编号_____

南京航空航天大学

毕业设计

题　目　基于大疆无人机平台的自主飞行和航迹规划研究与实现

学生姓名　金铭

学　　号　161410104

学　　院　计算机科学与技术

专　　业　计算机科学与技术

班　　级　161410104

指导教师　翟象平

二〇一八年六月

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：基于大疆无人机平台的自主飞行和航迹规划研究与实现）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）：

基于大疆无人机平台的自主飞行和航迹规划研究与实现

摘 要

随着科技快速成长，尖端技术在许多的行业都发挥着巨大的作用。在无人机领域中，消费级应用和行业应用的比例也逐渐增大，让人们有了一种全新的便利的生活模式。

为了操作方便简单，并且在农业喷洒、电力测绘方面能有更大地用途，本项目提出了基于模拟退火算法来实现无人机自主飞行与航迹规划的想法，并将其设计成移动应用。在本项目的研究中具体包括以下几点：旅行商问题研究、无人机控制和通讯研究、移动应用开发、DJI Mobile SDK 开发、高德地图 API 开发。

在项目中需要计算出在整个飞行的过程中，从起始点开始飞行在每一个航点都只经过一次的情况下，然后再回到起始飞行点的最优的路程。为了解决这个问题，项目将会把航点问题和旅行商类比并且使用模拟退火算法来解决。

在项目中主要需要明白电机运动和飞行的关系，以及开发出的移动应用怎么与无人机通讯。最后本项目采用的是控制设备利用现有遥控器已有的无线信号来控制无人机的方式。移动设备利用 USB 线连接控制器，控制器再与无人机使用无线传输连接。移动设备和控制器需要运用到的通讯协议是串行接口协议。同时遥控器与无人机利用大疆推出的 light bridge 图传技术进行通讯。

DJI Mobile SDK 和高德地图 API 主要是分别为开发提供了针对于无人机控制通讯和地图操作的接口，使得开发过程更加快速简单。

最后，本项目主要会实现利用移动应用控制无人机航点设置、航点优化以及自主飞行功能。

关键词：旅行商问题，模拟退火算法，安卓应用，DJI Mobile SDK，高德地图 API，无人机，自主飞行，航迹规划

Research and Implementation of Route Planning Based on DJI UAV Platform

Abstract

With the development of technology, cutting-edge technology has been gradually applied to consumer applications and industrial applications, to facilitate people's lives. In the field of UAV, the proportion of consumption level application and industry application is also increasing, providing a new way of life for people.

In order to operate easily, and have more use in agricultural spraying and electric power mapping, this project proposes the idea of realizing unmanned aerial vehicle autonomous flight and navigation point planning based on simulated annealing algorithm, and designs it into mobile application. This enables operators to operate faster and more conveniently. The research in this project includes the following points: the traveling salesman problem research, the UAV control and communication research, the mobile application development, the DJI Mobile SDK development, and the Gaode Map API development.

Traveling salesman problem mainly solves the problem of waypoint planning in projects. In the project, we need to calculate the shortest distance from the starting point to all the waypoints and finally back to the origin. Finally, this project is going to use simulated annealing algorithm to solve this problem.

The research of UAV control and communication is the foundation of the project. In the project, we mainly need to apply the relationship between motor movement and flight, and how to develop mobile applications to communicate with UAVs. At last, this project adopts the way of controlling the equipment to control the UAV by using the existing wireless signal of the remote controller. The control device connects the remote controller with the USB line, and the remote controller is

connected with the UAV again. The control device communicates with the remote controller through the serial interface protocol. At the same time, the remote controller communicates with UAV using light bridge transmission technology developed by DJI.

The main problem to be solved in mobile application development is Android's application development based on JAVA language, and the necessary functions are implemented on this basis.

DJI Mobile SDK and Gaode Map API mainly provide the interface for the development of UAV control communication and map operation, which makes the development process more rapid and simple.

Finally, this project will mainly realize the use of mobile applications to control UAV navigation point setting, flight point optimization and autonomous flight function.

Key Words：Traveling salesman problem; Simulated annealing algorithm; Android application; DJI Mobile SDK; Gaode Map API; UAV; autonomous flight; waypoint planning

目录

第一章 绪论

1.1 课题研究的目的和意义

1.1.1 研究目的

随着科技水平的提高和消费水平的增长，越来越多的尖端技术开始逐渐走向消费级，应用于不同的各行各业。不仅如此，人们对生活的便利程度也开始大幅度地提高。

在时代发展中，无人机技术在很多行业都得到了体现并且开始大力发展。由于门槛的逐渐降低，更多的非专业民众和行业中的工作人员也开始使用无人机。因此，专门为无人机而开发的地面站应用和控制程序变得越来越易操作，简单并且便携。不同的技术和无人机技术结合起来的研究，也开始让研究工作者们开发出更具有行业特点的产品。

本文的目的在与开发出一款更加简单，编写的无人机自主飞行以及航点规划的产品。将自主飞行和航迹规划应用到很多的行业，比如农业应用以及无人机快递应用。这样在显著地提高了效率的同时，人力的消耗也相对没有那么多了。而且避免了人为的损失以及危险情况的发生。

1.1.2 研究意义

无人机（UAV）是的含义为无人驾驶飞机，是利用无线遥控电子设备或自主研究开发的应用装置去操控的无人的飞机。

无人机的种类非常之多，可分为固定翼，旋翼等。其中，旋翼无人机是应用比较广泛地。除此之外，伞翼和扑翼也是如此。而且，无人飞艇。其中，旋翼无人机是当前无人机种类中应用做广泛，研究最深入的无人机类型，是国内外很多实验室，企业等的研究重点。

旋翼无人机是由微机电系统集成的，其中的地面控制系统用于检测无人机传回来的各种数据以及用指令对其进行一系列的操控。机载飞控和通讯系统用于对无人机在航行状态下的各种传感器数据的采集、计算、调控，以及将重要数据传送给地面控制系统。旋翼无人机按照旋翼数量上区分，旋翼无人机一般有类似于直升机的单旋翼，但其实与直升机也不同，还有几个旋翼的无人机[1]。多旋翼无人机一般有四轴四旋翼，六轴六旋翼以及

四轴八旋翼等[3]。 旋翼无人机具有能够垂直的方向起飞和降落的功能。除此之外，准确的悬停、控制非常精准、遥控方便和更人性化等优点，所以旋翼无人机除了可以在军用领域得到应用，在消费领域也开始有了发展。在消费领域中，由于操作人员的水平层次的不同，对于地面控制系统的设计非常重要。地面控制系统应该更加地简单，安全，以及适合大多数人群使用。并且在大多数人都能操控的同时，还要保证其的行业应用性。比如农业喷洒中的自主飞行，电力巡检以及物流运输的航点优化。

如今，无人机技术的发展越来越趋向于多元化。在娱乐方面，无人机可用于航拍，摄影，拍照，遥感等功能。在测控方面，无人机可以进行地理测绘，大气监测，特殊气象探测，三维建模等。在行业方面，无人机能够用于农药喷洒，交通监管，电力巡检，领空保护等。在科研方面，无人机能过进行新的技术的求证，科学考察，地质勘探，考古研究等。

所以无人机的应用非常广泛，其自主飞行起着关键的作用。比如对于航线可以自行规划，让无人机前往指定的航点飞行等。在一些人类无法到达的地域，四周环境是很难去考察的。这时候通过无人机自身传感器将数据进行采集并且自行控制其飞行是很有必要的。这样不仅能够避免人为错误导致的人力、物力以及财力损失，还能够更加保护操作人员的安全。

在自主航线规划时，更应该有对于航线的优化。例如在电力巡检中，任务点一般都在比较偏远的地区，而且巡检人员需要管理的范围非常的大。假如航线的优化使得无人机在更短的时间内完成相同的工作。这将会使得工作效率和安全保障都会有显著地提高。在物流运输中，对于经过各个城市路线的路径规划显得尤为重要。而且由于国内消费水平的提高和线上购物人群的增多，物流运输的任务也很多。所以在无人机替代传统运输方式的工作中，优化航线使其能够在更优更短的距离内到达更多的地点，这样大大的缩短了飞行总共所需要花费的时间。

1.2 国内外发展现状

1.2.1 无人机飞行控制国内外发展现状

（1）国内发展现状

我国无人机的研究是从 20 世纪中旬开始的，最先是自主研制的用于军用领域的无人

侦察机。80 年代初时开始运用于部队，逐渐的研发出“长空一号”靶机，其为军用无人机。高空无人照相侦察机系列也广泛地应用到了各个领域。20 世纪 90 年代开始在不同领域进行初步探索。由西安无人机研究发展中心主要研制的系列化小型无人机系统被国务院称为“中华之最（1949-1995）”。其也名为爱生技术公司，目前，该公司已经成功研制出 ASN 系列无人机，该无人机系列国家科技进步奖[1]。

在军用无人机领域，还开发了“WZ-2000”隐身无人机（UAV），它具有“蜂王”无人机的远航和自主导航能力[1]。直升机，这是由中国南方航空公司开发的，具有自动控制、超长距离的飞行。“翔鸟”每小时达到 150～180 公里，耐力时间为 4 小时。

随着无人机技术的普及和人们对生活水平要求的提高，消费级无人机和行业应用无人机的需求越来越大，无人机市场呈现了巨大的增长。在无人机消费市场被试探进而打开的同时，一些互联网科技企业也加入到此行列中，如腾讯科技、小米科技等。在硬件行业，海康威视也加入其中。其中要数大疆创新、零度智控和亿航科技是发展的较为成熟，特别地，大疆创新无人机以市场 70%的份额遥遥领先于其他对手。2017 年，大疆创新总销售额达 100 个亿；2018 年总销售额高达 180 个亿，可见其市场空间还非常的大[4]。

大疆创新的无人机 Phantom 系列的开始到 Mavic 系列的销售到 Spark 系列。 每一次进步都是无数工程师日夜工作的结果。 DJI 的产品已经变得越来越复杂，地面控制系统也被设计为更人性化。 例如，DJI GO，DJI GO 4 和 DJI ASSISTANT 2 用于调试设备用于消费群体控制产品。

（2）国外发展现状

国外无人机研究开始的较早，在这几十年的发展中，国外军方已经将无人机技术成熟的应用于军队中。随着技术越来越成熟，无人机的设计、性能、稳定、作用等方面都有着极大的进步和翻天覆地的变化。

早在 1939 年，美国就开始研制无人靶机。 所有 FIREBEE 系列都是在战争中发展起来的[4]。 20 世纪 50 和 60 年代，美国开发了 Fire Peak，Pioneer，Hunter，Predator 和 Global Hawk 等无人机。这些系列的无人驾驶飞机被用于越战，海湾战争，科索沃战争。 自 1993 年以来，美国的“Till”系列无人机出现在战场上。 此后，高空空域，长途飞行，长期电池寿命等技术逐渐发展起来。 在 20 世纪末期，无人机开始广泛应用于各行各业，出现了可垂直起降，自由悬停，灵活控制，操作简便的无人机[2]。 在 20 世纪 80 年代，日本的雅

马哈公司研制出用于农药喷洒的无人直升机，以减少农业成本和劳作时间。21 世纪初，日本也研制出了用于检测森林灾害、地址勘测的无人机。此外，美国国家航天局和以色列当局也多次进行合作，让技术在民用无人机领域得到了更好地发展。在国外，运用于无人机民用领域的地面控制技术也在逐渐地发展与完善。美国的软件公司 SKYCATCH 研发出一款可用于无人机地图测绘的手机应用。Free skies 科技公司研发出的 Copilot Basic 移动应用主要用于民用 3D 成像等[6]

1.2.2 无人机航点规划国内外发展现状

在一些情况下我们需要对无人机的航线做一定的调整以保证它可以实现我们需要实现的功能，这就叫做无人机航迹优化。该优化必须保证其能够在可接受范围内搜索出最佳路径。不过在不同的场合，对于无人机规划的要求都不一样。

在航迹规划中需要考虑的因素比我们想象的要复杂很多，如飞行时的安全问题：保证在飞行任务工程中安全是最重要的；飞行时的不可抗力：如物理限制，地理因素，气候环境等都是无人机航迹规划需要考虑的部分；飞行时的实时性：在飞行时会出现很多不可控的意外情况，所以我们需要实时观测无人机的飞行参数和状态以达到更优的情况等等[19]。

1980 年前期，国内的研究学者就已经开始了关于地形跟踪和回避障碍物的技术研究。直到 20 世纪 90 年代初，我国在无人机较低空的突击和防御等、对于无人机的仿真做了深入研究。我国的无人机航迹规划技术目前正在朝着多学科结合，人工智能的方向飞速发展。不过相对来说，国内的资源和水平较国外来说还有很大的进步空间。

在 20 世纪 80 年代初期，国内研究学者已经开始了地形跟踪和避障的工程研究[7]。直到 20 世纪 90 年代初，该国也取得了一定的研究成果。 例如，博士 来自北航的王立新调查低空飞机的普及情况。 孙保亭教授在控制仿真领域进行了深入的研究。 中国的无人机轨道规划技术目前正朝着多学科融合的方向发展，人工智能的方向正在迅速发展[7]。但相对而言，与其他国家相比，国内资源和水平仍有很大提升空间。

从上世纪 80 年代开始，美国的无人机航迹规划技术几乎是走在世界发展领域的前列。不仅投入的大量的资源来进行研究，更将其研究成果运用到了军用领域。美国根据不同种类如海陆空军队的需要，分别都有不同的无人机任务规划系统。所以，在海湾战争中，航迹规划技术更得到了充分地利用。

随着无人机控制技术的深入和普及，门槛的降低和操作人员趋于大众化。无人机航迹规划也在消费级和行业应用当中得到了显著的应用。在约束和不可抗力因素的条件下，根据不同的任务情况，以最高的效率得到更优的路径。

例如在 DJI 的电脑版地面站应用中，无人机可以自己规划航线，或者根据操控人员指定的航线飞行。在农业领域的方面，无人机可以直接实时地检测农田的情况，然后根据此情况绘制图像。

图 1.1 大疆 PC 地面站例图

1.3 研究的主要内容和结构安排

1.3.1 主要内容

本文研究内容一是无人机控制应用的设计和开发，二是加载地图通过跟地图的交互达到对航点的上传，三是通过已开放的 SDK 实现对无人机和操控，四是运用合适的优化算法来对路径进行优化并且在应用中成功实现。

1.3.2 结构安排

本文章节安排如下所示：

第一章为绪论，主要介绍了该项目的研究目的和意义，对无人机做了简单的描述和该项目目前在国内外的研究现状。

第二章描述了无人机自主飞行与航点规划应用开发概述及技术基础。分别对研究的

无人机类型，飞行通讯，用到的开发平台以及各种 SDK 和 API 做了详细的介绍。

第三章和第四章主要对优化算法和 TSP 问题做了介绍，并且根据实际情况和算法的仿真结果选择了最适合的路径优化算法再将最后的结果用 JAVA 代码实现，实现后运用到项目的实际开发过程中。

第五章和第六章介绍了项目的方案设计和开发以及实现的详细过程，并对其进行总结。

第七章主要是对测试环境做了简单的介绍并且给出该项目的测试结果，以及对测试结果的分析和展示。

第八章为总结与展望，主要总结该项目存在不足的地方和希望以后能够有所改进。

第二章 自主飞行和航迹规划应用开发概述及技术基础

2.1 旋翼无人机

无人机的种类非常之多，旋翼无人机因其操作方便、能够垂直起降、结构稳定等众多优点成为研究的重点。不仅如此，目前多旋翼类无人机已经广泛地应用于航拍、测绘、巡检、物流等各个领域。所以本文所研究的项目也将使用多旋翼无人机作为飞行器（以下简称无人机）[3]。

下面将简单地介绍以下无人机的组件以及硬件设备。为了更直观地了解其构成，我们以 DJI PHANTOM 系列无人机为参照。

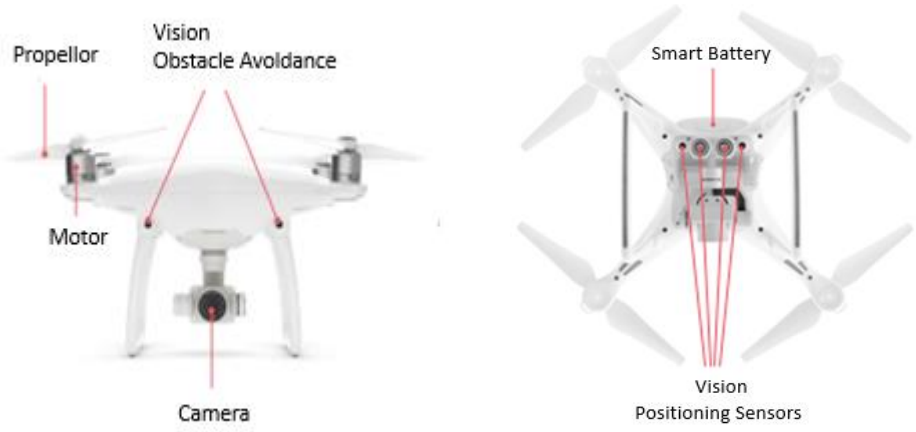


图 2.1 大疆精灵无人机正视图和俯视图

（1）推进系统（Propulsion）：推进系统包括螺旋桨（Propeller）和电机（Motor）。当电机安装上螺旋桨以后，可以为无人机提供垂直推力。每一个旋翼上垂直推力的大小可通过电机的工作来调整，当电机的转速变快，带动起来的垂直推力增加，反之也成立。无人机可通过垂直推力的改变，来达到飞行的悬停、旋转、上升、下降、水平前进、水平后退、水平左移、水平右移。

（2）传感器（Senor）：目前的无人机多依靠各种传感器的同时精准工作，来进行多传感器信息融合最后实现避障、测距、测速、测绘等等一系列工作。其中最重要的传感器包括：加速度计，陀螺仪，指南针，气压计，超声波传感器，摄像头和卫星定位系统。

加速度计用于测定当前无人机的加速度，陀螺仪用于测定无人机是否水平，指南针用来测定无人机的方向，摄像头用作图像的采集并且为后期的图像处理做准备，采集无人机绝对距离这些信息我们将采用气压计，超声波传感器用于测定无人机与周围物体的相对距离，卫星定位系统用于确定无人机当前的位置。这些传感器用来确定当前的状态以及预测飞机未来的状态及其周围的环境。

（3）飞行控制器（Flight Controller）:飞行控制器是无人机组件中非常重要的一个部分。飞行控制器相当于一台机载计算机，用于整合各个传感器的信息，然后根据飞行需要来调整每个螺旋桨的推力以及按要求来飞行。

（4）相机（Camera）：是无人机应用的附加部分，主要用于各种应用环境。摄像机可以在本地录制图像和视频数据，并且可以将采集到的信息传输到地面站或者移动设备上。

（5）视觉避障与定位系统（Vision Obstacle Avoidance & Positioning）：为了适应不同的需求，比如说在室内无 GPS 的环境下飞行。用于视觉避障与定位的传感器和摄像头（或者提供图像，或者提供二维信息）可以帮助无人机感知周围的世界。这些传感器和摄像头一般固定在无人机的某个方向或者某几个方向，被用来检测附近是否有障碍物。位于无人机下方的这些传感器用于确定相对的地面位置。以保证在无 GPS 的情况下能够提供精确的速度估计和稳定的悬停。现在大量用的视觉避障通常都是双目视觉避障或多目视觉避障。

（6）智能电池（Smart Battery）：智能电池提供了无人机运行时所需要的能量，一般为锂电池。目前为了提高续航能力也有研发出氢氧燃料电池。智能电池和飞行控制器结合起来可以预估剩余的飞行时间并且给出一定的警告和操作。

（7）遥控器和移动设备（Remote Controller & Mobile Device）：在无人机操作中，需要存在无人机和操作人员之间的媒介，这个媒介就是遥控器。

为了适应更多的需求和信息的传递，通常需要搭载显示设备。一般都整装在遥控器上的显示设备还有目前应用非常多的将移动设备用于显示设备的情况。

除此之外，在通过 WIFI 通讯的情况下，有些无人机也可以直接使用移动设备进行控制。

本项目的研究采用大疆创新的 Mrtice100 开发型无人机作为飞行平台（以下简称无人机）。

图 2.2 大疆 M100 无人机图示

2.2.坐标系介绍

目前，通常用两个坐标系来描述无人机飞行的具体位置和方向。一个是相对于机身和机身框架的坐标系，被称为 body 系；一个是相对于地面和世界位置的坐标系，被称为 word 系[18]。

（1）机身坐标系（Body Coordinate System）

机身坐标系是用来描述无人机自身的飞行状态的。机身坐标系一共有三个轴，分别为 X 轴、Y 轴和 Z 轴。我们将根据右手定则来建立三个坐标轴与机身方向的关系。

设定质量中心为原点 O,无人机的前方为 X 轴的方向，无人机的右方为 Y 轴的方向。那么穿过飞机底部的垂直向下的方向就是 Z 轴的方向。

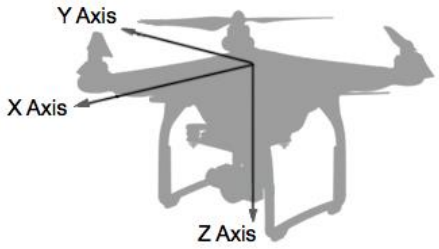


图 2.3 机身坐标系图示

根据机身坐标系就能够描述相对于自身的飞行方向。X 轴的正向平移和反向平移分别表示了无人机前进飞行和后退飞行的方向；Y 轴的正向平移和反向平移分别表示了无人机向左飞行和向右飞行的方向；Z 轴的正向平移和反向平移分别表示了无人机向上飞行、向下飞行的方向。

除此之外，无人机的正旋转方向也可以用坐标轴规则来描述。

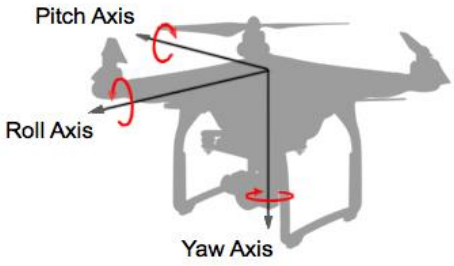


图 2.4 无人机旋转方向图示

如图所示，沿着 X 轴旋转被定义为滚转（Roll），沿着 Y 轴旋转被定义为俯仰（Pitch），沿着 Z 轴旋转被定义为偏航（Yaw）。

另外想让无人机执行滚转、俯仰、偏航的任务时一定要小心是否与 X、Y、Z 轴飞行方向冲突，以免造成不必要的损失。

（2）地面坐标系（Ground Coordinate System）

描述无人机在地面（世界）坐标系的位置时，将无人机的正向与 X 轴正向对应。X 的正向为北，Y 的正向为东，根据右手定则，Z 轴的正向为下。这个定义被称为 North-East-Down（NED）。

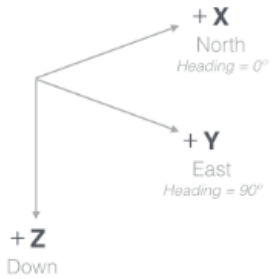


图 2.5 地面坐标系图示

在描述中，NED 坐标系的原点通常是世界中的一个位置，如起飞时的位置。0 °航角表示的是正北的方向，90 °航角表示的是正东的方向。

2.3 智能定位控制（Intelligent Orientation Control）

IOC 在 Mobile SDK 中被叫做飞行定位模式。飞行定位模式定义无人机应该如何解释飞行指令。也就是选择了无人机飞行所对应的参考目标。

在默认地情况下，无人机的相对参考目标是自身以及自身的前进方向。所以当给无人

机一个右命令时，无人机会朝着自身前进方向的右方飞去。但当无人机操控人员正前方与无人机前进方向不一样时，操作会显得有些困难。在超视距飞行时，无人机操作人员也无法准确判断无人机第一视角的飞行方向。

为了解决这类情况，我们定义了两种模式，分别为 Course Lock 和 Home Lock。这两种模式能够解决无人机飞行指令是相对于无人机操作人员还是无人机。

（1）Course Lock

Course Lock 模式规定了无人机应该相对于固定的航向移动。如在这种模式下当我们选定了以 0°航向飞行，那么不管当时无人机的偏航有多少，无人机都会朝着正北方飞行。

（2）Home Lock

Home Lock 模式规定了无人机应该相对于固定原点移动。当选择前进或后退时，无人机会远离或者靠近固定原点飞行。当选择向左飞行或者向右飞行时，无人机会沿着固定原点到自身距离的圆线向左或向右飞行。

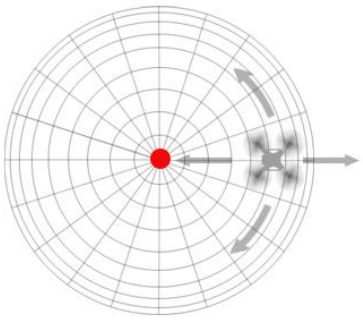


图 2.6 无人机 home lock 飞行定位模式图示

2.4 飞行通讯

在无人机飞行时，通讯是一件非常重要的事情。无人机操控人员需要向无人机发送指令，无人机需要正确接收指令并且将有关信息反馈给无人机操控人员或各种检测操控设备，一般信息都通过无线链路来传播。根据飞行需要，不同的无人机具有不同的通讯方式。

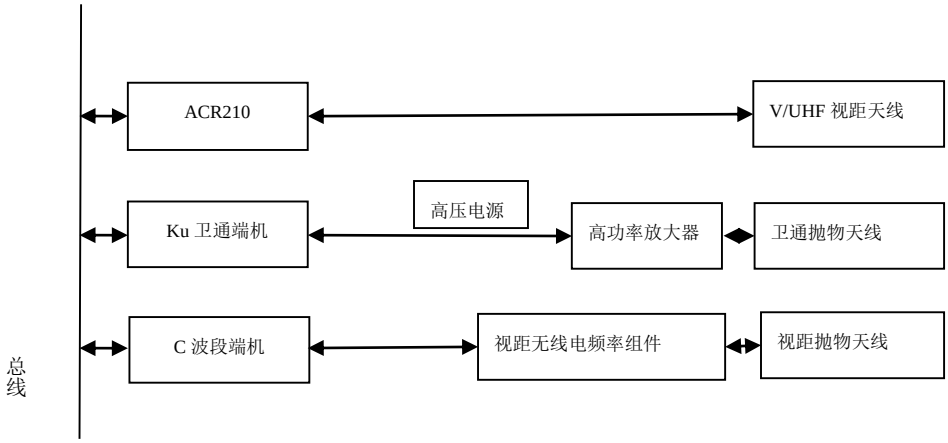


图 2.7 军用无人机通信需求

对于军用无人机，通信要求是极高的安全等级，范围和低通信延迟。 因此，对于这种无人机，通常使用可提供超视距和中继通讯的卫星。 在中高海拔的长途飞行中，无人机通常在 7-8 公里的高度飞行。美国的“捕食者”视线传输使用 5.25 至 5.85 GHz 的 C 波段，卫通数据链使用 12.5 至 18 GHz 的 Ku 范围，并且还使用语音中继信道。 这种超短波 ARC-210 波频率范围为 30～512 MHz[15]。

对于消费级和行业应用级无人机，通信需求与军用无人机相比较为简单,无线图传功能性好。即使需要超视距飞行，距离也在较小的范围以内。目前消费级无人机的通讯系统所使用的设备一般为遥控设备和显示设备结合；控制设备。其中，控制设备通讯可分为利用 Wi-Fi 无线信号传输和利用遥控器已有的无线信号传输。

在通讯系统中，一般需要有的为遥控系统和图传系统。遥控系统将飞行指令以一定的通讯协议发送给无人机。图传系统是无人机根据需要将摄像头的图像、飞行数据等信息再传送到地面设备。目前图像传输系统使用的信号频率为 5.8 GHz，GPS 使用的信号为 1.2 GHz，遥控系统所使用的信号是 2.4Ghz。

在本文项目中，采用的是控制设备利用现有遥控器已有的无线信号来控制无人机的方式来操作。

控制设备利用 USB 线连接遥控器，遥控器再与无人机无线连接。同时遥控器与无人机利用大疆研究出的 light bridge 图传技术进行通讯。

2.5 开发平台

本项目所使用的无人机设备是 DJI M100。针对于 DJI 的飞控系统，该公司专门开放

了 SDK（Software Development Kit）即软件开发工具包供开发者使用。目前的 SDK 一共有三种，分别是支持移动设备的、支持机载设备的和支持视觉开发的。

DJI Mobile SDK 目前支持开发两种操作系统的移动应用，一是 iOS 系统，另一个是 android 系统。

2.5.1 iOS 系统

开发出 IOS 系统的公司为苹果公司，目前苹果公司已经成功地将它运用到手机、平板、电脑、穿戴设备等[5]。iOS 系统架构层次一共有四层，它们有底层到顶层分别为 Core operating system 层、core service 层、media 层、touch application 层。iOS 系统的优点在于它的安全性，其将安全级别调的很高并写入了系统硬件底层。对于保护隐私等方面有着非常大的好处。

目前开发基于 iOS 系统的移动应用的语言有 Objective-c 和 Swift，开发平台为 X-code。

2.5.2 Android 系统

Android 是谷歌公司推出的，安装在移动设备上的系统，可以支持各种设备，如 mobile phones, tablet computers, televisions, on-board devices 和 wearable devices。 据统计，其全球市场份额已超过 80%。在 2007 年，谷歌免费向全世界开放了整个系统的源代码。 所有手机制造商都可以免费使用 Android 系统，因此快速席卷全球。

Android 系统架构层次同样分为四层，它们分别为应用软件层、应用框架层、系统运行库层、Linux 内核层。目前开发基于 Android 系统的应用的语言为 java，开发平台有 Eclipse 和 Android Studio。但是现在 Google 公司已经几乎不再对 Eclipse 进行维护，并且在极力推荐 Android Studio 开发平台。基于需求是使用方便、操作简单，本项目将使用 Android 系统进行开发[6]。

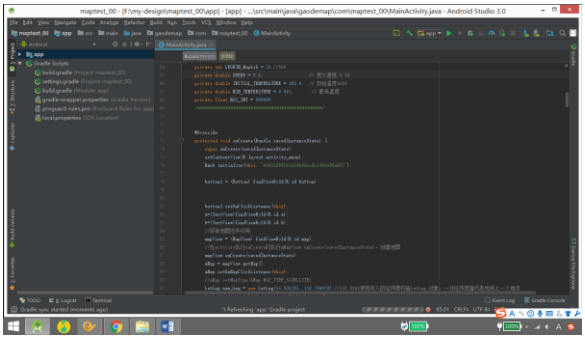


图 2.8 Android 系统开发环境

2.6 Mobile SDK 与高德地图 API

上文提到，本项目决定开发基于 Android 系统的应用来实现各项功能。所以具体使用到的语言为 Java，开发平台为 Android Studio,用到的 SDK 分别有 Android SDK、DJI Mobile SDK、与 GAODE MAP API。

2.6.1 Mobile SDK

DJI Mobile SDK 是 DJI 专门为自己的飞无人机和手持设备所开放的移动应用的软件开发工具包，能够实现很多对无人机和手持设备的控制任务。SDK 能够对无人机的云台、相机、通信、电池、飞控、遥控器等一系列设备进行控制。

在使用 SDK 进行开发时要详细地了解它的类别，其一共有四个类别，分别是管理类（SDK Manager）、产品类（Product）、组件类（Component）、任务类（Mission）、任务控制类（Mission Control）。

管理类：包括 SDK 的注册，产品的连接，提供对产品本身的访问

产品类：针对于无人机和手持设备，支持基本的产品性能和主要的产品组件

组件类：该类提供了组件的控制，信息状态，和包含的子组件

任务类：该类描述了不同的任务比如航点任务，主动跟随任务和性能状态等

任务控制类：该类处理任务的执行，并且可以设置让一些任务同时执行

2.6.2 高德地图 API

软件工程师可以用高德地图提供的接口在自己的应用当中使用高德地图的部分功能。其优点是通俗易懂，操作简单并且用户使用率非常高。目前高德地图提供各种不同设备的开发，可使用的功能不仅有二维和三维世界的地图功能，定位功能，导航功能，除此之外还有关于出行路线的规划功能，搜索某个地点的功能。目前更开放出了支持一部分开放室内场所的室内搜索功能，室内定位功能。

第三章 旅行商问题与航点规划概述及技术详解

3.1 旅行商问题简介

旅行商问题也称为旅行推销员问题,其英文描述是 Travelling salesman problem ，以下简称为 TSP 问题。给出的描述是从一个城市的起点到所有城市到起点的最短路径。 这是一个经典的组合优化问题也是一个 NP 难的问题，即多项式复杂性的非确定性问题。 事实上，根据图论，TSP 问题是完全无向加权图中权重最小的哈密尔顿回路[8]。

早在 1975 年，欧拉在当时提出了骑士环游问题，就是在国际象棋的 64 个棋盘中走完 64 个格子并且返回到起点。直到 1954 年，TSP 问题有了进展，当时的研究者运用线性规划中的割平面法解决了美国 64 个城市的 TSP 问题。后来还提出了分支界限法来求解此问题。

TSP 问题还分为好几类，分别是经典 TSP 问题、不对称 TSP 问题、配送收集 TSP 和多人旅行商问题以及多目标旅行商问题。

经典 TSP：在一个带权的完全无向图里寻找权值最小的哈密顿回路就成为经典的旅行商问题。

不对称 TSP：假如两个点的权值不一定相等，例如在交通环境中,高峰期 A->B 用的时间比 B->A 长，这就叫做不对称 TSP。其实在生活中，不对称 TSP 问题的情况比经典 TSP 的情况更常见。

配送收集 TSP：此类问题是根据如今的物流配送问题而产生的。假如将从需求点发往配送中心的货物和从配送中心发往需求点的货物都放在一辆或限量的货车上，怎样才能求出行驶路线的最短哈密顿回路。

多人旅行商问题：这在车辆调度中，物流中都有着重要的意义。即在每个城市只被一个人经过的情况下，n（n>1）个人最后把所有城市都走完的最短路径。

多目标旅行商问题：此问题研究的是每条路径都有多个权值，所以要在此情况下使回路上的权值都尽可能小。

由于旅行商问题在很多地方都有着非常重要的作用。比如物流配送、航线安排、路径

规划等问题上，还有现在的印制电路板穿孔、在人类基因排序中绘制放射性杂交图、光缆布线、晶体结构分析等。所以大量的研究学者都在攻克这一问题。

在早期，研究者常用非常精确的算法去解决该问题如枚举法、回溯法、分支界限法、线性规划法、动态规划法等。但随着研究的深入，当计算量呈爆炸型增长的时候，精确算法往往就效率比较低了。如今，研究者从很多自然现象和规则中发现了很多解决优化问题的思想。进而衍生出了现在的近似算法和启发式算法，近似算法其实并不能保证找到的解为最优解，但一般可求得近似解。如：遗传算法、模拟退火算法、蚁群算法、神经网络等[8]。目前，大量的实验数据已经能够保证近视算法和启发式算法能够在数据量非常大的情况下将误差控制在 1%以内。

3.2 TSP 问题算法比较

目前研究 TSP 问题的算法有很多，对算法的分类还没有一个的具体的定论。本论文研究时分类方式是按照算法类型来分。

按照算法类型：一种是仿生类算法，仿生类算法顾名思义就是根据自然现象发现解决组合优化问题的思想[11,12,13]，如 Genetic 算法, ant colony 算法, simulated annealing 算法等，仿生类算法也是经典的优化算法。非仿生类算法的意思是利用数学的观点来解决组合优化的问题，在这种类型的算法当中又分为两大类型的算法，一种是精确算法另外一种是局部启发式搜索算法。精确算法如：枚举法、动态规划法等；局部启发式搜索算法如贪心法等[9]。

3.2.1 基于非仿生类的算法

（1）精确算法

a) 枚举法

枚举法顾名思义是将所有可能的路径列举出来最后得到最短的路径。将起点设为根节点，依次将没有经过的城市作为叶子节点直到所有城市都已经完成。判断其中哪一条路径最短，哪一条最短哪一条就是最优的路径[10]。

枚举法的优点在于算法简单，最终能够保证得到精确解。枚举法例如城市数量有 13 个，那么使用该方法需要计算 13*（13-1）！种路径，其运行时间可以用小时来计算。所以

在进行数量较多的 TSP 问题规划时，不宜用枚举法。

b) 动态规划法

动态规划的思想为一个大的问题分解成多个阶段的问题，再根据问题的联系逐个解决以至于得到最后大问题的最优解。由此可见，将动态规划的思想运用到求解 TSP 问题是可行的。

其基本的原理是，我要求得起始城市经过其他城市后到达起始城市的最优解，那么在此之前我先在剩下的城市作为一个城市集合，再从中选取一个城市。问题就变为从起始城市，到达这个选定的城市的距离加上再剩下的城市的城市集后回到起始城市的最短路径。这样层层分解下去，变成很多个子问题。最后求得该总问题的答案。

抽象为数学方程可表示成：假设起始城市为 S，d（i,V）表示的是城市 i 经过剩余城市集 V 回到起始城市的最短路径。当 V 为空集时，则 $d(k,V'-\{k\})$表示城市 i 到起始城市没有其他剩余城市，所以 d（i,V）等于 C_{is}，其意思是表示两个城市的距离。当 V 不为空集时，d（i,V）表示的是城市 i 经过剩余城市集 V 回到起始城市的最短路径。可写成 $d(i,V')=\min\{C_{ik}+d(k,V'-\{k\})\}$，再求解这个子问题的最优解。K 表示你在当前剩余城市集 V’中所选择的城市。

由此可以看出，在精确算法中，动态规划算法的时间复杂度比枚举法要高很多。所以在一定数量内，可以使用动态规划方法求解 TSP 问题。但当城市数量很大时，显然动态规划法所用的时间依旧很长。

（2）局部启发式搜索算法

贪心算法的意思是在当前的环境下，只考虑当下的情况去求得一个最优的解。贪心策略指的是在贪心算法中得到局部最优解的方法，贪心策略是根据具体问题而定的，所以没有一个确定的贪心策略。

在 TSP 问题中，贪心算法具体描述是首先遍历与起始城市连接的所有城市，以最短距离的城市作为下一个城市，不断地循环直到回到最初的城市。

下面给出核心算法说明：

算法 3.1 贪心算法

输入：n:节点数；a:连接矩阵

输出：a[i][j~n-1]中的最小元素

描述：

1：定义行、列允许矩阵 row[n]

2：赋初值：s=0,i=0

3：

1：while row[i]=1

2：j=0,m=a[i][0],k=0

3：找到第一个允许访问的节点 a[i][j]

4：寻找 a[i][j~n-1]中的最小元素

5：end while

其中，n 为节点数也就是城市数量。连接矩阵 a 为各个城市之间的距离，如果城市之间没有连通则设为无穷。行列允许矩阵表示该城市是否被访问过，如果被访问过则设为 0，没有被访问过则设为 1。

由于贪心算法是只从当前状态下获取最优解的一种算法，做得出的答案只是局部的最优解。在贪心算法中不能够保证最终获得的解为最优的最终解，但是贪心算法的效率相对而言还是很高的。并且，当前很多启发式算法本质上都是从贪心算法和随机算法结合而来的，所以贪心算法的重要性还是非常高的。

3.2.2 基于仿生类的算法

（1）蚁群算法

研究发现当蚂蚁在寻找食物时会通过释放信息素来告知蚁群是否经过这条路，信息素的浓度和路程呈正相关。当下一只蚂蚁经过时，会以前面蚂蚁留下信息素的大小来判断自己是否走这条路并且留下一定的信息素。许多次循环以后，逐渐的蚁群会找到一条通往目的地的最短路径。不过，随着时间的推移，信息素的浓度也会有一定量的衰减[14]。流程图如下：

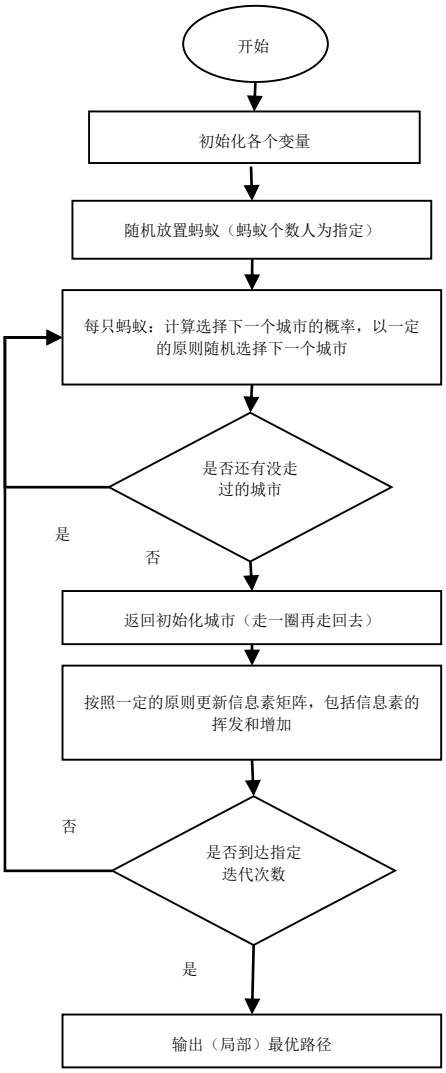


图 3.1 蚁群算法流程图

蚁群算法采用了正反馈机制和并行计算，在效率上要高于其他类似的算法。但由于迭代次数的限制，在计算过程中可能还没有达到最优解就已经停止计算，所以容易陷入局部最优解中。

（2）模拟退火算法

模拟退火算法是研究者根据物理固体退火现象的得出的一种优化算法，在 1982 年由 Kirkpatrick 发现。

在物理上，退火的意思为物体从在温度很高，内能很大的情况下逐渐降低温度，内能变低。当物体重新变为晶体的时候，内能变为最低。如图所示，首先物体处于非晶体的状态，我们将固体温度升高，固体内能增加从而其内部的分子变为无序状态；再让物体缓慢

冷却，每一次冷却物体都能找到一个平衡态；当最后温度不再下降时，物体变为基态，其内能最小，分子也从无序变成有序排列[20]。

所以，内能的变化就可以看作是 TSP 问题中寻找最短路径的变化。但是要注意的是退火和淬火的区别，退火是让温度徐徐降温，是物体缓慢达到稳定状态；而淬火是让物体急速降温，其所凝结的非晶体并不是内能最低的状态。

模拟退化算法中很重要的一部分就是 Metropolis 准则。Metropolis 准则的意思是粒子在温度 T 时趋于平衡的概率为 e-△E/(KT)，其中 e 为温度 T 时的内能，Δ E 为其改变量，k 为玻尔兹曼常数（单个气体分子的平均平动动能随热力学温度 T 变化的系数，根据此准侧来判定是否能够接受当前温度的内能。

根据物理固体退火过程而研究出的模拟退火算法具有相当大的优越性，精确算法能够得到最优解但是效率太低；局部启发式搜索算法如贪心算法虽然效率较精确算法高但是精确度不够高；在经典算法中如蚁群算法遗传算法等虽然效率较高、但很容易走入局部最优解。所以在计算量大，需要保证效率的同时又要保证精确度的情况下，模拟退火算法是很好的选择，因为模拟退火算法能够依据 Metropolis 准则接收一定的恶意解，从而跳出走入局部最优解。

3.2.3 实验结果对比

本小节选择了贪心算法和模拟退火算法进行实验比较，实验环境为 VS2010，Windows8。本文分别选择了城市数量为 6、24、48 的城市组作为测试数据。针对每一组数据分别用贪心算法和模拟退火算法来求解，实验数据结果如下所示：

表 3.1　城市数量为 6（已知最短路径长度为 15.5375）

	仿生算法	非仿生算法
	模拟退火算法	贪心算法（最近邻）
最短路径长度/m	15.6350	16.2450
平均路径长度/m	15.6350	16.2450
平均耗时/s	1.25	0.12

表 3.2 城市数量为 14（已知最短路径长度为 30.8785）

	仿生算法	非仿生算法
	模拟退火算法	贪心算法（最近邻）
最短路径长度/m	30.8785	31.3707
平均路径长度/m	30.8785	31.3707
平均耗时/s	10.37	0.15

表 3.3 城市数量为 48（已知最短路径长度为 335.2）

	仿生算法	非仿生算法
	模拟退火算法	贪心算法（最近邻）
最短路径长度/m	349.6	368.3
平均路径长度/m	349.6	368.3
平均耗时/s	16.53	0.70

注：平均路径长度是算法运行 10 次求得路径长度的平均长度，平均耗时是指 10 次运行所耗时间的平均值。

分析实验结果，我们可以得到如下结论：

由表可知，对于城市数量规模不大的时候，最近邻算法和模拟退火算法都可以求到近似的有效解的，如表 3.1 所示。但是当城市数量规模越来越大的时候，模拟退火算法的效果就比最近邻算法效果好，如表 3.3 所示。不过在耗时中，由于模拟退火算法中循环次数很高，所以时间上会比最近邻算法大很多。

除此之外，在贪心算法中，由于贪心策略不同所得到算法的效率和准确度会不一样。

3.3 基于模拟退火算法航点规划问题的表示和实现

在航线的优化中，每一个经纬度构成的坐标（航点）相当于 TSP 问题中的城市的位置；航点与航点的距离相当于不同城市之间的距离；TSP 问题相当于在航点之间，在最有效率的情况下找到一条最优的航线使无人机从起点起飞经过所有航点有且仅有一次最后再回到起点。前文已经提到模拟退火算法是根据物理固体退火过程而研究出的启发式随机算法[17]。

下面给出 Metropolis 抽样算法的描述，见公式 3.1：

$$p=\begin{cases}1 & ,\ E(x_{new})<E(x_{old})\\ \exp\left(-\frac{E(x_{new})-E(x_{old})}{T}\right) & ,\ E(x_{new})\geq E(x_{old})\end{cases} \tag{3.1}$$

其中 X_{new} 是根据前面的随机算法的出来的新解，然后 X_{old} 表示的是；$E(X_{new})$表示新解路径的长度，$E(X_{old})$表示旧解路径的长度。当新解路径的长度小于旧解路径的长度时就接受这个解，当新解路径的长度大于旧解路径的长度时，应该用那$\exp\left(-\frac{E(x_{new})-E(x_{old})}{T}\right)$概率来判断是否接受这个新解[14]，其中 T 为获得当前新解时的温度。流程图如图所示：

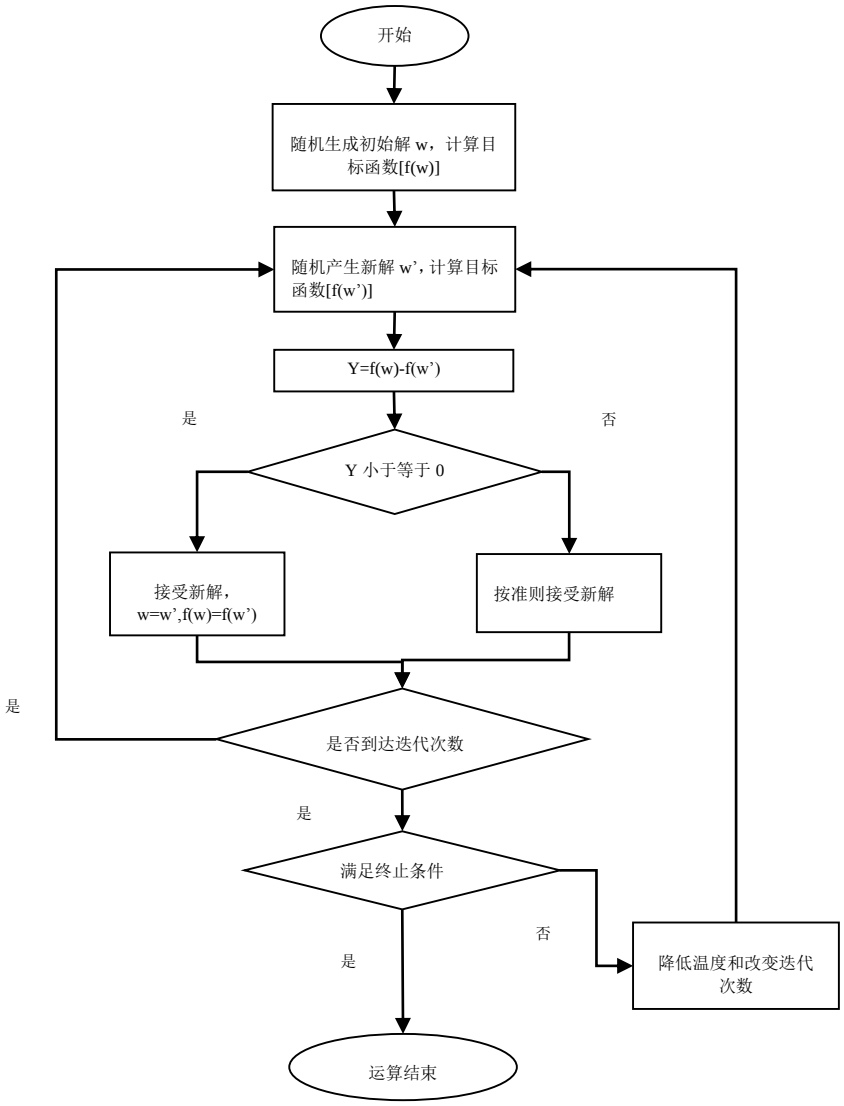


图 3.2 模拟退火算法流程图

由此可知，模拟退火算法中有很重要的几个要素，这几个要素对最后的结果有着很大的影响。它们分别是问题规模、初始温度、退火速度、Mapkob 链长度、随机算法。

航点的数量越大的时候算法的效率会降低；如果初始温度设置的较低、退火速度设置的较快、或者 Mapkob 链的长度较短，算法虽然收敛的快但是得到最优解的概率会变小；同样随机算法也会对结果有影响，假如随机算法的选取比较单一，那么得到最优解的概率也会变小。

根据以上情况，结合航点任务航点多、解空间起伏较大、对效率和精确度都有一定要求的情况下。本项目中将初始温度设置为 1000，退火速度设置为 0.98，Mapkob 链长度设置为 500 以保证航点任务的准确性和效率。

3.4 基于模拟退火算法航点规划方案的实现

在将模拟退火算法应用到航点规划的实际过程中，一共有以下几个重要部分。

（1）利用航点的经纬度将航点之间的距离转换为二维矩阵。核心代码为：

```
for(int i=0;i<Vex_num;i++){
        for(int j=0;j<Vex_num;j++)
        {
            distans= AMapUtils.calculateLineDistance(latLngs.get(i),latLngs.get(j));
            G.arcs[i][j]=distans;
        }
    }
```

（2）用随机算法对初始路径中的航点进行交换、逆置、移位等以得到新的路径，从中选取优化路径。核心代码为：

```
int choose=(int)(Math.random()*3);;
 if (choose == 0)
 {    // 随机交换任意两个城市的位置
      char temp = bestSolution.path[i];
      bestSolution.path[i] = bestSolution.path[j];
      bestSolution.path[j] = temp;
 }else if (choose == 1)
 {    // 随机逆置城市的位置
      List<Character> Temp =new ArrayList<>();
      int n=i;
      for(int m =n;m<=j;m++){
```

```
                Temp.add(bestSolution.path[i]);
                i++;
        }
            i=n;
            Collections.reverse(Temp);
            int a=0;
            for(int m=n;m<=j;m++){
                bestSolution.path[i]=Temp.get(a);
                i++;
                a++;
            }
          //    reverse(bestSolution.path + i, bestSolution.path + j);
        }
        else{     // 随机移位城市的位置
            if (j+1 == G.vex_num) //查看是否是边界，是否需要处理
            {
                newSolution = bestSolution;
                return newSolution;
            }

            List<Character> Temp =new ArrayList<>();
            int n=i;
            for(int m =n;m<=j;m++){
                Temp.add(bestSolution.path[i]);
                i++;
            }
            i=n;
            Collections.rotate(Temp,(int)Math.random()*(Temp.size()));
            int a=0;
            for(int m=n;m<=j;m++){
                bestSolution.path[i]=Temp.get(a);
                i++;
                a++;
            }
           // rotate(bestSolution.path + i, bestSolution.path + j, bestSolution.path + j + 1);
        }
```

（3）这个优化路径是否可以被接受要使用 Metorpolis 准则。核心代码为：

```
 if ((int)exp((Best_solution.length_path - Current_solution.length_path) / Current_Temperature)*100 >
(random.nextInt(101)))
                    {
                        Best_solution = Current_solution;
                    }
```

第四章 无人机自主飞行与航迹规划应用方案设计

4.1 需求规定

4.1.1 对功能的规定

本项目包括 APP 注册、无人机连接、地图基本操作、航点任务管理、根据航点执行任务几大功能。

（1）具体要求如下：

a)APP 注册:

在用户第一次打开应用时，需要对 APP 进行相应的注册，注册成功后才能够使用相应的 SDK（即软件开发工具包）。本项目主要用到的 SDK 有 DJI mobile SDK 和 Gaode MAP SDK，所以要在提供这两种软件开发工具包的运营商那里获取其 SDK key[16]。

b)无人机连接

打开 APP 时，为应用设置一个监听器，不停地监听是否有无人机进行连接。当再 APP 注册后无人机连接成功，才能够继续其他的功能。

c)地图基本操作

利用网络实时加载地图，确保能够看到无人机目前在的位置和成功地添加航点、设置飞行参数，再将所有的信息上传至无人机。并且在无人机进行航点任务的时候实时在地图上加载轨迹。

d)航点任务管理

对任务的各种信息进行设定，比如速度、航向等。

e)根据航点执行任务

包括上传航点任务、执行飞行任务、飞行任务中紧急暂停、稳定降落。

（2）确定参与者

飞行操控人员、无人机

（3）确定用例

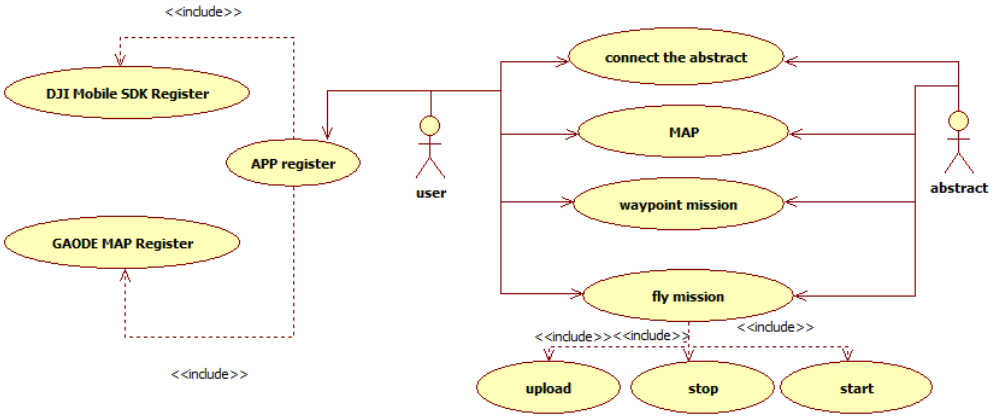


图 4.1 用例图

（4）用例说明

a) APP 注册用例

用例名：APP 注册管理

描述：应用 SDK 的注册

参与者：用户

前置条件：应用没有被注册过

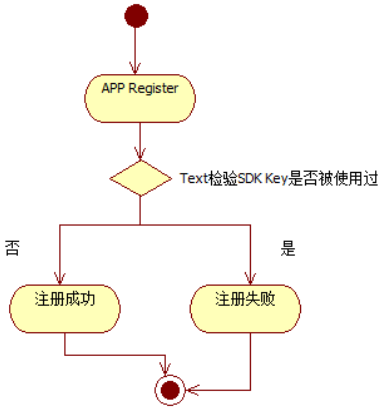


图 4.2 APP 注册流程图

b) 无人机连接

用例名：无人机连接管理

参与者：用户、无人机

前置条件：应用注册成功

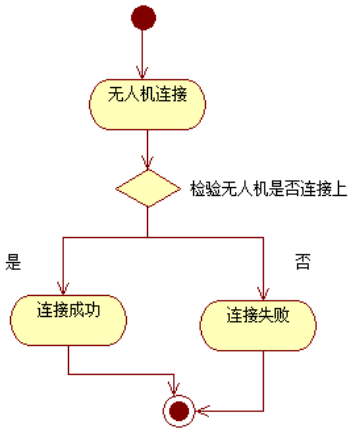


图 4.3 无人机连接流程图

c) 地图基本操作

用例名：地图基本操作管理

参与者：用户

前置条件：无人机已连接

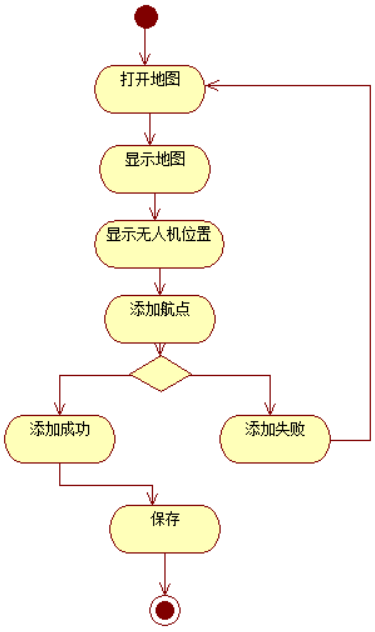


图 4.4 地图基本操作流程图

d) 航点规划

用例名：航点规划管理

参与者：用户

前置条件：航点构造成功

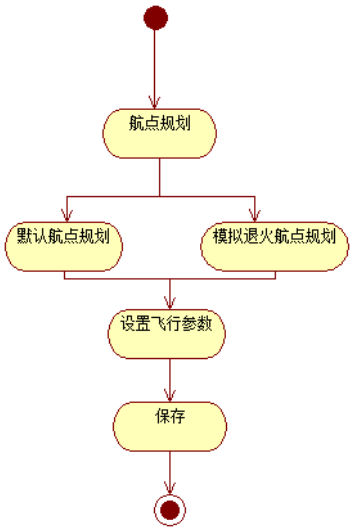


图 4.5 航点规划流程图

e) 飞行任务

用例名：飞行任务管理

参与者：用户 无人机

前置条件：航点规划成功

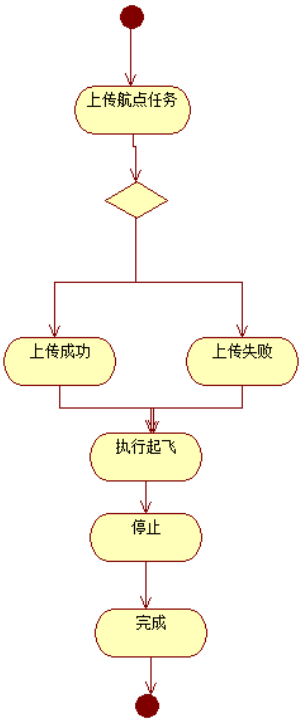


图 4.6 飞行任务管理流程图

4.1.2 对性能的规定

（1）精度

保证数据上传的准确性

保证实时显示轨迹的时候的准确性

（2）时间特性要求

飞行器连接所用时间不超过 3 秒

航线规划时所用时间不超过 1 秒

上传航线任务所用时间不超过 0.5 秒

（3）灵活性

操作方式：以手机触摸屏为主要操作方式；

运行环境：在安卓 4.0 系统到安卓 5.0 系统之间；

计划的变化或改进：整体的包容性大，将每个方法封装起来，运用到的时候就调用这个方法，那么当这个块出错的时候就只改这一块就行了；

（4）故障发生要求

发生错误时，第一时间保证操作人员安全和无人机是否能被控制。使无人机能够有自我保护措施，在发生错误时能够自动悬停或者缓慢降落。

4.1.3 运行环境规定

（1）设备

处理器和内存：P1 及以上，内存大于 32M；

外存容量：10G

设备：移动应用，无人机；

数据通信设备：USB，控制器，无人机；

功能键及其他专用硬件：快捷键，遥控器

（2）接口

高德地图接口，DJI Mobile SDK 接口

（3）控制

串行接口协议和飞行通讯协议

4.2 概要设计

根据需求规定，系统总体结构主要包括移动控制应用、遥控器和外部设备（无人机）。

在通信方面，基于串行接口协议（USB 协议）和 OTG 技术，将遥控器作为 Slave、手机作为 host，移动控制应用通过 USB 设备将数据传送给遥控器。遥控器和无人机之间通过 DJI 的 Lightbridge 技术进行传输。Lightbridge 图传系统采用 2.4GHz 频率进行传输，Lightbridge 中控制器的链路为 2.4G,并且加上了 OSD 飞行数据叠加系统。

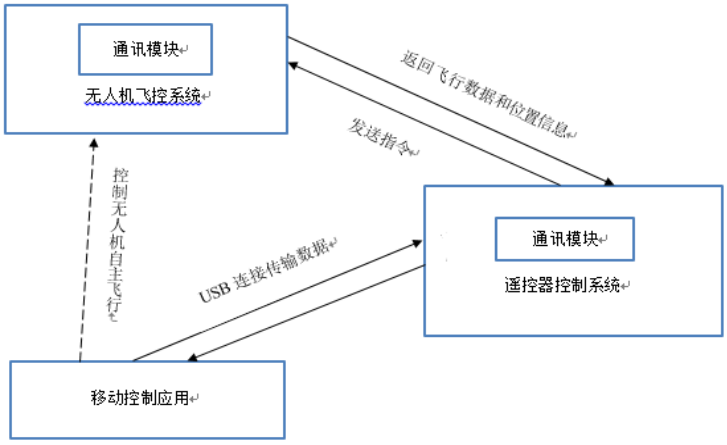


图 4.7 系统结构图

在整体结构中分为 APP 注册、无人机连接、地图基本操作、航点任务管理、根据航点执行任务五大模块。

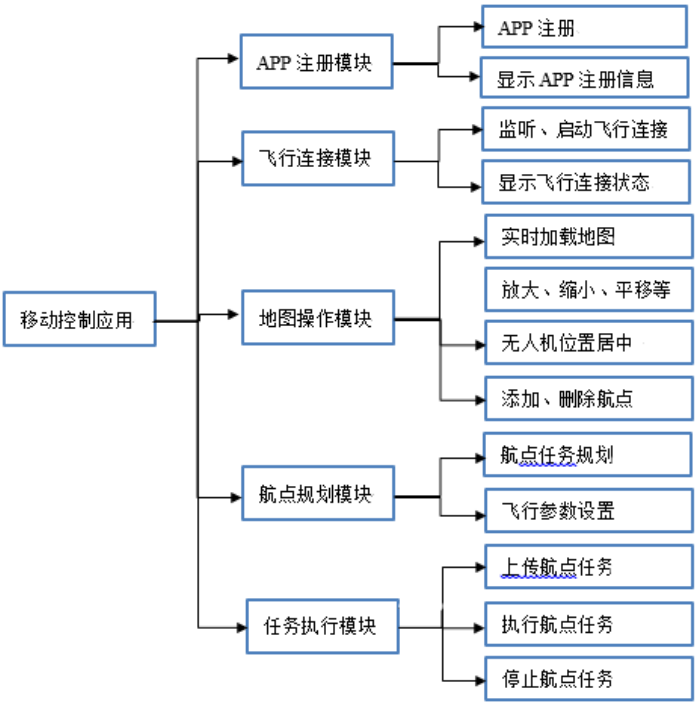


图 4.8 任务模块图

4.3 详细设计

4.3.1 界面设计

根据 APP 的结构，应用将界面分为三大部分，分别是 APP 注册无人机连接界面、主功能界面、设置飞行参数界面。

（1）APP 注册无人机连接界面

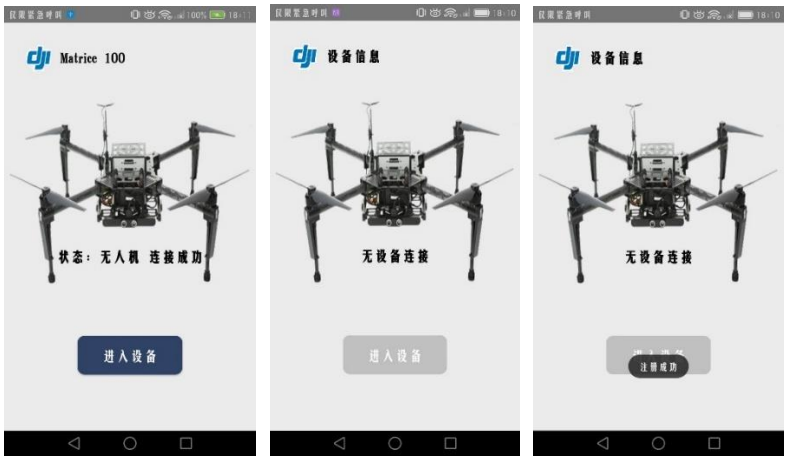
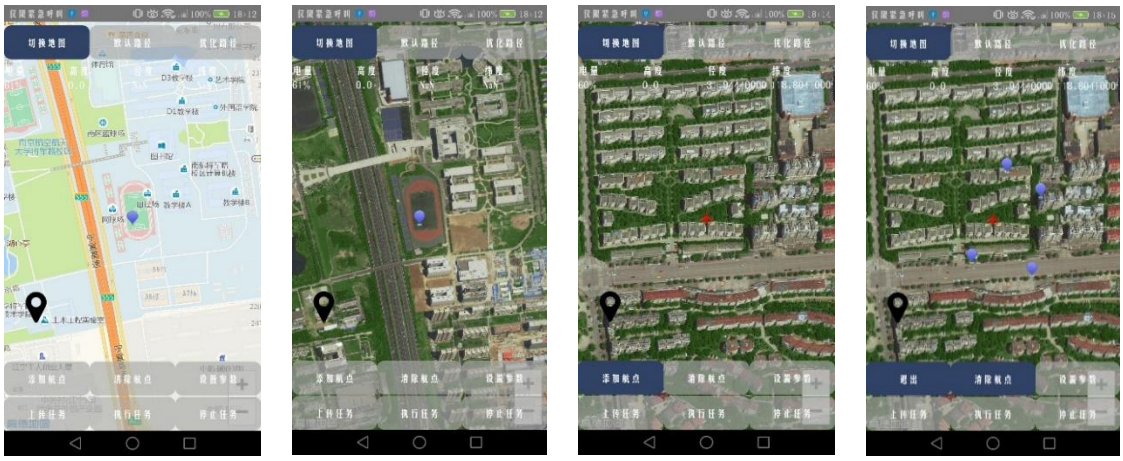


图 4.9 APP 注册无人机连接界面

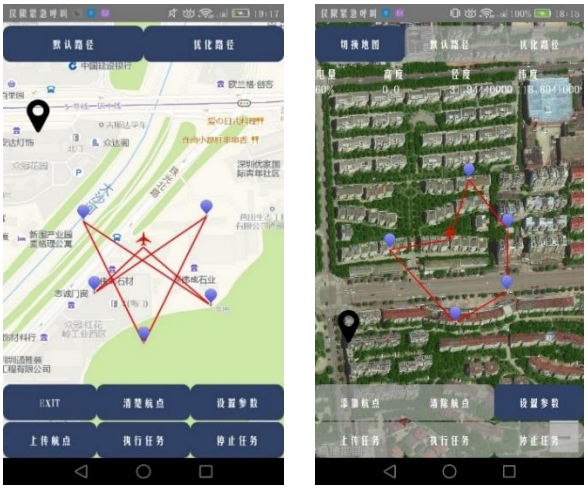
界面的状态分别为 APP 未注册界面、APP 注册成功界面和连接成功界面

（2）主功能界面



（a）地图加载　（b）切换地图　（c）无人机定位　（d）无人机航点添加

图 4.10 地图显示界面



（a）默认路径　　（b）优化路径规划

图 4.11 无人机航点规划

点击进入设备以后，应用加载地图，地图可以有标准模式和卫星模式。添加航点后，可以进行默认路径规划也可进行优化路径规划。

（3）飞行参数设置界面

图 4.12 飞行参数设置

界面为飞行参数设置界面，分别对无人机的高度、速度、完成后降落方式以及航向进行设置。

4.3.2 功能模块设计

该应用的事物逻辑还是以功能模块来描述较为方便

（1）APP 注册

打开应用后 APP 会自动注册。若注册成功则会返回注册成功并开始无人机连接；反之会返回其具体缘由

（2）飞行连接模块

在 APP 注册成功后，启动飞行连接功能并实时监听飞行连接状态。假如连接成功，则返回连接产品信息并且可以打开地图；假如连接失败，则返回提示信息以及失败原因。

（3）地图操作模块

打开地图后，可以使用相应的手势操作。点击定位的图标后，界面会将无人机的位置显示在地图的正中；假如失败，则会提示定位功能出错，尝试重新定位。

前面的操作正确以后，点击添加按钮，则可以对无人机添加航点；点击清除按钮，则可以对当前航点进行清除。

（4）航点规划模块

这里一共有两种模式，一种是默认路径另一种是优化路径。点击默认路径按钮，航点顺序将与添加航点顺序相同；点击优化路径按钮，航点顺序会以模拟退火算法规划出来的航点顺序进行排序。

点击设置参数按钮，即可对飞行的高度、速度、完成后降落方式以及航向进行设置。

（5）任务执行模块

在前面的工作都做好的时候，点击上传按钮，控制器会将任务上传到无人机；假如上传失败，则返回失败信息以及失败原因。

点击执行按钮，假如成功则会返回成功信息并且无人机开始执行任务、地图会实时显示无人机的位置；假如失败则会返回失败信息以及失败原因。

点击停止按钮，无论在任何状态下，无人机都会停止飞行任务等待下一步指令。

第五章 无人机自主飞行与航点规划应用整体开发与实现

5.1 APP 注册

先检查是否开启权限。在权限开启的情况下，添加 DJI Mobile SDK 和 Gaode Map SDK。

```
<meta-data
    android:name="com.dji.sdk.API_KEY"
    android:value="f9310c5169f9e0df9be84b17" />

<!-- 启用高德地图服务 -->
<meta-data
    android:name="com.amap.api.v2.apikey"
    android:value="83e49eb55706a86f1d9e2e90d5484204" />
```

开启 SDK 服务:

```
int permissionCheck = ContextCompat.checkSelfPermission(this,
android.Manifest.permission.WRITE_EXTERNAL_STORAGE);//权限
int permissionCheck2 = ContextCompat.checkSelfPermission(this,
android.Manifest.permission.READ_PHONE_STATE);//权限
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M || (permissionCheck == 0 &&
permissionCheck2 == 0)) {

    //This is used to start SDK services and initiate SDK.被用来开始 SDK 服务和开启 SDK
    DJISDKManager.getInstance().registerApp(this, mDJISDKManagerCallback);
} else {
    Toast.makeText(getApplicationContext(), "Please check if the permission is granted.",
Toast.LENGTH_LONG).show();//请检查权限是否被授予
```

5.2 飞行器连接

连接产品前要保证 APP 注册成功，在连接前设置监听，实时地保持连接信息的更新。

```
//为了接收移动设备的连接改变，注册广播接收器
IntentFilter filter = new IntentFilter();
filter.addAction(DJIDemoApplication.FLAG_CONNECTION_CHANGE);//连接是否改变
registerReceiver(mReceiver, filter);
```

开始连接无人机

```
public static synchronized BaseProduct getProductInstance() {//得到产品实例
    if (null == mProduct) {
        mProduct = DJISDKManager.getInstance().getProduct();
    }
```

显示无人机连接状态和信息

```
BaseProduct mProduct = DJIDemoApplication.getProductInstance();//连接产品

if (null != mProduct && mProduct.isConnected()) {//产品连接上
    Log.v(TAG, "刷新 SDK 成功");//刷新 SDK 成功
    mBtnOpen.setEnabled(true);

    String str = mProduct instanceof Aircraft ? "无人机" : "手持设备";
    mTextConnectionStatus.setText("状态: " + str + "  连接成功");

    if (null != mProduct.getModel()) {
        mTextProduct.setText("" + mProduct.getModel().getDisplayName());//显示产品的名字
    } else {
        mTextProduct.setText(R.string.product_information);//产品信息
    }
```

5.3 地图基本操作

5.3.1 加载地图

初始化地图容器

```
mapView = (MapView) findViewById(R.id.map);
mapView.onCreate(savedInstanceState);

initMapView();
```

加载地图

```
private void initMapView() {
    //初始化地图控制对象
    if (aMap == null) {//定义 AMap 地图对象的操作方法与接口
        aMap = mapView.getMap();//获取地图控制器 AMap 对象。
```

5.3.2 切换地图

点击切换地图，地图可在卫星模式和普通模式下切换

5.3.3 放大、缩小、平移

在 Boolean 等于 1 的时候，开启此项功能；在 Boolean 等于 0 的时候，关闭此项功能。

```Java
UiSettings.setScrollGesturesEnabled(boolean);
```

5.3.4 无人机位置居中

获取无人机当前位置

```
private void updateDroneLocation(){
    LatLng pos = new LatLng(droneLocationLat, droneLocationLng);
    final MarkerOptions markerOptions = new MarkerOptions();//标记
    markerOptions.position(pos);//设置当前 MarkerOptions 对象的经纬度。
markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.aircraft));//标记的图标
    latLngs.add(pos);
    runOnUiThread(new Runnable() {//利用 Activity.runOnUiThread(Runnable)把更新 ui 的代
码创建在 Runnable 中
            @Override
        public void run() {
            if (droneMarker != null) {//地图上一个点绘制图标
                droneMarker.remove();//删除当前图标
            }
            if (checkGpsCoordination(droneLocationLat, droneLocationLng)) {//正确
                droneMarker = aMap.addMarker(markerOptions);//加一个标记在地图上
```

对地图进行操作，实现缩放功能，让无人机标记显示在地图中心

```
LatLng pos = new LatLng(droneLocationLat, droneLocationLng);
//??
float zoomlevel = (float) 18.0;//缩放级别
CameraUpdate cu = CameraUpdateFactory.newLatLngZoom(pos, zoomlevel);//返回一个
CameraUpdate 对象
aMap.moveCamera(cu)
```

5.3.5 添加、删除航点

用下面的这句话监听地图容器

```
aMap.setOnMapClickListener(this);
```

重写 onMapClick 方法，使其一直获得手势交互时点击的位置

在地图上添加点：

```
private void markWaypoint(LatLng point){
    //Create MarkerOptions object 创建一个记录点目标
    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(point);
    String s1 = String.valueOf(point.latitude);
    String s2 = String.valueOf(point.longitude);
markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE
)).title(s1).snippet(s2);//用蓝色
    Marker marker = aMap.addMarker(markerOptions);//加入标记
    mMarkers.put(mMarkers.size(), marker);
    //添加优化路径中的航点
    LatLng latLng =new LatLng(point.latitude,point.longitude);
    latLngs.add(latLng);
}
```

删除航点：

```
case R.id.clear: {
    runOnUiThread(new Runnable() {//更新 ui 界面
        @Override
        public void run() {
```

设置航点：

```
waypointList.clear();
for(int i=0;i<latLngs.size()-2;i++){
    Waypoint newwaypoint =    new Waypoint(latLngs.get(i+1).latitude,latLngs.get(i+1).longitude,
altitude);
    if (waypointMissionBuilder != null) {
        waypointList.add(newwaypoint);//构造航点

waypointMissionBuilder.waypointList(waypointList).waypointCount(waypointList.size());//任务点
的数量

    } else {
        waypointMissionBuilder = new WaypointMission.Builder();
        waypointList.add(newwaypoint);
        waypointMissionBuilder.waypointList(waypointList).waypointCount(waypointList.size());
    }
    Log.w("waypointlatings",latLngs.get(i+1).toString());
    Log.w("waypoint",waypointList.get(i).coordinate.toString());
```

5.4 航点任务规划

5.4.1 航点规划

添加好航点后，可以分为默认路径和优化路径。默认路径按照添加顺序进行航点规划。下面介绍优化航点任务的核心算法：

（1）将航点距离转换为邻接矩阵

```
for(int i=0;i<Vex_num;i++){
            for(int j=0;j<Vex_num;j++)
            {
                distans= AMapUtils.calculateLineDistance(latLngs.get(i),latLngs.get(j));
```

（2）在 MIN_TEMPERATURE 和 mapkob 链规定的范围内进行计算，每次都找到一个优化的解作为当前解

```
while(MIN_TEMPERATURE < Current_Temperature){
        for (int i = 0; i < LEGNTH_Mapkob; i++)
    {
        Current_solution = FindNewSolution(G, Best_solution_copy);
```

（3）在优化中，以随机数控制当前航点改变个数以及排列方式。排列方式分别有交换、逆置、平移三种。

```
int choose=(int)(Math.random()*3);;
        if (choose == 0)
        {    // 随机交换任意两个城市的位置
            char temp = bestSolution.path[i];
            bestSolution.path[i] = bestSolution.path[j];
            bestSolution.path[j] = temp;
        }else if (choose == 1)
        {    // 随机逆置城市的位置
            List<Character> Temp =new ArrayList<>();
            int n=i;
            for(int m =n;m<=j;m++){
                Temp.add(bestSolution.path[i]);
                i++;
        }
            i=n;
            Collections.reverse(Temp);
            int a=0;
            for(int m=n;m<=j;m++){
                bestSolution.path[i]=Temp.get(a);
                i++;
                a++;
            }
          //   reverse(bestSolution.path + i, bestSolution.path + j);
        }
        else{    // 随机移位城市的位置
            if (j+1 == G.vex_num) //查看是否是边界，是否需要处理
            {
                newSolution = bestSolution;
                return newSolution;
            }

            List<Character> Temp =new ArrayList<>();
            int n=i;
            for(int m =n;m<=j;m++){
                Temp.add(bestSolution.path[i]);
                i++;
            }
```

（4）最后根据 Metropolis 准则来判断当前计算得出的解可不可以被接受

```
if ((int)exp((Best_solution.length_path - Current_solution.length_path) / Current_Temperature)*100 > (random.nextInt(101)))
                    {
                        Best_solution = Current_solution;
                    }
                    else{
                        // cout<<"不接受当前解."<<end }
```

5.4.2 设置飞行参数

在飞行参数界面，分别对飞行高度、速度、完成后降落方式、以及航向进行设置。

```
waypointMissionBuilder = new WaypointMission.Builder().finishedAction(mFinishedAction)
                                                      .headingMode(mHeadingMode)
                                                      .autoFlightSpeed(mSpeed)
                                                      .maxFlightSpeed(mSpeed)
                                                      .flightPathMode(WaypointMissionFlight
PathMode.NORMAL);
```

5.5 航点任务执行

5.5.1 上传任务

将航点和飞行参数都规划好以后，将任务进行上传

```
private void uploadWayPointMission(){//上传点任务

    getWaypointMissionOperator().uploadMission(new CommonCallbacks.CompletionCallback() {
        @Override
        public void onResult(DJIError error) {
            if (error == null) {
                setResultToToast("上传任务成功!");
            } else {
                setResultToToast("上传任务失败, error: " + error.getDescription() + " 请再重试一遍.");
                getWaypointMissionOperator().retryUploadMission(null);
```

5.5.2 执行任务

上传成功后才可以执行任务，主要代码如下

```
private void startWaypointMission(){//开始任务

    getWaypointMissionOperator().startMission(new CommonCallbacks.CompletionCallback()
{
        @Override
        public void onResult(DJIError error) {
            setResultToToast("开始执行任务: " + (error == null ? "Successfully" :
```

5.5.3 停止任务

在执行任务过程当中，可以随时停止任务以防止意外情况发生。停止时无人机将悬停在当前高度。

```
private void stopWaypointMission(){//停止任务

    getWaypointMissionOperator().stopMission(new
CommonCallbacks.CompletionCallback() {
        @Override
        public void onResult(DJIError error) {
            setResultToToast("任务停止: " + (error == null ? "Successfully" :
```

第六章 无人机自主飞行与航迹规划应用测试与分析

6.1 DJI ASSISTANT 2 介绍

在测试中我们需要用到 DJI 提供的模拟测试软件 DJI Assistant2。

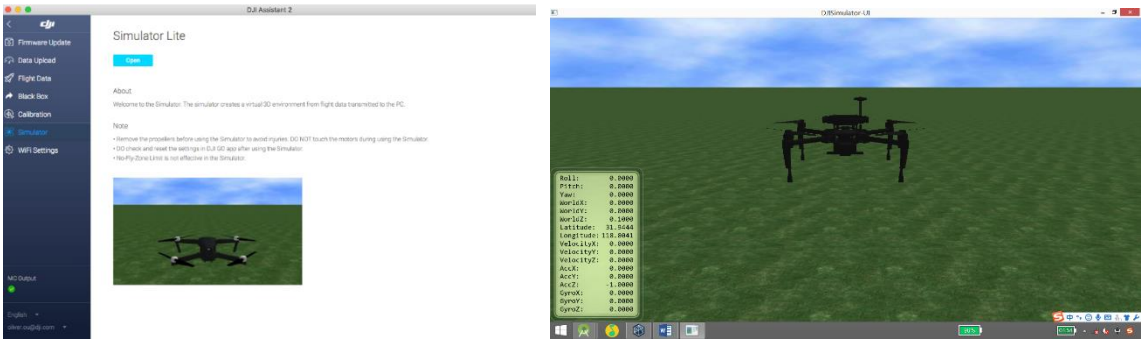


图 6.1 DJI 调参和模拟飞行软件 Assistant 2

将无人机利用 USB 线与 PC 机进行连接，连接成功才可以进入 DJI Assistant2。在模拟测试软件中，我们可以查看当前无人机的硬件信息、飞行数据；上传和下载飞行数据；并且会对无人机的各个传感器进行校准以保证其稳定性；其中有一个模拟飞行的功能，就是将无人机实体连接 PC 机，手动模拟此时无人机的位置，给出无人机模拟环境。

在模拟飞行中，首先设置当前无人机虚拟的经纬度、风速等一系列数据。设置后利用控制设备控制此无人机，就能够看到其在模拟飞行中的实时状态。

模拟飞行的优点在于安全性能高，仿真性强。在本项目中，可以在安全地环境下准确的观测到无人机的飞行状态。

实际飞行后，也可通过 DJI Assistant2 查看实际的飞行数据，从而验证应用功能的正确性。

6.2 飞行测试步骤

（1）：首先打开电脑上的调参软件然后用 USB 线无人机和电脑连接，最后也用

USB 线将控制器和移动设备连接

（2）：点击无人机开关按钮，再打开遥控器，等待其指示灯由红色变为绿色就表示连接成功

（3）：打开调参软件中的模拟器，输入模拟初始定位的经纬度

（4）：打开移动应用，选择添加航点

（5）：选择默认路径或者优化路径，设置飞行参数

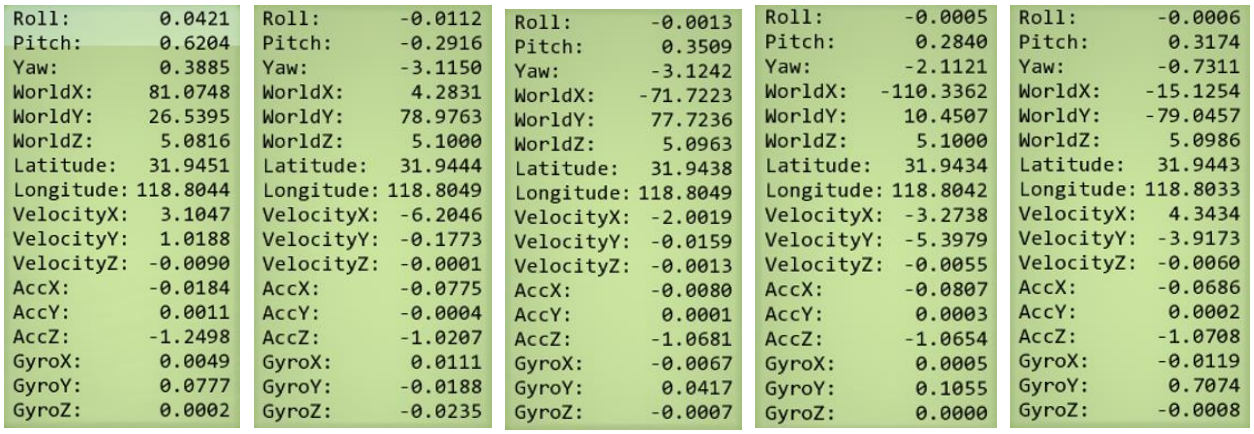
（6）：选择上传任务，等待任务上传成功

（7）：选择执行任务，任务开始执行

6.3 测试结果

6.3.1 模拟飞行测试结果

选取 5 个航点，执行任务，观察模拟器中的数据是否正确。如图所示：



（a） （b） （c） （d） （e）

图 6.2 到达各点时模拟器数据展示

在图中，可以看到 worldZ 代表当前无人机所在高度，Latitude 和 Longitude 分别代表径度和纬度。图中的数据表明无人机准确的到达了这五个航点。

6.3.2 航点路径优化测试结果

选取 22 个航点，规划路径如下：

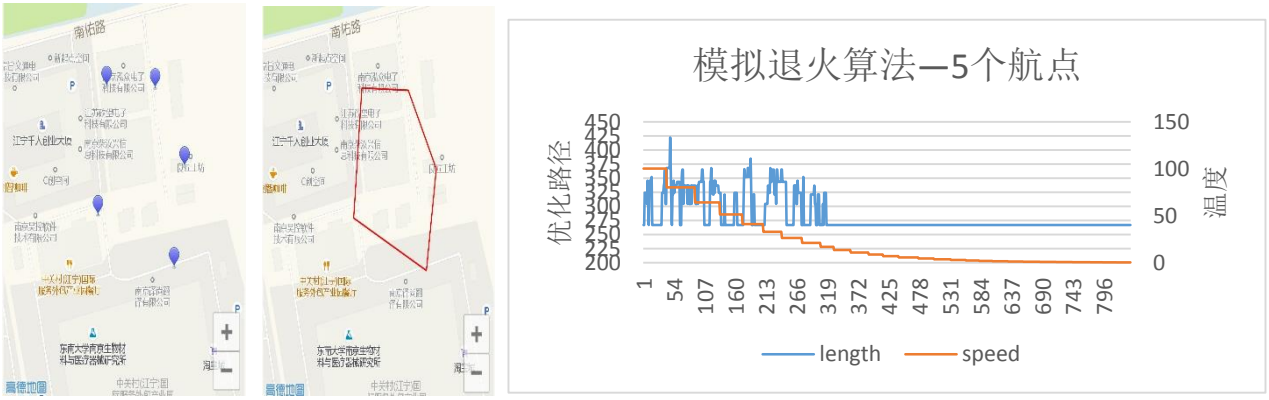
（a）航点优化前　　(b）航点优化后　　（c）任务执行中

图 6.3 航点优化过程

（a）表示航点优化前，界面显示的是手动添加的航点。（b）表示优化路径后的航线。（c）表示的是任务执行，任务执行时，地图中的红色飞机标志会随着无人机实时的经纬度而动态地发生改变，并且显示在界面上。

模拟退火算法部分结果展示如下：

（1）设置航点个数为 5

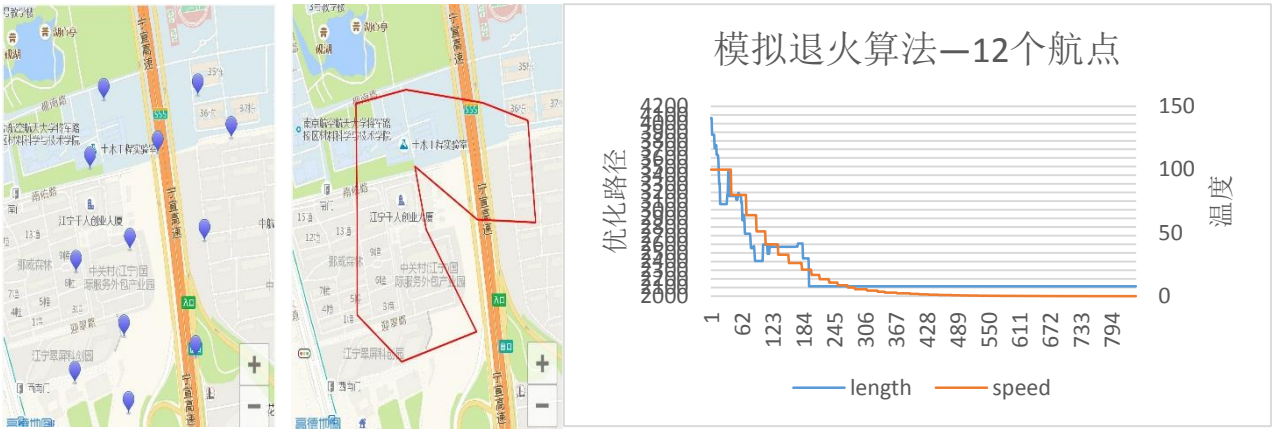


（a）添加航点时　　（b）航点优化　　（c）数据图

图 6.4 航点个数为 5 时数据展示

注：在数据图中，左边的纵坐标代表优化路径的长度、右边的纵坐标代表温度，横坐标代表运算的次数。蓝色的线表示的是优化路径的变化、橙色的线表示的是温度的变化。下同。

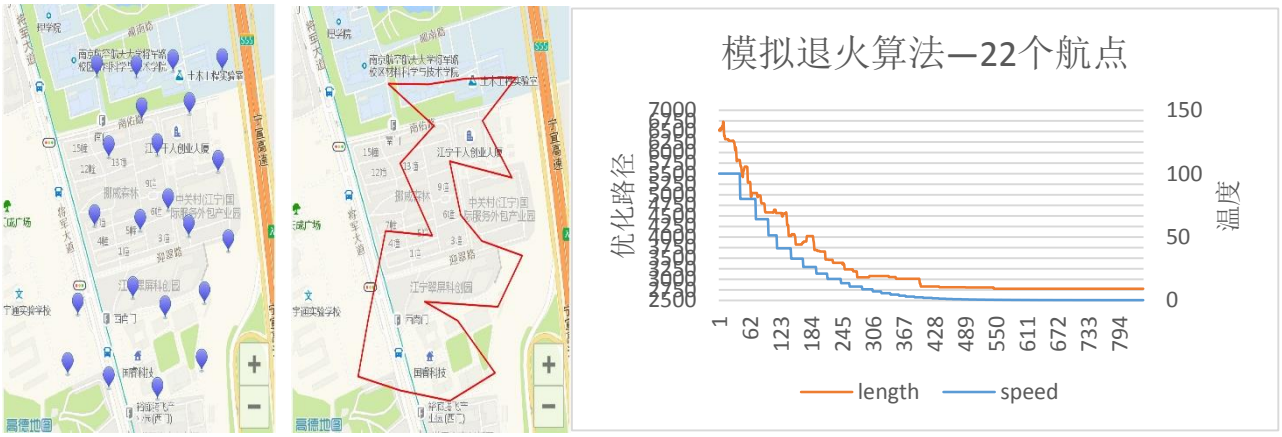
（2）设置航点个数为 12



（a）添加航点时　　（b）航点优化　　（c）数据图

图 6.5 航点个数为 12 时数据展示

（3）设置航点个数为 22



（a）添加航点时　　（b）航点优化　　（c）数据图

图 6.6 航点个数为 22 时数据展示

6.3.3 数据结果分析

由图 6.4，图 6.5 和图 6.6 中的数据图对比可得知，当航点规模问题较小的时候，因为循环次数的限制，模拟退火算法无法体现出它的优越性。在图 6.4（c）中，当航点数为 5 的时候，算法的收敛性并不高，因为航点数越少所对应的路径条数也就越少。不过在图 6.5（c）图和图 6.6（c）可以看出，当航点数越来越大时，算法的收敛性变强，得到最终的解。

第七章 总结与展望

7.1 总结

毕业设计已经到了最后阶段，大学生涯也进入尾声。在这个阶段中我也学到了很多在学习中没有学到的知识。在做毕业设计时，系统地将安卓应用开发、无人机接口技术和高德地图接口技术学习了一遍并加以应用，研究了 TSP 问题的各种算法，并试着将算法应用到论文当中；在写论文时，也更深入的理解了各方面的知识比如飞行原理、飞行通讯等，为以后的学习打下了扎实的基础。

在本项目中还存在着一些不足之处，界面的交互性、应用的稳定性、算法的效率等都有待提高。除此之外，在实际的飞行中，保证其安全也是很重要的。比如设定范围，当无人机超过其范围就自动不允许飞行等。不过在这过程中，我对自己的成果还是较为满意的，希望在未来能够越做越优良。

7.2 展望

关于项目，希望在以后的研究中能够让其路径优化算法得到更好地应用；同时在将旅行商问题和无人机路径规划问题结合的这种情况下，路径优化算法能够得到更深入的研究并且投入使用。因为无论是在农业喷洒、测绘、检测等各种行业应用中，怎样最有效率的航迹规划是非常重要的。

关于自己，希望以后能够坚定不移的走自己所想要的路。从不停止学习、学以致用；从不停止反思，持续提高；但无论在未来的道路上做出什么样的选择，都要告诉自己要懂得满足、不用后悔。

参考文献

[1] 秦博, 王蕾. 无人机发展综述[J]. 飞航导弹, 2002(8):4-10.
[2] 殷强. 四旋翼无人机自主控制系统研究[D]. 天津大学, 2012.
[3] 王文建. 四旋翼无人机控制系统设计[D]. 新疆大学, 2017.
[4] 陈晓莹. 大疆汪滔:就是要在“无人机”领域让中国人成为第一[J]. 南方企业家, 2017(9).
[5] Miller F P, Vandome A F, Mcbrewster J. iOS (Apple)[M]. Alphascript Publishing, 2010.
[6] Felt A P, Ha E, Egelman S. Android permissions:user attention, comprehension, and behavior[C]// Proceedings of the Eighth Symposium on Usable Privacy and Security. ACM, 2012:1-14.
[7] 巴海涛. 无人机航迹规划研究[D].西北工业大学,2006.
[8] Nöhring F. Traveling Salesman Problem[J]. Operations Research, 2007, 4(1):61-75.
[9] 樊骥, 吴红梅. 基于最优化理论几种算法的比较[J]. 才智, 2011(9):53.
[10] 曹新谱. 算法设计与分析[M]. 湖南科学技术出版社, 1984.
[11] 戴三, 陈恭洋, 周云才. 旅行商问题(TSP)算法比较[J]. 计算机与数字工程, 2013, 41(9):1445-1447.
[12] 王晓东. 数据结构:C++语言版[M]. 科学出版社, 2008.
[13] 石利平. 模拟退火算法及改进研究[J]. 信息技术, 2013(2):176-178.
[14] 霍艳丽. 面向路径规划的多策略和变异算子蚁群算法研究[D]. 南昌大学, 2015.
[15] 丁鲲, 吴志建. 对“全球鹰”无人机通信网干扰作战仿真实验研究[J]. 航天电子对抗, 2015, 31(5):5-8.
[16] 刘波. 基于 Mobile SDK 的无人机自主飞行控制技术研究[D]. 西安石油大学, 2017.
[17] 高晓静, 李俊山, 赵宗涛,等. 基于模拟退火算法的航迹规划方法研究[J]. 微电子学与计算机, 2000, 17(5):10-14.
[18] 许云清. 四旋翼飞行器飞行控制研究[D]. 厦门大学, 2014.
[19] 叶听. 无人机地面站软件设计及航迹规划算法研究[D]. 浙江大学, 2016.
[20] Bertsimas D, Tsitsiklis J. Simulated Annealing[J]. Statistical Science, 1993, 8(1):10-15.

致　　谢

转眼间四年的大学生活已经进入尾声。本文是我大学期间学习研究的见证和总结，在此之际，谨向那些过去给予了我无私帮助和耐心指导的老师和同学表达最由衷的感谢。向自己，表达衷心的感谢。

首先，感谢我的毕设指导老师翟象平老师。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。感谢翟老师在过去的时间内，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。

感谢我的朋友们，感谢我们有幸能遇见，能一起相处四年的时光。你们是我大学生涯中最宝贵的财富。

感谢我自己，能够一直不放弃，不抛弃，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。

最后感谢我的家人，感谢他们的养育之恩，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，谢谢你们！