编号_____

南京航空航天大学

毕业设计

题　目　　基于四旋翼无人机的轨迹生成研究

学生姓名	陆鸿
学　　号	161540110
学　　院	计算机科学与技术学院
专　　业	物联网工程
班　　级	1615401
指导教师	翟象平　副教授

二〇一九年六月

南京航空航天大学

本科毕业设计（论文）诚信承诺书

本人郑重声明：所呈交的毕业设计（论文）（题目：基于四旋翼无人机的轨迹生成研究）是本人在导师的指导下独立进行研究所取得的成果。尽本人所知，除了毕业设计（论文）中特别加以标注引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写的成果作品。

作者签名：　　年　月　日

（学号）：

基于四旋翼无人机的轨迹生成研究

摘　要

随着机器人技术的不断进步，运动规划问题作为其核心问题在最近几年不断地被深入研究，而作为运动规划中的一部分，路径规划与轨迹生成是十分重要的。机器人的一个基本问题就是在保证目标安全、不与障碍物碰撞的前提下，如何使得对象从环境的初始域前往目标域，当然，对于规划算法的时间要求也相继提出。本文主要提出了基于随机采样的路径规划与轨迹生成方法充分地对于可行域进行采样。由于该种方法不需要对于整体环境进行精确的数学建模，所以在保证机器人安全的前提下，能够快速地解决规划问题。本文同时提出了全新的路径柔顺算法与同时更新双向搜索树的方法，提高了规划路径的可行性与规划效率。本文选择了控制灵活的四旋翼无人机进行实验，四旋翼无人机利用四个螺旋桨的互相配合能够达到六自由度的快速响应与运动，基于四旋翼无人机去检验生成轨迹无疑是一个很好的选择。

关键词：四旋翼无人机，基于采样，快速搜索随机树，双向快速搜索随机树

Research on Trajectory Generation for Quadrotor Unmanned Aerial Vehicle

Abstract

Robotics technology have been improved and advanced in recent years. Meanwhile, its applications are emerging and spreading worldwide, from data collection, surveillance, rescue to military operations and deliveries. Being an important part of robotics issues, motion planning has been studied through last decades. In motion planning, path planning and trajectory generation is of great importance since it solves the basic problem in robotics: moving robot from initial configuration to goal configuration without obstacle collisions. For effectiveness and shorter planning time, this paper implemented sampling-based algorithm to solve planning problems rather than explicit modelling in various environments. With a simple but controllable construction of the frame, quadrotor UAV is selected for experiment for its agility and 6-DOF provided by its four propellers. This paper also proposed path-smoothing algorithm for quadrotor UAVs to follow the path more safety and efficiently. Also, a brand-new algorithm, simultaneously-update bidirectional RRT, is introduced to improve trackability and efficiency of generated trajectory.

Key Words：Robotics, sampling based, RRT, bidirectional RRT

目 录

第一章 引 言

1.1 课题研究背景

人工智能（Artificial Intelligence）技术[1]在自动化与机器人领域逐渐扮演着重要的角色，这一切都是算法与云服务结合产生的巨大效益。现代的移动与合作服务机器人都适当地内置了一定的人工智能技术。机器人都在自动地进行决策，一个简单的例子就是一个机器人能在工作站环境下将不同的物品移动到所需要的地方[2]。这些机器人在有人的工作环境中进行移动，那么一定需要将其所处的环境进行建模，这样才能进行路径的构建与适应动态环境，如此的应用是工业界目前最为常见的形式。

算法技术能让机器在环境中运行的更加得有效，我们相信 AI 技术正在从一开始的“打扰”工业界向颠覆传统工业并创造巨大产值的方向发展。随着机器人技术的发展，它将改变工厂与生活的运作方式，使得社会前进。将 AI 技术嵌入到机器人中去是极具工业价值的，同时，这也具有社会与商业价值，目前浮现的自动驾驶与无人机技术已经体现出其强大的颠覆能力，如果法律法规成熟之后将极大的改变人类的生活方式。

在 Breguet 和 Richet[3]第一次尝试四旋翼无人机飞行之后，经过多年的发展，无人机作为空中机器人成为了领域中飞快发展的一类。全世界范围内每年投入到无人机科技的开发资金大约为 64 亿美元，并且随着无人机的发展，其商业与军事应用的花费将会预计上升至 115 亿美元。据统计，在 2019 年末，商用小型无人机的盈利金额预计将达到 51 亿美元，同时，在 2020 年末小型无人机的保有量将会达到 700 万架。将相关的技术运用到无人机上是十分必要的，本论文也正是基于四旋翼无人机进行各种环境下的路径规划与轨迹生成任务。

1.1.1 国内发展趋势

连续四年，中国已经成为世界最大的工业机器人市场[36]。在 2016 年，中国已有总计 90,000 件机器人的销售量，该数字较 2015 年相比已经上浮很多并且代表了全世界 30%的市值。世界机器人协会的报告指出，自 2016 年起，中国的在运行的工业机器人的数量是

最多的，到 2020 年，中国将预计生产出 150,000 件工业机器人并且有 950,300 个工业机器人在一线进行生产活动。

虽然我国的工业机器人数量很多，但是其分布密度却低于世界水平，但是这与中国的国情有关，工业区分布范围广泛并且中国劳动人口基数巨大。数据显示，中国的每 10,000 劳动力享有的机器人数量为 68 件。其中大多数的机器人需求量是由国际企业产生的，而国内的生产则大量依靠国外技术。自从习近平主席提出了“中国制造 2025”之后，机器人行业无疑成为了大力发展的领域之一，对于计算机行业来讲是一个十分好的发展方向，但同时也面临了很大的挑战。

在四旋翼无人机方面，中国的企业表现的十分出色，无人机的龙头企业大疆创新科技有限公司推出了很多全球性产品，引领了四旋翼无人机的潮流。从航拍机切口进入，大疆迅速在 2014 年占领了全球的四旋翼无人机份额，其中 80%的产品销往海外，其主流消费级机型“大疆精灵”更是家喻户晓。为了与行业需求相结合，大疆也推出了很多的行业应用无人机，致力于为政府、公共事业机构以及企业客户提供以无人机飞行平台为基础的行业解决方案与定制化服务。在这些业务的完成过程中，支撑技术是必不可少的，大疆也相应的开发出了相应的应用框架与配套的硬件，使得其在全球四旋翼无人机开发领域占据了十分重要的地位。

1.1.2 国外发展趋势

国外的工业机器人已经达到了较高的程度，除了中国之外，韩国、日本、美国以及德国是全世界范围内机器人发展与销售量最大的几个国家[36]。日本截止 2019 年为止，销量上升了 18%，总量为 45,566 件，该数量为日本历史上最高的交易数量。虽然日本国内的自动化水平与前几年基本持平，但是电子行业依然是日本机器人行业发展的主要动力之一，日本基本保持了较高的工业自动化水平。韩国在机器人供应上近几年呈现下降的趋势，截止 2019 年为 39,732 件，而此前韩国曾达到 41,373 件的交易峰值，随着韩国电子行业的发展，机器人交易量在之后依然会呈上升趋势。机器人装配业在美国也达到了几年来的顶峰，约为 33,192 件，而美国的生产工业的增长驱动是美国国内对于增长美国工业自动化的政策所驱使的。德国作为世界上第五体量的机器人生产与装配的国家，其趋势一直处于上升状态，截至 2019 年，其销量已经达到了 21,404 件并且其国内一直保持了比较高的工业自动化水平。

在四旋翼无人机方面，外国的企业表现得也很出色，3D Robotics 作为北美最大的面向个人用户的无人机厂商，同样也是大疆在北美最大的竞争对手，3D Robotics 在 2019 年 4 月发布的 Solo 无人机，号称大疆“精灵”杀手。另一个消费级的无人机公司是法国的 Parrot 公司，Parrot 曾经以无线、语音、蓝牙和车载环绕音响等产品闻名，但其无人机在欧洲十分的火热，其 Rolling Spider 产品是一款可拍照、会爬墙的四轴飞行器，其期间无人机 Bebop 售价仅 500 美元，是大疆和 3D Robotics 同类产品售价的一半。总之，大疆、3D Robotics 和 Parrot 构成了世界消费级四旋翼无人机三足鼎立的局面，而这些公司在无人机技术方面的投资都是巨大的。无人机的避障技术、导航技术、跟踪技术、飞行性能等关键指标都在产业的进步中不断地更新与发展。

1.2 本文研究方向

机器人（Robotics）是一个交叉学科的领域，关于机器人的一般问题可以包含很多的方面，关乎材料、能源、导航与控制、人工智能算法、机器人伦理与法律等各个方面。本文主要研究方向与重点是机器人在运动规划方向的问题。

关于机器人的运动规划与移动是计算机科学家研究最多的领域，机器人的运动规划可以分为很多的层次，具体如图 1.1 所示。大致框架上一定具备这四个层次，感知层、定位层、规划层和控制层，每一层次都有不同的任务来进行。当然，现有技术还进行了在有人/无人环境下的框架设计，其中定位与规划层可以按照需求互相转换的。

本文所要解决的问题，具体就是四旋翼无人机在运动中的规划层的内容，在现有的技术中，相关的研究十分的丰富，有以根据是否利用全局地图信息进行分类的在线或离线算法，有以根据是否对环境信息建模的经典与启发式算法。各种规划算法的性质与效果都不一样并且具有不同的优缺点。在具体的应用场景中，需要建立起对于环境信息的采集、计算处理和运动实施的框架才能使得机器人成功地完成相应的任务。下面将先介绍规划层在整体框架的上下文内容，从而更好的理解路径规划与轨迹生成的功能与意义。

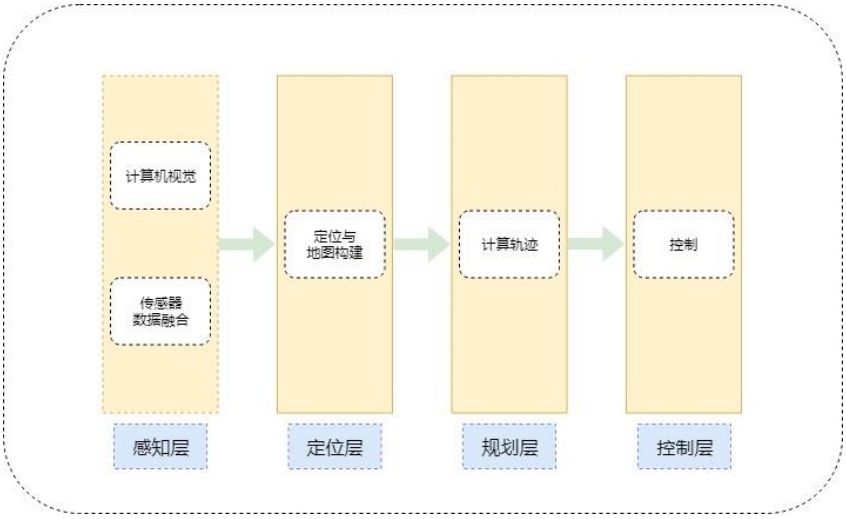


图 1.1 机器人运动规划框架

感知层包含了探测（Sensing）与感知（Perception）两部分，其中探测层通过各种传感器获得数据，例如雷达、激光扫描仪、摄像头、光流计和 GPS 等等。在感知部分进行对于探测数据的推理与表示，实现对环境的理解，包含了对障碍物的检测，道路标志的检测和前后交通物信息等等。同时，结合云计算技术，可以使用已有世界模型（World Model）的信息进行数据的进一步处理得到位姿，交通流量等信息。

定位层也是机器人自主导航的一个重要部分，机器人需要定位、地图建立和规划路径来进行合理的运动。基于上一层的感知层，定位层有很多的分类，比如单一使用摄像头信息的视觉定位与地图构建（Visual SLAM）[2]等。机器人的定位与地图建立是相互依赖的：机器人为了在环境中准确的定位需要精确的全局或者局部地图，但是想要构建正确的地图又需要准确的定位信息，所以这层中的问题被称作同时定位与建图（SLAM）问题。那么该层中采取的方法完全取决于上一层的数据传递，结合越多的感知信息，机器人的定位与建图肯定会更加准确。关于 SLAM 相关的研究也很丰富，Jorge Fuentes-Pacheco 等人[1]对于视觉 SLAM 技术进行了整体的调研，W. Gouda 等人[2]对人形机器人的 SLAM 技术进行了总结。总之同时定位与建图已经可以成功的运作在大环境中，当然其也有很多可以改进的地方。

规划层总体也包含了许多处理的步骤，可以包含任务规划和路径规划，其中由于导航或者规划的要求，任务点规划一般需要十几秒，路径规划一般需要 0.1 至 10 秒的时间，导航与规划层的具体框架可以如图 1.2 所示。在路径规划完成之后所需要的就是将路径轨

迹信息转换为控制指令交付控制层去控制对象进行运动。

图 1.2 规划层基本流程

控制层则是负责控制对象的动态系统的稳定并且按照上一层的信息来运行相关的指令使得控制对象按照相应的要求运行并且保证系统的稳定性与安全性。这个方向的工作更多需要控制科学与理论来支撑，需要很多内置在对象内部的传感器与驱动器实现，与上层感知层的对外传感器有所不同。

本论文主要的研究方向为上述规划层中的内容，目的就是利用已有的环境信息，不管是全局地图信息还是局部地图信息，使得机器人能够有效地在该环境中建立模型规划出可行路径。规划层在整体框架中属于比较重要的一部分，其路径质量关系着任务能否成功执行。

1.3 研究的主要内容与结构安排

1.3.1 主要内容

本文研究内容一是对于路径规划与轨迹生成算法进行了分类与分析；二是实现了基于采样的规划算法快速搜索树算法，使得四旋翼无人机在不同环境中都快速能找到安全的行驶轨迹；三是提出了转角的顺滑算法，使得轨迹更加柔顺并且符合四旋翼无人机的飞行条件；四是提出并实现了同时更新的双向快速搜索树算法，提升了路径规划与轨迹生成的速度。

1.3.2 结构安排

本文章节安排如下：

第一章为引言，主要介绍了课题研究背景与研究方向，并且对国内外的发展趋势做了一定的介绍。

第二章描述了四旋翼无人机的发展，并且对于四旋翼无人机的动力学进行了建模，同时也对于规划算法做了整理与分类，阐述了相关算法的计算过程，分析了其原理与计算效果，之后对于本文采取的基于采样的规划算法做了详细的讲解，分析了改类算法的一般流程与特性。

第三章和第四章主要是对于快速搜索随机树方法的研究与分析，整理了单向与双向搜索树的算法，同时提出了轨迹柔顺的算法，并且实现后运用到了项目的实际开发过程中。

第五章主要对于实验的设计与实现做了解释，对于实验结果进行了展示与分析。

第六章为总结与展望，主要总结研究中不足之处与希望以后能有所改进之处。

第二章　路径规划与轨迹生成

本章主要罗列了普遍存在的路径规划算法，并且对于路径规划算法进行了分类与简单的介绍，同时仔细讨论了基于采样的路径规划算法。其次，本文基于四旋翼无人机进行了轨迹规划相关的工作。由于四旋翼无人机具有悬停的特性，在四个螺旋桨的配合下，能够较好的产生六自由度的运动效果，所以相较于无人车，四旋翼无人机能够有较好的轨迹跟踪能力。

二十世纪时，人们就开始尝试四旋翼飞行器，Breguet 和 Richet[3]第一次真正提出了四旋翼飞行器概念，他们第一次将四旋翼无人机成功地飞到距离地面的几英尺高度。随着控制科学的发展，提出的标准控制器被运用到无人机的环境中[4][5]。近代随着 PD 与 PID 控制器的发展，这些技术也广泛地被运用。由于最近几年计算机科学与微芯片技术的发展，飞行平台也运用了神经网络来进行高度稳定、位置控制以及其他的控制需求。然而无人机现在也面临一些瓶颈问题，比如锂电池等现有的电池技术严重限制了无人机的飞行时长。当然，对于续航能力的提升也有相应的解决方案，比如减轻飞行平台的重量，将机身改为碳纤维材料配件；配置高性能比的直流无刷电机来产生更好的升力比；同时，A. B. Junaid[6]也提出了无人机无线充电的概念，吉林大学的 Z. N. Liu 等人[7]则提出了充电平台的概念来作为无人机任务规划中的辅助设施来助力无人机能够达到更长的续航效果来完成任务。

2.1 系统模型

2.1.1 四旋翼无人机建模

具体无人机的模型如图 2.1 所示，我们将$\mathbf{p} = [x, y, z]^\top$定义为无人机相对于惯性系$\{\varepsilon\}$的位置向量（position vector），无人机的欧拉角我们利用$\mathbf{\Phi} = [\phi, \theta, \psi]^\top$来表示，其中$\phi$表示关于 x 轴的翻转角，$\theta$表示关于 y 轴的俯仰角，$\psi$表示关于 z 轴的偏移角。四个电机分别单独的放置在四个机臂的顶端，以图为例，电机 1 与 3 是按照逆时针旋转的，相反的，电机 2 与 4 则是按照顺时针旋转，这样设计的目的是为了抵消内部惯量使得无人机能够

稳定。当四个电机的旋转角速度$w_i \geq 0$时，则能够相应地产生向上的升力$F_i, i \in \{1,2,3,4\}$。总体上一共有 12 个变量来描述无人机的状态。比如，位置向量$\mathbf{p} = [x, y, z]^\top$，线性速度$\mathbf{v} = [u, v, w]^\top$，欧拉角$\mathbf{\Phi} = [\phi, \theta, \psi]^\top$以及角速度$\dot{\mathbf{\Omega}} = [e_1, e_2, e_3]^\top$。

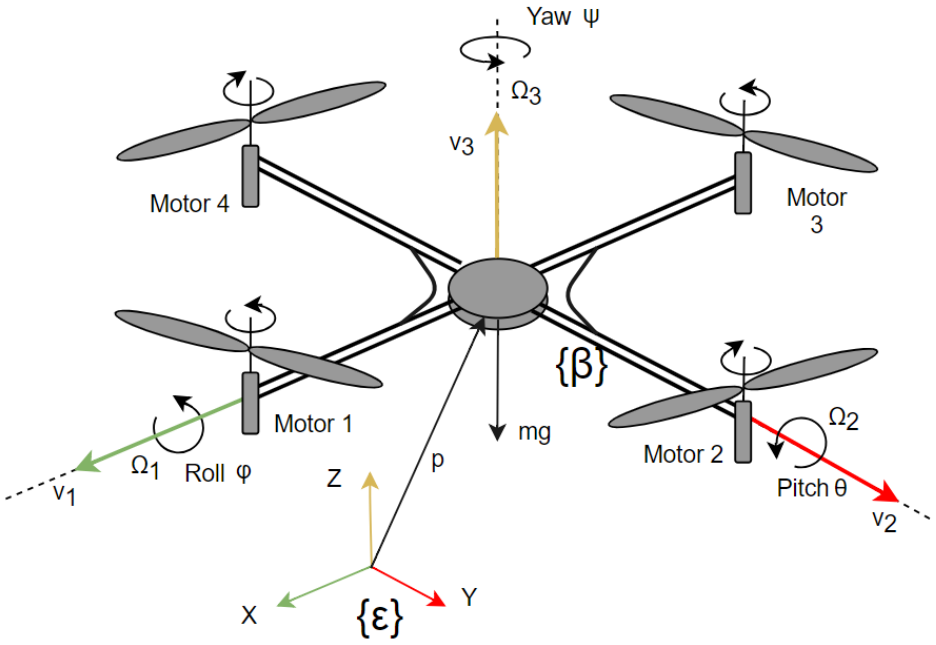


图 2.1 四旋翼无人机结构

线性力与角力矩需要被计算来衡量两个输入数值。我们有关于角速度生成上升力的公式如下：

$$F_i = k_b w_i^2, \quad i \in \{1,2,3,4\}, \tag{2.1}$$

其中k_b表示螺旋桨静态升力常数，表示为$k_b = C_T \rho A r^2$。根据相关文献[8][9]的表述，ρ表示空气密度，r和$A = \pi r^2$分别表示螺旋桨半径与其覆盖的面积。C_T表示螺旋桨的非维度协参数，这个参数依赖于螺旋桨的几何特性。

显然的，我们可以利用以上的控制输入来建立四旋翼无人机的动力学方程，以下动力学方程适用于如图 x 所示的四旋翼无人机：

$$\begin{cases} F = k_b(w_1^2 + w_2^2 + w_3^2 + w_4^2) \\ u_1 = k_b(w_2^2 - w_4^2) \\ u_2 = k_b(w_3^2 - w_1^2) \\ u_3 = k_r(w_1^2 - w_2^2 + w_3^2 - w_4^2) \\ u_4 = w_1 + w_3 - w_2 - w_4 \end{cases}, \tag{2.2}$$

其中，F表示由螺旋桨旋转生成的向上的升力，其方向垂直于 X-Y 平面，u_1和u_2表示了

成对的螺旋桨产生的升力。u_3表示了用来旋转的力。k_r表示了螺旋桨的空气动力阻力系数：$k_r = C_Q \rho A r^3$，其中，$C_Q = C_T \sqrt{C_T/2}$表示了螺旋桨的力矩协系数。

利用无人机的动力学方程，我们可以通过控制四旋翼无人机的输入来使其进行相应的任务，在本文中我们利用四旋翼无人机对轨迹进行相应的跟踪。随着四旋翼无人机的保有量越来越高，基于四旋翼无人机进行路径规划与路径生成是十分具有实用性与意义的。

2.1.2 路径规划问题建模

自动化机器人最想要的关键特征就是能在环境中自动导航的能力，许多机器人如今都用在了矿业与农业等场景中，这些场景往往需要机器人进行移动并且在多样的环境中执行一些任务，这些环境往往是不可预测的。路径规划与轨迹的生成，因此是机器人运动规划中的关键问题，路径规划与生成可行轨迹利用简洁的数学方法表示，将一个动态控制系统定义为：

$$dx/dt = f(x,u), x(0) = x_{init}, \tag{2.3}$$

其中，x是状态变量，u是控制变量；给定障碍物集合$X_{obs} \subset R^d$，给定目标域集合$X_{goal} \subset R^d$，机器人的运动规划目标就是找到控制序列，使得上式符合对于任何$t \in R_+$都有$x(t) \notin X_{obs}$，同时在时间$t < T$时有$X(t) \in X_{goal}$，则认为规划成功。对于有限时间T中没有找到路径，则认为规划失败。

随着软硬件质量与水平的提高，传感器与通信技术的发展，地面站与机载软件的质量都有了显著的提高，现有的无人机大多是遥控与跟踪预定的轨迹；但是高度自动化的无人机应该根据不同的环境制定相应的飞行轨迹，并且这些轨迹能够适应环境与机体自身动力学的约束，保证无人机能够顺利的按照轨迹飞行。

2.2 路径规划与轨迹生成算法

数学家与计算机学家等研究人员在经历了多年的实践与研究之后，诞生了很多轨迹生成的技术，并且迅速产生了很多的衍生算法；如何将物体在不触碰到障碍物的前提下，从初始状态移动到目标状态成为了机器人轨迹规划的最基本问题之一。从 Lozano-Pérez [10] 在 1979 年引入空间规划（spatial planning）这一方法开始，不同的运动规划问题就转变为

了在位形空间中寻找无碰撞路径的问题，并且使用了统一的数学方法进行了求解。之后，Jeffrey R. Hartline[11]证明了运动规划问题是 NP 完全问题；现有的成熟的方法大多是一些经典算法的衍生版本，如路径图法（Roadmap），栅格法（Cell Decomposition），势场法（Potential Field）以及精确的数学规划法(Mathematical Programming)。理论上来讲，这些经典算法不是一定互斥的，在开发路径规划器的时候常常将其中一些结合在一起。

很多路径规划算法在不同的环境中被提出与实践，这些方法包含了经典方法像势场法、bug 算法、路径图法和分解法等等，还有启发式算法如神经网络、模糊逻辑和进化算法等，这些方法在很多方面都各有优缺点。全局或者局部的路径规划器在指定的环境中利用这些算法去找到一条最优的路径供机器人去进行运动，其中前者提供了高层的低解决度（low-resolution high-level plan）的规划路径，后者能给出属于全局的部分高解决度的路径。当然，在相关文献[12][13]中提出了混合规划器的思想，使得路径规划在不同环境中更加有效，同时也在一定程度上优化了规划的路径，使得其能达到最优或者渐进最优的效果。

2.2.1 启发式算法

启发式的路径规划方法，虽然一直被用来和经典的路径规划算法进行比较，但是由于算法特征和人类的行为学习很像，还是得到了很足够的重视，启发式算法不能保证能求到解，但是如果他们能求解路径的话，速度将是经典算法无法比拟的。一些常见的启发式算法在下面简单的进行了介绍。

（1）逻辑模糊算法

逻辑模糊算法第一次由 Zadeh 在 1965 年提出，这个方法对人类的启发式经验与基于感知的行为进行了重现与应用，并且在机器人领域应用很广泛。逻辑模糊控制器（Fuzzy Logic controller）在处理真实世界不确定性方面十分的有效，并且不需要环境的绝对模型。Chelsea Sab[14]等人提出了应用于 2D 环境下无人机控制方向的 FLC 的规划器，其包含了模糊逻辑与过滤顺滑的功能。Mahdi Fakoor[15]等人提出了在人形机器人领域利用模糊马尔科夫决策过程进行路径规划与轨迹生成。这些应用都取得了不错的效果。

（2）遗传算法

以蚁群算法为代表的遗传算法也是最近几年科研人员研究与改进的对象。第一个将蚁群算法应用在路径规划的例子由 Deneubourg[16]最先提出，通过仿生的手段达到了优化的

目的。基本的遗传算法如图 2.2 所示。

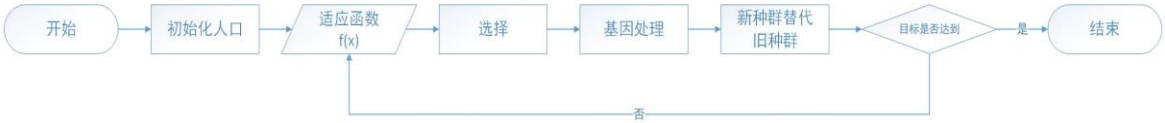


图 2.2 遗传算法基本流程

蚁群算法模拟了蚁群的一些特性，将整个域进行了路径图建模，并且在始末点初始化两个蚁群与蚁群的人口，分别派出蚂蚁向始末点前进，并且留下信息素（信息素浓度会随着时间下降）；后面的蚂蚁根据自己周围信息素的浓度进行一定概率的选择；结果将在蚂蚁相遇或者分别找到目标点之后显示出来。实验结果表明遗传算法的结果能减少中间域的数量，当然是在可接受时间范围内。

（3）人工神经网络

人工神经网络也是重点的研究方向之一，这个方法是由生物激发的方法，其基本的计算单位是叫神经元的单位。这个技术已经被广泛用于许多的研究优化、学习与模式识别中，神经网络能在复杂的环境中生成简单与最优的路径并且保持一定的整体特征性。Kevin van Hecke[17]等人提出了自监督学习的概念，研究了机器人应该按照什么样的学习行为去训练从而达到执行任务时的鲁棒性。也有使用粒子群优化算法与神经网络一起进行混合规划的，PSO 负责为神经网络产生的轨迹进行优化，使得其更加符合实际运动时的情景。

2.2.2 经典算法

经典的算法自从 2000 年时就被广泛使用，这些算法能够证明要么存在一条路径，要么就能证明环境中不存在可行的路径解，其在真实环境中运用主要局限于不能应对不确定性，一些比较经典的路径规划算法在以下部分进行了举例与讨论。

（1）栅格法

在栅格法中，我们将可行域分解为简单格子集合并且计算出每个格子的邻居格子关系。从初始域到终止域的非碰撞路线将由互相连接的栅格网络计算出来，其实也可以简化为图论搜索的方法，使用栅格法[18]实现出来。

（2）人工势场法

势场法首先由 Oussama Khatib[19]提出来，受到重力势场的启发，整个域可以整体建模为势场的互相叠加。整体势场是吸引势场域排斥势场的总和，其中吸引势场是目标域

对机器人的吸引函数，排斥势场是障碍物对机器人的排斥函数。其形式可以分别用数学进行如下表示，吸引势场是对于全局有效的，其定义为$U_{att}(q)$：

$$U_{att}(q)=\frac{1}{2}\zeta d^2(q,q_{goal}), \tag{2.4}$$

排斥势场只在障碍物附近一定距离中有效，所以其形式为分段函数，其定义为$U_{rep}(q)$：

$$U_{rep}(q)=\begin{cases}\frac{1}{2}\eta(\frac{1}{D(q)}-\frac{1}{Q^*})^2, D(q)\le Q^* \\ 0, D(q)>Q^*\end{cases}, \tag{2.5}$$

利用对域的整体建模，使得整体环境像重力势场一样使得机器人从初始点向目标点进行前进，当然如果在复杂环境下势场法过多的计算复杂度势必会受到影响，这也是它的一个弊端之一。

（3）路径图法

在路径图法中我们在可行域中抽取可行点集合将其组成一个可行点网络或路径，并且搜索路径就局限于该网络中。利用路径图法我们可以将路径规划问题简化为图论搜索问题。比较有名的路径图法是可视图法（VG），Voronoi 图法，Sihouette 和次目标（Subgoal）网络。其中可视图法在原理上是将凸障碍物的顶点作为特征点，并且将各个障碍物之间的特征点互相链接起来构成可视图，如图 2.3 所示；Voronoi 图法将空间切分成各个部分，其中每个部分包含靠近特定物体的点，Priyadarshi Bhattacharya 等人[19]在提出了利用 Voroni 图进行最短路径的选择从而解决规划问题。

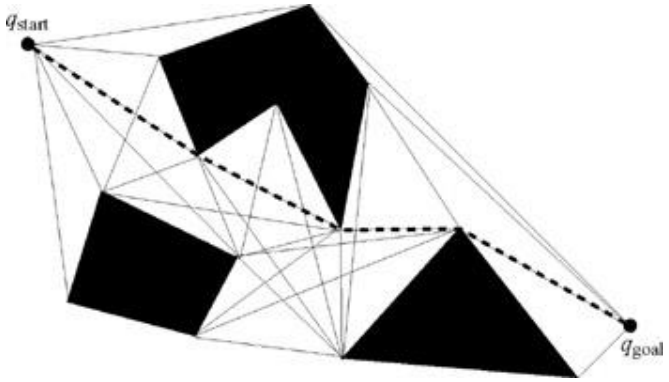


图 2.3 可视图法

由路径图法衍生出来一种算法十分的值得去研究，这种类型的算法拥有经典与启发式的两种特性，首先它肯定具有路径图法的采样特性，试图在可行域中建立可行的网络，其次他又有随机性，利用概率方法使得路径规划摆脱局部最小的问题。

在基于采样的路径规划算法中，例如快速搜索随机树，没有必要对整个域进行建模，

其在可行域中进行随机采样，并且将信息存入数据结构中，当目标域被采样得到之后，我们可以在已存储的数据结构中进行路径的搜索从而得到一条可行路径。基于采样的方法是概率完备但不是最优的方法，当然在不断的改进下还是得到了很大的进步。最近提出的一类用来解决路径规划的算法在实施起来十分的成功。基于采样的方法拥有概率的特性以及求解完备性，本论文采用了基于采样的方法进行了无人机的路径规划任务。

2.3 基于采样的路径规划与轨迹生成

在上述的所有方法中，经典算法大多需要对所处环境进行建模，无论是精确还是模糊建模都需要较大的计算量；另外，启发式算法如遗传算法等也需要极大的计算量与计算时间，这在 2D 环境下或许能够解决问题，但是一旦搜索域的维度增长，那么一般的上述规划器（planner）可能就会失效，因为收敛速度会急剧减缓，同时计算量也会大大增加。在一些场景中存在一些“移动问题”，例如在发动机设计时，为了零部件的维修需要确认有些零件能够在以后的时间被移出，利用经典的算法很难在三维空间中去进行计算或者根本无法进行计算。这类问题被称为“移动问题”（mover's problem），其中最为经典的是钢琴移动问题[20]，即如何在一个三维空间中将一架钢琴从初始点移动到最终点。

2.3.1 基于采样的规划方法综述

我们可以讲基于采样的规划方法看作一个黑盒，其输入是初始域与目标域的信息，一旦规划器获得该信息，那么就能返回具体的规划结果。如果能找到路径，就能返回一条可行的、无碰撞的可行路径，否则返回失败。关于基于采样的规划方法，其层次结构可以如图 2.4 所示。

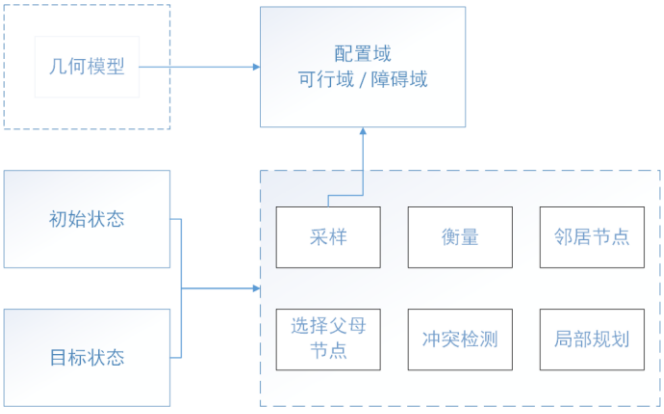


图 2.4 基于采样的规划方法层次结构

在每个采样规划器的构成中都有图示的基本 6 个部分，在本部分将简要介绍其概要的包含内容。

采样：这个部分是随机或者伪随机地对可行域进行采样，从而得到下一个可行状态，并将其信息加入到路径图或者树中。采样点可以在可行域中，也可以在障碍物中。采样过程是采样规划算法最主要的部分，也是其优势所在；

衡量：给定两个配置域 A、B，衡量部分返回一个值或者是代价来显示从 A 到 B 所需的努力（effort）。衡量可以有多种的标准，可以抽象成代价，也可以利用时间或者能耗作为衡量标准。

邻居节点：这是一个搜索部分返回的值，用来寻找距离最新采样域的最近或者在其邻居范围内的节点域信息。有些文章将其表述为接近搜索或者最近节点。

选择父母节点：这个过程将邻居节点与新采样点进行了连接，已有的节点被视为父母节点。

局部规划：这个过程对于两个配置域之间的连接进行规划，直观地，人们习惯利用直线将其连接起来，但是对于一些机器人来讲，直线轨迹不符合其动力与动理学方程。所以局部规划有利于提高机器人运行的安全性

冲突检测：这个过程一般返回布尔型值：成功或者失败。如果两个配置域之间有障碍域则返回失败，否则返回成功。

以上的每一个部分都有论文进行了相关的研究。在采样过程中，相关文献[21][22]提出了 Voronoi 图中的内轴采样概率提高的方法来利于对于配置域的完全采样；N. M. Amato [23]提出了向障碍物边界采样的强制性采样方法；类似于强制采样，V. Boor 等人[24]提出了基于高斯的适应性概率采样方法；S. Zheng 等人[25]提出了连接测试方法用来检测配置域中狭窄区域的存在情况；当然，相关文献[26][27]利用混合的方法提出了采样策略，能够较好地对于已有的采样策略进行整合。实验显示在每一个场景中，没有任何单一的采样策略能够优于其余的采样策略。作为采样规划算法中最主要的一部分，采样过程现在仍是一个活跃的研究领域。在同样比较重要的冲突检测过程中，一些规划器利用了反馈机制来引导搜索。相关文献[28][29][30]采用了采样检测策略，同时，有些文献[31][32]提升了环境的联通性来进行障碍物的检测。一些文献[33][34][35]也利用了慵懒规划策略，该策略直到其需要进行障碍物检测才进行。

2.3.2 基于采样的规划方法的特性

基于采样的规划方法一个最重要的性质就是它不需要对整个运动环境进行精确的建模，也就是说不需要对环境中的障碍物域进行建模，也不用对可行域进行建，一定程度上讲，采样的规划算法对于环境的接触是有局限性的。最近障碍物检测算法对于采样规划算法有了很大的帮助，未来所有在避障检测算法中的进步都会对基于采样的规划算法产生直接的推动作用。

另外一个重要的性质是基于采样的规划方法可以达到某些形式的完成度（some form of completeness），完成度即要求规划算法总是能够在渐进约束的时间范围内能正确地相应规划请求。一个完全完成度规划器（complete planner）在大于三自由度的机器人上是很难实现的，因为其高维复杂性使得规划问题变得很难。然而，一个较弱的完成度可以达到，即如果环境中存在一条路径，那么规划器在时间允许的条件下就能找到它。如果基于采样的规划器其采样过程是随机的，那么改规划器的完成度可以称为概率完成度（probabilistic completeness）。对于快速搜索随机树来讲其具有概率完成度的特征。

第三章　快速搜索随机树方法

3.1 快速随机搜索树搜索过程

快速搜索随机树（Rapidly-exploring Random Tree）是基于采样的算法，通过利用碰撞检测从而对于状态空间进行随机采样来探索路径是否存在。由于它具有不用对于环境空间进行建模的特性，所以能够非常有效地解决低维复杂和高维的运动规划问题。RRT能够在有效的可行域中充分地进行采样行为。算法利用采样得到的信息，能够较快的规划一条可行路径。

本章描述了简单快速搜索随机树（sRRT）的计算过程。sRRT 算法的目标是返回一颗单向的由初始域q_{init}到目标域q_{goal}的搜索树。如果搜索树能够探索至目标域，则可以通过对得到的搜索树抽象出一条可行的主干路径。

首先，由于需要对于状态空间进行充分的采样，生成新的节点是十分必要的，新节点的生成过程如下：在生成新节点q_{new}之前，配置域中会生成随机点q_{rand}；其次，计算出q_{nearst}，即搜索树中距离q_{rand}最近的节点；最终，新的节点q_{new}将以一定的步长从q_{nearst}向q_{rand}方向生成：

$$q_{new} = q_{nearst} + \delta * \frac{q_{rand} - q_{nearst}}{||q_{rand} - q_{nearst}||} \tag{3.1}$$

其中δ是固定的步长，$||q_{rand} - q_{nearst}||$ 表示q_{rand}和q_{nearst}之间的距离，表示$||q_{rand} - q_{nearst}||$的方法有很多，一般使用欧式距离表示两点之间的距离。

其次，需要对于生成的节点进行冲突检测，从而保证生成轨迹的安全性。在生成了q_{new}之后，需要对其进行判断是否在可行域中，若q_{new}不在可行域中，则进行下一轮迭代，若q_{new}在可行域中，则需要判断其与父节点q_{nearst}的边$edge(q_{nearst}, q_{new})$是否在可行域中，如果其和障碍域相交，则舍弃并且进入下一轮迭代。

最后，算法需要提出跳出循环的条件，即成功搜索至目标域。如果q_{new}搜索到了目标域则可以表示搜索树找到了一条可行路径并且返回一颗单向搜索树。另一种终止条件是

如果在预定的迭代次数中没有找到路径，则视为规划失败，返回无解。

快速搜索随机树的算法流程如图 3.1 所示。其中 i 表示迭代次数，k 表示迭代次数上限。

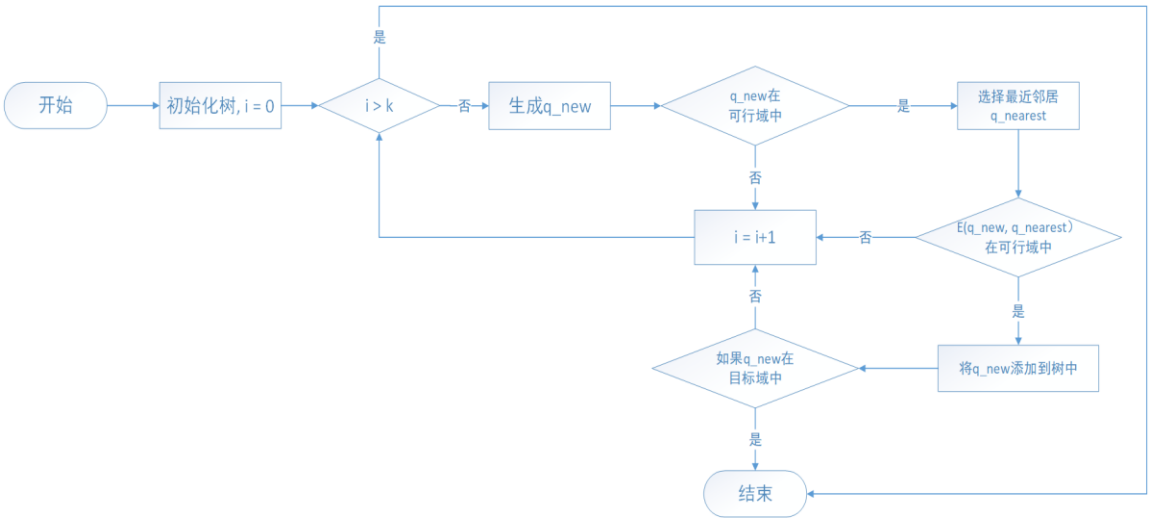


图 3.1 sRRT 算法流程图

为了形象地表示 sRRT 的规划过程，本文在图 3.2 中展示了 sRRT 的生长过程的例子。

图（a）展示了整个域（Configuration Space）的信息，其中黄色的点为初始域，绿色为目标域，灰色的为障碍物域，我们的目标是利用 sRRT 算法规划出一条非碰撞路径，使得机器人能顺利从初始域到达目标域；初始化时将初始点q_{init}加入搜索树 T 中；

图（b）表示的是以固定概率生成随机点q_{rand}；（c）表示从搜索树 T 中找到距离q_{init}最近的点q_{nearst}，一般的衡量距离都是欧式距离；

图（d）表示如果q_{nearst}和q_{new}组成的边在可行域中的话，那么就将新生成的点加入到搜索树中，并且将新边也加入到搜索树的数据结构中；

图（e）~（f）表示了循环生成q_{new}的过程；特殊的，（g）~（h）图表示了当生成q_{new}之后$Edge(q_{nearst}, q_{new})$不在可行域中，而是被障碍物域阻隔，这种情况不符合路径要求，所以要舍弃，我们得到搜索树的结果如图（i）所示；

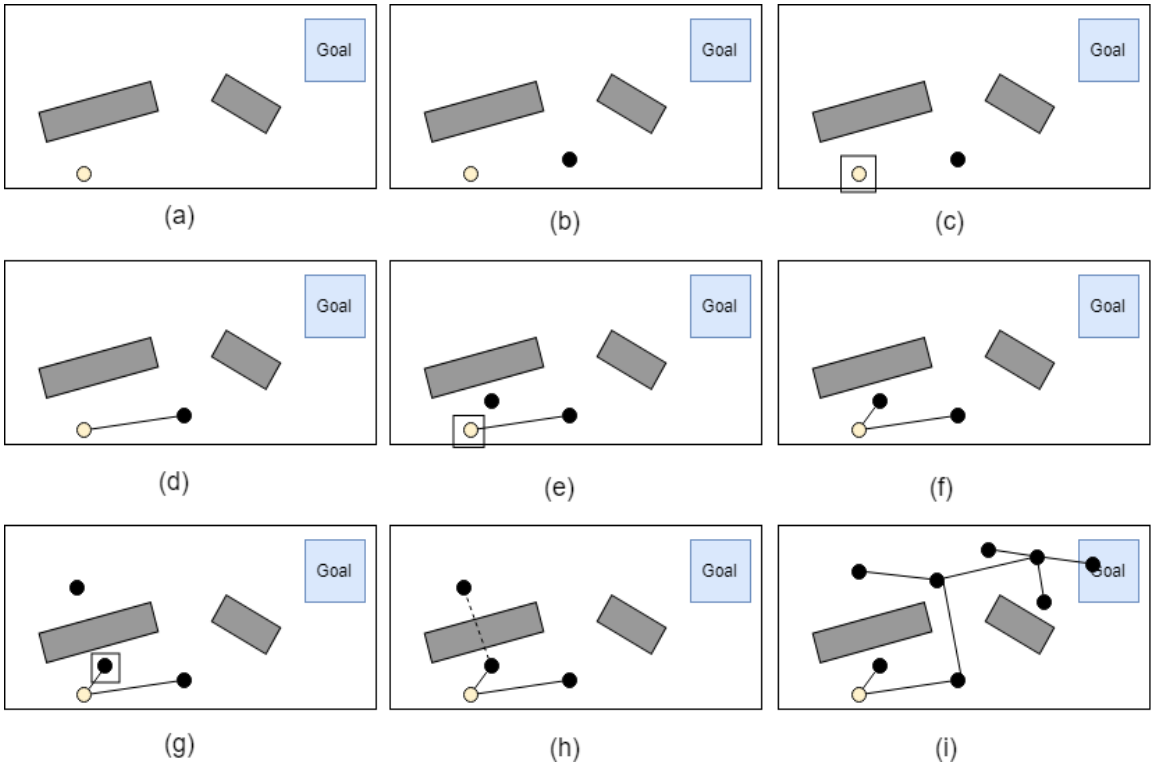


图 3.2 sRRT 搜索举例

具体的，通过伪代码能更加精确地了解 sRRT 搜索过程，其中规划主体如算法 3.1 所述。第三行表示算法将在在预定的迭代次数 k 中进行搜索树的探索操作。算法 4~6 行表示生成新节点q_{new}，而 7~13 行则表示对于生成的q_{new}和新生成的边$Edge(q_{nearst}, q_{new})$进行相应的冲突检测，任何一步冲突检测没有通过则自动舍弃q_{new}，并且进入下一次搜索迭代中。直到 14 行q_{new}可以探索到目标域为止，算法跳出循环并且返回搜索树。当然，如果迭代次数超过阈值 k，则表示路径规划失败，返回规划失败。

算法 3.1　简单快速搜索随机树

输入：初始域q_{init}；目标域q_{goal}

输出：搜索成功返回 T = 搜索树，失败返回搜索失败

1　　T.add(q_{init});

2　　p = RandomProbability( );

3　　**For** i = 1→k

4　　以 p 概率将q_{goal}作为q_{rand}，以 1 - p 的概率在域中随机生成q_{rand} ；

5　　计算得到搜索树 T 中距离q_{rand}距离最近的q_{nearst};

6 计算$q_{new} = GenerateQNew(q_{rand}, q_{nearst}, \delta)$;
7 **If** q_{new} 不在域中
8 **Continue**;
9 **End If**
10 **If** q_{new}不在障碍物域中
11 **If** $Edge(q_{nearst}, q_{new})$在可行域中
12 Vertices ← q_{new};
13 Edges ← $Edge(q_{nearst}, q_{new})$;
14 **If** q_{goal}在搜索树范围内
15 Vertices ← q_{goal};
16 Edges ← $Edge(q_{touched}, q_{goal})$;
17 **Return** T
18 **End If**
19 **End If**
20 **End If**
21 **End For**
22 **Return** 规划失败

当得到搜索树的路径规划结果之后，需要进行一些处理之后才能生成相应的轨迹。在 sRRT 算法中的碰撞检测已经保证了路径的安全性，即不会与障碍物域发生碰撞的前提下，首先需要利用剪枝算法计算出主干路径；进一步，通过抽象算法对主干路径进行简化；最后经过柔顺算法使得转角更加适合无人机进行通过。这样生成的轨迹才是符合四旋翼无人机运行的。

在算法 3.2 中，本文提出了 sRRT 的剪枝算法，在得到搜索树结果后，我们更需要从搜索树结果中解析出一条主干路径。由于 sRRT 只存在一条主干路径（即从初始点至目标点的路径）的特性，所以可以对除了主干树枝以外的树枝进行剪枝操作。为了不将目标域节点作为叶子节点进行删除，在算法第 4 行中首先将其从算法删除范围中排除，然后 5~9 行循环对于叶子节点进行删除与更新。最终通过算法 3.2 的处理我们可以得到一条主干路径。

算法 3.2 sRRT 剪枝算法

输入：搜索树 = T；目标域 = q_{goal}

输出：剪枝后的的关键路径 T’

1 FirstVertices ← 数据结构 Edge 中的第一个顶点信息；
2 SecondVertices ← 数据结构 Edge 中的第二个顶点信息；
3 LeafVertices ← $FindLeafVertiecs(FirstVertices, SecondVertices)$；
4 将q_{goal}从 LeafVertices 中删除，保留主路径；
5 **While** LeafVertices 非空
6 更新 Vertices，从 Vertices 中删除 LeafVertices；
7 更新 Edge，从 Edge 中删除叶子边；
8 LeafVertices ← $FindLeafVertiecs(FirstVertices, SecondVertices)$；
9 **End While**
10 **Return** T’

在得到搜索树的主干路径之后，往往从初始域到目标域的路径都不是最优的，路径会存在蜿蜒或者 Z 字型的情况。为了解决这个问题，算法 3.3 提取了剪枝路径中的几个关键节点，使得生成的简化路径既能满足在可行域中，又能满足现实中的机器人运行要求。

如算法 3.3 的第 3 行所示，算法从主干路径中的第二点进行搜索，同时对$Edge(q_{index}, q_i)$进行冲突检测，即表示在部分可行域中尽可能地简化主干路径，只要两点连线不与障碍物域相交。算法 3.3 将得到一条最简化的安全路径。

算法 3.3 RRT 路径抽象（skeleton）算法

输入：关键路径 T'

输出：简化路径 P

1 index ← 1;
2 AbstractedVertices ← Vertices(index);
3 **For** i = 2 → VerticesQuantity(T')
4 **If** $Edge(q_{index}, q_i)$ 与障碍物域冲突
5 index ← i - 1;

```
6        AbstractedVertices ← AbstractedVertices ∪ Vertices(index);
7      End If
8    End For
9    利用 AbstractedVertices 构建一条抽象出来的路径 P；
10   Return P
```

在获得了简化路径之后，虽然可行性大大提高，但是转角处仍会存在转角过大，与障碍物距离过近的情况，争对这种情况我们需要对转角进行顺滑从而使得轨迹更加符合运行条件。

在算法 3.4 中，如第 1 行所示，算法对于简化路径中除了始末点之外的节点都进行了顺滑操作，第 5 行首先在转角处依靠简化路径与转角节点形成控制三角形；算法第 6 行表示在三角形的约束下利用 Bézier 方法生成一条柔顺的曲线，第 7 行表示将生成的顺滑轨迹加入到整体轨迹之中去。在经过了以上算法的处理之后我们能从简化路径中提取出一条从初始域到目标域的可行柔顺路径。

算法 3.4　RRT 路径顺滑（smooth）算法

输入：简化路径 P

输出：顺滑轨迹 Trajectory

```
1    TurningVertices ← Vertices \ {q_init ∪ q_goal}
2    Curve ← {};
3    Trajectory ← {};
4    For i = 1 → VerticesQuantity(TurningVertices)
5      在转角点 TurningPoint 依靠路径 P 设置凸控制三角形 CvxPolygon;
6      在 CvxPolygon 中生成顺滑曲线 C；
7      Curve ← {Curve∪C}；
8    End For
9    将直线部分与曲线部分 Curve 合并加入 Trajectory;
10   Return Trajectory
```

3.2 快速随机搜索树路径柔顺方法

本节内容具体阐述上一部分提出的顺滑算法 3.4 的原理，顺滑算法利用了基于样条的曲线生成方法，能使得一次转弯角度较大的斜角转变为无人机可行的轨迹。

在数值分析中，样条属于一种特殊函数，其定义由多项式分生成。样条英文名 Spline 来源于可变形样条工具，通常是造船或者工程图纸中运用到的用来描绘光滑曲线的工具。显然的，生成光滑曲线需要插入一些点才能在其约束下利用多项式去生成。同时一般情况下，利用样条插值比多项式插值效果好，例如利用低阶样条插值能产生和高阶多项式插值类似的效果，并且还可以避免称为“龙格现象”的一种数值不稳定的现象出现。低阶的样条插值还具有“保凸”的性质。本论文柔顺算法中就利用了相关的方法去生成了二次 Bézier 曲线。

给定两个点P_0和P_1，Bézier 的一次曲线定义如下：

$$\mathbf{B}(t) = (1-t)\mathbf{P}_0 + t\mathbf{P}_1,\ 0 \le t \le 1, \tag{3.2}$$

给定三个点P_0，P_1和P_2，利用一次 Bézier 曲线定义进行迭代，我们能得到二次 Bézier 曲线（Quadratic Bézier Curve）定义如下：

$$\mathbf{B}(t) = (1-t)[((1-t)\mathbf{P}_0 + t\mathbf{P}_1)] + t[((1-t)\mathbf{P}_1 + t\mathbf{P}_2)],\ 0 \le t \le 1, \tag{3.3}$$

简化后能够利用如下的公式进行曲线的生成：

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2,\ 0 \le t \le 1, \tag{3.4}$$

公式中的 t 可以表示为赋予路径以时间的信息，也可以表示采样比例，用来迭代生成采样点序列形成一条顺滑的轨迹，本论文依照二次 Bézier 曲线提出了转折节点的顺滑方法，如图 4.2 所示，蓝色表示原始路径，其中最高点是原始搜索树的节点之一，在原始路径分别前后取点P_1和P_2，构成如图的绿色的控制三角形。我们可以将无人机的路径由$P_1 \to P_2 \to P_3$转变为$P_1 \to P_3,\ s.t.\ P_2$，通过将P_2转化为控制点我们可以生成一条光滑曲线。

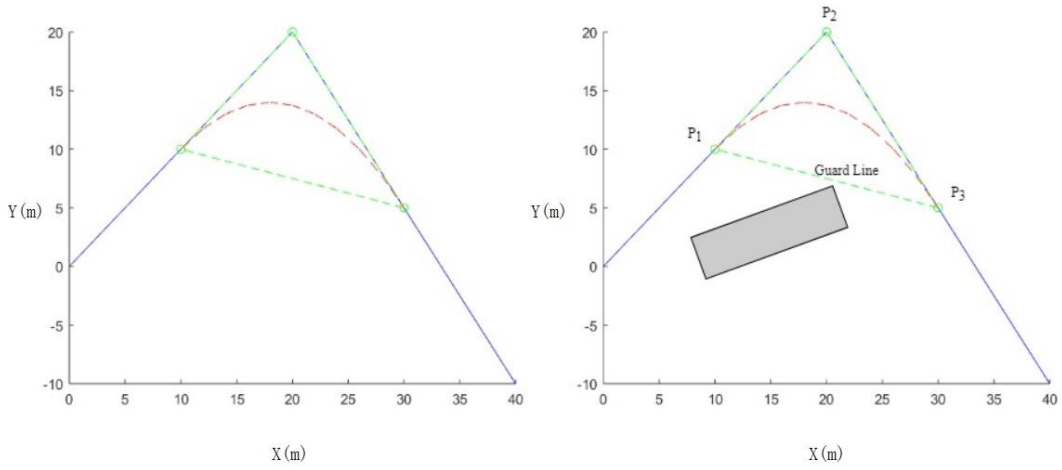


图 4.2 路径柔顺算法

值得注意的是，由于域中有障碍物域，在构成控制三角形时，我们将P_1P_3称为 GuideLine，本论文采取冗余的方法，要求 GuideLine 在可行域中，不能与障碍物域冲突从而一定保证了曲线路径一定在可行域中。

另外，作为一条可运行的路径，必须要求在入点P_1和出点P_3是连续且可导的才能保证轨迹的鲁棒性。对于$\mathbf{B}(t)$进行求导，可以得到下式：

$$\mathbf{B}'(t) = 2(1-t)(\mathbf{P}_2 - \mathbf{P}_1) + 2t(\mathbf{P}_3 - \mathbf{P}_2), \tag{3.5}$$

对于入点P_1是 t = 0 的情况得到$\mathbf{B}'(0) = 2(\mathbf{P}_2 - \mathbf{P}_1)$，表明轨迹在曲线入点处时是顺滑的，同理出点$P_3$对应 t = 1 的情况得到$\mathbf{B}'(1) = 2(\mathbf{P}_3 - \mathbf{P}_2)$表明轨迹在曲线出点处也是顺滑的。保证了二次 Bézier 曲线运用在 sRRT 中是连续可行的。

第四章　双向快速搜索随机树方法

单向的 sRRT 算法通过对于状态空间进行充分的采样能够探索出一条可行的路径，但由于其每次只能够单方向的对于状态空间进行搜索，效率较低。本章主要介绍了双向的快速搜索随机树方法，双向 RRT 能够从始末点双向地对于状态空间进行采样搜索，平均迭代次数中，双向 RRT 算法无疑能够探索更多的空间，并且相向生长的特性保证了其找到路径的概率完备性，使得路径规划的效率上升。

4.1 基本双向搜索树

基本的双向搜索树的设计思想是：从初始域和目标域出发，交替构建两颗搜索树；在每次的迭代的过程中，两颗搜索树总是彼此朝着对方拓展，直至两棵树相遇为止，基本的双向 RRT 构建流程如图 4.1 所示。可以看到，在迭代中，首先对①树进行新节点$q_{new,1}$的生长；如果其有效，则以$q_{new,1}$为目标进行②树新节点$q_{new,2}$的生长；如果$q_{new,2}$也是有效并且可以接触到$q_{new,1}$，则表示两棵搜索树存在交点，也就表示算法得到了一条连接初始点与目标点的路径。值得注意的是，算法并不以固定一颗树作为迭代中的①树，在每次迭代结束之前都会进行 Swap 操作，达到“交替搜索”的目的，这样能够使得两棵搜索树更加均衡，同样的，对于状态空间的采样更加广泛均匀。

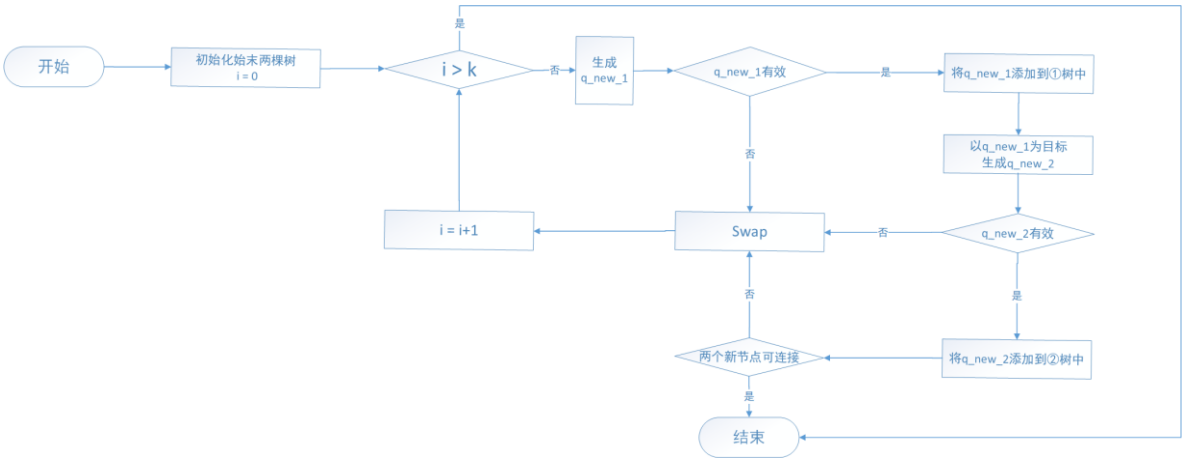


图 4.1 双向 RRT 算法流程图

为了形象地表示双向 RRT 的规划过程，本文在图 4.2 中展示了双向 RRT 的生长过程的例子。图（a）表示初始化两棵搜索树的初始节点，分别位于初始域与目标域中。

图（b）表示先从初始搜索树方面开始拓展节点，并且寻找新节点$q_{new,1}$的方式与简单 RRT 一致，均是由$q_{nearst,1}$向$q_{rand,1}$方向生长δ距离所得到的。

图（c）表示在目前搜索树迭代完成之后，切换更新另一颗搜索树的节点，并且在这次的迭代中，目标域搜索树是将$q_{new,1}$作为随机点进行拓展从而保证了两棵搜索树向同一个方向进行搜索，图中可见目标域的搜索树也开始拓展了。

图（d）展示了在拓展中的特殊情况，在初始域搜索树进行完更新迭代之后，目标域搜索树的新节点$q_{new,2}$与最近节点$q_{nearst,2}$之间存在障碍物域，所以在这次迭代之中，目标域搜索树并没有能够进行新节点的探索。

图（e）表示在进行了 SWAP(T_{init}, T_{goal}) 操作之后，目标域搜索树开始了更新树节点的操作，向目标域搜索树方向加入了新的节点$q_{new,1}$。

图（f）~（g）表示了迭代结束的条件，当两棵搜索树的新节点$q_{new,1}$与$q_{new,2}$在一定距离内并且中间没有障碍域时，可以认为路径寻找成功，将$q_{new,1}$与$q_{new,2}$连接起来即可得到初始域到目标域的路径。

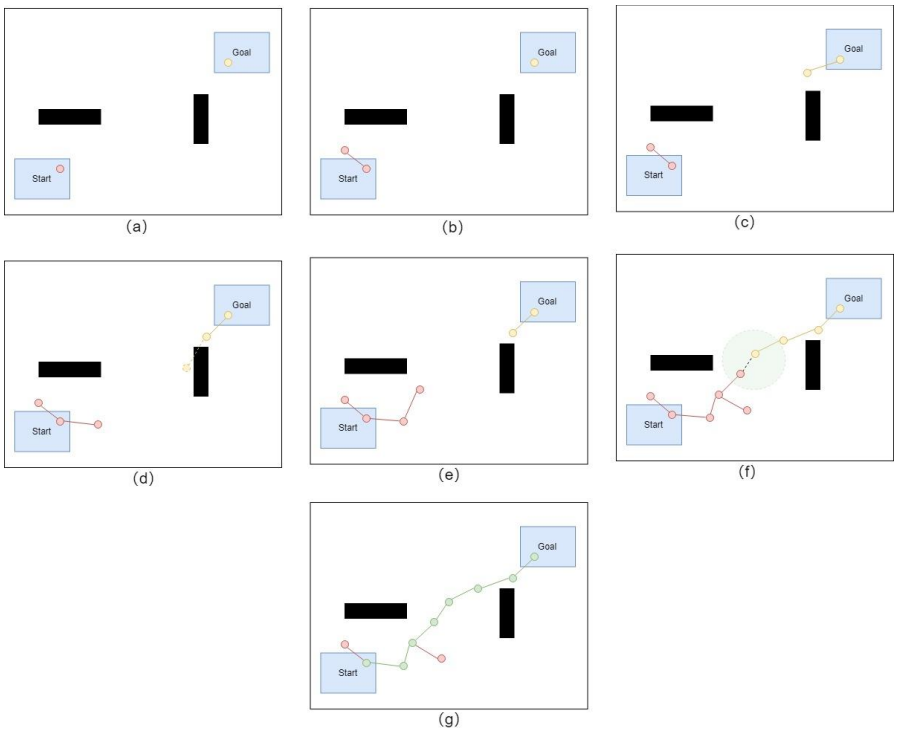


图 4.2 双向搜索树搜索举例

具体的，通过伪代码能更加精确地了解双向 RRT 搜索过程，基本双向搜索树的算法主体流程如算法 4.1 所示，在一定的迭代次数 k 中，如果算法寻找到路径，则返回一条可行的路径，否则返回失败。算法 4.1 中的第 2~3 行，在第一棵搜索树中生成新的节点$q_{new,1}$，如果$q_{new,1}$不在可行域中则进行下一轮迭代，如果在可行域中，4~5 行对第二棵树生成新的节点$q_{new,2}$，如果$q_{new,1}$=$q_{new,2}$，那么返回搜索树，否则进行了 Swap 操作之后进入下一次迭代生长。通过算法 4.1，我们将得到两棵搜索树，分别以初始点与目标点为根节点。

算法 4.1　双向搜索树 Bidirectional RRT

输入：T_{init}；T_{goal}

输出：成功返回搜索树 T，失败返回规划失败

```
1     For i = 1→k
2       随机生成q_rand
3       q_new,1 ← Extend RRT(T_1, q_rand)
4       If q_new,1 有效
5         q_new,2 ← Extend RRT(T_2, q_new,1)
6         If q_new,1 = q_new,2
7           Return Path
8         End If
9         SWAP(T_init, T_goal)
10      End If
11    End For
12    Return 规划失败
```

算法 4.1 中的 Extend 算法如算法 4.2 所示，算法 4.2 的函数形式为 Extend RRT(T, q)，其中 T 表示搜索树，q 表示搜索树向着点 q 的方向生长。算法中的 1-2 行生成了新节点q_{new}，第 3 行判断了新节点是否在可行域中，如果不在则返回 NULL，如果在可行域中，则在第 4 行中判断边$E(q_{near}, q_{new})$是否与障碍物域有交点，如果有，则拓展节点不成功，返回 NULL，否则将q_{new}添加到树中，并且返回q_{new}。

双向搜索树通过两个方向对于整体环境的搜索，大大加快了对于可行域空间的采样速度，使得算法能够更快地找到可行解。

算法 4.2 Extend RRT 算法

输入： T_{init}; T_{goal}

输出： 成功返回q_{new}，失败返回 NULL

```
计算得到搜索树 T 中距离q_rand距离最近的q_nearst
计算q_new = GenerateQNew(q_rand, q_nearst, δ)
If q_new不在障碍物域中
  If Edge(q_nearst, q_new)在可行域中
    Vertices ← q_new;
    Edges ← Edge(q_nearst, q_new);
    Return q_new
  End If
End If
Return NULL
```

4.2 同时更新双向搜索树

基本双向搜索树利用“双向”的特点，能够较快地对于状态空间进行采样，使得结果能够较快地被规划出来，但是其迭代中的交替增长的本质仍然采用了随机的性质，这个性质使得两棵树的搜索范围会增大，虽然这样设计保证了两棵搜索树可以脱离局部最小的陷阱，但总体上对于普遍存在众多可行域连续的情况，其搜索会由于随机采样这一特点而比较发散。

为了改进基本双向搜索树随机采样这一特点，我们在改进的双向 RRT 中增加了贪婪的策略，使其搜索较为收敛。同时对于可能产生的局部最小的陷阱，利用随机采样这一方法对于其他可行域再次进行探索以建立其他可行路径。

总的来说，相比于基本双向搜索树，同时更新双向搜索树算法添加了贪心算法的相关思想，“同时更新”即先使两棵搜索树中最近的两个节点进行相向的同时增长，当其中任何一方遇到障碍物域或者陷入局部陷阱时，就采用随机算法使得搜索树向其他可行域进行充分的采样，以建立其他的可行路径网络；然后在下次迭代中再从两棵搜索树中选出最近的节点进行相向生长，以此循环，直到两个新的节点在连接范围并且没有障碍物域

阻隔的情况下，连接两个新生成的节点并且返回相应的可行路径。具体算法流程如图 4.3 所示。

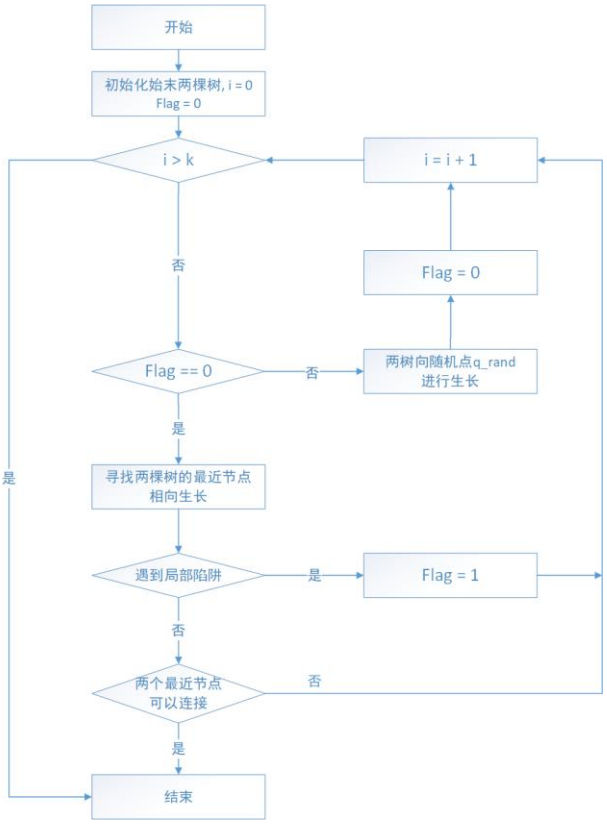


图 4.3 改进的双向搜索树流程图

具体的，通过伪代码能更加精确地了解改进双向 RRT 搜索过程，在算法 4.3 中，Flag 表示两棵搜索树陷入局部最小的标志状态，Flag = 1 表示两树在同时拓展时遭遇了无法顺利生长的问题，Flag = 0 则表示两树可以利用贪心策略相互彼此生长。算法 4.3 中第 3-4 表示采取贪婪策略从两棵树中寻找最近的两个节点进行相向生长，第 5-9 行表示相向生长遇到阻碍，将 Flag 标定为 1 并且进行下一轮迭代。第 10-12 行表示如果能在贪心策略下找到路径则返回搜索树。第 13-17 行用来解决当搜索树陷入了局部最小的情况，两棵搜索树通过向随机产生的节点q_{rand}进行同时生长对剩余的空间进行探索，将 Flag 标定为 0，进行下一次迭代，直到贪心策略能够将两个最近节点连接为止。

算法 4.3　同时更新双向搜素树 （Simultaneously-update Bidirectional RRT）

输入：T_{init}; T_{goal}; Flag

数据：成功返回搜索树 T，失败返回规划失败

For i = 1→k
  If ~ Flag（没有遇到障碍物或者陷入局部陷阱）
    寻找两棵搜索树中的最近节点 V_{Src} 和 V_{Dst}
    两棵树分别从 V_{Src} 和 V_{Dst} 相向生长
    If 搜索树遇到障碍物或者陷入局部陷阱
      生长没有遇到该情况的搜索树
      Flag ← 1
      Continue
    End If
    If 新生长的节点New_{Src} 和 New_{Dst} 可以连接
      Return Path
    End If
  Else
    随机生成 q_{rand}
    T_{init} 和 T_{goal} 向 q_{rand} 生长，探索其余可行域
    Flag ← 0
  End If
End For
Return 规划失败

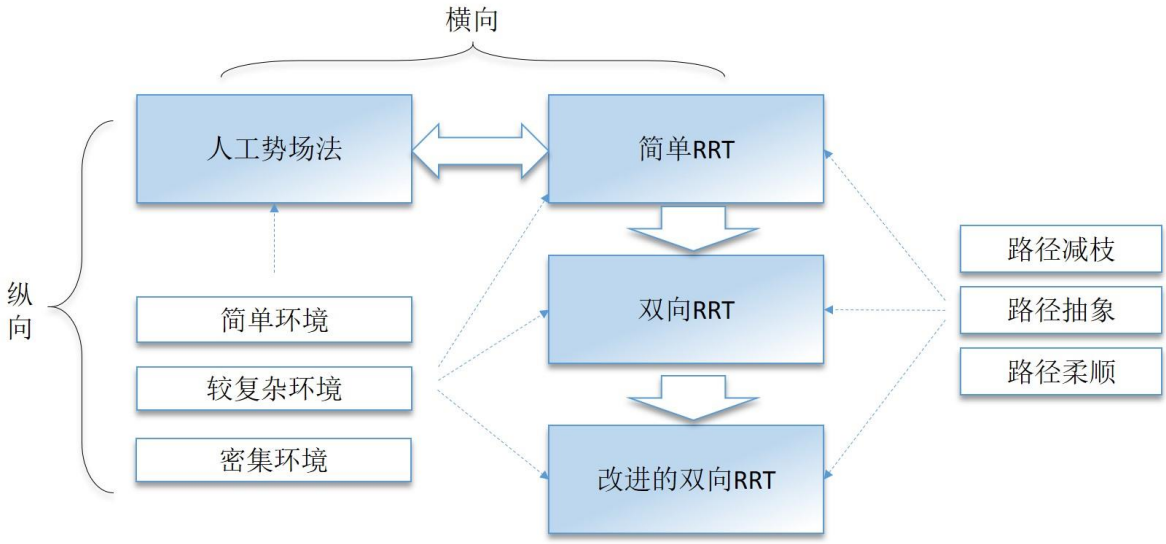
第五章　实验部分

5.1 实验环境与信息

实验所用的环境如下所示：

1. 操作系统 Windows 10 Professional Version
2. 软件及版本：MATLAB R2017a
3. 版本控制工具：Git
4. 代码托管地址：https://github.com/Lewis-Lu/RRTSimulation

实验的流程如下：



首先，实验部分在简单的实验环境下横向比较了经典算法和单向 RRT 之间的规划能力；其次在纵向，从深层次比较了 RRT 及其衍生算法的规划能力，并且在 RRT 算法规划成功后，利用相关的轨迹简化与柔顺算法得出可行的飞行轨迹。

5.2 实验结果

本实验验证了多方面的结果，全面地比较了各种算法之间的特点与性能。首先将基于采样的算法与经典的建模类算法进行比较，体现其在复杂环境应用场景上的优势与快速

性；其次分别在复杂场景下利用 sRRT，双向 RRT 以及同时更新双向 RRT 来进行了实验，测试提出算法的鲁棒性、轨迹柔顺性以及生成的效果。

5.2.1 人工势场法与基于采样算法比较

人工势场法属于对环境进行精确建模类的算法，其主要原理在第三章已经给出，主要利用吸引函数（3.2）与排斥函数（3.3）对于整体环境进行建模，使得目标点处于最低点，机器人借助“势场”的作用达到安全行驶到目标点的目的。由于需要对于具体的障碍物域进行数学建模，所以在障碍物众多的环境中，经典类算法常常需要花费很长的时间才能生成路径，而这个往往是我们所无法忍受的。并且如果目标点处于障碍物域旁时，利用人工势场法计算后，机器人是无法到达目标点的。而基于采样的方法只需要对于可行域进行快速的采样以形成可行的网络节点而无需对于整体的环境进行精确的数学建模。下面给出两种算法对比实验的结果。

图 5.1 中显示的是较为简单环境下的人工势场法与简单快速搜索随机树的生成路径的比较。（a）是由人工势场法生成的轨迹，显然其生成的轨迹符合其算法原理，由于需要对于障碍物进行建模，其排除函数必然也是高次函数，所以其生成的轨迹必然是曲线的，其行进代价较高并且路径中转弯过多不利于机器人的行驶；（b）是由 RRT 算法生成的轨迹，经过了第四章提出的算法处理过后，其生成的轨迹显然是简单并且代价较小的一种，主要原因还是由采样特性决定，没有将障碍物域纳入衡量的范围中。

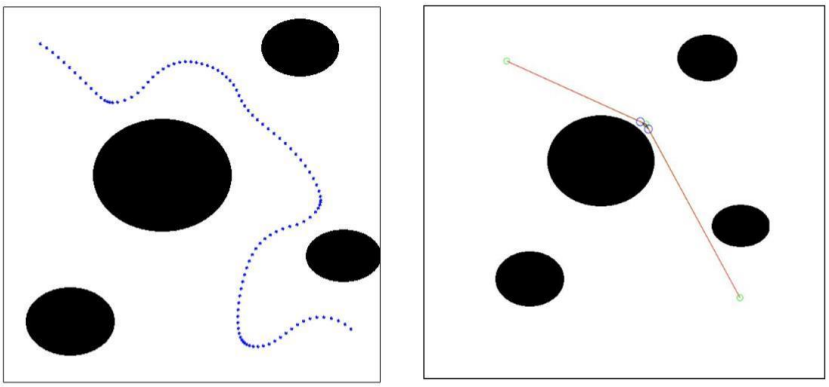


图 5.1 简单环境下经典与基于采样算法的比较

本文也在密集障碍物环境对两种算法进行了实验，从图 5.2 的（a）可见，势场法显然不适合在密集环境中进行轨迹生成的任务，究其原因，主要是因为多个障碍物模型可

能会形成局部最小，使得计算在收敛过程陷入该局部最小中从而导致规划失败。图（b）展示了 RRT 等基于采样算法在密集环境中的良好性能，结合轨迹抽象与轨迹柔顺算法，我们可以在该环境中很好地生成一条可行轨迹。

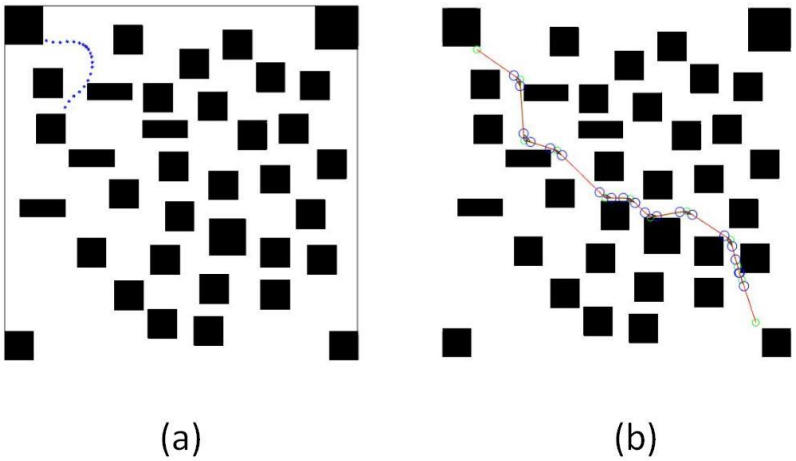


图 5.2 密集环境下经典与基于采样算法的比较

5.2.2 简单快速探索随机树实验

该部分的实验在不同环境下检验了第三章的 RRT 算法，已经提出的柔顺算法。实验结果如下所示。

图 5.3 表示了 sRRT 在简单环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

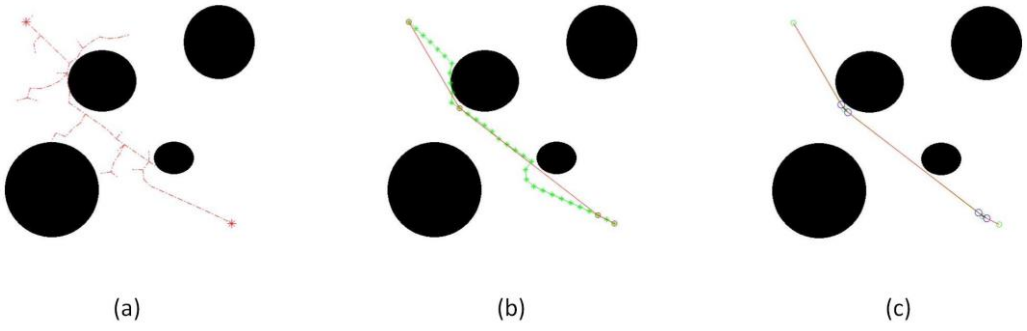


图 5.3 简单环境下 sRRT 实验

图 5.4 表示了 sRRT 在相对复杂环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法

处理的最终轨迹结果。

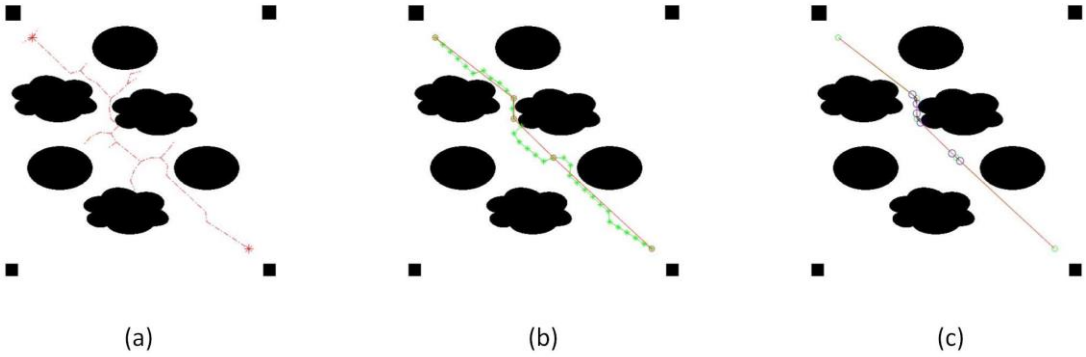


图 5.4 稍复杂环境下 sRRT 实验

图 5.5 表示了 sRRT 在迷宫环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

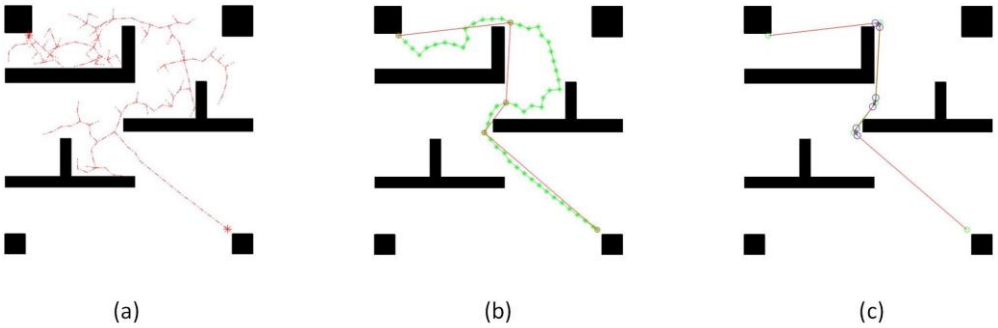


图 5.5 迷宫条件下 sRRT 实验

图 5.6 表示了 sRRT 在复杂环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

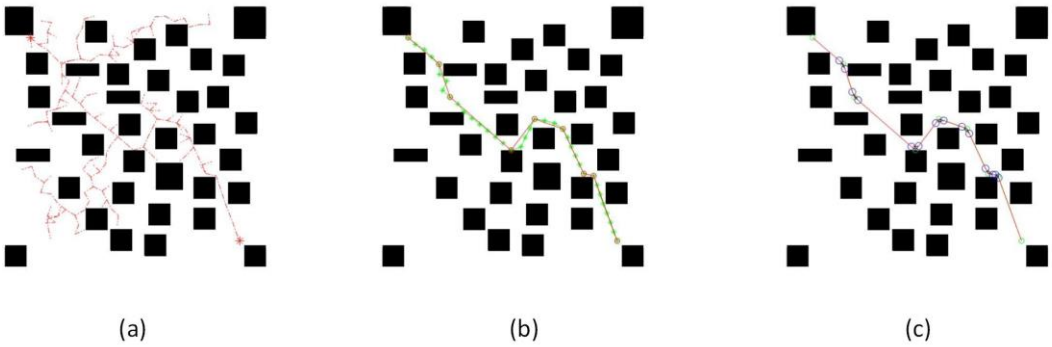


图 5.6 复杂环境下 sRRT 实验

为了更好地展示生成的轨迹细节，本部分还展示了相关的转角柔顺细节。如图 5.7 所示，该图展示了在较为复杂环境下的最后生成轨迹细节，可以看到在控制三角形的约束下轨迹拐角得到很好的顺滑与保护，确保了轨迹不会与障碍物进行接触使得无人机能够安全的飞行。

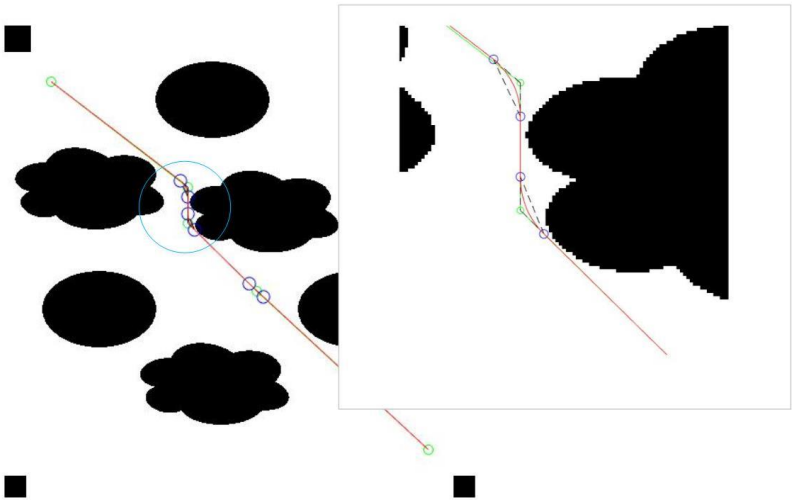


图 5.7 SRRT 较为复杂环境转角细节

在复杂环境中，RRT 算法也呈现出了它的优势，能够快速地规划出轨迹，并且图 5.8 也展示出了柔顺算法在复杂环境下的鲁棒性，能够在转角处自适应地进行控制三角形大小的规划；并且在转角密集的地方能够避免控制点交错，轨迹出现 Z 字型的情况。

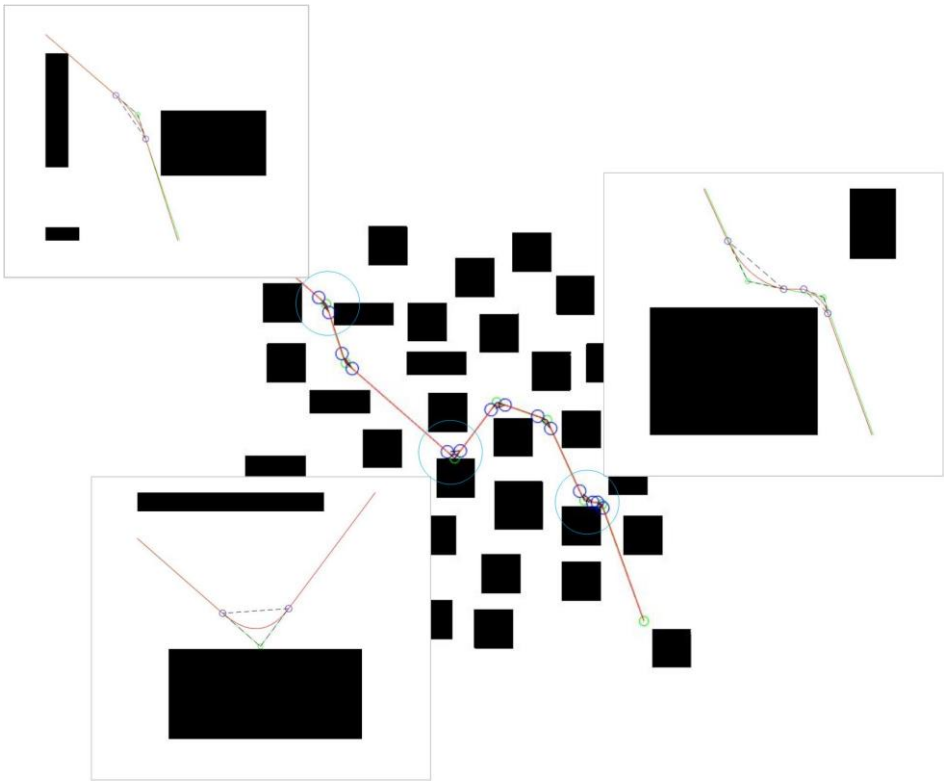


图 5.8 sRRT 复杂环境转角细节

5.2.3 双向快速探索随机树实验

对于双向 RRT 算法，最主要是其在原有的基础上加快了搜索的进程，如图 5.9 所示，其中红色表示初始域搜索树，蓝色表示目标域搜索树，两棵搜索树交替采样整个可行域，在两棵搜索可以连接之后就形成了一条可行路径。

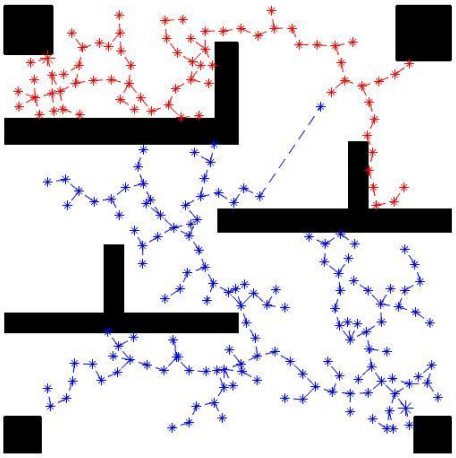


图 5.9 biRRT 迷宫环境采样结果

图 5.10 表示了 biRRT 在迷宫环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

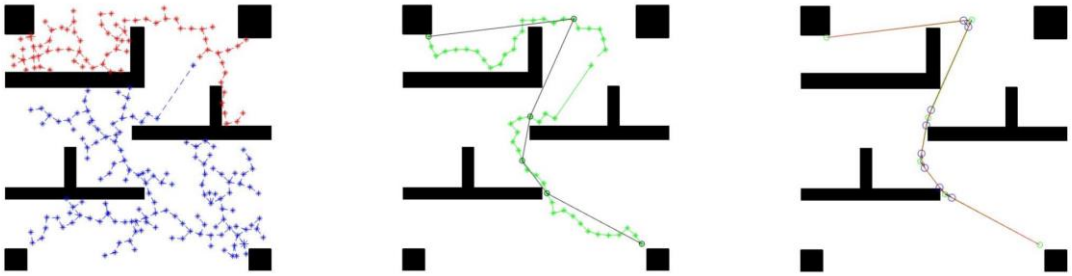


图 5.10 biRRT 迷宫环境轨迹生成

图 5.11 表示了 biRRT 在复杂环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

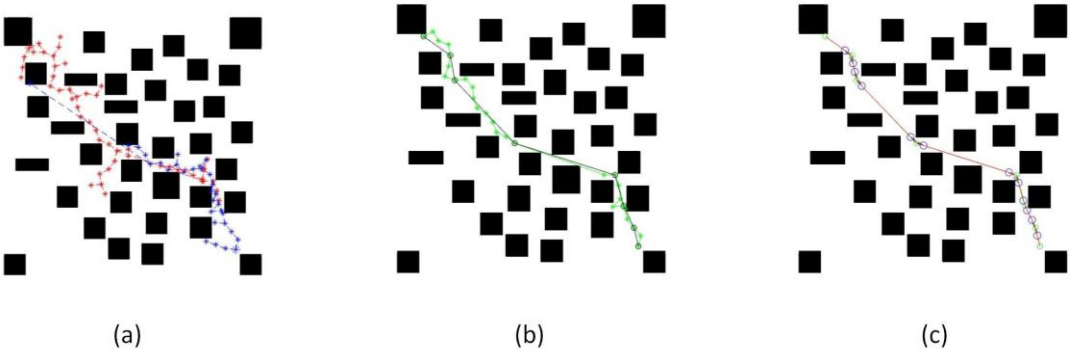


图 5.11 biRRT 复杂环境轨迹生成

5.2.4 同时更新双向快速随机树实验

不同于基本双向 RRT 算法，提出的同时更新双向 RRT 算法使得两棵搜索树的节点能够更加快速地互相同时增长，类似贪心算法的思想使得两个节点能够较快的实现连接。当然为了防止陷入局部最小的环境，算法也利用了随机方法来将搜索树拜托局部最小环境。同时由于使用了贪心算法的概念，两棵树生长的节点数量也会相较于基本双向 RRT

更少，一定程度上节省了计算资源。

图 5.12 表示了同时更新 biRRT 在复杂环境中的轨迹生成结果，其中，（a）表示 RRT 原始规划结果；（b）表示通过抽象与提取算法得到的主要路径的结果；（c）表示通过柔顺算法处理的最终轨迹结果。

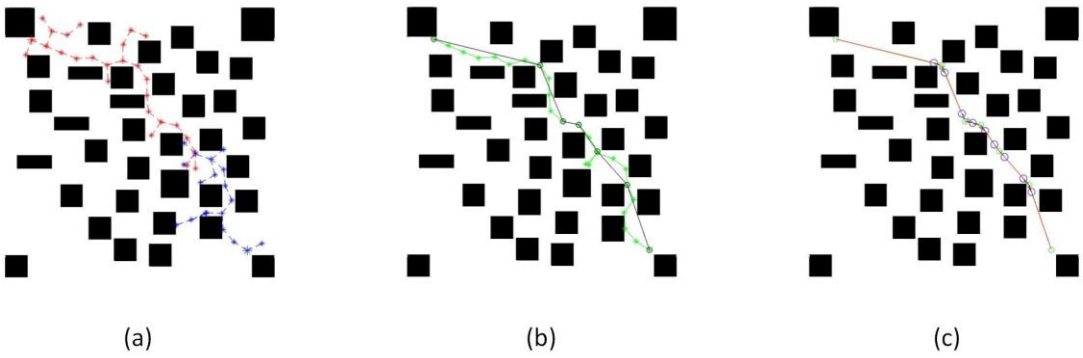


图 5.12 同时更新 biRRT 复杂环境轨迹生成

5.3 实验分析与总结

5.3.1 规划时间分析

该部分对于每一个 RRT 及其衍生算法进行了独立的批量实验，分别以复杂环境作为标准，进行了 100 次实验结果，并且记录下来了每次规划的时间并计算出了各个算法的平均规划时间。

图 5.13 表示简单 RRT 规划时间与平均规划时间；图 5.14 表示基本双向 RRT 规划时间与平均规划时间；图 5.15 表示同时更新双向 RRT 规划时间与平均规划时间。

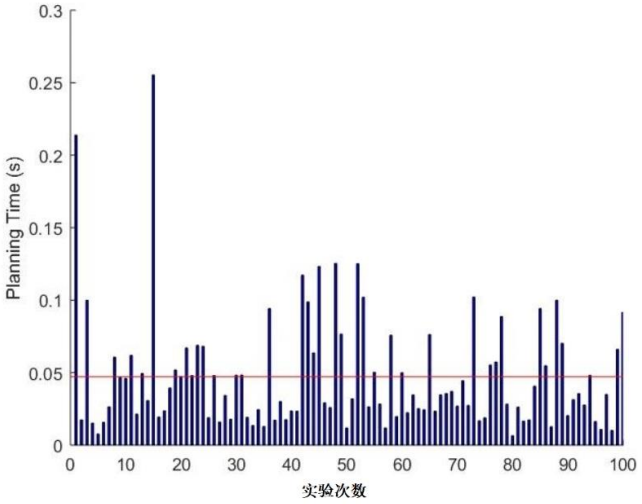


图 5.13 基本 RRT 复杂环境规划时间分布

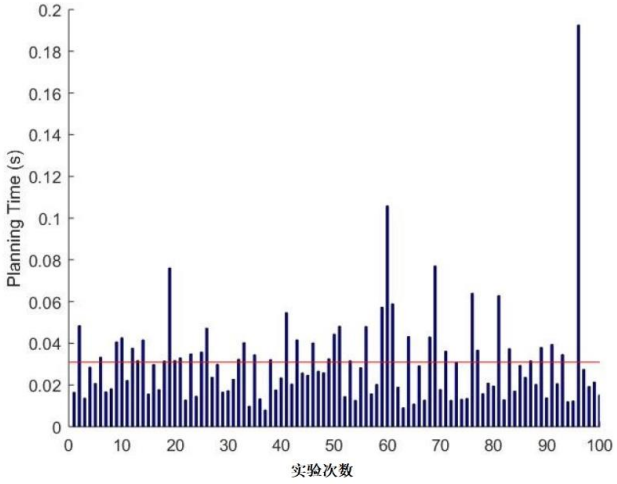


图 5.14 基本 biRRT 复杂环境规划时间分布

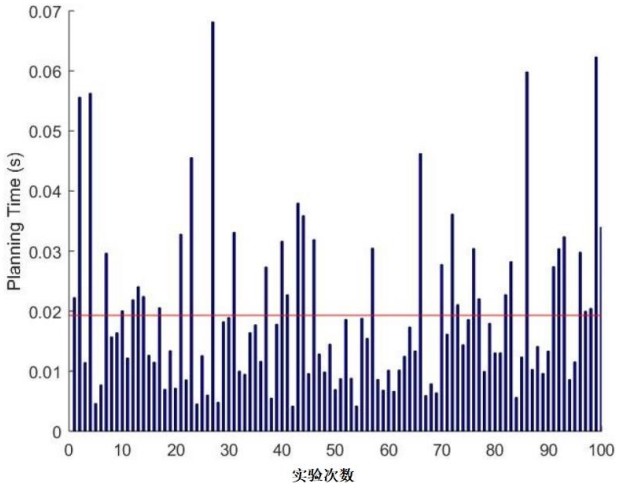


图 5.15 同时更新 biRRT 复杂环境规划时间分布

其中基本 RRT 算法即单向搜索树的平均规划时间为 0.0472s，基本双向 RRT 算法的平均规划时间为 0.0310s，同时更新双向 RRT 算法的平均规划时间为 0.0234s。从一定程度上验证了算法发展的有效性。

5.3.2 搜索树分析

该部分以复杂环境为基础对于快速搜索树产生的搜索路径进行了简单的分析，从路径质量，节点数量来进行一些基础的比较，来探讨算法发展的关系。

图 5.16 表示了复杂环境中搜索树结果，其中，（a）表示基本 RRT 原始规划结果；（b）表示基本双向 RRT 原始规划结果；（c）表示同时更新双向 RRT 原始规划结果。

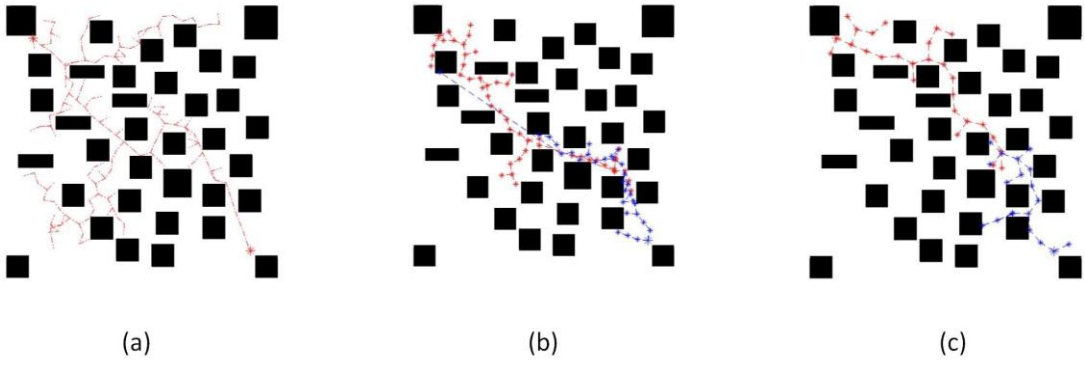


图 5.16 复杂环境下搜索树分析

从图中可以清楚的看到单向 RRT 充分的采样了可行域环境，充分展示了随机树算法

相较于其他经典算法的快速特性。基本双向 RRT 算法的采样数量明显较单向 RRT 减少，因为双向搜索树互相作为信息参照能够更加了解相关的环境信息。而使用了一定贪婪策略的同时更新 RRT 算法产生了更加少的节点，两棵搜索树之间产生了较少的交错情况，更加高效地使得两课搜索连接到了一起。

5.3.3 实验总结

快速搜索随机树（RRT）算法有用基于采样与经典算法的特性，通过对于可行域的快速采样建立可行路径的网络，并且向目标域拓展可行节点。

简单 RRT 算法由于是单向搜索算法，所以需要较大的代价向目标方向进行大规模的采样建立路径的工作，因此规划的时间会较长，并且其随机生成的节点数量也会较多。

基本双向 RRT 搜索树由于采用了双搜索树的搜索方式，并且有两棵搜索树进行对于可行空间的采样，其搜索效率基本是单向 RRT 算法的一倍，由于是两棵搜索树交替相向生长保证了其可以利用互相的采样信息从而避免了不必要的冗余采样。

同时更新双向 RRT 算法使两棵搜索树同时向一个中间随机点生长。采用了类似贪婪的策略，使得两棵树之间最近的点不断地相向生长，并且在两点遇到障碍物或者局部陷阱的时候，再利用随机算法使搜索树采样其余可行域拓展可行区域网络从而保证了搜索能够拜托局部陷阱等不可行的情况。避免了一定程度上的不必要的随机采样，提出的算法采样数量相较于基本双向 RRT 算法是下降的。该算法中，寻找两棵搜索树中最近节点是时间消耗较大的，但由于其具有贪婪的性质，同时更新双向 RRT 算法的规划时间还是得到了一定程度上的提高。

本部分的一系列的实验验证了基于采样算法相较于经典算法在规划时间上的优势，更加体现出了其在复杂环境中规划的速度与效率的优势。同时，也验证了双向搜索树算法的有效性，并且规划时间大致是单向搜索树规划时间的一半。最后也验证了提出的算法的可行性，验证了其效率与路径规划与轨迹生成等功能的有效性。

第六章　总结与展望

6.1 总结

本次毕业论文任务让我深入理解了轨迹生成的相关工作与应用场景，对于基于采样的规划算法 RRT 进行了相应的改进。在实际运用中，能够使规划算法在较短的时间内利用较少的计算资源是十分重要的，当然，每一个算法都有自己的局限性，个人以为，只有结合具体的应用场景，结合不同的规划算法才能达到相应的最佳规划效果。

本项目中也存在很多可以改进的地方，由于基础的 RRT 算法没有带入路径长度的信息，所以无法保证得到的路径是最短的，即路径长度得不到相应的最短优化，后期的工作可以将路径长度，环境干扰强度等相关的信息考虑进去进行一个综合的优化，得到适合具体任务的较优轨迹。

同时，在之前规划路径的基础上，再次规划时可以利用已有的规划信息，实现计算资源的节约并且能较快的得到可行的路径，这个在实时性要求较高的环境中是十分重要的。

6.2 展望

关于研究方向，希望能够在以后的研究中更好地应用；希望能够在将来把算法应用到 3D 环境中，并且对于动态环境能够更好地进行适应与轨迹生成的工作。RRT 相关的衍生算法需要不断地进行创新从而满足各种不同条件的规划需求。

参 考 文 献

[1] Fuentes-Pacheco, J., Ruiz-Ascencio, J. & Rendón-Mancha, J.M. Artificial Intelligence Review[M], Netherlands: Springer Press, 2015. 43~55.

[2] W. Gouda, W. Gomaa and T. Ogawa. Vision based SLAM for humanoid robots: A survey[A]. in 2013 Second International Japan-Egypt Conference on Electronics, Communications and Computers[C]. Cairo, Egypt: IEEE, 2013: 170~175.

[3] Leishman, J Gordon. The Breguet-Richet quad-rotor helicopter of 1907[J]. Vertiflite, 2010, 47(3): 58~60

[4] M. Belkheiri, A. Rabhi, A. E. Hajjaji, C. Pegard. Different linearization control techniques for a quadrotor system[A]. in 2nd. Int. Conf. on Communications, Computing and Control Applications (CCCA)[C]. Turin, Italy: ACM, 2012: 1~6.

[5] B. Zhu and W. Huo. Trajectory linearization control for a quadrotor helicopter[A]. in 8th IEEE Int. Conf. on Control and Automation (ICCA)[C]. Kunming, China: IEEE, 2010: 34~39.

[6] A. B. Junaid, A. Konoiko, Y. Zweiri. Autonomous Wireless Self-Charging for Multi-Rotor Unmanned Aerial Vehicles[J]. Energies, 2017, 10(6):803.

[7] Z. N. Liu, X. Q. Liu, L. J. Yang, D. Leo and H. W. Zhao. an Autonomous Dock and Battery Swapping System for Multirotor UAV[J]. AIAA J, 2018, 17(2): 10~15.

[8] R. Mahony, V. Kumar, and P. Corke. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor[J]. IEEE Rob. Autom. Mag, 2012, 19(3): 20~32.

[9] A. H. Pedersen. Test and Modelling of Four Rotor Helicopter Rotors[D]. Denmark: DTU, 2006.

[10] Tomas Lozano-Perez. Spatial Planning: A Configuration Space Approach[J], IEEE Transactions on Computers, 1983, C(32): 108~120.

[11] Jeffrey R. Hartline, Ran Libeskind-Hadas. The Computational Complexity of Motion Planning[J], SIAM Review, 2003, 45(3): 543~557.

[12] Lim Chee Wang, Lim Ser Yong, Marcelo H. Ang Jr. Hybrid global path planning and local navigation implemented on a mobile robot in indoor environment[A]. in Proceedings of the 2002 IEEE International Symposium on Intelligent Control[C]. Vancouver, Canada: IEEE, 2002: 821~826.

[13] Andrea Signifredi, Bombini Luca, Alessandro Coati. A general-purpose approach for global and local path planning combination[A]. in IEEE 18th Internal Conference on Intelligent Transportation Systems[C]. Las Palmas: IEEE, 2015: 996-1001.

[14] Sabo, Chelsea and Cohen, Kelly. Fuzzy Logic Unmanned Air Vehicle Motion Planning[J]. Hindawi Publishing Corp, 2012, 1(13): 13~20.

[15] Mahdi Fakoor, Amirreza Kosari, Mohsen Jafarzadeh. Humanoid robot path planning with fuzzy Markov decision processes[J]. Journal of Applied Research and Technology, 2016, 14(5): 300~310.

[16] Deneubourg, J, Clip, P. and Camazine. Ants, buses and robots-self-organization of transportation systems[J]. Proc. From Perception to Action, 1994, 1(2): 12~23.

[17] Kevin van Hecke, Guido de Croon, Laurens van der Maaten, Daniel Hennes, Dario Izzo. Persistent self-supervised learning principle: from stereo to monocular vision for obstacle avoidance[J]. International Journal of Micro Air Vehicles, 2016, 10(2): 821~827.

[18] Keil, J. M., and Sack, J, R. Minimum decomposition of polygonal objects[J]. Comp. Geom. 1985, 8(3): 197~216.

[19] Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots[J]. Int. J. of Robotics Research, 1986, 5(1): 90~99.

[20] David Wilson, J H Davenport, M England, R J Bradford. A "piano movers" problem reformulated[A]. in SYNASC 2013: 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. Timisoara, Romania: IEEE, 2013: 53~60.

[21] J. Canny. A Voronoi method for the piano-movers problem[A]. in Proc. IEEE Int. Conf. Robot. Automation[C]. Canada: IEEE, 1985: 530~535.

[22] O. Takahashi and R. J. Schilling. Motion planning in a plane using generalized Voronoi diagrams[J]. IEEE Trans. Robot. Automation, 1989, 5(2): 143~150.

[23] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning[J]. in Proc. IEEE Int. Conf. Robot. Automation, 1996, 1(1): 113~120.

[24] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planner[J]. Proc. IEEE Int. Conf. Robot. Automation, 1999, 2(10): 1018~1023.

[25] S. Zheng, D. Hsu, J. Tingting, H. Kurniawati, and J. H. Reif. Narrow passage sampling for probabilistic roadmap planning[J]. IEEE Trans. Robot, 2005, 21(6): 1105~1115.

[26] D. Hsu and S. Zheng. Adaptively combining multiple sampling strategies for probabilistic roadmap planning. Proc. IEEE Conf. Robot., Autom. Mechatron, 2004, 1(2): 774~779.

[27] S. Thomas, M. Morales, T. Xinyu, and N. M. Amato. Biasing samplers to improve motion planning performance. Proc. IEEE Int. Conf. Robot. Autom, 2007, 10(2): 1625~1630.

[28] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain[J]. Proc. IEEE ICRA, 2005, 1(1): 3856~3861.

[29] L. Jaillet, A. Yershova, S. M. La Valle, and T. Simeon. Adaptive tuning of the sampling domain for dynamic-domain RRTs[J]. Proc. EEE/RSJ Int. Conf. IROS, 2005, 1(1): 2851~2856.

[30] C. Peng and S. M. LaValle. Reducing metric sensitivity in randomized trajectory design[J]. Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst, 2001, 1(1): 43~48.

[31] D. Hsu and S. Zheng. Adaptively combining multiple sampling strategies for probabilistic roadmap planning[J]. Proc. IEEE Conf. Robot., Autom. Mechatron., 2004, 2(1): 774~779.

[32] J. Denny and N. M. Amato. The toggle local planner for sampling-based motion planning[J]. Proc. IEEE ICRA, 2012, 1(1): 1779~1786.

[33] C. L. Nielsen and E. E. Kavraki. A two-level fuzzy PRM for manipulation planning[J]. Proc. IEEE/RSJ Int. Conf. IROS, 2000, 3(1): 1716~1721.

[34] R. Bohlin and E. E. Kavraki. Path planning using lazy PRM[J]. Proc. IEEE ICRA, 2000, 1(1): 521~528.

[35] J. Denny, K. Shi, and N. M. Amato. Lazy toggle PRM: A single-query approach to motion planning[J]. Proc. IEEE ICRA, 2013, 1(1): 2407~2414.

[36] “International Federation of Robotics Report”, https://ifr.org/, accessed：2019.6.1.

致　谢

转瞬之间四年的大学生活已经进入尾声。对于这四年充满感激之情。

首先，感谢我的毕设指导老师翟象平副教授。此次的毕业设计和毕业论文都是在翟老师的指导和督促下完成的。感谢翟老师从我大二作为我的本科生导师开始，对我学习和生活给予的指导和关怀。当我的研究工作遇到困难而停滞不前时，是翟老师帮助我树立信心，为我指明解决问题的方向；当我取得阶段性成果而开始沾沾自喜时，是翟老师督促我对课题进一步深入，帮助我不断地提高进步。

我要感谢帮助过我的同学，感谢给予我帮助和关怀学长学姐们，无论是在我的学习上还是在各个方面对我的指导；感谢和我一起共同努力的学弟和同学，我们一起完成一个又一个难题。

其次感谢田睿同学对于我学习与生活的关心，能够让我在快乐美好的时光中不断地探索与学习新的知识，在我烦躁的时间里能够静下心来研究课题。互相竞争并且共同进步，十分感谢她在我身边的陪伴。

感谢我自己，能够一直不放弃自己，始终坚信自己的理想，无论怎么样都坚持地走下去。感谢我自己，给予我的所有力量和信心。

最后感谢我的父母，感谢他们为我的成长倾注心血，感谢他们不求回报的默默支持。他们的热情和支持是我继续前行的动力，对他们的回报将是我铭记在心的奋斗目标，爸爸妈妈我爱你们！

附　录

1. 单向快速随机搜索树算法代码

```
function [result, time, vertices, edge] = classicRRT(map, q_start, q_goal, p,
delta_q)

addpath(genpath('util'));
addpath(genpath('utilEnv'));

k = 10000;

edge = double.empty(0,2);
vertices = double.empty(0,2);
vertices = q_start;

[mapHeight, mapWidth] = size(map);

if map(int32(q_start(2)),int32(q_start(1))) == 0
    disp('Start Point in obstacle region.\n failed.');
    return;
end

if map(int32(q_goal(2)),int32(q_goal(1))) == 0
    disp('Start Point in obstacle region.\n failed.');
    return;
end
```

```
tic;
for i = 1:k % iteration limit is k
    % generate random point in map
    % probability p, using q_goal as q_rand
    if QGoalSelectionProbability < p
        q_rand = q_goal;
    else
        q_rand = QRandGeneration(mapHeight,mapWidth);
    end

    % select nearest point in SET vertices
    q_near = QNearSelection(vertices,q_rand);

    % generate new point in map
    q_new = QNewGeneration(q_near, q_rand, delta_q);

    check_q_new = int32(floor(q_new));

    if all(isnan(check_q_new))
        continue;
    end

    % check q_new out of boundary
    if check_q_new(1) < 1 || check_q_new(1) > mapWidth || check_q_new(2) < 1 ||
check_q_new(2) > mapHeight
        continue;
    end
```

```
    % new point is in free space
    if map(check_q_new(2), check_q_new(1)) == 1
        % edge {q_new,q_near} is in free space
        if QNewQNearEdgeInFreeSpace(map, q_near, q_new)
            % add vertices and edge
            vertices = [vertices; q_new];
            edge = [edge; q_near; q_new];
            % q_goal is reachable, return
            if QGoalTouched(map,q_new,q_near,q_goal)
                [~,touchpoint] = QGoalTouched(map,q_new,q_near,q_goal);
                % q_near touched
                if touchpoint == 1
                    vertices = [vertices; q_goal];
                    edge = [edge; q_near; q_goal];
                % q_new touched
                else
                    vertices = [vertices; q_goal];
                    edge = [edge; q_new; q_goal];
                end
                time = toc;
                result = 1;
                return;
            end
        end
    end
end
result = 0;
```

```
end
```

2. 双向快速搜索随机树代码

```
function [result, time, verticesSrc, edgesSrc, verticesDst, edgesDst] = biRRT(map, q_start,
q_goal, delta_q)

addpath(genpath('utilFunc'));
addpath(genpath('utilMap'));

k = 10000;

verticesSrc = double.empty(0,2);
verticesDst = double.empty(0,2);
edgesSrc = double.empty(0,2);
edgesDst = double.empty(0,2);

% initilize data arrays
verticesSrc = [verticesSrc; q_start];
verticesDst = [verticesDst, q_goal];

[mapHeight, mapWidth] = size(map);

% initialize tree
v1 = verticesSrc;
e1 = edgesSrc;
v2 = verticesDst;
e2 = edgesDst;

tic;
for i = 1:k
    q_rand = QRandGeneration(mapHeight, mapWidth);
    q_near1 = QnearstFind(v1, q_rand);
    q_new1 = QnewGeneration(q_near1, q_rand, delta_q);

    if QnewQnearInFreeSpace(map, q_near1, q_new1)
        % T1 extend successfully
        v1 = [v1; q_new1];
        e1 = [e1; q_near1; q_new1];
        q_near2 = QnearstFind(v2, q_new1);
        q_new2 = QnewGeneration(q_new1, q_near2, delta_q);
        if QnewQnearInFreeSpace(map, q_near2, q_new1)
```

```
            % T2 extend succesfully
            v2 = [v2; q_new2];
            e2 = [e2; q_near2; q_new2];
            if FinalEdgeInFreeSpace(map, q_new1, q_new2)
                time = toc;
                if v1(1,:) == q_start
                    verticesSrc = v1;
                    edgesSrc = e1;
                    verticesDst = v2;
                    edgesDst = e2;
                else
                    verticesSrc = v2;
                    edgesSrc = e2;
                    verticesDst = v1;
                    edgesDst = e1;
                end
                result = 1;
                return;
            end
        end
    end
    % swap
    v_temp = v1;
    e_temp = e1;
    v1 = v2;
    e1 = e2;
    v2 = v_temp;
    e2 = e_temp;
end
time = toc;
result = 0;
end
```

3. 同时更新双向快速搜索随机树

```
function [result, time, verticesSrc, verticesDst, edgesSrc, edgesDst] = simulbiRRT(map,
q_start, q_goal, p, delta_q)
addpath(genpath('utilEnv'));
addpath(genpath('utilFunc'));
addpath(genpath('utilMap'));

k = 10000;
```

```
verticesSrc = double.empty(0,2);
verticesDst = double.empty(0,2);
edgesSrc = double.empty(0,2);
edgesDst = double.empty(0,2);

verticesSrc = [verticesSrc; q_start];
verticesDst = [verticesDst, q_goal];

[mapHeight, mapWidth] = size(map);

rand_flag = 0;

tic;
for i = 1:k
    if p > QGoalSelectionProbability
        rand_flag = 1;
    end

    % use flag to judge whether to connect or randomize

    if rand_flag
        % use randomized point to draw out of local minima
        RandSrc = QRandGeneration(mapHeight, mapWidth);
        %         RandDst = QRandGeneration(mapHeight, mapWidth);
        RandDst = RandSrc;

        NearSrc = QnearstFind(verticesSrc, RandSrc);
        NearDst = QnearstFind(verticesDst, RandDst);
    end

    if rand_flag == 0
        [CloseInSrc, CloseInDst] = FindTwoClosestPoint(verticesSrc, verticesDst);

        if Dist(CloseInSrc, CloseInDst) < delta_q
            if FinalEdgeInFreeSpace(map, CloseInSrc, CloseInDst)
                verticesSrc = [verticesSrc; CloseInDst];
                edgesSrc = [edgesSrc; CloseInSrc; CloseInDst];
                result = 1;
                time = toc;
                return;
```

```
            end
        end

        RandSrc = CloseInDst;
        RandDst = CloseInSrc;

        NearSrc = CloseInSrc;
        NearDst = CloseInDst;
    end

    NewSrc = QnewGeneration(NearSrc, RandSrc, delta_q);
    NewDst = QnewGeneration(NearDst, RandDst, delta_q);

    CheckSrc = int32(NewSrc);
    CheckDst = int32(NewDst);

    % out of map, continue.
    if CheckSrc(1) < 1 || CheckSrc(1) > mapWidth || CheckSrc(2) < 1 || CheckSrc(2) >
mapHeight ...
            ||CheckDst(1) < 1 || CheckDst(1) > mapWidth || CheckDst(2) < 1 || CheckDst(2) >
mapHeight
        continue;
    end

    if map(CheckSrc(2), CheckSrc(1)) && ...
            map(CheckDst(2), CheckDst(1))


        if  ~QnewQnearInFreeSpace(map, NearSrc, NewSrc)
            if QnewQnearInFreeSpace(map, NearDst, NewDst)
                verticesDst = [verticesDst; NewDst];
                edgesDst = [edgesDst; NearDst; NewDst];
            end
        end

        if ~QnewQnearInFreeSpace(map, NearDst, NewDst)
            if QnewQnearInFreeSpace(map, NearSrc, NewSrc)
                verticesSrc = [verticesSrc; NewSrc];
                edgesSrc = [edgesSrc; NearSrc; NewSrc];
            end
        end
```

```
        if QnewQnearInFreeSpace(map, NearSrc, NewSrc)
            if QnewQnearInFreeSpace(map, NearDst, NewDst)
                % both extend

                verticesSrc = [verticesSrc; NewSrc];
                edgesSrc = [edgesSrc; NearSrc; NewSrc];
                verticesDst = [verticesDst; NewDst];
                edgesDst = [edgesDst; NearDst; NewDst];

                if Dist(NewSrc, NewDst) < 50
                    if  FinalEdgeInFreeSpace(map, NewSrc, NewDst)
                        verticesSrc = [verticesSrc; NewDst];
                        edgesSrc = [edgesSrc; NewSrc; NewDst];
                        result = 1;
                        time = toc;
                        return;
                    end
                end
            end
        end
    end
end
result = 0;
time = toc;
end


function d = Dist(NewSrc, NewDst)
d = sum((NewSrc - NewDst).^2);
end
```