mathbf{q} \in O_{inside}} \|\mathbf{E}^{-1}(\mathbf{p} - \mathbf{d})\|$
- 4:   按比例膨胀  $\xi^0$  使其表面经过  $\mathbf{p}_j^c$
- 5:    $\mathbf{a}_j \leftarrow 2\mathbf{E}^{-1}\mathbf{E}^{-T}(\mathbf{p}_j^c - \mathbf{d})$
- 6:    $b_j \leftarrow \mathbf{a}_j^T \mathbf{p}_j^c$
- 7:   去除  $O_{remain}$  中不在半空间  $H_j(\mathbf{a}_j, b_j)$  之内的所有点
- 8:    $j \leftarrow j + 1$
- 9: **end while**
- 10:  $\mathbf{A} \leftarrow [\mathbf{a}_0 \mathbf{a}_1 \dots]^T, \mathbf{b} \leftarrow [b_0 b_1 \dots]^T$
- 11: **return**  $\mathcal{P}(\mathbf{A}, \mathbf{b})$

所生成的椭球保持初始的球形，而附近由障碍物的路径段所生成的椭球则被“压扁”；所有凸多面体串成整个安全飞行走廊，注意这里加入了局部包围框（即一些额外的半空间），以将飞行走廊限制在路径周围一定范围内。

在效率方面，该算法具有可观的计算速度，类似图中的飞行走廊的生成耗时在数十毫秒到百毫秒以内，当然，耗时也会与点云中点的数量呈一定正相关。

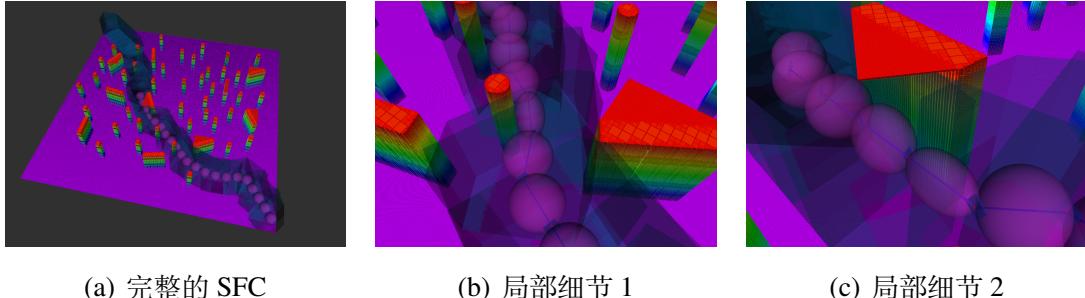


图 3-11 SFC 生成示意图

### 3.4 本章小结

本章首先对  $SE(3)$  空间中的 RRT 算法进行设计，确定了  $SE(3)$  状态空间中的采样、插值和度量等操作，介绍了用于近邻搜索的 GNAT 数据结构，设计了基于点云地图和基于八叉树地图的两种有效性检测方法，并对设计好的 RRT 算法进行了效果测试及分析；随后设计并实现了安全飞行走廊的生成算法，并给出了实测效果。综上所述，本章完成了对轨迹规划前端算法的设计与实现，为后端部分提供了可行路径和安全约束等初始信息。

## 第 4 章 后端轨迹生成算法设计

### 4.1 引言

轨迹规划的后端部分的主要任务是根据前端生成的初始可行路径或安全飞行走廊，通过拟合、优化的方法得到一条具备时间律的连续、平滑且安全的轨迹输入到飞行器的控制器中。本章介绍基于优化的后端轨迹生成算法设计，4.2节介绍几何约束下多旋翼的轨迹优化原理；4.3.1小节构建了几何约束下过驱动多旋翼飞行器 6 自由度轨迹规划的形式；4.3.2小节介绍基于欧拉角姿态表示的过驱动飞行器  $SE(3)$  轨迹生成算法设计；4.3.3小节介绍基于四元数姿态表示的过驱动飞行器  $SE(3)$  轨迹生成算法设计。

### 4.2 几何约束下多旋翼的轨迹优化原理

几何约束下多旋翼的轨迹优化框架 GCOPTER<sup>[15]</sup> 由 Wang 等人于 2021 年提出，该框架在对多旋翼无人机施加复杂的安全约束、苛刻的动力学约束以及自定义需求的情况下依然能保证极高的计算效率和轨迹质量，其优秀性能催生了许多优秀成果<sup>[6,45]</sup>。

GCOPTER 论文的主要贡献概括为以下几点：

- (1) 给出了多阶段控制代价 (control effort) 最小化问题的最优充要条件，并基于此条件设计了最小控制代价轨迹类 MINCO；
- (2) 使用数学技巧消去轨迹时间分配的约束和中间点的空间空间约束；
- (3) 构造时间积分罚函数软化连续时间约束；
- (4) 利用 2 和 3 化约束优化问题为无约束优化问题，使用拟牛顿法 (quasi-Newton methods) 高效求解。

本课题期望将此框架扩展到过驱动飞行器的 6 自由度  $SE(3)$  轨迹规划中，下面先对框架原理进行详细介绍。

#### 4.2.1 问题形式

多旋翼无人机等微分平坦机器人的轨迹规划任务常常在其平坦输出空间中进行，所期望得到的结果是一条平坦输出  $z$  的轨迹  $z(t) : [0, T] \mapsto \mathbb{R}_m$ ，对于 OmniHex 这样的全驱动多旋翼飞行器而言有  $m = 6$ 。

GCOPTER 的优化目标是在满足一定的几何约束和连续时间约束的前提下使

轨迹尽量平滑且尽量满足设定的时间分配要求，这个目标通过最小化时间正则化的导数平方积分控制代价来实现。几何约束包括构型空间中的空间约束，代表飞行器可以处于的安全区域，表达为：

$$z(t) \in \mathcal{F}, \forall t \in [0, T] \quad (4-1)$$

其中  $\mathcal{F}$  为构型空间（平坦输出空间）中的无障碍物区域。连续时间约束包含其他对飞行器状态  $x(t)$  和控制输入  $u(t)$  的所有约束，如动力学限制、其他与特定任务相关约束等。所有这些自定义约束记为：

$$\mathcal{G}_D(x(t), u(t)) \leq \mathbf{0}, \forall t \in [0, T] \quad (4-2)$$

利用微分平坦关系可以将  $\mathcal{G}_D$  转化为对平坦输出及其有限阶导数的约束（图2-9）：

$$\mathcal{G}(z(t), \dot{z}(t), \dots, z^{(s)}(t)) \leq \mathbf{0}, \forall t \in [0, T] \quad (4-3)$$

其中  $\mathcal{G}$  包含  $n_g$  个约束。

假设要最小化平方积分的导数阶数为  $s \in \mathbb{N}_+$ ，即取  $v = z^{(s)}$ ，并记：

$$z^{[s-1]} = \left[ z^T \ z^T \ \cdots \ z^{(s-1)T} \right]^T \in \mathbb{R}^{ms} \quad (4-4)$$

$$z_i^{[s-1]} = \left[ z_i \ \dot{z}_i \ \cdots \ z_i^{(s-1)} \right]^T \in \mathbb{R}^s \quad (4-5)$$

则整个轨迹优化过程即为求解下述约束优化问题：

$$\min_{z(t), T} \int_0^T v(t)^T v(t) dt + \rho(T) \quad (4-6a)$$

$$s.t. \quad v(t) = z^{(s)}(t), \forall t \in [0, T], \quad (4-6b)$$

$$\mathcal{G}(z(t), \dot{z}(t), \dots, z^{(s)}(t)) \leq \mathbf{0}, \forall t \in [0, T], \quad (4-6c)$$

$$z(t) \in \mathcal{F}, \forall t \in [0, T], \quad (4-6d)$$

$$z^{[s-1]}(0) = \bar{z}_o, z^{[s-1]}(T) = \bar{z}_f. \quad (4-6e)$$

其中  $\rho(T)$  为时间正则项，它代表了轨迹总时间所期望满足的性质，是控制代价与总时间之间的权衡。例如，如果希望飞行器尽快到达重点，即总时间越  $T$  短越好，那么可取为  $\rho_s(T) = k_\rho T$ ；如果希望轨迹总时间尽可能接近给定的期望值  $T_\Sigma$ ，那么可取为  $\rho_s(T) = k_\rho(T - T_\Sigma)^2$ ；如果希望  $T$  严格固定为  $T_\Sigma$ ，则可取为：

$$\rho_f(T) = \begin{cases} 0 & if \ T = T_\Sigma \\ \infty & if \ T \neq T_\Sigma \end{cases} \quad (4-7)$$

#### 4.2.2 最优性条件与 MINCO 轨迹类

为了使轨迹更容易地满足安全约束和动力学约束，通常将平坦输出轨迹  $z(t)$  表示为分段多项式，一条  $k$  阶  $M$  段分段多项式轨迹形式如下：

$$z(t) = \mathbf{c}_i^T \boldsymbol{\beta}(t - t_{i-1}), t \in [t_{i-1}, t_i], i = 1, 2, \dots, M \quad (4-8)$$

其中  $\mathbf{c}_i \in \mathbb{R}^{(k+1) \times m}$  为第  $i$  段多项式的系数矩阵， $\boldsymbol{\beta}(t) = [1 \ t \ \dots \ t^N] \in \mathbb{R}^{(k+1)}$  为多项式基底。

先不考虑约束  $\mathcal{F}$  和  $\mathcal{G}$  以及时间正则项  $\rho(T)$ ，考虑如下  $M$  阶段控制代价最小化问题：

$$\min_{\mathbf{z}(t)} \int_{t_0}^{t_M} \mathbf{v}(t)^T \mathbf{v}(t) dt \quad (4-9a)$$

$$s.t. \quad \mathbf{v}(t) = z^{(s)}(t), \forall t \in [t_0, t_M], \quad (4-9b)$$

$$z^{[s-1]}(0) = \bar{z}_o, z^{[s-1]}(T) = \bar{z}_f, \quad (4-9c)$$

$$z^{[d_i-1]}(t_i) = \bar{z}_i, i = 1, 2, \dots, M-1, \quad (4-9d)$$

$$t_{i-1} < t_i, i = 1, 2, \dots, M. \quad (4-9e)$$

在这个问题中，时间段  $[t_0, t_M]$  被  $(M + 1)$  个固定的时间截  $t_1, \dots, t_{M-1}$  分割成  $M$  段；此外，在每段的终点  $t_i$  处平坦输出的前  $d_i - 1 < s$  阶导数也被固定为  $z_i \in \mathbb{R}^{md_i}$ 。在 GCOPTER 的论文中给出了多阶段最优控制问题式 (4-9) 的最优充要条件：

**定理 4.1 (最优充要条件)** 优化问题式 (4-9) 存在唯一一个最优解  $z^*(t)$ ：  
 $[t_0, t_M] \mapsto \mathbb{R}^m$ ；且轨迹  $z$  是优化问题式 (4-9) 的最优解，当且仅当如下条件同时成立：

- (1)  $z^*(t) : [t_0, t_M] \mapsto \mathbb{R}^m$  是一个  $(2s - 1)$  次多项式， $1 \leq i \leq M$ ；
- (2)  $z^*(t)$  满足式 (4-9d) 所示的边界条件；
- (3)  $z^*(t)$  满足式 (4-9c) 所示的中间条件；
- (4) 对任意的  $i = 1, \dots, M-1$ ,  $z^*(t)$  在  $t_i$  处  $\bar{d}_i - 1$  阶连续可微，其中处  $\bar{d}_i = 2s - d_i$ 。

利用这个最优充要条件，就可以无需计算代价泛函，以  $O(M)$  的时间复杂度求出这条最优轨迹。

记式 (4-9) 的最优轨迹  $z^*(t)$  的总系数矩阵为；

$$\mathbf{c} = [\mathbf{c}_1^T \ \dots \ \mathbf{c}_M^T]^T \in \mathbb{R}^{2Ms \times m} \quad (4-10)$$

时间分配向量记为：

$$\mathbf{T} = [t_1 - t_0 \ \dots \ t_M - t_{M-1}]^T = [T_1 \ \dots \ T_M]^T \in \mathbb{R}_+^M \quad (4-11)$$

记式(4-9)中的边界条件为  $\mathbf{D}_0, \mathbf{D}_M \in \mathbb{R}^{s \times m}$ , 中间条件为  $\mathbf{D}_i \in \mathbb{R}^{d_i \times m}$ , 这样根据定理4.1中关于中间时刻  $t_i$  处导数的取值条件和连续性条件可以写为下式:

$$[\mathbf{E}_i \ \mathbf{F}_i] \begin{bmatrix} \mathbf{c}_i \\ \mathbf{c}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_i \\ \mathbf{0}_{\bar{d}_i \times m} \end{bmatrix}, i = 1, 2, \dots, M-1 \quad (4-12)$$

其中  $\mathbf{E}_i, \mathbf{F}_i \in \mathbb{R}^{2s \times 2s}$ , 且有:

$$\mathbf{E}_i = [\boldsymbol{\beta}(T_i) \ \cdots \ \boldsymbol{\beta}^{(d_i-1)}(T_i) \ \boldsymbol{\beta}(T_i) \ \cdots \ \boldsymbol{\beta}^{(\bar{d}_i-1)}(T_i)]^T \quad (4-13)$$

$$\mathbf{F}_i = [\mathbf{0} \ -\boldsymbol{\beta}(0) \ \cdots \ -\boldsymbol{\beta}^{(\bar{d}_i-1)}(0)]^T \quad (4-14)$$

特别地, 定义  $\mathbf{F}_0, \mathbf{E}_M \in \mathbb{R}^{s \times 2s}$  为:

$$\mathbf{F}_0 = [\boldsymbol{\beta}(0) \ \cdots \ \boldsymbol{\beta}^{(s-1)}(0)]^T \quad (4-15)$$

$$\mathbf{E}_M = [\boldsymbol{\beta}(T_M) \ \cdots \ \boldsymbol{\beta}^{(s-1)}(T_M)]^T \quad (4-16)$$

于是, 根据定理4.1可构造出关于最优轨迹系数矩阵  $\mathbf{c}$  的线性方程组:

$$\mathbf{Mc} = \mathbf{b} \quad (4-17)$$

其中  $\mathbf{M} \in \mathbb{R}^{2Ms \times 2Ms}$  仅与时间分配向量  $\mathbf{T}$  有关, 具有带状结构;  $\mathbf{b} \in \mathbb{R}^{2Ms \times m}$  仅与中间条件有关。二者形式如下:

$$\mathbf{M} = \begin{bmatrix} \mathbf{F}_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{E}_1 & \mathbf{F}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_2 & \mathbf{F}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_{M-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E}_M \end{bmatrix} \quad (4-18)$$

$$\mathbf{b} = [\mathbf{D}_0^T \ \mathbf{D}_1^T \ \mathbf{0}_{m \times \bar{d}_i} \ \cdots \ \mathbf{D}_{M-1}^T \ \mathbf{0}_{m \times \bar{d}_{M-1}} \ \mathbf{D}_M^T]^T \quad (4-19)$$

由最优解的存在性和唯一性, 可以保证  $\mathbf{M}$  的非奇异性, 然后利用带状矩阵的PLU分解就可以以  $O(M)$  的线性时间和空间复杂度求解出  $\mathbf{c}$ ; 同时对任意的时间分配、边界条件和中间条件,  $\mathbf{M}$  和  $\mathbf{b}$  均可以以线性复杂度构建。这样, 通过直接应用定理4.1, 无需代价泛函即可用极低的复杂度得到多阶段最优控制问题式(4-9)的最优解。需要注意的是, 如果在算法实现时采用的时直接LU分解, 那么需要调整  $\mathbf{M}$  和  $\mathbf{b}$  中行的顺序来避免主对角线上出现0。

对于多旋翼飞行器来说, 飞行轨迹的安全性通常决定于它的空间特性, 而动力学限制通常决定于它的时间属性, 因此将轨迹参数分为两组: 第一组包含轨迹

必须通过的中间点；第二组包含轨迹中不同片段的时间分配向量。基于这些参数，利用定理4.1即可得到轨迹的系数矩阵  $\mathbf{c}$ ，这个系数矩阵对应的就是在给定的阶数、时间分配和中间条件下具有最小控制量的轨迹，若  $s = 4$ ，那么得到的就是 Mellinger 等人提出的最小化加速度轨迹<sup>[3]</sup>。Wang 等人将这类以中间点和时间分配为参数，使用定理4.1构建出的轨迹称作 MINCO 轨迹<sup>[15]</sup>。

记中间点为  $\mathbf{q} = [\mathbf{q}_1 \cdots \mathbf{q}_{M-1}] \in \mathbb{R}^{m \times (M-1)}$ ， $\mathbf{q}_i \in \mathbb{R}^m$  为指定在  $t_i$  时刻的中间条件 ( $d_i - 1 = 0$ )，则 MINCO 轨迹定义为  $\mathcal{T}_{\text{MINCO}}$  定义为

$$\mathcal{T}_{\text{MINCO}} \stackrel{\text{def}}{=} \{\mathbf{p}(t) : [t_0, t_M] \mapsto \mathbb{R}^m \mid \mathbf{c} = \mathbf{c}(\mathbf{q}, \mathbf{T}), \forall \mathbf{q} \in \mathbb{R}^{m \times (M-1)}, \mathbf{T} \in \mathbb{R}_+^M\} \quad (4-20)$$

其中  $\mathbf{c}(\mathbf{q}, \mathbf{T})$  即是由定理4.1决定的整条轨迹的系数矩阵，这样轨迹的参数就由系数转化为了  $\mathbf{q}$  和  $\mathbf{T}$ ，实现了参数降维。

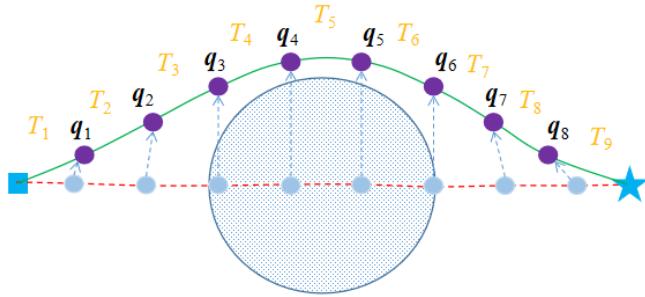


图 4-1 MINCO 轨迹的空间变形示意图

现在如果有约束存在，则调整  $\mathbf{q}$  就可以对  $\mathcal{T}_{\text{MINCO}}$  进行空间上的变形，使整条轨迹不与障碍物发生碰撞（图4-1）；调整  $\mathbf{T}$  则可以对  $\mathcal{T}_{\text{MINCO}}$  进行时间上的变形，使之满足动力学限制。GCOPTER 框架就是通过 MINCO 的时空变形来优化目标函数，从而得到一条给定需求下的最优轨迹。

#### 4.2.3 MINCO 轨迹类中的梯度计算

上小节提到，基于 MINCO 的轨迹优化框架通过调整轨迹参数  $\mathbf{q}$  和  $\mathbf{T}$  来优化给定的目标函数。及自定义的轨迹优化优化目标函数为一个二阶连续可微的标量函数  $F(\mathbf{c}, \mathbf{T})$ ，且其梯度已知，那么这个目标函数在  $\mathcal{T}_{\text{MINCO}}$  中可以计算如下：

$$H(\mathbf{q}, \mathbf{T}) = F(\mathbf{c}(\mathbf{q}, \mathbf{T}), \mathbf{T}) \quad (4-21)$$

在对目标函数进行优化时，通常需要用到其梯度信息，即需要计算  $\partial H / \partial \mathbf{q}$  和  $\partial H / \partial \mathbf{T}$ 。显然计算  $H$  的值与求解  $\mathcal{T}_{\text{MINCO}}$  中任意一条轨迹（即计算  $\mathbf{c}(\mathbf{q}, \mathbf{T})$ ）具有相同的复杂度  $O(M)$ ，下面介绍从已知的  $\partial F / \partial \mathbf{c}$  和  $\partial F / \partial \mathbf{T}$  计算  $\partial H / \partial \mathbf{q}$  和  $\partial H / \partial \mathbf{T}$  的线性复杂度方法。

首先给出如下结论，这里省去推导过程：

$$\frac{\partial H}{\partial \mathbf{q}_i} = \left( \mathbf{M}^{-T} \frac{\partial F}{\partial \mathbf{c}} \right)^T \mathbf{e}_{(2i-1)s+1} \quad (4-22)$$

其中  $\mathbf{e}_j$  代表单位矩阵  $\mathbf{I}_{2Ms}$  的第  $j$  列。计算过程中需要对  $\mathbf{M}^T$  求逆，直接的求逆运算具有  $O(M^3)$  的时间复杂度。注意到先前求解  $\mathbf{c}$  时对  $\mathbf{M}$  进行了 PLU 分解，可以利用这里的结果来避免求逆，减小复杂度。令式 (4-22) 中括号中表达式的结果为  $\mathbf{G} \in \mathbb{R}^{2Ms \times m}$ ，则得到关于  $\mathbf{G}$  的线性方程组：

$$\mathbf{M}^T \mathbf{G} = \frac{\partial F}{\partial \mathbf{c}} \quad (4-23)$$

记  $M$  的 PLU 分解结果为  $\mathbf{M} = \mathbf{P}\mathbf{L}\mathbf{U}$ ，则  $\mathbf{M}^T$  也存在 PLU 分解，有  $\mathbf{M}^T = \bar{\mathbf{L}}\bar{\mathbf{U}}\mathbf{P}^T$ ，其中

$$\bar{\mathbf{L}} = \mathbf{U}^T (\mathbf{U} \circ \mathbf{I})^{-1}, \bar{\mathbf{U}} = (\mathbf{U} \circ \mathbf{I}) \mathbf{L}^T \quad (4-24)$$

其中需要求逆的只是一个对角矩阵，符号  $\circ$  为矩阵的 Hadamard 乘积。那么现在就可以用线性复杂度求出  $\mathbf{G}$  了，为方便起见，将  $\mathbf{G}$  写成如下分块形式：

$$\mathbf{G} = \left[ \mathbf{G}_0^T \ \mathbf{G}_1^T \ \cdots \ \mathbf{G}_{M-1}^T \ \mathbf{G}_M^T \right]^T \quad (4-25)$$

其中  $\mathbf{G}_0, \mathbf{G}_M \in \mathbb{R}^{s \times m}$  而  $\mathbf{G}_i \in \mathbb{R}^{2s \times m}, i = 1, 2, \dots, M-1$ 。这样  $H$  对  $\mathbf{q}$  的梯度就可以写为：

$$\frac{\partial H}{\partial \mathbf{q}} = \left[ \mathbf{G}_1^T \mathbf{e}_1 \ \cdots \ \mathbf{G}_{M-1}^T \mathbf{e}_1 \right] \quad (4-26)$$

相似地可以得到  $H$  对  $T_i$  的梯度：

$$\frac{\partial H}{\partial T_i} = \frac{\partial F}{\partial T_i} - \text{tr}\{\mathbf{G}_i^T \frac{\partial \mathbf{E}_i}{\partial T_i} \mathbf{c}_i\} \quad (4-27)$$

#### 4.2.4 几何约束的消去

在 MINCO 轨迹类中，进行时空变形可使轨迹在满足可行性约束的同时保持局部光滑性。基于  $\mathcal{T}_{\text{MINCO}}$  的轨迹优化问题的形式如下：

$$\min_{\mathbf{q}, \mathbf{T}} J(\mathbf{q}, \mathbf{T}) = J_q(\mathbf{q}, \mathbf{T}) + \rho(\|\mathbf{T}\|_1) \quad (4-28a)$$

$$s.t. \quad \mathcal{G}(\mathbf{z}(t), \dot{\mathbf{z}}(t), \dots, \mathbf{z}^{(s)}(t)) \leq \mathbf{0}, \forall t \in [0, T], \quad (4-28b)$$

$$\mathbf{q}_i \in \mathcal{P}_i, i = 1, 2, \dots, M-1, \quad (4-28c)$$

$$\mathbf{T} > \mathbf{0}, \quad (4-28d)$$

$$\mathbf{z}^{[s-1]}(0) = \bar{\mathbf{z}}_o, \mathbf{z}^{[s-1]}(T) = \bar{\mathbf{z}}_f. \quad (4-28e)$$

其中  $J_q(\mathbf{q}, \mathbf{T}) := J_c(\mathbf{c}, \mathbf{T})$ ,  $J_c(\mathbf{c}, \mathbf{T})$  为由轨迹多项式系数和时间分配确定的控制代价, 具有解析的表达式和梯度<sup>[15]</sup>;  $\mathcal{P}_i$  为第  $i$  个中间点  $\mathbf{q}_i$  所被分配到的 SFC 凸多面体。

几何约束式(4-28c)和式(4-28d)是必要的, 但会在一定程度上影响优化速度下面介绍消去几何约束使时空变形免受不等式约束组合困难 (combinatorial difficulty) 的限制的方法。

#### 4.2.4.1 时间约束的消去

对时间分配向量  $\mathbf{T}$  的两种约束会对带约束优化过程产生不利影响:

- (1) 如式(4-28d)所示的那样, 要求  $\mathbf{T}$  中的每个元素都大于 0, 这是由实际意义决定的, 并且如图4-2所示, 当  $\mathbf{T}$  中的任意一个元素在优化过程中接近于 0 时, 都会导致  $J_q$  趋向于无穷大, 因为  $\mathbf{q}$  中不存在两个相邻且相同的点;
- (2) 如图4-2所示, 当  $\rho = \rho_f$  时, 会加入一个额外的约束  $\sum_{i=1}^{M-1} T_i < T_\Sigma$ , 使得  $\mathbf{T}$  的可行区域进一步缩减。

在上述两种情况下, 病态的 Hessian 矩阵以及可行域边界处出现的不可行搜索步长都会拖慢优化过程的收敛。在 GCOPTER 中, 采取微分同胚映射 (diffeomorphism) 的手段来消去上述时间约束。

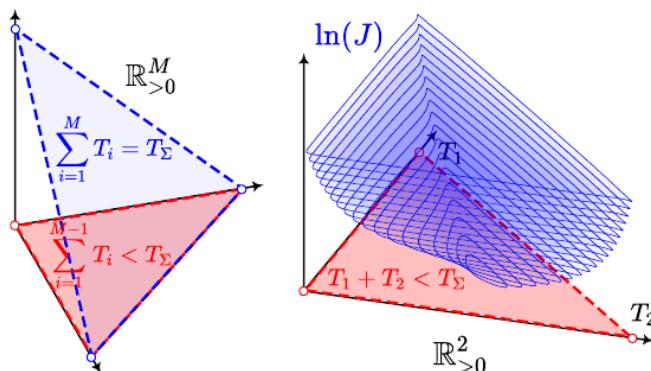


图 4-2 时间约束示意图

对于要求总时间严格固定, 即  $\rho = \rho_f$  的情形,  $\mathbf{T}$  的可行域描述为如下集合:

$$\mathcal{T}_f = \{\mathbf{T} \in \mathbb{R}_+^M \mid \|\mathbf{T}\|_1 = T_\Sigma\} \quad (4-29)$$

那么可以证明  $\mathcal{T}_f$  与  $\mathbb{R}^{M-1}$  是微分同胚的, 其中一个  $C^\infty$  的微分同胚映射如式(4-30)所示

$$T_i = \frac{e^{\tau_i}}{1 + \sum_{j=1}^{M-1} e^{\tau_j}} T_\Sigma, i = 1, 2, \dots, M-1, \quad (4-30a)$$

$$T_M = T_\Sigma - \sum_{i=1}^{M-1} T_i \quad (4-30b)$$

利用这个映射，把目标变量由受约束的  $\mathbf{T}$  转化为自由变量  $\tau$ ，这样就摆脱了可行域边界附近的不利条件，且此时时间正则项恒为 0。若令  $\partial J_q / \partial \mathbf{T} = [\mathbf{g}_a^T \mathbf{g}_b]^T$ ，则  $J$  对  $\tau$  的梯度如下式所示：

$$\frac{\partial J}{\partial \tau} = \left( \frac{(\mathbf{g}_a - \mathbf{g}_b \cdot \mathbf{1}) \circ e^{[\tau]} }{1 + \|e^{[\tau]}\|_1} - \frac{(\mathbf{g}_a^T e^{[\tau]} - g_b \|e^{[\tau]}\|_1) e^{[\tau]}}{(1 + \|e^{[\tau]}\|_1)^2} \right) T_\Sigma \quad (4-31)$$

其中  $e^{[\cdot]}$  表示向量·按元素求指数函数值。那么可以看到，由  $\tau$  正向计算  $J$  以及反向计算梯度仍然只需要  $O(M)$  的时间和空间复杂度。对于总时间不严格固定，即  $\rho = \rho_s$  的情形，可以用  $\mathbf{T} = e^{[\tau]}$  作为从  $\mathbb{R}^M$  到作为从  $\mathbb{R}_+^M$  的微分同胚映射，注意此时在计算目标函数值和梯度时需要加上时间正则项。

可以证明，上述变换不会抹去  $J$  原有的局部最小值点，也不会引入额外的局部最小值点。

#### 4.2.4.2 空间约束的消去

对于式 (4-1) 所示的空间约束，要求对整条轨迹上的点都成立。对于由凸多面体组成的复杂空间约束，这个要求通过两个阶段来实现：(1) 按顺序将中间点分配到组成安全飞行走廊的凸多面体中，如式 (4-28c) 所示；(2) 在给定分辨率下使整条轨迹满足空间约束。本小节处理阶段 (1)，阶段 (2) 的处理将在 4.2.5 小节介绍。

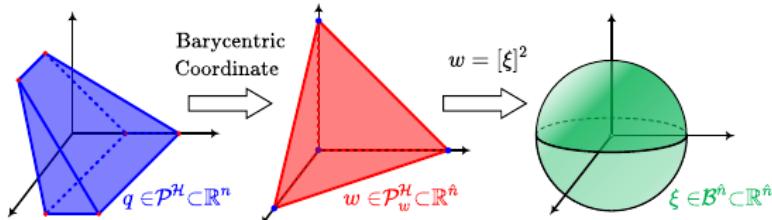


图 4-3 凸多面体约束转化示意图

记一个  $\mathbb{R}^n$  中的凸多面体  $\mathcal{P}$  的  $(\hat{n}+1)$  个顶点为  $[\mathbf{v}_0 \cdots \mathbf{v}_{\hat{n}}]$ ，记  $\hat{\mathbf{v}}_i = \mathbf{v}_i - \mathbf{v}_0$ ，及  $\hat{\mathbf{V}} = [\hat{\mathbf{v}}_1 \cdots \hat{\mathbf{v}}_{\hat{n}}]$ ，这样每一个中间点  $\mathbf{q} \in \mathbb{R}^n$  在重心坐标系 (barycentric coordinate system) 下就可以表示为：

$$\mathbf{q} = \mathbf{v}_0 + \hat{\mathbf{V}}\mathbf{w} \quad (4-32)$$

其中  $\mathbf{w} = [w_1 \cdots w_{\hat{n}}]^T \in \mathbb{R}^{\hat{n}}$  为重心坐标的后  $\hat{n}$  个元素。凸多面体  $\mathcal{P}$  内的每一个点对应的  $\mathbf{w}$  构成的点集是  $\mathbb{R}^{\hat{n}}$  中的标准单纯形 (standard simplex)：

$$\mathcal{P}_w^{\hat{n}} = \{\mathbf{w} \in \mathbb{R}^{\hat{n}} \mid \mathbf{w} \geq \mathbf{0}, \|\mathbf{w}\|_1 \leq 1\} \quad (4-33)$$

这样凸多面体  $\mathcal{P}$  就可以用 V-表示法表达为：

$$\mathcal{P} = \{v_0 + \hat{V}w \mid w \in \mathcal{P}_w^{\hat{n}}\} \quad (4-34)$$

换句话说，凸多面体约束  $q \in \mathcal{P}$  实际上被转化为了一个  $\mathbb{R}^{\hat{n}}$  中的标准单纯形约束  $w \in \mathcal{P}_w^{\hat{n}}$  (图4-3)。此单纯形约束还可以进一步用非线性变换消去：首先用按元素的平方变换  $w = [x]^2, \forall x \in \mathbb{R}^{\hat{n}}$  消去非负约束，这样单纯形约束  $w \in \mathcal{P}_w^{\hat{n}}$  转化为了球约束  $x \in \mathcal{B}^{\hat{n}}$ ，其中：

$$\mathcal{B}^{\hat{n}} = \{x \in \mathbb{R}^{\hat{n}} \mid \|x\|_2 \leq 1\} \quad (4-35)$$

然后再如图4-4所示使用逆球极投影 (inverse stereographic projection) 变换和一个

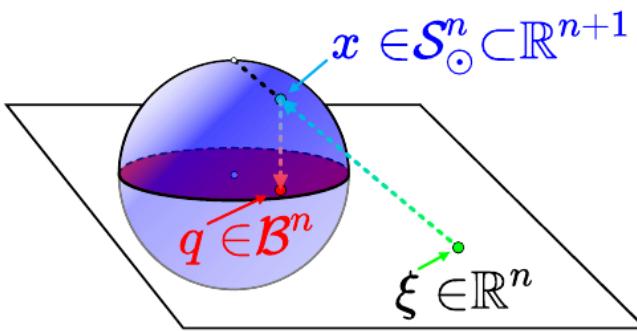


图 4-4 球约束消去示意图

正交投影变换可以消去球约束，最后得到一个从整个  $\mathbb{R}^{\hat{n}}$  到凸多面体  $\mathcal{P}$  的映射：

$$f_{\mathcal{H}}(\xi) = v_0 + \frac{4\hat{V}[\xi]^2}{(\xi^T \xi + 1)^2} \in \mathcal{P}, \forall \xi \in \mathbb{R}^{\hat{n}} \quad (4-36)$$

当  $\xi$  在整个  $\mathbb{R}^{\hat{n}}$  中自由运动时，经由映射  $f_{\mathcal{H}}$  得到的  $\mathbb{R}^{\hat{n}}$  中的像始终位于凸多面体  $\mathcal{P}$  内。这样，与时间约束类似，用凸多面体表示的空间约束被用引入新的目标变量  $\xi$  的方式消去了。在优化过程开始之前会给定的初始  $q_i$ ，其对应的初始  $\xi_i$  可以通过最小化  $\|f_{\mathcal{H}}(\xi_i) - q_i\|_2^2$  的值来得到。

若记  $\partial J / \partial q_i$ ，则  $J$  关于对应的  $\xi_i$  的梯度就可以写为：

$$\frac{\partial J}{\partial \xi_i} = \frac{8\xi_i \circ \hat{V}^T g_i}{(\xi_i^T \xi_i + 1)^2} - \frac{16g_i^T \hat{V}[\xi]^2}{(\xi_i^T \xi_i + 1)^3} \xi_i \quad (4-37)$$

上述变换同样不会抹去  $J$  原有的局部最小值点，也不会引入额外的局部最小值点。

#### 4.2.5 连续时间约束 $\mathcal{G}$ 的处理

式 (4-28b) 表达的连续时间  $\mathcal{G}$  约束包含无穷多个不等式约束，计算机难以处

理。GCOPTER 中使用时间积分罚函数 (time integral penalty function) 法来处理  $\mathcal{G}$ , 通过对约束越界量的积分, 将  $\mathcal{G}$  软化为了目标函数的一部分。

对任意一条轨迹  $\mathbf{z}(t) : [0, T] \mapsto \mathbb{R}^m$ , 定义:

$$\mathbf{I}_{\mathcal{G}}^k[\mathbf{z}] \stackrel{\text{def}}{=} \int_0^T \mathbf{K}(\mathcal{G}(\mathbf{z}(t), \dots, \mathbf{z}^{(s)}(t))) dt \quad (4-38)$$

其中  $k \in \mathbb{R}_+$ ,  $\mathbf{K}(\cdot) = \max[\cdot, \mathbf{0}]^k$  代表先按行与 0 比较取最大值组成最大值向量, 然后将最大值向量按元素取  $k$  次幂。那么轨迹  $\mathbf{z}$  的时间积分罚函数定义为:

$$I_{\mathcal{G}}[\mathbf{z}] \stackrel{\text{def}}{=} \boldsymbol{\chi}^T \mathbf{I}_{\mathcal{G}}^3[\mathbf{z}] \quad (4-39)$$

其中  $\boldsymbol{\chi} \in \mathbb{R}_{\geq}^{n_g}$  为惩罚权重向量, 令  $k=3$  可以保证罚函数的二阶可微性。通常惩罚权重的元素应该足够大, 这样如果没有约束被打破, 罚函数值将保持为 0; 而一旦  $\mathbf{z}(t)$  的某个部分打破了约束, 罚函数值就会急剧增加。通过把  $I_{\mathcal{G}}[\mathbf{z}]$  作为一项加入目标代价函数, 优化后的轨迹将会倾向于满足连续时间约束  $\mathcal{G}$ 。

由于计算机无法直接处理连续函数, 故使用数值积分的方式计算罚函数  $I_{\mathcal{G}}[\mathbf{z}]$  的值。定义采样函数  $\mathcal{G}_v : \mathbb{R}^{2s \times m} \times \mathbb{R}_+ \times [0, 1] \mapsto \mathbb{R}^{n_g}$  为:

$$\mathcal{G}_v(\mathbf{c}_i, T_i, \hat{t}) = \mathcal{G}\left(\mathbf{c}_i^T \boldsymbol{\beta}(T_i \cdot \hat{t}), \dots, \mathbf{c}_i^T \boldsymbol{\beta}^{(s)}(T_i \cdot \hat{t})\right) \quad (4-40)$$

则第  $i$  段轨迹的数值积分  $I_i : \mathbb{R}^{2s \times m} \times \mathbb{R}_+ \times \mathbb{N} \mapsto \mathbb{R}_+$  为:

$$I_i(\mathbf{c}_i, T_i, \kappa_i) = \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \boldsymbol{\chi}^T \mathbf{K}(\mathcal{G}_v(\mathbf{c}_i, T_i, \frac{j}{\kappa_i})) \quad (4-41)$$

其中自然数  $\kappa_i$  决定了数值积分的分辨率, 影响着最后优化出的轨迹在多大程度上倾向于满足约束  $\mathcal{G}$ 。将每段轨迹式 (4-41) 相加即得到整条轨迹时间积分罚函数的数值近似:

$$I_{\Sigma}(\mathbf{c}, \mathbf{T}) = \sum_{i=1}^M I_i(\mathbf{c}_i, T_i, \kappa_i) \quad (4-42)$$

进一步, 在  $\mathcal{T}_{\text{MINCO}}$  中式 (4-42) 可以用  $I_{\Sigma}(\mathbf{c}(\mathbf{q}, \mathbf{T}), \mathbf{T})$  计算。其梯度可以用 4.2.3 小节所介绍的方法以  $O(M)$  的复杂度计算出来。

经过上述约束消去和软化操作, 基于 MINCO 的轨迹优化就从约束优化问题式 (4-28) 转化为以下无约束优化问题:

$$\min_{\xi, \tau} J(\mathbf{q}(\xi), \mathbf{T}(\tau)) + I_{\Sigma}(\mathbf{c}(\mathbf{q}(\xi), \mathbf{T}(\tau)), \mathbf{T}(\tau)) \quad (4-43)$$

### 4.3 几何约束下过驱动飞行器的 $SE(3)$ 轨迹优化算法设计

基于几何约束轨迹优化框架 GCOPTER, Han 等人开发出了针对欠驱动四旋

翼飞行器的  $SE(3)$  轨迹规划器 Fast-Racing<sup>[6]</sup>，其计算效率与此前仅有的欠驱动多旋翼飞行器的  $SE(3)$  规划框架<sup>[7]</sup>相比有压倒性的优势，充分证明了 GCOPTER 在（欠驱动多旋翼飞行器） $SE(3)$  规划上的适用性。本课题的目标是将这种适用性拓展到 OmniHex 等过驱动多旋翼飞行器上。

与欠驱动多旋翼飞行器不同的是，过驱动多旋翼飞行器可以跟踪 6 自由度轨迹，即姿态轨迹也可以独立跟踪而不与位置部分耦合。因为这个原因，在为过驱动多旋翼飞行器设计几何约束下的轨迹优化算法时，需要为 6 自由度轨迹建立自己的问题形式，并且重新设计相应的惩罚项  $I_\Sigma(\mathbf{c}, \mathbf{T})$ 。本节首先构建出针对过驱动多旋翼飞行器 6 自由度轨迹优化的问题形式，随后分别就欧拉角和四元数两种姿态表示法下的轨迹优化算法进行设计。

### 4.3.1 优化问题构建

#### 4.3.1.1 6 自由度轨迹优化问题的形式

根据全（过）驱动飞行器的平坦输出形式（式 (2-19)），目标轨迹  $\mathbf{z}(t) : [t_0, t_M] \mapsto \mathbb{R}^6$  可以表示为：

$$\mathbf{z}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \boldsymbol{\sigma}(t) \end{bmatrix} \in \mathbb{R}^6, \forall t \in [t_0, t_M] \quad (4-44)$$

根据定理4.1，将  $\mathbf{z}(t)$  表示为  $(2s - 1)$  分段多项式：

$$\mathbf{z}(t) = \mathbf{c}_i^\top \boldsymbol{\beta}(t - t_{i-1}) = \left[ \mathbf{c}_i^P \ \mathbf{c}_i^\sigma \right]^\top \boldsymbol{\beta}(t - t_{i-1}), \forall t \in [t_{i-1}, t_i], i = 1, 2, \dots, M \quad (4-45)$$

其中  $\mathbf{c}_i \in \mathbb{R}^{2s \times 6}$  为第  $i$  段轨迹的系数矩阵，而  $\mathbf{c}_i^P \in \mathbb{R}^{2s \times 3}$  和  $\mathbf{c}_i^\sigma \in \mathbb{R}^{2s \times 3}$  分别为系数矩阵对应于位置和姿态参数的部分。进一步，我们令分段多项式轨迹  $\mathbf{z}(t)$  的中间点为：

$$\mathbf{q} = \left[ \mathbf{q}_1 \cdots \mathbf{q}_{M-1} \right] \in \mathbb{R}^{6 \times (M-1)}, \mathbf{z}(t_i) = \mathbf{q}_i \in \mathbb{R}^6, \forall i \in \{1, 2, \dots, M-1\} \quad (4-46)$$

其中  $\mathbf{q}$  可分为位置部分和姿态部分：

$$\mathbf{q}_i = \left[ \mathbf{q}_i^{P^\top} \ \mathbf{q}_i^{\sigma^\top} \right]^\top; \mathbf{q}_i^P, \mathbf{q}_i^\sigma \in \mathbb{R}^3 \quad (4-47)$$

$$\mathbf{q} = \left[ \mathbf{q}^{P^\top} \ \mathbf{q}^{\sigma^\top} \right]^\top; \mathbf{q}^P, \mathbf{q}^\sigma \in \mathbb{R}^{3 \times (M-1)} \quad (4-48)$$

在基于 MINCO 的原始优化问题式 (4-28) 中，直接对整个平坦输出中间点施加了硬空间约束（式 (4-28c)），即硬性要求这个中间点对应的平坦输出必须是可行的、与障碍物无碰撞的。换到本问题的场景下，就需要对  $SE(3)$  非欧构型空间中

的可行区域进行表达，这是十分困难的。于是本课题针对性地对问题形式式(4-28)进行了修改：仅对中间点的位置部分施加硬性空间约束  $\mathbf{q}_i^p \in \mathcal{P}_i$ ，可行区域  $\mathcal{P}_i$  由  $\mathbb{R}^3$  中的安全飞行走廊表达，而与姿态相关的飞行器整体位于安全区域内的约束将完全作为  $\mathcal{G}$  的一部分用4.2.5小节所介绍的方法处理，不对  $\mathbf{q}_i^\sigma$  作任何硬约束。

于是，本课题中设计的过驱动飞行器 6 自由度轨迹优化问题的形式如下：

$$\min_{\xi, \mathbf{q}^\sigma, \tau} J(\mathbf{c}(\mathbf{q}^p(\xi), \mathbf{q}^\sigma, \mathbf{T}(\tau)), \mathbf{T}(\tau)) + I_\Sigma(\mathbf{c}(\mathbf{q}^p(\xi), \mathbf{q}^\sigma, \mathbf{T}(\tau)), \mathbf{T}(\tau)) \quad (4-49)$$

其中：

$$J(\mathbf{c}, \mathbf{T}) = J_c(\mathbf{c}, \mathbf{T}) + \rho(\|\mathbf{T}\|_1) = \int_{t_0}^{t_M} \mathbf{z}^{(s)}(t)^T \mathbf{z}^{(s)}(t) dt + k_\rho \|\mathbf{T}\|_1 \quad (4-50)$$

其关于  $\mathbf{c}$  和  $T$  的梯度为：

$$\frac{\partial J}{\partial \mathbf{c}} = \frac{\partial J_c}{\partial \mathbf{c}} \quad (4-51)$$

$$\frac{\partial J}{\partial \mathbf{T}} = \frac{\partial J_c}{\partial \mathbf{T}} + k_\rho \mathbf{1} \quad (4-52)$$

式中  $J_c$  及其梯度在 GCOPTER 的论文中给出了解析表达式<sup>[15][28]</sup>。

#### 4.3.1.2 连续时间约束 $\mathcal{G}$ 及时间积分罚函数的形式

几何约束轨迹优化框架的一个很大的优势在于，可以通过设计不同的连续时间约束  $\mathcal{G}$  使之灵活且高效地适应不同的任务需求。本课题的  $\mathcal{G}$  包括对速度、加速度、角速度大小的限制（动力学约束）以及希望飞机整体（近似为图3-5所示的长方体）位于安全飞行走廊内的  $SE(3)$  安全约束。

固定每段轨迹的数值积分分辨率为  $\kappa \in \mathbb{Z}_+$ ，于是对于动力学约束，直接写出其时间积分罚函数的形式：

$$E_v = W_v \int_{t_0}^{t_M} \mathcal{K}(\|\dot{\mathbf{p}}(t)\|_2^2 - v_{max}^2) dt \quad (4-53)$$

$$E_a = W_a \int_{t_0}^{t_M} \mathcal{K}(\|\ddot{\mathbf{p}}(t)\|_2^2 - a_{max}^2) dt \quad (4-54)$$

$$E_\omega = W_\omega \int_{t_0}^{t_M} \mathcal{K}(\|\boldsymbol{\omega}(t)\|_2^2 - \omega_{max}^2) dt \quad (4-55)$$

式中  $W_v$ 、 $W_a$  和  $W_\omega$  分别为对应惩罚项的权重， $\mathcal{K}(\cdot) = \max(\cdot, 0)^3$ 。

由凸集的性质，飞行器整体包含在安全飞行走廊内等价于长方体的 8 个顶点全部在安全飞行走廊内，因此如图4-5所示，其实可以将飞行器的形状用任意合适形状的凸多面体近似<sup>[6]</sup>。长方体 8 个顶点在机体坐标系  $\mathcal{F}_b$  下的坐标为：

$$\hat{\mathbf{q}}_v = \left[ \pm \frac{l_x}{2} \pm \frac{l_y}{2} \pm \frac{l_z}{2} \right]^T, v = 1, 2, \dots, 8 \quad (4-56)$$

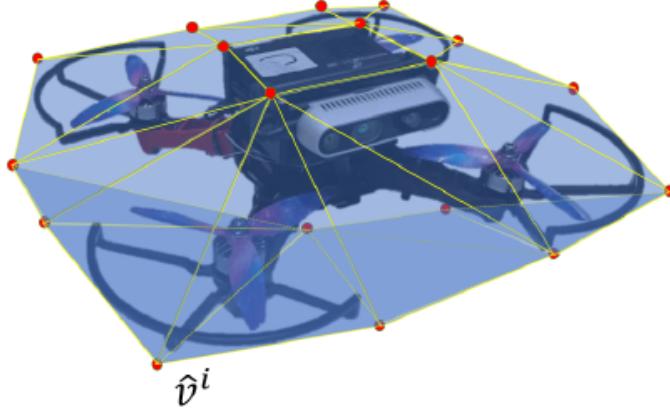


图 4-5 用凸多面体近似飞行器的形状

则轨迹上  $t$  时刻这 8 个顶点在世界坐标系  $\mathcal{F}_W$  下的坐标为：

$$\mathbf{q}_v(t) = \mathbf{p}(t) + \mathbf{R}(\sigma(t))\hat{\mathbf{q}}_v, v = 1, 2, \dots, 8 \quad (4-57)$$

其中  $\mathbf{R}(\sigma(t))$  二阶连续可微。

在优化过程中，我们把每段轨迹都约束到一个凸多面体中，第  $i$  段轨迹所在的凸多面体用 H-表示法用线性不等式约束表示为：

$$\mathcal{P}_i = \left\{ \mathbf{p} \in \mathbb{R}^3 \mid \mathbf{a}_{i,k}^\top \mathbf{p} - b_{i,k} \leq 0, \|\mathbf{a}_{i,k}\|_2 = 1, k = 1, 2, \dots, K_i \right\} \quad (4-58)$$

接下来就可以根据<sup>[9]</sup> 的思想构造出关于  $SE(3)$  安全约束的时间积分碰撞惩罚：

$$E_c = W_c \sum_{i=1}^M \int_{t_{i-1}}^{t_i} \sum_{v=1}^8 \sum_{k=1}^{K_i} \mathcal{K}(\mathbf{a}_{i,k}^\top \mathbf{q}_v(t) - b_{i,k}) dt \quad (4-59)$$

将式 (4-53)、式 (4-54)、式 (4-55) 和式 (4-59) 用矩形法则进行近似得：

$$E_v \approx I_v(\mathbf{c}, \mathbf{T}) = W_v \sum_{i=1}^M \sum_{j=1}^{\kappa} \mathcal{K}(\left\| \dot{\mathbf{p}}(t_{i-1} + \frac{j}{\kappa} T_i) \right\|_2^2 - v_{max}^2) \frac{T_i}{\kappa} \quad (4-60)$$

$$E_a \approx I_a(\mathbf{c}, \mathbf{T}) = W_a \sum_{i=1}^M \sum_{j=1}^{\kappa} \mathcal{K}(\left\| \ddot{\mathbf{p}}(t_{i-1} + \frac{j}{\kappa} T_i) \right\|_2^2 - a_{max}^2) \frac{T_i}{\kappa} \quad (4-61)$$

$$E_\omega \approx I_\omega(\mathbf{c}, \mathbf{T}) = W_\omega \sum_{i=1}^M \sum_{j=1}^{\kappa} \mathcal{K}(\left\| \omega(t_{i-1} + \frac{j}{\kappa} T_i) \right\|_2^2 - \omega_{max}^2) \frac{T_i}{\kappa} \quad (4-62)$$

$$E_c \approx I_c(\mathbf{c}, \mathbf{T}) = W_c \sum_{i=1}^M \sum_{j=1}^{\kappa} \sum_{v=1}^8 \sum_{k=1}^{K_i} \mathcal{K}(\mathbf{a}_{i,k}^\top \mathbf{q}_v(t_{i-1} + \frac{j}{\kappa} T_i) - b_{i,k}) \frac{T_i}{\kappa} \quad (4-63)$$

下面求取上述罚函数关于  $\mathbf{c}$  和  $\mathbf{T}$  的梯度。观察罚函数的结构，其由一系列关于  $\mathcal{K}(\cdot) = \max(\cdot, 0)^3$  的项求和得到，我们只需计算每个求和项为正，即超出约束部分的梯度；对于约束满足的部分，对应梯度则为 0。我们令：

$$I_{v_{ij}} = W_v (\|v\|_2^2 - v_{max}^2)^3 \frac{T_i}{\kappa} \quad (4-64)$$

$$I_{a_{ij}} = W_a (\|a\|_2^2 - a_{max}^2)^3 \frac{T_i}{\kappa} \quad (4-65)$$

$$I_{\omega_{ij}} = W_\omega (\|\omega\|_2^2 - \omega_{max}^2)^3 \frac{T_i}{\kappa} \quad (4-66)$$

$$I_{c_{ijvk}} = W_c (\mathbf{a}_{i,k}^\top \mathbf{q}_v - b_{i,k})^3 \frac{T_i}{\kappa} \quad (4-67)$$

其中  $v, a, \omega, q_v$  和  $R$  为  $t = t_{i-1} + \frac{j}{\kappa} T_i$  时的取值。则式 (4-64)、式 (4-65)、式 (4-66) 和式 (4-67) 分别关于  $c$  和  $T$  的梯度见附录A。

由于旋转矩阵  $R$  和角速度  $\omega$  等与姿态有关的量与  $\sigma$  紧密相关，故上述罚函数正向求值和反向求梯度的过程会因姿态参数化方法的不同而不同。可以选取各种合理的姿态参数化方法来规划姿态轨迹  $\sigma(t)$ ，只要使  $R$  和  $\omega$  等姿态相关量关于  $\sigma$  二阶连续可微即可，不同的姿态参数化方法给轨迹优化问题式 (4-43) 所带来的差异将只体现在惩罚项  $I_\Sigma$  中。

### 4.3.2 基于欧拉角姿态表示的过驱动飞行器 $SE(3)$ 轨迹生成

本小节采用 ZYX 欧拉角来表示姿态，即取轨迹  $z(t)$  的姿态部分  $\sigma(t)$  为：

$$\sigma(t) = \epsilon(t) = \begin{bmatrix} \phi(t) \\ \theta(t) \\ \psi(t) \end{bmatrix} \quad (4-68)$$

则旋转矩阵  $R$  和角速度  $\omega$  可表示为：

$$R(\sigma) = R(\epsilon) = \begin{bmatrix} \cos \psi \cos \theta & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\ \sin \psi \cos \theta & \sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi \\ -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix} \quad (4-69)$$

$$\omega(\sigma, \dot{\sigma}) = \omega(\epsilon, \dot{\epsilon}) = W\dot{\epsilon} = \begin{bmatrix} \dot{\phi} \cos \theta - \dot{\psi} \cos \phi \sin \theta \\ \dot{\theta} + \dot{\psi} \sin \phi \\ \dot{\phi} \sin \theta + \dot{\psi} \cos \phi \cos \theta \end{bmatrix} \quad (4-70)$$

式中  $W$  是欧拉角速率到世界坐标系下机体角速度  $\omega$  的变换矩阵，其表达形式为：

$$W = \begin{bmatrix} \cos \theta & 0 & -\cos \phi \sin \theta \\ 0 & 1 & \sin \phi \\ \sin \theta & 0 & \cos \phi \cos \theta \end{bmatrix} \quad (4-71)$$

$R$  与  $\omega$  关于  $c$  和  $T$  的梯度具有解析的表达式，详见附录B。

### 4.3.3 基于四元数姿态表示的过驱动飞行器 $SE(3)$ 轨迹生成

四元数是刚体姿态表示的一种常见形式。与欧拉角表示法的主要区别在于，四元数表示法不存在奇异性，同一个姿态有且仅有两个单位四元数与之对应，且这两个单位四元数互为加法逆元。

Terzakis 等人提出过利用球极投影参数化四元数的方式。以单位四元数超球面的一个极点  $\mathbf{Q}_0 = [1 \ 0 \ 0 \ 0]^T$  作球极投影，就在除去该极点的超球面与整个  $\mathbb{R}^3$  空间建立了一个连续光滑的双射，该映射的逆映射的表达式为：

$$\mathbf{Q} = \mathbf{Q}(\varrho) = \begin{bmatrix} \frac{\varrho^T \varrho - 1}{\varrho^T \varrho + 1} & \frac{2\varrho^T}{\varrho^T \varrho + 1} \end{bmatrix}^T \in SO(3), \forall \varrho \in \mathbb{R}^3 \quad (4-72)$$

显然用  $\varrho$  作为姿态的参数化表示是满足二阶连续可微的条件的，因此平坦输出轨迹也可以表示为：

$$\boldsymbol{\sigma}(t) = \varrho(t) \quad (4-73)$$

旋转矩阵和角速度用单位四元数  $\mathbf{Q} = [w \ x \ y \ z]^T$  表示为：

$$\mathbf{R}(\mathbf{Q}) = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2x^2 - 2z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (4-74)$$

$$\boldsymbol{\omega} = 2\mathbf{U}\dot{\mathbf{Q}} = 2\mathbf{U}\mathbf{G}^T\dot{\varrho} \quad (4-75)$$

其中：

$$\mathbf{U} = \begin{bmatrix} -x & w & -z & y \\ -y & z & w & -x \\ -z & -y & x & w \end{bmatrix} \in \mathbb{R}^{3 \times 4} \quad (4-76)$$

$$\mathbf{G} = \begin{bmatrix} \frac{\partial w}{\partial \varrho} & \frac{\partial x}{\partial \varrho} & \frac{\partial y}{\partial \varrho} & \frac{\partial z}{\partial \varrho} \end{bmatrix} \in \mathbb{R}^{3 \times 4} \quad (4-77)$$

$\mathbf{G}$  以及罚函数梯度的具体形式详见附录C。

### 4.3.4 算法实现

如式 (4-50) 所示，本优化器不固定轨迹总时间，且时间正则项与总时间成正比，也就是说优化时会倾向于使轨迹更快地到达终点；此外，本课题在实现优化器时使用的时间约束消去微分同胚变换并非指数函数，而是参考 GCOPTER 官方实现<sup>①</sup> 使用下述形式的微分同胚映射：

<sup>①</sup> <https://github.com/ZJU-FAST-Lab/GCOPTER>

$$T_i = \begin{cases} 1 + \tau_i(0.5\tau_i + 1), & \text{if } \tau_i > 0 \\ 1/(1 + \tau_i(0.5\tau_i - 1)), & \text{if } \tau_i \leq 0 \end{cases} \quad (4-78)$$

为了提高优化器的复用性、充分发挥几何约束优化器灵活性的优点，对于自定义约束  $\mathcal{G}$ ，在实际实现时将时间积分罚函数抽象为了 SE3TimeIntPenalty 类，提供 evaluate() 公共接口，此接口返回罚函数值及其关于  $\mathbf{c}$  和  $\mathbf{T}$  的梯度，当面对不同的任务需求时，只需继承 SE3TimeIntPenalty 类，用需求对应的罚函数重写 evaluate() 接口，并与优化器关联即可。

在实现中采用 L-BFGS 算法<sup>[46]</sup> 作为无约束优化算法，Wang 有开源的算法实现<sup>①</sup>，但其中仅使用 Lewis 等人提出的线搜索方法<sup>[47]</sup>，该方法若在一定迭代次数后不能找到满足条件的点则视为线搜索失败，随之整个优化过程也宣告失败；受限于计算机对极小数字的分辨能力，在本课题这种目标函数数值特性不太好的情况下就很难优化成功。于是本课题在重新实现一遍 L-BFGS 算法的同时，使用的线搜索策略变为：若某次 Lewis 线搜索失败，则使用 Backtracking 法，这样可确保优化过程不会因为线搜索失败而中断。

本课题中上述优化器使用 C++17 标准实现。

#### 4.4 本章小结

本章首先详细介绍了几何约束下多旋翼飞行器轨迹优化框架的基本原理，给出了其基本问题形式、重要技巧等；随后根据过驱动飞行器的特点，结合几何约束优化的核心思想，确定了适用于过驱动飞行器 6 自由度  $SE(3)$  轨迹优化的问题形式，并给出了相关公式的推导，设计出了几何约束下过驱动飞行器轨迹优化的算法框架；最后给出了基于欧拉角和基于四元数的两种适用于本框架的姿态参数化方式。

本章完成了过驱动飞行器轨迹规划后端部分的算法设计。至此，一个完整的过驱动飞行器轨迹规划框架设计完毕。

<sup>①</sup> <https://github.com/ZJU-FAST-Lab/LBFGS-Lite>

## 第 5 章 规划器测试与实验

### 5.1 引言

### 5.2 规划器效果测试

本节通过在各种给定的环境中运行本课题开发的过驱动飞行器轨迹器来测试其效果，并将结果使用 Rviz<sup>[48]</sup> 等可视化工具进行展示。

测试过程中环境地图作为已知量，主要使用点云和八叉树两种表达形式，规划方式为全局规划。重点关注输出轨迹的避障特性以及规划过程的计算效率。未经特殊说明，本节实验的规划器参数设置如表5-1 计算过程中物理量均取其在国际单位制下的数值大小。

表 5-1 实验中规划器的参数设置

参数	取值
长方体尺寸	(1.0m, 1.0m, 0.35m)
RRT 步长	0.5
RRT 采样概率	0.9
初始轨迹中间点间距	3.0m
$k_p$	100.0
$(v_{max}, W_v)$	$(0.8m \cdot s^{-1}, 1 \times 10^4)$
$(a_{max}, W_a)$	$(5.0m \cdot s^{-2}, 1 \times 10^4)$
$(\omega_{max}, W_\omega)$	$(0.8rad \cdot s^{-1}, 1 \times 10^4)$
$W_c$	$9 \times 10^4$

#### 5.2.1 随机地图中的轨迹规划

本小节在使用现成随机地图生成程序生成的随机环境（如图5-1 (a)所示）中进行规划。该程序可以根据人为指定的地图范围、障碍物数量即分辨率将障碍物随机布撒到环境中，并以点云的形式输出。障碍物分为柱形（图5-1 (b)）和圆圈形（图5-1 (c)），二者的尺寸范围也可以人为设置。

图5-2中分别展示了使用基于欧拉角和基于四元数两种姿态表示法在随机地图中规划出的轨迹，轨迹多项式的次数为 7（即  $s = 4$ ），地图尺寸为  $50m \times 50m$ ，其中随机分布有 300 个柱形障碍物和 30 个圆圈形障碍物，轨迹起始条件和终止条件下速度、加速度、加加速度以及姿态对应的 RPY 角均为 0。

图5-2中一系列半透明浅蓝色凸多面体为生成的飞行走廊，而深蓝色带状物则

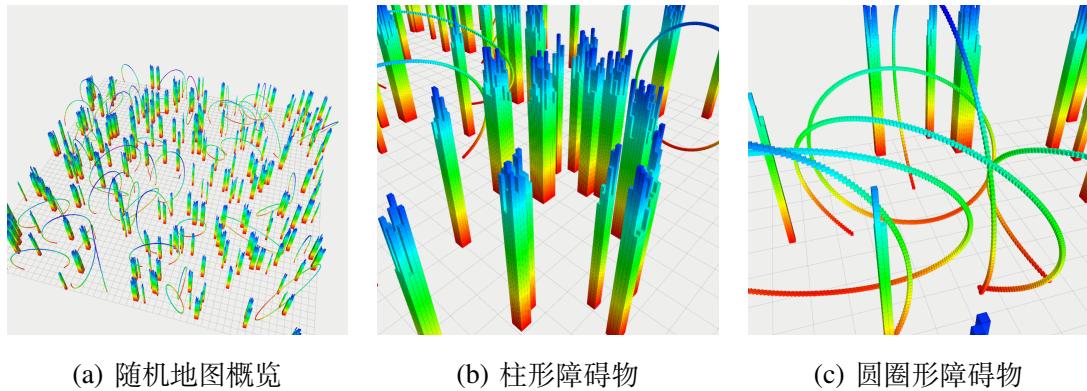


图 5-1 随机地图示意图

是由近似表示飞行器形状的长方体组成轨迹可视化的 6 自由度轨迹。可以看到，生成的轨迹均被成功约束在了安全飞行走廊内，且具有足够的平滑性。

如图5-3所示分别画出了上述两条轨迹的动力学特性，图5-3 (a)和图5-3 (c)所展示的是整条轨迹中速率、加速率和角速率的变化，红色虚线表示的是动力学限制  $v_{max}$ 、 $a_{max}$  和  $\omega_{max}$ ，可见轨迹的动力学特性被有效地约束住了，且在时间正则项的作用下，轨迹在大部分时间内都达到了所限制的最大速率。图5-3 (b)和图5-3 (d)所展示的则是速度、加速度和角速度向量的轨迹，图中淡黄色的球所表示的则是  $v_{max}$ 、 $a_{max}$  和  $\omega_{max}$ 。

上述两种规划结果的前端相关数据如表5-2所示。表中数据与图5-2表现出了两种不同规划方式各自的典型特征，

(1) 基于欧拉角的方法轨迹优化速度通常比基于四元数的方法更快；

(2) 相较基于欧拉角的方法而言，基于四元数的方法对狭小的空间和障碍物更为“敏感”：基于四元数的方法规划出的轨迹在穿过狭小的空间或者经过靠近障碍物的地方时更倾向于倾斜姿态来避免碰撞；而基于欧拉角的方法显得更为“懒惰”，常常靠近障碍物处也不会利用姿态控制来规避障碍物，所以轨迹上经常会有刮蹭发生。

表 5-2 随机地图规划两种方式前端相关数据

姿态规划方式	RRT 耗时 (秒)	生成 SFC 耗时	SFC 凸多面体数	轨迹生成耗时 (秒)
四元数	0.206	0.951	14	10.799
欧拉角	0.431	0.898	20	4.609

首先分析优化效率的不同，考虑到两种规划方式的差异最终都体现到了罚函数  $I_\Sigma$  值与梯度的计算上，表5-3进一步给出了相关数据作对比。其中  $t_{col}$ 、 $t_{dyn}$  及  $t_{pen}$  分别表示单次计算碰撞惩罚项、动力学约束惩罚项以及整个罚函数的耗时。可见

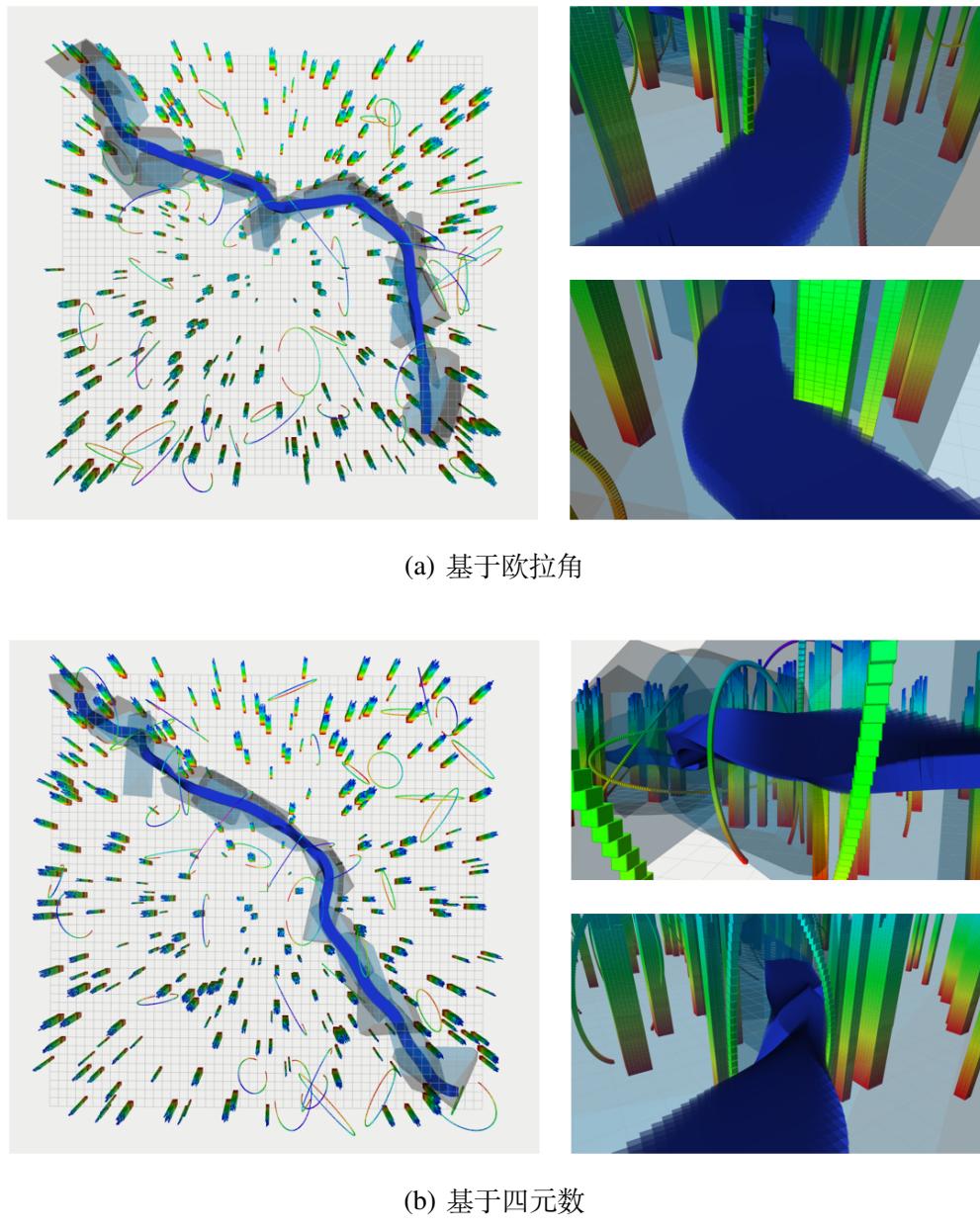


图 5-2 在随机地图中进行规划得到的轨迹及其局部细节

在单次计算罚函数的耗时以及拟牛顿法迭代次数上四元数法均比欧拉角法更有优势，但是最后的总优化时间却比欧拉角法要慢得多，这说明在线搜索阶段四元数法的迭代次数远多于欧拉角法的，于是可以推断四元数法目标函数的数值特性不利于线搜索。至于欧拉角法以更快的速度优化出的轨迹对障碍物敏感程度不够的原因，推测是由于欧拉角法的目标函数存在较多的局部最小值。

为更细致地研究两种不同的姿态规划方法在安全性和快速性，本课题进一步设计了定量实验。

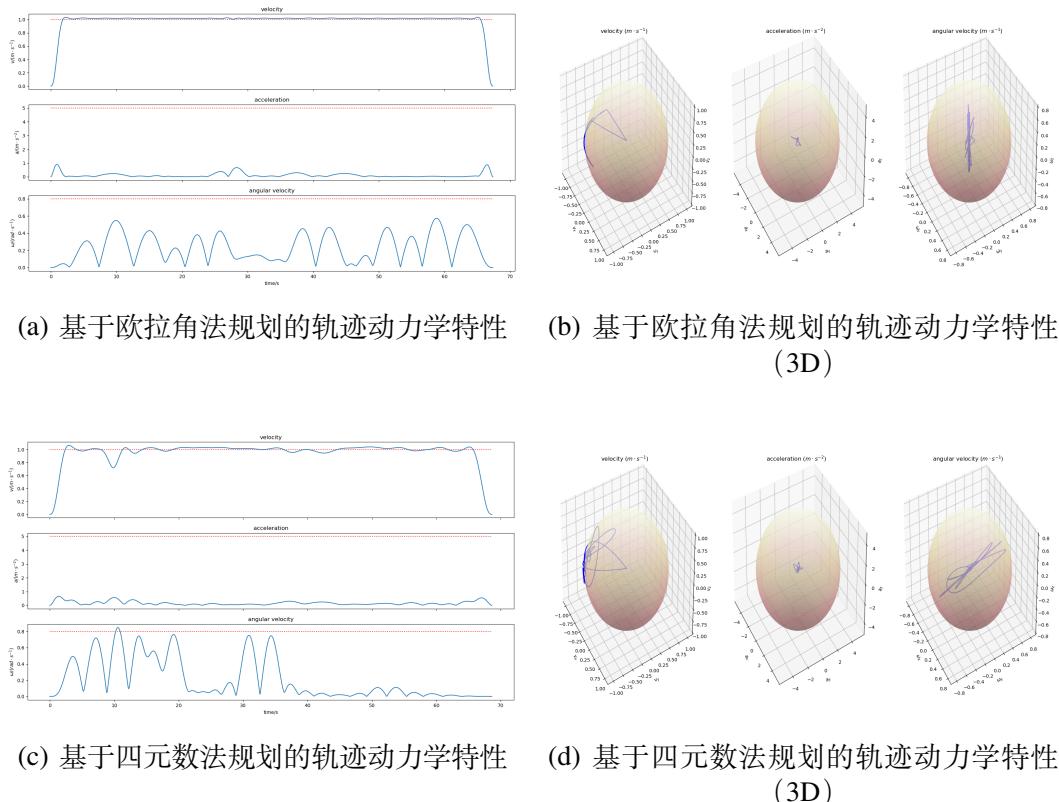


图 5-3 所得轨迹的动力学特性示意图

表 5-3 随机地图规划两种方式的效率分析

姿态规划方式	$t_{\text{col}}$ (毫秒)	$t_{\text{dyn}}$ (毫秒)	$t_{\text{pen}}$ (毫秒)	L-BFGS 迭代次数	轨迹生成耗时 (秒)
四元数	1.564	0.307	1.877	30	10.799
欧拉角	1.891	0.261	2.159	1788	4.609

为衡量安全性，首先给出成功轨迹的定义：

多次规划，记录两种姿态表示法的成功率

### 5.2.2 狹窄空间中的轨迹规划

测量两种姿态表示法下计算一次罚函数所花的时间

表 5-4 两种姿态规划方式在穿越狭窄缝隙的场景下的一些数据对比

姿态规划方式	$t_{\text{col}}$ (毫秒)	$t_{\text{dyn}}$ (毫秒)	$t_{\text{pen}}$ (毫秒)	L-BFGS 迭代次数	轨迹生成耗时 (秒)
四元数	0.182	0.055	0.242	90	0.838
欧拉角	0.230	0.052	0.286	2244	0.842

### 5.3 仿真避障测试

### 5.4 实物避障测试

### 5.5 本章小结

## 结 论

学位论文的结论作为论文正文的最后一章单独排写，但不加章标题序号。

结论应是作者在学位论文研究过程中所取得的创新性成果的概要总结，不能与摘要混为一谈。博士学位论文结论应包括论文的主要结果、创新点、展望三部分，在结论中应概括论文的核心观点，明确、客观地指出本研究内容的创新性成果（含新见解、新观点、方法创新、技术创新、理论创新），并指出今后进一步在本研究方向进行研究工作的展望与设想。对所取得的创新性成果应注意从定性和定量两方面给出科学、准确的评价，分（1）、（2）、（3）…条列出，宜用“提出了”、“建立了”等词叙述。

## 参考文献

- [1] TEDRAKE R. Underactuated Robotics [M/OL] . 2022. <http://underactuated.mit.edu>.
- [2] FLIESS M, LÉVINE J, MARTIN P, et al. Flatness and defect of non-linear systems: introductory theory and examples [J] . International Journal of Control, 2003, 61 (6) : 1327-1361.
- [3] MELLINGER D, KUMAR V. Minimum snap trajectory generation and control for quadrotors [C/OL] // 2011 IEEE International Conference on Robotics and Automation, [S.l.] , 2011 : 2520-2525. <http://dx.doi.org/10.1109/ICRA.2011.5980409>.
- [4] BRY A, RICHTER C, BACHRACH A, et al. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments [J] . The International Journal of Robotics Research, 2015, 34 (7) : 969-1002.
- [5] GAO F, WANG L, ZHOU B, et al. Teach-Repeat-Replan: A Complete and Robust System for Aggressive Flight in Complex Environments [J/OL] . IEEE Transactions on Robotics, 2020, 36 (5) : 1526-1545. <http://dx.doi.org/10.1109/TRO.2020.2993215>.
- [6] HAN Z, WANG Z, XU C, et al. Fast-Racing: An Open-source Strong Baseline for SE(3) Planning in Autonomous Drone Racing [J] , 2021.
- [7] LIU S, MOHTA K, ATANASOV N, et al. Search-based motion planning for aggressive flight in se (3) [J] . IEEE Robotics and Automation Letters, 2018, 3 (3) : 2439-2446.
- [8] WANG Z, ZHOU X, XU C, et al. Geometrically constrained trajectory optimization for multicopters [J] . IEEE Transactions on Robotics, 2022.
- [9] YANG S, HE B, WANG Z, et al. Whole-Body Real-Time Motion Planning for Multicopters [C] //2021 IEEE International Conference on Robotics and Automation (ICRA), [S.l.] , 2021 : 9197-9203.
- [10] BRESCIANINI D, D'ANDREA R. Design, modeling and control of an omnidirectional aerial vehicle [C] //2016 IEEE international conference on robotics and automation (ICRA), [S.l.] , 2016 : 3261-3266.

- [11] PARK S, LEE J, AHN J, et al. Odar: Aerial manipulation platform enabling omnidirectional wrench generation [J]. IEEE/ASME Transactions on mechatronics, 2018, 23 (4) : 1907-1918.
- [12] ALLENSPACH M, BODIE K, BRUNNER M, et al. Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight [J]. The International Journal of Robotics Research, 2020, 39 (10-11) : 1305-1325.
- [13] RYLL M, BÜLTHOFF H H, GIORDANO P R. A novel overactuated quadrotor unmanned aerial vehicle: Modeling, control, and experimental validation [J]. IEEE Transactions on Control Systems Technology, 2014, 23 (2) : 540-556.
- [14] KAMEL M, VERLING S, ELKHATIB O, et al. The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle [J]. IEEE Robotics & Automation Magazine, 2018, 25 (4) : 34-44.
- [15] WANG Z, ZHOU X, XU C, et al. Geometrically Constrained Trajectory Optimization for Multicopters [J], 2021.
- [16] BRESCIANINI D, D' ANDREA R. Computationally efficient trajectory generation for fully actuated multirotor vehicles [J]. IEEE Transactions on Robotics, 2018, 34 (3) : 555-571.
- [17] MORBIDI F, BICEGO D, RYLL M, et al. Energy-efficient trajectory generation for a hexarotor with dual-tilting propellers [C] // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), [S.I.], 2018 : 6226-6232.
- [18] PANTIC M, OTT L, CADENA C, et al. Mesh manifold based riemannian motion planning for omnidirectional micro aerial vehicles [J]. IEEE Robotics and Automation Letters, 2021, 6 (3) : 4790-4797.
- [19] QUIGLEY M, CONLEY K, GERKEY B, et al. ROS: an open-source Robot Operating System [C] // ICRA Workshop on Open Source Software : Vol 3, [S.I.], 2009 : 5.
- [20] KOENIG N, HOWARD A. Design and use paradigms for gazebo, an open-source multi-robot simulator [C] // 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) : Vol 3, [S.I.], 2004 : 2149-2154.
- [21] MEIER L, HONEGGER D, POLLEFEYS M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms [C] // 2015 IEEE international conference on robotics and automation (ICRA), [S.I.], 2015 : 6235-6240.

- [22] TAYLOR R M, HUDSON T C, SEEGER A , et al. VRPN: a device-independent, network-transparent VR peripheral system [C] //Proceedings of the ACM symposium on Virtual reality software and technology , [S.l.] , 2001 : 55-61.
- [23] BODIE K, TAYLOR Z, KAMEL M, et al. Towards efficient full pose omnidirectionality with overactuated mavs [C] //International Symposium on Experimental Robotics , [S.l.] , 2018 : 85-95.
- [24] FLIESS M, LÉVINE J, MARTIN P, et al. Flatness and defect of non-linear systems: introductory theory and examples [J] . International journal of control, 1995, 61 (6) : 1327-1361.
- [25] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths [J] . IEEE transactions on Systems Science and Cybernetics , 1968, 4 (2) : 100-107.
- [26] HARABOR D, GRASTIEN A. Online graph pruning for pathfinding on grid maps [C] //Proceedings of the AAAI Conference on Artificial Intelligence : Vol 25 , [S.l.] , 2011 : 1114-1119.
- [27] KAVRAKIL E, SVESTKA P, LATOMBE J-C , et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces [J] . IEEE transactions on Robotics and Automation , 1996, 12 (4) : 566-580.
- [28] LAVALLE S M, OTHERS. Rapidly-exploring random trees: A new tool for path planning [J] , 1998.
- [29] KAVRAKI L E, KOLOUNTZAKIS M N, LATOMBE J-C. Analysis of probabilistic roadmaps for path planning [J] . IEEE Transactions on Robotics and automation , 1998, 14 (1) : 166-171.
- [30] FRAZZOLI E, DAHLEH M A, FERON E. Real-time motion planning for agile autonomous vehicles [J] . Journal of guidance, control, and dynamics , 2002, 25 (1) : 116-129.
- [31] KARAMAN S, FRAZZOLI E. Sampling-based algorithms for optimal motion planning [J] . The international journal of robotics research , 2011, 30 (7) : 846-894.
- [32] GAMMELL J D, SRINIVASA S S, BARFOOT T D. Informed RRT\*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic [C] //2014 IEEE/RSJ International Conference on Intelligent Robots and Systems , [S.l.] , 2014 : 2997-3004.

- [33] ORTHEY A, TOUSSAINT M. Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications [J]. arXiv preprint arXiv:1906.01350, 2019.
- [34] KUFFNER J J. Effective sampling and distance metrics for 3D rigid body path planning [C] //IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 : Vol 4, [S.l.], 2004 : 3993-3998.
- [35] BENTLEY J L. Multidimensional binary search trees used for associative searching [J]. Communications of the ACM, 1975, 18 (9) : 509-517.
- [36] BRIN S. Near neighbor search in large metric spaces [J]. Proc. of 21st Conf. on very large database, Zurich, Switzerland, 1995, 1995.
- [37] SUCAN I A, MOLL M, KAVRAKI L E. The open motion planning library [J]. IEEE Robotics & Automation Magazine, 2012, 19 (4) : 72-82.
- [38] PAN J, CHITTA S, MANOCHA D. FCL: A general purpose library for collision and proximity queries [C] //2012 IEEE International Conference on Robotics and Automation, [S.l.], 2012 : 3859-3866.
- [39] HORNUNG A, WURM K M, BENNEWITZ M, et al. OctoMap: An efficient probabilistic 3D mapping framework based on octrees [J]. Autonomous robots, 2013, 34 (3) : 189-206.
- [40] RUSU R B, COUSINS S. 3d is here: Point cloud library (pcl) [C] //2011 IEEE international conference on robotics and automation, [S.l.], 2011 : 1-4.
- [41] MOHTA K, WATTERSON M, MULGAONKAR Y, et al. Fast, autonomous flight in GPS-denied and cluttered environments [J]. Journal of Field Robotics, 2018, 35 (1) : 101-120.
- [42] GAO F, WU W, GAO W, et al. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments [J]. Journal of Field Robotics, 2019, 36 (4) : 710-733.
- [43] LIU S, WATTERSON M, MOHTA K, et al. Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments [J/OL]. IEEE Robotics and Automation Letters, 2017, 2 (3) : 1688-1695. <http://dx.doi.org/10.1109/LRA.2017.2663526>.
- [44] DEITS R, TEDRAKE R. Computing large convex regions of obstacle-free space through semidefinite programming [G] //Algorithmic foundations of robotics XI. [S.l.] : Springer, 2015 : 109-124.

- [45] ZHOU X, WEN X, WANG Z, et al. Swarm of micro flying robots in the wild [J]. *Science Robotics*, 2022, 7 (66) : eabm5954.
- [46] LIU D C. On the limited memory method for large scale optimization [J]. *Mathematical Programming B*, 1989, 45 (3) : 503-528.
- [47] LEWIS A S, OVERTON M L. Nonsmooth optimization via quasi-Newton methods [J]. *Mathematical Programming*, 2013, 141 (1) : 135-163.
- [48] KAM H R, LEE S-H, PARK T, et al. Rviz: a toolkit for real domain data visualization [J]. *Telecommunication Systems*, 2015, 60 (2) : 337-345.

## 哈尔滨工业大学深圳校区本科生毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学深圳校区攻读学士学位期间，所提交的毕业设计（论文）《过驱动飞行器的轨迹规划》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：\_\_\_\_\_ 日期： 年 月 日

## 致 谢

衷心感谢导师 XXX 教授对本人的精心指导。他的言传身教将使我终生受益。

.....

感谢哈深 L<sup>A</sup>T<sub>E</sub>X 论文模板 HITSZTHESIS !

## 附录 A 罚函数中正求和项的梯度

$$\frac{\partial I_{v_{ij}}}{\partial \mathbf{c}_i^P} = 6W_v \frac{T_i}{\kappa} (\|\mathbf{v}\|_2^2 - v_{max}^2)^2 \boldsymbol{\beta}^{(1)} \left( \frac{j}{\kappa} T_i \right) \mathbf{v}^T \quad (A-1)$$

$$\frac{\partial I_{v_{ij}}}{\partial \mathbf{c}_i^\sigma} = 0 \quad (A-2)$$

$$\frac{\partial I_{v_{ij}}}{\partial T_i} = W_v \frac{(\|\mathbf{v}\|_2^2 - v_{max}^2)^2}{\kappa} \left[ (\|\mathbf{v}\|_2^2 - v_{max}^2) + \frac{6jT_i}{\kappa} \mathbf{v}^T \mathbf{a} \right] \quad (A-3)$$

$$\frac{\partial I_{a_{ij}}}{\partial \mathbf{c}_i^P} = 6W_a \frac{T_i}{\kappa} (\|\mathbf{a}\|_2^2 - a_{max}^2)^2 \boldsymbol{\beta}^{(2)} \left( \frac{j}{\kappa} T_i \right) \mathbf{a}^T \quad (A-4)$$

$$\frac{\partial I_{a_{ij}}}{\partial \mathbf{c}_i^\sigma} = 0 \quad (A-5)$$

$$\frac{\partial I_{a_{ij}}}{\partial T_i} = W_a \frac{(\|\mathbf{a}\|_2^2 - a_{max}^2)^2}{\kappa} \left[ (\|\mathbf{a}\|_2^2 - a_{max}^2) + \frac{6jT_i}{\kappa} \mathbf{a}^T \mathbf{j} \right] \quad (A-6)$$

$$\frac{\partial I_{a_{ij}}}{\partial \mathbf{c}_i^P} = \mathbf{0} \quad (A-7)$$

$$\frac{\partial I_{\omega_{ij}}}{\partial \mathbf{c}_i^\sigma} = 6W_\omega \frac{T_i}{\kappa} (\|\boldsymbol{\omega}\|_2^2 - \omega_{max}^2)^2 (\omega_x \frac{\partial \omega_x}{\partial \mathbf{c}_i^\sigma} + \omega_y \frac{\partial \omega_y}{\partial \mathbf{c}_i^\sigma} + \omega_z \frac{\partial \omega_z}{\partial \mathbf{c}_i^\sigma}) \quad (A-8)$$

$$\frac{\partial I_{\omega_{ij}}}{\partial T_i} = W_\omega \frac{(\|\boldsymbol{\omega}\|_2^2 - \omega_{max}^2)^2}{\kappa} \left[ (\|\boldsymbol{\omega}\|_2^2 - \omega_{max}^2) + \frac{6T_i}{\kappa} \boldsymbol{\omega}^T \frac{\partial \boldsymbol{\omega}}{\partial T_i} \right] \quad (A-9)$$

$$\frac{\partial I_{c_{ijvk}}}{\partial \mathbf{c}_i^P} = \frac{3W_c T_i}{\kappa} (\mathbf{a}_{i,k}^T \mathbf{q}_v - b_{i,k})^2 \boldsymbol{\beta} \left( \frac{j}{\kappa} T_i \right) \mathbf{a}_{i,k}^T \quad (A-10)$$

$$\left[ \frac{\partial I_{c_{ijvk}}}{\partial \mathbf{c}_i^\sigma} \right]_{mn} = \frac{3W_c T_i}{\kappa} (\mathbf{a}_{i,k}^T \mathbf{q}_v - b_{i,k})^2 \text{tr} \left\{ \hat{\mathbf{q}}_v \mathbf{a}_{i,k}^T \frac{\partial \mathbf{R}}{\partial [\mathbf{c}_i^\sigma]_{mn}} \right\} \quad (A-11)$$

$$\frac{\partial I_{c_{ijvk}}}{\partial T_i} = W_c \frac{(\mathbf{a}_{i,k}^T \mathbf{q}_v - b_{i,k})^2}{\kappa} \left[ (\mathbf{a}_{i,k}^T \mathbf{q}_v - b_{i,k}) + 3T_i \left( \frac{j}{\kappa} \mathbf{a}_{i,k}^T \mathbf{c}_i^{P,T} \boldsymbol{\beta}^{(1)} \left( \frac{j}{\kappa} T_i \right) + \text{tr} \left\{ \hat{\mathbf{q}}_v \mathbf{a}_{i,k}^T \frac{\partial \mathbf{R}}{\partial T_i} \right\} \right) \right] \quad (A-12)$$

式中  $\mathbf{j}$  为加速度。

## 附录 B 基于欧拉角姿态表示的罚函数梯度

## 附录 C 基于四元数姿态表示的罚函数梯度