审定成绩：_____

重庆邮电大学
毕业设计（论文）

中文题目	基于 RflySim 的四旋翼无人机轨迹跟踪控制仿真系统开发
英文题目	**Development of Trajectory Tracking Control Simulation System for Quadrotor UAV based on RflySim**
学院名称	自动化学院/工业互联网学院
学生姓名	彭彦泽
专　　业	机器人工程
班　　级	**08912002**
学　　号	**2020212944**
指导教师	李永福　　教授
答辩组负责人	王颋　　教授

二〇二四 年 六 月

重庆邮电大学教务处制

自动化学院本科毕业设计(论文)诚信承诺书

本人郑重承诺：

我向学院呈交的论文《基于 RflySim 的四旋翼无人机轨迹跟踪控制仿真系统开发》，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明并致谢。本人完全意识到本声明的法律结果由本人承担。

年级　2020

专业　机器人工程专业

班级　08912002

承诺人签名　彭彦泽

2024 年 5 月 25 日

学位论文版权使用授权书

本人完全了解重庆邮电大学有权保留、使用学位论文纸质版和电子版的规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。本人授权重庆邮电大学可以公布本学位论文的全部或部分内容，可编入有关数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等复制手段保存、汇编本学位论文。

（注：保密的学位论文在解密后适用本授权书。）

学生签名：彭彦泽　　指导老师签名：

日期：　年　月　日　　日期：　年　月　日

摘要

四旋翼无人机由于其优秀的灵活性与机动能力，备受国内外研究的关注。随着控制技术的改进和成本的下降，它们的使用扩展到许多领域，包括航空摄影，森林火灾监测，河流监测，环境监测，搜索和救援，基础设施检查，产品交付，娱乐和无人机赛车等。由于四旋翼无人机动态特性复杂，对飞行员的操作技能要求较高，因此，开发一个高效、稳定的轨迹跟踪控制系统对于提高无人机的自主性和安全性至关重要。

本文主要研究工作如下：

四旋翼飞行器的动力学建模过程首先涉及对其结构的介绍，继而深入探讨其垂直升降、航向调整、俯仰及翻滚四种核心运动方式。最后，通过应用牛顿-欧拉动力学原理，推导出了位置动力学方程和姿态动力学方程，从而构建出完整的四旋翼飞行器动力学模型。

在四旋翼无人机的路径规划策略中，首先对各种搜索算法进行了论述和比较，决定采用 A*算法作为路径规划的基础。利用 PX4 控制器的 Offboard 模式特性，建立了 Simulink 模型以实现与 Rflysim 的交互。最终，通过单线激光雷达测距和 UDP 通信，实现了基于 Python 的 A*算法与 Simulink 模块间的通信，以此达成四旋翼无人机的路径规划目标。

在设计四旋翼无人机的控制器时，以 PID 控制理论为指导，分别构建了位置控制器和姿态控制器。接下来，对控制器进行仿真结果分析，通过微调 PID 参数，找到了一组理想的参数设置，并绘制了相应的伯德图以展示控制系统性能。

关键词：X 型四旋翼无人机；PID 控制；A*算法；Rflysim

Abstract

Quadrotor UAVs have received much attention in domestic and international research due to their excellent agility and maneuverability. With the improvement of control technology and cost reduction, their use extends to many non-military applications, including aerial photography, forest fire monitoring, river monitoring, environmental monitoring, search and rescue, infrastructure inspection, product delivery, recreation and UAV racing. Due to the complex dynamic characteristics of quadrotor UAVs and the high demands on the pilot's operational skills, the development of an efficient and stable trajectory tracking control system is essential to improve the autonomy and safety of UAVs. The main research work of this thesis is as follows:

The dynamics modeling process of the quadrotor first involves the introduction of its structure, followed by an in-depth discussion of its four core motions: vertical lift, heading adjustment, pitch and roll. Finally, by applying the Newton-Euler dynamics principle, the position dynamics equations and attitude dynamics equations are derived so as to construct a complete quadrotor dynamics model.

In the path planning strategy of the quadrotor UAV, various search algorithms are firstly discussed and compared, and it is decided to adopt the A* algorithm as the basis of path planning. Using the Offboard mode feature of the PX4 controller, a Simulink model was built to realize the interaction with Rflysim. Eventually, the communication between the A* algorithm and the Simulink module were realized through single-line LIDAR ranging and UDP communication as a way to reach the goal of path planning for the quadcopter UAV.

In designing the controller of the quadrotor UAV, the position controller and attitude controller are constructed separately with PID control theory as a guide. Next, the simulation results of the controllers are analyzed, and a set of ideal parameter settings are found by fine-tuning the PID parameters, and the corresponding Bird diagrams are plotted to demonstrate the control system performance.

Keywords: X-type quadcopter UAV; PID control; A* algorithm; Rflysim

目录

第 1 章 引言

1.1 研究背景和意义

无人机是无人驾驶的飞行器，其操作依赖于远程控制设备和内置的自动控制系统，不配备飞行员。装备有自动驾驶仪和程序控制装置，地面控制站、舰船或母机上的操作员通过雷达等设备进行监控、定位、遥控、遥测和数据传输。无人机在民用领域已有广泛的应用，涵盖了农业、能源、环保、土地、海洋、水利等多个行业。已发展成熟的用途包括航拍、农业保护和电力设施检查；而像紧急救援和物流配送等新应用正在逐渐成熟[1]。

无人飞行器主要分为两类：固定翼和旋翼。固定翼无人机以其出色的稳定性和滑翔能力，常用于灾害预防和环境监测。相比之下，旋翼无人机由导航、通信、动力、控制系统及微型传感器、微型处理器等组成，能通过调整旋翼转速执行复杂的飞行任务，如定点停留、俯仰倾斜和垂直起降，其灵活性使它能胜任固定翼无法完成的任务，因此在各个领域都有广阔的应用前景，吸引了全球众多研究者投身于旋翼无人机的研究。

1907 年，法国学者 Louis Breguet 研制出世界上首架四旋翼无人机，但由于缺乏有效的控制技术，飞行稳定性不足，控制效果不尽如人意，未能实现真正的自主飞行[2]。

随着新材料、微电子机械系统、传感器技术和飞行控制技术的进步，多旋翼无人机的体积和重量显著减轻，结构和稳定性得到大幅提升，从而推动了多旋翼无人机的快速发展。

近年来四旋翼飞行器依据尺寸和操控方式，通常可归为三类[3]：遥控型，如美国 Dragan flyer 公司的 Dragan flyer III；小型无人机，如宾夕法尼亚大学的 HXM-4 和瑞士洛桑联邦理工学院的 OS4 系列；微型无人机，代表性的有斯坦福大学的 Helicopter、麻省理工学院航天实验室的视觉技术驱动的自主飞行器，以及苏黎世联邦理工学院的无人机。这些实验室研发的四旋翼飞行器具备高超的机动性和灵活性，能完成许多精度要求高且复杂的任务。在商业领域，法国的 Parrot 公司以其内置 GPS 的四轴飞行器备受推崇，还有 Lockheed Martin 公司的 Indago 四轴飞

行器，以及大疆公司推出的 M200 系列和“御”Mavic 系列飞行器，这些产品均获得了市场的广泛认可和消费者的热烈追捧。

针对四旋翼飞行器非线性、强耦合、欠驱动以及对外部干扰敏感等特点，设计姿态稳定性强、轨迹跟踪精度高和抗干扰性强的控制策略是一项艰巨的任务[4]。通过采用模拟仿真这一手段，我首先构建无人机系统的数学模型，然后围绕这个模型进行深入的研发和实验验证，最后的目标是将理论成果实际应用到真实的无人机操作中。相较于传统的实验方法，仿真开发与测试拥有显著优势，如只需基本的计算机设备，便可在室内环境中进行，成本低廉且不受场地限制。这样，我们可以在虚拟空间内精细调整和优化四旋翼无人机的轨迹跟踪控制算法，从而规避实际飞行中的风险和成本开支。通过这种方式，仿真平台成为了一种理想的试验场，为提升无人机性能提供了安全且经济的解决方案。因此，开发一个高效、稳定的轨迹跟踪控制仿真系统对于提高无人机的自主性和安全性至关重要。

1.2 国内外研究现状

1.2.1 国外研究现状

在轨迹规划和飞行控制研究方面，美国国防部高级研究计划局的快速轻量级自主项目旨在创建一种新的导航、感知、规划和控制算法，使无人机将在没有外部通信或 GPS 的情况下，以高达 20 米/秒的速度自主感知和机动穿越未知的室内/室外环境[5]。瑞士苏黎世大学机器人与感知实验室团队针对四旋翼在狭窄间隙中自主飞行问题，考虑几何、动态和感知约束规划了包括接近段和穿越段的穿框轨迹，实现了只利用机载传感和计算，在没有事先知道间隙位置的情况下，通过狭窄的间隙解决并实现自主攻击飞行的工作[6]。文献[7]中提出了一个鲁棒和有效的四旋翼运动规划系统的快速飞行在三维复杂的环境利用 B 样条的凸包特性，结合来自欧几里得度量场的梯度信息和动态约束，提高了轨迹的平滑度和清晰度。

在四旋翼的飞行控制算法研究方面，苏黎世大学团队对高机动轨迹的跟踪控制进行了深入研究。文献[8]提出了一种新颖的体速率控制器和一种迭代推力混合控制方案，分别在不需要学习的情况下提高了四旋翼的轨迹跟踪性能和减小了偏航控制误差。文献[9]提出一种基于时变系统模型的 LQR 控制方法，实现和评估一

个状态依赖的 LQR 能够在提供通用性和性能的同时进行机载计算。文献[10]证明了受线性转子阻力作用的四旋翼动力学模型在其位置和方向上是平坦的，利用这个性质大大降低了轨迹跟踪误差。文献[11][11]实现了 NMPC 对动态不可行轨迹的跟踪控制，文献[12]提出了一种新的混合自适应 NMPC 算法 L1-NMPC，它可以在线学习模型的不确定性并立即进行补偿，以最小的计算开销大大提高非自适应基线的性能，可以准确地跟踪高度敏捷的赛车轨迹，最高速度为 70 公里/小时，相对于非自适应 NMPC 基线提供约 50%的跟踪性能改进。文献[13]提出了一个新的控制问题公式，允许在线更新状态和控制权重，仿真验证了与采用固定权重的 NMPC 标准解决方案相比，执行轨迹的精确度最多可提高 70%。

1.2.2 国内研究现状

国防科技大学致力于四旋翼飞行器原型的研发，深入探究了其飞行特性与动力学基础，构建了相应的数学模型。之后，他们运用自抗扰控制[14]和反步控制[15]等多种控制策略来开发控制器，并通过仿真测试验证了这些控制算法的实用性和效率。

上海交通大学的学者利用 Newton-Euler 方程建立了详尽的四旋翼飞行器数学模型。他们分别采用了传统的 PID 控制和反步控制技术来设计控制系统，并在 Simulink 环境下构建仿真模型进行实验。通过对实验结果的比较分析，他们凸显了反步控制的优势，并成功地将其应用到实际操作中[16]。

浙江大学的 FAST-Lab 团队提出了创新性的策略，聚焦于机器人集群的行为以及机器人的独立路径规划问题。机器人集群通过实时动作捕捉，实现随机移动障碍物下安全平稳的轨迹移动，在仅靠机载摄像头、机载计算芯片资源和传感器的情况下，实现了在野外复杂树林环境下感知周围障碍物、定位自身位置、实时规划飞行路径等系列操作[17]。

唐嘉宁等人针对传统 A*算法在无人机路径规划时效率低下、路径点存在大量冗余，且路径转折较多的缺点，在《基于改进 A*算法的无人机路径规划研究》中提出一种基于双向机制的改进 A*算法[18]。刘栩粼等人针对四旋翼无人机在轨迹跟踪控制过程中存在的不确定性和外部扰动，在《基于前馈补偿的 PD 四旋翼无人机

轨迹跟踪控制》中提出了一种基于前馈补偿的 PD 轨迹跟踪控制。这些算法在提高无人机的稳定性和控制精度方面具有很大的潜力[19]。

1.3 主要内容和工作安排

本篇论文一共分为 5 章，内容结构和每章节安排如下：

第 1 章是引言，本章节首先阐述了研究课题的背景及其重要性，接着综述了当前国际及国内在四旋翼无人机的跟踪控制与轨迹规划领域的研究进展。

第 2 章是四旋翼无人机动力学建模，本章通过合理的假设，选择合适的坐标系，并运用牛顿-欧拉方程，详细构建了四旋翼无人机的动力学模型。

第 3 章是四旋翼无人机路径规划，搭建 Simulink 模块实现与 Rflysim 的通信，利用单线激光雷达测距和 UDP 通信，选取 A*算法实现四旋翼无人机的路径规划。

第 4 章是四旋翼无人机轨迹跟踪控制，本章基于 PID 控制理论，分别设计了位置和姿态控制器，并通过仿真分析验证了控制效果。

第 5 章是总结与展望，第一部分对全文进行工作总结，第二部分对后续研究工作展望。

第 2 章 四旋翼无人机动力学建模

2.1 四旋翼无人机的结构及工作原理

2.1.1 四旋翼无人机的结构

四旋翼无人机具备卓越的动态操控能力，有六个维度的运动能力，涵盖前后的线性移动、左右的横向移动、垂直的上下升降、偏航旋转、俯仰倾斜以及滚转动作。作为四输入六输出的欠驱动系统，它有四个动力输入，却产生六个状态输出。耦合特性存在于其六个自由度之中，通过对四个旋翼的转速调控，可以操纵这六个自由度的动作。

如图 2.1 所示，四旋翼无人机的构造呈现 X 型布局，四个电机分别位于机体臂的四个末端，通过电机驱动旋翼旋转来实现飞行操作。

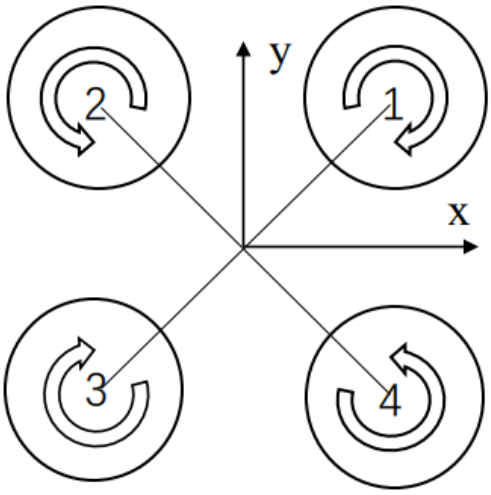


图 2.1 四旋翼无人机结构图

2.1.2 四旋翼无人机的工作原理

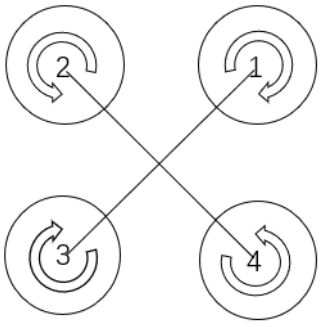
1）垂直升降：如图 2.2(a)所示，电机 1 和电机 3 按照顺时针方向转动，电机 2 和电机 4 则以相同的逆时针角速度旋转。四个电机的协同工作使得四个旋翼旋转，进而产生升力，使飞行器能够垂直起降。在电机旋转过程中，空气阻力会作用于

旋翼，产生扭力矩。电机 1 和 3 的顺时针旋转与电机 2 和 4 的逆时针旋转产生相抵消的扭力矩，从而消除对机体的影响。

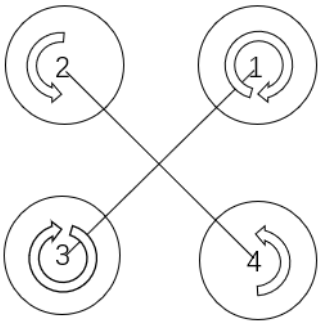
2）偏航运动：在四旋翼无人机中，每个旋翼产生的推力会形成对飞机的扭矩，导致飞机围绕垂直于飞行平面的中心轴旋转，即偏航。如图 2.2(b)所示，通过增大对角线上的电机 1 和 3 的转速，同时减小相邻的电机 2 和 4 的转速，保持整体升力不变，但电机 1 和 3 产生的反扭力矩大于电机 2 和 4，促使飞机沿逆时针方向偏航。

3）俯仰运动：飞行器绕 y 轴转动，使机身前后倾斜，即为俯仰。如图 2.2(c)所示，提升电机 1 和 2 的转速，降低电机 3 和 4 的转速，且变化量相等。这将增加旋翼 1 和 2 的升力，减少旋翼 3 和 4 的升力，形成的不平衡力矩使得机身绕 y 轴旋转。反之，降低电机 1 和 2 的转速，提高电机 3 和 4 的转速，则机身会沿 y 轴向相反方向旋转，实现俯仰运动。在此过程中，垂直升力会产生水平分量，从而使无人机能前后移动。

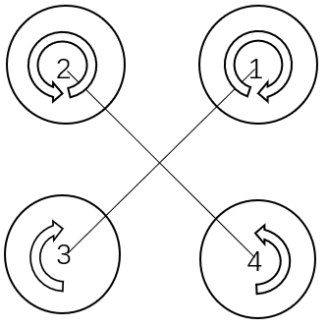
4）滚转运动：飞行器沿 x 轴转动，机身左右倾斜，即为滚转。如图 2.2(d)所示，提高电机 1 和 4 的转速，降低电机 2 和 3 的转速，且变化量相等。这将增强旋翼 1 和 4 的升力，减弱旋翼 2 和 3 的升力，产生不平衡力矩使机身绕 x 轴旋转。相反地，降低电机 1 和 4 的转速，增加电机 2 和 3 的转速，机身将沿 x 轴向另一侧旋转，完成滚转运动。同样地，滚转时，垂直升力也会倾斜产生水平分量，帮助无人机进行左右移动。



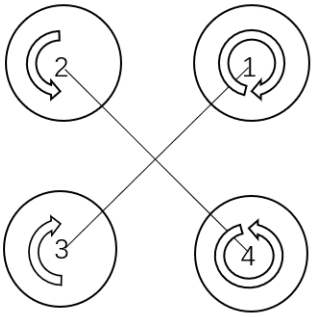
（a）垂直升降



（b）偏航运动



（c）俯仰运动



（d）滚转运动

图 2.2 四旋翼无人机运动状态

2.2 四旋翼无人机动力学建模

2.2.1 坐标系

地面坐标系(X,Y,Z)：坐标原点为地面起飞点，Z 轴垂直地面向上，X 轴为东方向，Y 轴为北方向。

机体坐标系(x,y,z)：坐标原点为无人机的重心，z 轴垂直机体向上，x 轴位于相邻机臂的角平分线上，y 轴由右手螺旋定则确定。

将机体坐标系沿 x 轴方向转动滚转角 ϕ，然后沿 y 轴转动俯仰角 θ，最后沿 z 轴转动偏航角 ψ，就可以得到地面坐标系。

R_x、R_y、R_z 的旋转矩阵如下所示：

$$R_x=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi & \cos\phi\end{bmatrix}\quad R_y=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}\quad R_z=\begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

机体坐标系变换到地面坐标系的变换矩阵 R：

$$R(\phi,\theta,\psi)=\begin{bmatrix}\cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi-\sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi+\sin\psi\sin\phi\\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi+\cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi-\cos\psi\sin\phi\\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi\end{bmatrix}$$

2.2.2 四旋翼无人机动力学模型

针对四旋翼无人机的构建材质及飞行条件，假定其为坚硬无变形的对称刚体，同时排除陀螺效应的影响。在低空低速的飞行场景下，气动效应微乎其微，因此可以省略不计，同时忽略无人机因弹性而产生的形变和振动现象。

本节运用牛顿欧拉方程推导四旋翼无人机的动力学模型。

根据牛顿第二定律有：

$$F = m\frac{dv}{dt}, M = \frac{dL}{dt} \tag{2.1}$$

式中，F——作用在四旋翼无人机上的合力

m——四旋翼无人机的质量

v——四旋翼无人机的质心速度

M——合力矩

L——合角动量

无人机所受的升力定义为T，并且垂直于机体平面向上，表示为$F = \begin{bmatrix} 0 & 0 & T \end{bmatrix}^T$，则在地面坐标系，

$$\begin{aligned} F_e &= RF \\ &= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \end{aligned} \tag{2.2}$$

即，$F_e = T\begin{bmatrix} \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}$，$T = \sum_{i=1}^{4} b\omega_i^2$

根据牛顿第二定律得，

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{F_e}{m} - g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.3}$$

综上所述得到地面坐标系下无人机的位置动力学方程：

$$\begin{cases} \ddot{x} = b\sum_{i=1}^{4}\omega_i^2\left(\cos\psi\sin\theta\cos\phi+\sin\psi\sin\phi\right)/m \\ \ddot{y} = b\sum_{i=1}^{4}\omega_i^2\left(\sin\psi\sin\theta\cos\phi-\cos\psi\sin\phi\right)/m \\ \ddot{z} = b\sum_{i=1}^{4}\omega_i^2\left(\cos\theta\cos\phi\right)/m-g \end{cases} \tag{2.4}$$

根据刚体的欧拉方程，得：

$$\sum M = I\dot{W}+W\times(IW) \tag{2.5}$$

其中，

$$I=\begin{bmatrix} I_x & & \\ & I_y & \\ & & I_z \end{bmatrix}, W\times(IW)=\begin{bmatrix} i & j & k \\ p & q & r \\ I_x p & I_y q & I_z r \end{bmatrix}=\begin{bmatrix} rq\left(I_z-I_y\right) \\ pr\left(I_x-I_z\right) \\ pq\left(I_y-I_x\right) \end{bmatrix}$$

式中，p、q、r——机体坐标系的滚转角、俯仰角和偏航角所对应角速度

I ——惯性张量

M ——合力矩

通过整理得：

$$\begin{cases} M_x = I_x\dot{p}+\left(I_z-I_y\right)qr \\ M_y = I_y\dot{q}+\left(I_x-I_z\right)pr \\ M_z = I_z\dot{r}+\left(I_y-I_x\right)pq \end{cases} \tag{2.6}$$

进而得到：

$$\begin{bmatrix} M_{Tx} \\ M_{Ty} \\ M_{Tz} \end{bmatrix}=\begin{bmatrix} l\left(F_4-F_2\right) \\ l\left(F_3-F_1\right) \\ -M_{D1}+M_{D2}-M_{D3}+M_{D4} \end{bmatrix}=\begin{bmatrix} lb\left(\Omega_4^2-\Omega_2^2\right) \\ lb\left(\Omega_3^2-\Omega_1^2\right) \\ d\left(-\Omega_1^2+\Omega_2^2-\Omega_3^2+\Omega_4^2\right) \end{bmatrix} \tag{2.7}$$

式中，b——无人机旋翼的升力系数

Ω_i——旋翼转速

d ——反扭矩系数

l ——四旋翼无人机臂长

根据欧拉方程得，角速度运动方程：

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi}\sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta \\ -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{bmatrix} \tag{2.8}$$

在姿态角旋转小角度时，$\sin\phi \approx \sin\theta \approx \sin\psi \approx 0, \cos\phi \approx \cos\theta \approx \cos\psi \approx 1$，上式可以简化为：

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.9}$$

定义：

$$\begin{cases} U_1 = \sum_{i=1}^{4} F_i = b\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \\ U_2 = F_4 - F_2 = b\left(\Omega_4^2 - \Omega_2^2\right) \\ U_3 = F_3 - F_1 = b\left(\Omega_3^2 - \Omega_1^2\right) \\ U_4 = b\left(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2\right) \end{cases} \tag{2.10}$$

根据以上推导，得到四旋翼无人机的动力学模型如下：

$$\begin{cases} \ddot{x} = \left(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi\right)U_1 / m \\ \ddot{y} = \left(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi\right)U_1 / m \\ \ddot{z} = \left(\cos\theta\cos\phi\right)U_1 / m - g \\ \ddot{\phi} = \left(\dfrac{I_y - I_z}{I_x}\right)\dot{\theta}\dot{\psi} + \dfrac{l}{I_x}U_2 \\ \ddot{\theta} = \left(\dfrac{I_z - I_x}{I_y}\right)\dot{\phi}\dot{\psi} + \dfrac{l}{I_y}U_3 \\ \ddot{\psi} = \left(\dfrac{I_x - I_y}{I_z}\right)\dot{\phi}\dot{\theta} + \dfrac{U_4}{I_z} \end{cases} \tag{2.11}$$

2.3 本章小结

本章节专注于探讨 X 型无人直升机的动力学特性，特别关注其核心组件和操作机制。首先，我详细描绘了这种飞行器的基本构造，其中四个电动机均匀分布在机体臂的基座上，它们通过精确控制旋翼的旋转，实现了精准的航空操作。我们深入剖析了其关键的运动模式：垂直起降、偏航操控、俯仰机动以及滚转响应，

这些动作的实现都依赖于精确的力学原理。进一步地，我们摒弃了低空低速环境下气动效应的微小影响，以及对机体弹性性质的考虑，因为这些因素在我们的研究中被简化处理。利用牛顿-欧拉方程这一经典力学工具，我们系统地建立了四旋翼无人直升机的动力学模型，涵盖了位置变化的动力学描述和姿态变化的动力学方程，这为我们后续的控制策略设计和性能分析提供了坚实的理论基础。

第 3 章 四旋翼无人机路径规划

3.1 常用搜索算法

深度优先探索：这是一种策略，它按照深度优先的原则进行搜索。从起始点开始，首先深入探索当前节点的所有未访问邻居，然后再回溯以探索其他分支，犹如一棵树的逐层递进。在实施过程中，每个节点需要保存其前驱节点信息，以便于路径逆向追踪，形成一条路径。

Prim 算法：Prim 算法专注于寻找图中节点间的最小成本连接路径。它逐步构建最小生成树，每次从已选取节点中添加与其相连且成本最低的新节点。即使存在不同路径成本的差异，如地形起伏，Prim 算法依然适用。

贪心最佳搜索：当预先得知每个节点至终点的直接距离时，贪心最佳搜索会利用这些信息，优先处理距离终点最近的节点。虽然效率高，但在遇到障碍时，未必能确保找到最短路径。

A*算法：结合了贪心最佳搜索与 Dijkstra 算法的优势，A*算法是一种启发式搜索策略。它基于启发式函数评估每个节点的优先级，同时考虑节点到起点的成本和预测到终点的成本。常见于游戏中的 NPC 或网络游戏中 BOT 的路径规划。

A*算法在状态空间中搜索，先评估并选取最佳路径，随后从该路径继续搜索，直至目标。它减少了无效搜索，提升了效率，并且在满足评估函数单调性的条件下，保证能找到最短路径。

公式(3.1)表示 A*算法的估算函数：

$$f(\mathrm{n}) = g(n) + h(n) \tag{3.1}$$

其中，$f(n)$是从初始状态经由状态n到目标状态的代价估计，$g(n)$是在状态空间中从初始状态到状态n的实际代价，$h(n)$是从状态n到目标状态的最佳路径的估计代价。启发函数$h(n)$的选取：曼哈顿距离，切比雪夫距离，欧几里得距离等。

在路径查找问题中，状态代表图中的节点，而代价对应于节点间的距离。为了确保找到最短路径，关键在于估价函数$f(n)$的设计，特别是如何通过$h(n)$来衡量从当前状态到目标的状态距离。通常，估价函数的选择可以分为以下三种情形：

1）当$h(n)<d(n)$实际到达目标状态的距离较大时，搜索过程会涉及大量节点，其探索领域广泛，导致效率偏低，但却能确保找到最优解决方案。

2）若$h(n)=d(n)$，即预估距离恰好等于实际最短距离，搜索策略会沿着这条预设的高效路线进行，此时的搜索效率将达到顶峰。

3）反之，$h(n)>d(n)$距离估计偏保守，导致搜索节点减少，范围受限，效率提升，然而无法确保找到全局最优解。

特别地，当估价函数值$h(n)=0$时，A*算法退化为经典的 Dijkstra 方法，确保找到绝对的最短路径。

若估价函数$h(n)$始终小于从初始状态到当前状态的实际成本，A*算法保证能寻找到最短路径。估价函数值越低，A*的扩展行为越频繁，速度自然减缓。

当估价函数$h(n)$恰好等于实际移动成本时，A*将严格遵循最佳路径，不再拓展非最优路径，从而实现极高的运行速度，在特定场景下，A*的表现堪称卓越。

如果估价函数$h(n)$有时超过实际成本，虽然不能确保找到最短路径，但可能会带来更快的搜索速度。

特别地，当$h(n)$相对于实际成本有显著优势，A*则近乎变成以$h(n)$为主的贪婪优先搜索策略。

3.2Rflysim 与 Simulink 通信

无人机的 Offboard 功能代表了一种控制策略，是由 PX4 官方控制器所支持的，它倾向于将实际飞行任务下放给机载或地面主控计算机（核心控制器），允许系统全面负责飞机的动态管理，包括速度调整、空间定位以及姿态控制，从而释放操作员对底层执行层面的关注，专注于高级的视觉导航和群集算法创新。

如图 3.1 所示，通过监听并解析端口的 UDP 数据包，CopterSim 作为中间介质，将信息转化为结构体的数据单元，随后进行深入的数据处理并呈现出来。

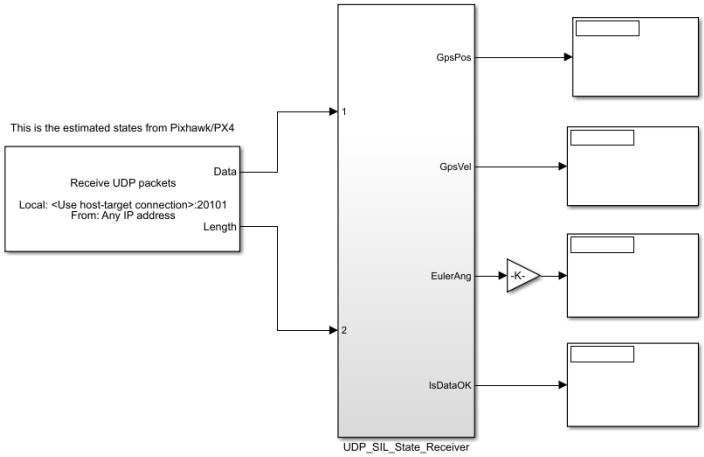


图 3.1 UDP 接收模块

如图 3.2 所示，首先，通过移动 VeZ 滑块，模拟油门提升过中心位置，能够精确控制飞机在 *Z* 轴上的垂直起升；接着，调节 VeX 模拟俯仰杆操作，实现了飞机在前后方向的精准移动，调节 VeY 滑块模拟横滚杆操作，实现了飞机在前后和左右方向的精准移动；最后，移动 Yaw 滑块模拟偏航杆操作，可调整飞机的偏航速率，确保其转向的精确性。

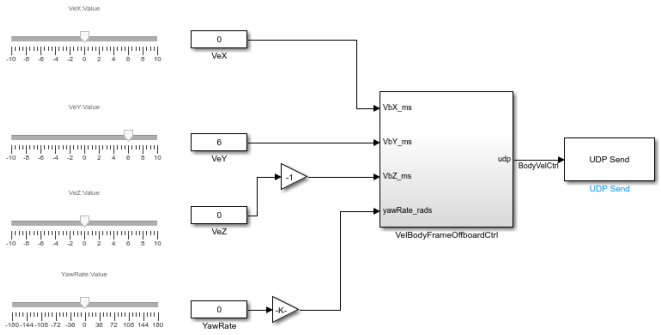


图 3.2 UDP 发送模块

Simulink 的 Offboard 模块：

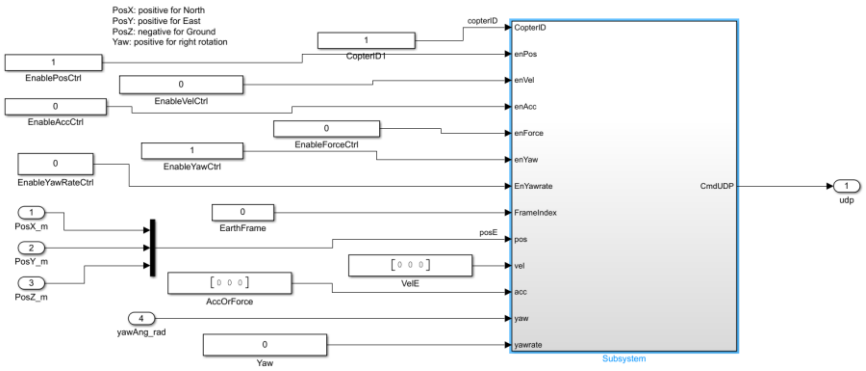


图 3.3 Offboard 模块

CopterSim 收到来自 Simulink 的 UDP 消息后，会发送 mavlink 消息 SET_POSITION_TARGET_LOCAL_NED 来实现飞机的速度、位置、与角度等控制。

3.3 激光雷达测距

如图 3.4 所示，激光雷达的运作始于发射器，它周期性地激发激光发生器，生成激光脉冲。这些脉冲经过激光调制器的调控，在光束控制器的指引下确定发射方向和激光束的数量。接着，发射光学系统将激光导向目标物体。扫描机制以恒定的速度旋转，完成二维平面的扫描，实时生成平面图像数据。

接收端，激光接收系统中的光电探测器捕获从物体反弹回来的激光，生成接收信号。随后，这些信号在信息处理系统中经历放大和数模转换的过程。通过信息处理模块的复杂计算，可以解析出物体表面的几何特征、物理特性等详细信息，从而构建出物体的三维模型。

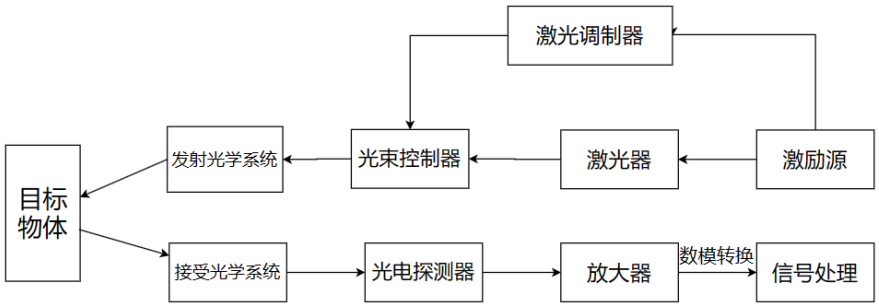


图 3.4 激光雷达原理图

本文选用的是单线激光雷达，这种雷达发射的激光束仅限于一条直线。单线激光雷达以其快速的扫描速率，适于实时的环境感应。对比多线激光雷达，它在角频率响应和灵敏度上更具优势。单线激光雷达的构造相对简洁，操作简便，且具备较高的可靠性。此外，它的成本是所有激光雷达类型中最低的，这使得它成为经济型选择。

首先通过运行 python 文件加载传感器设置，向 Rflysim3D 发送取图请求，并用共享内存图像转发，得到如图所示 3.5 雷达监测图像。

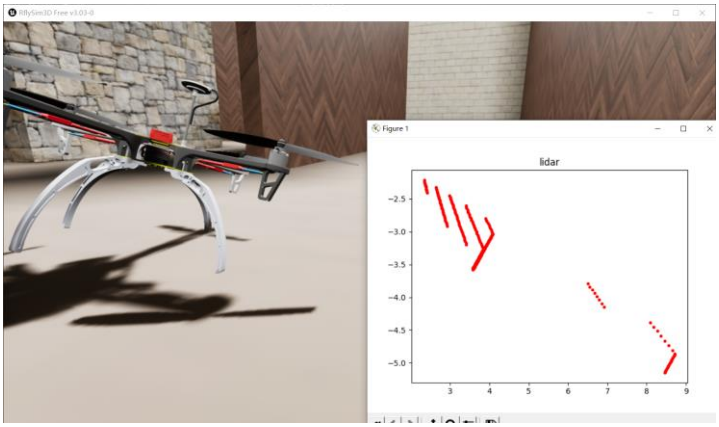


图 3.5 雷达检测

3.4 A*算法路径规划

算法使用场景如图 3.6 所示激光雷达扫描地图：

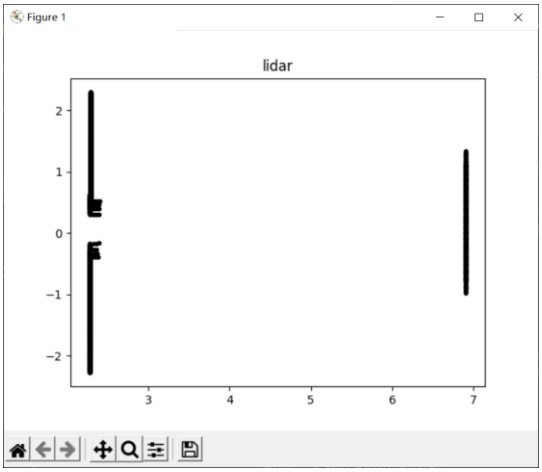


图 3.6 激光雷达扫描地图

Simulink 调用 UDP 模块实现与 python 通信如图 3.7 所示：

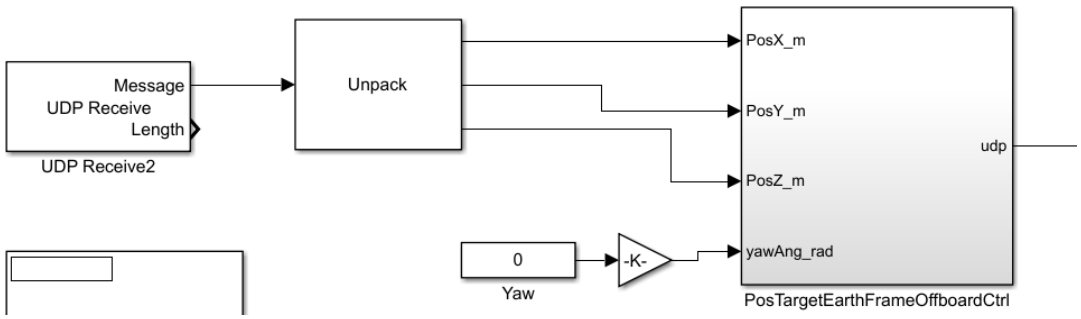


图 3.7 Simulink 与 python 通信

A*算法的代价函数 $g(n)=n$， $h(n)$ 选取曼哈顿距离，得到如图 3.8 所示路径规划效果图。

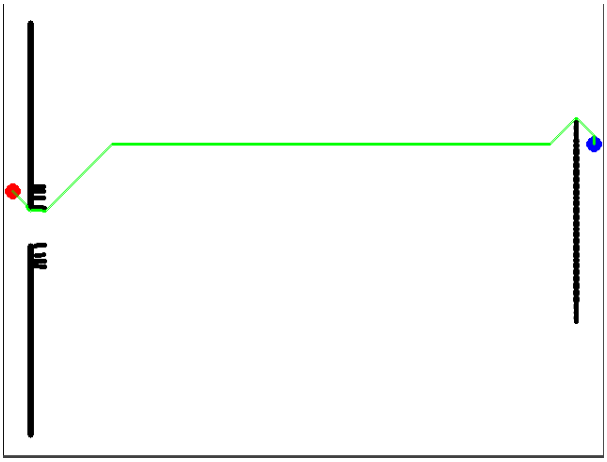


图 3.8 A* 路径规划

3.5 本章小结

本章针对四旋翼无人机路径规划，首先介绍了常见的搜索算法并比较，最终选取 A*算法进行路径规划。利用 PX4 官方控制器提供的 Offboard 模式功能，搭建 Simulink 模块实现与 Rflysim 的通信。最后利用单线激光雷达测距和 UDP 通信，将 A*算法 python 与 Simulink 模块通信，实现四旋翼无人机的路径规划。

第 4 章 四旋翼无人机轨迹跟踪控制

4.1 位姿控制

四旋翼飞行器属于欠驱动控制系统，其四个输入整体升力和三个轴向的扭矩，需调控六个输出位置坐标和姿态角度。如图 4.1 所示，设计此类无人机的控制器通常运用双层环结构，内部环针对飞行器的姿态角进行精确操纵，而外部环则负责管理飞行器的空间位置。这种内外部环的控制架构使得多旋翼飞行器能够执行各种飞行任务，包括垂直起降、定点悬停以及横向移动等不同飞行模式。

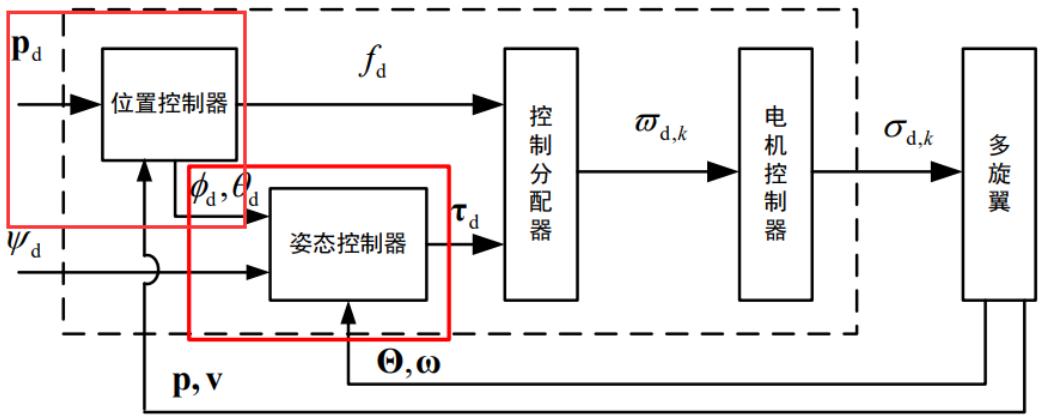


图 4.1 底层控制框架

Matlab/Simulink 位姿控制器仿真模块如 4.2 所示：

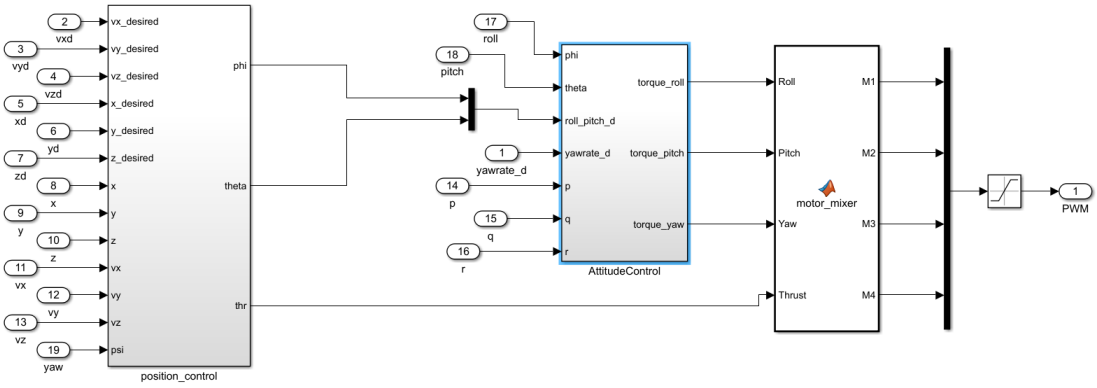


图 4.2 位姿控制器

4.2 四旋翼无人机模型

Matlab/Simulink 四旋翼无人机模型如 4.3 所示：

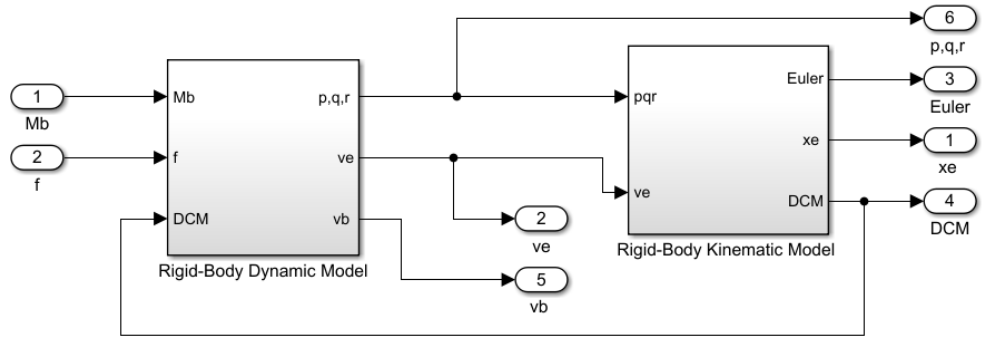


图 4.3 四旋翼无人机模型

Matlab/Simulink 位置动力学模型如图 4.4 所示：

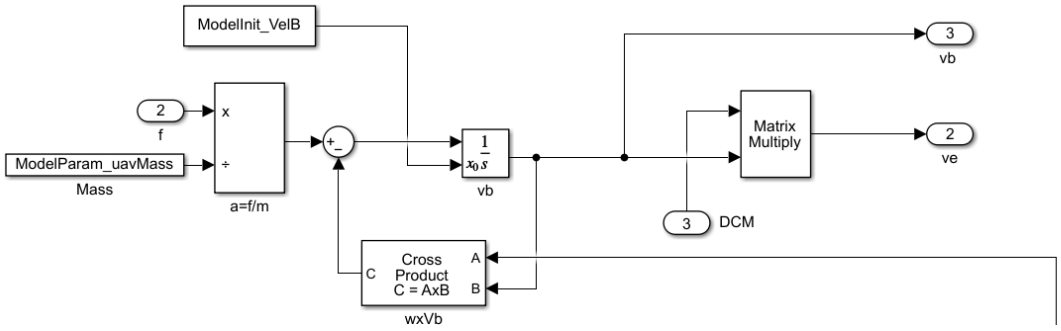


图 4.4 位置动力学模型

Matlab/Simulink 姿态动力学模型如图 4.5 所示：

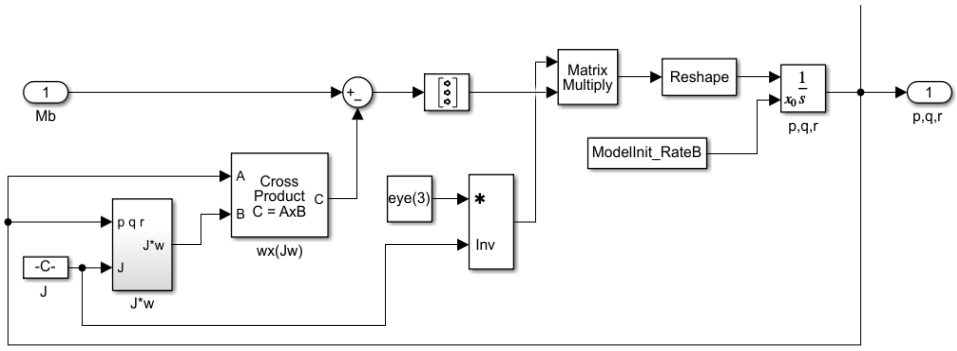


图 4.5 姿态动力学模型

Matlab/Simulink 位置运动学模型如图 4.6 所示：

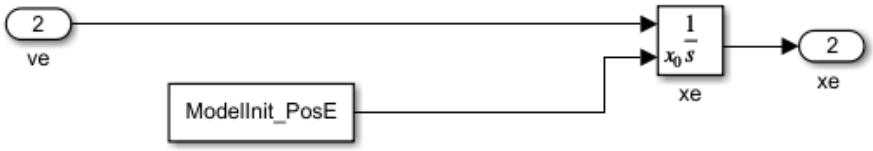


图 4.6 位置运动学模型

Matlab/Simulink 姿态运动学模型如图 4.7 所示：

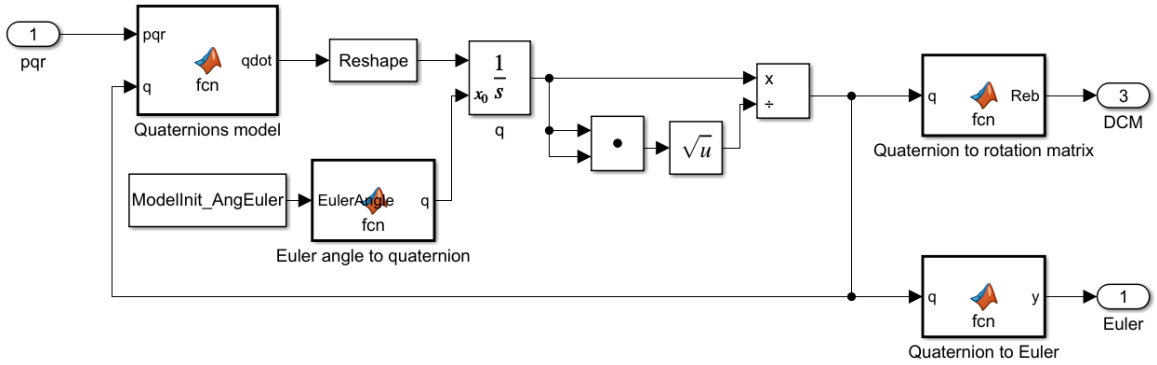


图 4.7 姿态运动学模型

4.3 位置控制

4.3.1 位置控制器原理

四轴飞行器的位置控制系统运用了 PID 控制策略，以调整飞行器在三维空间中的定位。该控制器依据当前坐标与目标坐标间的偏差，结合比例、积分和微分三项来生成控制指令。

1）比例部分（Proportional）：

比例项的响应基于当前误差的大小，其作用在于快速反应。控制指令的计算方法是将位置误差乘以比例系数（K_p），形成比例输出。比例系数的设定需依据

系统的动态行为和稳定性需求。增加比例系数可提升响应速度，但也可能引起过冲或振荡；反之，降低比例系数可能导致响应过慢，影响目标位置的准确追踪。

2）积分部分（Integral）：

积分项用于抵消持续的偏差，有助于消除由比例控制产生的静态误差。它是位置误差随时间积分的累积，再乘以积分系数（K_i）。选择合适的积分系数至关重要，过高可能导致系统过度校正及不稳定。

3）微分部分（Derivative）：

微分项依据位置误差的变化速率来计算控制指令，对快速变动的误差起到平滑和稳定的效果。这通过计算位置误差的导数并乘以微分系数（K_d）来实现，微分系数的大小影响了响应的平滑性和对误差变化的敏感性。

这三项比例、积分和微分的加权总和构成了 PID 控制器的输出控制信号。这个信号转化为电机的运行指令，通过调整旋翼的转速来操纵无人机的位置。PID 控制器的参数需依据具体应用和系统需求进行定制。通过不断地测试和调整控制增益，能够优化控制器性能，确保无人机能精确到达并稳定在期望位置，同时具备抗干扰能力。

4.3.2 位置控制器设计

位置控制器的设计使用位置环和速度环双环控制，内环为速度环，外环为位置环，如图 4.8 所示。

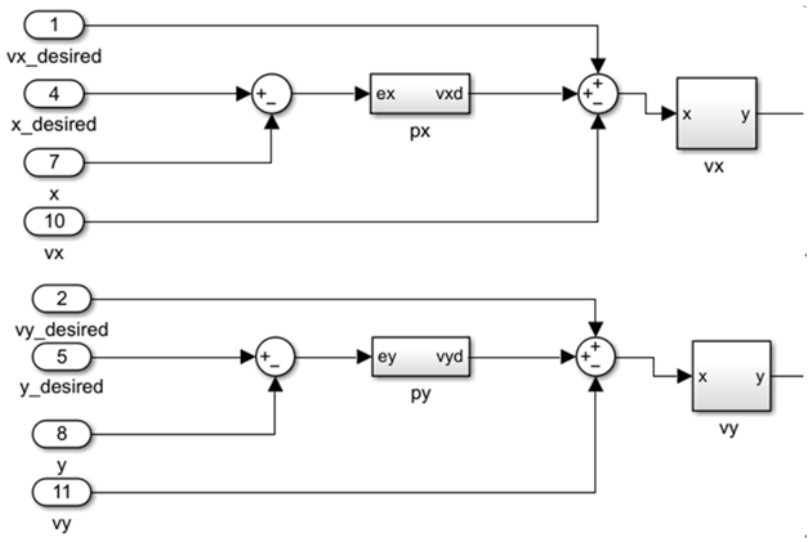


图 4.8 位置环和速度环控制

位置环采用 P 控制，位置期望值减实际值作为输入量，经过比例系数 Kp 增益，并限制在一定范围防止调节过度，然后输出量作为速度环的输入量，如图 4.9 所示。

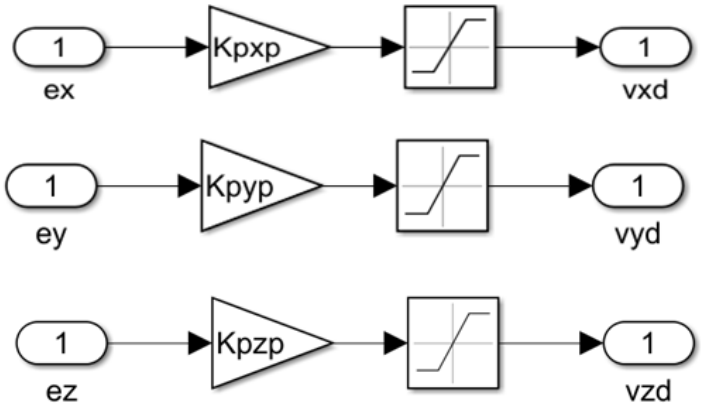


图 4.9 位置环 P 控制

速度环采用 PID 控制如图 4.10 所示，输入量经过比例、微分、积分三者的加权求和，输出所需加速度。

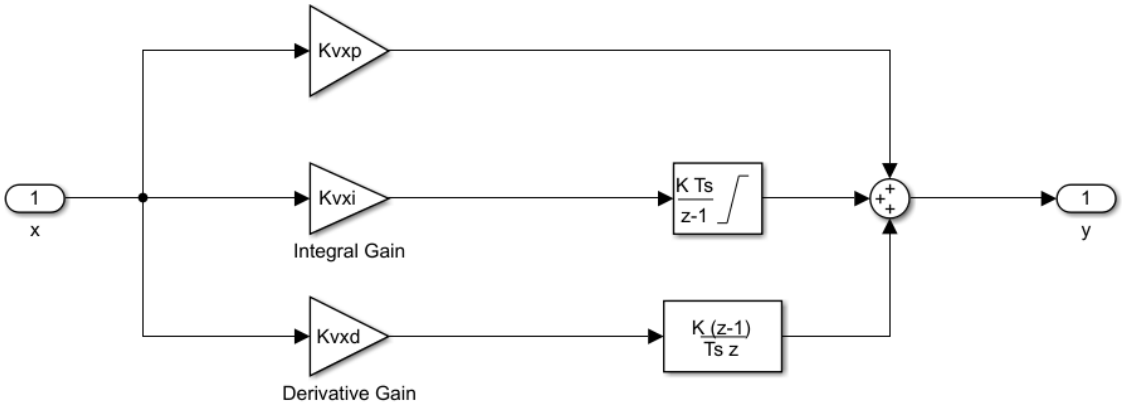


图 4.10 速度环 PID 控制

如图 4.11 所示，根据水平方向的加速度和偏航角计算，得出滚转角和俯仰角，并上下限饱和限幅防止无人机的过度抖动。

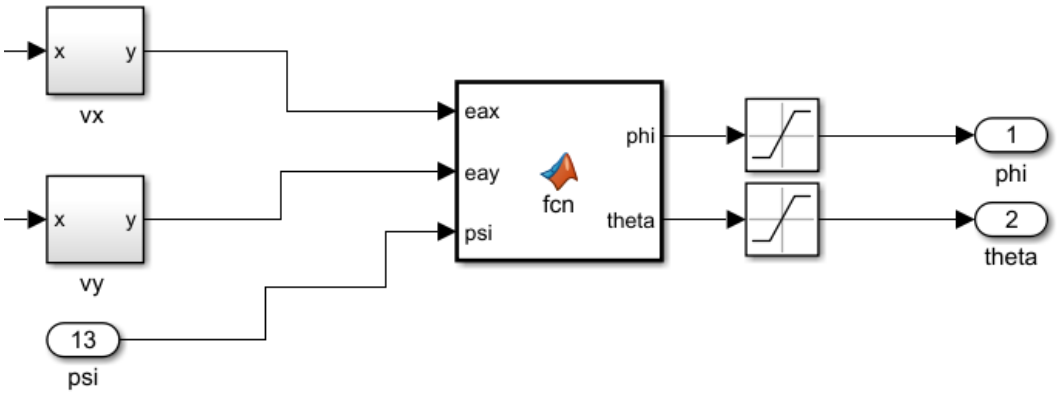


图 4.11 姿态角计算

Matlab Function 实现滚转角和俯仰角的计算，如图 4.12 所示。

```
function [phi, theta] = fcn(eax, eay, psi)
g = 9.8;
phi = (-sin(psi)*eax + cos(psi)*eay)/g;
theta = (-cos(psi)*eax - sin(psi)*eay)/g;
```

图 4.12 滚转角和俯仰角

4.3.3 位置控制器仿真

位置闭环控制如图 4.13 所示，位置控制器输出脉冲信号 PWM，通过调节 PWM 占空比的方式控制电机的转速，从而控制四旋翼无人机的位置、速度、姿态角、角速度，被控对象四旋翼无人机模型的输出量返回到位置控制器的输入端，形成位置的闭环控制。

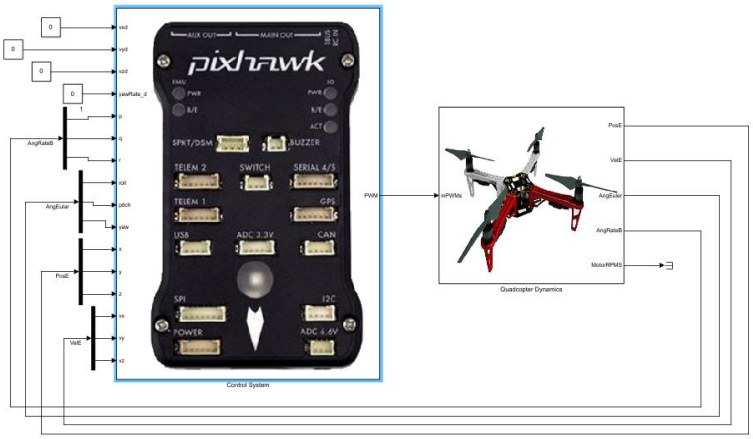


图 4.13 位置闭环控制

PID 控制调参时，先调节内环的速度环，在调节外环的位置环，避免调参混乱无法得到理想的曲线。速度环参数调节，以比例系数为例，速度的期望值为阶跃输入，令 K_p 分别设置为 1.5、2.0、2.5 得到如图 4.14 所示曲线图。

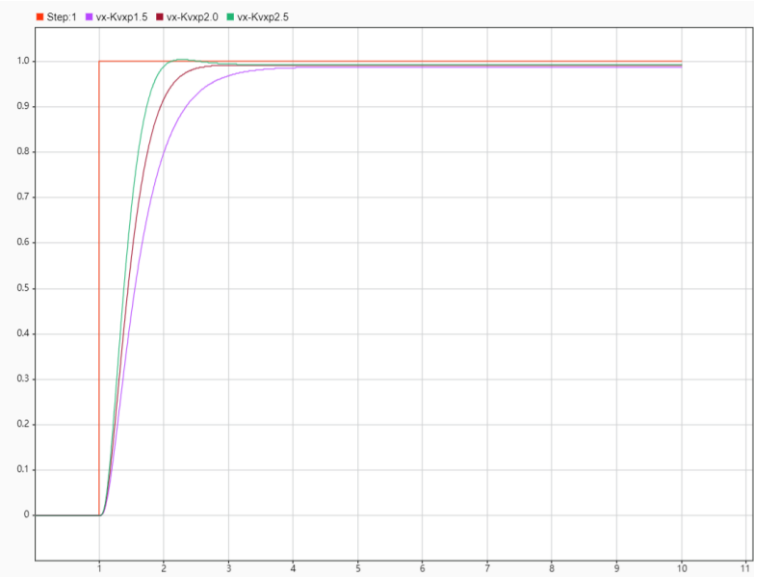


图 4.14 速度环 PID 控制调参

位置环参数调节，位置期望值为阶跃输入，令 K_p 分别设置为 0.8、1.0 或 1.2，得到位置环 P 控制曲线图，如图 4.15 所示。

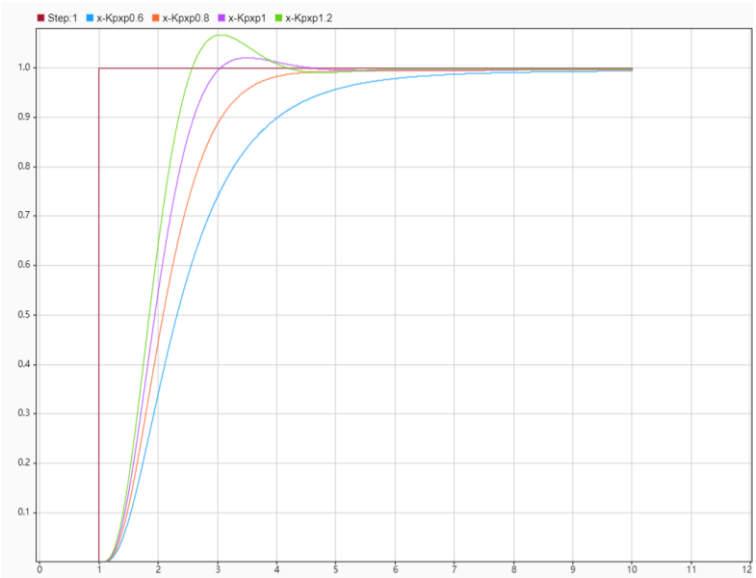


图 4.15 位置环 P 控制调参

Matlab 绘制位置期望值和位置实际值的 Bode 图如 4.16 所示，Gain Margin：15.3dB，At frequency：3.97rad/s：Phase Margin:65.6deg，At frequency：1.04rad/s。

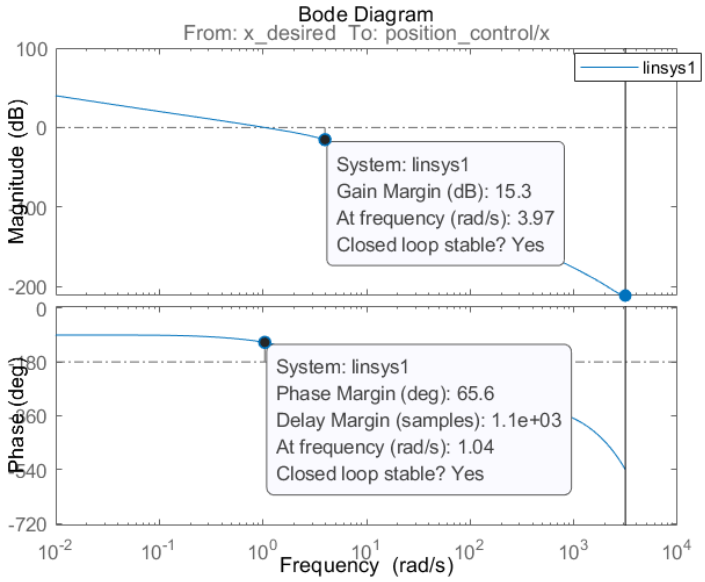


图 4.16 位置控制器 Bode 图

经过 PID 参数调节过程，得到速度环 PID 参数如表 4.1 所示：

表 4.1 位置控制器速度环 PID 参数

速度环参数	v_x	v_y	v_z
K_p	2.5	2.5	0.45
K_i	0.4	0.4	0.01
K_d	0.01	0.01	0.005

位置环 PID 参数如表 4.2 所示：

表 4.2 位置控制器位置环 PID 参数

位置环参数	x	y	z
K_p	1.0	1.0	4.0

4.4 姿态控制

4.4.1 姿态控制器原理

四旋翼无人飞行器的姿态控制单元是无人机操作系统的核心组件，其功能机制涉及多层面解析。首先，四旋翼的飞行奥秘在于四个旋转旋翼协同工作，它们通过调控转速和角度，精准地生成升力和推动力，从而控制无人机的飞行动态。在这样的系统中，精确控制旋翼的转速和方向是确保姿态稳定的基石。

姿态控制器的设计构建在严谨的 PID 调控策略上。PID 控制器作为一种普遍采纳的控制策略，通过对比设定的目标姿态与实时感知的无人机状态，利用比例（P）、积分（I）和微分（D）这三个参数，动态调整控制信号，确保飞行的平稳性。其工作原理是，比例控制侧重于即时误差的修正，反应迅速但可能产生超调；积分控制则旨在弥补比例控制下的长期误差，使系统趋于理想姿态，但过度可能导致系统不稳定；而微分控制则通过预测未来状态变化，减少超调并优化响应速度，然而过度依赖微分增益可能导致噪声干扰或系统不稳定。

1）比例控制：作为 PID 的核心，它通过比较目标姿态与实际测量值的差异，利用比例增益（K_p）进行实时调整，以迅速纠正偏差。然而，过高的比例增益可能导致系统过度反应和稳态误差。

2）积分控制：积分控制主要作用于消除比例控制中的残留误差，通过累计姿态误差，借助积分增益（K_i）进行补偿，以提高飞行器的跟踪精度。然而，不当的积分参数可能导致系统振荡。

3）微分控制：微分控制则关注速度和动态响应，通过计算姿态变化率，利用微分增益（K_d）进行预判，有助于减小超调并提升稳定性。然而，微分控制的使用需谨慎，以防引入额外的噪声或诱发系统不稳定。

4.4.2 姿态控制器设计

姿态控制器的整体设计如图 4.17 所示，输入为各期望值和实际值，输出姿态角的力矩，然后经过控制分配输出四旋翼无人机的四个电机的力矩。

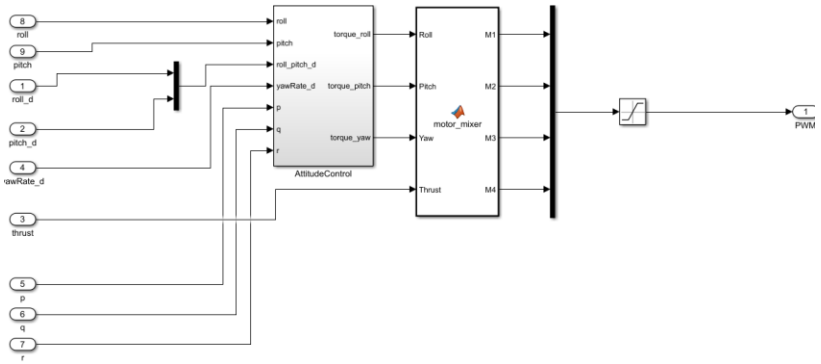


图 4.17 姿态控制器

姿态控制器的滚转控制和俯仰控制如图 4.18 所示，采用姿态环和角速度环双环控制。

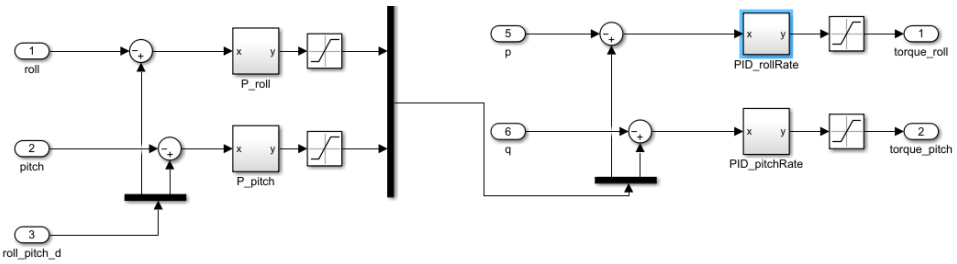


图 4.18 滚转控制和俯仰控制

姿态控制器的偏航控制如图 4.19 所示，采用角速度环 PID 控制。

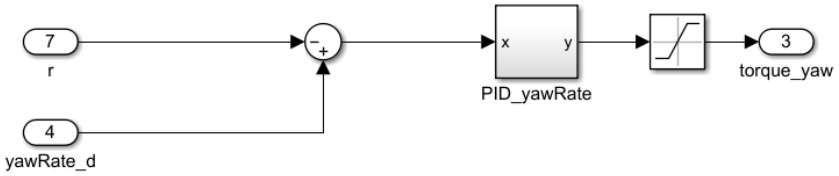


图 4.19 偏航控制

姿态环如图 4.20 所示采用 P 控制，姿态角的期望值减实际值，经过 Kp 增益并限幅，然后作为角速度环的输入。

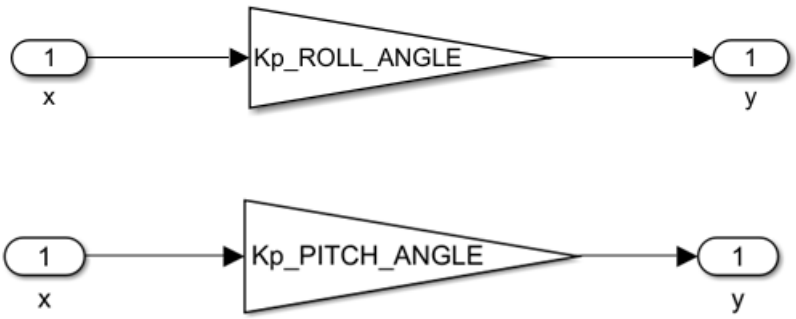


图 4.20 姿态角 P 控制

角速度环如图 4.21 所示采用 PID 控制，输入量经过比例、积分、微分三者的加权求和，输出角加速度。

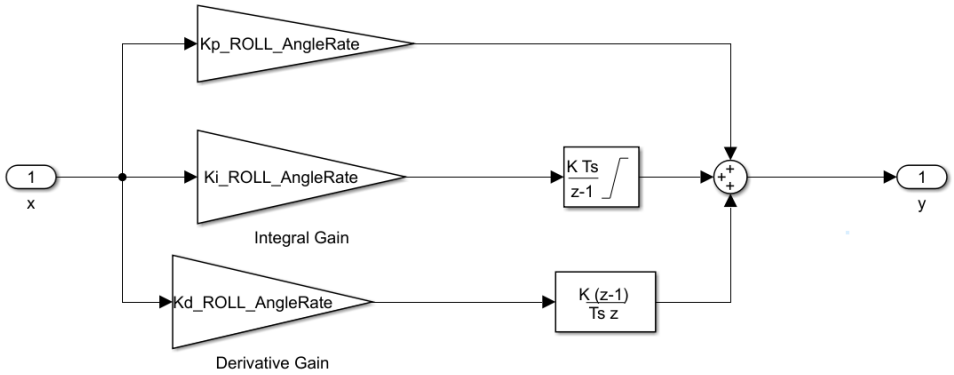


图 4.21 角速度 PID 控制

电机的力矩分配如图 4.22 所示，姿态角和无人机所需的升力作为输入，输出四个电机的力矩。

```
function [M1, M2, M3, M4] = motor_mixer(Roll, Pitch, Yaw, Thrust)

idle_PWM = 1000;
scale = 1000;

M1 = (Thrust - Roll + Pitch + Yaw) * scale + idle_PWM;
M2 = (Thrust + Roll - Pitch + Yaw) * scale + idle_PWM;
M3 = (Thrust + Roll + Pitch - Yaw) * scale + idle_PWM;
M4 = (Thrust - Roll - Pitch - Yaw) * scale + idle_PWM;
```

图 4.22 电机力矩分配

4.4.3 姿态控制器仿真

姿态闭环控制如图 4.23 所示，姿态控制器输出脉冲信号 PWM，通过调节 PWM 的占空比控制电机的转速，从而控制四旋翼无人机的姿态角、角速度，被控量返回输入姿态控制器的输入端形成闭环控制。

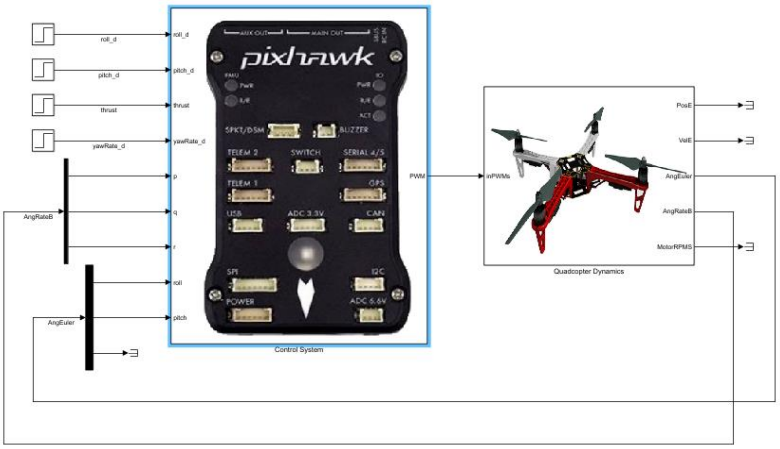


图 4.23 姿态控制器闭环控制

姿态控制器 PID 调参时，优先调节内环角速度环参数，在调节外环姿态环参数，避免调参混乱。内环角速度环参数调节，以比例系数为例，角速度期望输入为阶跃信号，令 K_p 为 0.06、0.08、0.1、0.12，得到如图 4.24 所示曲线图。

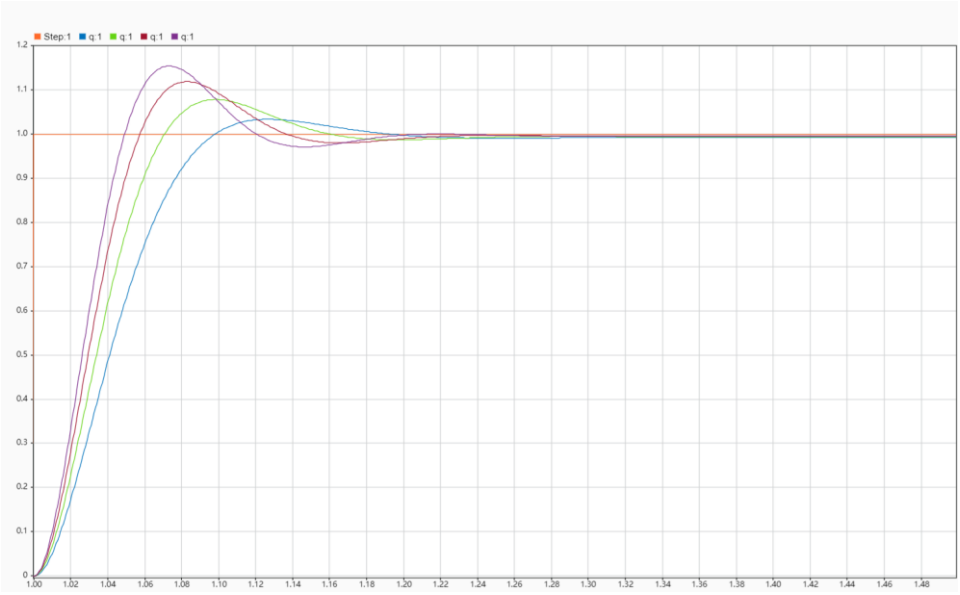


图 4.24 角速度环参数调节

姿态环参数调节，姿态角期望值为阶跃输入，令 K_p 为 6、6.5、7，得到如图 4.25 所示曲线图。

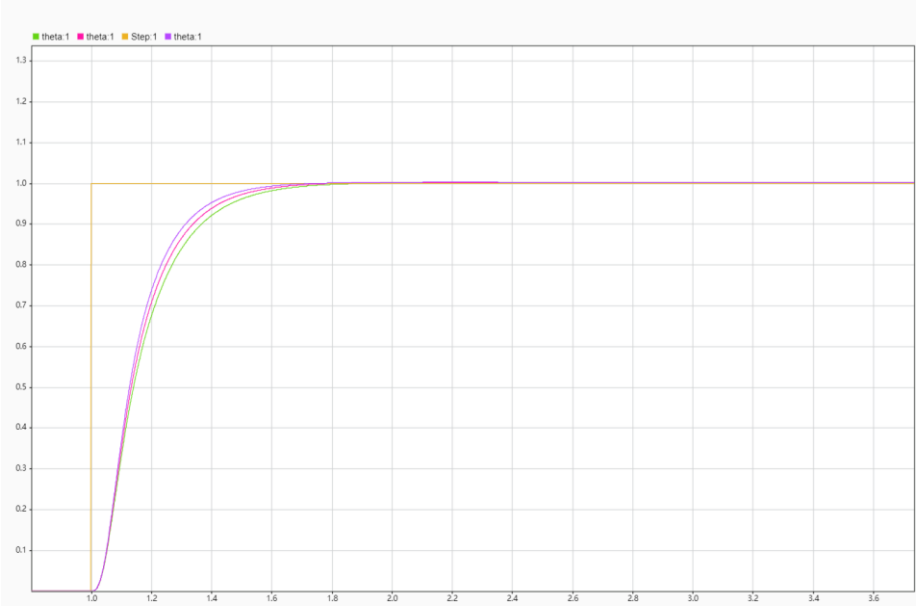


图 4.25 姿态环参数调节

Matlab 绘制姿态角的期望值和姿态角实际值的 Bode 图如图 4.26 所示，Gain Margin：29.3dB，At frequency：76.4rad/s：Phase Margin:80.6deg，At frequency：6.51rad/s。

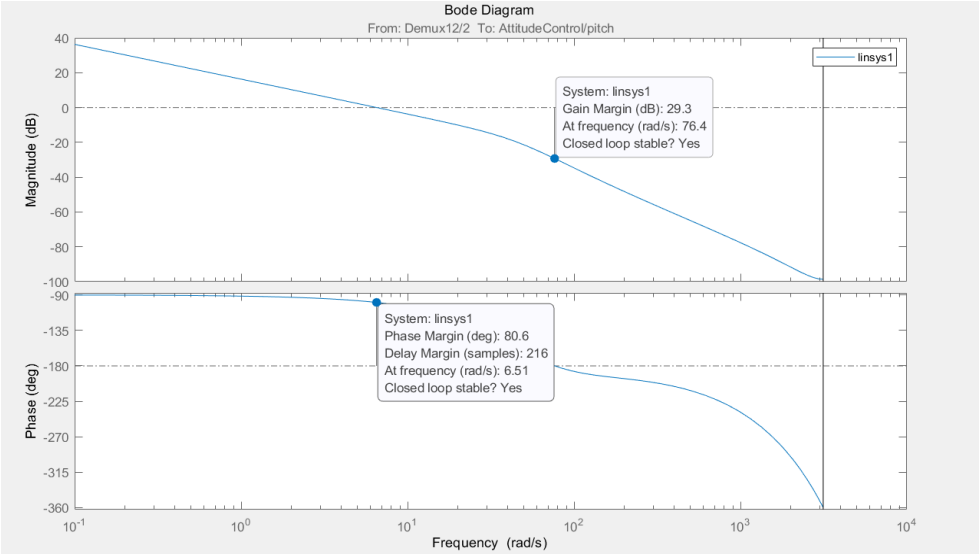


图 4.26 姿态控制器 Bode 图

经过 PID 参数调节过程，得到角速度环 PID 参数如表 4.3 所示。

表 4.3 姿态控制器角速度环 PID 参数

角速度环参数	$\dot{\theta}$	$\dot{\phi}$	$\dot{\psi}$
K_p	0.1	0.1	0.5
K_i	0.02	0.02	0.01
K_d	0.001	0.001	0.00

姿态环 PID 参数如表 4.4 所示：

表 4.4 姿态控制器姿态环 PID 参数

姿态环参数	θ	ϕ
K_p	6.5	6.5

4.5 本章小结

本章主要针对四旋翼飞行器的位置和姿态控制，基于 PID 理论分别设计了位置控制器及姿态控制器。四旋翼的控制系统划分为内环和外环结构，整合了双重闭环反馈机制，各个组件协同作用以实现四旋翼飞行器的有效操纵。后续对控制器仿真分析结果，通过调整 PID 参数，首先是速度环和角速度环的参数调节，然后是位置环和姿态环的参数调节，最终确定一组适宜的参数值，并绘制相应的伯德图。

第 5 章 总结与展望

5.1 主要工作

对四旋翼无人机的轨迹跟踪控制仿真进行研究，本论文的主要工作内容如下：

主要介绍了四旋翼无人机的历史背景和发展状况，国内外的研究现状以及本文的研究意义。

四旋翼无人机动力学建模，首先描述了该无人机的构造，接着详细阐述了四旋翼无人机如何通过垂直起降、偏航、俯仰和滚转四种基本运动模式来执行任务。最后，运用牛顿-欧拉动力学方程，推导出四旋翼无人机的位置动态方程和姿态动态方程，从而完成其全面的动力学建模。

四旋翼无人机路径规划，首先介绍了常见的搜索算法并比较，最终选取 A*算法进行路径规划。利用 PX4 官方控制器提供的 Offboard 模式功能，搭建 Simulink 模块实现与 Rflysim 的通信。最后利用单线激光雷达测距和 UDP 通信，将 A*算法 python 与 Simulink 模块通信，实现四旋翼无人机的路径规划。

四旋翼无人机位姿控制器设计，基于 PID 控制原理分别设计了位置控制器和姿态控制器。后续部分对仿真结果进行分析，通过调整 PID 参数，确定一组适宜的参数值，并绘制相应的伯德图。

5.2 后续研究工作展望

在不断的实验中也发现四旋翼无人机轨迹跟踪控制仿真系统存在一些不足的地方，需要未来进一步深入研究。

1）本文构建的动力学模型精炼了四旋翼无人机的关键动态特性，后续开展深入探究其内部结构与运动原理的研究，尤其针对其非线性和环境影响的特性。

2）常规的 PID 控制方法，尽管适用于某些简单系统，但在处理四旋翼无人机这种高度复杂的非线性系统时，其线性调控策略明显无法满足高精度控制和有效抗扰的需求，因此下一步结合反步滑模四旋翼姿态控制[20]进行研究。

参考文献

[1] 闫超, 涂良辉, 王聿豪. 无人机在我国民用领域应用综述[J]. 飞行力学,2022,40(03):1-6+12.

[2] Leishman J G. The Breguet-Richet quad-rotor helicopter of 1907[J]. 2001.

[3] 关成立, 杨岳. 四旋翼飞行器控制系统设计策略研究[J]. 现代计算机,2022,28(17):65-68.

[4] 赵志伟, 汤旭泽, 葛超. 四旋翼无人机 BSMC-ADRC 姿态控制以及轨迹跟踪[J/OL]. 控制工程:2024-05-19:1-9.

[5] S. Paschall and J. Rose, "Fast, lightweight autonomy through an unknown cluttered environment: Distribution statement: A — Approved for public release; distribution unlimited," *2017 IEEE Aerospace Conference*, Big Sky, MT, USA, 2017, pp. 1-8.

[6] D. Falanga, E. Mueggler, M. Faessler and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017, pp. 5774-5781.

[7] B. Zhou, F. Gao, L. Wang, C. Liu and S. Shen, "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight," in *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529-3536, Oct. 2019.

[8] M. Faessler, D. Falanga and D. Scaramuzza, "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight," in *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 476-482, April 2017.

[9] P. Foehn and D. Scaramuzza, "Onboard State Dependent LQR for Agile Quadrotors," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 6566-6572.

[10] M. Faessler, A. Franchi and D. Scaramuzza, "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories," in *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620-626, April 2018.

[11] S. Sun, A. Romero, P. Foehn, E. Kaufmann and D. Scaramuzza, "A Comparative Study of Nonlinear MPC and Differential-Flatness-Based Control for Quadrotor Agile Flight," in *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357-3373, Dec. 2022.

[12] Hanover D, Foehn P, Sun S, et al. Performance, Precision, and Payloads: Adaptive Nonlinear MPC for Quadrotors[J]. 2021.

[13] D. Kostadinov and D. Scaramuzza, "Online Weight-adaptive Nonlinear Model Predictive Control," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2020, pp. 1180-1185.

[14] 聂博文. 微小型四旋翼无人直升机建模及控制方法研究[D]. 国防科学技术大学,2007.

[15] 王俊生, 马宏绪, 蔡文澜. 基于 ADRC 的小型四旋翼无人直升机控制方法研究[J]. 弹箭与制导学报,2008(03):31-34+40.

[16] 李文巧. 基于自适应 PD 控制的四旋翼飞行器研究[D]. 重庆大学,2022.

[17] C. Ma *et al.*, "Decentralized Planning for Car-Like Robotic Swarm in Cluttered Environments," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, MI, USA, 2023, pp. 9293-9300.

[18] 唐嘉宁, 彭志祥, 李孟霜. 基于改进 A*算法的无人机路径规划研究[J]. 电子测量技术, 2023, 46(8):99-104.

[19] 刘栩鄴, 胡德清. 基于前馈补偿的 PD 四旋翼无人机轨迹跟踪控制[J]. 舰船电子工程,2022,42(07):55-60.

[20] 薛晶勇, 王斌锐. 指数型时变增益反步滑模四旋翼姿态控制[J]. 控制工程,2022,29(05):935-943.

致谢

光阴荏苒，大学生活即将结束，四年的学习生活使我受益匪浅。经历大半年时间的磨砺，毕业论文终于完稿，回首大半年来收集、整理、思索、停滞、修改直至最终完成的过程，我得到了许多的关怀和帮助，现在要向他们表达我最诚挚的谢意。

首先，我要感谢李永福指导导师。您的专业知识、严谨的学术态度和无私的指导使我在这个过程中受益匪浅。您的耐心和细心让我在遇到困难时能够坚持下去，您的鼓励和支持让我在取得成就时更加自信。没有您的悉心指导，我无法完成这份论文。

其次，我要感谢我的同学们和师兄。我们一起度过了许多难忘的时光，一起探讨问题，一起分享知识和经验。你们的存在让我的生活充满乐趣和活力。我很幸运能够与你们成为同学，也希望我们的友谊能够持续一生。

最后，我要感谢我的家人们。你们一直是我最坚实的后盾，在我遇到困难和挫折时给予我最大的支持和鼓励。你们的关心和支持让我能够在大学道路上勇往直前，不断追求进步和发展。没有你们的支持，我无法走到今天这一步。

附录 A

```
#Python 与 simulink 通信
import time
import math
import sys
import VisionCaptureApi
import PX4MavCtrlV4 as PX4MavCtrl
import cv2
import matplotlib.pyplot as plt
import UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()
import numpy as np
import socket
import struct
import x
import y

fig = plt.figure()


def viz_matplot(points, name="lidar"):
    plt.clf()
    plt.title(name)
    ax = fig.add_subplot(111, projection='3d')
    x = points[:, 0]  # x position of point
    y = -points[:, 1]  # y position of point
    z = points[:, 2]  # z position of point

    diff = np.abs(z - vis.ImgData[i, 3])
    indices = np.where(diff < 0.1)
    x_values = x[indices]
    y_values = y[indices]

    ax.scatter(y,   # y
               x,   # x
               z,   # z
               c=z,  # height data for color
               cmap='rainbow',
               marker=".")
    ax.axis()
    plt.pause(0.0001)
    plt.ioff()


def viz_matplot_2d(points, height, name="lidar"):
```

```
    plt.clf()
    plt.title(name)
    x = points[:, 0]  # x position of point
    y = -points[:, 1]  # y position of point
    z = points[:, 2]  # z position of point

    diff = np.abs(z - height)
    indices = np.where(diff < 0.1)
    x_values = x[indices]
    y_values = y[indices]

    plt.scatter(x_values, y_values, marker=".", c='red')
    plt.axis()
    plt.pause(0.0001)
    plt.ioff()


def send_cmd(x_d, y_d, z_d):
    # 创建一个 INET, Datagram (UDP) 套接字
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    doubles = [x_d, y_d, z_d]

    # 将 double 数据转换为字节
    data = struct.pack('d'*len(doubles), *doubles)

    # 定义 Simulink 服务器的主机名和端口号
    host = "127.0.0.1"
    port = 1230

    # 发送数据
    s.sendto(data, (host, port))

    # 关闭套接字
    s.close()


# Create a new MAVLink communication instance, UDP sending port (CopterSim's receving port) is 20100
mav = PX4MavCtrl.PX4MavCtrler(1)

# The IP should be specified by the other computer
vis = VisionCaptureApi.VisionCaptureApi()

# Send command to UE4 Window 1 to change resolution
# 设置 UE4 窗口分辨率，注意本窗口仅限于显示，取图分辨率在 json 中配置，本窗口设置越小，资源需求越少。
ue.sendUE4Cmd('r.setres 1280x720w', 0)
```

```
ue.sendUE4Cmd('t.MaxFPS 90', 0)  # 设置 UE4 最大刷新频率，同时也是取图频率
time.sleep(2)

# VisionCaptureApi 中的配置函数
vis.jsonLoad()  # 加载 Config.json 中的传感器配置文件

isSuss = vis.sendReqToUE4()  # 向 RflySim3D 发送取图请求，并验证
if not isSuss:  # 如果请求取图失败，则退出
    sys.exit(0)
vis.startImgCap()  # 开启取图，并启用共享内存图像转发，转发到填写的目录
isShowCloud = True
# 下面的程序按 30Hz 读取点云数据
# 注意：如果要去看点云的实际接收频率，请搜索 img_udp_thrdNew 函数
lastTime = time.time()
num = 0
lastClock = time.time()
while True:

    send_cmd(y.astar(num),x.astar(num), 0)
    #print('num:',num,x.astar(num),y.astar(num))
    num=num+1
    lastTime = lastTime + 1.0
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()

    for i in range(len(vis.hasData)):
        if vis.hasData[i]:
            height = vis.ImgData[i][2] - 1.47
            print('height: ', height)
            viz_matplot_2d(vis.Img[i], height)
            print('PosAngNum: ', vis.ImgData[i])  # 雷达在地面坐标系下的位置、姿态和点云总数
            print('TimeStmp: ', vis.timeStmp[i])  # 当前的数据时间戳
            vis.hasData[i] = False
```

```
#AStar
import cv2
import numpy as np
import time
import VisionCaptureApi
import PX4MavCtrlV4 as PX4MavCtrl
import math
import threading
import copy

img = cv2.imread("star.png")
# 初始化，默认像素值为 255 表示可通行，否则不可通行
img[img < 50] = 0
img[img > 50] = 255
# cv2.imshow("raw", img)
# cv2.waitKey(0)

# 在实际应用中，因为路径规划需要考虑载体运动学模型，把载体当做质点来考虑规划出来的轨迹往往需要做进一步的工作(如根据运动学约束,轨迹平滑，插值等)
# 如果不考虑运动学模型，可行情况下对障碍物做膨胀处理，这样轨迹不会贴着实际障碍物
grid = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
grid[grid <= 200] = 1  # 1:表示不可通行
grid[grid > 200] = 0  # 0:表示可通行

grid = grid.tolist()
origin_p = [70, 180]  # 设置起点
goal_p = [38, 760]  # 设置目标点


def AStar(grid: list, begin_p: list, goal_p: list, cost=2):
    assert ((grid[begin_p[0]][begin_p[1]] != 1)
            and (grid[goal_p[0]][goal_p[1]] != 1))
    # the cost map which pushes the path closer to the goal
    heuristic = [[0 for row in range(len(grid[0]))]
                 for col in range(len(grid))]
    for i in range(len(grid)):
        for j in range(len(grid[0])):
            heuristic[i][j] = abs(i - goal_p[0]) + \
                abs(j - goal_p[1])
            if grid[i][j] == 1:
                # 在启发式地图里增加额外惩罚
                heuristic[i][j] = 100000

    # 遍历方向
    delta = [[-1, -1],
             [-1, 0],
```

```
            [-1, 1],
            [0, -1],
            [0, 1],
            [1, -1],
            [1, 0],
            [1, 1]]

    close_matrix = [[0 for col in range(len(grid[0]))] for row in range(
        len(grid))]  # 用于存已经走过了路
    close_matrix[begin_p[0]][begin_p[1]] = 1
    action_matrix = [[0 for col in range(len(grid[0]))]
                     for row in range(len(grid))]  # 存储可能的路

    x = begin_p[0]
    y = begin_p[1]
    g = 0
    f = g + heuristic[begin_p[0]][begin_p[0]]
    cell = [[f, g, x, y]]

    found = False
    resign = False

    while not found and not resign:
        if len(cell) == 0:
            resign = True
            return None, None
        else:
            cell.sort()
            cell.reverse()
            next = cell.pop()
            x = next[2]
            y = next[3]
            g = next[1]
            f = next[0]

            if x == goal_p[0] and y == goal_p[1]:
                found = True
            else:
                for i in range(len(delta)):
                    x2 = x + delta[i][0]
                    y2 = y + delta[i][1]
                    # 判断可否通过那个点
                    if x2 >= 0 and x2 < len(grid) and y2 >= 0 and y2 < len(grid[0]):
                        if close_matrix[x2][y2] == 0 and grid[x2][y2] == 0:
                            g2 = g + cost
                            f2 = g2 + heuristic[x2][y2]
                            cell.append([f2, g2, x2, y2])
                            close_matrix[x2][y2] = 1
                            action_matrix[x2][y2] = i
```

```
    invpath = []
    x = goal_p[0]
    y = goal_p[1]
    invpath.append([x, y])  # 路径从起点到终点，因此需要反转
    while x != begin_p[0] or y != begin_p[1]:
        x2 = x - delta[action_matrix[x][y]][0]
        y2 = y - delta[action_matrix[x][y]][1]
        x = x2
        y = y2
        invpath.append([x, y])

    path = []
    for i in range(len(invpath)):
        path.append(invpath[len(invpath) - 1 - i])
    return path, action_matrix


print(img.shape)
cv2.circle(img, (origin_p[1], origin_p[0]), 4, (0, 0, 255), 8)
cv2.circle(img, (goal_p[1], goal_p[0]), 4, (255, 0, 0), 8)

path, a = AStar(grid, origin_p, goal_p)
print(path)
for i in range(len(path)):
    cv2.circle(img, (path[i][1], path[i][0]), 1, (0, 255, 0), 1)

cv2.imshow("path", img)
cv2.waitKey(0)
```