

# 《无人机摄像及应用》结课报告

## 航迹规划算法与实现

刘宇轩<sup>1</sup>, 李顺龙<sup>2</sup>, 郭亚鹏<sup>2</sup>, 王安东<sup>2</sup>, 刘一诺<sup>1</sup>, 何家震<sup>1</sup>, 齐露雅<sup>1</sup>,  
石艾琳<sup>1</sup> 与张芮嘉<sup>1</sup>

<sup>1</sup> 哈尔滨工业大学 《无人机摄像及应用》(TS22505)2025 秋季班 第三组

<sup>2</sup> 哈尔滨工业大学 交通科学与工程学院 桥梁结构安全评定青年科学家工作室

\* 班级、学号等信息请参考文后致谢部分

## 摘要

在全球桥梁数目与规模逐渐增大的背景下，传统的纯人力方式与手动操控无人机方法已经难以维持多种工作进行。因此，引入自动化控制的无人机飞行工作已经成为共识。

而航迹规划是无人机自动控制中必不可少的一环，就此，本文将对航迹规划问题进行以下研究操作：

（一）航迹图图形化或栅格化。

（二）使用多种路线规划算法对多种给定飞行情形进行自动航线规划，包括基于图的传统基本航迹规划算法，例如模拟算法，迪杰斯特拉算法 (Dijkstra) 和 IDA\* 搜索算法，以及基于机器学习的现代智能路线规划算法，例如遗传算法、蚁群算法 (Ant Colony Algorithm, ACA)、改进灰狼优化算法 (Improved Grey Wolf Optimizer, IGWO) 和布谷鸟搜索算法 (Cuckoo Search, CS)，以满足不同现实情境下的不同工作需求。

（三）利用航迹平滑算法，包括三次样条插值法 (Cubic Spline Interpolation, Spline 插值) 和贝塞尔曲线法 (Bézier Curve) 算法，对航迹进行实际航迹可行化处理。

**关键字:** 无人机，航迹规划，规划算法，最优化问题

## **Abstract**

Path planning is an essential component of autonomous drone control. In this regard, this paper will conduct the following research on the path planning issue:

(1) Graphical or grid-based representation of path maps.

(2) Utilization of various route planning algorithms for automated route planning under multiple given flight scenarios, including traditional graph-based fundamental path planning algorithms, such as simulation algorithms, Dijkstra's algorithm, and IDA\* search algorithm, as well as modern intelligent route planning algorithms based on machine learning, such as genetic algorithms, ant colony algorithms, improved grey wolf optimization algorithms, and cuckoo search algorithms, to meet different operational requirements in various real-world situations.

(3) Application of path smoothing algorithms, including the cubic spline interpolation method and the Bézier curve algorithm, to ensure the feasibility of paths in practical flight operations.

**Keywords:** UAV, Drone, Flight path planning, Planning algorithm, Optimization problem

# 目录

摘要	I
Abstract	II
1 绪论	1
2 航迹规划	2
2.1 航迹规划的概念	2
2.2 航迹规划的实质	2
2.3 航迹规划的一般步骤	2
3 环境空间建模/航迹图图形化或栅格化	3
3.1 环境建模的概念	3
3.2 栅格法环境建模	3
3.2.1 栅格法的概念	3
3.2.2 实例分析与示例伪代码	3
3.3 图形法环境建模	6
3.3.1 图形法的概念	6
3.3.2 图形法的使用	6
3.3.3 实例分析与示例伪代码	6
4 航迹规划算法与实现	8
4.1 传统基本航迹规划算法	8
4.1.1 概论	8
4.1.2 模拟算法	8
4.1.3 Dijkstra 单源最短路算法	10
4.1.4 IDA* 搜索算法	11
4.2 现代智能航迹规划算法	12
4.2.1 遗传算法	12
4.2.2 蚁群算法 (ACA)	12
4.2.3 改进灰狼算法 (IGWO)	12

4.2.4	布谷鸟搜索算法 (CS)	12
<b>5</b>	<b>航迹平滑算法</b>	<b>13</b>
5.1	三次样条插值法 (Spline 插值)	13
5.2	贝塞尔曲线法 (Bézier Curve)	13
<b>6</b>	<b>结论</b>	<b>14</b>
<b>7</b>	<b>参考文献</b>	<b>15</b>
<b>8</b>	<b>原创性声明</b>	<b>16</b>
<b>9</b>	<b>致谢</b>	<b>17</b>
<b>10</b>	<b>附录</b>	<b>18</b>

# 1 绪论

截至 2025 年末，我国公路桥梁数量已经超过 110.81 万座 [1]。桥梁正式投用后，因服役年限的增加与自然因素的损耗，使得公路桥梁自身结构受损，增加事故风险，带来安全隐患。为了保证桥梁的后续使用，保障使用者的安全，桥梁安全维护成为桥梁工作中不可或缺的一环。但由于桥梁数量与规模的大幅增加，传统的人工巡检与手动操控无人机已经难以满足桥梁的安全运维需求，亟需探索新的方式，继续提高巡检的效率、安全性和准确性 [2]。

基于航迹规划的自动化无人机控制方式结合了无人机成本低、操作性良好与自动化操作的精细、可复用性高等多方面优点，综合考虑环境、任务、安全等多方面因素，能够找到多条最优解或次优解路径，显著地提升了公路桥梁检测领域的检测质量与效率。

而在航迹规划的过程中，算法在多个步骤中，如航迹图图化或栅格化，最优路径搜索和路径平滑曲线生成中起决定性作用。国内外已对这些算法进行了多年研究，尤其是路径搜索算法，其起源可追溯至 1959 年荷兰计算机科学家狄克斯特拉提出的用于解决有权图上单源最短路径问题的算法。而近年来随着机器学习的发展，许多新型的学习算法也被逐一提出，如蚁群、灰狼、布谷鸟算法等。

就此，本文将通过算法实现与实例验证，按步骤阐述并研究讨论几类算法如何高效优质地完成航迹规划操作，保障飞行任务顺利进行，验证其在无人机飞行中的重要价值。

## 2 航迹规划

### 2.1 航迹规划的概念

无人机航迹规划是指根据预设数字地图，在给定无人机性能指标、地理环境、作战任务等约束条件的前提下，规划出一条能够回避威胁区域并且实现最优或次优的航迹轨迹。

### 2.2 航迹规划的实质

航迹规划本质是多约束条件下（飞机性能约束、时间约束、资源约束），多目标函数（生存性最大、资源消耗最小）求极值的优化问题。规划出满足任务要求、无人机性能、导航、安全性等约束的较优航路。

航迹规划是一个 NP-hard 问题, 要得到最优航迹需要极大的计算量和内存需求，意味着需要大量的时间。实际应用时往往要求能够快速响应，远远超出规定时间得到的航迹不具有实际意义。因此，保证规定时间内规划出可行且尽量接近最优航迹的方法更具现实意义。

### 2.3 航迹规划的一般步骤

总体而言，无人机航迹规划通常包括对环境空间建模、分析约束条件、依据任务目标确定代价函数、选取合适的航迹规划算法进行规划以及航线平滑化等几个步骤组成。[3]

本文将针对环境空间建模、航迹规划与航线平滑化的算法作为主要内容进行分析报告。

## 3 环境空间建模/航迹图图形化或栅格化

### 3.1 环境建模的概念

环境建模是建立一个便于计算机进行航迹规划所使用的环境模型，即将实际物理空间抽象成算法能够处理的抽象空间的过程。

### 3.2 栅格法环境建模

#### 3.2.1 栅格法的概念

栅格法是一种常用的环境建模方法，通过将连续的空间离散化为连续有限数量的栅格单元，简化复杂环境的表示。这种方法广泛应用于路径规划和机器人导航领域，尤其适用于二维环境的建模。

栅格法的核心思想是将环境划分为若干相同大小的网格，每个网格用状态值表示是否被占用。通常用 1/True 的占用状态表示障碍物，用 0/False 的空闲状态表示可通行区域。其本质为表格。

栅格法的优点在于其简单性和易于计算机存储与处理的特性。

#### 3.2.2 实例分析与示例伪代码

我们任意地绘制一张分层地形图。

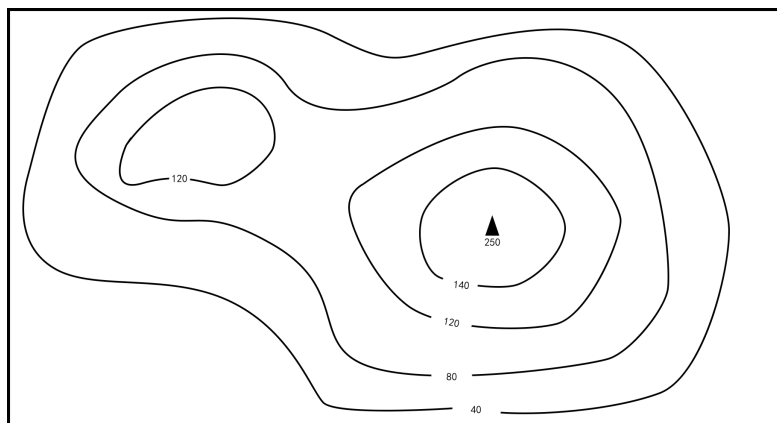


图 1: 分层地形图

首先进行栅格化。



---

**Algorithm 1:** 地图栅格化

---

**Data:** 原地图  $Map$ , 栅格大小  $X$

**Result:** 栅格  $Grid$

```
1  $Grid.Height \leftarrow \text{ceil}(Map.Length/X);$   
2  $Grid.Width \leftarrow \text{ceil}(Map.Width/X);$   
3 for  $i \leftarrow 1$  to  $Grid.Height$  do  
4   for  $j \leftarrow 1$  to  $Grid.Width$  do  
5      $Grid.creatSquare(i,j,X);$   
6      $Grid.Data[i][j].Hight \leftarrow$   
        $\text{maxOfMap}(Map,i * X,(i + 1) * X,j * X,(j + 1) * X);$   
7   end  
8 end  
9  $\text{showPictures}(Map,Grid);$ 
```

---

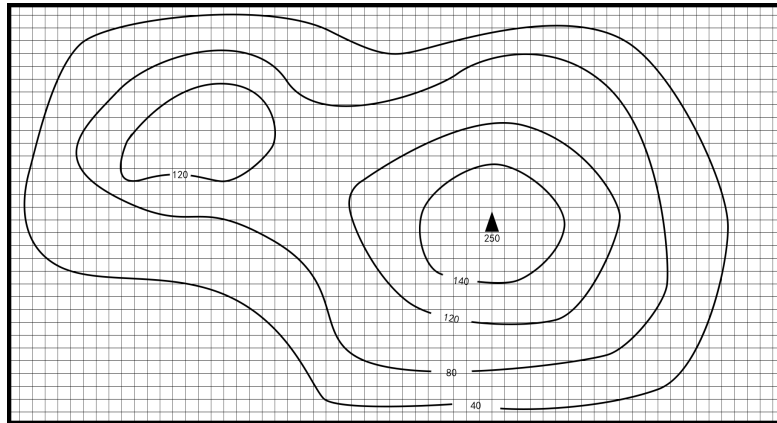


图 2: 栅格化后的地图

之后以 120m 高度作为分界, 将栅格地图进行二值化处理:

---

**Algorithm 2:** 地图栅格二值化

---

**Data:** 原地图 *Map*, 栅格 *Grid*, 分界高度 *HSet*

```
1 for  $i \leftarrow 1$  to Grid.Hight do
2   for  $j \leftarrow 1$  to Grid.Width do
3     if Grid[ $i$ ][ $j$ ].Hight > HSet then
4       Grid[ $i$ ][ $j$ ].RefuseFlag  $\leftarrow$  True;
5       Grid[ $i$ ][ $j$ ].Color  $\leftarrow$  0x46831a;
6       Grid[ $i$ ][ $j$ ].Alpha  $\leftarrow$  48;
7     end
8     else
9       Grid[ $i$ ][ $j$ ].RefuseFlag  $\leftarrow$  False;
10      Grid[ $i$ ][ $j$ ].Color  $\leftarrow$  0x831a1a;
11      Grid[ $i$ ][ $j$ ].Alpha  $\leftarrow$  48;
12    end
13  end
14 end
15 showPictures(Map,Grid);
```

---

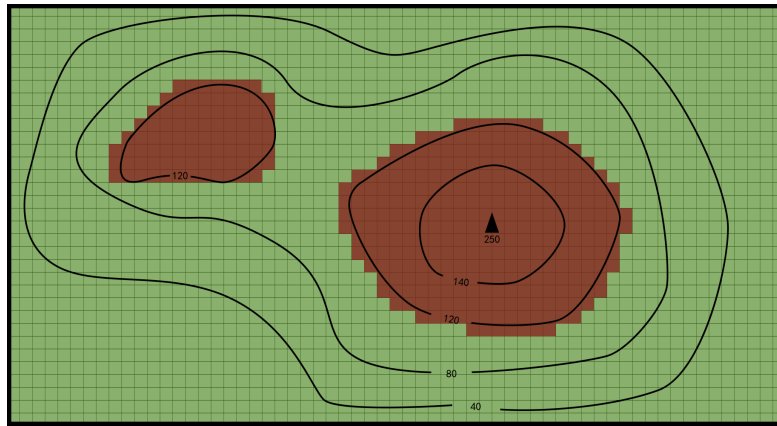


图 3: 完成栅格法环境建模的地图

这样就完成了航迹图栅格化，即环境空间建模。

### 3.3 图形法环境建模

#### 3.3.1 图形法的概念

图形法也是一种常用的环境建模方法，通过将连续的空间离散化为非连续有限数量的点/块单元，简化复杂环境的表示。这种方法广泛应用于路径规划和导航领域，尤其适用于二维且极广范围的建模。

图形法的核心思想是将环境划分为若干可通行点位，并用道路连接，每个道路用量值表示长度/耗时/代价等。其本质为无向图。

图形法的优点同样在于其简单性和易于计算机存储与处理的特性，并且可以大幅缩小问题规模。

#### 3.3.2 图形法的使用

图形法环境建模有手动标记和自动识别两种方式。

手动标记即人工标记可通行点位并设置点间可通行道路，输入计算机后自动处理。自动识别一般使用四叉树算法等进行障碍物多边形化简化，然后自动生成可通行点位并用道路连接。

本文使用手动标记法，辅以自动处理程序完成图形法环境建模。

#### 3.3.3 实例分析与示例伪代码

我们仍使用上一节栅格化时的分层地形图。

人工标记地图的可通行点位和道路：

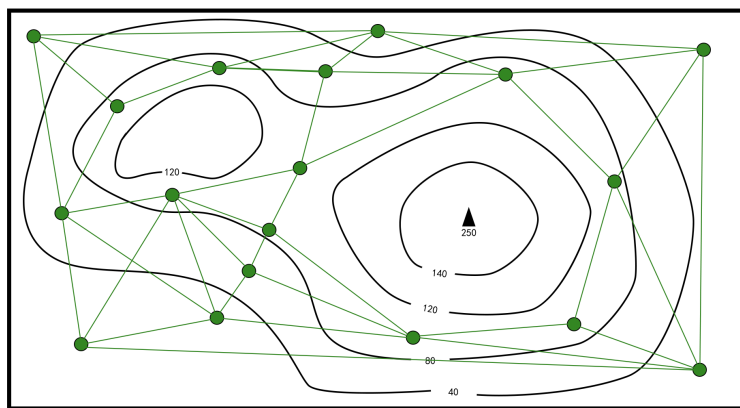


图 4: 图形化后的地形图

利用自动处理程序将分散的点与路数据合并为一张无向图：

---

**Algorithm 3:** 自动记录图形化地图

---

**Data:** 点集  $\mathbf{N}$ , 路集  $R$

**Output:** 无向图  $D$

```
1 for  $i \leftarrow 1$  to  $N.Num$  do  
2   |    $D.Dot.pushback(N.Data[i]);$   
3 end  
4 for  $i \leftarrow 1$  to  $R.Num$  do  
5   |    $D.Rd.newRoad(R.Data[i].From, R.Data[i].To, R.Data[i].Value);$   
6 end
```

---

这样就完成了航迹图图形化，即环境空间建模。

## 4 航迹规划算法与实现

### 4.1 传统基本航迹规划算法

#### 4.1.1 概论

传统基本航迹规划算法是基于搜索算法与最短路算法发展的，属于计算机算法学中的图论部分。

其特点为：基于数学分析，处理速度较快，代码复杂度较小，适合偏小规模且静态的问题。

#### 4.1.2 模拟算法

模拟算法严格上并不属于一种算法，它是处理几种静态需求问题的方法，例如矩形简单网格、多边形简单网格、矩形双重网格、环绕飞行等。

此处我们以矩形简单网格/2D 往复式航迹规划方法作为实例演示模拟算法。

以哈尔滨工业大学二校区功夫菜园鸟瞰图为例：



图 5: 功夫菜园鸟瞰图

---

**Algorithm 4:** 2D 往复式航迹规划

---

**Data:** 地图 (规划区)  $Map$ , 相邻航线间距  $D$ , 转向区外延距离  $E$

**Output:** 航迹规划点集  $F$

```
1  $N \leftarrow \text{ceil}(Map.Width/D);$ 
2 for  $i \leftarrow 0$  to  $N - 1$  do
3   if  $i \% 2 = 0$  then
4     if  $i \neq 0$  then
5        $F.\text{newPoint}(Map.Begin.La-E, Map.Begin.Lo+i*D);$ 
6     end
7      $F.\text{newPoint}(Map.Begin.La, Map.Begin.Lo+i*D);$ 
8      $F.\text{newPoint}(Map.End.La, Map.Begin.Lo+i*D);$ 
9     if  $i \neq N - 1$  then
10       $F.\text{newPoint}(Map.End.La+E, Map.Begin.Lo+i*D);$ 
11    end
12  end
13 else
14   if  $i \neq N - 1$  then
15      $F.\text{newPoint}(Map.End.La+E, Map.Begin.Lo+i*D);$ 
16   end
17    $F.\text{newPoint}(Map.End.La, Map.Begin.Lo+i*D);$ 
18    $F.\text{newPoint}(Map.Begin.La, Map.Begin.Lo+i*D);$ 
19   if  $i \neq 0$  then
20      $F.\text{newPoint}(Map.Begin.La-E, Map.Begin.Lo+i*D);$ 
21   end
22 end
23 end
```

---

最终生成的航迹点集按先后顺序连直线在原地图上的情况:



图 6: 功夫菜园鸟瞰图-航迹点图

#### 4.1.3 Dijkstra 单源最短路算法

Dijkstra 单源最短路算法是荷兰计算机科学家 E.W.Dijkstra 发现的一种用于求解非负权图上单个起点至单个终点的最短路径的算法。事实上，我们可以在相同时间内找到从给定源点到图中所有点的最短路径，因此这个算法有时也被称为单源最短路径算法 [4]。

对于一个有  $n$  个点， $m$  条路的非负权图，其朴素实现方法的时间复杂度为  $\mathcal{O}(n^2)$ 。若使用其他数据结构处理该问题，可以进一步降低时间复杂度。如基于斐波那契堆的优化可做到  $\mathcal{O}(n \log n + m)$ ，使用优先队列或线段树维护的做法可以达到  $\mathcal{O}(m \log n)$ 。而目前理论最优实现为 2025 年 6 月 15 日，由清华大学交叉信息院段然研究团队发现的针对无向图的，不依赖于排序算法，而是融合 Dijkstra 与 Bellman-Ford 两种最短路算法的实现方式。其时间复杂度可缩小至  $\mathcal{O}(m \log^{2/3} n)$  [5]。此时要根据图的疏密程度提前手动或解决过程中自动选择优化方式。

Dijkstra 算法的核心过程如下：

---

**Algorithm 5:** Dijkstra 单源最短路算法

---

**Data:** 点集  $T$ , 起点编号  $B$

**Output:** 已确定最短路长度的点集  $S$

```
1  $Dis[0 \text{ to } T.num] \leftarrow 0;$ 
2  $Dis[B] \leftarrow 0;$ 
3 while  $!T.empty()$  do
4    $Next \leftarrow$  The point in  $T$  who have the confirmed shortest path.;
5    $S.pushback(Next);$ 
6   for  $i \leftarrow 0$  to  $Next.RdNum$  do
7      $Dis[Next.RoadData[i].To] \leftarrow \min(Dis[Next.RoadData[i].To]$ 
8        $, Dis[Next] + Next.RoadData[i].Length)$ 
9   end
10 end
```

---

#### 4.1.4 IDA\* 搜索算法

IDA\* 是一种混合了迭代加深算法与 A\* 的搜索算法。其中迭代加深算法是一种每次限制搜索深度 (剪枝) 的深度优先搜索 (DFS) 算法, 而 A\* 搜索算法是一种在带权有向图上找到单个起点与终点之间的最短路径的改进广度优先搜索 (BFS) 算法。

由于 IDA\* 使用了动态更新的剪枝阈值  $C$  和估价函数  $h(x)$ , 所以其没有严格数学证明下的  $\mathcal{O}$  时间复杂度。

IDA\* 的实现样例 (包含主搜索函数 (A\*,BFS) 与副搜索函数 (迭代加深,DFS)):



---

**Algorithm 6:** IDA\* 搜索算法

---

**Data:** 估价函数  $h(X)$ , 起点编号  $B$

**Output:** 各点 ? 及其最短路长度 ?

```
1  $Dis[0 \text{ to } T.num] \leftarrow 0;$ 
2  $Dis[B] \leftarrow 0;$ 
3 while  $!T.empty()$  do
4    $Next \leftarrow$  The point in  $T$  who have the confirmed shortest path.;
5    $S.pushback(Next);$ 
6   for  $i \leftarrow 0$  to  $Next.RdNum$  do
7      $Dis[Next.RoadData[i].To] \leftarrow \min(Dis[Next.RoadData[i].To]$ 
       $, Dis[Next] + Next.RoadData[i].Length)$ 
8   end
9 end
```

---

## 4.2 现代智能航迹规划算法

### 4.2.1 遗传算法

### 4.2.2 蚁群算法 (ACA)

### 4.2.3 改进灰狼算法 (IGWO)

### 4.2.4 布谷鸟搜索算法 (CS)

## 5 航迹平滑算法

### 5.1 三次样条插值法 (Spline 插值)

### 5.2 贝塞尔曲线法 (Bézier Curve)

## 6 结论

## 7 参考文献

- [1] 中华人民共和国交通运输部.2024 年交通运输行业发展统计公报 [R/OL].(2025-06-12)[2025-10-04].[https://xxgk.mot.gov.cn/2020/jigou/zhghs/202506/t20250610\\_4170228.html](https://xxgk.mot.gov.cn/2020/jigou/zhghs/202506/t20250610_4170228.html).
- [2] 欧林联. 无人机自动化巡检在城市桥梁运维中的探索和应用 [J]. 福建建设科技,2025,(05):97-100.DOI:CNKI:SUN:FJJK.0.2025-05-024.
- [3] 王硕, 李洋, 赵蕴龙, 等. 无人机航迹规划算法综述 [J/OL]. 哈尔滨工程大学学报,1-14[2025-10-04].<https://link.cnki.net/urlid/23.1390.U.20250616.1544.003>.
- [4] Javaid,Muhammad Adeel.Understanding Dijkstra’s Algorithm[J/OL].(2013-04-10)[2025-10-18].<http://dx.doi.org/10.2139/ssrn.2340905>.
- [5] Ran Duan, Jiayi Mao, Xiao Mao, Xinkai Shu, and Longhui Yin. 2025.Breaking the Sorting Barrier for Directed Single-Source Shortest Paths[R/OL].(2025-06-15)[2025-10-18] In Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC ’25). Association for Computing Machinery, New York, NY, USA, 36 - 44. <https://doi.org/10.1145/3717823.3718179>

## 8 原创性声明

## 9 致谢

## 10 附录