深 圳 大 学

本 科 毕 业 论 文（设计）

题目: **基于模型的微型四旋翼飞行系统设计与开发**

姓名:  **车凯威**

专业:  **自动化**

学院:  **机电与控制工程学院**

学号:  **2016110110**

指导教师: **张博**

职称：  **副教授**

2020年 5 月 11 日

**深圳大学本科毕业论文（设计）诚信声明**

本人郑重声明：所呈交的毕业论文（设计），题目《基于模型的微型自主四旋翼飞行机器人飞控系统设计与开发》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。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注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。



毕业论文（设计）作者签名：

日期： 2020 年 5 月 11 日

摘 要

随着科学技术的发展，无人机逐步向微型化，高集成以及高性能方向发展。并且随着微型四旋翼飞行器相关技术的发展，其在室内和低空环境作业的优势愈发明显，使其的应用范围越来越广泛。同时，在近些年来图像处理技术和人工智能快速发展，使得这些技术在自主导航和目标搜索领域有非常好的应用，这些因素让视觉控制技术引入到无人机的开发变成一种趋势。因此，本文针对微型四旋翼飞行器在低空环境作业的场景，使用通过“基于模型设计”（MBD）的方法对飞行控制算法进行开发，并对算法的适应场景进行分析。

首先，本文研究的无人机原型为Parrot的微型四旋翼无人机，此款无人机也同时为本文的研究对象，因为官方提供了该无人机的数学模型，后期开发非常便捷。同时本文分析了四旋翼飞行器的基本结构和飞行原理，以及在特定的条件下得到此微型四旋翼无人机的数学模型。

其次，本文的研究内容为：设计三种视觉控制算法，来实现无人机沿着特定颜色的路径飞行，并且使用通过基于模型设计的方法来快速验证和部署。

所设计的控制系统控制任务有两个：第一是需要先建立无人机的物理模型，并设计无人机飞行的路径。第二是开发三种不同的飞行控制算法进行控制，使无人机完成整体路径的飞行。并研究这三种方法对于飞行的飞行时间、整体误差以及飞行路途中三种不同角度，也就是微小角度，中等角度和大角度所产生的误差。最终得到不同方法的适应场景，并证明了“基于模型设计”应用于飞行控制系统设计的可行性。

最后，本文得出以下结论：本文提出了三种不同的飞控算法进行开发，通过根据基于模型设计的方法进行了快速的验证，分别是设计了“三圆轨迹生成算法”的飞控系统，可以对微小的变化敏感地跟踪。设计了“散点探测循迹算法”的飞控系统，对大范围不确定路径更好的跟踪。设计了“重心循迹算法”的飞控系统，相比于“散点探测循迹算法”，更好地实现调速。对比三种方法的结果找到适合于不同情境下的无人机飞行控制系统，可以快速实现个性化开发。

关键词：四旋翼；基于模型设计；飞行控制； 视觉控制；

**Abstract**

Micro UAV is an electric autonomous vehicle which can take off and land vertically. Its structure is more compact, so it can produce more lift than its weight. At the same time, with the development of the related technology of the micro UAV, its advantages in indoor and low altitude environment are more and more obvious, making its application scope more and more extensive. At the same time, with the rapid development of image processing technology and artificial intelligence in recent years, it has a very good application in the field of autonomous navigation and target search, which makes the introduction of visual control technology into the development of UAV become a trend. Therefore, in this paper, we use the MBD method to develop the flight control algorithm and analyze the adaptability of the algorithm in the low altitude environment.

First of all, the prototype of the UAV studied in this paper is Parrot's micro quadrotor UAV, which has the official mathematical model and is very convenient for later development. At the same time, this paper analyzes the basic structure and flight principle of the UAV, and obtains the mathematical model of the micro UAV under specific conditions, and studies the control system theory of the UAV.

Secondly, this paper designs three visual control algorithms based on the model method to realize the UAV flying along a specific color path. There are two tasks in the control system: the first is to build the physical model of UAV and design the flight path of UAV. The second is to develop three different flight control algorithms to control the UAV to complete the overall path of flight. The errors caused by these three methods for flight time, overall error and three different angles on the flight path, namely, small angle, medium angle and large angle, are studied. Finally, the adaptive scenarios of different methods are obtained, and the feasibility of applying model-based development to the design of flight control system is proved.

Finally, this paper draws the following conclusions: according to the concept of model-based development, three different flight control algorithms are proposed for development, namely, the flight control system of "three circle trajectory generation algorithm" is designed, which can track small changes sensitively. The flight control system of "scatter detection tracking algorithm" is designed to better track the large-scale uncertain path. The flight control system of "center of gravity tracking algorithm" is designed. Compared with "scattered point detection tracking algorithm", it can achieve better speed regulation. Comparing the results of the three methods, we can find the UAV system suitable for different situations, which can quickly realize the personalized deployment.

**Key words：** UAV, model-based design, fight control, computer vision control

目录

摘要 Ⅲ  
**ABSTRACT** Ⅴ

[1 绪论 1](#_Toc40133536)

[1.1 课题研究背景 1](#_Toc40133537)

[1.2 国内外研究现状 1](#_Toc40133538)

[1.2.1微型四旋翼飞行器研究现状 1](#_Toc40133539)

[1.2.2基于模型设计（MBD）研究现状 3](#_Toc40133540)

[1.2.3无人机视觉导航与控制研究现状 4](#_Toc40133541)

[1.3 研究目的及意义 4](#_Toc40133542)

[1.4 课题的主要研究内容 5](#_Toc40133543)

[2 四旋翼无人机物理建模 6](#_Toc40133544)

[2.1 四旋翼无人机基本结构 6](#_Toc40133545)

[2.2 四旋翼无人机坐标 6](#_Toc40133546)

[2.3 四旋翼无人机飞行原理 7](#_Toc40133547)

[2.4 四旋翼无人机数学模型建立 8](#_Toc40133548)

[3 三种无人机视觉控制算法设计 9](#_Toc40133549)

[3.1 视觉控制算法的控制任务 9](#_Toc40133550)

[3.2 仿真模型搭建 10](#_Toc40133551)

[3.2.1整体仿真模型的搭建 10](#_Toc40133552)

[3.3.1 可视化模块 11](#_Toc40133553)

[3.1.1 FCS模块设计 12](#_Toc40133554)

[3.3数据处理方法 13](#_Toc40133555)

[3.4 视觉控制算法设计思路 15](#_Toc40133556)

[3.5 三圆轨迹生成算法 16](#_Toc40133557)

[3.5.1图像处理算法 17](#_Toc40133558)

[3.5.2飞行控制算法 23](#_Toc40133559)

[3.5.3仿真数据分析 27](#_Toc40133560)

[3.6散点探测循迹算法 30](#_Toc40133561)

[3.5.1图像处理算法 30](#_Toc40133562)

[3.5.2 飞行控制算法 33](#_Toc40133563)

[3.5.2 仿真数据分析 33](#_Toc40133564)

[3.7重心循迹算法 35](#_Toc40133565)

[3.3.1图像处理算法 36](#_Toc40133566)

[3.3.2飞行控制算法 38](#_Toc40133567)

[3.3.2仿真数据分析 41](#_Toc40133568)

[4 三种无人机视觉控制算法仿真结果对比 44](#_Toc40133569)

[4.1 三种算法飞行速度与整体误差比较 44](#_Toc40133570)

[4.2 三种算法对于角度反应的比较 44](#_Toc40133571)

[4.1 结论 50](#_Toc40133572)

[5 总结与展望 51](#_Toc40133573)

[5.1 总结 51](#_Toc40133574)

[5.2 展望 51](#_Toc40133575)

[致谢 53](#_Toc40133576)

[参考文献 54](#_Toc40133577)

[附录一 三圆轨迹生成算法视觉处理 56](#_Toc40133578)

[附录二 三圆轨迹生成算法飞控函数 56](#_Toc40133579)

[附录三 散点探测循迹算法飞控函数 57](#_Toc40133580)

[附录四 重心循迹算法视觉处理 57](#_Toc40133581)

[附录五 重心循迹算法飞控函数 58](#_Toc40133582)

[附录六 转换系数求取函数 59](#_Toc40133583)

1 绪论

1.1 课题研究背景

本文使用的微型四旋翼飞行器一种电动的、能够垂直起降的自主飞行器，其结构上为非共轴式碟形飞行器。与常规四旋翼式飞行器比较而言，其设计的结构更加紧凑，因此可以产生相比于其重量来说更大的升力，同时其四个旋翼抵消掉，反扭转力，因此不需要专门设计反扭转矩桨。随着近些年来新材料、微型机电系统、微型，惯导系统和飞行控制系统的技术进步，微型四旋翼飞行器发展非常迅猛[1]。

总的来说微型四旋翼飞行器很适合在低空环境作业，比方说室内和城镇的交通系统以及，茂密的丛林，在这些地方执行监测的任务对于大型的无人机来说是非常复杂的，因此这一类微型四旋翼飞行器具有很好的民用和军用的用途。同时在火星探测的任务当中，微型四旋翼飞行器也是非常重要的研究方向[2]。

同时微型四旋翼飞行器相比较于其他的飞行器，很大的优势在于它的机械结构相对而言比较简单，控制的方式只需要通过控制四个马，达的转速，并且飞行的机动性会更加的好其次，微型四旋翼飞行器具有较高的操作能力，这个体现在能够在小范围内起飞，在空中悬浮，飞行以及，着陆。甚至可以飞得离目标非常的接近，而不需要像传统的直升机，不能非常靠近目标进行观察。

同时这些微型的四旋翼飞行器同时这些微型的四旋翼飞行器也为自动控制，现在控制、智能控制，三维路径规划等领域提供了非常好的学习平台和实验平台，因此具有非常高的研究价值。

1.2 国内外研究现状

1.2.1微型四旋翼飞行器研究现状

目前世界上的微型四旋翼飞行器，一般来说可以分成以下三大类，微型四旋翼飞行器、小型四，旋翼飞行器以及遥控航模四旋翼飞行器。

到现在，微型和小型四旋翼飞行器的基础理论已经取得很好的进展，但是如果要走向市场，仍然有非常多的关键技术有待克服[3]。

**（1）最优化总体设计**

对微型四旋翼飞行器进行总体的设计的时候，通常需要遵循这些原则：尺寸更小、重量更轻、速度更快、成，本与功耗更低。但同时我们知道这些原则通常会存在矛盾。比方说在其他条件相同时，尺寸与速度会成反比。因此在整体的设计时，需要综合各方面的因素，一方面要选取合适的性能所对应的材料以减少飞行器的重量，另一方面也要考虑尺寸和飞行速度，通过系统化的设计，使得总体设计最优。

**（2）能源与动力**

微型四旋翼无人机的动力装置有以下：直流电机、电机驱动的模，块，减速箱，旋翼，以及光电编码盘。能源由机载的飞行电池提供。我们知道微型四旋翼飞行器的尺寸受其重量影响，同时能源与动力的装置是最重要的部分，通常能够占到70%的比例，所以我们如果能够研制更加高效而且更加轻的能源与动力装置，就能够更好的将微型四旋翼飞行器的尺寸变得更小。

其次，能源与动力的装置的升级也能够更好的提升飞行器的效率，使得微型四旋翼飞行器能够在不增大其电源的情况之下，有更长时间的续航。因此研究能源与动力装置也是非常重要的方向。

**（3）数学模型的建立**

为了实现更好的控制，则必须对微型四旋翼飞行器在各种情况之下的数学模型和物理模型都要有精确的描述。但是在正常的飞行中，微型四旋翼飞行器会受到很多种物理效应的作用，比方说其自身的重力，风力以及旋翼的惯性。因此即便建立了静态的数学和物理模型，也难以建立，进干扰之下的数学和物理模型，同时因为其尺寸较小，质量比较轻，因此受到外界的干扰的时候更加容易变形，因此相比于常见的四旋翼飞行器，其更难获得准确的模型。所以建立微型四旋翼飞行器的数学模型，需要继续研究低雷诺，数条件下的空气动力学问题。

**（4）飞行控制**

微型四旋翼飞行器具有6个自由度，同时它有4个输入，他是一个具有强、非线性、多变量的欠驱动系统。这些都使得它的飞行控制系统设计比较麻烦，同时受限于其尺寸和重量，其传感器一般相对较小，精度较低，所以其控制的，难度将会大大增加。

总的来说，姿态的控制是飞行控制的关键。目前全世界使用较广的飞行控制算法有以下几种：经典 PID 、线性、非线性、智能控制。

比例－积分－微分（PID)控制原理相对简单， 其根据系统产生的误差进行负反馈修正，因为其是非常经典的控制方法，因此经过了大规模的验证，技术相对比较成熟，而且在工程领域上面容易实现，

线性控制是现代控制理论的重要部分，现代控制理论在系统的分析与控制器的设方面理论相对比较完备。这种线性控制器核心的设计理念是将四旋翼的，数学和物理模型，在工作点附近进行线性化，以此来得到线性的模型以此来得到线性的模型。常用的控制这些线性模型的方法有以下两种：LQR ，控制和 H∞ ，控制。

上面提到的线性控制其实是一种损失了模型精度的方法，因为实际上的四旋翼飞行系统是很典型的飞信新系统，但是我们的线性控制是要建立在线性化的四旋翼数学和物理模型上，这种方式损失了模型的精度。但是直接采用非线性化的控制相对较难目前常用于四旋翼飞行系统的非线性控制有以下几种：反步法控制、滑模控制以及反馈线性化控制。

智能控制是现在非常火热的一个领域，原因是因为人工智能在近些年来迎来了井喷式的发展。控制核心在于降低对模型精确度的依赖，拥有一部分的自适应性，常见的智，能控制方法有以下几种：模糊逻辑控制、神经网络控制。

介绍完了以上几种常见的飞行控制方法，但是我们知道在目前的应用领域，通常仍是以PID控制为核心。因为采用其他稍微先进的控制方面仍然有较大的不确定性。随着无人机的控制技术不断发展，以及民用与军事用途的需求不断的提升，微型四旋翼飞行，器的集群化控制以及自主化控制将是很重要的发展方向。

**（5）导航与定位**

微型的四旋翼飞行器主要的活动空间为近地面，比方说城区，森林室内，但是我们知道在这些地方GPS往往不能够正常工作，存在很严重的定位通信与导航方面的各种，各样的问题，因此需要结合惯性导航系统以及其他传感器技术才能够更好的进行定位导，航与控制，尤其是靠近地面的环境相对来说比较复杂，而且干扰的因素也比在高空环境，更加的多。比方说，风速的不确定性，强磁场的干扰，低空环境下各种信号的干扰，这些都对当前的通信技术有了非常大的考验。本文主要的研究方向也就是这里的视觉导航。

1.2.2基于模型设计（MBD）研究现状

基于模型设计的理念为利用一个集中的平台，对系统进行设计和开发，同时可以进行仿真参数的在线修改，仿真数据的实时读取等功能。整个过程包括模型建立、算法开发、测试算法、产品生产[6][17][18]。

在设计阶段，模型的建立非常考验工程师的能力，建立的过程中，需要不断地对各种参数和指标进行测试，以确保模型的准确性。同时对于算法而言，这样的过程仍是不可或缺的，在这个阶段，基于模型的设计能提供的价值如下[10]：

**（1）数学仿真**

可以验证模型是否准确，误差为多少，以及验证算法是否符合相应指标。

**（2）软件在回路中仿真（SIL）**

整个控制系统运作过程中，检验实时性，检验采样的步长，检验延时环节的影响等因素。

**（3）原型化系统仿真（RCP）**

此时控制系统的对象均为真实的物体，比方说真实的直流电机等，通过这样的测试可以很好地测试系统的性能，同时可以更快速地完成从原型到产品的过程。

**（4）系统半实物仿真（HILS）**

完成了系统控制器的开发后，对这个系统在各种环境下进行测试，可以使得系统在上市前提前暴露问题，进行修正。

基于模型设计的理念，在国外有广泛的使用嵌入的产品，有非常多样。基于模型设计的基础在于模型的构建，而传统的模式的基础在于实体的实验。Matlab提供了非常完.善的基于模型设计的成套系统。

一个典型的例子就是东风汽车公司，工程师们使用基于模型设计的开发理念，并在，Matlab平台进行开发，历时18个月开发了一套精巧的电池管理系统。这个电池管理系，统装备在其旗下的混合动力公交车中，该车的碳排放明显减少，电源利用率显著提高了三成。开发的时候，重要的参数都是一边设计一边调试出来的，同时从构想到理论验证，再到交付产品只花了18个月，远远比传统开发的30个月要短的多[10]。

1.2.3无人机视觉导航与控制研究现状

现在的室外无人机基本利用GPS、惯性导航等方式提供的信息，但是在室内和低空，环境中GPS信号大大减弱，因此很大程度上影响了飞行的精度。同时，在近些年来图像，处理技术和人工智能的快速发展，使得视觉控制技术引入到无人机的开发变成一种趋势。，GPS失效是，视觉控制通过摄像头获取视觉信息，以此来控制飞行。其次，GPS通常，的精度为米级，但是相对高清度的视觉控制可以在室内达到厘米级，因此这个领域具有广泛的应用场景[4]。

视觉控制是指利用计算机视觉来对飞行器进行控制，核心技术为图像处理与控制理论。

视觉控制的核心，都是在于如何减少误差为：

为图像的测量值；为视觉特征点的矢量；为摄像头的参数；为期望的特征值。

通过对的不同，视觉控制可以分为两个领域：基于位置的视觉伺服（PBVS），，基于图像的视觉伺服（IBVS），PBVS的为三维参数，需要从图像中，读取。IBVS的为特征点，即在图像中目标点的坐标。

其中PBVS的特点是可以直观定义空间坐标，但是依赖于位姿估计的精度。

IBVS的特点是无需三维的定位，因此复杂度较低，但是需要额外获得深度信息。

本文采用的是IBVS的控制思想，其控制的流程图如下[14][18]：

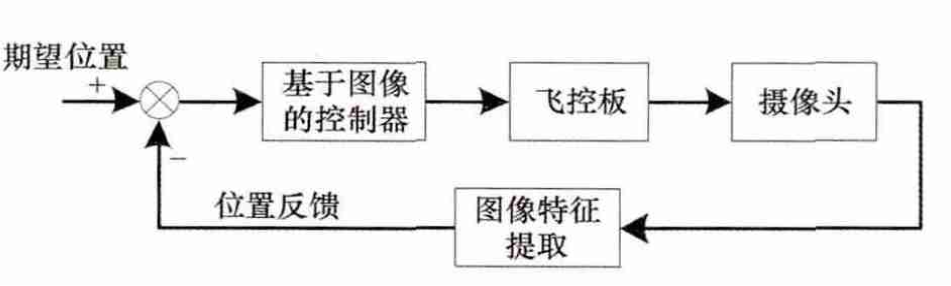


图1-1 Parrot mambo的基本组成

1.3 研究目的及意义

虽然经过大量的研究、长期的发展与改进之后，人们利用传统的四旋翼无人机控制方法，已经能对在已知环境模型的环境下的四旋翼无人机，通过建立基于任务的精确数学模型设计控制系统，完成较为稳定、准确与快速的控制。但是在面对非常多个性化需求的场景时，这种控制与开发便会相对吃力。

以巡逻机为例详细叙述，城市的低空的空域现在越来越复杂，因此低空域的事件常，具有突发性和隐蔽性等特点，容易成为城市的安全隐患，比方说，仓储的安保，城市路，边的乱停乱放现象，因此需要完善低空空域的监控。首先，对于这些监控而言，反应的，迅速性以及活动的灵活性是首要考虑的，故通常采用四旋翼的无人机来巡逻低空，为了，能够更加灵活的处理这些突发事件，无人机必须能够进行自主的航路规划，绕开飞行禁区以及实体的障碍。在航路规划之后能够自动控制飞行，实现航路的跟踪以及对特定目标的识别，这些功能对无人机的二次开发提出了非常高的要求

针对这些二次开发，本文引入“基于模型设计”的方式对无人机系统进行二次开发。同时我们注意到传统的无人机飞行基本使用GPS、惯性导航等方式提供的信息，但精度相对较低。近年来，伴随着图像处理技术快速发展，使得视觉控制技术引入到无人机的开发中来，无人机开始搭载视觉系统，根据视觉系统来进行控制并导航。用这种方法进行开发，是对飞行控制系统的新的探索[14]。

1.4 课题的主要研究内容

本文的研究主要是基于模型的微型四旋翼飞控系统设计与开发，重点研究如何通过基于模型设计的方式实现对四旋翼无人机的控制，以及四旋翼无人机视觉导航与控制，同时分析基于模型设计对四旋翼无人机控制的可行性与效果。并将尝试通过导入路径模，型，在Simulink中建立四旋翼无人机的物理模型与操作环境，并设计该四旋翼无人机，的运动控制系统，以达到遍历跑道的目的；再将仿真完成的控制器通过代码生成方式，配置到实体机上。

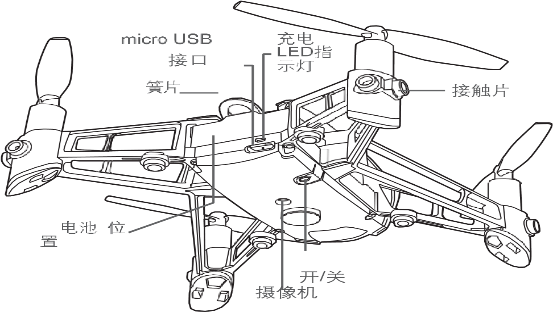
研究的对象为Parrot的微型四旋翼无人机。研究目标为设计飞行控制算法，来实，现无人机沿着特定颜色的路径飞行。并研究这些方法对于飞行的整体误差以及飞行路途中三种不同角度，也就是微小角度，中等角度和大角度所产生的误差。最终得到不同方法的适应场景，并证明了基于模型设计应用于飞行控制系统设计的可行性。

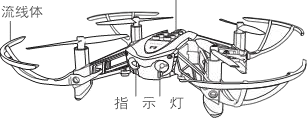
具体的控制系统控制任务有两个：第一是需要先建立无人机的物理模型，并设计无人机飞行的路径。第二是开发三种不同的飞行控制算法进行控制，使无人机完成整体路径的飞行。并研究这三种方法对于飞行的各种影响。

2 四旋翼无人机物理建模

2.1 四旋翼无人机基本结构

通常来说，四旋翼飞行器有两部分构成，第一为机械部分，包括电池、电机、飞行，器机架、以及螺旋桨构成；第二位为控制部分，包括MCU、传感器。四旋翼飞行器通过，MCU给电机发送指令，控制流电机的方向和速度，进而控制四旋翼的前进、上升、旋转。





。图2-1 Parrot Mambo的基本组成

2.2 四旋翼无人机坐标

四旋翼在运动导航过程中，依赖不同的坐标系进行运动控制的解算。本报告在控制，过程中主要使用了四旋翼无人机的机身坐标系和导航坐标系。其中，Parrot 四旋翼无人机的机身坐标系定义如图1.2所示：

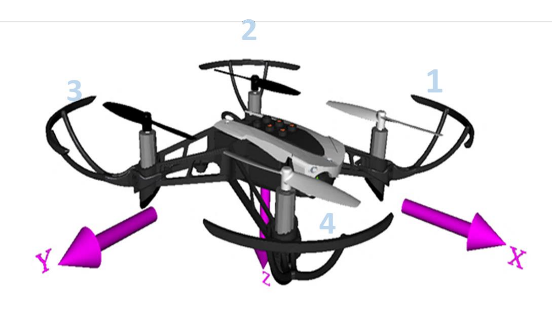


图2-2 四旋翼无人机的机身坐标系定义

坐标系各轴定义如下所示：

X轴从重心开始，指向四架飞机机头的方向。

Y轴从重心开始，指向四架飞机的右侧。

Z轴从重心开始，按照右手定则从四分之一飞行器向下。

其次，另一个坐标系是导航坐标系。其X 轴指向北方，Y轴指向东方向，Z 轴指向地面，如图示：

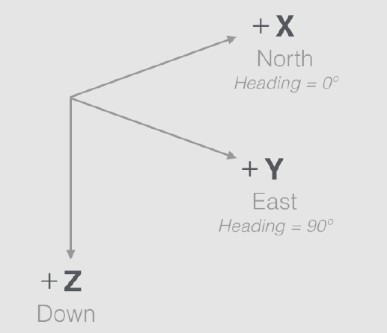


图2-3 导航坐标系

完成以上坐标的定义之后，便可以将从传感器上获得的数据变换到地理坐标系上，从而实现导航求解。坐标的变换可以通过欧拉解 321 的变换方式实现，其矩阵具有以下的表达式。



2.3 四旋翼无人机飞行原理

四旋翼飞行器的飞行动作是通过四个旋翼的转速不同的来决定的。四旋翼的主要飞行动作包括有偏航运动、俯仰运动、垂直上升运动以及横滚运动。

无刷直流电机的控制方面，转子1相对于Z轴正旋转。它与xy平面平行，与x轴成-45度角；转子2相对于机体的Z轴负旋转。它与xy平面平行，距离x轴-135度； 转子3的旋转方向与转子1相同。它与xy平面平行，与x轴成135度角；转子4的旋转方向与转子2相同。它与xy平面平行，与x轴成45度角。如图1.4所示。假设飞行器的各个飞行动作的前提是保持竖直方向上的总的升力不变，即四旋翼飞行器此刻在某一个水平面内的情况下，为实现飞行器的上升、偏航、俯仰、横滚，四个无刷直流电机的控制方式如下所示：

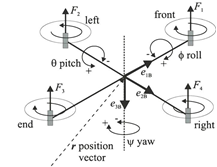


图2-4 四旋翼运动学解析

根据上述飞行器原理，可以知道各个飞行动作与电机编号之间关系，如下表所示：

**表1.1 电机转速与飞行器动作的关系**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **电机1** | **电机2** | **电机3** | **电机4** |
| **向 上** | **+** | **+** | **+** | **+** |
| **偏 航** | **+** | **-** | **+** | **-** |
| **俯 仰** | **+** | **-** | **-** | **+** |
| **横 滚** | **-** | **-** | **+** | **+** |

2.4 四旋翼无人机数学模型建立

下面不加推导的给出四旋翼的质心运动方程：



其中转动方程：













其中Ω表示是在机体坐标下的角速度，其和欧拉角的变化率在滚转角和俯仰角和小的时候可以等价。

3 三种无人机视觉控制算法设计

3.1 视觉控制算法的控制任务

本文通过基于模型的方法设计飞行控制算法，来实现无人机沿着特定颜色的路径飞行。并研究这些方法对于飞行的整体误差以及飞行路途中三种不同角度，也就是微小角度，中等角度和大角度所产生的误差。因此所设计的控制系统控制任务有两个：

第一是需要先建立无人机的物理模型，并设计无人机飞行的路径。

第二是开发不同的飞行控制算法进行控制，使无人机完成整体路径的飞行。并研究这不同方法对于飞行的各种影响。

为了整体测试三种视觉控制算法的利弊，设计了带有直道、微小角度弯道、中等角度弯道、以及大角度弯道的路径，路径为红色，地面为绿色或灰色。这种颜色导引带可以为无人机提供准确的方向信息。无人机在经过路径这些重要部分时，能否稳定的沿着轨迹飞行，有无偏离路径的情况，以及偏离路径程度如何等参数，体现了循迹算法的优劣。仿真会分析以下数据和参数：

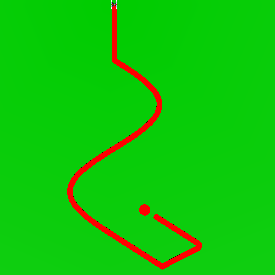


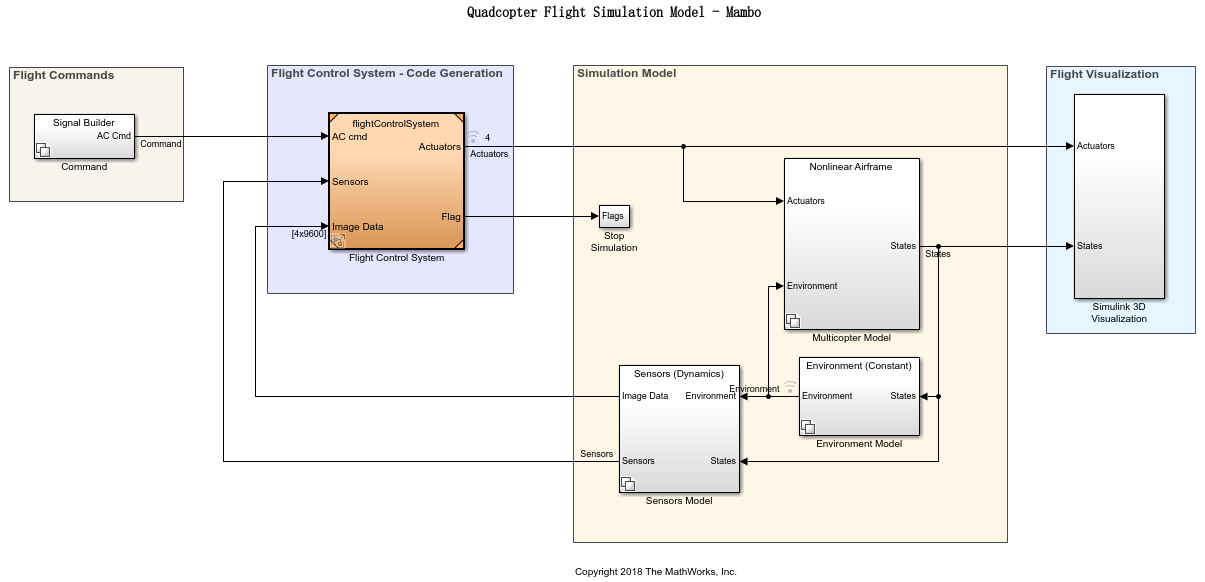
图3-1-1 模拟飞行路径

1. 整体飞行轨迹
2. 整体飞行时间
3. 整体误差
4. 三种弯道的超调量
5. 三种弯道的震荡程度
6. 三种弯道的峰值时间
7. 三种弯道的稳定时间

3.2 仿真模型搭建

3.2.1整体仿真模型的搭建

本文采用Parrot的微型四旋翼飞行器，在Matlab中有Parrot的Simulink Support Package for PARROT，其中，已经有现成的 Parrot微型四旋翼飞行器的数学和物理模型以及可视化平台。整体系统如下图：

图3-2-1 整体仿真系统

有四大模块，其中多自由度模型、传感器模型、环境模型在Simulink仿真的时候可以保持参数不变。即通过Command模块产生飞行指令，结合FCS(Flight Control System)实现飞行控制。通过设计Simulink 3D Visualization的3D环节实现路径的设计，然后结合FCS的图像处理部分实现路径识别并且将其转换为飞行指令。

根据一般的，各个模块的作用总结如下图：

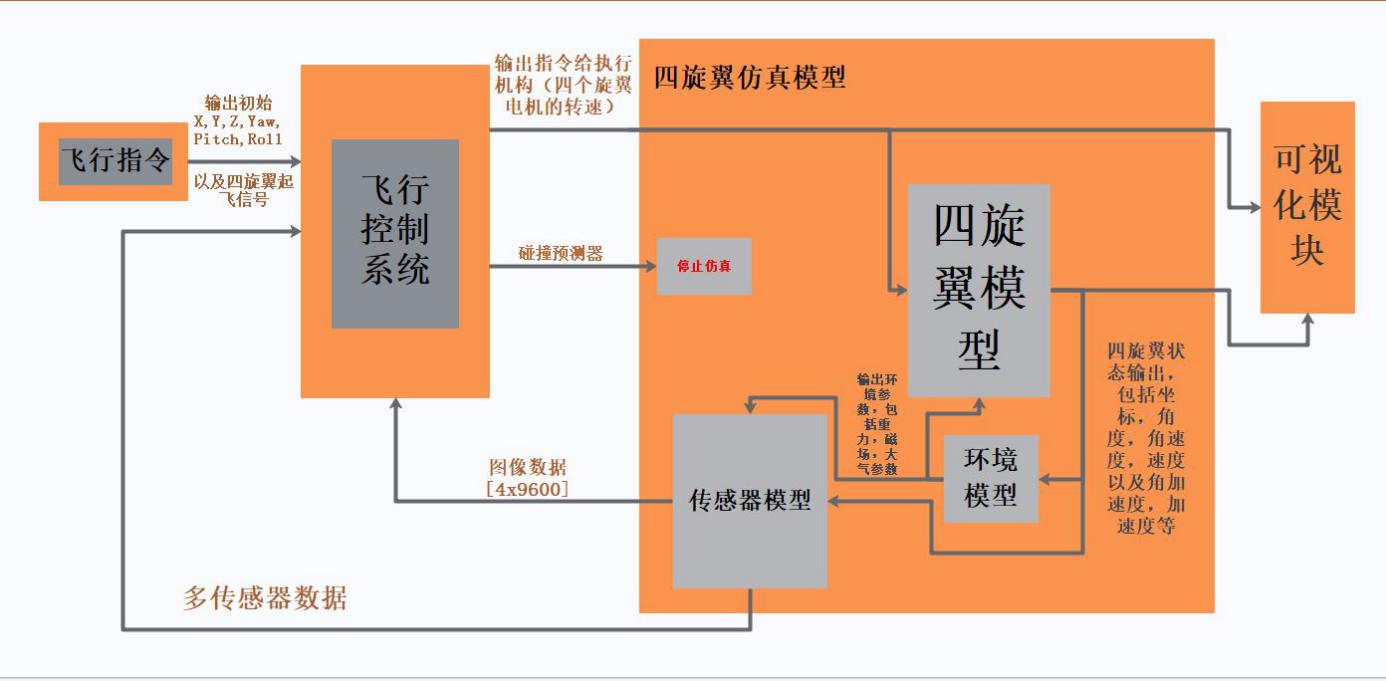
****

图3-2-2 整体仿真系统说明

3.3.1 可视化模块

该模块是用于可视化无人机飞行的整体轨迹，如下图：

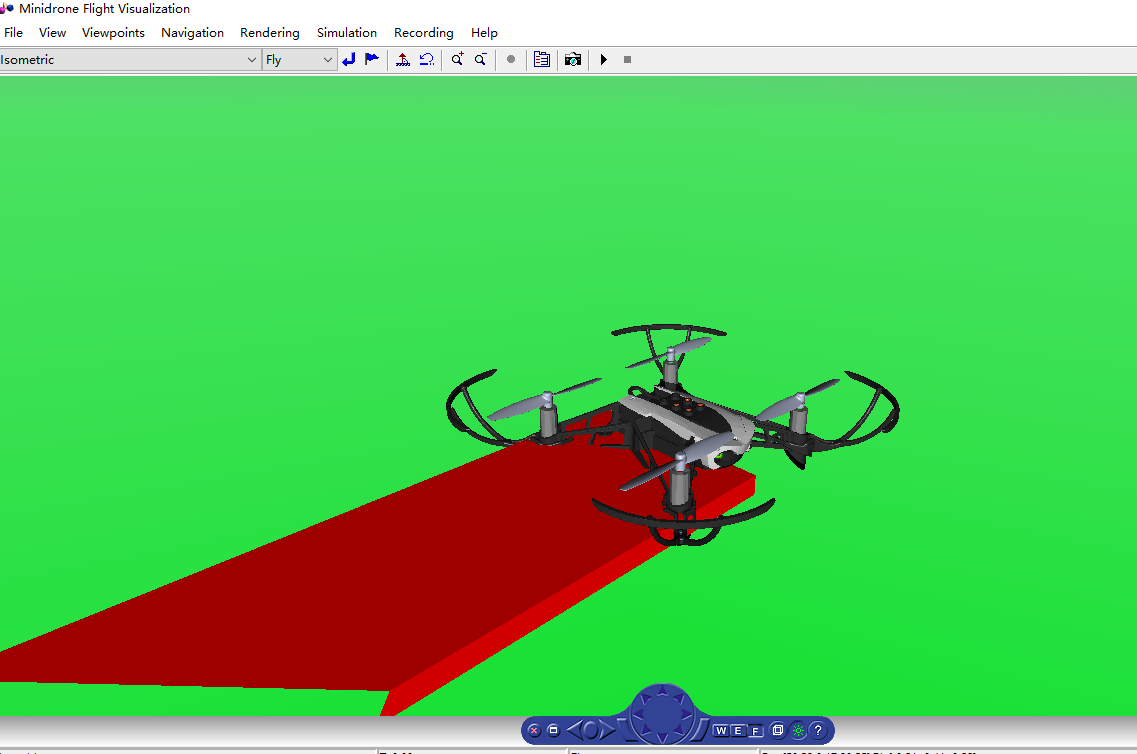


图3-2-3 可视化模块示意图

3.1.1 FCS模块设计

该模块设计是整个飞行控制以及图形识别的重要模块。

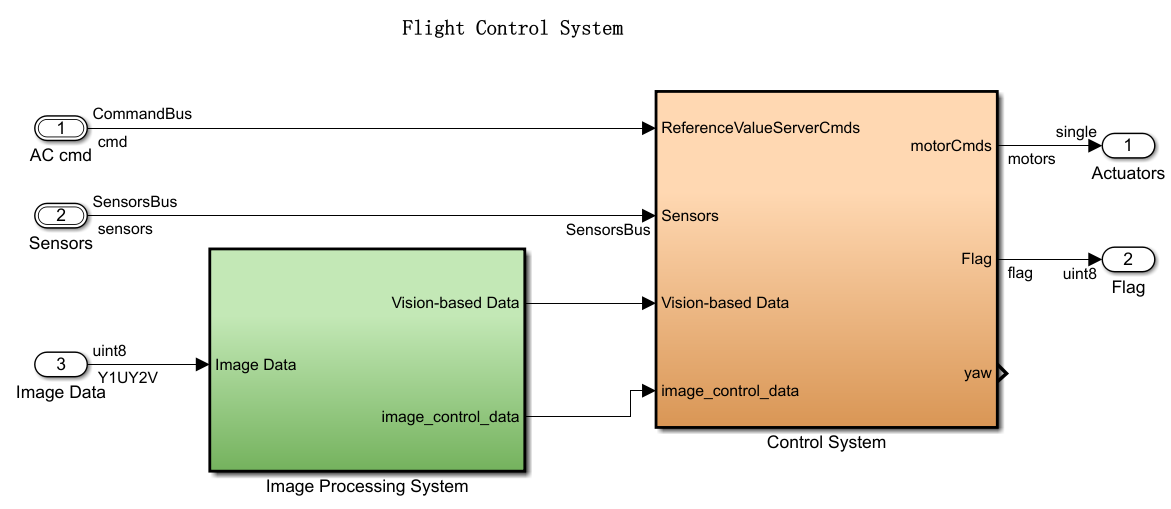


图3-2-4 FCS飞行控制系统整体图

在此处先分析Control System系统。重要的是令图像数据作为次要输入，而Command Bus是主要控制命令。在Control System, 里面包括了基于传感器数据的状态估计算法以及模型文件自带的PID控制算法。即State Estimator模块和Controller模块。



图3-2-5 飞行控制系统控制部分

基于Path Planning模块设计为基于图像数据的飞行。

基于State Estimator模块可以设计状态估计器，这是实物飞行的重要步骤，但在Simulink仿真的过程中可以保持不变。

基于Controller模块可以设计不同算法。此处采用PID算法。

下文描述的三种视觉控制算法是对上述的FCS的“Image Process System”和“Control System”中的“Path Planning”进行修改。

3.3数据处理方法

以上六个参数，分别通过以下方法获取并处理：

1. 整体飞行轨迹

将飞行后的轨迹的坐标读取，并将期望的红色路径进行二值化，把轨迹和期望的红色路径放在同一张图中。

1. 整体飞行时间

测量从起飞到降落所花费的所有时间，读取Simulink中的运行时间。

1. 整体误差

根据参数5（三个弯道的震荡程度）同样的方法测量每一个时刻的误差，然后进行累加。

1. 三种弯道的超调量

以上三种方法均是基于“目标角度”与“实际角度”的误差，通过PID调控来消除误差，因此以“目标角度”与“实际角度”的误差作为控制量，记作：

超调量指的是绝对值最大值。

1. 三种弯道的震荡程度

震荡程度使用方差来计算：

1. 三种弯道的峰值时间

峰值时间取超调了最大值所在的时间点。数据处理中会将时间\*3倍，以使得坐标轴相等。

1. 三种弯道的稳定时间

经过测定，大约在偏差角度在0.3°以内，则对行走完全无影响，故取稳定在0.3°以内的第一次时间为稳定时间。数据处理中会将时间\*3倍，以使得坐标轴相等。

同时，将路径弯道划分为三个等级



图3-3-1 微小角度：0-10°



图3-3-2 中等角度：10-45°

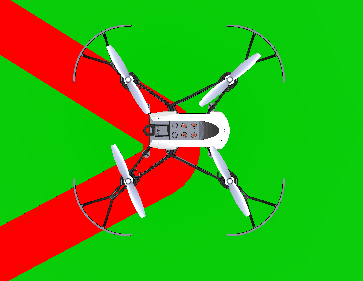


图3-3-3 大角度：45°以上

对于以下每个算法进行上述六个参数的测量。

3.4 视觉控制算法设计思路

根据上节所描述的视觉控制的任务，要在此基础之上设计视觉控制算法。

经典的视觉控制算法有以下：

基于位置的视觉伺服（PBVS），基于图像的视觉伺服（IBVS），PBVS的为三维参数，需要从图像中读取。IBVS的为特征点，即在图像中目标点的坐标。本文采用的是IBVS的控制思想。

核心的控制思路是，对路径进行准确的识别，然后根据无人机的位置以及识别出来的路径，通过某种算法获得准确的航迹点，这个航迹点也就是用来控制无人机飞行的目标点。获得航迹点之后，再根据航迹点使用飞行控制算法来实现跟踪。

下面为其控制思路的流程图：

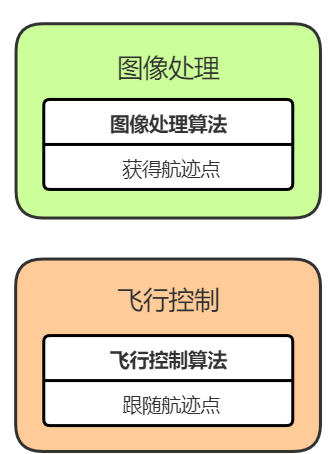


图3-4-1 整体控制的流程图

下面为其控制框图：

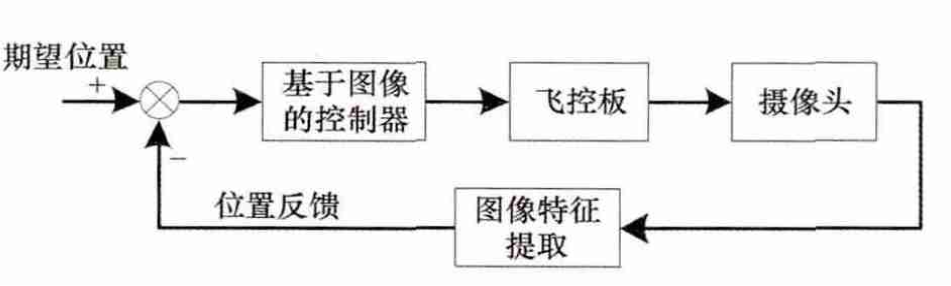


图3-4-2 整体控制的程序框图

下面为其Simulink控制程序框图：

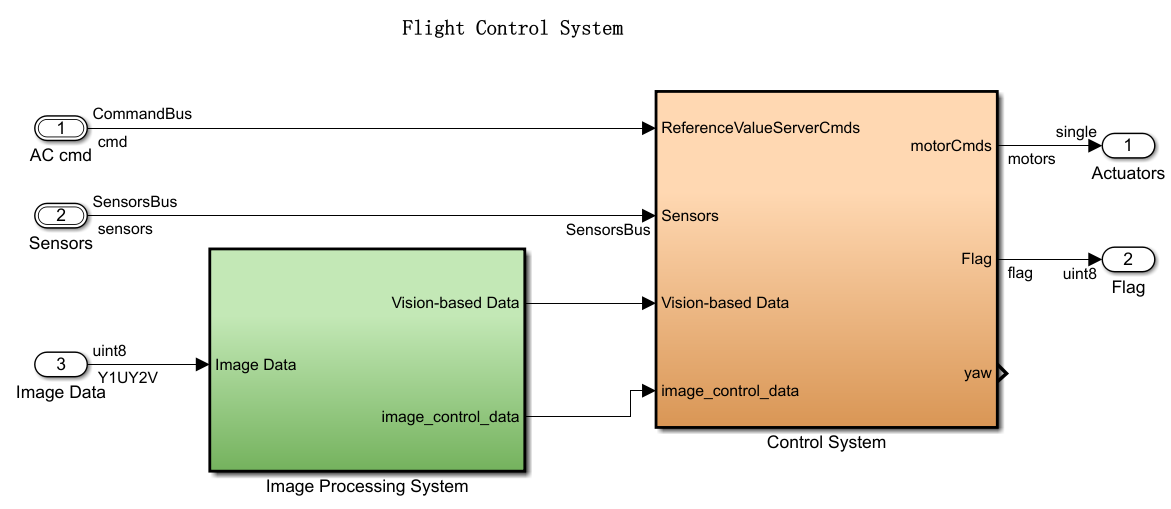


图3-4-3 整体控制的Simulink程序框图

任务的核心是对航迹点的获取。下文基本围绕航迹点的获取，使用了三种方式，分别为：三圆轨迹生成算法、散点探测循迹算法、重心循迹算法。下面对每一种方式进行详述。

3.5 三圆轨迹生成算法

前文提到在飞行系统的视觉控制当中，有两部分非常重要，第一部分是图像的处理，也就是获得对应的航迹点，第二部分是飞行控制，也就是对航迹点的跟踪。在“三圆轨迹生成算法”中，图像处理所使用的方案为“形态学提取算法”，通过形态学的腐蚀处理有效地获得路径轨迹。对于轨迹规划，此方法纠正了飞行姿态用来跟踪轨道，构建了“三圆轨迹生成算法”以获得超前控制，构建了“分级调控算法”让控制航迹点更光滑。控制器使用PID控制。

整体的算法逻辑如下：

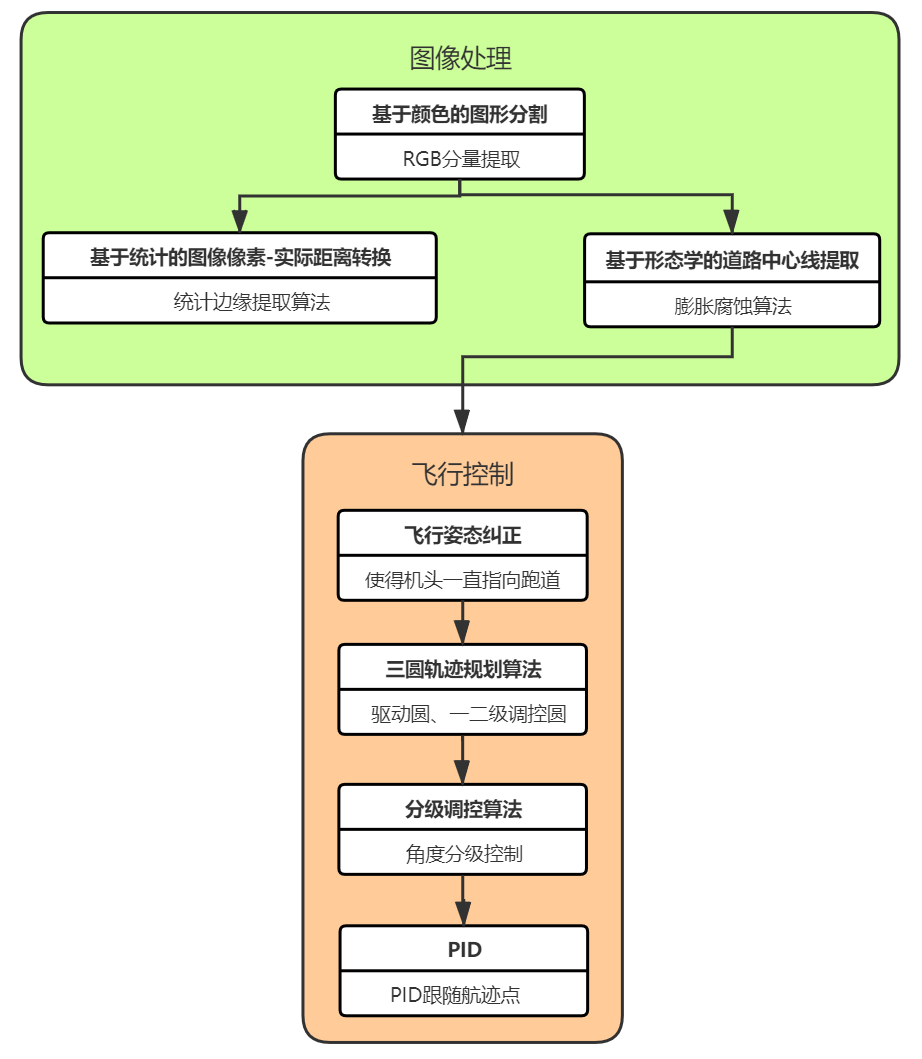


图3-5-1 三圆轨迹生成算法的流程图

3.5.1图像处理算法

图像处理是将相机得到的图像数据进行处理分析，最终得到轨道的轨迹序列，为下一步飞行路径规划及控制做准备。其中，对于图像需要先进行颜色分割，然后通过形态学的腐蚀处理有效地获得路径轨迹，同时需要把图像坐标和实际坐标进行转换。

1. 基于颜色的图像分割：

对获取到的摄像机数据，即RGB值进行阈值分割，得到需要的红色轨道的二值图像。阈值分割后，获得二值化的黑白图，用于进一步分析。算法逻辑如下：

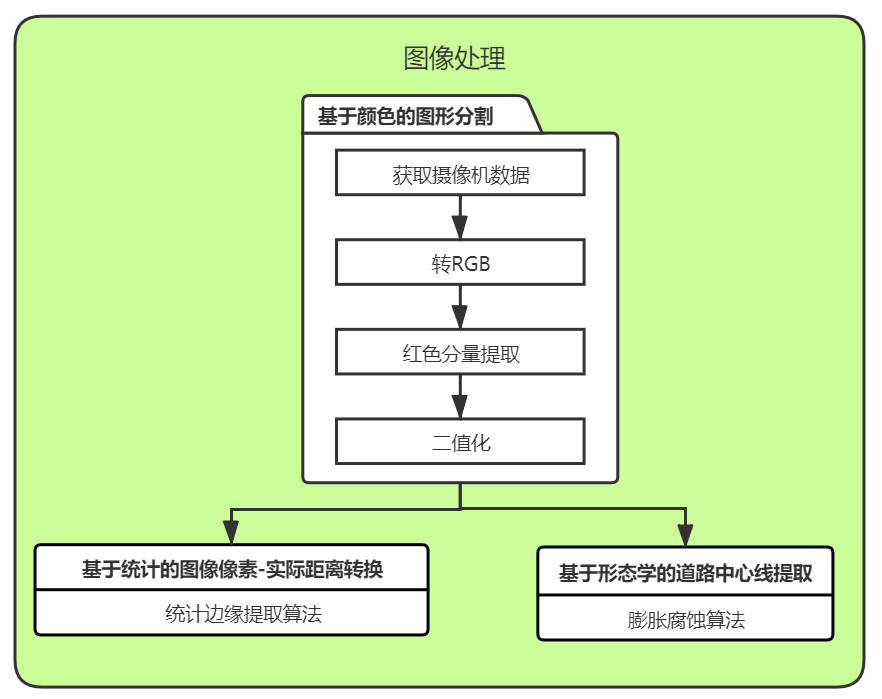


图3-5-2 三圆轨迹生成算法视觉部分的流程图

算法处理结果如下

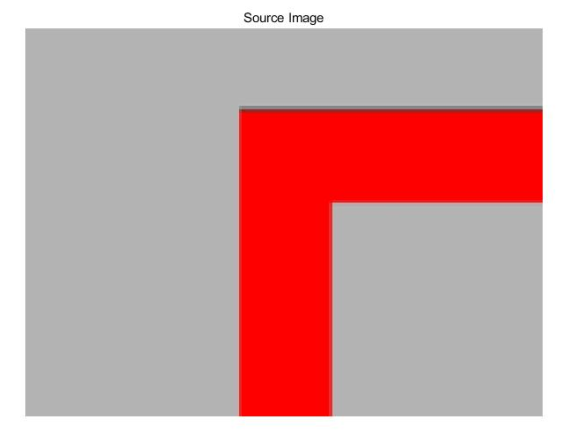


图3-5-3 道路原图

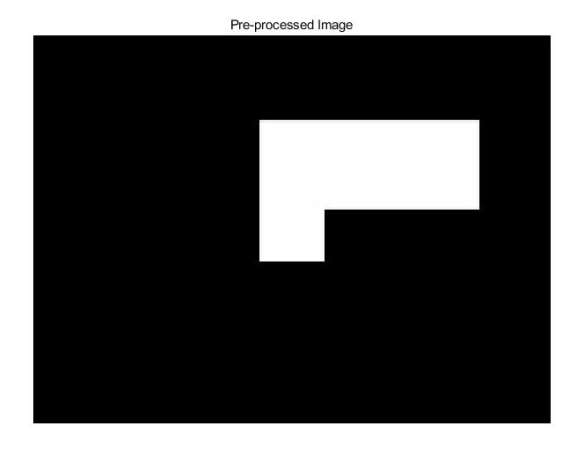


图3-5-4 二值化图像

1. 基于形态学的道路中心线提取

对红色二值化的轨迹，进行形态学提取，获得其中心轨迹，以用于下一步控制。下面解释形态学提取原理。

数学形态学的核心是用一定形态结构的元素去影响图像中的特征值，具体的影响方法为提取以及度量，以实现图像识别。在二值化的图像中，形态学的基本运算包括击中击不中变换、膨胀以及腐蚀。

其中，击中击不中变换经常被使用作图像的细化，

设S表示图像，B是结构元素，且B由两个不相交的部分和占组成，即B满足

则S被B击中定义为：

或

本文采用的形态学处理为形态学细化。细化也就是把一片区域转化成一条曲线

在该项目中，用到的形态学处理主要是形态学细化。所谓细化，通常是把一个具有一定面积的区域用一条曲线来代表它。数学表示如下：

图像集合S使用结构元素B进行细化用S⮾B表示。可表示为

采用结构元素序列{B}进行细化，细化过程可表示为

其中，结构元素序列{B} = {B1,B2,B3,…,Bn}。

因此，细化处理过程是先通过使用对进行细化，然后使用经过对进行细化，同时再进行细化处理，一直循环，直到对使用 进行细化，最终得到一条细线。

对于当前项目来说，对刚刚获得的二值化图像进行腐蚀后得到完整的上半部分轨迹。

算法逻辑如下：

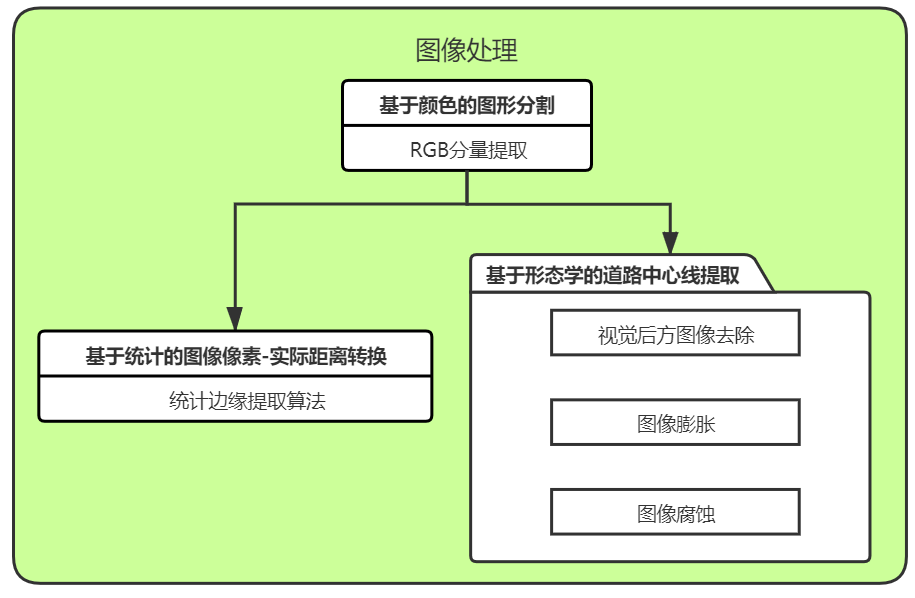


图3-5-5 三圆轨迹生成算法形态学提取的流程图

在Simulink中的程序如下：

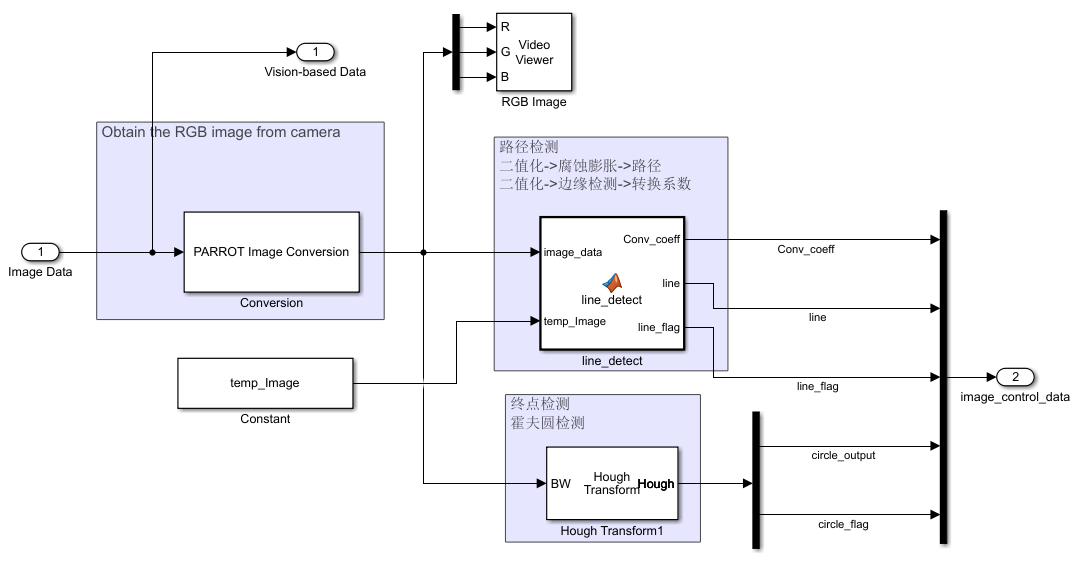


图3-5-6 三圆轨迹生成算法视觉部分的Simulink程序框图

其中，“PARROT image Conversion”用于将原始的图像像素转化为RGB值。“RGB Image”可以将RGB值读取出来，用于观察当前视角。“line\_detect”为核心的路径检测函数（模块内核心代码见附录一），有两个作用，分别为读取中心路径和获得转换系数。“Hough Transform”为霍夫圆检测，用于判断是否到达终点。

算法结果如下：

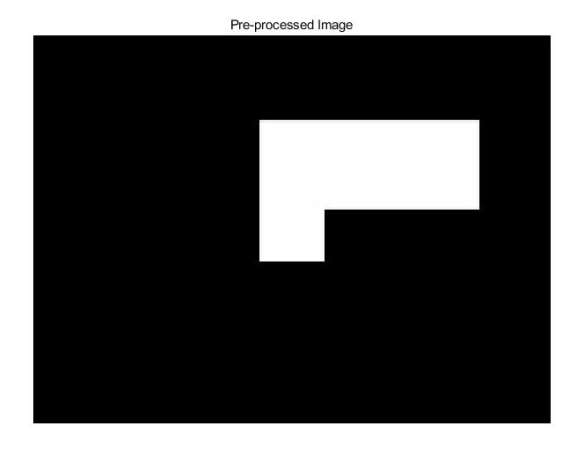


图3-5-7 二值化图像

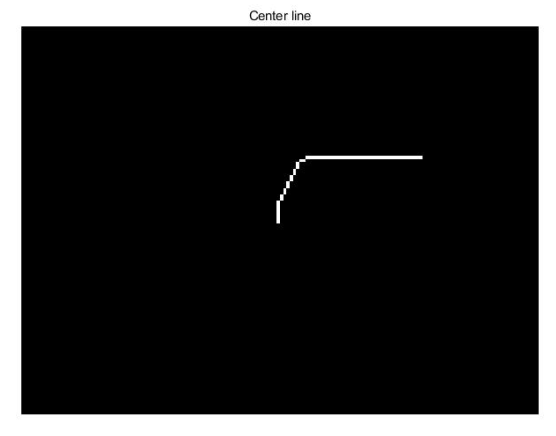


图3-5-8 形态学处理后图像

1. 基于统计的图像像素-实际距离的转换

图像处理后不仅仅需要获得此帧的轨迹，也需要获得的是图像像素和实际距离之间的转换，此处采用统计像素转换法。其原理如下：

本项目中，需要通过摄像头来获取下一时刻的航迹点，那么，其中的问题解决的关键就在图像坐标和实际坐标的转换关系是否确定。因为摄像头固定在无人机的正下方，故关系可通过为相似三角形原理近似求解，即

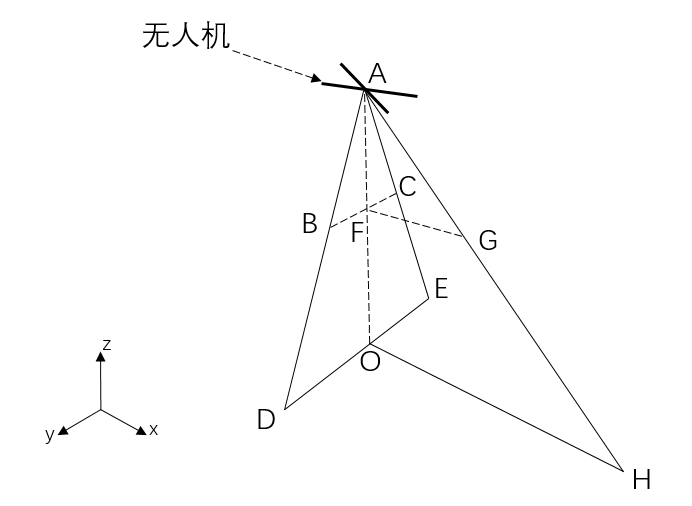


图3-5-9 像素转换原理图

根据相似三角形原理可得，

即

其中，BC指图像中道路宽度的像素距离，FG指图像中航迹点与当前位置的像素距离，DE指道路宽度的实际距离，OH指航迹点与当前位置的实际距离。具体的算法逻辑如下：

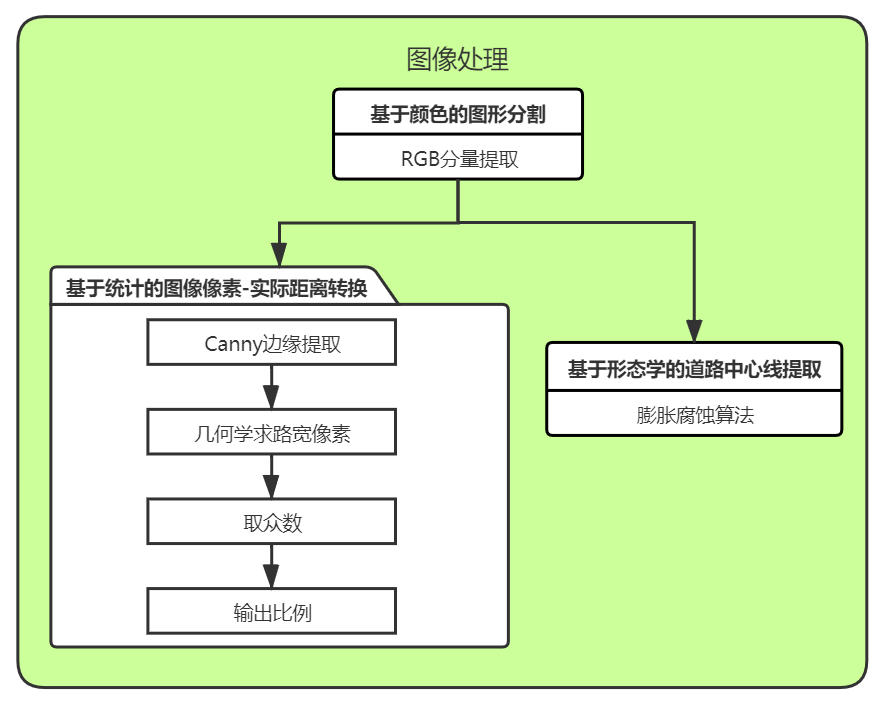


图3-5-10 三圆轨迹生成算法视觉部分距离转换流程图

算法结果如下：

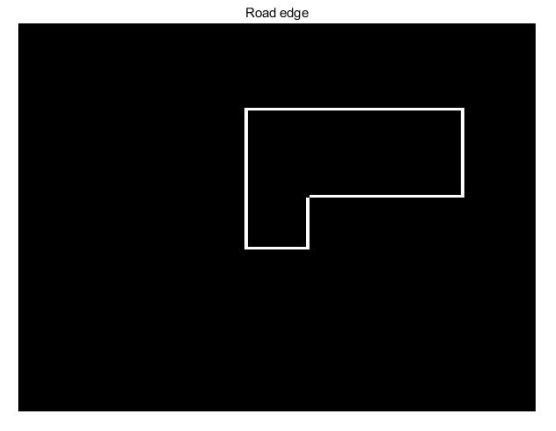


图3-5-11 边缘提取结果

对原本轨道进行边缘提取，获得轨道长度以及对应的转换系数。

3.5.2飞行控制算法

前面的图像处理最终得到当前帧的中心线轨迹。根据此轨迹设计了一套“三圆轨迹生成算法”以及“分级调控算法”把轨迹变为能驱动无人机前进的一个个“航迹点”，然后用控制器追踪“航迹点”，以达到完整控制效果。

轨迹规划的目的是某一帧的中心线轨迹其实是不可以直接用于控制，因为在图像处理得到的大角度转角是不考虑飞机模型的限制，因此直接将轨迹用于控制会导致超大的超调。如下图所示。因此我们需要对每一帧的中心线轨迹进行规划，以获得适合控制的“航迹点”，并用此“航迹点”来求出飞机飞行速度和转角的角度。主要用到了两种方法：三圆轨迹生成算法以及分级调控算法。算法控制流程如下：

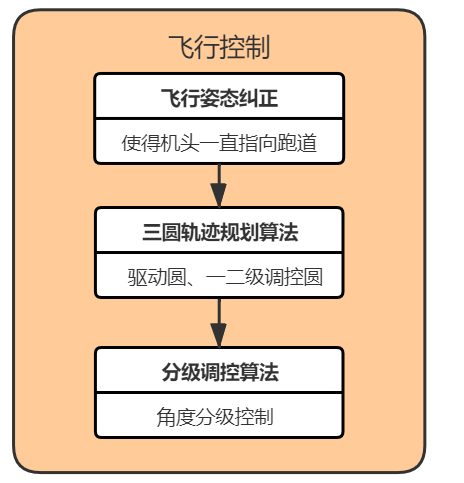


图3-5-12 三圆轨迹生成算法控制部分流程图

1. 飞行姿态纠正

使用轨迹控制，最好获得与轨道相同的方向，因此对飞行姿态进行纠正，具体为把机头方向一直对着轨道中心线指向。如下图所示：

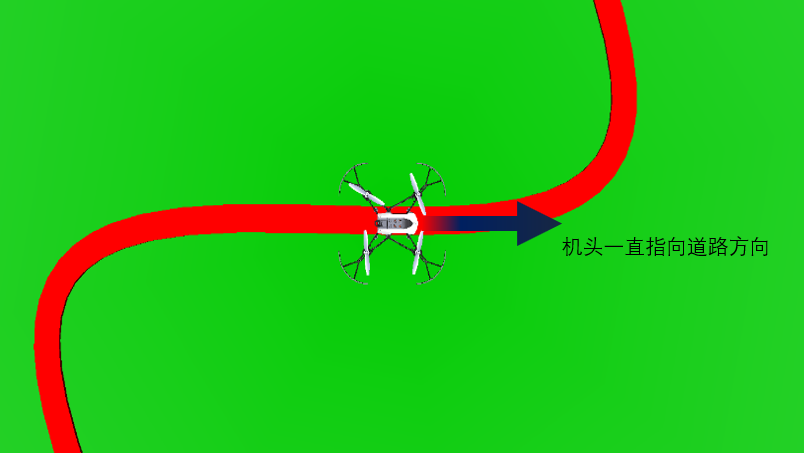


图3-5-13 飞行姿态纠正示意图

1. 三圆轨迹生成算法

通过大量实验发现，无人机能否更好跟踪轨迹很大程度上取决于转角能够给得恰当，若一次把图像中的转角用于控制，则会有大超调；若缩小转角，则跟不上转弯而飞出轨道，因此需要解决三个问题：超前规划，转角插值，速度规划（转角速度不同于直道速度）。

三圆轨迹生成算法原理图如下：

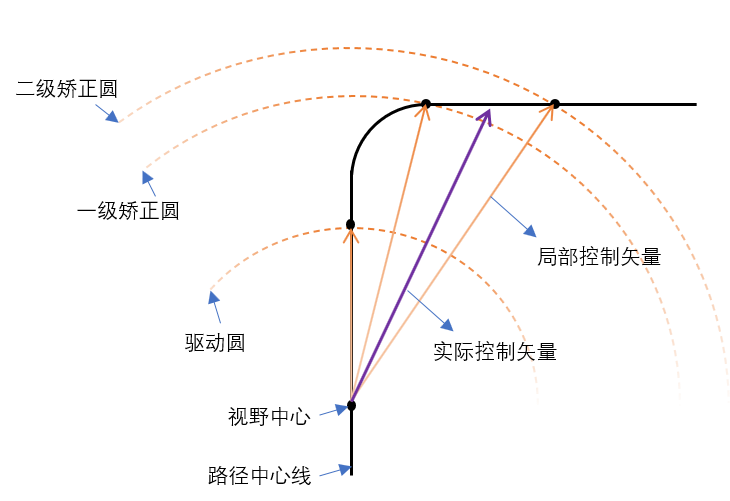


图3-5-14 三圆轨迹生成算法示意图

把当前帧获得的轨迹进行分割规划，分为“驱动圆”“一级矫正圆”“二级矫正圆”。其中驱动圆为离无人机视觉中心最近的圆，切割轨迹得到“驱动航迹点”，但是仅仅以此航迹点为驱动会缺乏预测性，无法提前转弯，依据同样的方法切割得到“一级矫正航迹点”“二级矫正航迹点”，将三个航迹点进行加权后获得“控制航迹点”。

按照前文的约定，以逆时针为正方向，记坐标系中三个圆与轨道相切的角度分别为。同时对进行加权操作：

其中为“敏感度”，由实验法测定。

为了获得更加光滑的控制效果，将多帧的“控制航迹点”进行加权，获得最终的“控制航迹点”。

1. 分级调控算法

当获得“控制航迹点”后，需要将此航迹点“缓慢”地给控制器进行跟踪，因为如前文所述，无人机无法立刻反应跟踪此点，如果一次给过大转角，则会导致角速度和角加速度过大，导致转动惯量过大，无人机无法及时停止，因而造成超调。因此构建了分级调控算法，用于实现对于“大角度”则降速快转（但仍然比原来的转速慢得多），“小角度”则加速慢转的功能。分级后给出最终的“控制航迹点”，是一个光滑连续的序列，接下来会传到控制器进行跟踪控制。

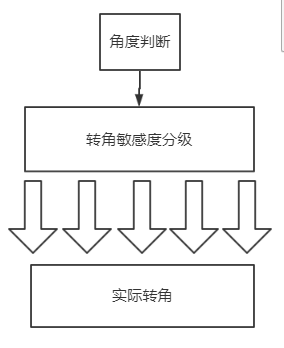


图3-5-15 分级调控示意图

原理图如下：

也就是改变的参数，由实验法测得。最终的角度控制参数：

在Simulink中的程序如下：

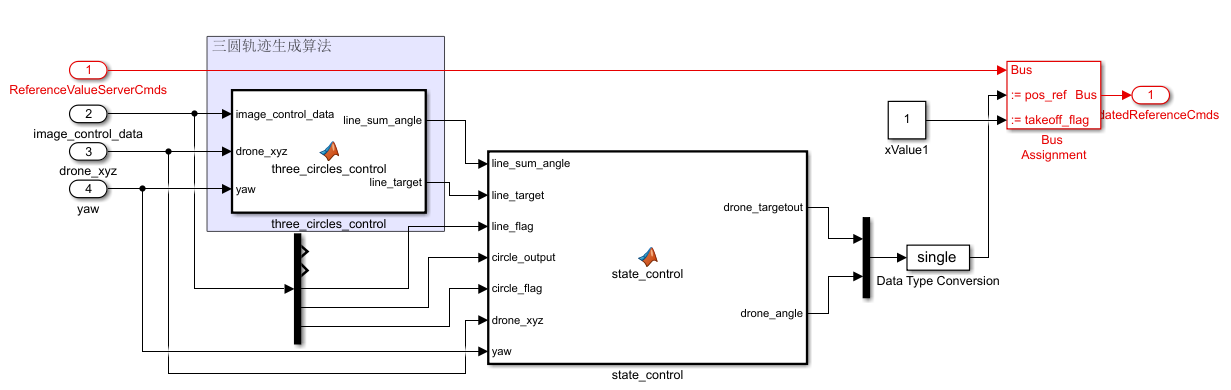


图3-5-16 三圆轨迹生成算法控制部分Simulink程序框图

其中，“three circles control”模块为核心算法，使用“三圆轨迹生成算法”来对航迹点进行跟踪（模块内核心代码见附录二），“state control”模块可以判断当前是否检测到终点，用于状态的转换。整体的输出为需要控制的角度误差，下一步由PID来消除误差。

1. PID控制

PID控制算法有两种模式：位置式和增量式，这里采用的是增量式的双闭环PID控制，如下式所示：

其中为PID控制器的输出，为比例、积分、微分参数，为期望与实际误差，在这里为：

经典的PID控制框图为：

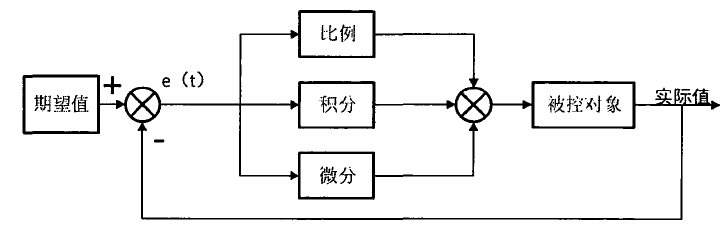


图3-5-17 PID控制示意图

在Simulink中的程序如下：

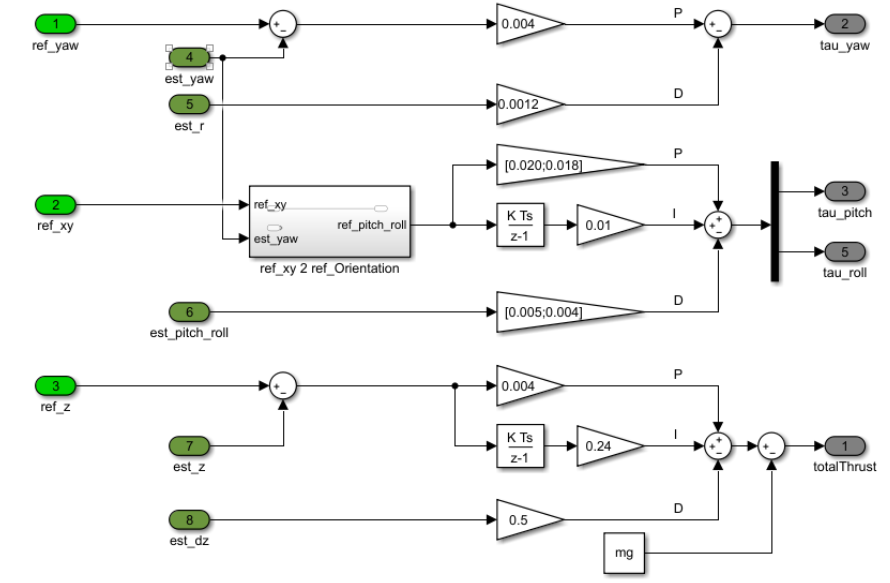


图3-5-16 三圆轨迹生成算法控制部分PID的Simulink程序框图

3.5.3仿真数据分析

整体飞行路径为：

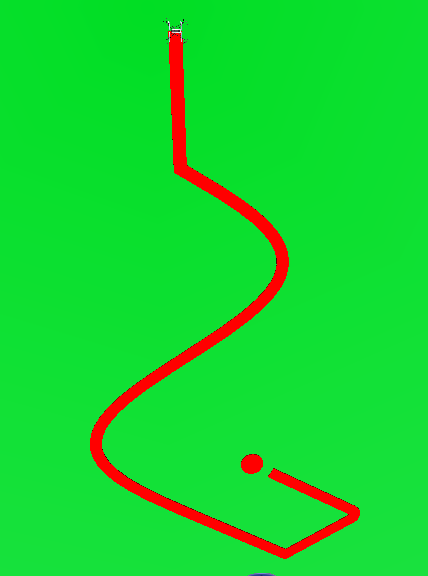
  

图3-5-17 飞行路径俯视图 图3-5-18 飞行路径形态学处理图 图3-5-19 反向操作图

整体飞行轨迹（黑色为路径，蓝色为实际飞行轨迹）：

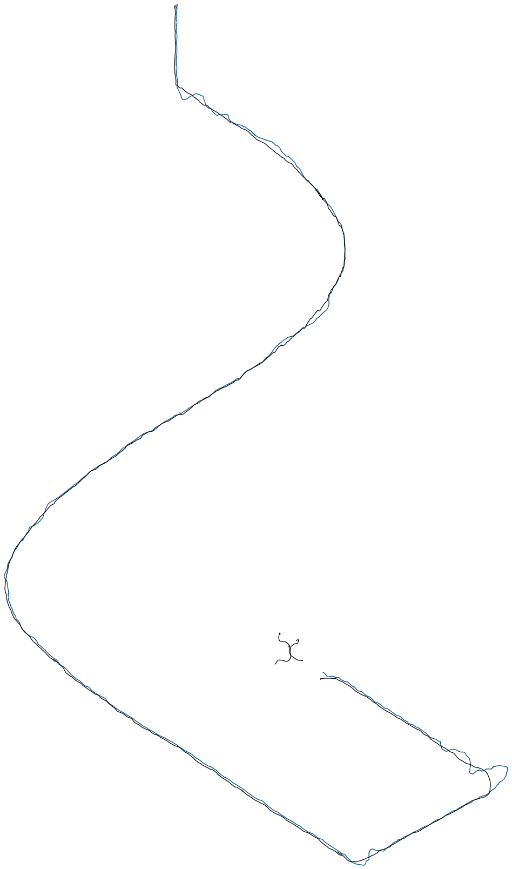


图3-5-20 三圆轨迹生成算法飞行轨迹对比图

整体飞行时间：

飞行五次时间的平均值为：43.68s

整体误差：

通过角度的整体方差计算，飞行五次的平均误差为：137.54

三个不同角度的转弯数据图：

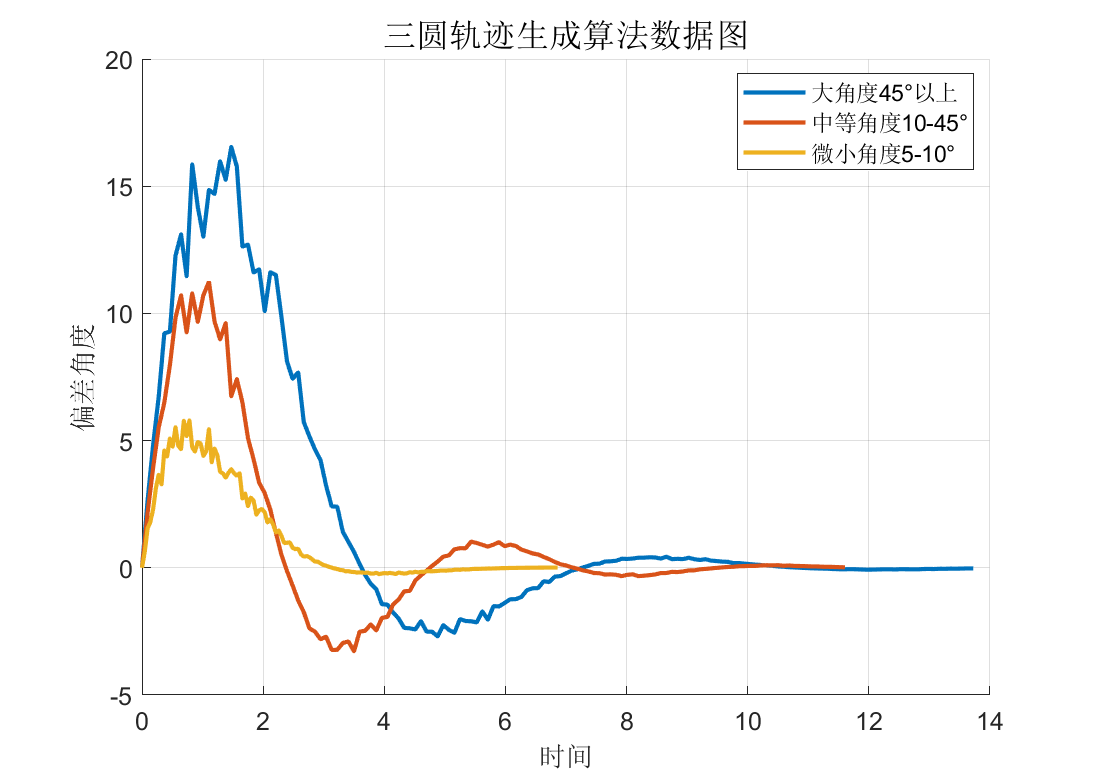


图3-5-21 三圆轨迹生成算法三个不同角度的转弯的数据图

三个不同角度的转弯的参数对比：

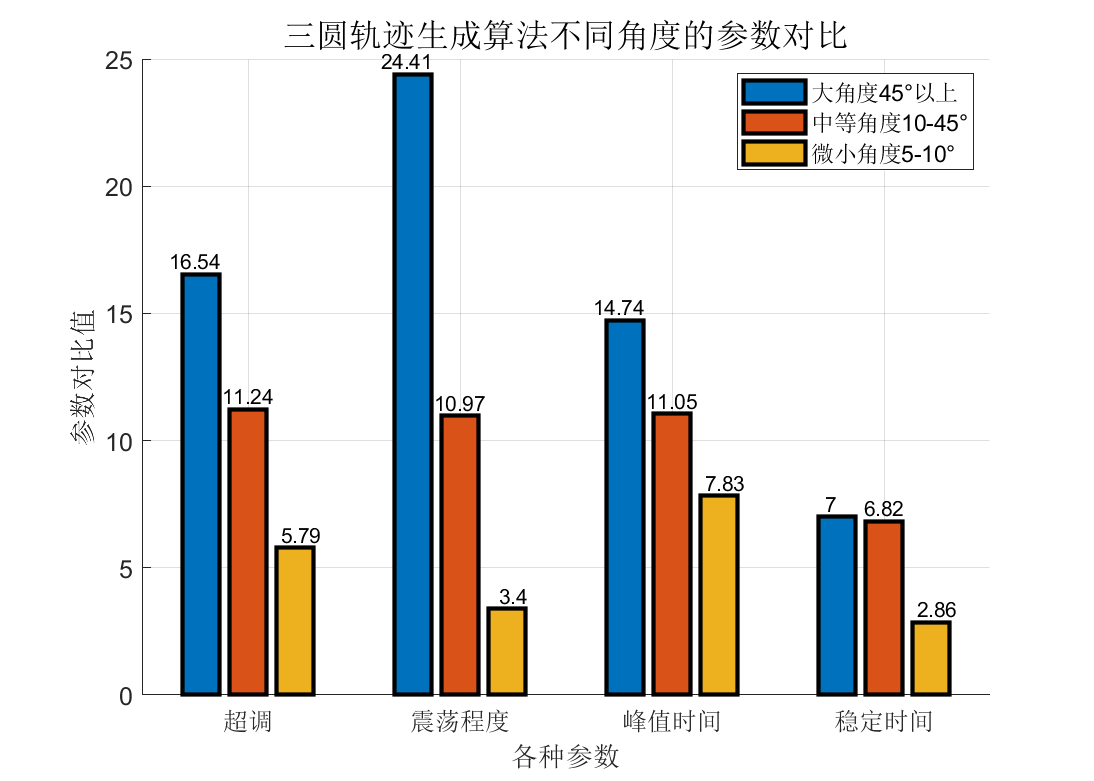


图3-5-22 三圆轨迹生成算法三个不同角度的转弯的参数对比图

可以看出，三圆轨迹生成算法在微小角度上反应非常迅速，而且超调量低，调节时间长。但是对于较大的角度，反应较慢，而且超调很高。

3.6散点探测循迹算法

此方法与上一方法核心差别在于同样是获得距离变换后的中心线后，此法采取散点探测循迹算法来跟踪，优势在于不用转机头，但缺点在于反应较慢，对细微变化不敏感。



图3-6-1 散点探测循迹算法整体流程图

3.5.1图像处理算法

如方法一“三圆轨迹生成算法”中的图像处理相同。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| （1）路径原图 | （2）二值化图像 | （3）边缘检测图像 | （4）距离变换图像 |
| 图3-6-2 散点探测循迹算法图像处理部分 | | | |

散点探测循迹算法逻辑：

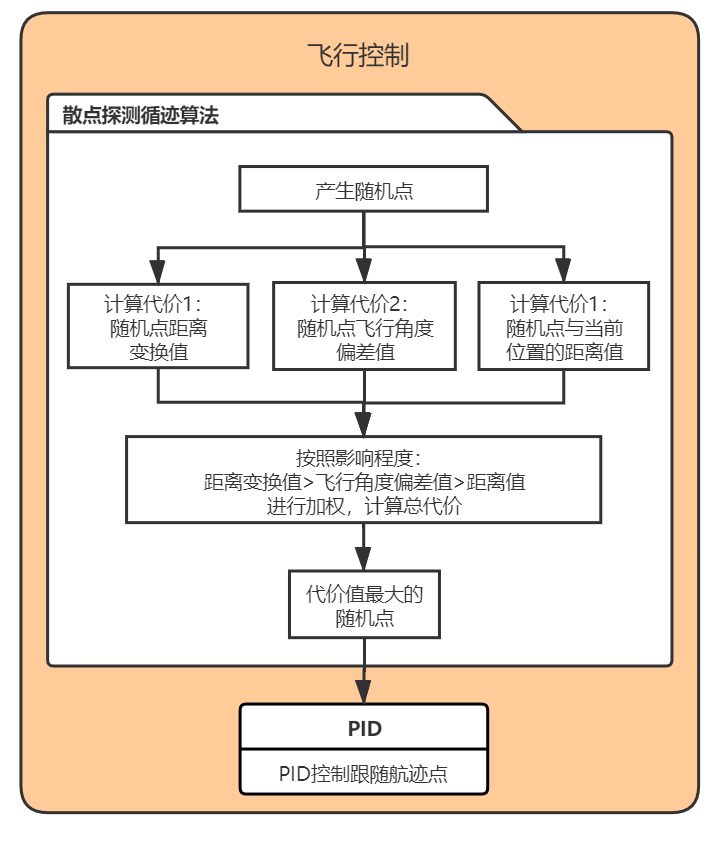


图3-6-3 散点探测循迹算法飞行部分流程图

1. 随机点生成

在图像坐标系内，以无人机当前位置为圆心，以25个像素点为半径，在圆内随机生成1500个点。取极轴正方向为图像坐标系的X轴正向，则随机点的坐标可以用极坐标表示，即：



其中r为生成的随机点在极坐标下的极径，r应小于等于25个像素，为随机点在范围内随机生成的极角。为了保证r小于等于25个像素，可以在[0,1]范围内随机生成k，令



式中R为25个像素。利用此方法可以在以25个像素点为半径的圆内较好的生成1500个随机点。



图3-6-4 区域内1500个随机点

1. 随机点三个指标计算

其中，代价1计算，即随机点的距离变换值：

利用“三圆轨迹生成算法”中距离变换图像生成方法，计算随机点的距离变换值，需要注意的是，越靠近路径边缘的随机点，距离变换值越小，在距离变换图像中越暗，越靠近路径中央的随机点的距离变换值越大，在得到的图像中的越亮，如图3.11所示。因此可以根据距离变换值的大小，判断随机点在路径中央与否，易知，需要挑选的点应是距离变换值较大的点。



图3-6-5 距离变换值较大的点

取距离最小的值，作为当前随机点的距离变换值。

代价2计算，即随机点的飞行角度偏差值：

随机点的极角 ，即随机点与坐标原点连线与X轴正向的夹角，与上一时刻无人机的飞行方向的角度差的大小，可以保证随机点是否在上一时刻无人机的飞行方向上。通过选取角度差较小的随机点，可以使得无人机的运动方向不会发生突变，可以解决无人机的往复运动及振荡问题，能够使得无人机平稳顺滑的运行并度过各种弯角。

代价3计算，即随机点与当前位置的距离值：

计算随机点到无人机当前位置的距离，即随机点在极坐标系下的极径，可知这个距离最大为25个像素点。选取的点距离无人机当前位置越大，则无人机下一时刻在路径上前进的越远，即沿着路径飞行的速度越快。

1. 随机点三个指标加权

我们将距离变换值以及与当前位置的距离值以及角度偏差时进行加权。其中经过实验发现，与当前位置的距离值影响最大，其次是角度的偏差值最后是距离变换值。

取出加权后的最大值点，将其作为航迹点进行控制。

在Simulink中的程序如下：

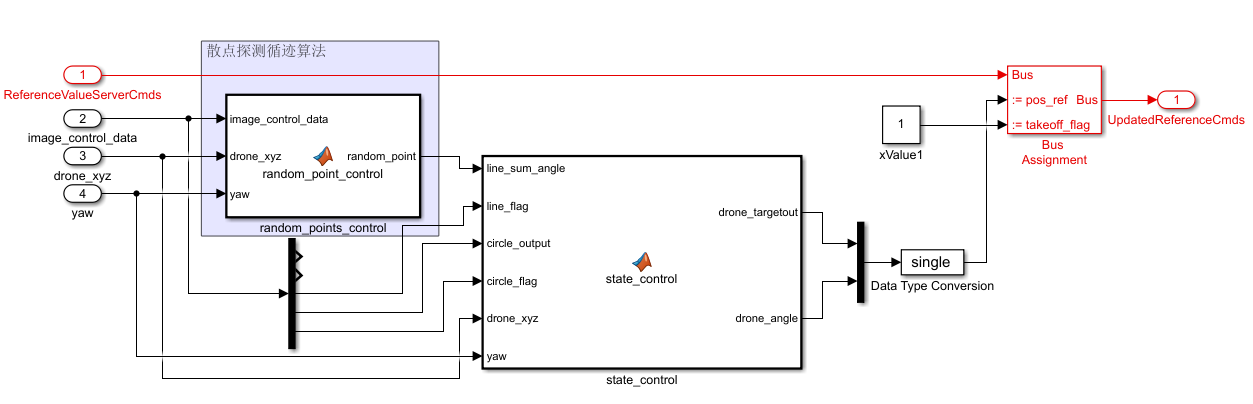


图3-6-6 散点探测循迹算法飞行部分Simulink程序框图

其中，“random points control”模块为核心算法，使用“散点探测循迹算法”来对航迹点进行跟踪（模块内核心代码见附录三），“state control”模块可以判断当前是否检测到终点，用于状态的转换。整体的输出为需要控制的角度误差，下一步由PID来消除误差。

3.5.2 飞行控制算法

飞行控制算法使用PID，详细请看“三圆轨迹生成算法”。

3.5.2 仿真数据分析

整体飞行轨迹（黑色为路径，红色为实际飞行轨迹）：

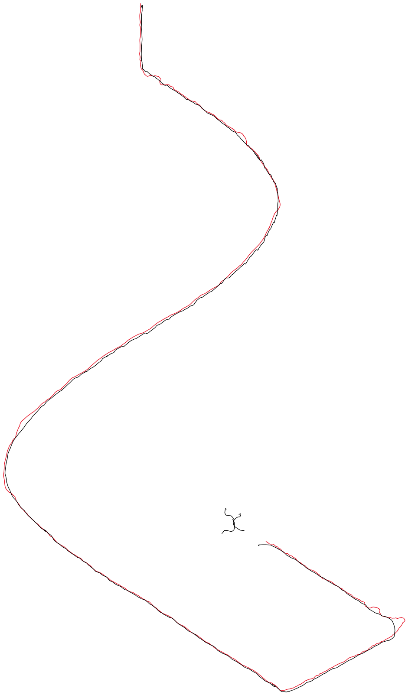


图3-6-7 散点探测循迹算法飞行轨迹对比图

整体飞行时间：

飞行五次时间的平均值为：39.18s

整体误差：

通过角度的整体方差计算，飞行五次的平均误差为：157.67

三个不同角度的转弯数据图：

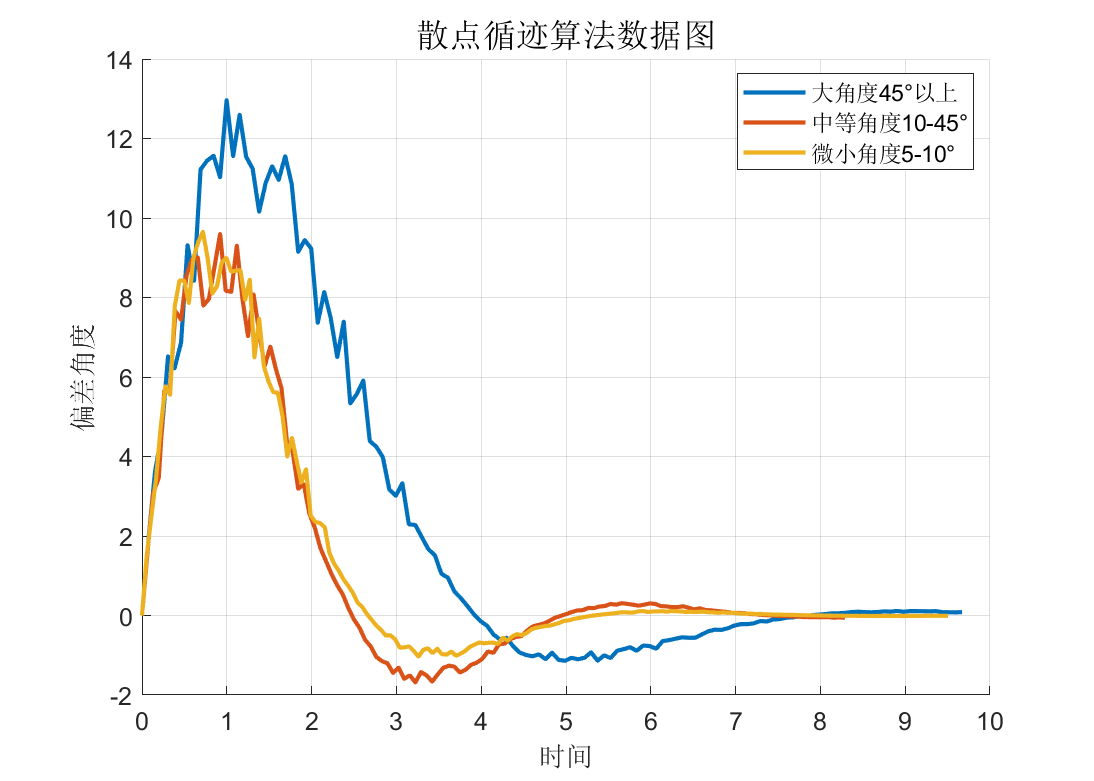


图3-6-8 散点探测循迹算法三个不同角度的转弯数据图

三个不同角度的转弯的参数对比：

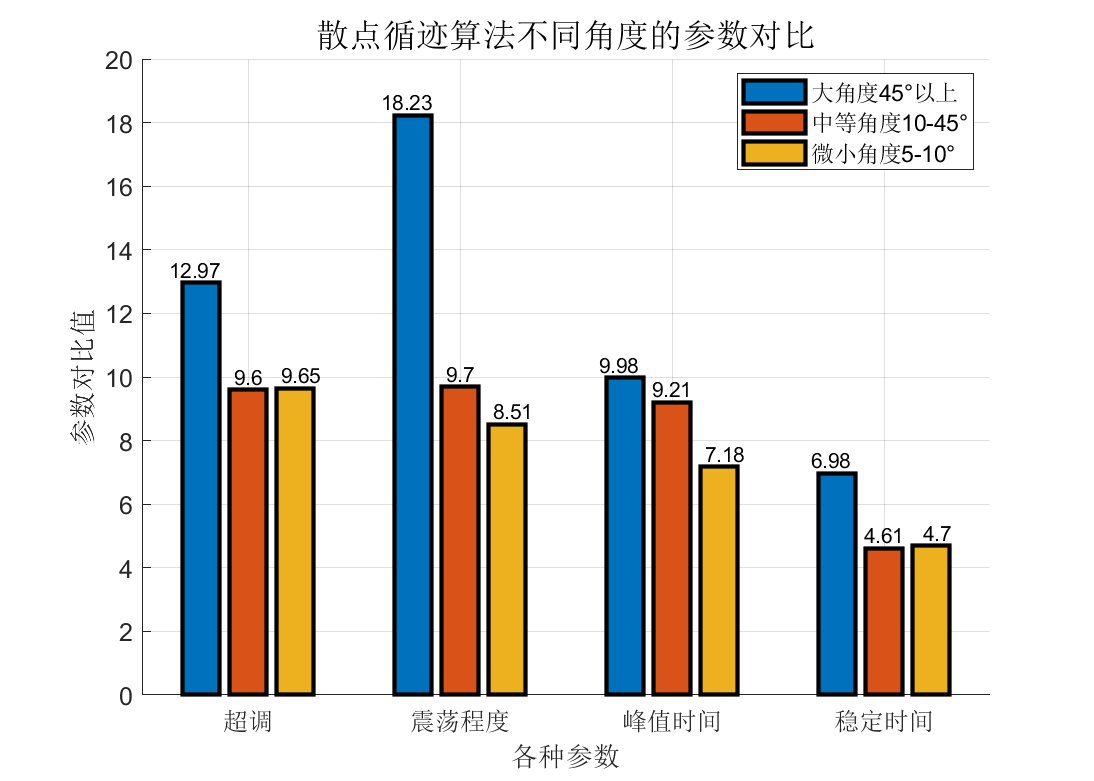


图3-6-8 散点探测循迹算法三个不同角度的参数对比图

可以看出，三圆轨迹生成算法在微小角度上反应非常迅速，而且超调量低，调节时间长。但是对于较大的角度，反应较慢，而且超调很高。此方法与上一方法核心差别在于同样是获得距离变换后的中心线后，此法采取散点探测循迹算法来跟踪，优势在于不用转机头，但缺点在于反应较慢，对细微变化不敏感。

3.7重心循迹算法

重心循迹算法改变了图像处理的方式以及驱动控制方式。图像识别主要部分包括图像分割、高斯分布处理、图像块识别三个步骤。同时在飞行控制中，由于路径路线是变化的，因此用统一速度全程飞行的话，过快可能在转角会脱离路径，过慢则浪费了直道的时间。我们通过在不同形状路径给定不同的速度，使得无人机在直道和小弧度转弯处能飞得更快一些。整体结构如下：

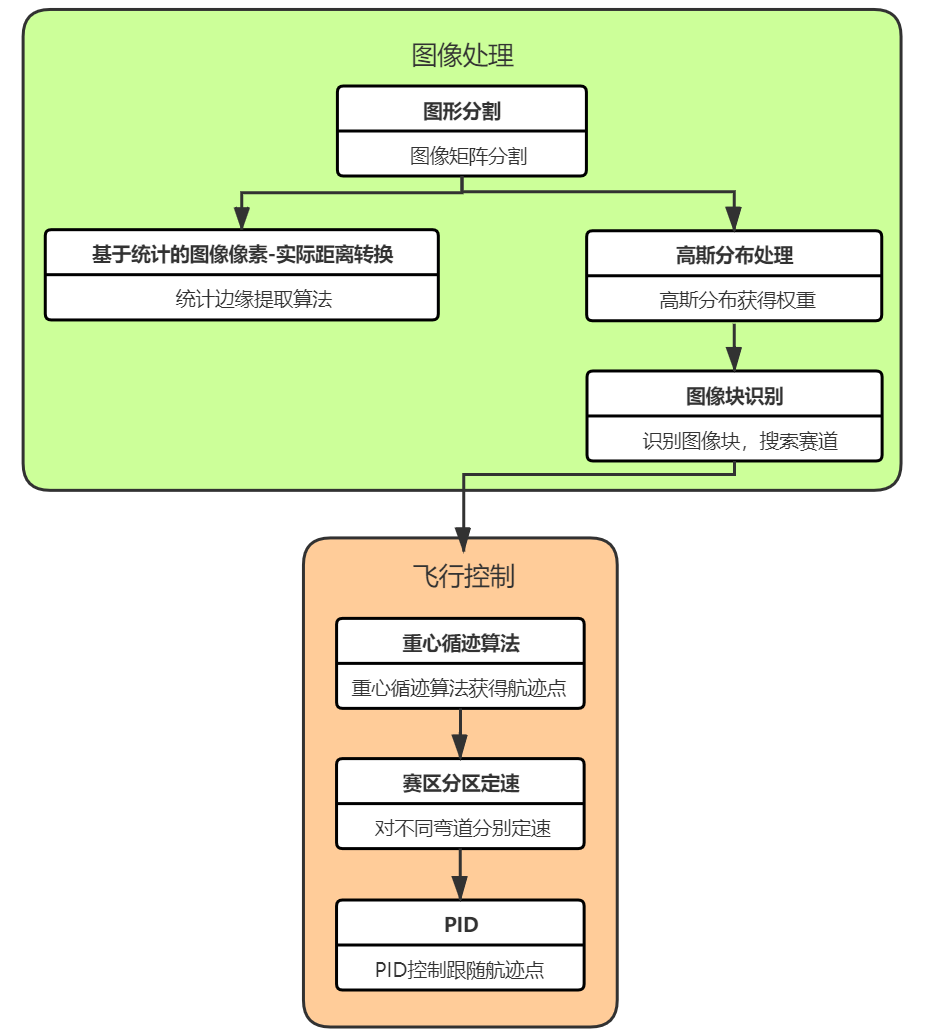


图3-7-1重心循迹算法整体流程图

3.3.1图像处理算法

图像块识别的目的是搜索路径，通过检测机身位置周边的矩阵来确定路径的位置和方向。搜索分两个方向进行，一个是对机身所在区域的邻域进行搜索，另一个是取邻域中两两相邻的矩阵块的中间区域进行搜索，保证无人机前进方向是往路径信息最多的方向。

1. 图像分割

在得到了 RGB 图像之后，即可开始进行图像的分割。原始图像的大小是 120×

160 的，如果是直接利用原始大小的图像进行处理，那么庞大的运算量有可能导致无人

机的控制出现难以预料的情况，因此在这里采取了图像分割的方法。为了进一步精简

运算量，将图像分割部分的输入图像设置为二值化的图像，考虑到路径的实际情况，

故将该输入图像选为 RGB 图像中的 R 通道经阈值处理后的二值化图像，在该通道上几

乎包含了所有的路径信息，也包含了背景信息，因为在该通道上灰度值很小的像素即

可认定为背景，而灰度值很大的像素即可认定为路径。

图像分割的操作比较简单，对输入的灰度图进行分割，分割成若干个 10×10 的区

域，然后对每个区域内的像素值进行求和，进而形成 12×16 的矩阵，此即为图像分割

的最终矩阵。

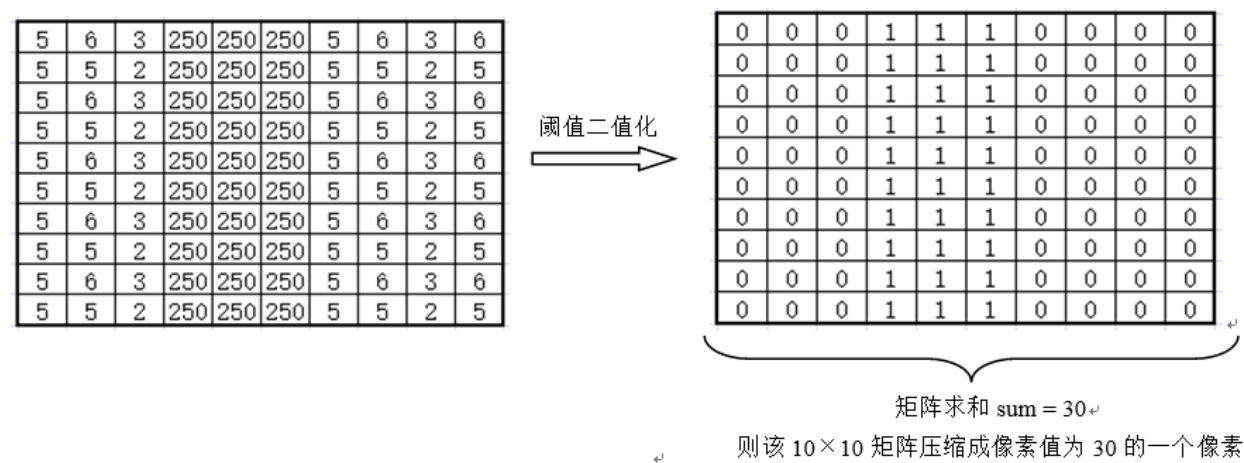


图3-7-2 图像分割示意图

1. 高斯分布处理

高斯分布处理的目的在于改变图像分割后的 12×16 矩阵中每个元素的权重，思路

为先计算 12×16 矩阵的重心位置，然后设定越靠近重心位置的元素的权重越高，越疏

远的元素的权重越低。这样做即可以在下一步图像块识别的时候让无人机偏向图像重

心位置移动。下面是具体实现的方法：

首先是求取图像重心的坐标。依次将每个元素的行索引 a 和列索引 b 组成一组坐

标（a , b），并与该元素的值 n 相乘，得到（a\*n , b\*n），再将所有元素的坐标相加得到（A , B），除以矩阵所有元素的值的求和 N，取整后即可得到重心的坐标。

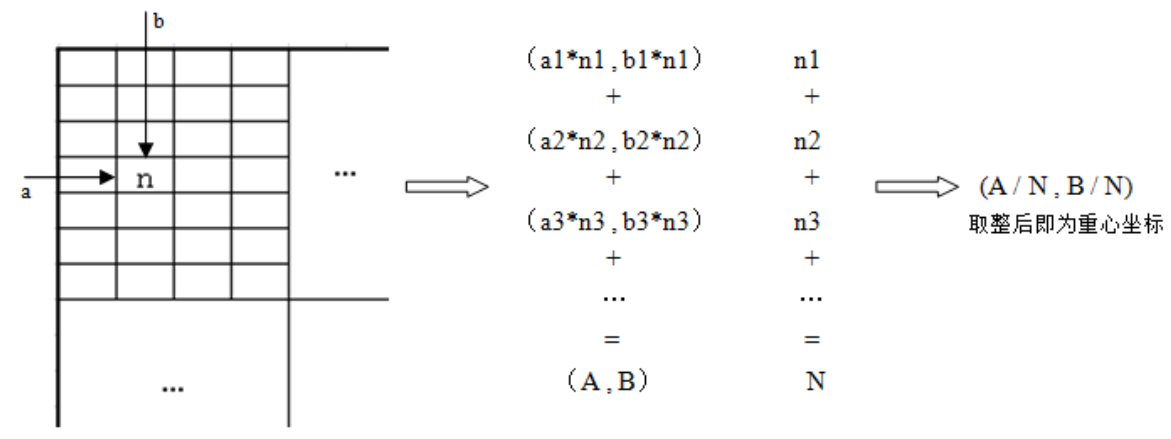


图3-7-3 重心处理示意图

接着是构建高斯分布的矩阵。高斯分布矩阵是呈现二维高斯帽的形式分布，即中

心的数值最大，往边缘方向一圈一圈地逐次减小数值，高斯分布矩阵中的数值即为权

重分布。一般而言高斯分布是非线性递减的，在这里出于简化的目的，采用了线性递

减的方式。然后将重心位置的元素与高斯帽的中心相乘，即重心位置的权重最大，依

次往外权重逐次减小。这样操作的效果就是在处理前的 12×16 矩阵中所有表示路径的

元素的值都相同，但是处理后即使 12×16 矩阵有了方向性，即重心位置的数值会更

大，这样使得图像块识别的处理更为方便和精确。

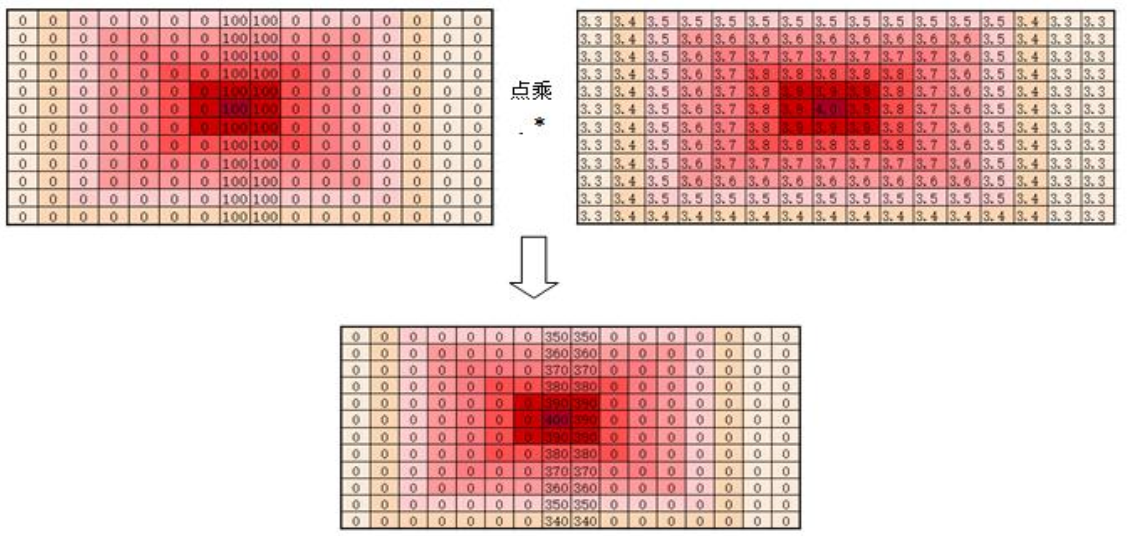


图3-7-4 高斯分布处理示意图

上图即为当无人机在直道上行走时高斯分布处理的过程示意图。

1. 图像块识别

该步骤的目的是搜索路径，即通过检测机身位置周边的矩阵来确定路径的位置和

方向。搜索分两个方向进行，一个是对机身所在区域的八邻域进行搜索，另一个是取

八邻域中两两相邻的矩阵块的中间区域，两种方向如下图所示。

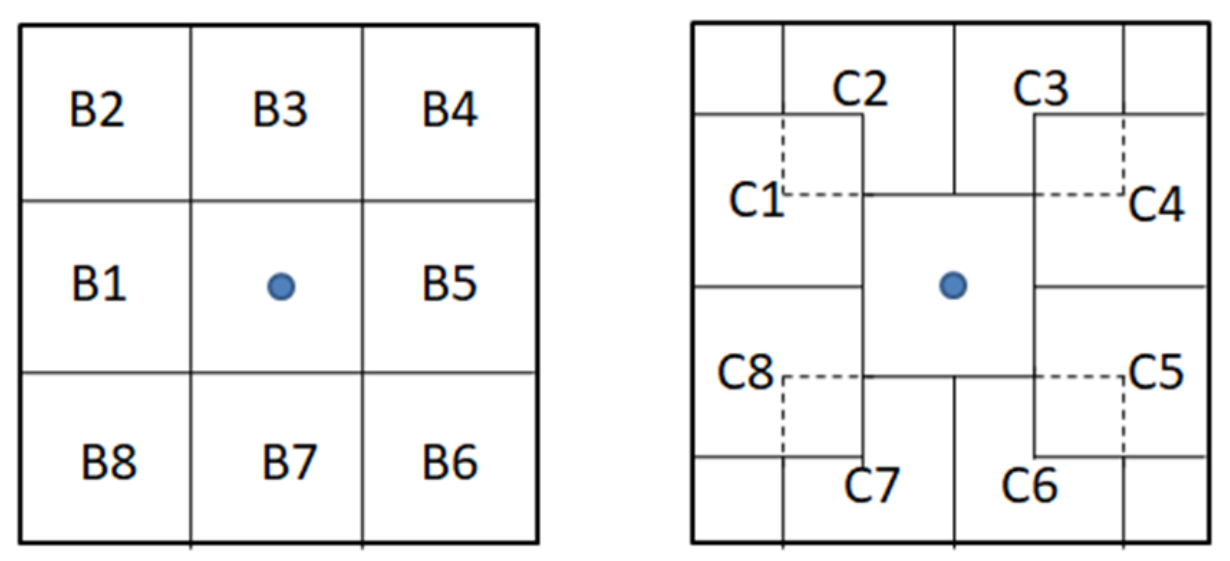


图3-7-5 图像块分割示意图

搜索路径的原理是检测与前进方向有关的 B 类矩阵或者是 C 类矩阵中的最大值，

然后确定接下来该往哪一个方向前进。例如飞机在向前飞行，在当前时刻检测到 B3 矩

阵的数值最大，那么飞机即继续往 B3 的中心飞行，即角度为 90°，同理，若为 C4 最

大，那么即往 22.5°的方向前进，以此类推。这里之所检测最大值，是因为当某一块

区域中的像素值越大，说明该区域内含有的路径部分越多，那么往这个方向前进的概

率即越大，那么在结合了上一步的高斯分布处理，可以更好地保证前进方向是往重心

方向且是包含最多路径信息的方向。

在Simulink中的程序如下：



图3-7-6 重心循迹算法图像处理部分Simulink程序框图

其中，“PARROT image Conversion”用于将原始的图像像素转化为RGB值。“RGB Image”可以将RGB值读取出来，用于观察当前视角。“line\_detect”为核心的路径检测函数（模块内核心代码见附录四），有两个作用，分别为计算得到分区权重和获得转换系数。“Hough Transform”为霍夫圆检测，用于判断是否到达终点。

3.3.2飞行控制算法

1. 确定飞行方向

想要飞到终点，无人机的飞行方向很重要，由于飞行途中相机所拍摄到的画面并

没有方向信息，只能从起始点就明确飞行方向，因此我们通过对机身周围的 6\*6 大小

的矩阵分割处理，以确定飞机在该区域内的方向。再加上由红色重心点确定的大方向

作为指引，最终确定飞机的飞行方向。其中 6\*6 矩阵分割成有重叠域的 17 个 2\*2 矩阵（图 3-1-1），然后通过当前飞行方向来确定下一时刻的方向。

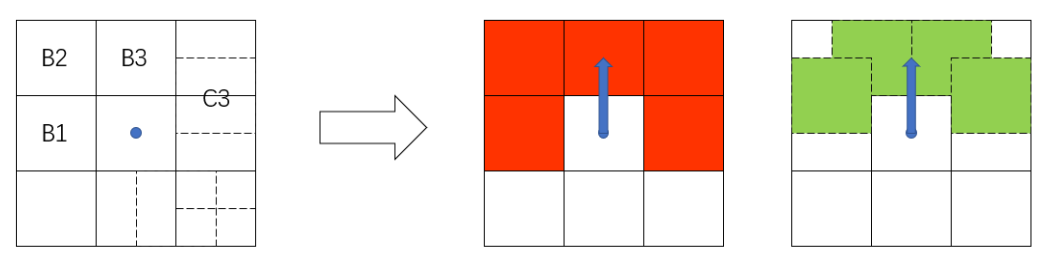


图3-7-7 矩阵分割及飞行方向示意图

首先将 6\*6 矩阵分为 B、C 两部分，B（图 3-1-1 实线）部分主要针对直道或非锐

角弯设计，它能较好地处理钝角、直角弯等较缓和赛段，C（图 3-1-1 虚线）部分则是

为了能让无人机顺利飞过锐角的急转弯而设计的。事实上，在无人机全程飞行中，两

部分都在起作用，能较好地配合处理路径路线。

如上图，当无人机飞行方向为箭头所指方向时，则扫描红色、绿色所标注的块，

通过比较这些区块的路径占比，找出占比最大的小块（目标块），为下一时刻的前进

方向。重心点则通过高斯矩阵加成，将原图像处理成包含重心加成的图像，被用作上

述扫描的基础矩阵。

每个小区块都对应一个飞行方向，其方向由区块中心点与机身中心（即全图中心）点的斜率所得到，因此只要求出目标块中心与机身中心连线的斜率后就能得到所

需飞行的 X、Y 方向速度比例 Speedx:Speedy。

X = 𝑋0 + ∆𝑋 ∆𝑋 = 𝑆𝑝𝑒𝑒𝑑𝑥 ∗ sin 𝑆𝑙𝑜𝑝𝑒

Y = 𝑌0 + ∆𝑌 ∆𝑌 = 𝑆𝑝𝑒𝑒𝑑𝑦 ∗ cos 𝑆𝑙𝑜𝑝𝑒

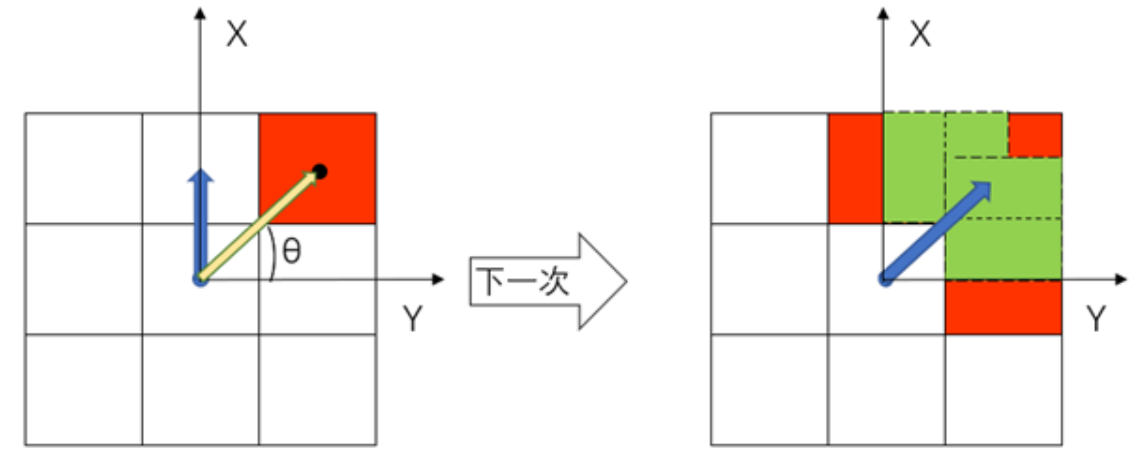


图3-7-8 Slope 检测与坐标示意图

图中的黄色箭头表示本图像处理周期内所确定的下一周期飞行方向，飞机则

按该θ角算得的 Speedx，Speedy 比例来控制飞行方向。为防止小扰动影响到下一次方

向判断，所以就以本次求得的黄箭头方向为下一图像处理周期的初始方向。以此类

推，后面的飞行方向都是这样确定的。

1. 路径分区定速

在飞行过程中，由于路径路线是变化的，因此用统一速度全程飞行的话，过快可

能在转角会脱离路径，过慢则浪费了直道的时间。我们通过在不同形状路径给定不同

的速度，使得无人机在直道和小弧度转弯处能飞得更快一些。图像处理部分以霍夫变

换方式能较准确区分出直道、弧弯、直角弯、斜直线以及路径终点，所以我们所给定

各赛段速度比例为直道：弧弯：直角弯：斜直线：路径终点=14：8：5：11：8。

此部分需要接入来自图像处理的路径信息，当布尔量 Hough\_ num = 1 则为直

道，然后以当前飞行方向为参考可进一步确定直道的方向。若是平行 X 方向的直道则

Speedx : Speedy = 28: 5 , 平行 Y 方向则反过来，为的是在保证直道快速飞行的情况下防止无关方向的大幅震荡。另一个布尔量 cir 则是为了区分弧弯和直角弯等，cir = 1 时表示路径为弧弯，以略小于直道的速度通过，cir = 0 时为直角弯和锐角弯则慢速拐过去。



图3-7-9 直道-高速飞行示意图

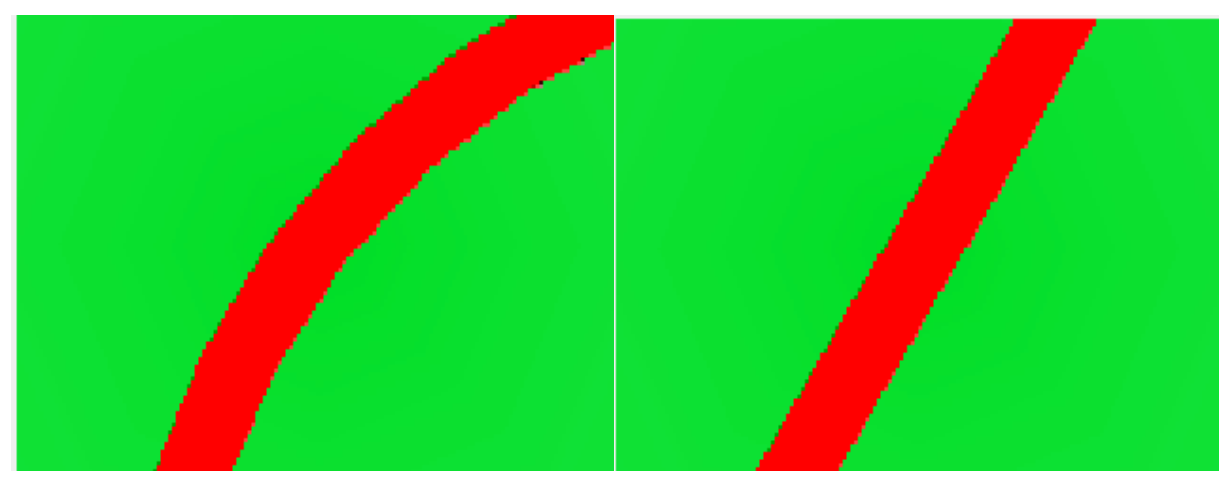


图3-7-10 弧线/斜直道-中速飞行示意图

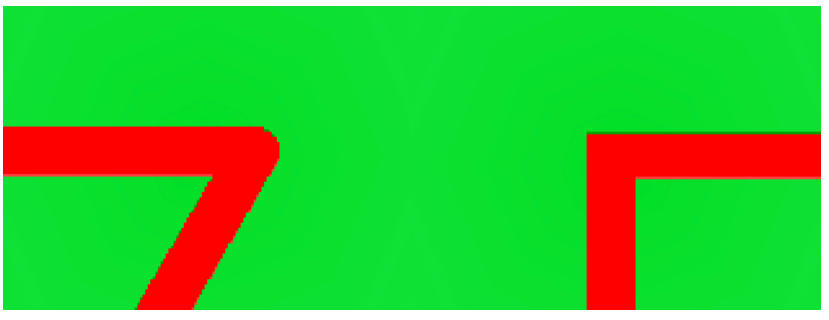


图3-7-11 弯道-低速飞行示意图

在这样分段调速处理后，无人机转弯稳定性提高很多，兼顾了转弯效率和直道速

度，最终跑完全程后时间缩短且震荡点也明显减少。

在Simulink中的程序如下：



图3-7-12 重心循迹算法飞行控制部分Simulink程序框图

其中，“center gravity control”模块为核心算法，使用“重心循迹算法”来对航迹点进行跟踪（模块内核心代码见附录五），“state control”模块可以判断当前是否检测到终点，用于状态的转换。“分区测速”区域直接采用Simulink Block 设计，方便参数条数，作用为对不同角度进行调速。整体的输出为需要控制的角度误差，下一步由PID来消除误差。

3.3.2仿真数据分析

整体飞行轨迹（黑色为路径，橙色色为实际飞行轨迹）：

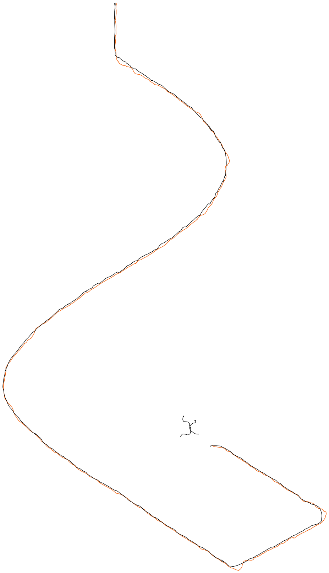


图3-7-13 重心循迹算法飞行轨迹对比图

整体飞行时间：

飞行五次时间的平均值为：33.42s

整体误差：

通过角度的整体方差计算，飞行五次的平均误差为：98.72

三个不同角度的转弯数据图：

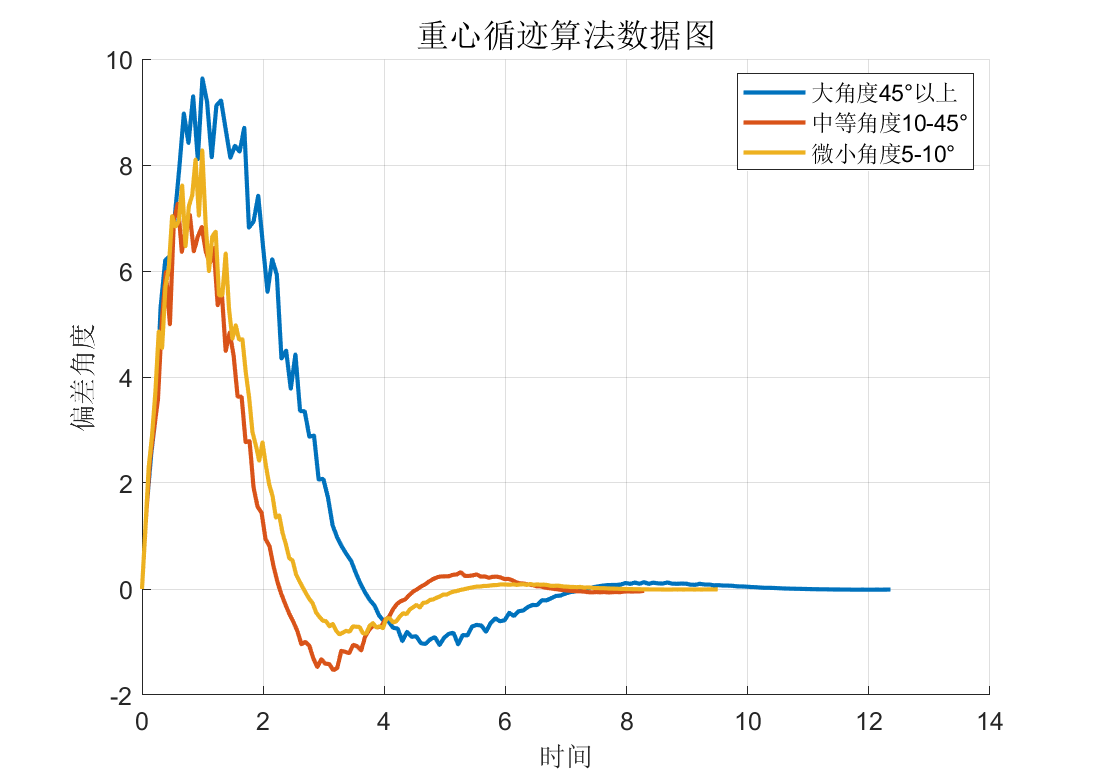


图3-7-13 重心循迹算法三个不同角度的转弯数据图

三个不同角度的转弯的参数对比：

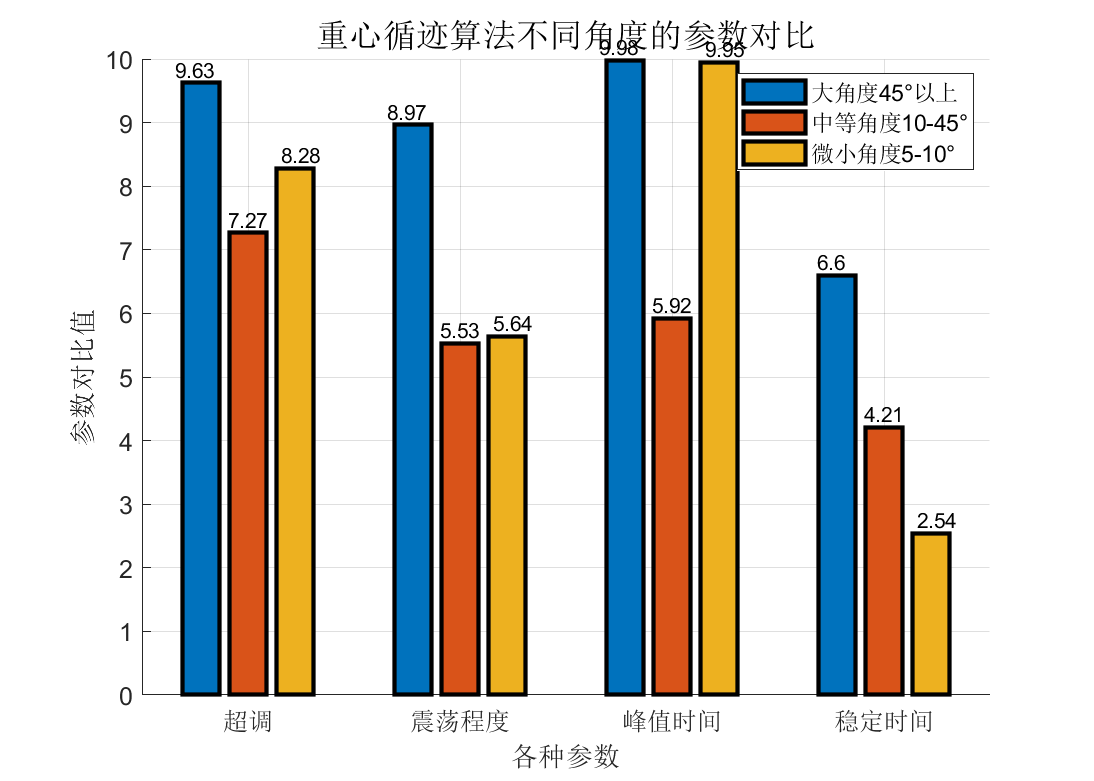


图3-7-14 重心循迹算法三个不同角度的参数对比图

可以看出，三圆轨迹生成算法在微小角度上反应非常迅速，而且超调量低，调节时间长。但是对于较大的角度，反应较慢，而且超调很高。

4 三种无人机视觉控制算法仿真结果对比

4.1 三种算法飞行速度与整体误差比较

三种方式对于飞行速度和整体的误差的影响不相同，为了更加直观地分析三种算法的仿真数据，以下将三种方法的飞行时间和整体误差进行横向比较。

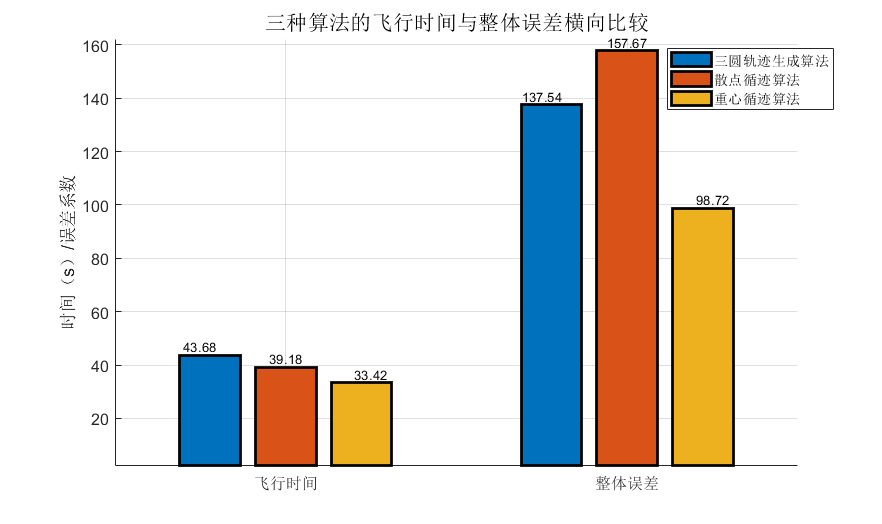


图4-1-1 三种方法的飞行时间和整体误差对比图

由上面的，数据分析可以知道，“三圆轨迹生成算法”速度最慢，“散点探测循迹算法”其次，“重心循迹算法”最快。同时，“散点探测循迹算法”有较快的速度的同时，整体误差也较大。

4.2 三种算法对于角度反应的比较

三种方式对于微小角度、中等角度、大角度的反应各不相同，为了更加直观地分析三种算法的仿真数据，以下将三种角度的四种参数进行横向比较。

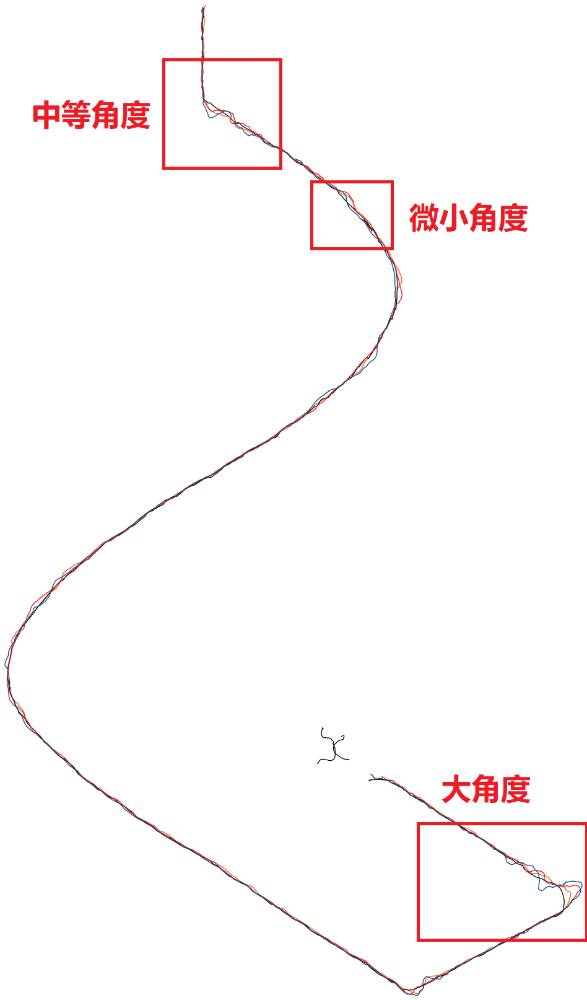


图4-2-1 三种方法的飞行轨迹对比图

将飞行的路径以及三种方法整体的轨迹放在同一张图上进行比较，并且选取出其中具有代表性的微小角度，中等角度，以及大角度各测量5次的平均值进行比较。角度的选取，如上图所示。

三种方法不同角度的转弯轨迹：

微小角度的转弯轨迹图：

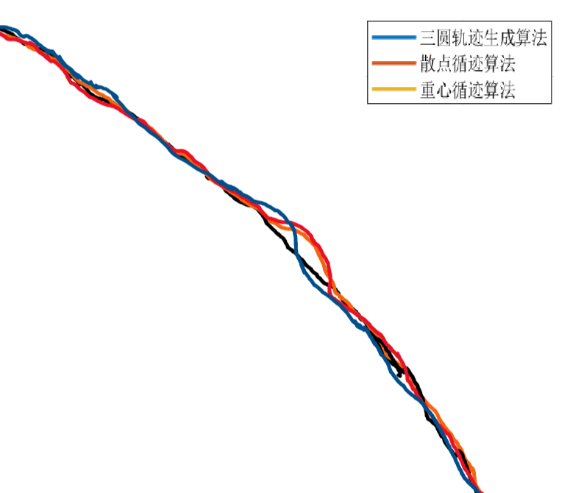


图4-2-2 三种方法的微小角度飞行轨迹对比图

中等角度的转弯角度图：

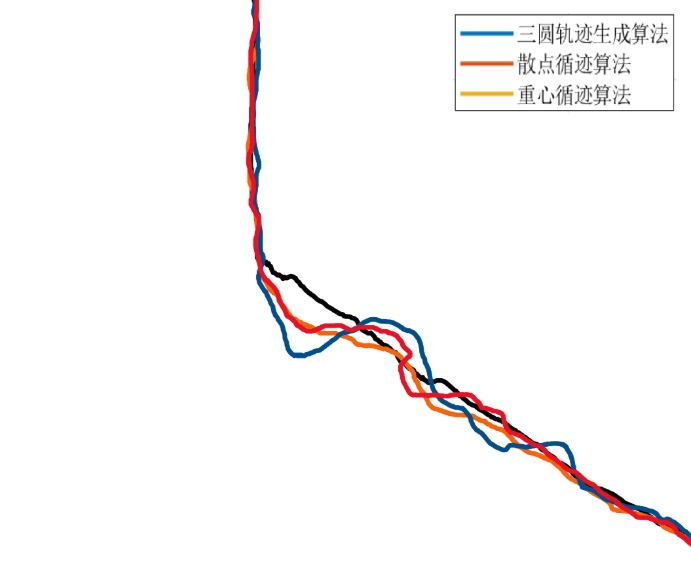


图4-2-3 三种方法的中等角度飞行轨迹对比图

大角度的转弯角度图：

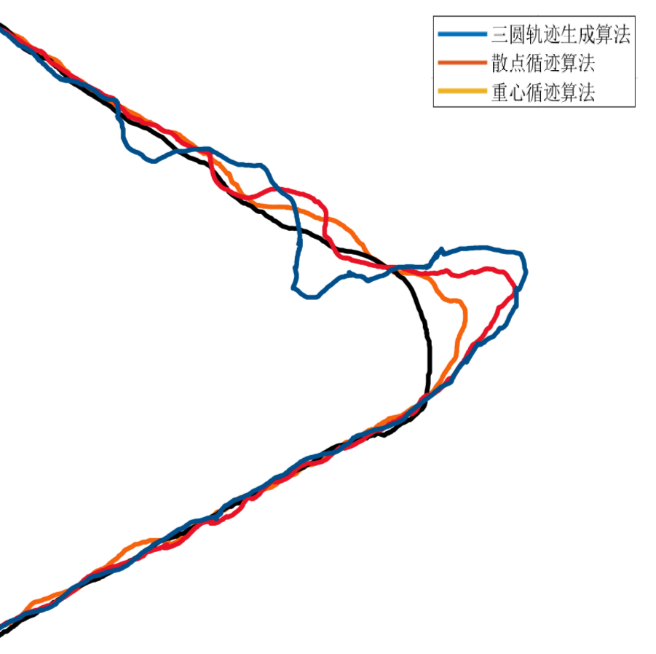


图4-2-4 三种方法的大角度飞行轨迹对比图

以上三幅图为微小角度，中等角度与大角度的直观图，下面对三种角度的数据进行对比，核心的对比数据为角度的偏差值。

大角度三种方法数据对比：

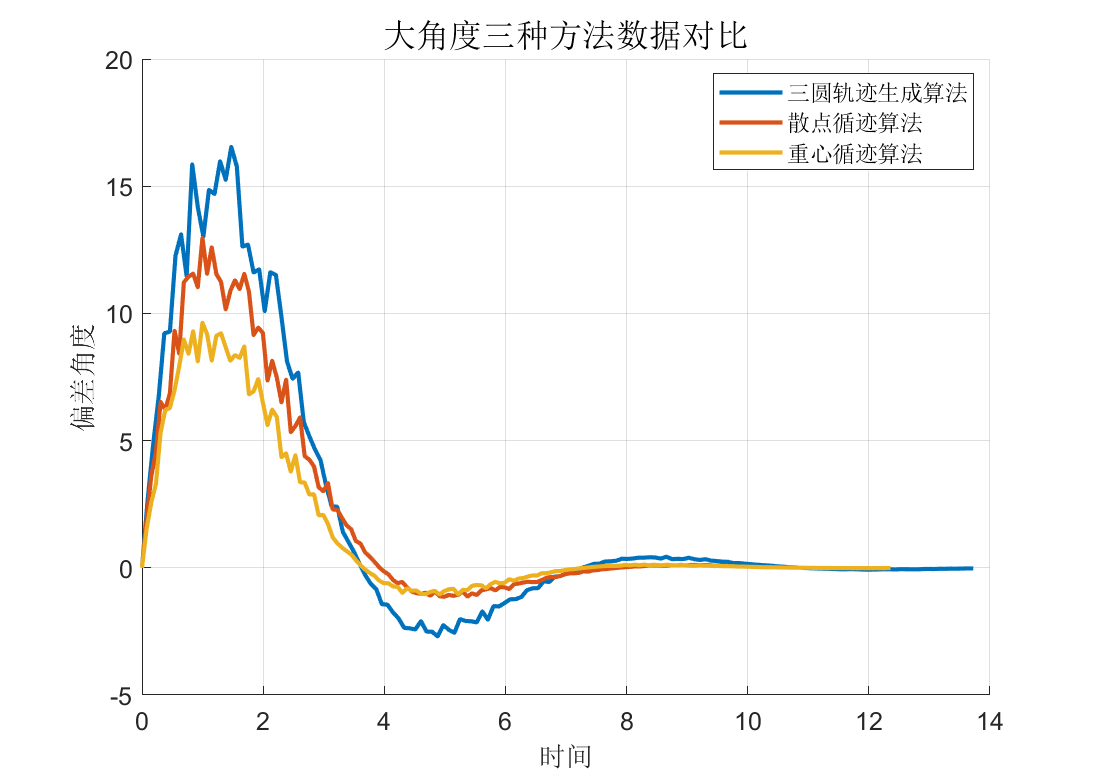


图4-2-5 三种方法的大角度数据对比图

大角度三种方法参数对比：

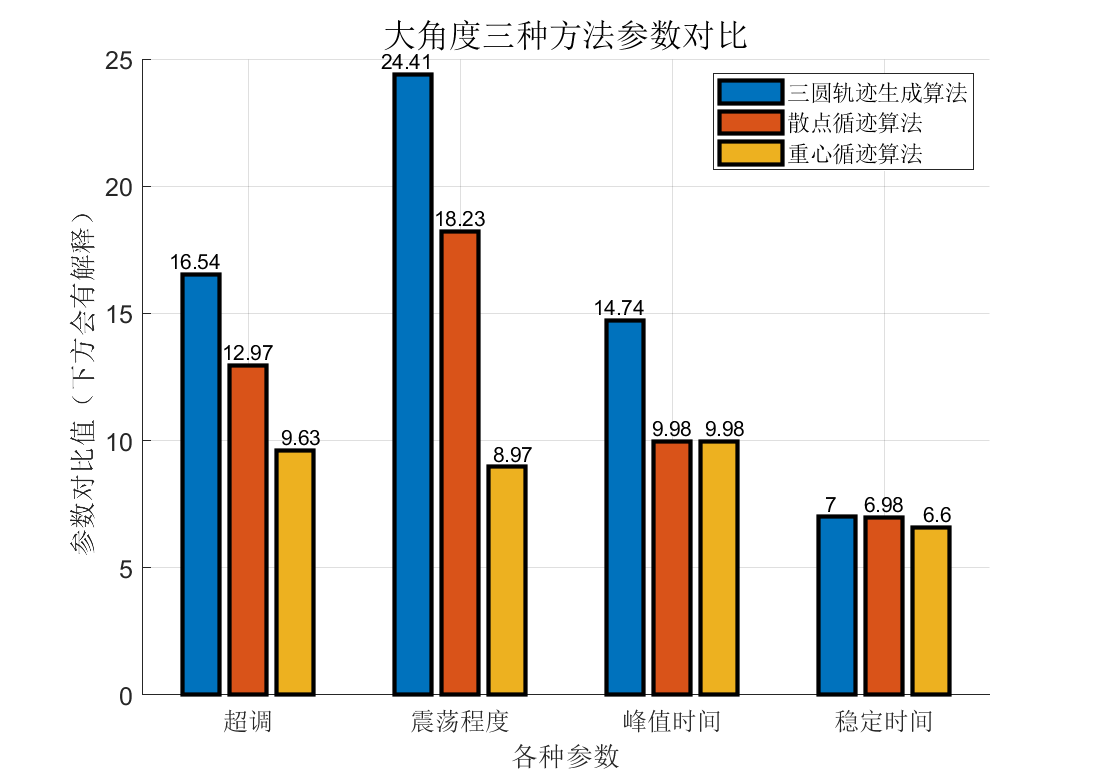


图4-2-6 三种方法的大角度参数对比图

可以明显看出，在大角度上“三圆轨迹生成算法”表现非常差，超调高达17%，震荡程度与峰值时间也远大于其他两种方法。“散点探测循迹算法”表现中等，超调约13%，震荡程度在18左右，但是反应速度较快。“重心循迹算法”表现最好，超调不到10%，震荡程度约9，稳定时间也非常快。因此在大角度上，“重心循迹算法”表现更加优异。

中等角度三种方法数据对比：

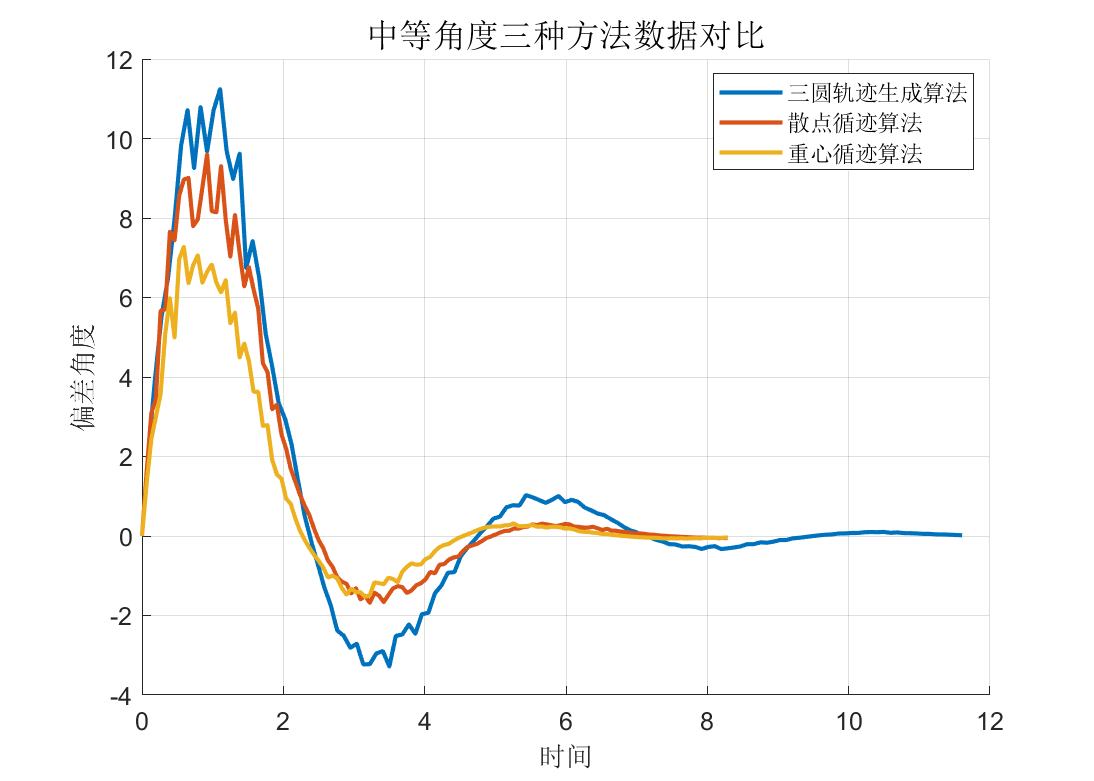


图4-2-7 三种方法的中等角度数据对比图

中等角度三种方法参数对比：

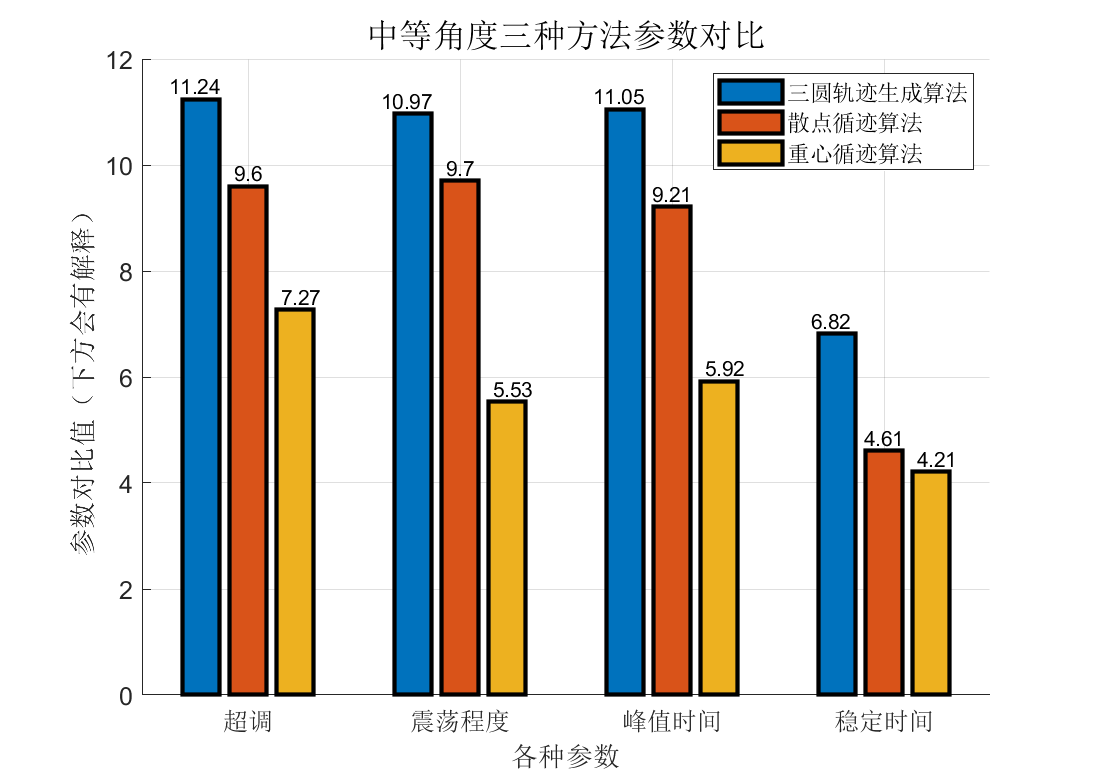


图4-2-8 三种方法的中等角度数据对比图

可以看出，在中等角度上“三圆轨迹生成算法”同样表现非常差，超调达到11%，震荡程度与峰值时间也远大于其他两种方法，都约为11。“散点探测循迹算法”表现也较差，超调约10%，震荡程度在10左右，但是反应速度较快。“重心循迹算法”表现仍然最好，超调7%，震荡程度约5，稳定时间与“散点探测循迹算法”几乎一样。因此在中等角度上，“重心循迹算法”表现更加优异。

微小角度三种方法数据对比：

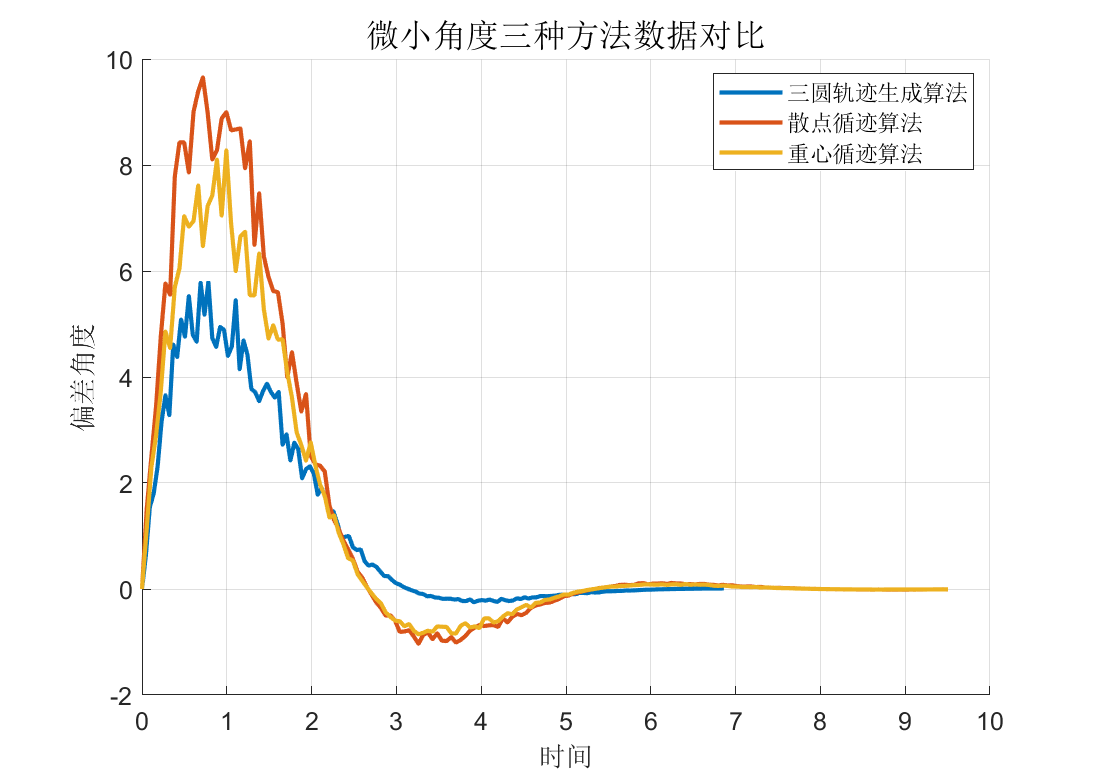


图4-2-9 三种方法的微小角度数据对比图

微小角度三种方法参数对比：

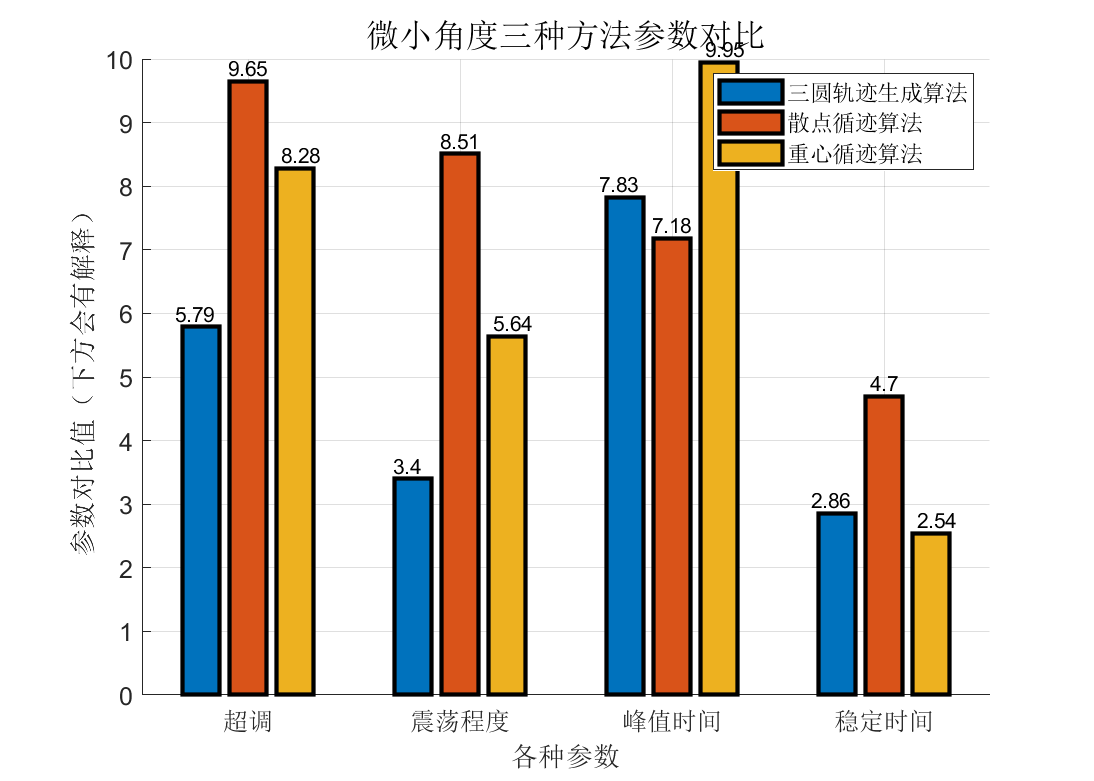


图4-2-10 三种方法的微小角度数据对比图

可以看出，在微小角度上差异非常明显，“散点探测循迹算法”同样表现非常差，超调达到10%，震荡程度达9，稳定时间也较慢。“重心循迹算法”同样较差，超调约9%，震荡程度在6左右，但是稳定时间较快。与大角度和中等角度不同的是“三圆轨迹生成算法”在小角度上表现最好，超调6%，震荡程度约3，峰值时间与稳定时间也非常小。因此在微小角度上，“三圆轨迹生成算法”表现更加优异。

4.1 结论

由上面的数据分析可知，“三圆轨迹生成算法”可以对于微小的变化更加敏感的跟踪，但同时由于其驱动圆离视野较近，同时一级和二级的调控圆没能及时的更新信息，包括因为权重的不足或者因为路径变化过于崎岖，对于大角度的转动来说，尤其显得非常的不足。具体体现在相比于其他两种算法，对于中等角度和大角度，其调节的时间更长，超调和误差。对此，这一种方法最佳的应用场景为相对长途的移动，对于角度不用很大偏差的场景，比方说在丛林上空。

“散点探测循迹算法”对于大范围的不确定路径表现更加优异，原因是散点的探测具有随机性，因此对于很大规模的变化，能够非常敏感的捕捉到，因为权值相等，如果偏差更大的话，使得驱动的点更加准确。但同时这一种方法相比较于其他两种方法，虽然速度具有一定的优势，但是对于微小的变化非常的不灵敏，这也是由于刚刚所说的，对于大范围的捕捉的灵敏导致了对小范围捕捉的丧失。因此这一种方法最佳的适应场景为需要大范围、大角度进行移动的场景，比方说室内。

最后这种算法“重心循迹算法”，是基于以上两种方法提出来的更加优化的算法，相比于算法“散点探测循迹算法”，这种算法使用了分区的概念，可以弥补散点的随机性导致的大范围误差，具体为对于核心区（靠近路径的地方），予以更高的权重。这样既能够保证在微小变化的时候，不至于被远距离的点给影响太大，也能保证在大范围移动的时候，不至于受到近距离的微小变化而影响，因此相比较于其他两种方法而言，这种方法虽然在微小角度上并不如“三圆轨迹生成算法”好，但是对于中等角度以及大角度以及整体的反应时间，整体的误差而言，这种方法是最优的。因此这种方法适应于大部分的场景。

5 总结与展望

5.1 总结

本文首先阐述了国内外对于四旋翼飞行器、视觉控制与基于模型设计的现状，然后引入了“基于模型设计”的理念，实现快速的二次开发以及传感器数据反馈和纠正，开发了三种不同的飞控算法，并对比其结果进行对比，找到适合于不同情境下的无人机系统，可以快速实现个性化部署。

本文主要完成了一下五个方面的内容：

1. 本文以Parrot的微型四旋翼无人机为基础，分析了四旋翼飞行器的基本结构和飞行原理，以及在特定条件下得到此微型四旋翼无人机的数学模型，进行了四旋翼无人机的控制系统理论研究。同时研究了基于模型设计的背景以及意义、视觉控制的背景以及意义。
2. 本文提出了三种不同的飞控算法进行开发，通过根据基于模型设计的方法进行了快速的验证，控制任务实现了让无人机沿着特定颜色的路径飞行。并研究了这三种方法对于飞行的飞行时间、整体误差以及飞行路途中三种不同角度所产生的误差。最终得到不同方法的适应场景，并证明了基于模型设计应用于飞行控制系统设计的可行性。
3. 设计了“三圆轨迹生成算法”的飞控系统，可以对微小的变化敏感地跟踪。
4. 设计了“散点探测循迹算法”的飞控系统，对大范围不确定路径更好的跟踪。
5. 设计了“重心循迹算法”的飞控系统，相比于“散点探测循迹算法”，更好地实现调速。

通过实验验证，基于模型的微型自主四旋翼飞行机器人飞控系统设计与开发能快速部署，基本实现快速开发，尽管如此，本课题还存在一些问题：

1. 所采用的无人机比较单一，为Parrot Mambo 四旋翼无人机，其余无人机大多没有Matlab的物理模型和仿真系统。因此应用此方法开发时，前期的物理模型建立和仿真系统建立会比较费时。
2. 由于现实情况不允许，没有进行实物测试，经过前期的调研和小规模实验，实物中的数据，如PID等，需要实际修改。

5.2 展望

在传统无人机领域，我们会遇到非常多的开发问题以及现实的问题，比方说城市的低空巡逻亦或者是对于农业用地的，这些等等问题都是人机的应用过程当中非常常见的。

为了使得这些二次开发的过程变得更加的迅捷。我们研究了基于模型设计的方法，这个方法可以为开发人员新功能的研发提供了一种非常高效的开发模式，这种模式开发人，将代码清除，避免了人们引入的一些错误，非常好的提高了开发的效率。

我们可以发现。未来这样的个性化场景只会越来越多。正如我在前文所说的，基于模型设计之中，我们还遇到了很多问题。其中最重要的一个问题并不是模型建立之后我们如何进行优化，恰恰是如何准确地建立模型本身。因为我们要知道一个飞行系统，要准确的进行控制，就必须要有准确的仿真，而准确的仿真依赖于准确的模型。物理模型的建立一旦出现问题，则后面的仿真即使非常准确，也会出现非常大的偏差。

同时在现实生活当中几乎很难找到完全相同的物理模型，每一种不同型号的无人机都有完全不一样的结构参数。这些结构参数又要对应着不同的使用条件。要在指定的使用条件里面建立合适的物理模型需要非常大量的工作。而且即便建设出来之后，还要不同场景的进行校正，你使得这个物理模型尽量贴近于现实物体。这无疑是耗费巨大的人力物力。

如果未来我们能够，进行结构化的设计，同时把设计出来的成果进行模型设计的仿真，然后依照仿真的结果对实物进行加工并检验。这样整一套流程的闭环，如果能够发展完善，就可以减少我上述所说的物理模型建立困难的事情，因为这个困难的事情在第一步就已经很好的解决。

如果对于个性化设计的产品则需要对其进行更加简化的物理建模。比方说，经过多次的模型建立和测试，找到关键参数进行测试，而不至于每次都推倒重做。

其次，对于运算平台而言，基于模型设计的理念，在设计的时候非常直观，但是同时会出现代码植入的时候，大量的冗余以及运行速度低下。

未来需要在减少代码冗余以及设计与实物制作一体化方面发展

致谢

在本文的最后，我要感谢在本科期间对我学习进行过指导和帮助的各位老师，本文用到的各种控制算法，或多或少都是他们曾经教授过的。同时非常感谢我的师兄师姐和同学，在校园里面的生活使我获益良多，在此对他们表示最真诚的感谢。

在这里，我也要衷心感谢我的毕设指导老师张博老师，通过张博老师的机器人集群规划的创新短课，我认识到了“基于模型设计”（MBD）这个全新的领域。同时，在毕设期间，张博老师都给与了我很大的帮助。从一开始的毕设选题，调试过程中出现的问题，到最后整篇论文的完成和审核，张博老师都十分耐心的给与我意见。大四毕业迷茫之际，非常感谢张博老师给与我的帮助，让我严格要求自己，跨过一道又一道难关。

其次，我也要感谢我的小组成员何登桂。在毕设期间，我碰到了很多困难，一些问题久久不能解决使得我的心态出现了问题。但是何登桂在不断地鼓励我不要放弃，在他的帮助下，我解决了很多问题，一步一个脚印，把毕设一个模块一个模块地搭起来，最后完成了整个毕业设计，所以在这要衷心感谢我的小组成员何登桂。

最后，向评阅本文的各位老师致以感谢。

参考文献

1. 赵旭.基于OPENCV视觉处理的旋翼无人机识别规划的设计[J].电子技术与软件工程,2019(03):63.
2. 刘焕晔.小型四旋翼飞行器飞行控制系统研究与设计[D].上海:上海交通大学,2009.
3. 岳基隆,张庆杰,朱华勇.微小型四旋翼无人机研究进展及关键技术浅析[J].电光与控制,2010,17(10):46-52
4. 郑栋梁. 基于图像的四旋翼无人机视觉伺服控制[D].上海交通大学,2018.
5. 郭丹, 刘杰, 郭丹, et al. 基于模型设计与传统方式的联合设计开发[J]. 单片机与嵌入式系统应用, 2012, 12(9):16-19.
6. 王黎明. 控制系统基于模型的系统工程开发方法研究[D]. 西安电子科技大学, 2013.
7. 原劲鹏, 杨涤, 杨旭. 基于Matlab的多功能一体化开发平台应用研究[J]. 计算机仿真, 2006, 23(2).
8. 陈怀民, 史旭涛, 马松辉, et al. 基于模型的无人机飞控软件设计与实现[C]// 2014（第五届）中国无人机大会. 0.
9. 徐颖. 基于matlab的嵌入式系统程序开发的研究[D]. 厦门大学, 2011.
10. 尧伟文, 曹云峰, 庄丽葵. 一种基于模型开发飞控系统的方法[J]. 计算机与数字工程, 2017(2).
11. 罗喜霜, 宋亮, 雷玮, et al. 基于模型的预警无人机飞控系统开发与验证[J]. 计算机测量与控制, 2017, 25(12).
12. 吕强,马建业,王国胜,林辉灿,梁冰.基于视觉伺服的小型四旋翼无人机自主飞行控制研究进展[J].科技导报,2016,34(24):68-73.
13. 钱超超,吕勇.一种基于Linux的视觉识别四旋翼控制系统[J].电子技术,2017,46(01):47-50.
14. 步青. 无人机视觉智能控制相关技术研究[D].浙江工业大学,2017.
15. 谈青艳. 基于视觉引导的多旋翼无人机地面动目标跟踪技术研究[D].南京航空航天大学,2017.
16. 姚博誉,路平,杨森, 等.四旋翼飞行器飞行控制技术综述[J].航空兵器,2020,27(1):9-16
17. Bernard Martino,Boreggio Mauro,Degetto Massimo,Gregoretti Carlo. Model-based approach for design and performance evaluation of works controlling stony debris flows with an application to a case study at Rovina di Cancia (Venetian Dolomites, Northeast Italy).[J]. The Science of the total environment,2019,688.
18. XiaoBin Xu,Zhao Wang,YiMin Deng. A software platform for vision-based UAV autonomous landing guidance based on markers estimation[J]. Science China Technological Sciences,2019,62(10).
19. C. Patruno,M. Nitti,A. Petitti,E. Stella,T. D’Orazio. A Vision-Based Approach for Unmanned Aerial Vehicle Landing[J]. Journal of Intelligent &amp; Robotic Systems,2019,95(2).
20. Wilbert G. Aguilar,Leandro Álvarez,Santiago Grijalva,Israel Rojas. Monocular Vision-Based Dynamic Moving Obstacles Detection and Avoidance[M].Springer International Publishing:2019-08-06.
21. . Model Based Design of Embedded Systems[J]. Journal of Automation and Control,2017,5(2).
22. Qiang Gao,Mei Du,Yuehui Ji. The Controller Design of Quadrotor UAV Based on Internal Model Control[A]. 中国自动化学会控制理论专业委员会.第36届中国控制会议论文集（A）[C].中国自动化学会控制理论专业委员会:中国自动化学会控制理论专业委员会,2017:6.
23. Wooh Yun Kim,Ji Wook Kwon,Ji Won Seo. Formation Control of Quadrotor UAVs by Vision-Based Positioning[J]. Applied Mechanics and Materials,2015,4092.
24. Steven A.P. Quintero,João P. Hespanha. Vision-based target tracking with a small UAV: Optimization-based control strategies[J]. Control Engineering Practice,2014,32.
25. Model Based Design and HIL Simulations[J]. American Journal of Mechanical Engineering,2013,1(7).
26. Ming Zuo,Ying Liu,Yi Qian,Xiong Wen Hu,Xiao Chuan Zhao,Jin Hua Wang. Model-Based Design of UAV Autopilot Software[J]. Advanced Materials Research,2013,2534.
27. M. Eggersmann,R. Schneider,W. Marquardt. Understanding the interrelations between synthesis and analysis during model based design[J]. Computer Aided Chemical Engineering,2003,15.
28. Motus L.,Kääramees K.. A Model Based Design of Distributed Control System Software[J]. Elsevier,1982,15(3).

附录一 三圆轨迹生成算法视觉处理

1. %% 找红色
2. r = image\_data(1);
3. g = image\_data(2);
4. b = image\_data(3);
5. flag = 1;
6. Image = r;
7. index = g(:)>50 | b(:)>50;
8. Image(index) = 0;
9. Image = Image.\*temp\_Image;
10. [width,height] = size(Image);
12. %% 找路径
13. skeletonizedImage = bwmorph(Image, 'thin', inf);
14. edgeImage = bwperim(Image);
15. Image2 = bwperim(skeletonizedImage);

附录二 三圆轨迹生成算法飞控函数

1. %% 三圆轨迹生成算法
2. cut = [45;50;55];
4. cut\_XY = zeros(length(cut(:)),4);
5. **for** i = 1:length(cut\_XY(:,1))
6. cut\_XY(i,1) = center\_x - cut(i);
7. cut\_XY(i,2) = center\_y - cut(i);
8. cut\_XY(i,3) = center\_x + cut(i);
9. cut\_XY(i,4) = center\_y + cut(i);
10. end
12. sum\_vector = zeros(length(cut(:)),3);
14. **for** i = 1:length(cut\_XY(:,1))
15. I1 = Image\_roi(cut\_XY(i,1):cut\_XY(i,3),cut\_XY(i,2):cut\_XY(i,4));
16. index = find(I1(:)==1);
17. position = zeros(length(index),3);
18. I1\_center\_x = 1+cut(i);
19. I1\_center\_y = 1+cut(i);
20. **if** ~isempty(index)
21. **for** k=1:length(index)
22. [position(k,1),position(k,2)] = ind2sub(size(I1),index(k));
23. position(k,1) = I1\_center\_x - position(k,1);
24. position(k,2) = position(k,2) - I1\_center\_y;
25. position(k,3) = sqrt((position(k,1))^2+(position(k,2))^2);
26. end
27. index1 = position(:,3) <= cut(i)+0.5 & position(:,3) >= cut(i)-0.5 & position(:,1) > 0;
28. position = position(index1,:);
29. end
30. **if** ~isempty(index)
31. **if** length(position(:,1))==1
32. sum\_vector(i,1) = position(1,1);
33. sum\_vector(i,2) = position(1,2);
34. temp = sqrt(sum\_vector(i,1)^2 + sum\_vector(i,2)^2);
35. sum\_vector(i,1) = sum\_vector(i,1)/temp;
36. sum\_vector(i,2) = sum\_vector(i,2)/temp;
37. sum\_vector(i,3) = atan2(sum\_vector(i,2),sum\_vector(i,1));
38. end
39. end
40. end
41. % vector = 0.25\*sum\_vector(1,:)+0.5\*sum\_vector(2,:)+0.25\*sum\_vector(3,:);
42. vector = 0.3\*sum\_vector(1,:)+0.3\*sum\_vector(2,:)+0.4\*sum\_vector(3,:);
43. **if** abs(sum\_vector(1,1)) < 1e-5 && abs(sum\_vector(1,2)) < 1e-5
44. flag = -1;
45. end
46. line = vector(3);
47. line\_flag = flag;

附录三 散点探测循迹算法飞控函数

1. %% 散点探测循迹算法
2. %生成随机点
3. theta=0:0.01:360;
4. % 利用极坐标得到圆的坐标
5. Circle1=x0+R\*cos(theta);
6. Circle2=y0+R\*sin(theta);
7. % 随机生成num\_Dian个半径
8. r=R\*sqrt(rand(1,num\_Dian));
9. seta=2\*pi\*rand(1,num\_Dian);
10. % 得到点的坐标
11. x=x0+r.\*cos(seta);
12. y=y0+r.\*sin(seta);
14. %计算代价
15. **for** i = 1:length(x)
17. index = length(x);
18. position = zeros(index,3);
19. **if** ~isempty(index)
20. **for** k=1:length(index)
21. %代价1
22. position(k,1) = sqrt((x(i)-x\_index(k))^2+(y(i)-y\_index(k))^2;
23. %代价2
24. position(k,2) = x(i)\*cos(seta)-index(k)\*cos(seta);
25. %代价3
26. position(k,3) = sqrt((position(k,1))^2+(position(k,2))^2);
27. end
28. index1 = position(:,3) <= cut(i)+0.5 & position(:,3) >= cut(i)-0.5 & position(:,1) > 0;
29. position = position(index1,:);
30. end
31. end

附录四 重心循迹算法视觉处理

1. %% 分区及像素计算
2. R\_numMatrix = zeros(12,16);
3. R\_stright=image\_data(:,20:140);
4. **for** i=1:1:12
5. **for** j=1:1:16
6. m = 10\*(i-1)+1;
7. n = 10\*(j-1)+1;
8. **for** m = m:1:m+8
9. **for** n = n:1:n+8
10. **if**(R\_Matrix(m,n) > 230)
11. R\_numMatrix(i,j) = R\_numMatrix(i,j)+1;
12. end
13. end
14. n = 10\*(j-1)+1;
15. end
16. end
17. end

附录五 重心循迹算法飞控函数

1. %% 重心循迹算法
2. %计算重心高斯矩阵
3. outOrigin=R\_numMatrix;
4. Deno=sum(sum(R\_numMatrix));
5. xFocus = (1:1:12) \* R\_numMatrix;
6. xFocus = sum(xFocus(:));
7. yFocus = R\_numMatrix \* (1:1:16)';
8. yFocus = sum(yFocus(:));
9. Focus = [xFocus yFocus]/Deno;
11. %四舍五入
12. **if**(mod(Focus(1),1)>0.5)
13. Focus(1)=Focus(1)-mod(Focus(1),1)+1;
14. **else**
15. Focus(1)=Focus(1)-mod(Focus(1),1);
16. end
18. **if**(mod(Focus(2),1)>0.5)
19. Focus(2)=Focus(2)-mod(Focus(2),1)+1;
20. **else**
21. Focus(2)=Focus(2)-mod(Focus(2),1);
22. end
24. a=ones(25,33);
25. weight\_center = 4;
26. weight\_minus = 0.1;
27. a(:,1:1:4)=ones(25,1)\*((weight\_center-15\*weight\_minus):weight\_minus:(weight\_center-12\*weight\_minus));
28. a(:,30:1:33)=ones(25,1)\*((weight\_center-12\*weight\_minus):-weight\_minus:(weight\_center-15\*weight\_minus));
29. **for** i=13:-1:1
30. **if**(i==13) a(13,17)=weight\_center;**continue**; end
31. a((13-i):1:(13+i),(17-i))=weight\_center-i\*weight\_minus;
32. a((13-i):1:(13+i),(17+i))=weight\_center-i\*weight\_minus;
33. a((13-i),(17-i):1:(17+i))=weight\_center-i\*weight\_minus;
34. a((13+i),(17-i):1:(17+i))=weight\_center-i\*weight\_minus;
35. end
36. a1=6;a2=7;
37. **for** i=1:1:16
38. **if**(Focus(1)==i) a1=i;end
39. **if**(Focus(2)==i) a2=i;end
40. end
42. weight=a((13-a1):1:(13+12-a1-1),(17-a2):1:(17+16-a2-1));
43. r=R\_numMatrix .\* weight;
45. origin=outOrigin;
46. y = r;
47. ab=[a1 a2];

附录六 转换系数求取函数

1. %% 求出转换系数
2. % 等分处理,取众数，求出路宽,
3. % 需要从图像的x,y方向上进行截取的原因是，为了保证无人机能转弯。
4. % 因为后面求出了转换系数，只考虑x方向的话，到转弯处，转换系数会发现剧烈变化，从而影响输出。
5. temp = zeros(26,1);
6. k = 1;
7. **for** i = 10:10:width-10 %水平方向取11个值
8. index = find(edgeImage(i,:) == 1);
9. **if** ~isempty(index)
10. temp(k) = max(index) - min(index);
11. **else**
12. temp(k) = k;
13. end
14. k = k+1;
15. end
16. **for** i = 10:10:height-10 %垂直方向取15个值
17. index = find(edgeImage(:,i) == 1);
18. **if** ~isempty(index)
19. temp(k) = max(index) - min(index);
20. **else**
21. temp(k) = k;
22. end
23. k = k+1;
24. end
25. road\_image\_length = mode(temp);
27. %转换系数，即真实距离与图像像素距离之比,一个像素对应的实际距离
28. Conv\_coeff = 0.15/road\_image\_length;
29. **if** Conv\_coeff > 0.01
30. Conv\_coeff = 0.01;
31. end
32. Conv\_coeff = 9.4e-3