

电子科技大学  
UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

# 硕士学位论文

MASTER THESIS



论文题目 动态环境下无人机主动避障技术研究

学科专业 信息与通信工程

学 号 202021010234

作者姓名 李智

指导教师 骆春波 教授

学 院 信息与通信工程学院

分类号 V249.1 密级 公开

UDC 注 1 629.05

# 学 位 论 文

## 动态环境下无人机主动避障技术研究

(题名和副题名)

李智

(作者姓名)

指导教师

骆春波 教授

电子科技大学 成都

(姓名、职称、单位名称)

申请学位级别 硕士 学科专业 信息与通信工程

提交论文日期 2023 年 5 月 29 日 论文答辩日期 2023 年 6 月 5 日

学位授予单位和日期 电子科技大学 2023 年 6 月

答辩委员会主席 李晓峰

评阅人 杨拥军 周亮

注 1: 注明《国际十进分类法 UDC》的类号。

# **Research on Active Obstacle Avoidance Technology for UAVs in Dynamic Environments**

A Master Thesis Submitted to  
University of Electronic Science and Technology of China

Discipline: **Information and Communication Engineering**

Student ID: **202021010234**

Author: **Li Zhi**

Supervisor: **Prof. Luo Chunbo**

School: **School of Information and Communication  
Engineering**

## 摘要

无人机在民用和军事场景显示出了良好的应用潜力。这些现实场景往往是动态的，存在各种不规则的障碍物。为了保证飞行安全，无人机需要具有对动态环境中的障碍物进行有效躲避的能力。本论文从运动规划层面针对动态环境中障碍物的躲避问题进行研究，运用运筹学相关理论，在路径搜索、飞行走廊生成、轨迹优化与紧急避障规划等几个重要方面进行分析与研究。所做的主要工作如下：

1. 在路径规划方面，针对主流路径搜索算法未利用已知障碍物信息导致搜索时间较长的问题，本论文提出一种快速路径搜索算法。该算法首先使用一个自适应策略获取当前地图的最佳膨胀系数，然后以该最佳膨胀系数来执行改进 Hybrid A\* 算法。本论文重点在启发式函数、遮挡惩罚项和自适应策略三个方面进行了研究创新。启发式函数能够利用已知障碍物信息并引入膨胀系数来估计真实代价；遮挡惩罚项使用随距离变化的惩罚函数来跟踪运动目标；自适应策略利用若干次迭代找到最佳膨胀系数，优化搜索时间。实验表明本论文提出的快速路径搜索算法能够平衡路径长度与运动目标观测数目的要求，有利于在复杂场景中快速规划无人机的最优路径，大幅缩短路径搜索时间。

2. 针对动态环境中在全局路径基础上规划的轨迹因为未包含避障约束的原因具有与障碍物发生碰撞的概率较高的问题，本论文提出一种基于端点任意直线栅格计算方法的飞行走廊生成算法。该算法首先计算初始飞行走廊，然后利用端点任意直线的栅格计算方法来计算走廊边界，最后在边界内搜索障碍物栅格并更新飞行走廊。实验结果表明该算法解决了 Bresenham 方法忽略栅格尺寸导致的直线栅格计算不准确的问题，能准确计算飞行走廊栅格位置与数目，大幅降低走廊生成时间。

3. 在轨迹规划方面，针对现有避障建模方法引入非凸约束和较多辅助变量导致轨迹求解时间长、难度大的问题，本论文提出一种动态障碍物避障建模与轨迹优化算法。根据动态障碍物运动轨迹在飞行走廊内的位置将其分为三类，然后基于区间边界约束与瞬时避障约束建模最优轨迹为一个凸二次规划问题并完成求解。仿真实验结果验证了算法在代表性动态场景下均能发挥良好的避障效果。

4. 针对现有紧急避障算法存在抖动与陷入局部极小值的问题，本论文提出一种紧急避障规划算法。首先建模得到障碍物平面方程，然后设计平滑避障轨迹，以实现障碍物进行提前规避。实验结果表明该算法能够使无人机在靠近障碍物时实现平稳快速远离，进一步提高了无人机在动态场景下的避障能力。

**关键词：**无人机，主动避障，路径规划，轨迹规划，紧急避障

## ABSTRACT

Unmanned Aerial Vehicles (UAVs) have demonstrated their potential in various military and civilian scenarios. Real-world situations often involve dynamic and irregular obstacles, necessitating UAVs to avoid obstacles effectively to ensure safe flight. This thesis focuses on motion planning for obstacle avoidance in dynamic environments and analyzes several critical aspects, including path search, flight corridor generation, trajectory optimization, and emergency obstacle avoidance planning, to develop a motion planning system for obstacle avoidance. The thesis's main contributions are summarized as follows:

1. A fast path search algorithm is proposed to address the problem of path search algorithms not fully utilizing known obstacle information, leading to long search time. The algorithm employs an adaptive strategy to determine the optimal inflation factor of the current map and utilizes this optimal inflation factor to perform an improved Hybrid A\* algorithm. Three research innovations, namely heuristic functions, occlusion penalty terms, and adaptive strategies, are emphasized. Experiments reveal that the proposed algorithm can balance path length optimality and motion target observation, effectively plan optimal paths for UAVs in complex scenarios, and reduce path search time.

2. A flight corridor generation algorithm based on the endpoint arbitrary straight-line grid calculation method is proposed to tackle the problem of inaccurate straight-line grid calculation and long generation time of existing flight corridor algorithms. The algorithm first calculates the initial flight corridor, determines the corridor boundary using the line with arbitrary endpoints grid calculation method, searches for obstacle grids within the boundary, and updates the flight corridor. The proposed algorithm significantly reduces the flight corridor generation time.

3. A dynamic obstacle avoidance modeling and trajectory optimization algorithm is proposed to address the problem of long and difficult trajectory solutions in existing obstacle avoidance modeling methods due to the introduction of non-convex constraints and more auxiliary variables. The algorithm classifies dynamic obstacle trajectories into three categories based on their positions in the flight corridor and models the optimal trajectory as a convex quadratic programming problem based on interval boundary constraints and instantaneous obstacle avoidance constraints. Simulation experimental results demonstrate

that the algorithm performs well in all representative dynamic scenarios.

4. An emergency obstacle avoidance planning algorithm is proposed to address the problem of jitter and local minima in current emergency obstacle avoidance algorithms. The algorithm models the obstacle wall equation and designs a smooth obstacle avoidance trajectory to achieve advanced avoidance of the obstacle. Experimental results confirm that the algorithm can make the UAV move away smoothly and quickly when it is close to the obstacle, thereby enhancing the UAV's obstacle avoidance capability in dynamic scenarios.

**Keywords:** UAV, Active obstacle avoidance, Path planning, Trajectory planning, Emergency obstacle avoidance

# 目 录

第一章 绪 论 .....	1
1.1 论文研究背景与意义 .....	1
1.2 国内外研究现状 .....	2
1.2.1 路径规划算法研究现状 .....	3
1.2.2 飞行走廊生成算法研究现状 .....	4
1.2.3 避障轨迹优化算法研究现状 .....	5
1.2.4 紧急避障算法研究现状 .....	5
1.3 研究内容与创新 .....	6
1.4 论文组织结构 .....	8
第二章 动态环境下快速路径搜索算法 .....	9
2.1 引言 .....	9
2.2 动态环境下快速路径搜索算法 .....	9
2.2.1 算法整体流程 .....	9
2.2.2 基于已知障碍物信息的启发式函数 .....	10
2.2.3 随距离变化的遮挡惩罚项 .....	12
2.2.4 寻找最佳膨胀系数的自适应策略 .....	13
2.3 仿真实验与结果分析 .....	15
2.3.1 仿真实验条件设置 .....	15
2.3.2 算法整体效果对比试验 .....	15
2.3.3 算法各项对比试验 .....	17
2.4 本章小结 .....	21
第三章 基于端点任意直线栅格计算方法的飞行走廊生成算法 .....	22
3.1 引言 .....	22
3.2 Bresenham 画线算法原理 .....	22
3.3 基于端点任意直线栅格计算方法的飞行走廊生成算法 .....	24
3.3.1 端点任意直线的栅格计算方法 .....	24
3.3.2 飞行走廊生成算法 .....	28
3.4 仿真实验与结果分析 .....	29
3.4.1 端点任意直线的栅格计算方法的对比实验 .....	29
3.4.2 飞行走廊生成算法对比实验 .....	31



3.5 本章小结 .....	34
<b>第四章 动态环境下避障轨迹优化算法 .....</b>	<b>35</b>
4.1 引言 .....	35
4.2 动态障碍物避障建模 .....	35
4.2.1 障碍物分类方法 .....	35
4.2.2 障碍物建模方法 .....	36
4.3 避障轨迹优化算法 .....	39
4.3.1 最小体积基函数 .....	40
4.3.2 二次规划问题建模求解 .....	41
4.4 仿真实验与结果分析 .....	47
4.5 本章小结 .....	52
<b>第五章 动态环境下紧急避障规划算法 .....</b>	<b>53</b>
5.1 引言 .....	53
5.2 紧急避障规划算法设计 .....	53
5.2.1 算法整体流程 .....	54
5.2.2 单个墙面避障规划 .....	55
5.2.3 多个墙面避障规划 .....	58
5.3 仿真实验与结果分析 .....	60
5.4 本章小结 .....	62
<b>第六章 全文总结与展望 .....</b>	<b>64</b>
6.1 全文工作总结 .....	64
6.2 未来研究展望 .....	65
致 谢 .....	66
参考文献 .....	67
攻读硕士学位期间取得的成果 .....	72

## 图目录

图 1-1 无人机主流技术栈 .....	1
图 1-2 本论文研究框架.....	7
图 2-1 动态环境下快速路径搜索算法流程图 .....	10
图 2-2 改进 Hybrid A* 算法流程图 .....	10
图 2-3 启发式函数设计原理。(a) 求解 OBVP 问题得到的预测轨迹；(b) 无人 机可能的实际飞行轨迹 .....	11
图 2-4 障碍物几何特征不同的环境中的重规划真实轨迹。(a) 森林环境下的 重规划轨迹；(b) 办公室环境下的重规划轨迹 .....	13
图 2-5 不同路径搜索算法的实验结果对比，(a)(b)(c)(d) 中浅色曲线表示运 动原语，深色折线表示路径，红色圆形表示运动目标的运动路线。 (a)(e)：地图 A，快速路径搜索算法计算的路径以及与观测目标的 距离；(b)(f)：地图 A，FT 算法计算的路径以及与观测目标的距离； (c)(g)：地图 B，快速路径搜索算法计算的路径以及与观测目标的距 离；(d)(h)：地图 B，FT 算法计算的路径以及与观测目标的距离 .....	16
图 2-6 不同地图中使用不同启发式函数所得运动原语与路径，浅色曲线表 示运动原语，深色折线表示路径。(a)(c)(e)(g)：不考虑障碍物的启发 式函数的效果；(b)(d)(f)(h)：考虑障碍物的启发式函数的效果 .....	17
图 2-7 不同膨胀系数的启发式函数的效果，四行图片依次表示在地图一、二、 三、四中的试验结果。(a)(f)(k)(p)I=1；(b)(g)(l)(q)I=5；(c)(h)(m)(r)I=10； (d)(i)(n)(s)I=15；(e)(j)(o)(t)I=20 .....	19
图 2-8 不同膨胀系数下算法的运行时间.....	19
图 2-9 与常数型遮挡惩罚项的效果对比。(a) 地图二，常数型遮挡惩罚项； (b) 地图二，随距离变化的遮挡惩罚项；(c) 地图三，常数型遮挡惩罚 项；(d) 地图三，随距离变化的遮挡惩罚项.....	20
图 3-1 使用栅格中心点计算直线占据栅格的缺陷 .....	22
图 3-2 Bresenham 算法的思想。(a) 候选点确定；(b) 距离计算 .....	23
图 3-3 直线端点位置不同时的栅格选择。(a) 直线端点位置在栅格中心点； (b) 直线端点位置在栅格任意点.....	24
图 3-4 飞行走廊生成算法流程框图.....	28

图 3-5 飞行走廊生成原理 .....	29
图 3-6 $d$ 、 $d_1$ 、 $d_2$ 计算方法 .....	29
图 3-7 两种直线栅格计算方法的效果, 黑色方框表示算法计算的栅格, 实线表示待栅格化的直线, 虚线表示 Bresenham 算法实际栅格化的直线。 (a)(c): Bresenham 算法计算的栅格; (b)(d): 端点任意直线的栅格计算方法计算的栅格 .....	30
图 3-8 论文提出的飞行走廊生成算法的效果。(a) 地图一, 单个走廊效果; (b) 地图三, 单个走廊效果; (c) 地图一, 从 (3,1) 到 (9,9), 整个飞行走廊效果; (d) 地图三, 从 (9,2) 到 (1.5, 9.1), 整个飞行走廊效果 .....	31
图 3-9 两种飞行走廊生成算法的效果, 地图一: 从 (1.5, 9) 到 (9,2); 地图二, 从 (5,1) 到 (4.5, 9); 地图三, 从 (1,1) 到 (9,7); 地图四, 从 (1,1) 到 (9,9)。(a)(d)(g)(j): 本论文提出算法计算的飞行走廊; (b)(e)(h)(k): 已有的凸多面体飞行走廊生成算法计算的飞行走廊; (c)(f)(i)(l): 基于上述两种形状的飞行走廊计算的轨迹 .....	33
图 4-1 三种动态障碍物分类。(a) 第一类障碍物; (b) 第二类障碍物; (c) 第三类障碍物 .....	36
图 4-2 第一类障碍物建模处理方法 .....	36
图 4-3 第二类障碍物进一步细分 .....	37
图 4-4 第二类普通障碍物的建模处理方法 .....	37
图 4-5 多个第二类障碍物的建模处理方法 .....	38
图 4-6 $t_e = t_2$ 特殊情况建模处理方法 .....	38
图 4-7 交点位于起点 $p$ 左侧的动态障碍物建模处理方法 .....	39
图 4-8 第三类障碍物建模处理方法 .....	39
图 4-9 三种算法在地图一中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度 .....	48
图 4-10 三种算法在地图二中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度 .....	49

图 4-11 三种算法在地图三中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度 .....	50
图 4-12 三种算法在地图四中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度 .....	51
图 5-1 紧急避障规划算法整体流程.....	54
图 5-2 $M_{range}$ 的计算原理 .....	54
图 5-3 测距数据拟合墙面 .....	54
图 5-4 两种避障运动轨迹原理。(a) 第一种运动轨迹; (b) 第二种运动轨迹.....	56
图 5-5 两种避障轨迹的终点位置 .....	59
图 5-6 选择新墙面避障轨迹的原理示意图 .....	59
图 5-7 算法在三种简单场景下的避障效果。(a)(b)(c): 无人机在三个场景中的避障轨迹; (d)(e)(f): 无人机在三个场景中障碍物的距离 .....	60
图 5-8 紧急避障规划算法与人工势场法的实验对比效果。(a)(b)(c): 本论文提出的紧急避障规划算法效果; (d)(e)(f): 人工势场法效果 .....	61
图 5-9 两种算法在复杂环境下的避障效果。(a)(b)(c): 紧急避障规划算法效果; (d)(e)(f): 人工势场法效果 .....	62
图 5-10 两种算法在沿折线路径运动的避障效果。(a) 紧急避障规划算法的效果; (b) 人工势场法的效果.....	62

## 表目录

表 2-1	不同路径搜索算法的实验结果对比 .....	16
表 2-2	使用不同启发式函数的搜索点数与降低的搜索时间比例 .....	18
表 2-3	自适应策略找到的最佳膨胀系数与执行改进 Hybrid A* 算法的次数 ....	19
表 2-4	两种遮挡惩罚项的比较 .....	20
表 3-1	不同栅格分辨率情况下两种算法的覆盖率 .....	30
表 3-2	限定遍历范围与全局地图遍历所用时间 .....	32
表 3-3	两种飞行走廊算法的计算时间与后端轨迹计算时间 .....	34
表 4-1	两种避障轨迹规划算法的轨迹计算时间 .....	52
表 4-2	障碍物数量不同时两种算法的轨迹计算时间 .....	52

## 缩略词表

英文缩写	英文全称	中文全称
FIFO	First In First Out	先入先出
JPS	Jump Point Search	跳点搜索算法
MINVO	Minimum Volume Basis	最小体积基函数
OBVP	Optimal Boundary Value Problem	最优边界值问题
PRM	Probabilistic Roadmap Method	概率路图法
RRT	Rapidly-exploring Random Tree Method	快速拓展随机树法
SDP	Semi-Definite Programming	半正定规划
UAV	Unmanned Aerial Vehicle	无人航空载具

## 第一章 绪论

### 1.1 论文研究背景与意义

无人机全称无人航空载具 ( Unmanned Aerial Vehicle , UAV ), 是指不载人的可以自主运动的航空飞行器, 它主要包括固定翼无人机和旋翼无人机两种类型。无人机自上个世纪诞生以来, 就一直受到人们广泛关注, 无人机最初以固定翼无人机为主, 且多用于军事用途。进入 21 世纪之后, 旋翼无人机的应用变得更加广泛。伴随着控制技术、通信电子和计算机相关技术的进步, 无人机逐渐朝着小型化、自动化和智能化的方向发展 [1]。相关技术的进步与商业模式的成熟使得人们能够在许多民用场景频繁看到无人机的身影, 常见的包括航拍摄影、物流运输、电力巡检、灾难搜索救援等应用场景 [2-5]。在这些场景中, 无人机会搭载专业设备, 利用自身传感器感知周围环境, 并根据任务需求和周围障碍物来实时规划飞行路线, 在保证自身安全的同时, 顺利完成相应任务。无人机目前之所以能够在众多民用领域大放异彩, 背后靠的是一整套核心关键技术的支持。如图1-1所示, 目前智能无人机的主流技术栈包括: 感知、预测、规划和控制四大模块 [6-8]。

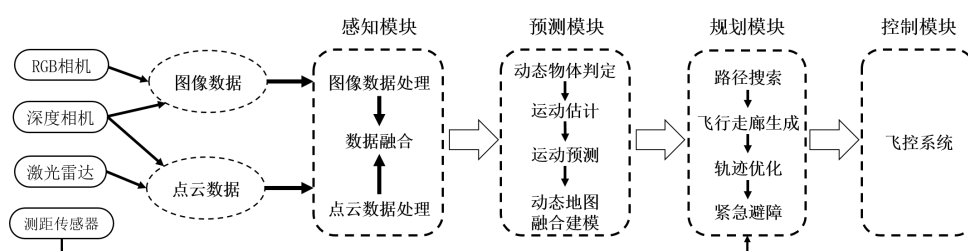


图 1-1 无人机主流技术栈

感知模块的主要功能是借助无人机机载传感器来感知周围环境, 为其它模块提供环境信息。常见机载传感器有 RGB 相机、深度相机和激光雷达等, 它们能获取周围环境的图像或点云数据, 然后交由相应的数据加工与融合算法处理, 最终得到一个表示周围环境的静态虚拟地图。

预测模块的主要功能是根据传感器数据来识别运动物体并预测其将来的运动情况。它首先要根据传感器数据来判断是否存在运动物体, 其次使用运动估计方法来估计物体的运动信息, 然后根据运动物体的历史运动信息来预测其未来运动情况, 最后将其与静态环境地图结合建模, 得到完整的飞行环境地图。

规划模块的主要功能是根据任务要求与环境信息, 生成安全可执行的平滑飞行轨迹。规划模块既可以利用感知模块建模的环境地图, 也可以直接利用传感器

测量数据。规划模块通常包括前端路径搜索、飞行走廊生成、后端轨迹优化和紧急避障这几部分。前两部分用于在全局地图中规划路径，前端路径搜索部分会得到一个全局的、粗糙的飞行路径；飞行走廊用于表示无人机飞行路径周围的自由区域。后两部分用于在局部环境中优化轨迹，轨迹优化部分会将局部路径进行优化，得到一个平滑的、可执行的运动轨迹 [9]。紧急避障作为轨迹规划补充部分，直接利用传感器数据进行快速规划，弥补前面部分无法处理的紧急情况。

控制模块的主要功能是使用各种控制算法控制无人机来跟踪上述模块生成的飞行轨迹。控制模块往往是由各种飞控系统构成，它们负责无人机各种基础飞行动作的底层控制，目前已较为成熟。

在前述应用场景中，无人机往往需要面对各种不同的任务场景与飞行环境，比如城市楼宇环境、森林环境、灾难环境等。在这样的环境中，无人机不仅要躲避各种不规则的静态障碍物，还要躲避随机出现的各种运动物体。这样的环境由于存在着不规则障碍物与随机运动的物体，因此无法用有限的规则进行描述，常被称为非结构化环境 [10]。对于静态环境，可以一次性在全局地图中执行运动规划算法来生成全局最优轨迹。相比静态环境，动态环境中的障碍物会随时间发生变化，这需要不断重复运行运动规划相关算法来对环境最新情况规划轨迹。这种方式可以很好适应动态环境，但随之而来就需要算法效率与速度更高、计算时间更短，而现有的运动规划相关算法计算时间较长，速度慢，难以在动态环境下实时运行。其次，相比于静态环境，动态环境中存在运动障碍物，不仅要考虑常规的配置空间，还要考虑时间维度。这极大地增加了运动规划相关算法的难度，导致现有算法对运动障碍物的避障效果欠佳，而且算法计算难度的提升也进一步延长了计算时间。因此，在运动规划层面开展动态环境下避障技术研究，缩短算法计算时间，提升对运动障碍物的避障效果，能够进一步增强无人机在动态环境下的生存能力，有利于无人机应用于更广泛的场景中。

## 1.2 国内外研究现状

本文从运动规划层面对动态环境中的无人机避障问题进行了研究，针对运动规划各子层面在动态环境中存在的问题，设计了一个自主高效的主动躲避障碍物的运动规划系统，为无人机飞行安全提供多层保障，实现安全快速避障，节省成本和能耗，进一步提升无人机的自主化、智能化。因此，本论文接下来对运动规划领域的路径规划、飞行走廊、避障轨迹优化和紧急避障方面的工作进行综述。



### 1.2.1 路径规划算法研究现状

很多学者对路径规划问题做了大量研究工作，这些工作大体可以分为基于采样的算法、基于搜索的算法和基于动力学可行性的算法。

基于采样的路径规划算法包括概率路图法（ Probabilistic Roadmap Method , PRM ）[11]、快速拓展随机树算法（ Rapidly-exploring Random Tree Method , RRT ）[12]、RRT\* 算法 [13,14] 等。这类算法对整个状态空间进行采样，将得到的采样点连边构图，在得到的图中进行路径规划。基于采样的路径规划算法更适合高维空间，在低维无人机地图空间中，该类算法产生最优路径所付出的采样代价较大。

基于搜索的路径规划算法更适用于低维度的地图环境，此类算法包括 Dijkstra 算法 [15]、A\* 算法 [16] 和跳点搜索（ Jump Point Search , JPS ）算法 [17] 等。Dijkstra 算法的策略是使用当前节点距出发点的最短距离来拓展节点，这种拓展方式没有倾向性，效率低下。A\* 算法利用贪婪思想，使节点代价除了距离代价外，还包含启发式代价，启发式代价常使用欧式距离、曼哈顿距离和对角距离，这克服了 Dijkstra 算法没有倾向性的问题。JPS 算法设计了 Look Ahead Rule 和 Jumping Rules 来打破 A\* 算法存在的节点对称性问题，但该算法只能用于标准栅格地图中。该类算法在低维空间中搜索速度更快，但生成路径为折线，未考虑无人机动力学可行性，后续优化过程中极有可能无法得到可执行的轨迹。

基于动力学可行性的路径规划算法包括 State Lattice Planning 算法 [18] [19]、Hybrid A\* 算法 [20] 和 Kinodynamic RRT\* 算法 [21] 等。该类算法由于考虑了无人机系统的微分特性，产生的路径更符合无人机实际运动特性，能够减小后端轨迹优化部分的计算负担。在该类算法中，Hybrid A\* 算法相比于 State Lattice Planning 算法具有任务倾向性的优势，规划效率更高；相比于 Kinodynamic RRT\* 算法具有原理简单、计算难度小的优势，因此成为无人机路径规划领域最常用的算法。在 Hybrid A\* 算法中，启发式函数的设计效果好坏会直接影响到算法的收敛速度，启发式代价  $h$  越接近于真实代价  $h^*$ ，则算法搜索数目越少，搜索时间越短。论文 [22] 首先建模得到基于运动原语的无人机优化问题，对该优化问题进行松弛得到线性二次最小时间问题，将其解析解作为启发式代价。Zhou 等人 [23] 借鉴论文 [24] 中计算运动原语的思想，考虑了连续状态条件与无人机动力学可行性，将启发式代价建模为最优边界值问题并进行求解，能够实现无人机到目标状态的实际代价的较好估计。这些启发式函数在无障碍物环境下具有很好效果，但其没有利用地图中的障碍物信息，求得的最优轨迹曲线往往会穿过障碍物，导致无人机后续实际飞行轨迹与其计算轨迹具有较大差异，出现启发式代价远离最优代价的问题，因此算法搜索速度仍较低，无法满足动态环境中快速搜索路径的需要。

### 1.2.2 飞行走廊生成算法研究现状

在无人机所处环境中，无障碍物的自由区域往往是非凸的，直接在非凸区域内寻找最优轨迹是极为困难的。通过划定飞行走廊的方式，在非凸的自由区域中切分出若干凸区域，然后在该子区域中寻找最优轨迹，能够大幅降低求解难度，实现无人机快速规划轨迹。无人机飞行走廊有凸多面体、简单几何形状等类型。

在凸多面体飞行走廊的相关研究中，论文 [25] 提出一种通过半正定规划迭代进行区域膨胀的算法，该算法在二次规划问题与半正定规划问题之间交替优化，从而得到自由空间中最大的凸多面体。论文 [26] 将该算法应用在无人机避障轨迹规划当中，降低了计算用时。受该算法的启发，论文 [27] 提出了一种凸多面体生成算法，以路径为骨架，对椭球体进行膨胀收缩得到最终凸多面体，该算法相比半正定规划算法轨迹质量相当，但速度更快。论文 [28] 提出一种凸簇膨胀算法，首先设计一个栅格簇聚类算法，使用快速凸包算法找到聚类栅格凸包，最后使用双重描述方法将凸包转化为超平面表示。将飞行走廊建模为凸多面体虽然能够扩大凸包体积，以此增加优化问题求解空间，但其耗费时间较长，而且扩大的求解空间实际上对轨迹质量提升较小。

简单几何形状表示飞行走廊能够降低计算难度，加快走廊生成速度，且能够减少优化问题约束数量，降低求解难度。在简单几何形状的飞行走廊研究中，Fei Gao 等人直接在点云地图上生成球形飞行走廊，该算法原理简单，计算速度快，但存在走廊不相交而无解的情况 [29,30]。后来，他们又提出基于栅格地图的边界盒子的飞行走廊生成算法，仍以路径点为起始栅格，在坐标轴方向进行膨胀来得到飞行走廊 [31]。针对边界盒子飞行走廊对自由区域利用不充分的问题，Jing Chen 等人设计了基于八叉树地图的算法，该算法增加了轨迹求解空间 [32]。在生成飞行走廊时，经常需要计算走廊内部以及边界所占据的栅格。最简单的方法是遍历整个栅格地图，依次判断哪些栅格在飞行走廊内部 [33]，这种方法过于粗暴且耗时很长。在计算走廊边界占据的栅格时，常用的算法是 Bresenham 算法 [34]，其能够实现对飞行走廊栅格的快速计算，但会忽略栅格尺寸，将栅格内所有点笼统使用栅格中心点代替。当地图栅格尺寸较大时，这会产生很大误差，计算的栅格无法很好地覆盖直线。这类飞行走廊生成算法都沿着坐标轴方向进行扩展，没有考虑路径走向，因此得到的凸区域对最优解的包含程度不高。除此之外，上述飞行走廊生成算法仅用于静态障碍物躲避，面对动态场景时无法辅助处理动态障碍物，具有较大的局限。

### 1.2.3 避障轨迹优化算法研究现状

避障轨迹优化算法包括静态障碍物避障轨迹规划算法与动态障碍物避障轨迹规划算法。

在静态障碍物避障轨迹规划算法研究中, Vijay Kumar 等人 [35] 发现并证明了无人机的微分平坦性质, 即无人机的状态可以仅由三维位置与偏航角以及它们的导数来表示, 这大幅降低了优化难度。论文 [36] 提出了最小化能耗的轨迹优化算法, 其将轨迹优化问题建模为二次规划问题, 使问题容易求解。最小化能耗的算法只约束了轨迹应通过的路径点, 这会产生与障碍物碰撞的轨迹, 因此, 人们提出了添加避障硬约束或软约束的算法。添加硬约束的算法有基于走廊的轨迹优化算法、贝塞尔曲线优化算法、添加稠密约束的算法和混合整型优化算法 [26,32,37,38] 等类型。添加硬约束的轨迹优化算法是一种精确的轨迹规划算法, 但现实中无人机感知的信息存在噪声, 导致该算法得到的轨迹可能无法执行。F Gao 等人 [39] 利用障碍物的梯度信息作为优化函数的软约束, 即将轨迹与障碍物的距离作为惩罚项, 使用数值优化方法来得到最优轨迹。

在动态障碍物避障轨迹规划算法研究中, 文章 [40] 在预测得到运动障碍物的运动轨迹之后, 在轨迹优化部分引入了动态物体避障约束, 建模得到非凸的二次约束二次规划问题。Wang 等人 [41] 在后端轨迹优化部分使用障碍物梯度信息与障碍物运动信息将运动物体避障作为惩罚项, 建模得到一个无约束问题。这类避障建模方法导致要求解非凸优化问题, 计算难度大且难以求得最优解。除了以上这些建模方法外, Vijay Kumar 等人 [42] 在建模避障约束时引入了整型变量, 将优化问题变为一个混合整型二次优化问题, 但该建模方式所求变量较多, 计算速度较慢, 不适合实时运行。可以看出, 现有轨迹规划避障算法常常建模得到形式复杂的非凸优化问题, 导致求解困难、计算速度慢, 无法满足实时躲避障碍物的要求。

### 1.2.4 紧急避障算法研究现状

无人机紧急避障算法主要用于当无人机距离障碍物较近时, 控制无人机快速躲避障碍物, 保证无人机飞行安全。紧急避障算法有几何引导法和势场函数法等, 下面分别对这两类算法进行介绍。

几何引导法主要包括碰撞锥和速度障碍物方法。碰撞锥的概念来自于论文 [43], 在二维平面上, 将无人机看作质点, 将运动障碍物看作圆形物体, 质点和圆的两个切线所形成的锥形区域就是碰撞锥, 它表示无人机可能与障碍物发生碰撞的速度区域。考虑到运动障碍物的速度, 就产生了速度障碍物算法 [44], 该算法将碰撞锥按运动障碍物的速度进行平移。为了解决速度障碍物算法的抖动问题,

相继有人提出了相对速度障碍物算法、混合相对速度障碍物算法 [45,46] 来解决抖动问题，但纯粹基于几何的方法无法根本上解决抖动问题，已有算法在复杂环境中仍然存在抖动，造成机械损耗。

势场函数法又称人工势场法，基本思想是：在目标上设计一个引力势函数，在障碍物上构造一个斥力势函数，从而在无人机的运动导航中，无人机受势能函数梯度影响，产生被目标吸引和被障碍物排斥的路径。原始的势场函数法是由 Khatib [47] 提出，但原始方法具有局部最小性与振荡的缺点。因此，后来产生了很多改进方法。虚拟力法将势场方法与确定的网格相结合来生成运动控制，计算每个每个网格产生的斥力并求和得到总斥力，该方法能够解决局部陷阱与振荡问题 [48]。谐波函数法 [49] 利用谐波函数的叠加性使用谐波函数作为势场函数，谐波函数满足拉普拉斯方程，通过使用有限微分法可以求解拉普拉斯方程，从而得到运动路径。势场函数法一般假设无人机具有较为强大的控制能力，忽略了无人机实际动力学可行性，因而无法可靠控制无人机避障，且在复杂场景中该算法极易陷入局部极小值。

### 1.3 研究内容与创新

本论文利用运筹学相关理论，针对动态环境下无人机如何主动避障的问题进行分析与研究。本文的研究基于当前无人机感知方面的良好研究进展，并根据计算机图形学相关理论知识来设计与实施仿真实验。

针对动态环境下现有运动规划算法计算时间长、避障效果差的问题，本论文在路径搜索、飞行走廊生成、避障轨迹优化与紧急避障规划等运动规划子层面进行分析并设计相应算法。论文整体研究框架如图1-2所示，本论文整体分为两大部分：全局路径规划和局部轨迹优化。全局路径规划部分主要用于在全局地图中找到全局最优路径，具体内容包括动态环境下快速路径搜索算法与飞行走廊快速生成算法。局部轨迹优化部分用于在全局路径的基础上进行局部优化，最终得到一条可执行的安全避障轨迹，其包括避障轨迹优化算法与紧急避障规划算法。本文具体研究内容和创新点如下：

(1) 本论文针对现有启发式函数未利用障碍物信息导致路径搜索时间较长的问题，提出一种快速路径搜索算法。该算法充分利用地图中的障碍物信息，实现了动态环境下对路径的高效搜索。该算法首先使用一个自适应策略来得到当前地图的最佳膨胀系数，然后以最佳膨胀系数来执行改进 Hybrid A\* 算法。该算法在启发式函数、遮挡惩罚项以及自适应策略三个方面进行了创新。在启发式函数上，利用障碍物信息，引入了膨胀系数，使其更加趋近最优代价。在遮挡惩罚项方面，

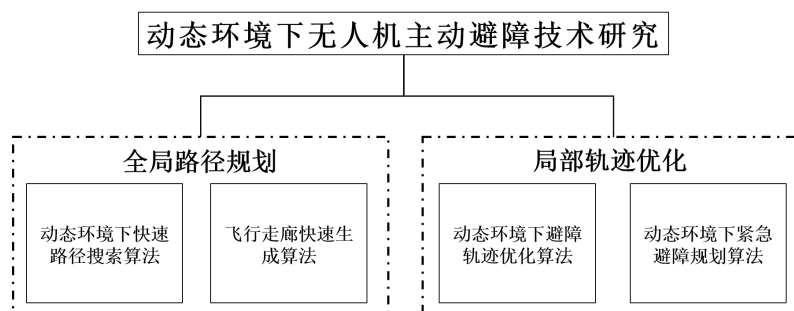


图 1-2 本论文研究框架

将其设计为随距离变化，用于及时跟踪运动目标。在自适应策略方面，该策略能够在若干次迭代后找到当前地图的最佳膨胀系数，大幅降低算法搜索时间。最后，相关仿真实验结果充分证明算法有效降低了路径搜索时间并展现了在多种环境下的适应性。

(2) 本论文针对现有飞行走廊生成算法存在的栅格计算不准确以及走廊生成时间长的问题，提出一种基于端点任意直线栅格计算方法的飞行走廊快速生成算法。该算法首先基于几何知识计算出初始走廊，然后使用端点任意直线的栅格计算方法计算边界栅格，最后使用广度优先遍历算法检查障碍物栅格，并据此更新飞行走廊。飞行走廊生成算法中使用的端点任意直线的栅格计算方法相比原始 Bresenham 算法多添加了一个验证栅格有效性的模块，从而将候选栅格从两个扩展到三个，并推导得到了两个阶段的判别项与递推公式。实验结果表明端点任意直线的栅格计算方法获得栅格更加准确，覆盖效果优于原始 Bresenham 算法；飞行走廊生成算法运算速度更快，为后续最优轨迹求解提供了有利条件。

(3) 本论文针对现有避障建模方法导致的轨迹优化模型非凸的问题，提出一种新的动态障碍物避障建模与轨迹优化算法。动态障碍物避障建模方法运用几何学理论设计，将动态障碍物分为三大类，根据不同躲避要求分别建立不同的瞬时避障约束和区间边界约束。动态障碍物避障约束建模完成后，使用 MINVO 基函数来构建无人机轨迹，推导了以控制点为优化变量的代价函数、起终点状态约束、轨迹连续性约束、瞬时边界约束、区间边界约束和动力学约束，将最小化急动度的避障轨迹优化问题建模为凸二次规划问题，并求解得到全局最优解。多组动态环境下的仿真实验结果验证了算法避障效果与鲁棒性。

(4) 本论文针对现有紧急避障算法抖动与陷入局部极小值的问题，提出一种紧急避障规划算法。首先设计基于测距模块的算法触发条件与状态切换准则；利用测距模块和无人机运动，将扫描得到的距离数据建模为静态障碍物几何边界，以此完成对环境的感知；将无人机此时运动状态作为初始条件，根据无人机与墙

面之间的相对位置，分别计算在单个墙面与多个墙面情况下的安全二阶运动轨迹，实现无人机在靠近障碍物时的平滑避障。通过在多个地图中的对比试验，验证了算法的避障效果，该算法实现了对物体的躲避远离，不会陷入障碍物构成的死角，为后续飞行提供了更高安全性和更大运动空间。

## 1.4 论文组织结构

本论文共六章，各章节内容安排如下：

第一章为绪论。首先阐述本论文研究课题的背景与意义，然后讨论了相关研究现状，其次给出本论文研究内容与创新贡献，最后说明本论文的章节安排。

第二章研究动态环境下的快速路径搜索方法。首先阐述本章研究意义与研究内容；然后给出快速路径搜索算法的整体流程；接着重点阐述三个方面的创新工作：考虑障碍物存在的启发式函数、随距离变化的遮挡惩罚项、寻找最佳膨胀系数的自适应策略；最后设计不同仿真试验分别在整体上和各个创新方面验证算法效果。

第三章研究基于端点任意直线栅格计算方法的飞行走廊生成算法。首先阐述了原始 Bresenham 画线算法的原理，然后阐述端点任意直线的地图栅格计算方法，接着对基于该栅格计算方法的飞行走廊生成算法的内容进行介绍。最后，设计对比实验分别分析验证端点任意直线的栅格计算方法与飞行走廊生成算法的效果。

第四章研究用于躲避动态环境中障碍物的轨迹优化算法。首先讨论了动态障碍物避障建模问题的重要性；然后设计出将动态障碍物分为三类进行避障建模的算法；接着建模完成以 MINVO 基函数的控制点为优化变量的优化问题；最后在 MATLAB 上实现算法代码并设计实验验证算法效果与鲁棒性。

第五章研究用于保证无人机飞行安全的紧急避障规划算法。首先阐明了紧急避障算法的重要性；然后提出紧急避障规划算法，包括算法整体流程、单个墙面与多个墙面的避障规划方法；最后设计仿真实验与人工势场避障方法的效果进行对比，验证算法有效性。

第六章为全文总结与展望。首先总结了本论文的所有工作内容，然后展望了未来值得进一步探索的方向。

## 第二章 动态环境下快速路径搜索算法

### 2.1 引言

路径搜索算法能够在无人机所处全局地图中搜索到一条全局最优路径，明确无人机整体飞行路线，降低后端轨迹求解的难度。启发式函数的设计效果好坏会直接影响到路径搜索算法的收敛速度，启发式代价  $h$  越接近于真实代价  $h^*$ ，则算法搜索数目越少，搜索时间越短。已有的基于求解最优边界值问题的启发式函数考虑了连续状态条件与无人机动力学可行性，对真实飞行代价的估计更加准确，因此搜索时间大幅减少。但该启发式函数假设地图中没有障碍物存在，求得的用来估计飞行代价的轨迹往往会穿过障碍物，这并不符合无人机实际飞行轨迹，导致估计代价远离真实代价，影响算法运行速度。除此之外，环境中也会存在需要跟踪观测的运动目标，已有启发式函数也无法有效处理这种情况。

本章提出一种动态环境下的快速路径搜索算法。首先给出快速路径搜索算法的整体流程，然后重点阐述在启发式函数、遮挡惩罚项、自适应策略上的创新。在启发式函数的设计上，本论文利用地图中的障碍物信息，引入膨胀系数的概念，其用于表示启发式代价的扩大倍数，这可以更好地估计实际代价，大幅缩短搜索时间。在遮挡惩罚项的设计上，本章使遮挡惩罚项随无人机与运动目标之间的距离变化，并将其加入启发式代价中，使得算法搜索点数相比常数型遮挡惩罚项更少。为获取具体地图的最佳膨胀系数，本章设计了一个自适应策略，其能够进一步降低搜索时间。最后，本章将提供多组仿真实验结果证明了该算法能够适应各种环境，并平衡最小代价要求和运动目标观测要求，大幅缩短路径搜索时间，减轻后端轨迹规划部分的求解压力。

### 2.2 动态环境下快速路径搜索算法

#### 2.2.1 算法整体流程

本论文提出了一种动态环境下快速路径搜索算法，该算法以 Hybrid A\* 算法为基础，充分利用地图中的障碍物信息，实现了动态环境下对路径的高效搜索。算法整体流程图如图2-1所示，该算法首先将深度相机或激光雷达数据在感知部分处理后得到的栅格地图作为输入；然后确定初始膨胀系数，并使用一个自适应策略搜索当前地图对应的最佳膨胀系数，该自适应策略会多次调用改进 Hybrid A\* 算法；在获取最佳膨胀系数之后，算法便正常执行改进 Hybrid A\* 算法。

改进 Hybrid A\* 算法整体流程如图2-2所示，首先改进 Hybrid A\* 算法会初始

化 OpenList 队列，接着判断 OpenList 中是否有待搜索的节点。若无，则结束算法；若有，进一步判断当前节点到达目标点。若到达目标点，则从目标点反向搜索得到路径；若未到达，则对当前节点进行拓展并生成运动原语，运动原语是以恒定控制量控制无人机一段时间后得到的飞行轨迹。生成运动原语之后，根据其是否与障碍物碰撞进行剪枝，然后分别计算累计代价  $g$ 、启发式代价  $h$  和遮挡惩罚项，最后更新节点总代价  $f$  并将其插入 OpenList 中。

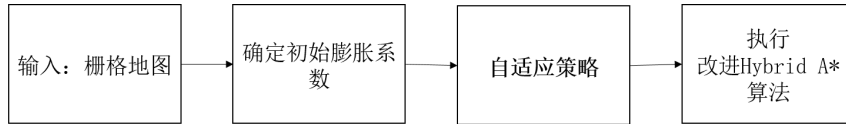


图 2-1 动态环境下快速路径搜索算法流程图

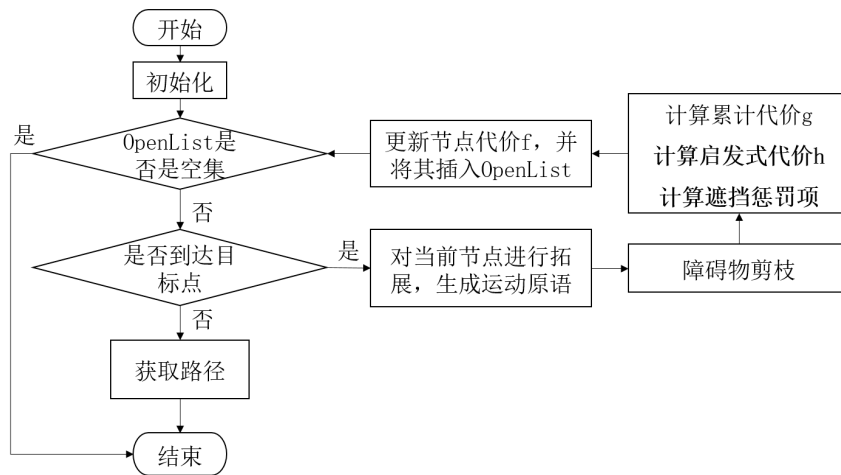


图 2-2 改进 Hybrid A\* 算法流程图

接下来重点对启发式函数、遮挡惩罚项，以及自适应策略这三方面的内容进行介绍。

### 2.2.2 基于已知障碍物信息的启发式函数

如图2-3(a)所示，已有的基于求解最优边界值问题 ( Optimal Boundary Value Problem , OBVP ) 的启发式函数 [22] 并没有考虑到环境中障碍物的存在，产生的轨迹具有很大概率会穿过各种静态障碍物，真实飞行轨迹应该类似于图2-3(b) 中绿色实线或虚线所代表的飞行轨迹。绿色实线类型的轨迹是无人机围着障碍物表面飞行同时趋向目标方向，绿色虚线类型的轨迹是无人机直接躲开前方有障碍物的区域直接飞到目标点。无论是哪一种类型的飞行轨迹，轨迹对应的代价都一定比现有的启发式函数得到的代价更大，因此在已有启发式代价  $h$  与真实代价  $h^*$  之间一定存在较大的提升空间，Hybrid A\* 算法的搜索时间上也存在较大的优化空间。



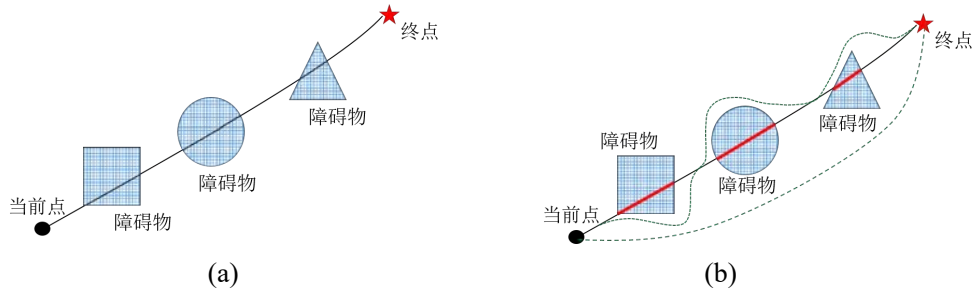


图 2-3 启发式函数设计原理。(a) 求解 OBVP 问题得到的预测轨迹；(b) 无人机可能的实际飞行轨迹

因此，本论文考虑到上述这些因素，设计出一个基于已知障碍物信息的启发式函数，在估计无人机飞行代价时将环境障碍物的影响考虑在内，进一步降低算法的搜索时间。该启发式函数首先要求解一条从当前状态到目标点的运动轨迹及其对应代价  $J$ 。本论文使用加速度  $a_k$  作为控制量，即  $u_k = a_k$ ，下标  $k$  表示哪一维度；状态量为  $s_k = (p_k, v_k)$ ，其中  $p_k$  表示位置， $v_k$  表示速度。当前状态与目标状态的位置和速度分别表示为  $s_{k0} = (p_{k0}, v_{k0})$  和  $s_{ke} = (p_{ke}, None)$ ，其中  $p_{k0}$ 、 $v_{k0}$  分别表示当前状态的位置与速度， $p_{ke}$  表示目标状态的位置。状态方程为  $\dot{s}_k = f(s_k, u_k) = (\dot{p}_k, \dot{v}_k) = (v_k, a_k)$ ，其中， $\dot{p}_k$  和  $\dot{v}_k$  分别表示位置和速度的导数。轨迹代价函数设置为：

$$J = h(s_k(T)) + \int_0^T g(s_k, u_k) dt \quad (2-1)$$

其中  $g(s_k, u_k)$  表示飞行过程中产生的代价， $h(s_k(T))$  表示是否满足最终状态约束的代价， $T$  表示最终时刻。两者具体形式为：

$$g(s_k, u_k) = a_k^2(t) + \rho, \quad h(s_k(T)) = \begin{cases} 0, p_k(T) = p_{ke} \\ \infty, p_k(T) \neq p_{ke} \end{cases} \quad (2-2)$$

$\rho$  表示时间惩罚系数，于是求解一条连接起始状态到目标状态的轨迹被建模为一个优化问题，对于该优化问题，本论文使用庞特里亚金最小化原理 [50] 直接求解得到最优轨迹为：

$$\begin{cases} p_k^*(t) = p_{k0} + v_{k0}t - \frac{1}{2}\beta_k^*t^2 + \frac{1}{6}\alpha_k^*t^3, t \in [0, T_m] \\ \alpha_k^* = \frac{6}{5T_m^3}(p_{k0} - p_{ke} + v_{k0}T_m) \\ \beta_k^* = \frac{12}{5T_m^2}(p_{k0} - p_{ke} + v_{k0}T_m) \end{cases} \quad (2-3)$$

其中， $T_m$  表示无人机最短飞行时间， $\alpha_k^*$  和  $\beta_k^*$  均表示最优轨迹函数中的多项式系数。式2-1可表示为关于  $T$  的多项式函数，因此最优轨迹对应的最优代价为  $J^* = J(T_m)$ 。

在得到运动轨迹对应飞行代价之后，需要对其进行膨胀。首先要得到原始预测轨迹中与障碍物发生碰撞的部分占整个轨迹的比例  $\alpha$ ，易知  $\alpha \in [0, 1]$ ，与障碍物发生碰撞的轨迹部分如图2-3(b) 中红色实线所示。计算  $\alpha$  的实际方法为在预测飞行轨迹所对应的时间段上进行离散采样，计算每个采样时刻对应的轨迹坐标，然后查询内存中的栅格地图矩阵来判断轨迹坐标所在的栅格是否为障碍物栅格。获取碰撞比例  $\alpha$  之后，根据碰撞比例将最优代价  $J^*$  修正为：

$$J' = (1 - \alpha)J^* + I \cdot \alpha J^* \quad (2-4)$$

其中， $(1 - \alpha)J^*$  表示未与障碍物发生碰撞的轨迹部分所对应的代价，剩余的  $\alpha J^*$  由于与障碍物的碰撞，需要重新规划这部分碰撞轨迹。重新规划得到的避障轨迹对应的代价应该与这部分碰撞轨迹所占的比例呈正相关，即碰撞比例越高，那么重规划轨迹越长，对应的代价越高，本论文设计了一个膨胀系数  $I(I \geq 1)$  用来表示这种正相关关系。通过这种方式，修正后的启发式代价能够更加接近真实的路径代价，即启发式代价  $h$  更加靠近其最优上限  $h^*$ ，从而大幅减少 Hybrid A\* 算法的搜索时间。

### 2.2.3 随距离变化的遮挡惩罚项

动态场景中可能包含无人机需要观测的运动目标，对于要观测的运动目标，本论文设计了随距离变化的遮挡惩罚项，并将其加入当前节点代价  $f$  中。无人机在飞行过程中需要保证运动目标尽可能保持在视场角范围之内，这要求运动目标尽量不被障碍物遮挡。本论文将这种要求建模为无人机质心与运动目标质心之间的连线不要穿过障碍物栅格，若连线穿过障碍物栅格，则说明观测目标被遮挡，无人机当前所处位置不够好。于是将当前位置所在栅格的启发式代价中增加一个遮挡惩罚项  $p$ ，使得改进 Hybrid A\* 算法在搜索过程尽量选择视角不会被遮挡的节点，从而最终找到一条既安全、又尽可能多观测运动目标的路径。遮挡惩罚项  $p$  的引入将当前节点代价  $f$  的计算公式变为：

$$f = g + h + p \quad (2-5)$$

其中  $g$  表示从起点到当前点的实际代价， $h$  表示从当前点到终点的预测代价， $p$  表示遮挡惩罚代价。目前已有工作 [51] 使用了一个较大的常数作为遮挡惩罚项，在搜索过程中，同一区域中不能观测到动态目标的点与能观测到动态目标的点具有相近的  $g$  和  $h$ ，但不能观测到障碍物的点还会添加遮挡惩罚项，使得最终遮挡点的代价  $f$  会大于不遮挡点的代价，从而同一区域内的不遮挡点有更大的概率被选中。由于遮挡惩罚项是固定的，且  $p \gg h$ ，所以刚开始搜索时遮挡点的  $g$  较小， $h$  相比

于  $p$  也较小；到了搜索后期，由于  $g$  的增大，导致后期搜索区域中无论是遮挡点还是非遮挡点，其  $f$  都会比前期不遮挡点的  $f$  大小相当或者更大，因此会有较大概率选到搜索前期的那些点，从而导致算法的无效搜索点数增多，降低算法收敛速度。

导致上述情况的主要原因是后期搜索点  $g$  值的增加到与前期搜索点的  $p$  值相近，算法优先选择  $f$  较小的前期搜索点。合理的修正思路应该要求前期搜索点  $p$  较大，后期搜索点  $p$  较小，从而算法在搜索后期更倾向于当前区域的点。本论文利用了当前点  $X_{current}$  到目标点  $X_{target}$  的欧氏距离设计出一个更加合理的遮挡惩罚项，具体计算公式如下所示：

$$p = \frac{A}{1 + e^{a-bd}}, \quad d = \|X_{target} - X_{current}\|_2, \quad A, a, b > 0 \quad (2-6)$$

该函数是 Sigmoid 函数的变形，当当前点距离目标点较远时，其属于前期搜索区域，因此其遮挡惩罚项应该较大；当当前点距离目标点较近时，其属于后期搜索区域，其遮挡惩罚项应该较小。系数  $A$  表示惩罚项幅度大小，系数  $a$  表示惩罚项快速变化的阈值位置，系数  $b$  表示惩罚项变化速率， $e$  表示自然底数。

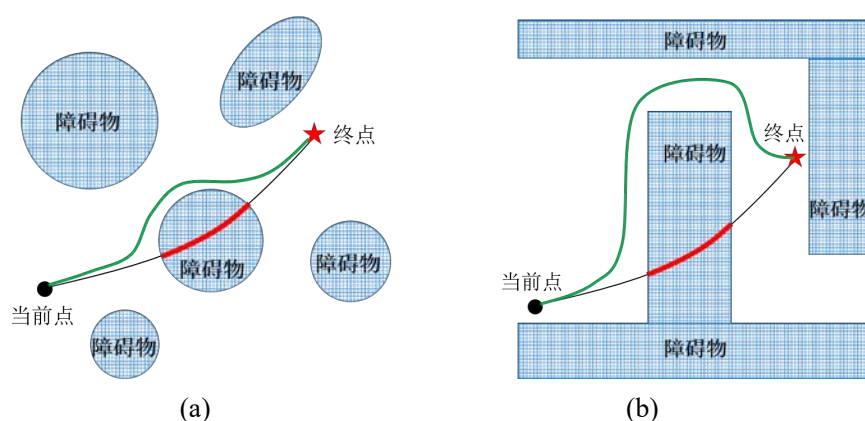


图 2-4 障碍物几何特征不同的环境中的重规划真实轨迹。(a) 森林环境下的重规划轨迹；(b) 办公室环境下的重规划轨迹

## 2.2.4 寻找最佳膨胀系数的自适应策略

无人机往往是在混乱程度均匀的环境中执行任务，例如森林环境、办公室环境等，这类环境中的障碍物几何特征和分布密度往往都是比较均匀的，不会发生较大程度的改变。如图2-4所示，对于碰撞比例相同、对应代价相同的两段轨迹，由于其所处环境中障碍物的几何特征不同，导致重规划的真实轨迹长度和相应代价也是截然不同的，即根据碰撞部分的代价放大到修正后的代价所应膨胀的倍数  $I$  也不同。因此容易得到一个结论：无人机具体飞行环境会有一个在统计上最佳的

膨胀系数，不同的环境由于几何特征不相同，其对应的最佳膨胀系数是不一样的。本论文设计出一个自适应策略来找到具体地图对应的统计上的最佳膨胀系数的一个样本值，具体算法细节如算法2-1所示。该自适应策略的内容为：

步骤一：对膨胀系数所在区间的左右边界以及中间点位置 *Left*、*Right* 和 *Mid* 进行初始化，将其初始化为同一种结构体，该结构体包含膨胀系数值 *Factor* 与根据该膨胀系数执行改进 Hybrid A\* 算法得到的搜索点数 *Count*，然后对迭代步长进行初始化。

**算法 2-1** 寻找最佳膨胀系数的自适应策略

**Data:** 栅格地图 *GridMap*  
**Result:** 最佳膨胀系数 *BestExpansionFactor*

```

1  初始化区间左右边界 Left 与 Right，中间点 Mid，步长 Step；
2  while Right.Count < Mid.Count do
3      更新 Left, Mid, Right；
4      执行改进 Hybrid A* 算法，更新 Right.Count；
5  end
6  while 区间长度大于阈值 do
7      Step  $\leftarrow$  Step/2；
8      根据 Step 更新 TempLeft.Factor 与 TempRight.Factor；
9      执行改进 Hybrid A* 算法，更新 TempLeft.Count 与 TempRight.Count；
10     if TempLeft.Count 和 TempRight.Count 均大于 Mid.Count then
11         Left  $\leftarrow$  TempLeft, Right  $\leftarrow$  TempRight；
12     else
13         if TempLeft.Count 大于 Mid.Count 且 Mid.Count 大于 Mid.Count then
14             Left  $\leftarrow$  Mid, Mid  $\leftarrow$  TempRight；
15         else
16             Right  $\leftarrow$  Mid, Mid  $\leftarrow$  TempLeft；
17         end
18     end
19 end
20 BestExpansionFactor  $\leftarrow$  Mid.Factor；

```

步骤二：寻找最佳膨胀系数所在的区间。在第一个循环中，首先判断膨胀系数右边界对应的搜索点数 *Right.Count* 是否大于中间位置的搜索点数 *Mid.Count*。若

不大于，则结束当前循环；若大于，则将左边界更新为中间位置的值、中间位置更新为右边界的值，右边界的值使用步长更新。然后执行 Hybrid A\* 算法，将得到的搜索点数作为右边界值的搜索点数，然后返回循环条件。

步骤三：缩小区间范围，逼近最佳膨胀系数。在第二个循环中，首先判断区间长度是否大于阈值。若小于阈值，则结束循环，中间节点为最佳膨胀系数；若大于阈值，则进入循环体对区间进行缩小。循环体内首先将步长 *Step* 减半，然后将左右边界分为往中间移动一个步长，得到临时左右边界的膨胀系数 *TempLeft.Factor*、*TempRight.Factor*，分别执行改进 Hybrid A\* 算法得到搜索点数 *TempLeft.Count*、*TempRight.Count*。对搜索点数与中间位置的搜索点数 *Mid.Count* 进行比较，根据不同的大小关系分别对区间左右边界进行更新，将区间缩小，更新完毕后进入下一次循环。

本论文提出的自适应策略，能够使得无人机在执行若干次改进 Hybrid A\* 算法后便能找到该地图环境对应的最佳膨胀系数，得到该值后，无人机便以此最佳膨胀系数来正常执行改进 Hybrid A\* 算法来快速寻找最优路径。

## 2.3 仿真实验与结果分析

### 2.3.1 仿真实验条件设置

本节在 MATLAB R2017a 中搭建仿真环境并设计仿真实验来对比分析所提出的快速路径搜索算法。仿真环境包括各种由基本几何形状拼接的障碍物，其使用二维栅格矩阵表示，栅格分辨率为  $l = 0.1m$ ；对于运动物体，论文将其建模为半径为  $R = 0.2m$  的圆形，并使用关于时间的三阶多项式来表示感知预测部分得到的运动物体预测轨迹。对于无人机，最大加速度为  $a_{max} = 3m/s^2$ ，最大速度为  $v_{max} = 5m/s$ 。对于每个实验，都设定无人机从 (0,0) 位置飞到 (10,10) 位置，起始点速度为 (1m/s,1m/s)，目标点速度不限制。

### 2.3.2 算法整体效果对比试验

本节首先在地图 A 与地图 B 上分别实验本章提出的快速路径搜索算法与论文 [51] 的 FT 算法的效果，在生成路径的质量、搜索时间以及是否观测到运动目标等指标上进行对比分析，其中，本论文将算法搜索的节点数目作为衡量路径搜索时间长短的指标。地图中包括静态障碍物与运动目标，静态障碍物使用黑色图形表示，对于要观测的运动目标，本实验假设其做匀速直线运动，使用圆形形状进行表示，并打印出其在无人机整个飞行过程中的运动路线。

图2-5(a)(b)(c)(d) 展示了两个算法生成的路径与算法搜索过程中计算的运动原

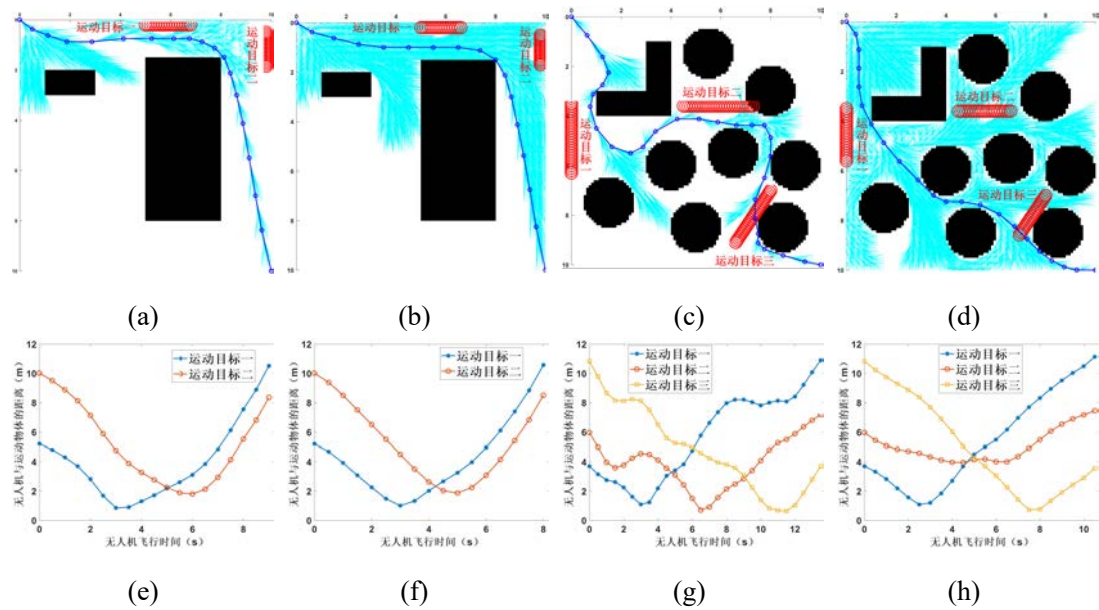


图 2-5 不同路径搜索算法的实验结果对比，(a)(b)(c)(d) 中浅色曲线表示运动原语，深色折线表示路径，红色圆形表示运动目标的运动路线。(a)(e)：地图 A，快速路径搜索算法计算的路径以及与观测目标的距离；(b)(f)：地图 A，FT 算法计算的路径以及与观测目标的距离；(c)(g)：地图 B，快速路径搜索算法计算的路径以及与观测目标的距离；(d)(h)：地图 B，FT 算法计算的路径以及与观测目标的距离

表 2-1 不同路径搜索算法的实验结果对比

	地图 A		地图 B	
	快速路径 搜索算法	FT 算法 [51]	快速路径 规划算法	FT 算法 [51]
搜索点数	403	1488	285	3665
是否观测到所有运动目标	是	是	是	否
（最佳）膨胀系数	7	1	11	1
自适应策略迭代次数	12	无	14	无

语，浅蓝色曲线表示算法生成的运动原语，深蓝色折线表示算法生成的最优路径。图2-5(e)(f)(g)(h) 表示的无人机沿生成路径飞行时与各运动目标的距离，当距离较近时，可以视为无人机能够有效观测运动目标。可以看出，本章提出的快速路径搜索算法计算的运动原语更少，且在两个地图中均生成了能够观测到每个运动目标的路径；而 FT 算法计算的运动原语数目更多，且只能在简单场景下生成符合要



求的路径，在复杂环境中无法产生观测运动目标的可靠有效路径。

表2-1详细展示了两个算法在各指标上的表现。本论文提出的算法相比 FT 算法，在两个地图中的搜索点数分别下降了 73% 和 92%，搜索点数会直接影响路径搜索时间，因此本论文提出的算法大幅度降低路径搜索时间。除此之外，可以看到，本章设计的自适应策略能够经过十余次迭代找到具体地图的最佳膨胀系数。

### 2.3.3 算法各项对比试验

为充分验证算法在各种复杂场景中的效果，本节将上一节实验的两个地图（地图 A、B）扩充到四个地图（地图一、二、三、四），其中，地图一、三分别代表原来的地图 A、B。

#### 2.3.3.1 启发式函数对比实验

本小节依次在地图一、二、三、四分别运行两次改进 Hybrid A\* 算法，第一次运行时使用基于求解 OBVP 问题的不考虑障碍物的启发式函数 [23]，第二次运行时使用本章设计的考虑障碍物存在的启发式函数，其中膨胀系数设置为 5。这四个实验用于分析两种不同启发式函数对算法搜索时间的影响。

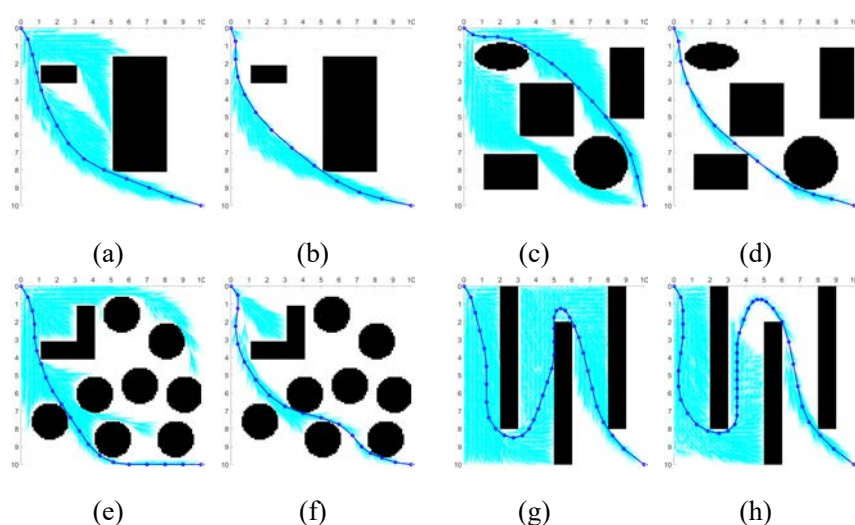


图 2-6 不同地图中使用不同启发式函数所得运动原语与路径，浅色曲线表示运动原语，深色折线表示路径。(a)(c)(e)(g)：不考虑障碍物的启发式函数的效果；(b)(d)(f)(h)：考虑障碍物的启发式函数的效果

图2-6依次表示四个实验的实验结果，从图中看出，无论在哪个地图中，使用考虑障碍物存在的启发式函数所生成的运动原语数目远少于使用不考虑障碍物的启发式函数的运动原语数目，且最终都找到了相同的最优路径。

表2-2表示四个实验中算法的搜索点数，相比于不考虑障碍物的启发式函数，

表 2-2 使用不同启发式函数的搜索点数与降低的搜索时间比例

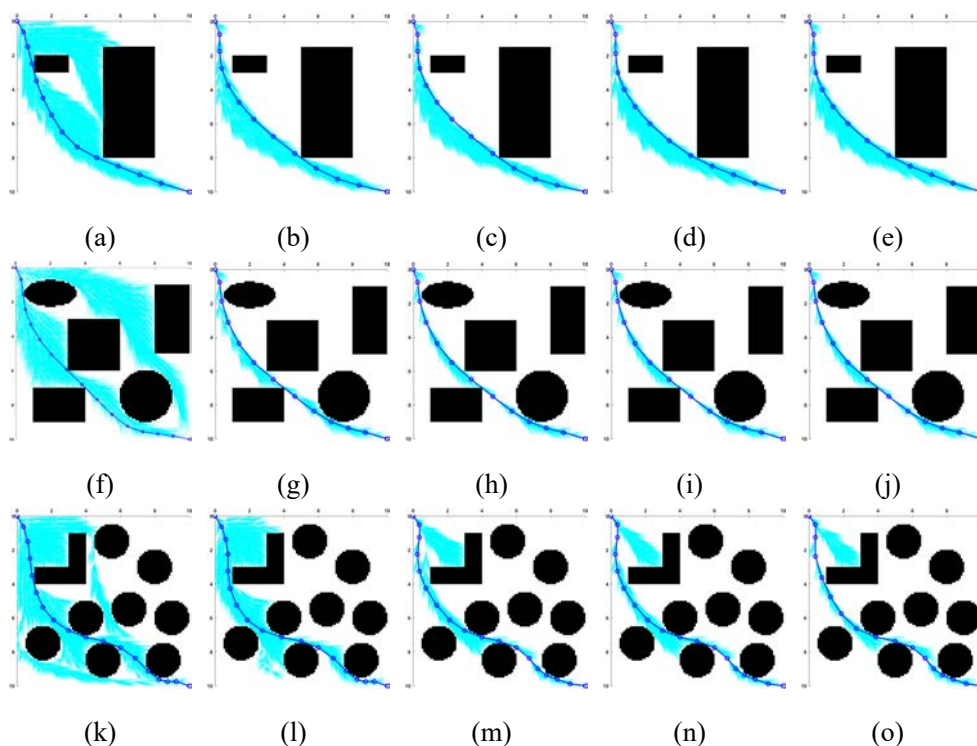
	地图一	地图二	地图三	地图四
不考虑障碍物 [23]	1128	1613	1458	3555
考虑障碍物	190	40	284	1982
搜索时间缩短	83.15%	97.52%	80.52%	79.36%

本论文提出的启发式函数对算法搜索时间的缩短基本都在 80% 以上。从以上这些实验结果，可以看出本论文提出的考虑障碍物存在的启发式函数具有非常好的效果，能够大幅降低搜索时间。

### 2.3.3.2 膨胀系数对算法速度提升的实验

本论文设计的考虑障碍物存在的启发式函数引入了膨胀系数的概念，本节在四个地图中分别实验多个膨胀系数，来观察不同膨胀系数对算法运行时间的影响。

从图2-7中可以看到，随着膨胀系数增大，运动原语的生成数量先逐渐减少，然后保持在某一量级上浮动或继续缓慢下降。从图2-8可以看出，不同膨胀系数对算法效率的提升效果不同，膨胀系数越大，算法搜索时间越短；但又有上限存在，后期随着膨胀系数增大，算法搜索时间缩短幅度变得不太明显。进一步发现，不同地图下的算法效率上限的对应膨胀系数也是不同的，该膨胀系数便是最佳膨胀系数，从图中可以目测四个地图的最佳膨胀系数分别在 9、5、11 和 7 附近。





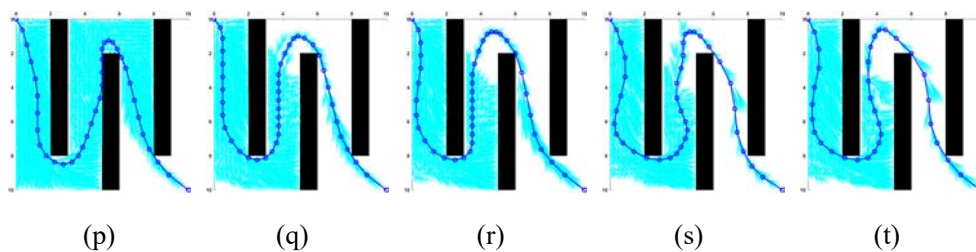


图 2-7 不同膨胀系数的启发式函数的效果，四行图片依次表示在地图一、二、三、四中的试验结果。(a)(f)(k)(p)I=1; (b)(g)(l)(q)I=5; (c)(h)(m)(r)I=10; (d)(i)(n)(s)I=15; (e)(j)(o)(t)I=20

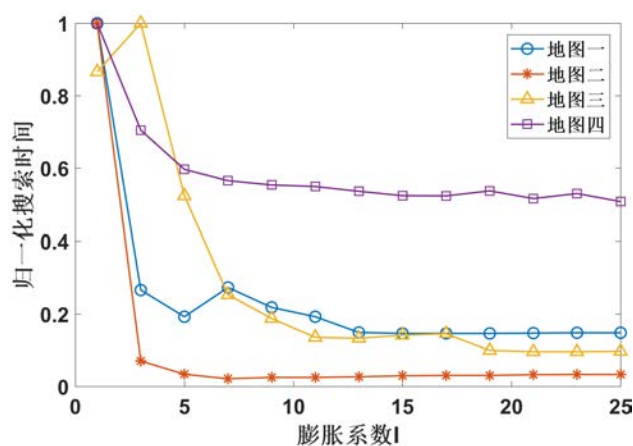


图 2-8 不同膨胀系数下算法的运行时间

### 2.3.3.3 自适应策略验证实验

表 2-3 自适应策略找到的最佳膨胀系数与执行改进 Hybrid A\* 算法的次数

	地图一	地图二	地图三	地图四
膨胀系数试验找到的最佳膨胀系数	9	5	11	7
自适应策略找到的最佳膨胀系数	7	6	11	8
自适应策略执行改进 Hybrid A* 算法的次数	11	9	13	11

本节分别在四个地图上运行自适应策略来检验该策略的有效性与准确性，实验结果如表2-3所示。从表中可以看出，自适应策略找到的最佳膨胀系数均在上一节所述的值附近，因此可以确认该策略具有较好的效果，且能够在执行十余次改

进 Hybrid A\* 算法后找到该地图环境对应的最佳膨胀系数。

### 2.3.3.4 遮挡惩罚项对比实验

本小节设计实验来对比分析已有的常数型遮挡惩罚项 [51] 与本章提出的随距离变化的遮挡惩罚项的效果。为节省篇幅，同时由于地图一与地图四的障碍物分布会造成无人机在大部分飞行时间都可以观察到运动目标，因此本小节在对比效果较好的地图二和地图三中开展试验。

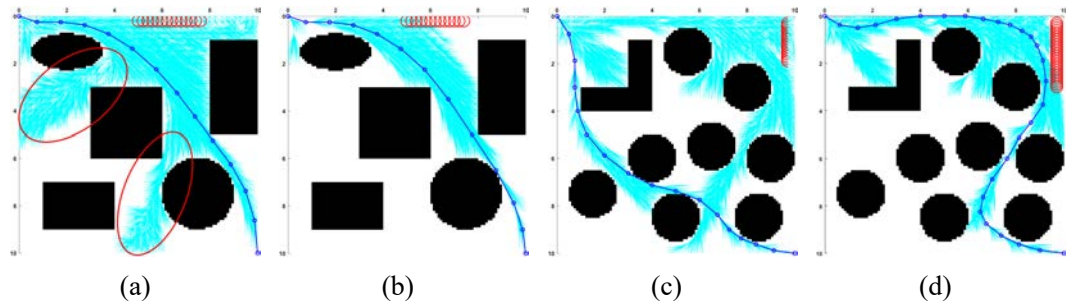


图 2-9 与常数型遮挡惩罚项的效果对比。(a) 地图二，常数型遮挡惩罚项；(b) 地图二，随距离变化的遮挡惩罚项；(c) 地图三，常数型遮挡惩罚项；(d) 地图三，随距离变化的遮挡惩罚项

实验结果如图2-9所示，从整体结果可以看到，遮挡惩罚项的加入使得搜索算法不再去寻找长度最短的路径，而是去寻找能够最大程度观测运动物体的路径。从实验结果图2-9(a)(b) 可以看出，常数型的遮挡惩罚项虽然也能够找到观测运动目标的最大路径，但其在搜索过程中搜索了许多不必要的节点，降低了算法的计算速度，如表2-4所示，其计算时间是本文提出的遮挡惩罚项的 3.1 倍。原因在于

表 2-4 两种遮挡惩罚项的比较

	地图二		地图三	
	常数型 遮挡惩罚项 [51]	随距离变化的 遮挡惩罚项	常数型 遮挡惩罚项 [51]	随距离变化的 遮挡惩罚项
搜索点数	2057	661	1391	759
轨迹飞行时间	6.5s	6s	8s	13.5s
是否观测到运动物体	是	是	否	是

常数型的遮挡惩罚项会导致后期搜索时有较大概率选择前期的搜索点，图2-9(a) 中的红框部分便是多被搜索的点。常数型遮挡惩罚项的这个缺陷严重情况下会导致

搜索算法最终找到的路径无法对运动目标进行观测，图2-9(c)(d)便展示了这种情况，由于后期搜索点的代价与前期搜索点的代价相近，导致算法在快要搜索到目标时却反过来从起点附近重新搜索，最终找到距离较短、但无法观测运动物体的路径。常数型遮挡惩罚项缺陷的具体理论分析细节见第 2.2.3 节，在此不再赘述。

## 2.4 本章小结

本章首先指出目前路径搜索算法未利用地图障碍物信息的问题，因此算法的运行时间具有很大的优化空间。然后提出了一种快速路径搜索算法，并给出算法的整体计算流程。接着，重点阐述了启发式函数、遮挡惩罚项与自适应策略这三方面的创新工作。启发式函数充分利用已知障碍物信息，引入膨胀系数的概念，更好地估计了实际飞行代价；遮挡惩罚项被设计为随距离变化，能够更好地观测运动目标，平衡路径代价最优要求与目标观测要求；自适应策略能够在十余次迭代后获取地图的最佳膨胀系数。最后，对比试验检验了本章提出的快速路径搜索算法的整体效果及其在各个指标上的优秀表现。

## 第三章 基于端点任意直线栅格计算方法的飞行走廊生成算法

### 3.1 引言

无人机飞行走廊通常是指表示无人机自由飞行空间的凸区域，该区域内不包含任何静态障碍物。飞行走廊的引入将非凸自由区域切分为若干凸区域，既实现了对障碍物的避障，又降低了后端轨迹规划的难度，实现快速计算轨迹。现有飞行走廊生成算法使用复杂的步骤来获取更大的凸区域，但极大延长了算法计算时间，在动态环境下无法实现快速生成飞行走廊。

在生成飞行走廊时，需要计算一段直线对应的地图栅格，最经典的算法是 Bresenham 算法，该算法计算过程只涉及整数加法与乘法，且乘法都是乘 2 操作，在计算机二进制环境中相当于移位操作，极大降低了计算难度和画线时间，能够加快飞行走廊生成速度。但 Bresenham 算法会忽略栅格尺寸，将栅格内所有点笼统使用栅格中心点代替，如图3-1所示，当地图栅格尺寸较大时，这会产生很大误差，计算的栅格无法很好地覆盖直线。

本章使用边界盒子（Bounding Box）作为无人机飞行走廊，提出一个基于端点任意直线的栅格计算方法的飞行走廊生成算法。本章首先对 Bresenham 算法思想进行阐述，然后基于其思想研究了端点任意直线的地图栅格计算方法，推导出其两个阶段的判别项与递推公式。在该方法基础上，本章提出飞行走廊快速生成算法，该算法首先运用几何知识构建初始走廊，使用前述方法计算走廊边界栅格，然后使用广度优先遍历算法检查障碍物栅格，其能够准确划定栅格搜索范围，大幅降低计算时间。最后，本章设计仿真对比实验来验证算法效果。

### 3.2 Bresenham 画线算法原理

如图3-2所示，Bresenham 算法的思想是通过每个栅格的中心位置构造出虚拟网络线，从直线的起点栅格出发，计算直线与各垂直网络线的交点以及栅格中心

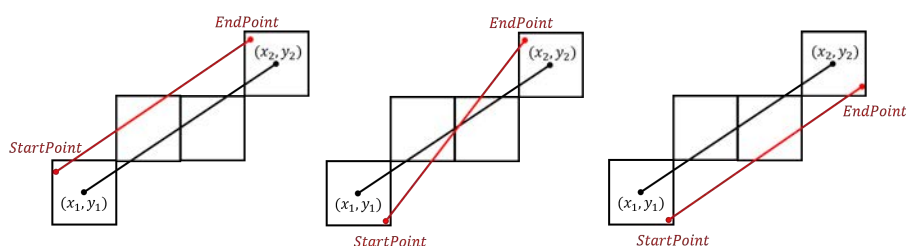


图 3-1 使用栅格中心点计算直线占据栅格的缺陷

点到交点的距离误差项，然后根据误差项的符号确定该列栅格中与此交点最近的栅格，将其作为表示直线的一个栅格。

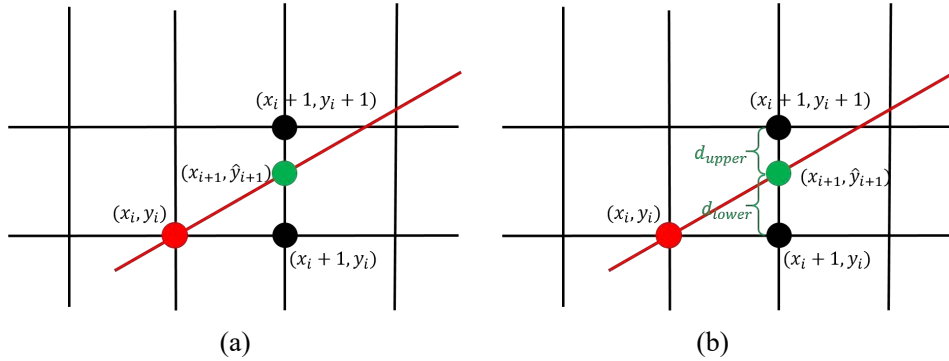


图 3-2 Bresenham 算法的思想。(a) 候选点确定；(b) 距离计算

Bresenham 算法会根据直线斜率绝对值选择步进轴，当绝对值小于 1 时，步进轴为 x 轴；绝对值大于 1 时，步进轴为 y 轴。以 x 轴为步进轴和以 y 轴为步进轴的对应算法步骤仅在符号上有所区别，本论文只对以 x 轴为步进轴的情况进行分析，步进轴为 y 轴的情况可按照相同思路得到。

设直线起点位置所在栅格的坐标为  $(x_1, y_1)$ ，终点栅格的坐标为  $(x_2, y_2)$ ，其中  $x_1, y_1, x_2, y_2$  都是整数。设直线方程为  $y = kx + b$ ，其中  $0 < k < 1$ ，斜率计算公式为  $k = \Delta y / \Delta x = (y_2 - y_1) / (x_2 - x_1)$ 。设当前所在栅格坐标为  $(x_i, y_i)$ ，则下一个栅格应该选在  $(x_i + 1, y_i)$  与  $(x_i + 1, y_i + 1)$  当中选择。

Bresenham 算法首先要计算交点坐标  $(x_{i+1}, y_{i+1})$ ，其计算公式为：

$$x_{i+1} = x_i + 1, \quad y_{i+1} = k(x_i + 1) + b \quad (3-1)$$

然后分别计算两个候选栅格中心点到交点的距离误差项  $d_{upper}$  和  $d_{lower}$  为：

$$d_{upper} = y_i + 1 - y_{i+1} = y_i + 1 - kx_{i+1} - b \quad (3-2)$$

$$d_{lower} = y_{i+1} - y_i = kx_{i+1} + b - y_i \quad (3-3)$$

接下来做差比较  $d_{upper}$  与  $d_{lower}$  的大小来判断选择哪一个点：

$$d_{lower} - d_{upper} = 2k(x_i + 1) - 2y_i + 2b - 1 \quad (3-4)$$

将式3-4两侧同乘  $\Delta x$ ，得到判别项  $p_i$ ：

$$\begin{aligned} p_i &= \Delta x(d_{lower} - d_{upper}) \\ &= 2\Delta y \cdot x_i - 2\Delta x \cdot y_i + (2b - 1)\Delta x + 2\Delta y \end{aligned} \quad (3-5)$$

在当前斜率下,  $\Delta x$  是正的, 因此可以通过判断  $p_i$  的正负来决定下一个栅格点, 这样便省去除法操作, 整个操作只有整数的加法和乘法。

进一步地, 计算当前判别项  $p_i$  和下一个判别项  $p_{i+1}$  之间的差值:

$$p_{i+1} - p_i = 2\Delta y - 2\Delta x(y_{i+1} - y_i) \quad (3-6)$$

式3-6是一个计算  $p_i$  的递推公式, 根据当前判别式  $p_i$  选择好  $y_{i+1}$  之后, 便可根据式3-6来计算出下一个栅格点的判别式  $p_{i+1}$ 。

### 3.3 基于端点任意直线栅格计算方法的飞行走廊生成算法

#### 3.3.1 端点任意直线的栅格计算方法

基于 Bresenham 画线算法的地图栅格计算方法要将直线两端点改为其所在栅格的中心点求解, 这会带来较大的误差。本论文提出一个端点任意直线的栅格计算方法, 不再将直线端点修改为栅格中心点, 使最终计算的栅格能够更好地覆盖原来的直线。

传统的 Bresenham 画线算法的直线端点是在栅格中心点, 假设直线斜率  $0 < k < 1$ , 当直线穿过栅格  $(i, j)$  时,  $i$  表示栅格所在的列数,  $j$  表示栅格所在的行数, 那么直线接下来只会在  $(i+1, j)$  或  $(i+1, j+1)$  中, 因此下一个栅格只会在其中选择, 如图3-3(a)所示。

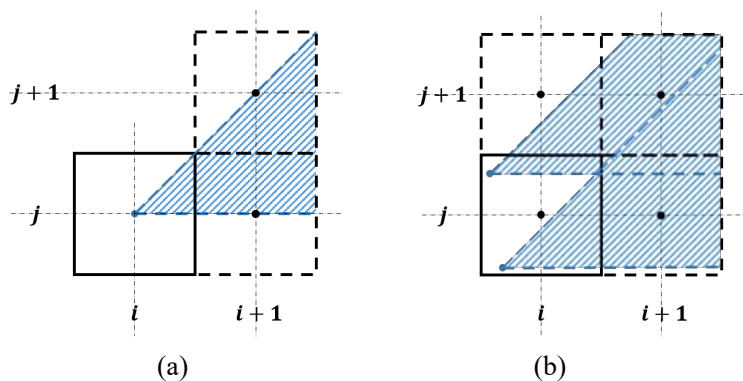


图 3-3 直线端点位置不同时的栅格选择。(a) 直线端点位置在栅格中心点;

(b) 直线端点位置在栅格任意点

如果直线端点可以选择栅格内任意点的话, 如图3-3(b)所示, 那么下一个栅格就可能在  $(i, j+1)$ ,  $(i+1, j)$ ,  $(i+1, j+1)$  这三个栅格中选择。

传统的 Bresenham 画线算法在步进轴上匀速增加, 在另一轴上非匀速增加。本论文提出的算法则要求在两个轴上都非匀速变化。根据上述思想, 本论文设计的方法首先会判断当前栅格是否有效, 若有效, 就执行正常迭代步骤; 若无效, 则

将当前栅格更新为非步进轴上的新栅格，并进入新的循环。接下来阐述该方法的原理。

假设当前栅格的索引坐标为  $(r_i, c_i)$ ，其对应的栅格中心点坐标为  $(x_{ci}, y_{ri})$ ，直线两端点坐标分别是  $(x_{start}, y_{start})$  和  $(x_{end}, y_{end})$ ，地图栅格尺寸为  $l$ 。类似地，本论文提出的算法也将直线分为斜率绝对值小于 1 的和绝对值大于 1 的两种情况进行处理。本论文仅对前一种情况进行详细讨论，后一种情况可按照相同思路进行操作。本论文使用  $p_{row}$  ( $p_{row} = \pm 1$ ) 和  $p_{col}$  ( $p_{col} = \pm 1$ ) 表示直线在  $y$  轴与  $x$  轴上的步进方向，其计算方式为：

$$\begin{cases} p_{col} = \text{sign}(x_{end} - x_{start}) \\ p_{row} = \text{sign}(y_{end} - y_{start}) \end{cases} \quad (3-7)$$

其中， $\text{sign}(x)$  是符号函数，当  $x$  大于 0 时，函数返回 1；当  $x$  小于 0 时，函数返回 -1。例如直线处于第一象限，其步进方向是  $x$  轴与  $y$  轴正方向，则  $p_{col} = 1, p_{row} = 1$ 。本论文定义直线方程为：

$$y = mx + B = \frac{p_{row} \cdot \Delta y}{p_{col} \cdot \Delta x} x + B, \quad \begin{cases} \Delta x = |x_{end} - x_{start}| \\ \Delta y = |y_{end} - y_{start}| \end{cases} \quad (3-8)$$

算法首先需要判断当前栅格是否有效，判断的思路是计算当前栅格与在非步进轴增长方向上的下一个栅格与直线在当前虚拟网格线的交点的距离，根据距离大小判断是否需要改变栅格。当前栅格中心位置为  $(x_{ci}, y_{ri})$ ，沿着  $y$  轴前进方向的下一个栅格的中心位置坐标为  $(x_{ci}, y_{ri} + p_{row}l)$ ，直线与虚拟网格线交点坐标为  $(x_{ci}, y_i)$ ，其中，

$$y_i = mx_{ci} + B \quad (3-9)$$

则两个栅格中心位置与交点距离分别为：

$$\begin{cases} d_{upper} = p_{row}(y_{ri} + p_{row}l - y_i) \\ d_{lower} = p_{row}(y_i - y_{ri}) \end{cases} \quad (3-10)$$

两者做差可得：

$$d_{lower} - d_{upper} = 2p_{col} \cdot \frac{\Delta y}{\Delta x} x_{ci} + 2B \cdot p_{row} - 2y_{ri} \cdot p_{row} - l \quad (3-11)$$

式3-11两侧同乘  $\Delta x (> 0)$  得栅格有效性判别项为：

$$\begin{aligned} \bar{p}_i &= \Delta x(d_{lower} - d_{upper}) \\ &= 2p_{col} \cdot \Delta y \cdot x_{ci} - 2p_{row} \cdot \Delta x \cdot y_{ri} + 2p_{row} \cdot B\Delta x - l\Delta x \end{aligned} \quad (3-12)$$

由于起点  $(x_{start}, y_{start})$  过直线，所以有  $y_{start} = \frac{p_{row} \cdot \Delta y}{p_{col} \cdot \Delta x} x + B$ ，于是：

$$B \cdot \Delta x = \Delta x \cdot y_{start} - p_{row} p_{col} \cdot \Delta y \cdot x_{start} \quad (3-13)$$

将式3-13代入式3-12得：

$$\bar{p}_i = 2 \begin{bmatrix} \Delta y & -\Delta x \end{bmatrix} \begin{bmatrix} p_{col} & 0 \\ 0 & p_{row} \end{bmatrix} \left\{ \begin{bmatrix} x_{ci} \\ y_{ri} \end{bmatrix} - \begin{bmatrix} x_{start} \\ y_{start} \end{bmatrix} \right\} - l \cdot \Delta x \quad (3-14)$$

设直线起点为  $(x_{start}, y_{start})$  所在栅格索引为  $(r_0, c_0)$ ，栅格中心点为  $(x_{c0}, y_{r0})$ 。则初始有效性判别项为：

$$\bar{p}_0 = 2 \begin{bmatrix} \Delta y & -\Delta x \end{bmatrix} \begin{bmatrix} p_{col} & 0 \\ 0 & p_{row} \end{bmatrix} \left\{ \begin{bmatrix} x_{c0} \\ y_{r0} \end{bmatrix} - \begin{bmatrix} x_{start} \\ y_{start} \end{bmatrix} \right\} - l \cdot \Delta x \quad (3-15)$$

根据式3-14，若  $\bar{p}_i \leq 0$ ，则  $d_{lower} \leq d_{upper}$ ，说明当前栅格比 y 轴前进方向的栅格距离交点更近，当前栅格有效；若  $\bar{p}_i < 0$ ，则  $d_{lower} > d_{upper}$ ，说明当前栅格比 y 轴前进方向的栅格距离交点更远，当前栅格无效，选择  $(x_{ci}, y_{ri} + p_{row}l)$ ，其对应栅格的索引为  $(r_i + p_{row}, c_i)$ 。

对当前栅格的有效性判断完毕后，算法进行下一个栅格的选择，其思想与 Bresenham 画线算法类似，也是判断栅格  $(r_i, c_i + p_{col})$ ， $(r_i + p_{row}, c_i + p_{col})$  和直线与虚拟网络线的交点的距离，选择距离更小的栅格。设交点坐标为  $(x_{i+1}, y_{i+1})$ ，有：

$$\begin{cases} x_{i+1} = x_i + p_{col} \cdot l \\ y_{i+1} = mx_{i+1} + B = m(x_i + p_{col} \cdot l) + B \end{cases} \quad (3-16)$$

类似地，分别计算距离为：

$$d_{upper} = p_{row}[y_{ri} + p_{row} \cdot l - m(x_{ci} + p_{col} \cdot l) - B] \quad (3-17)$$

$$d_{lower} = p_{row}[m(x_{ci} + p_{col} \cdot l) + B - y_{ri}] \quad (3-18)$$

$$d_{lower} - d_{upper} = 2m \cdot p_{row}(x_{ci} + p_{col} \cdot l) + p_{row} \cdot (2B - 2y_{ri}) - l \quad (3-19)$$

判别项  $p_i$  为：

$$p_i = p_{col} \cdot 2\Delta y \cdot x_{ci} - p_{row} \cdot 2\Delta x \cdot y_{ri} + (2B \cdot p_{row} - l)\Delta x + 2l\Delta y \quad (3-20)$$

根据上式，得到判别项初始值  $p_0$  为：

$$p_0 = 2 \begin{bmatrix} \Delta y, -\Delta x \end{bmatrix} \begin{bmatrix} p_{col} & 0 \\ 0 & p_{row} \end{bmatrix} \left\{ \begin{bmatrix} x_{c0} \\ y_{r0} \end{bmatrix} - \begin{bmatrix} x_{start} \\ y_{start} \end{bmatrix} \right\} + l \begin{bmatrix} \Delta y, -\Delta x \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad (3-21)$$



判别项做差可得：

$$p_{i+1} - p_i = 2l\Delta y + 2\Delta x \cdot p_{row}(y_{r_i} - y_{r_{i+1}}) \quad (3-22)$$

所以，如果当前判别项  $p_i \leq 0$ ，即  $d_{lower} \leq d_{upper}$ ，则选择栅格  $(r_i, c_i + p_{col})$ ，其坐标为  $(x_{ci} + p_{col} \cdot l, y_{ri})$ ，更新下一个判别项为  $p_{i+1} = p_i + 2l\Delta y$ ；若  $p_i < 0$ ，即  $d_{lower} < d_{upper}$ ，则选择栅格  $(r_i + p_{row}, c_i + p_{col})$ ，其对应的栅格中心点坐标为  $(x_{ci} + p_{col} \cdot l, y_{ri} + p_{row} \cdot l)$ ，更新下一个判别项为  $p_{i+1} = p_i + 2l\Delta y - 2l\Delta x$ 。

根据上述思路，栅格有三种更新方式，分别是只在 y 轴方向更新，只在 x 轴方向更新更新和同时在 x 轴、y 轴方向更新。在这三种更新方式中，都需要使用递推公式更新判别项  $\bar{p}_i$  和  $p_i$ ，本论文直接给出两个判别项在三种更新方式下的递推公式如下：

$$\left\{ \begin{array}{ll} \bar{p}_i(x_{ci}, y_{ri} + p_{row} \cdot l) & = \bar{p}_i(x_{ci}, y_{ri}) - 2l\Delta x \\ \bar{p}_i(x_{ci} + p_{col} \cdot l, y_{ri}) & = \bar{p}_i(x_{ci}, y_{ri}) + 2l\Delta y \\ \bar{p}_i(x_{ci} + p_{col} \cdot l, y_{ri} + p_{row} \cdot l) & = \bar{p}_i(x_{ci}, y_{ri}) + 2l(\Delta y - \Delta x) \\ p_i(x_{ci}, y_{ri} + p_{row} \cdot l) & = p_i(x_{ci}, y_{ri}) - 2l\Delta x \\ p_i(x_{ci} + p_{col} \cdot l, y_{ri}) & = p_i(x_{ci}, y_{ri}) + 2l\Delta y \\ p_i(x_{ci} + p_{col} \cdot l, y_{ri} + p_{row} \cdot l) & = p_i(x_{ci}, y_{ri}) + 2l(\Delta y - \Delta x) \end{array} \right. \quad (3-23)$$

综上所述，本论文给出端点任意直线的栅格计算方法的伪代码：

#### 算法 3-1 端点任意直线的栅格计算方法

**Data:** 直线端点  $(x_{start}, y_{start})$ 、 $(x_{end}, y_{end})$

**Result:** 直线占据的栅格 *GridSet*

- 1 计算端点  $(x_{start}, y_{start})$  和  $(x_{end}, y_{end})$  所在栅格的中心点  $(x_{c0}, y_{r0})$ 、 $(x_{c1}, y_{r1})$ ；
- 2 将  $(x_{c0}, y_{r0})$ 、 $(x_{c1}, y_{r1})$  放入 *GridSet*； $x_i \leftarrow x_{c0}$ ， $y_i \leftarrow y_{r0}$ ；根据公式计算  $\bar{p}_i, p_i$ ；
- 3 **while**  $x_i \neq x_{c1}$  **do**
- 4     **if**  $\bar{p}_i > 0$  **then**
- 5          $x_i \leftarrow x_i$ ， $y_i \leftarrow y_i + p_{row} \cdot l$ ； $p_i \leftarrow p_i - 2\Delta x \cdot l$ ， $\bar{p}_i \leftarrow \bar{p}_i - 2\Delta x \cdot l$ ；
- 6     **else**
- 7         将  $(x_i, y_i)$  放入 *GridSet*， $x_i \leftarrow x_i + p_{col} \cdot l$ ， $y_i \leftarrow y_i + (p_i > 0 ? p_{row}l : 0)$ ；
- 8          $p_i \leftarrow p_i + 2l\Delta y - (p_i > 0 ? 2l\Delta x : 0)$ ；
- 9          $\bar{p}_i \leftarrow \bar{p}_i + 2l\Delta y - (p_i > 0 ? 2l\Delta x : 0)$ ；
- 10    **end**
- 11 **end**

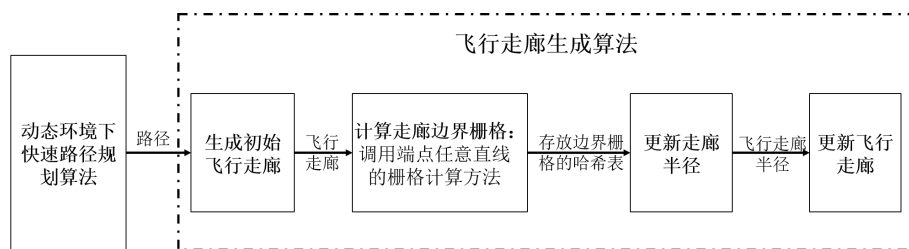


图 3-4 飞行走廊生成算法流程框图

### 3.3.2 飞行走廊生成算法

图3-4展示了飞行走廊生成算法的整体流程。飞行走廊快速生成算法首先对路径规划子模块得到的路径进行膨胀，得到一个平行于路径的边界盒子，然后使用上文提出的栅格计算方法计算飞行走廊边界对应栅格，最后对处于飞行走廊内的地图栅格遍历检查是否包含障碍物栅格，并以此调整飞行走廊大小，具体步骤如下文所述。

步骤一：生成初始飞行走廊。如图3-5所示，算法首先根据路径规划模块计算的路径为基础，生成一个中心线与路径重合的边界盒子，当前路径的两个端点坐标分别为  $(x_1, y_1)$  和  $(x_2, y_2)$ ，其连接的路径的直线方程为：

$$Ax + By + C = 0, A = y_2 - y_1, B = x_1 - x_2, C = x_2y_1 - x_1y_2 \quad (3-24)$$

本论文设边界盒子的边界与路径直线的距离为  $r$ ，则可以得到飞行走廊四个边界的方程，进而得到当前飞行走廊的范围为：

$$\begin{cases} D_1D_2: Bx - Ay + (Ay_2 - Bx_2) + r\sqrt{A^2 + B^2} \geq 0 \\ D_2D_3: Ax + By - r\sqrt{A^2 + B^2} \leq 0 \\ D_3D_4: Bx - Ay + (Ay_1 - Bx_1) - r\sqrt{A^2 + B^2} \leq 0 \\ D_4D_1: Ax + By + r\sqrt{A^2 + B^2} \geq 0 \end{cases} \quad (3-25)$$

步骤二：计算走廊边界栅格。根据路径得到飞行走廊之后，论文运用上一节提出的栅格计算方法来分别计算飞行走廊四个边界在地图中对应的栅格，最终得到围绕飞行走廊的边界栅格。

步骤三：更新走廊半径。将上一步计算的边界栅格保存在一个哈希表中，然后使用广度优先遍历算法来搜索飞行走廊中的障碍物栅格。使用一个队列来维护要遍历的栅格节点，首先选择飞行走廊内的任一栅格作为遍历起点，将其放入队列中，接着进入循环，每次从队列中弹出一个栅格节点，若其能在哈希表中找到，则说明当前栅格为边界栅格；若未能在哈希表中找到，首先将其相邻点加入队列中，然后判断当前点是否是障碍物栅格，如果是障碍物栅格，则如图3-6所示，分

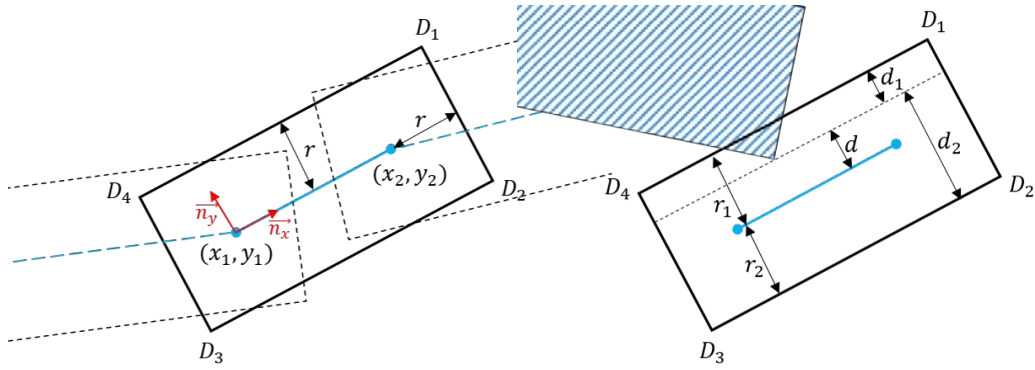


图 3-5 飞行走廊生成原理

 图 3-6  $d$ 、 $d_1$ 、 $d_2$  计算方法

别计算当前栅格与路径中轴线、两个走廊边界的距离，并根据规则：

$$d_1 < d_2 \quad ? \quad r_2 = \min \{r_2, d\} : r_1 = \min \{r_1, d\} \quad (3-26)$$

将走廊半径更新。

步骤四：更新飞行走廊。当所有飞行走廊内栅格都遍历完后，走廊半径也即更新完成，然后依照步骤一，再次计算更新后的飞行走廊。对路径规划模块的每一段路径都执行上述操作，便得到了完整的无人机飞行走廊，该飞行走廊将在后续轨迹规划模块中用来作为约束求解最优轨迹。

### 3.4 仿真实验与结果分析

#### 3.4.1 端点任意直线的栅格计算方法的对比实验

本节在 MATLAB 中设计实验来验证端点任意直线的栅格计算方法的有效性并与 Bresenham 算法 [34] 进行比较分析。

设置栅格分辨率  $l$  为 0.2m，设置直线起点与终点坐标分别为 (1.25, 0.58) 和 (4.3, 3.55)，结果如图3-7(a)(b) 所示。

本文提出直线覆盖率的概念来衡量算法的效果，它表示处在栅格内的直线部分占整段直线的比例，其计算公式为：

$$\eta = \frac{L_{occupied}}{L_{all}} \quad (3-27)$$

其中， $L_{occupied}$  表示直线在栅格中的长度， $L_{all}$  表示直线的总长度。图3-7(a)(b) 中端点任意直线栅格计算方法找到的栅格对直线的覆盖率为 63.46%，Bresenham 算法找到的栅格对直线的覆盖率为 54.95%。从图中可以看到，Bresenham 直线栅格计算方法由于将直线端点修改为所在栅格中心点，则导致实际计算的是连接栅格中

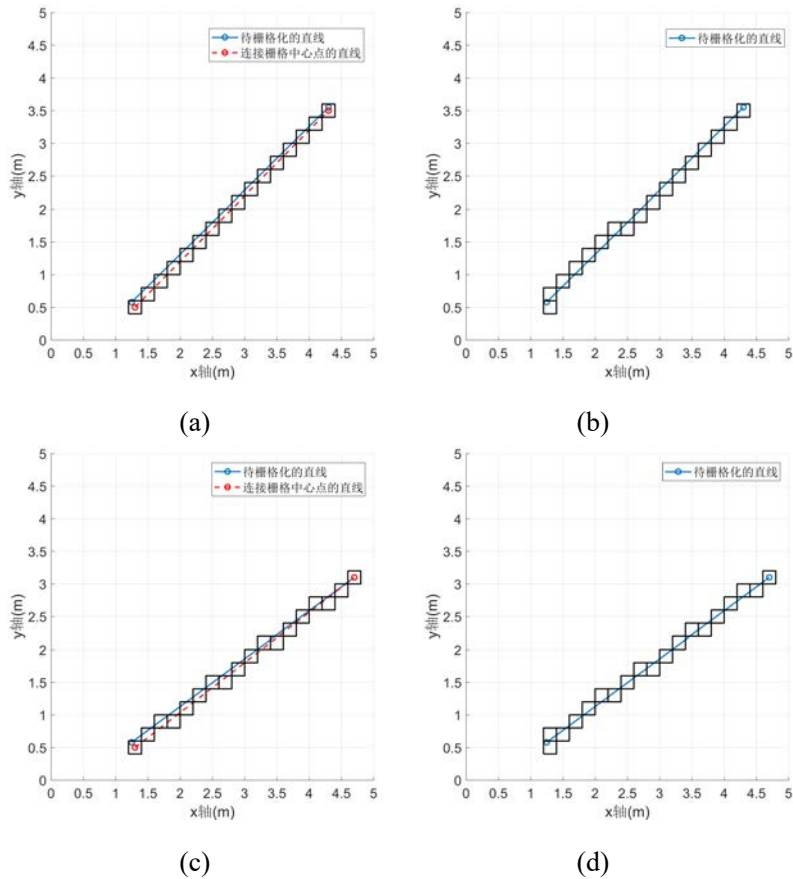


图 3-7 两种直线栅格计算方法的效果，黑色方框表示算法计算的栅格，实线表示待栅格化的直线，虚线表示 Bresenham 算法实际栅格化的直线。(a)(c): Bresenham 算法计算的栅格；(b)(d): 端点任意直线的栅格计算方法计算的栅格

表 3-1 不同栅格分辨率情况下两种算法的覆盖率

栅格分辨率 $l$	0.1m	0.2m	0.3m	0.4m	0.5m
端点任意直线的栅格计算方法的覆盖率	83.22%	84.82%	81.42%	83.12%	84.72%
Bresenham 算法的覆盖率 [34]	79.32%	70.13%	76.62%	72.53%	81.02%
栅格分辨率 $l$	0.6m	0.7m	0.8m	0.9m	1.0m
端点任意直线的栅格计算方法的覆盖率	79.72%	81.42%	86.51%	84.72%	84.72%
Bresenham 算法的覆盖率 [34]	79.72%	72.43%	56.74%	84.72%	82.92%

心点的直线所对应的栅格，其对原始真实直线的覆盖较差。图3-7(c)(d) 展示了端点

坐标分别是 (1.25, 0.58) 和 (4.7, 3.1) 的直线使用两种算法计算的栅格，端点任意直线的栅格计算方法与 Bresenham 算法的覆盖率分别是 79.42% 和 68.84%。表3-1表示在不同栅格分辨率情况下两种算法产生的栅格对直线的覆盖率，从表中可以看出端点任意直线的栅格计算方法的覆盖率在某些分辨率下大于 Bresenham 算法的覆盖率，在另外一些分辨率下两者相同，整体上，端点任意直线的栅格计算方法的覆盖效果要优于 Bresenham 算法。

### 3.4.2 飞行走廊生成算法对比实验

本节在 MATLAB 上设计相关实验来检验本章提出的飞行走廊算法的效果。首先设计实验验证了算法的有效性并展示算法具体效果，然后设计实验对比不同搜索范围的飞行走廊生成算法的搜索时间快慢，最后将本章算法与已有最新算法进行综合对比分析。

#### 3.4.2.1 飞行走廊算法效果验证试验

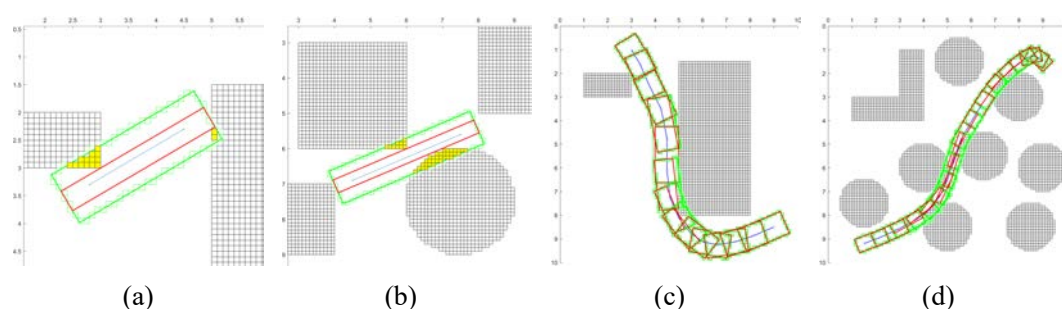


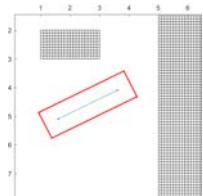
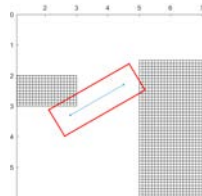
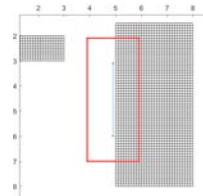
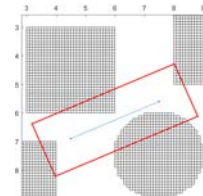
图 3-8 论文提出的飞行走廊生成算法的效果。(a) 地图一，单个走廊效果；(b) 地图三，单个走廊效果；(c) 地图一，从 (3,1) 到 (9,9)，整个飞行走廊效果；(d) 地图三，从 (9,2) 到 (1.5, 9.1)，整个飞行走廊效果

如图3-8所示，本论文使用 MATLAB 实现了基于端点任意直线栅格计算方法的飞行走廊生成算法，图 (a)(b) 中黑色方框表示障碍物栅格，绿色实线表示初始飞行走廊，绿色方框表示使用端点任意直线的栅格计算方法得到的边界栅格，黄色方框表示在飞行走廊内的障碍物栅格，红色实线表示根据走廊内障碍物栅格进行更新后的最终飞行走廊，可以看到，更新完后的飞行走廊内部不包含任何障碍物，是彻底的自由空间。从图 (c)(d) 中可以看出算法都生成了很好的飞行走廊，且每段飞行走廊之间都相互连接，保证了后续求解轨迹规划问题时有解。

### 3.4.2.2 不同搜索范围所用时间的对比试验

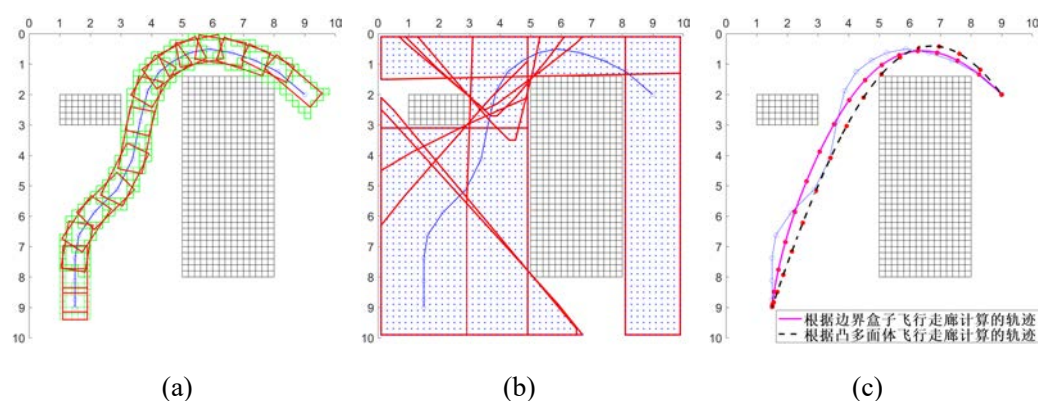
本小节设计实现了飞行走廊的四种常见情况，编写代码实现并对比了两种不同遍历方法的所耗时长。表3-2表示的在常见的四种飞行走廊情况下，在走廊边界栅格限定的范围内做遍历搜索的时间与在全局栅格地图上做遍历搜索的时间 [33] 对比。可以看到，使用边界栅格限定范围之后，大幅减少了栅格搜索数目，使得搜索遍历时间总体上缩短了约 75%。

表 3-2 限定遍历范围与全局地图遍历所用时间

不同飞行走廊				
在边界栅格 范围内遍历用时	2.02ms	3.96ms	5.63ms	6.44ms
在整个栅格地 图中遍历用时 [33]	13.42ms	12.94ms	15.62ms	17.67ms

### 3.4.2.3 与已有飞行走廊生成算法的对比试验

本节在四个分辨率为 0.2m 的地图中进行实验来对比分析本论文提出的飞行走廊生成算法与已有的凸多面体飞行走廊生成算法 [28] 的效果。图3-9展示了这两种算法在四个地图中生成的飞行走廊以及后端轨迹规划部分基于两种不同飞行走廊计算的无人机轨迹。可以看出，虽然凸多面体飞行走廊生成算法所计算的飞行走廊占据的自由空间更大，本论文提出的算法计算的飞行走廊占据空间更小；但基于两种飞行走廊而产生的轨迹质量相当，在地图三情况中，基于本论文飞行走廊生成算法所计算的轨迹质量要优于基于凸多面体飞行走廊生成算法所计算的轨迹。





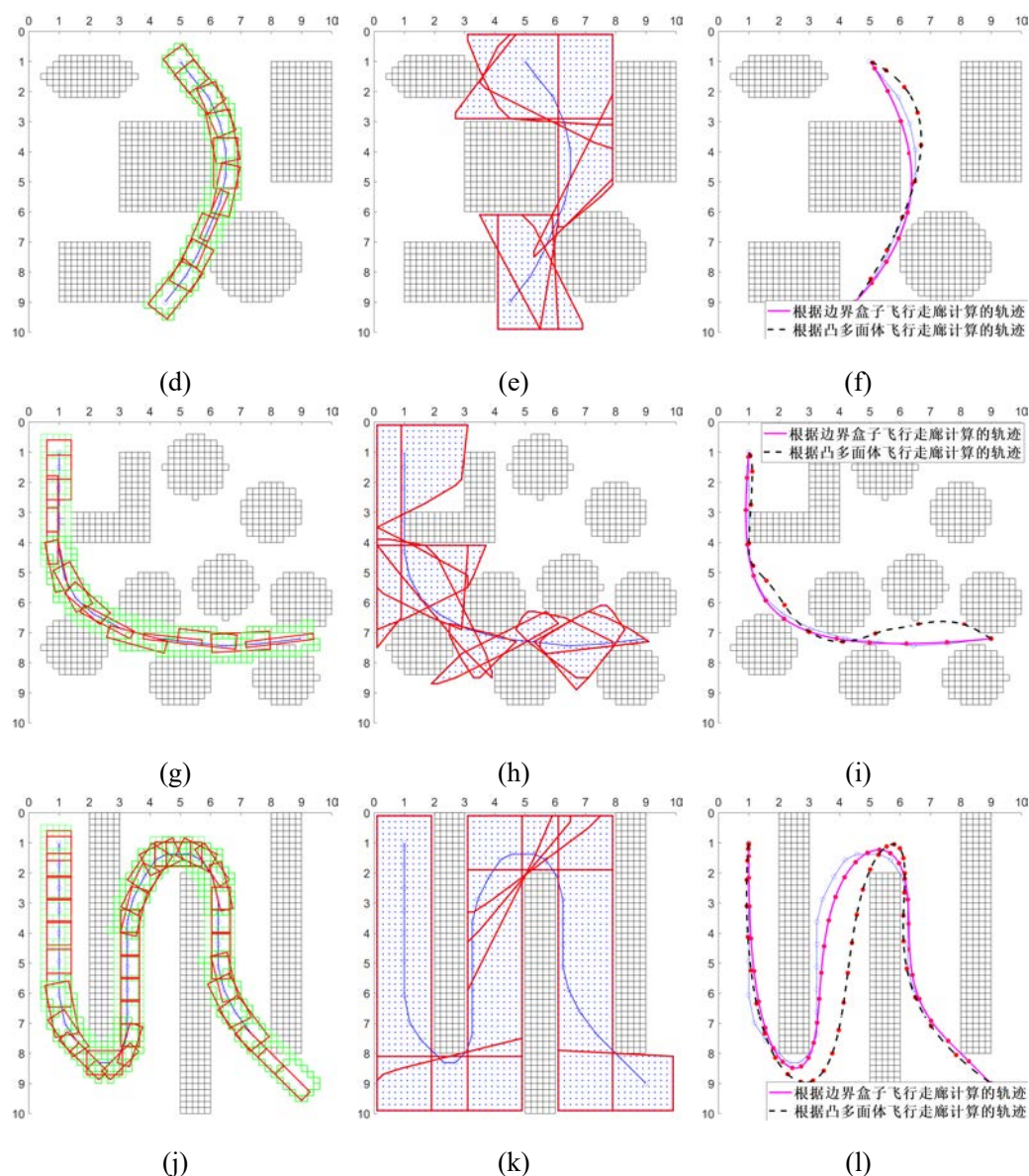


图 3-9 两种飞行走廊生成算法的效果，地图一：从 (1.5, 9) 到 (9,2)；地图二，从 (5,1) 到 (4.5, 9)；地图三，从 (1,1) 到 (9,7)；地图四，从 (1,1) 到 (9,9)。  
(a)(d)(g)(j)：本论文提出算法计算的飞行走廊；(b)(e)(h)(k)：已有的凸多面体飞行走廊生成算法计算的飞行走廊；(c)(f)(i)(l)：基于上述两种形状的飞行走廊计算的轨迹

表3-3展示了在四个地图中两种飞行走廊生成算法计算走廊所用时间以及基于生成的走廊计算轨迹所用的时间。可以看出，凸多面体飞行走廊生成算法的走廊计算时间是本论文提出的算法所消耗时间的 2 到 3 倍；除此之外，在大多数场景中，基于本论文提出算法所生成走廊的轨迹计算时间也少于基于凸多面体飞行走廊的轨迹计算时间，原因在于凸多面体飞行走廊边界约束复杂，给后端轨迹规划

带来大量约束条件。

表 3-3 两种飞行走廊算法的计算时间与后端轨迹计算时间

	飞行走廊计算时间			
	地图一	地图二	地图三	地图四
本论文提出的飞行走廊算法	0.55s	0.43s	0.45s	1.03s
凸多面体飞行走廊算法 [28]	1.76s	1.54s	1.52s	2.02s

	后端轨迹计算时间			
	地图一	地图二	地图三	地图四
本论文提出的飞行走廊算法	1.75s	0.58s	1.12s	10.23s
凸多面体飞行走廊算法 [28]	2.13s	0.59s	9.97s	9.89s

综上所述，本论文提出的飞行走廊生成算法计算时间更短，基于此计算的轨迹与基于凸多面体飞行走廊计算的轨迹质量相当，甚至在某些场景中更优，且轨迹计算时间更短。因此，本论文提出的飞行走廊生成算法的整体效果要优于已有算法。

### 3.5 本章小结

本章首先针对 Bresenham 算法忽略栅格尺寸带来的计算栅格不准确问题首先设计出一个端点任意直线的栅格计算方法，添加了验证当前栅格有效性的内容，扩展了候选栅格数目。然后本章基于该方法提出一种飞行走廊生成算法，该算法运用几何信息构建初始走廊，并使用改进的直线栅格计算方法得到边界栅格，并使用广度优先遍历算法检查障碍物栅格，并据此更新飞行走廊。仿真实验结果表明端点任意直线的栅格计算方法计算栅格更加准确，覆盖效果优于 Bresenham 算法；飞行走廊生成算法运算速度更快，能够在多种地图中生成安全有效的飞行走廊，为后续最优轨迹求解提供了有利条件。



## 第四章 动态环境下避障轨迹优化算法

### 4.1 引言

全局路径规划部分会得到一条粗糙的全局路径，需要通过局部轨迹优化来对全局路径进一步优化，计算出一条安全可执行的平滑轨迹。无人机所处动态环境中存在各种障碍物，这也要求最终得到的可执行轨迹能够躲避障碍物。环境中已知的障碍物包括静态和动态障碍物。对于静态障碍物的躲避，可通过在轨迹优化算法中添加飞行走廊约束的方式进行处理。对于动态障碍物的躲避，已有的局部轨迹优化算法往往将动态障碍物避障要求建模为某部分代价函数或约束，这会导致需要求解一个非凸优化问题，计算难度大、时间长，难以求得最优解，也无法实现实时躲避障碍物。

本章提出一种动态环境下避障轨迹优化算法，以生成能够躲避环境中已知障碍物的平滑轨迹。该算法首先将动态障碍物分为三类，运用几何理论分别处理，建模得到两种线性约束。然后使用 MINVO 基函数来构建无人机轨迹，结合前述两种避障线性约束、飞行走廊边界约束以及动力学约束等，将最小化急动度 (Minimum Jerk) 的避障轨迹求解问题建模为以 MINVO 基函数控制点为优化变量的凸二次规划问题，并调用 MATLAB 中的 `quadprog()` 函数求解得到全局最优解。最后，本章设计仿真实验来分析验证算法避障效果与鲁棒性。

### 4.2 动态障碍物避障建模

上一章得到的飞行走廊内部不包含任何静态障碍物，因此将飞行走廊约束加入轨迹优化算法中可容易实现对已知静态障碍物的躲避，但运动障碍物仍然会穿过飞行走廊。针对这一点，本章首先对动态障碍物进行避障建模。在已知运动障碍物运动轨迹情况下，该避障建模方法会根据运动物体相对于飞行走廊的位置将穿过飞行走廊的运动障碍物分为三类，对这三类动态障碍物分别使用不同的建模方法进行处理。若运动障碍物运动状态临时改变并且无法再准确预测其运动轨迹，无人机则运用紧急避障算法进行处理，算法具体内容将在下一章进行阐述。

#### 4.2.1 障碍物分类方法

动态障碍物分类如图4-1所示，图中长方形表示上一章得到的飞行走廊；走廊中心的线段表示该段飞行走廊对应的路径；曲线表示的是动态障碍物在该段飞行走廊对应的时段  $[t_1, t_2]$  内的运动轨迹。图4-1(a)所示的是第一类动态障碍物，

其在  $[t_1, t_2]$  内不会穿过走廊中心的路径所在的直线，且只会经过一个走廊边界；图4-1(b)所示的是第二类动态障碍物，其在  $[t_1, t_2]$  内会穿过路径所在直线，该类直线的子类型较多，处理方式复杂，具体细节在后文讲述；图4-1(c)所示的是第三类动态障碍物，其在对应时间段内不会穿过路径，且会经过飞行走廊的两个不同边界。

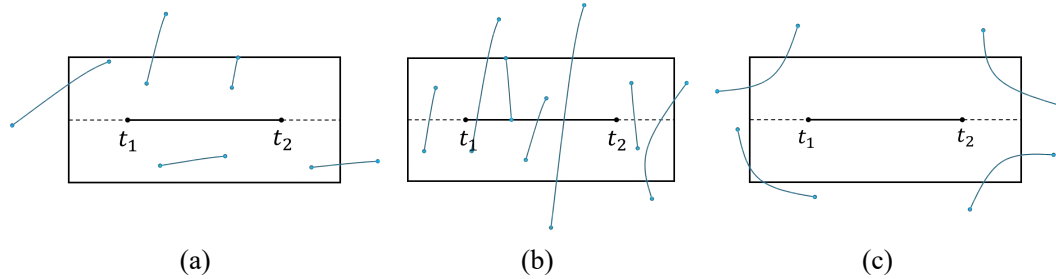


图 4-1 三种动态障碍物分类。(a) 第一类障碍物；(b) 第二类障碍物；(c) 第三类障碍物

接下来对动态障碍物分类方法进行阐述。对运动物体轨迹方程在  $[t_1, t_2]$  时间段内采样遍历，根据轨迹方程计算运动物体当前位置，判断其进入走廊时刻  $t_s$  与飞出走廊时刻  $t_e$ ；最后判断进入飞行走廊后，运动物体是否与当前路径所在的直线相交以及相交时间  $t_c$ 。若相交，则其属于第二类障碍物；否则，判断在  $t_s$  和  $t_e$  时刻是否到飞行走廊某边界的距离为 0，若两个时刻都在飞行走廊边界上，则其属于第三类障碍物；若只有一个时刻在飞行走廊边界上，则其属于第一类障碍物；如果以上条件都不满足，则表示运动物体不会穿过飞行走廊。

## 4.2.2 障碍物建模方法

### 4.2.2.1 第一类障碍物建模处理

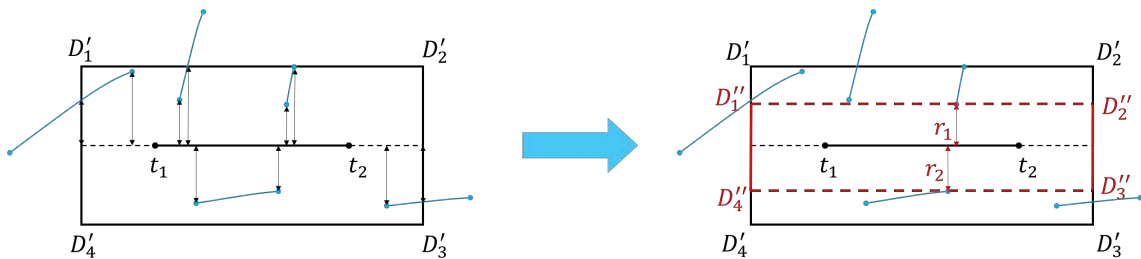


图 4-2 第一类障碍物建模处理方法

对于第一类障碍物，首先根据其运动轨迹方程分别计算  $t_s$  和  $t_e$  时刻对应位置，分别判断这两个位置属于哪一侧飞行走廊，并将该侧飞行走廊的半径进行压缩，遍历所有第一类障碍物，找到最小飞行走廊半径。如图4-2所示，对第一类障碍物

进行处理后，原来的用实线表示的走廊边界更新为虚线表示的走廊边界。最终第一类障碍物建模得到避障约束条件为：在  $[t_1, t_2]$  时间段里，无人机轨迹必须位于  $D_1''D_2''$ ， $D_2''D_3''$ ， $D_3''D_4''$ ， $D_4''D_1''$  这四条直线围成的区域里面，本论文将这种在某个时间区间内无人机轨迹都必须满足的恒定约束称为**区间边界约束**。

#### 4.2.2.2 第二类障碍物建模处理

第二类障碍物种类最多，本论文对其做了进一步的细分处理。如图4-3所示，过路径起点  $p$  作一条分界线，将整个飞行走廊分为 L 和 R 两个区域，本论文首先对 R 区域中的障碍物进行建模处理。

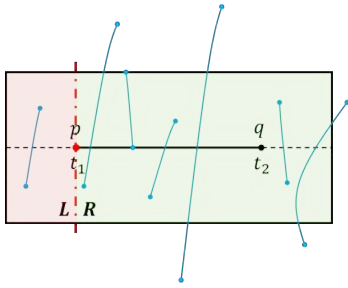


图 4-3 第二类障碍物进一步细分

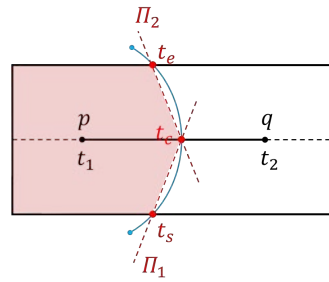


图 4-4 第二类普通障碍物的建模处理方法

第二类障碍物的运动轨迹都会与飞行走廊中心的路径直线相交，R 区域障碍物轨迹与路径直线的交点和路径终点  $q$  都位于同一侧。对于 R 区域中的障碍物，首先计算其运动轨迹在  $t_s$ 、 $t_c$  和  $t_e$  时刻的位置点，然后根据  $t_s$  与  $t_c$  时刻的位置点确定直线  $\Pi_1$ ，根据  $t_c$  与  $t_e$  时刻的位置点确定直线  $\Pi_2$ ，如图4-4所示，本论文设计的避障条件为在  $t_c$  时刻时无人机轨迹应当与路径起点  $p$  处于直线  $\Pi_1$  同一侧；在  $t_e$  时刻时，无人机轨迹应当与  $p$  同处于直线  $\Pi_2$  同一侧，最终得到的效果是无人机轨迹在  $[t_s, t_e]$  时间段内应该处于图4-4阴影区域。这里的思路是当无人机与障碍物相遇时，应作出让步，让障碍物先走，无人机在此处先减速，等障碍物穿过飞行走廊之后，再正常飞行。上述约束条件用公式表达为：

$$\begin{cases} A_{\Pi_1}x(t_c) + B_{\Pi_1}y(t_c) + C_{\Pi_1} \leq 0 \\ A_{\Pi_2}x(t_e) + B_{\Pi_2}y(t_e) + C_{\Pi_2} \leq 0 \end{cases} \quad (4-1)$$

从数学形式上看，该公式表示的避障约束条件只对某个时刻的无人机轨迹进行了约束，因此本论文将这种约束称为**瞬时边界约束**，即无人机在某个时刻需要在某个边界内的约束。

当有多个第二类障碍物时，如图4-5所示，无人机均需进行减速避让，由于无

人机在该段飞行走廊对应时段内的大多数时间都在避让，容易导致无人机无法在当前时段结束之前完整穿过飞行走廊，在数学形式上即会导致后续的优化问题最终无解。因此，当遇到第二类障碍物时，本论文会将当前飞行走廊对应的飞行时长加倍，以应对这一问题，保证无人机有充足飞行时间。

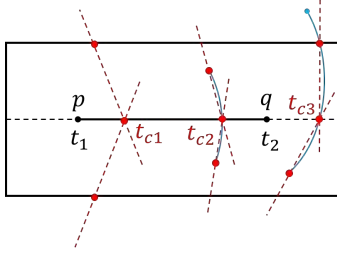


图 4-5 多个第二类障碍物的建模处理方法

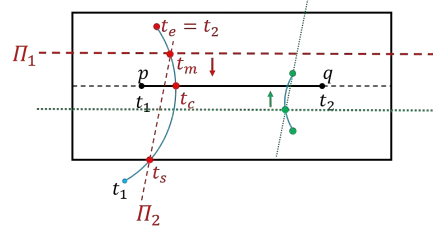


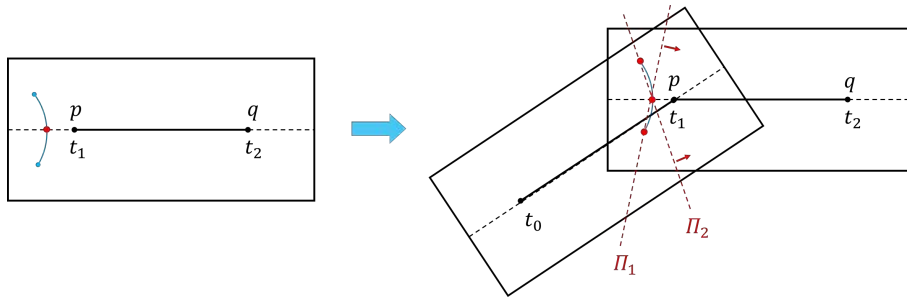
图 4-6  $t_e = t_2$  特殊情况建模处理方法

对于  $t_2 = t_e$  这种特殊情况来说，上述避障建模方法已不再适用，原因是上述避障思路是无人机对动态障碍物进行减速避让，等障碍物走远后再继续前进，而  $t_2 = t_e$  这种特殊情况会导致无人机在整个飞行时间段内都在避让障碍物，不再有时间完成剩余飞行。因此，针对这种特殊情况，本论文设计了如图4-6所示的建模方法。取  $t_c$  和  $t_e$  的中间时刻  $t_m = \frac{t_c + t_e}{2}$ ，计算其对应的障碍物位置，作直线  $\Pi_1$  过  $t_m$  对应位置点且平行于路径直线，过  $t_s$  与  $t_m$  计算另一条边界直线  $\Pi_2$ ，建模避障条件如下：

$$\begin{cases} \Pi_1 : A_{\Pi_1}x(t) + B_{\Pi_1}y(t) + C_{\Pi_1} \leq 0, & \forall t \in [t_1, t_2] \\ \Pi_2 : A_{\Pi_2}x(t_m) + B_{\Pi_2}y(t_m) + C_{\Pi_2} \leq 0 \end{cases} \quad (4-2)$$

式4-2中  $\Pi_1$  属于区间边界约束，要求无人机在整个飞行期间都不能越过该边界； $\Pi_2$  属于瞬时边界约束，其约束无人机只在  $t_m$  时刻不能越过边界直线。

L 区域障碍物的交点与路径终点  $q$  分处于路径起点  $p$  两侧，L 区域中的障碍物无法在当前飞行走廊中直接使用上述建模方法进行处理，原因是 L 区域的障碍物处于起点左侧，上述避障建模方法会要求无人机在障碍物飞行时段内处于起点左侧区域，而无人机刚开始时处于起点位置，在飞行过程中一直往右侧终点  $q$  方向飞行，因此这种约束会导致后续优化问题无解。本论文将这样的障碍物放在前一个飞行走廊中进行处理，如图4-7所示，按照前述思路，根据  $t_s$ 、 $t_c$ 、 $t_e$  三个时刻对应的位置点计算出两个边界直线  $\Pi_1$ 、 $\Pi_2$  然后设计两个瞬时边界约束将其添加到上一个飞行走廊的约束条件当中，要求在  $t_1$  时刻时无人机应该在  $\Pi_1$  和  $\Pi_2$  右侧，


 图 4-7 交点位于起点  $p$  左侧的动态障碍物建模处理方法

即与当前走廊起点位置  $p$  同一侧，避障约束具体公式为：

$$\begin{cases} \Pi_1 : A_{\Pi_1}x(t_1) + B_{\Pi_1}y(t_1) + C_{\Pi_1} \geq 0 \\ \Pi_2 : A_{\Pi_2}x(t_1) + B_{\Pi_2}y(t_1) + C_{\Pi_2} \geq 0 \end{cases} \quad (4-3)$$

通过在前一个飞行走廊中施加该瞬时边界约束，使得无人机在前一个飞行走廊中飞行时，在其终点时刻，即  $t_1$  时刻就要提前到达下一个飞行走廊的起点位置，通过这种抢跑的方式避开动态障碍物，而且这种建模处理方法不会影响前一个飞行走廊的飞行时长与其他约束。

#### 4.2.2.3 第三类障碍物建模处理

对于第三类障碍物的处理，与第一类障碍物的处理思路类似，首先根据动态障碍物运动轨迹计算  $t_s$  和  $t_e$  时刻对应位置点，然后根据位置点确定边界直线，并将该边界直线朝路径起点位置所在的一侧平移若干距离，如图4-8所示。对第三类障碍物处理后得到的避障约束条件为：在  $[t_1, t_2]$  时间段内，无人机轨迹必须与路径起点都位于  $\pi_1, \pi_2, \pi_3, \pi_4$  这些边界直线的同一侧，该约束也属于区间边界约束。

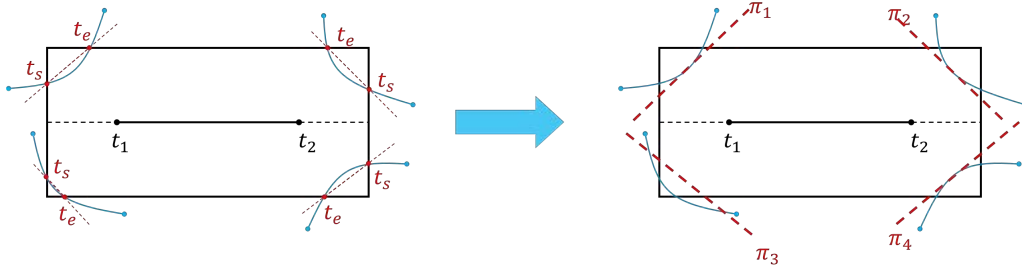


图 4-8 第三类障碍物建模处理方法

### 4.3 避障轨迹优化算法

本节将动态环境下的轨迹规划问题建模为一个 Minimum Jerk 的凸优化问题，使用论文 [52] 提出的最小体积基作为多项式轨迹的基函数，并添加连续性约束、

动力学约束以及上一节得到的动态障碍物瞬时边界约束和区间边界约束，最终得到一个具有线性约束的凸二次规划问题求解。下面首先对最小体积基函数进行介绍。

### 4.3.1 最小体积基函数

最小体积基函数 ( Minimum Volume Basis , MINVO ) 是一种多项式基函数。给定一个  $n$  阶多项式曲线，用该多项式基函数表示之后对应的控制点所形成  $n$  单纯形的体积要比使用贝塞尔曲线或者  $b$  样条曲线对应的控制点所形成的单纯形体积要更小。反过来，给定  $n$  单纯形，由 MINVO 基函数所得到的多项式曲线围成的凸包的体积相比贝塞尔曲线或者  $b$  样条曲线所围成的凸包的体积更大，换言之，给定相同的边界区域，使用 MINVO 基函数得到的曲线长度更长，相比于贝塞尔曲线和  $b$  样条曲线，表示无人机轨迹时无人机运动更敏捷更快速。设无人机多项式轨迹为  $N$  阶多项式，则 MINVO 基函数的控制点与多项式曲线各系数之间的转换公式为：

$$\mathbf{P} = \mathbf{A}_{MV} \mathbf{V} \quad (4-4)$$

其中  $\mathbf{P}$  表示  $N$  阶多项式曲线的系数矩阵， $\mathbf{V}$  表示  $(N+1)$  个控制点的顶点坐标矩阵。其具体组成如式4-5所示，式中的  $p_i^k$  表示  $k$  维度下时间的  $i$  次幂的系数， $v_i^k$  表示轨迹第  $i$  个控制点的  $k$  维度坐标分量。

$$\mathbf{P} = \begin{bmatrix} p_N^x & p_{N-1}^x & \cdots & p_1^x & p_0^x \\ p_N^y & p_{N-1}^y & \cdots & p_1^y & p_0^y \end{bmatrix}, \mathbf{V} = \begin{bmatrix} v_0^x & v_1^x & \cdots & v_{N-1}^x & v_N^x \\ v_0^y & v_1^y & \cdots & v_{N-1}^y & v_N^y \end{bmatrix} \quad (4-5)$$

$\mathbf{A}_{MV}$  为转换矩阵，不同的阶数  $N$  会对应具体值不同的转换矩阵，本论文打算采用 5 阶多项式曲线来表示无人机轨迹，对应转换矩阵为：

$$\mathbf{A}_{MV} = \begin{bmatrix} 0.7392 & 0.7769 & 0.3302 & 0.3773 & 0.0365 & 0.04589 \\ 1.503 & 1.319 & 1.366 & 1.333 & 0.121 & 0.002895 \\ 1.75 & 0.5424 & 2.777 & 0.9557 & 1.064 & 0.4512 \\ 1.75 & 0.5424 & 2.777 & 0.9557 & 1.064 & 0.4512 \\ 1.503 & 1.319 & 1.366 & 1.333 & 0.121 & 0.002895 \\ 0.7392 & 0.7769 & 0.3302 & 0.3773 & 0.0365 & 0.04589 \end{bmatrix} \quad (4-6)$$

设飞行走廊个数为  $M$ ，第  $i$  个飞行走廊内的多项式轨迹的系数向量  $p_{xi}$ 、 $p_{yi}$  和

控制点坐标向量  $v_{xi}$ 、 $v_{yi}$  分别为:

$$\begin{cases} p_{xi} = (p_{N,i}^x, p_{N-1,i}^x \dots p_{0,i}^x)^T \\ p_{yi} = (p_{N,i}^y, p_{N-1,i}^y \dots p_{0,i}^y)^T \end{cases} \quad \begin{cases} v_{xi} = (v_{0,i}^x, v_{1,i}^x \dots v_{N,i}^x)^T \\ v_{yi} = (v_{0,i}^y, v_{1,i}^y \dots v_{N,i}^y)^T \end{cases} \quad (4-7)$$

式中的  $p_{j,i}^k$  表示第  $i$  段轨迹中第  $j$  次幂的  $k$  维度坐标分量,  $v_{j,i}^k$  表示第  $i$  段轨迹的第  $j$  个控制点的  $k$  维度坐标分量, 向量的上标  $T$  表示矩阵转置, 若无特殊说明, 后文出现的上标  $T$  表示意思与此处相同。根据式4-4得:

$$\begin{bmatrix} v_{xi}^T \\ v_{yi}^T \end{bmatrix} A_{MV} = \begin{bmatrix} p_{xi}^T \\ p_{yi}^T \end{bmatrix} \Rightarrow \begin{cases} p_{xi} = A_{MV}^T v_{xi} \\ p_{yi} = A_{MV}^T v_{yi} \end{cases} \quad (4-8)$$

根据式4-8, 则由  $M$  个飞行走廊的  $M$  段飞行轨迹拼接而成的总的轨迹系数向量与控制点顶点向量的转换关系为:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \underbrace{\begin{bmatrix} A_{MV}^T & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_{MV}^T \end{bmatrix}}_{T_r} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = T_r \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad (4-9)$$

### 4.3.2 二次规划问题建模求解

#### 4.3.2.1 代价函数

设从无人机经过每个路径点的时间分别为  $T_0$ 、 $T_1$ 、...  $T_{M-1}$ 、 $T_M$ , 构造代价函数  $J$  为:

$$J = \sum_{i=1}^M \int_{T_{i-1}}^{T_i} (\mathbf{p}_{x,i}^{(3)}(t))^2 dt + \sum_{i=1}^M \int_{T_{i-1}}^{T_i} (\mathbf{p}_{y,i}^{(3)}(t))^2 dt \quad (4-10)$$

代价函数中,  $\mathbf{p}_{x,i}(t)$  和  $\mathbf{p}_{y,i}(t)$  分别表示第  $i$  段飞行走廊中无人机在  $x$  轴和  $y$  轴的解耦运动轨迹, Minimum Jerk 轨迹规划问题就是要求轨迹的关于时间的三阶导即 Jerk 物理量最小。对代价函数进一步推导可得:

$$\begin{aligned} J &= \sum_{i=1}^M \int_{T_{i-1}}^{T_i} (\mathbf{p}_{x,i}^{(3)}(t))^2 dt + \sum_{i=1}^M \int_{T_{i-1}}^{T_i} (\mathbf{p}_{y,i}^{(3)}(t))^2 dt \\ &= \sum_{i=1}^M p_{xi}^T Q_i p_{xi} + \sum_{i=1}^M p_{yi}^T Q_i p_{yi} \\ &= \begin{bmatrix} p_x \\ p_y \end{bmatrix}^T Q \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}^T T_r^T Q T_r \begin{bmatrix} v_x \\ v_y \end{bmatrix} \end{aligned} \quad (4-11)$$

式4-11中矩阵  $Q$  是块对角矩阵，构成  $Q$  的每一个块  $Q_i$  都是对称矩阵， $Q_i$  的具体形式如式4-12所示，因此  $Q$  是对称矩阵，则  $T_r^T Q T_r$  是对称矩阵。又因为代价函数  $J$  是对 Jerk 的平方进行积分，所以  $J > 0$ ，所以矩阵  $T_r^T Q T_r$  是正定矩阵，代价函数  $J$  是一个凸函数。

$$Q_i = \begin{matrix} & l = N & \dots & l = 3 \\ \begin{matrix} j = N \\ \vdots \\ j = 3 \end{matrix} & \begin{bmatrix} & & & \\ & & & \vdots \\ \dots & \frac{j(j-1)(j-2)l(l-1)(l-2)}{j+l-5} (T_i^{j+l-5} - T_{i-1}^{j+l-5}) & \dots & \\ & & & \vdots \end{bmatrix} \end{matrix} \quad (4-12)$$

#### 4.3.2.2 起终点状态约束

无人机轨迹方程需要满足起点与终点状态的等式约束，设无人机起点位置为  $(x_0, y_0)$ ，初始速度和初始加速度分别为  $(v_{x0}, v_{y0})$ 、 $(a_{x0}, a_{y0})$ ，对于无人机终点状态，本论文只对其位置进行约束，设置终点位置为  $(x_f, y_f)$ ，则起终点状态约束用公式表示为：

$$\begin{cases} \mathbf{p}_{x,1}(T_0) = x_0 \\ \mathbf{p}_{x,1}^{(1)}(T_0) = v_{x0} \\ \mathbf{p}_{x,1}^{(2)}(T_0) = a_{x0} \\ \mathbf{p}_{x,M}(T_M) = x_f \end{cases}, \begin{cases} \mathbf{p}_{y,1}(T_0) = y_0 \\ \mathbf{p}_{y,1}^{(1)}(T_0) = v_{y0} \\ \mathbf{p}_{y,1}^{(2)}(T_0) = a_{y0} \\ \mathbf{p}_{y,M}(T_M) = y_f \end{cases} \quad (4-13)$$

其中， $f^{(k)}$  表示函数  $f$  的  $k$  阶导，将式4-13中各多项式函数的时间多项式基与其系数分离之后可得到等式状态约束为：

$$\begin{bmatrix} \mathcal{A}_{p0}^x \dots 0, 0 \dots 0 \\ \mathcal{A}_{v0}^x \dots 0, 0 \dots 0 \\ \mathcal{A}_{a0}^x \dots 0, 0 \dots 0 \\ 0 \dots \mathcal{A}_{pf}^x, 0 \dots 0 \\ 0 \dots 0, \mathcal{A}_{p0}^y \dots 0 \\ 0 \dots 0, \mathcal{A}_{v0}^y \dots 0 \\ 0 \dots 0, \mathcal{A}_{a0}^y \dots 0 \\ 0 \dots 0, 0 \dots \mathcal{A}_{pf}^y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} x_0 \\ v_{x0} \\ a_{x0} \\ x_f \\ y_0 \\ v_{y0} \\ a_{y0} \\ y_f \end{bmatrix} \Rightarrow \mathcal{A} T_r \begin{bmatrix} v_x \\ v_y \end{bmatrix} = s \quad (4-14)$$

其中， $\mathcal{A}_{pi}^k$ 、 $\mathcal{A}_{vi}^k$ 、 $\mathcal{A}_{ai}^k$  分别表示  $k$  维度下第  $i$  段位置轨迹、速度轨迹、加速度轨迹的基。



## 4.3.2.3 轨迹连续性约束

除了保证无人机轨迹在起点与终点的约束外，还要保证  $M$  段无人机轨迹是连续的，具体包括在位置、速度和加速度上保证连续，用公式表示为：

$$\begin{cases} p : \mathbf{p}_{x,i}(T_i) = \mathbf{p}_{x,i+1}(T_i) \\ v : \mathbf{p}_{x,i}^{(1)}(T_i) = \mathbf{p}_{x,i+1}^{(1)}(T_i) \\ a : \mathbf{p}_{x,i}^{(2)}(T_i) = \mathbf{p}_{x,i+1}^{(2)}(T_i) \end{cases}, i = 1, 2, \dots, M-1 \quad (4-15)$$

类似于前一节的处理思路，将式4-15中各多项式函数的时间多项式基与其系数分离之后可得连续性约束为：

$$\underbrace{\begin{bmatrix} \mathcal{B}_p^x & 0 \\ 0 & \mathcal{B}_p^y \\ \mathcal{B}_v^x & 0 \\ 0 & \mathcal{B}_v^y \\ \mathcal{B}_a^x & 0 \\ 0 & \mathcal{B}_a^y \end{bmatrix}}_{\mathcal{B}} \begin{bmatrix} p_x \\ p_y \end{bmatrix} = O_{6(M-1) \times 1} \Rightarrow \mathcal{B}T_r \begin{bmatrix} v_x \\ v_y \end{bmatrix} = O_{6(M-1) \times 1} \quad (4-16)$$

其中， $\mathcal{B}_p^k$ 、 $\mathcal{B}_v^k$ 、 $\mathcal{B}_a^k$  分别表示在  $k$  维度下由位置轨迹、速度轨迹、加速度轨迹的基构成的矩阵， $O_{6(M-1) \times 1}$  表示  $6(M-1)$  行 1 列的全零向量。 $\mathcal{B}_p^k$  具体形式如式4-17所示，其中， $\mathcal{B}_{p,i}^k$  表示第  $i$  段位置轨迹对应的时间基构成的向量。 $\mathcal{B}_v^k$ 、 $\mathcal{B}_a^k$  形式与  $\mathcal{B}_p^k$  类似。

$$\mathcal{B}_p^k = \begin{bmatrix} \mathcal{B}_{p,1}^k & -\mathcal{B}_{p,1}^k & \cdots & \cdots & 0 \\ 0 & \mathcal{B}_{p,2}^k & -\mathcal{B}_{p,2}^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{B}_{p,M-1}^k & -\mathcal{B}_{p,M-1}^k \end{bmatrix} \quad (4-17)$$

## 4.3.2.4 瞬时边界约束

瞬时边界约束是指无人机轨迹在某个时刻要在某个边界范围内，这种约束主要来自于上一节中对第二类障碍物的处理建模。设瞬时边界约束的总数量为  $L$ ，瞬时边界约束的一般形式为：

$$a_k \mathbf{p}_{x,i}(t_k) + b_k \mathbf{p}_{y,i}(t_k) + c_k \leq 0, \quad k = 1, 2, \dots, L \quad (4-18)$$

式中,  $a_k, b_k, c_k$  表示第  $k$  个直线边界方程的系数, 将其进一步化简为:

$$\begin{aligned}
 & \underbrace{a_k[t_k^N, t_k^{N-1}, \dots, t_k^0]}_{\vec{a}_k} p_{x,i} + \underbrace{b_k[t_k^N, t_k^{N-1}, \dots, t_k^0]}_{\vec{b}_k} p_{y,i} + c_k \leq 0 \\
 & \Rightarrow \underbrace{[0 \dots \vec{a}_k \dots 0, 0 \dots \vec{b}_k \dots 0]}_{\vec{c}_k} \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq -c_k \\
 & \Rightarrow \mathcal{C}_k \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq -c_k
 \end{aligned} \tag{4-19}$$

将所有瞬时边界约束写入一个矩阵中为:

$$\underbrace{\begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_L \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} p_x \\ p_y \end{bmatrix} \leq \underbrace{\begin{bmatrix} -c_1 \\ \vdots \\ -c_L \end{bmatrix}}_{\mathbf{c}} \Rightarrow \mathcal{C} T_r \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq \mathbf{c} \tag{4-20}$$

#### 4.3.2.5 区间边界约束

区间边界约束包括飞行走廊边界约束, 第一类、第三类障碍物建模的约束, 还有少量第二类障碍物处理得到的约束, 设区间边界约束的总数目为  $H$ , 则约束的一般表示形式为:

$$a_h p_{x,i}(t) + b_h p_{y,i}(t) + c_h \leq 0, \forall t \in [T_{i-1}, T_i] \quad h = 1, 2, \dots, H \tag{4-21}$$

区间边界约束要求无人机轨迹在整个时间段内都处在某个边界范围内, 该约束无法在多项式系数上直接建模。考虑到多项式轨迹包含在由控制点构成凸包内, 若控制点处于某个边界范围内, 则其凸包内的多项式轨迹也会满足边界约束要求, 因此可将上述对多项式系数在整个时间段内的约束等价转换为该多项式轨迹对应的所有控制点都应该处在边界范围内, 用公式表示即为:

$$a_h \begin{bmatrix} v_{0,i}^x \\ \vdots \\ v_{N,i}^x \end{bmatrix} + b_h \begin{bmatrix} v_{0,i}^y \\ \vdots \\ v_{N,i}^y \end{bmatrix} + \begin{bmatrix} c_h \\ \vdots \\ c_h \end{bmatrix} \leq O_{(N+1) \times 1} \quad h = 1, 2, \dots, H \tag{4-22}$$

将上述表达式写为关于  $v_x$ 、 $v_y$  的矩阵形式为：

$$\begin{aligned}
 & \begin{bmatrix} a_h & \cdots & 0 & b_h & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_h & 0 & \cdots & b_h \end{bmatrix} \begin{bmatrix} v_{xi} \\ v_{yi} \end{bmatrix} \leq -c_h e_{(N+1) \times 1} \\
 & \Rightarrow \underbrace{[0 \cdots \mathcal{F}_{ah} \cdots 0 | 0 \cdots \mathcal{F}_{bh} \cdots 0]}_{\mathcal{F}_h} (N+1) \times 2M(N+1) \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq -c_h e_{(N+1) \times 1} \\
 & \Rightarrow \mathcal{F}_h \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq -c_h e_{(N+1) \times 1}
 \end{aligned} \tag{4-23}$$

上式中， $e_{(N+1) \times 1}$  表示  $N+1$  行 1 列的全 1 向量。将所有区间边界约束写到一个约束矩阵中为：

$$\underbrace{\begin{bmatrix} \mathcal{F}_1 \\ \vdots \\ \mathcal{F}_H \end{bmatrix}}_{\mathcal{F}} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq - \underbrace{\begin{bmatrix} c_1 e_{(N+1) \times 1} \\ \vdots \\ c_H e_{(N+1) \times 1} \end{bmatrix}}_{\Phi} \Rightarrow \mathcal{F} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq \Phi \tag{4-24}$$

#### 4.3.2.6 无人机动力学约束

最后，本论文对无人机的动力学约束进行建模，设无人机最大速度和最大加速度分别为  $v_{max}$  和  $a_{max}$ ，以  $x$  轴为例，其动力学约束的一般形式为：

$$\begin{cases} -v_{max} \leq p_{x,i}^{(1)}(t) = \sum_{j=1}^N j \cdot p_{j,i}^x \cdot t^{j-1} \leq v_{max}, & t \in [T_{i-1}, T_i], i = 1, 2, \cdots M \\ -a_{max} \leq p_{x,i}^{(2)}(t) = \sum_{j=2}^N j(j-1) \cdot p_{j,i}^x \cdot t^{j-2} \leq a_{max}, & t \in [T_{i-1}, T_i], i = 1, 2, \cdots M \end{cases} \tag{4-25}$$

无人机动力学约束是对无人机速度、加速度轨迹的区间边界约束，因此本论文使用区间边界约束的处理思路。首先要计算在使用 MINVO 基函数时无人机速度与加速度多项式轨迹的控制点公式，已知位置轨迹的控制点向量为  $v_{xi}$ ，设速度、加速度轨迹的控制点向量分别为  $v'_{xi}$  和  $v''_{xi}$ ，于是可推导出：

$$\begin{cases} \mathbf{A}_{MV}^T v'_{xi} = \bar{N} \mathbf{A}_{MV}^T v_{xi} \\ \mathbf{A}_{MV}^T v''_{xi} = \bar{N} \mathbf{A}_{MV}^T v_{xi} \end{cases} \Rightarrow \begin{cases} v'_{xi} = \mathbf{A}_{MV}^{T-1} \bar{N} \mathbf{A}_{MV}^T v_{xi} \\ v''_{xi} = \mathbf{A}_{MV}^{T-1} \bar{N} \mathbf{A}_{MV}^T v_{xi} \end{cases} \tag{4-26}$$

其中,  $\bar{N}$  为:

$$\bar{N} = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ N & 0 & \cdots & \cdots & 0 \\ 0 & N-1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \quad (4-27)$$

得到  $v'_{xi}$  和  $v''_{xi}$  后, 式4-25可等价于以下约束:

$$\begin{cases} -v_{max}e_{(N+1) \times 1} \leq v'_{xi} \leq v_{max}e_{(N+1) \times 1} \\ -a_{max}e_{(N+1) \times 1} \leq v''_{xi} \leq a_{max}e_{(N+1) \times 1} \end{cases} \Rightarrow \begin{cases} -v_{max}e_{(N+1) \times 1} \leq A_{MV}^{T-1} \bar{N} A_{MV}^T v_{xi} \leq v_{max}e_{(N+1) \times 1} \\ -a_{max}e_{(N+1) \times 1} \leq A_{MV}^{T-1} \bar{N} \bar{N} A_{MV}^T v_{xi} \leq a_{max}e_{(N+1) \times 1} \end{cases} \quad (4-28)$$

考虑  $M$  段轨迹均要满足动力学约束, 将其写入同一个约束矩阵中为:

$$\begin{aligned} -v_{max}e_{2M(N+1) \times 1} &\leq \underbrace{\begin{bmatrix} A_{MV}^{T-1} \bar{N} A_{MV}^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{MV}^{T-1} \bar{N} A_{MV}^T \end{bmatrix}}_{\mathcal{G}_v} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq v_{max}e_{2M(N+1) \times 1} \\ -a_{max}e_{2M(N+1) \times 1} &\leq \underbrace{\begin{bmatrix} A_{MV}^{T-1} \bar{N} \bar{N} A_{MV}^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{MV}^{T-1} \bar{N} \bar{N} A_{MV}^T \end{bmatrix}}_{\mathcal{G}_a} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq a_{max}e_{2M(N+1) \times 1} \end{aligned} \quad (4-29)$$

$$\begin{cases} \mathcal{G}_v \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq v_{max}e_{2M(N+1) \times 1} \\ -\mathcal{G}_v \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq v_{max}e_{2M(N+1) \times 1} \end{cases} \quad \begin{cases} \mathcal{G}_a \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq a_{max}e_{2M(N+1) \times 1} \\ -\mathcal{G}_a \begin{bmatrix} v_x \\ v_y \end{bmatrix} \leq a_{max}e_{2M(N+1) \times 1} \end{cases} \quad (4-30)$$

#### 4.3.2.7 二次规划问题求解

综合上述代价函数与约束推导, 设  $v = [v_x, v_y]^T$ , 本论文给出最终要求解的二次规划问题的数学形式:

$$\begin{aligned} \min J &= v^T T_r^T Q T_r v \\ s.t. &\begin{cases} AT_r v = s, BT_r v = O_{6(M-1) \times 1} \\ CT_r v \leq c, Fv \leq \Phi \\ \mathcal{G}_v v \leq v_{max} e_{2M(N+1) \times 1} \\ -\mathcal{G}_v v \leq v_{max} e_{2M(N+1) \times 1} \\ \mathcal{G}_a v \leq a_{max} e_{2M(N+1) \times 1} \\ -\mathcal{G}_a v \leq a_{max} e_{2M(N+1) \times 1} \end{cases} \end{aligned} \quad (4-31)$$

该二次规划问题的代价函数为凸函数, 约束均为线性约束, 因此该问题为凸二次规划问题, 存在全局最优解。对于该凸优化问题, 本论文使用内点法求解, 具体工程实现时使用 MATLAB 工具箱中的 `quadprog()` 函数进行求解。

#### 4.4 仿真实验与结果分析

本节在 MATLAB 中实现了上述轨迹优化算法, 运用 `quadprog()` 函数求解前述凸二次规划问题, 并复现了论文 [40] 提出的半正定规划 (Semi-Definite Programming, SDP) 算法与论文 [53] 提出的无约束优化算法作为对比算法来进行仿真实验。在实验之前, 设置无人机最大速度和加速度为  $v_{max} = 5m/s$ ,  $a_{max} = 3m/s^2$ , 并使用多项式函数来表示运动障碍物轨迹方程。然后本节在四个地图场景中分别运行三个算法, 观察并记录其计算的轨迹, 与障碍物的距离, 速度、加速度以及轨迹计算时间。

图4-9、4-10、4-11、4-12 分别展示了本论文提出的避障轨迹优化算法与 SDP 规划算法、无约束优化算法在四个地图场景中的实验结果, 图中用星号、五角星、三角形标识的轨迹分别表示三个动态障碍物的运动轨迹, 用圆点标识的细实线表示无人机运动轨迹。从四幅图的第一行子图中可以看出, 三个算法都能生成能够躲避障碍物的顺滑轨迹; 从第二行子图中可以看出, 本论文提出的算法计算轨迹与障碍物的距离相比两个对比算法的结果更远, 即本论文提出的算法生成的轨迹更加安全。除此之外, 从四幅图的第三四行子图中可以看出, 相比两个对比算法, 本章设计的算法计算的轨迹速度与加速度变化都更加平滑, 且一直处于动力学可行范围内。尤其是在像地图四这样的复杂场景中, 本论文提出算法计算的轨迹的速度与加速度平稳程度明显优于两个对比算法。

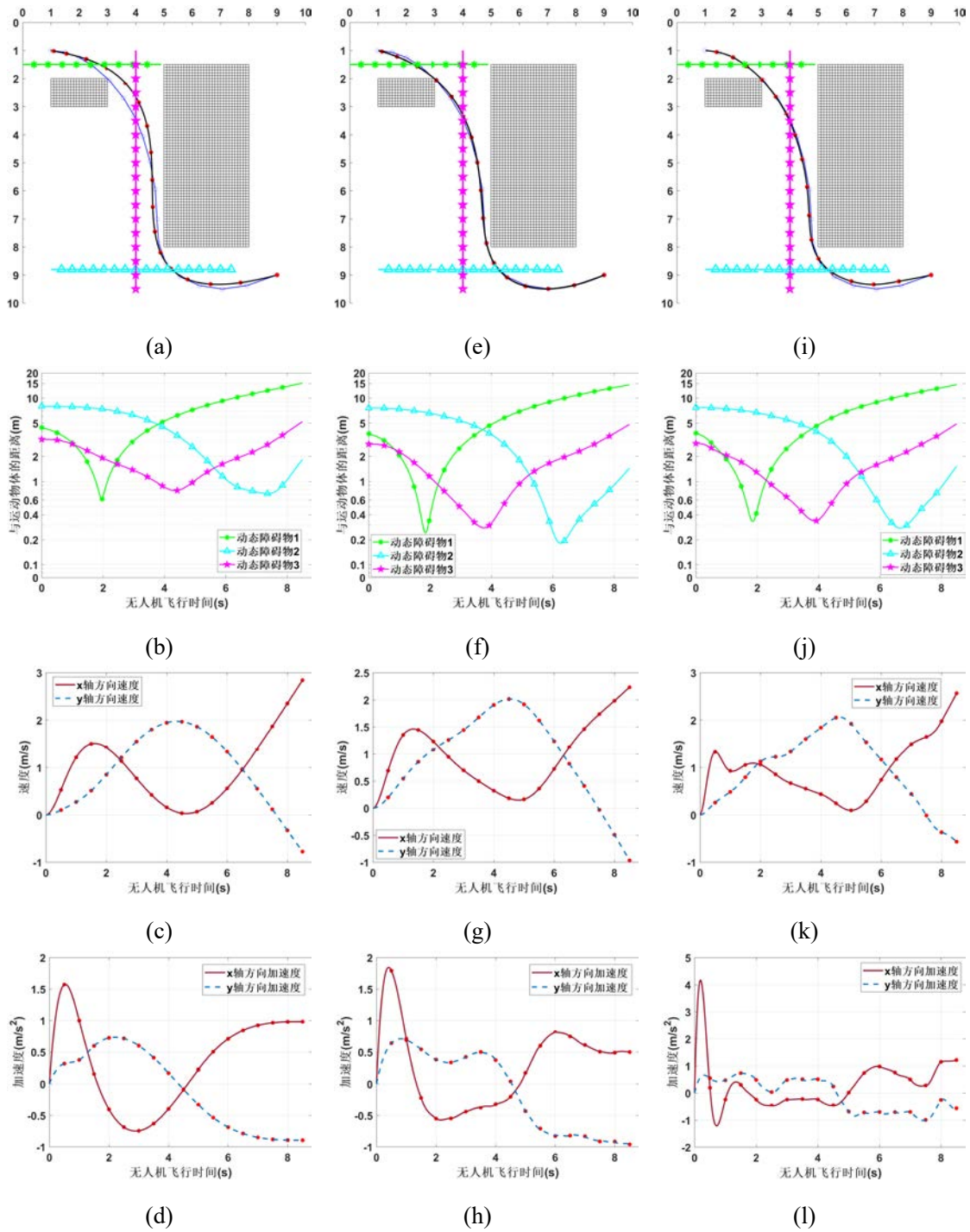


图 4-9 三种算法在地图一中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度

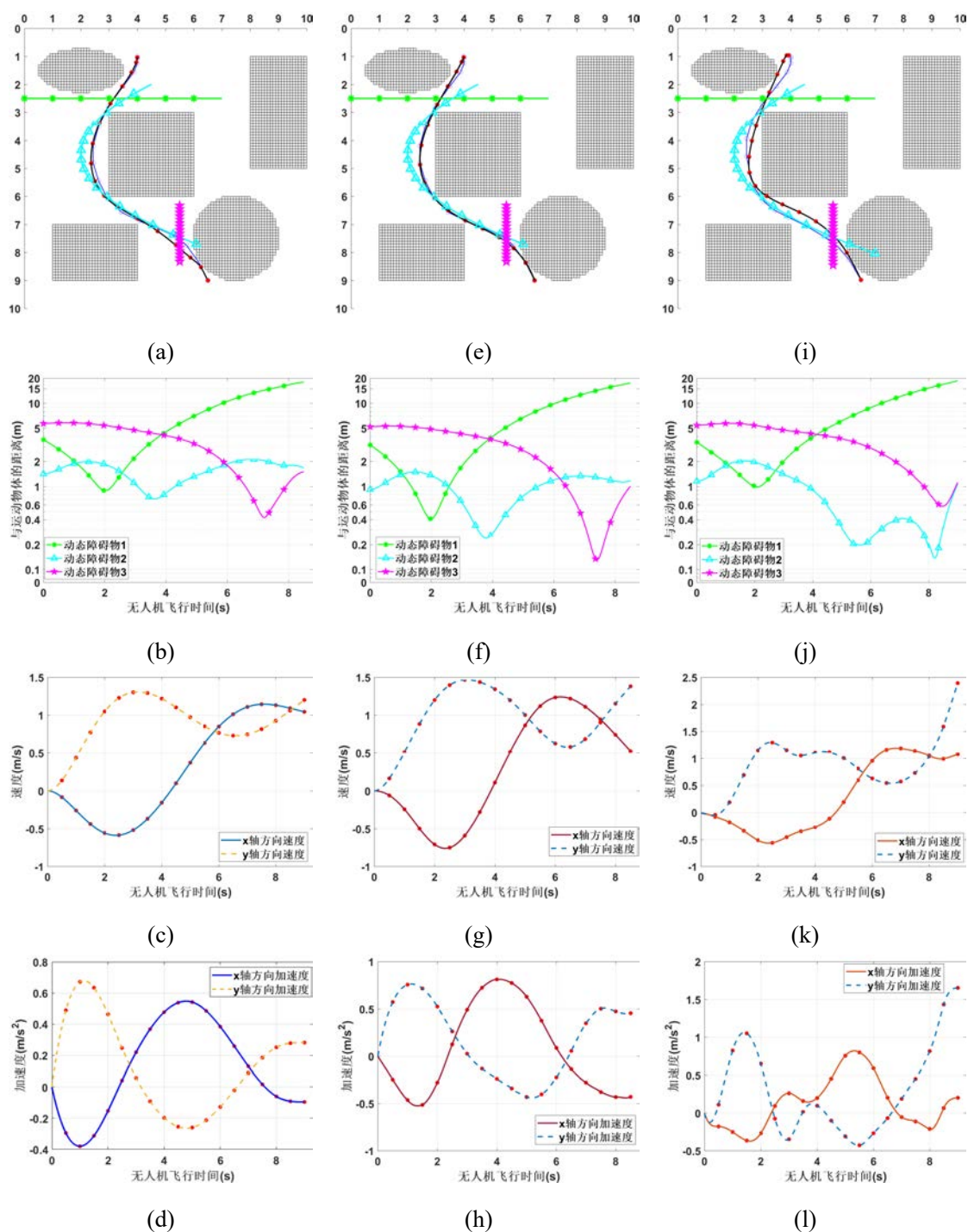


图 4-10 三种算法在地图二中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度

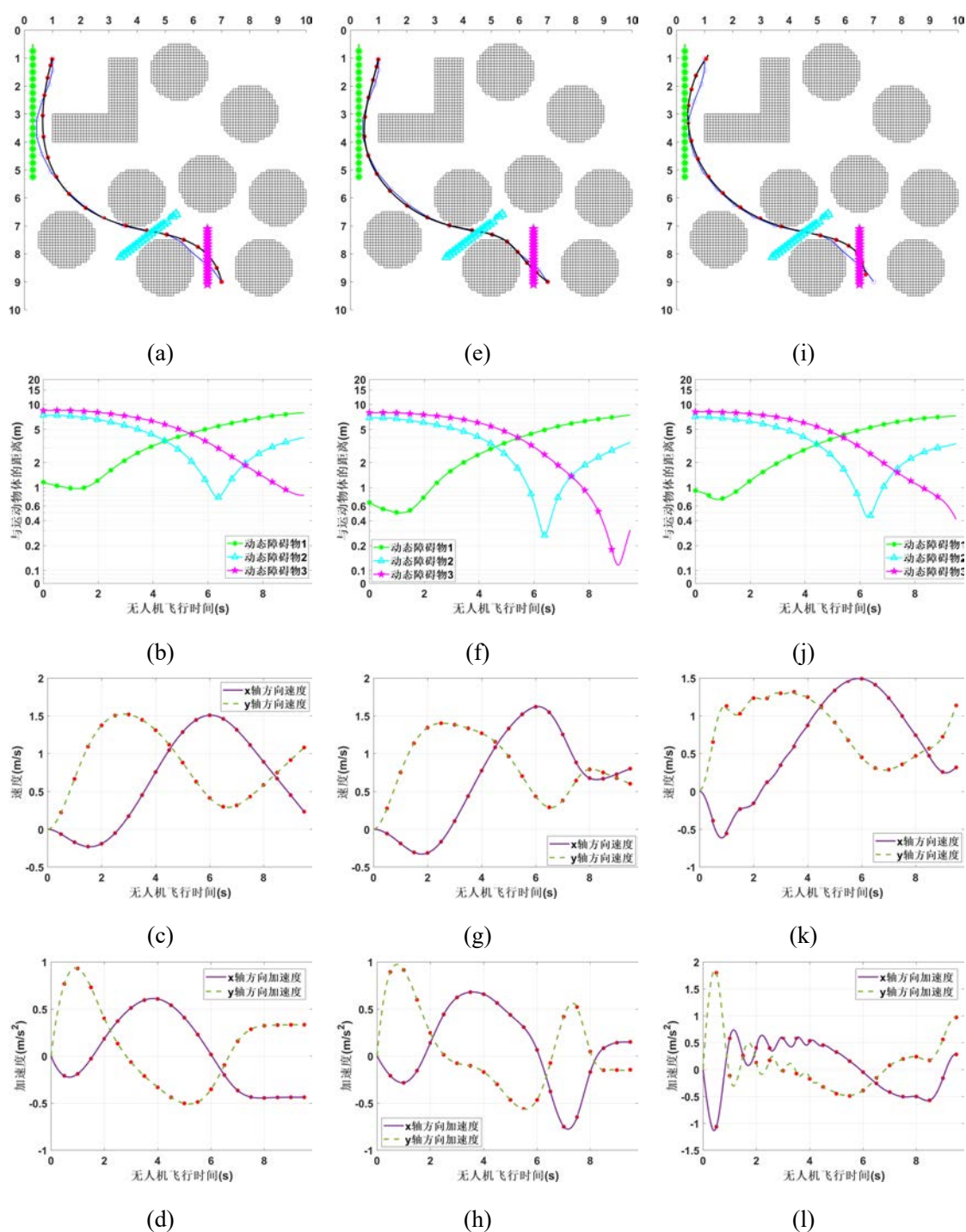


图 4-11 三种算法在地图三中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度



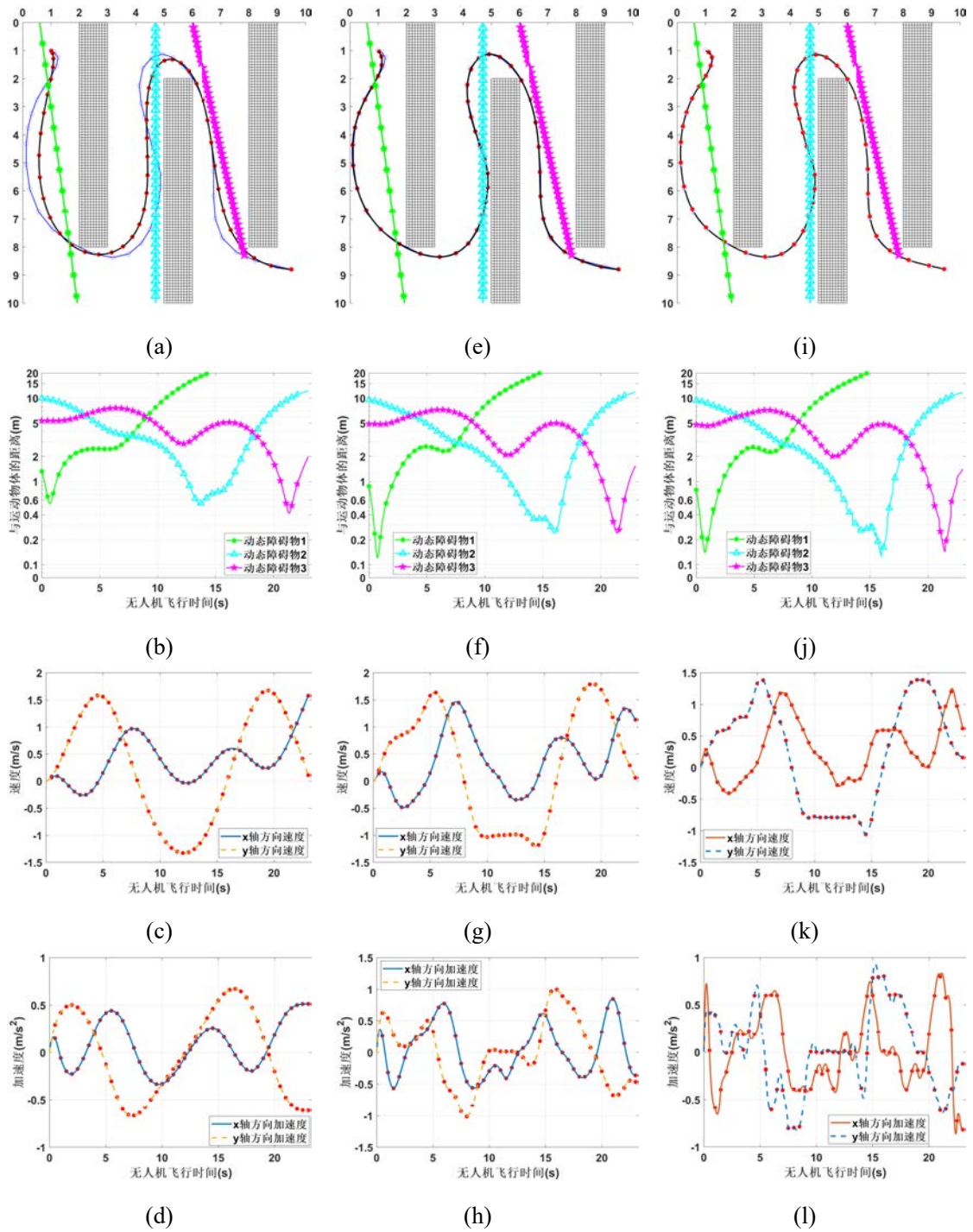


图 4-12 三种算法在地图四中生成的避障轨迹效果。(a)(b)(c)(d): 依次表示避障轨迹优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度和加速度; (e)(f)(g)(h): 依次表示 SDP 松弛规划算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度; (i)(j)(k)(l): 依次表示无约束优化算法计算的轨迹、轨迹与障碍物的距离、轨迹速度与加速度

表 4-1 两种避障轨迹规划算法的轨迹计算时间

	地图一	地图二	地图三	地图四
避障轨迹优化算法	1.11s	1.61s	1.74s	22.37s
SDP 松弛规划算法 [40]	20.24s	19.95s	21.54s	150.80s
无约束优化算法 [53]	15.93s	16.25s	25.66s	132.68s

表 4-2 障碍物数量不同时两种算法的轨迹计算时间

运动障碍物数量	3	6	9	12
避障轨迹优化算法	2.02s	2.11s	2.23s	2.99s
SDP 松弛规划算法 [40]	15.64s	28.33s	50.64s	75.47s
无约束优化算法 [53]	15.64s	33.25s	57.96s	90.53s

表4-1展示了三种算法在四个地图场景中的轨迹计算时间。从表中可以看出，本论文提出的算法的计算时间更短，相比两种对比算法，轨迹计算速度平均有 10 倍的提升。表4-2 展示了三种算法在不同障碍物数目情况下的轨迹计算时间，该实验所使用的场景是地图一，无人机从 (1,1) 飞到 (9,9)。从表中可以看出，随着要躲避的运动障碍物数量的增加，本论文提出的算法的计算时间也逐渐增加，原因在于避障约束数量增加，但计算时间仍旧维持在一个较小的数量级；反观两个对比算法，其轨迹计算时间随障碍物数量增加而大幅延长，这无法实现在动态环境中实时运行。从一系列实验的结果可以看出，本论文提出的轨迹规划算法能够在多种复杂环境中快速规划出一条安全顺滑、符合动力学可行性的飞行轨迹，且该算法具有较好的鲁棒性。

## 4.5 本章小结

本章针对动态环境中已知障碍物情况下的避障轨迹优化问题，提出了一种障碍物避障建模与轨迹优化算法。该算法首先根据动态障碍物与飞行走廊的相对位置将其分为三类，对这三类动态障碍物分别建模处理，得到瞬时边界约束与区间边界约束。得到动态障碍物避障约束后，本章使用 MINVO 基作为无人机多项式轨迹的基函数，推导了以控制点为优化变量的代价函数、起终点状态约束、轨迹连续性约束、瞬时边界约束、区间边界约束和动力学约束，最终轨迹优化问题建模为具有全局最优解的凸二次规划问题，在 MATLAB 中调用 quadprog() 函数求解得到最优轨迹。与已有算法在四个地图中的对比实验结果表明，算法计算的最优轨迹能够使无人机与动态障碍物保持一个相对安全的距离，同时轨迹具有平滑性与动力学可行性，且在多个地图中均发挥出良好的避障效果。

## 第五章 动态环境下紧急避障规划算法

### 5.1 引言

在轨迹优化算法计算出避障轨迹之后，无人机会跟踪执行该轨迹进行安全飞行。在大多数情况下，无人机都能与障碍物保持安全距离，顺利完成相应任务。但在某些情况下无人机会与障碍物距离过近，比如无人机传感器噪声影响与动态环境中障碍物发生变化，导致无人机感知模块建模的地图不准确、更新不及时以及缺失部分障碍物等，于是基于此地图计算的避障轨迹极有可能与实际环境中的障碍物发生碰撞。除此之外，实际无人机飞行控制系统也存在性能欠佳的问题，在跟踪执行避障轨迹时会存在误差，随着时间积累，累计误差最终会导致无人机偏离预定规划轨迹并产生与障碍物发生碰撞的风险。因此，在紧急情况下能够保证无人机安全的紧急避障算法便具有极其重要的研究价值。这样的紧急避障算法能够在局部环境中实时检测无人机与障碍物的距离，当无人机与障碍物具有较大碰撞风险时，紧急控制无人机远离障碍物，从而大幅提升无人机飞行的整体安全性。

现有的无人机紧急避障算法存在抖动问题并且会陷入局部极小值，这都会影响无人机在复杂场景中的避障飞行能力。本章提出一种动态环境下无人机紧急避障规划算法，其作为上一章避障轨迹优化算法的补充部分，当检测到无人机距离障碍物较近时，计算低阶平滑的避障轨迹控制无人机快速远离障碍物。本章首先利用测距模块与无人机自身运动扫描障碍物并将障碍物表面建模为墙面，并为其设计触发条件与状态切换流程。然后针对无人机单个墙面与多个墙面的避障场景设计出两种避障策略与运动轨迹。最后，搭建仿真环境并设计实验来验证算法有效性与鲁棒性。

### 5.2 紧急避障规划算法设计

本论文考虑在无人机的左侧、右侧和后方三个方向分别安装三个测距模块，在无人机正面由深度相机传感器负责观测，三个测距模块分别用来测量无人机三个方向上与障碍物的距离，测距模块是指可以测量无人机与障碍物距离的传感器，常用的有红外传感器、激光传感器和超声波传感器等。该算法的具体内容包括紧急避障算法整体流程、单墙面轨迹规划与多墙面轨迹规划，接下来一一进行阐述。

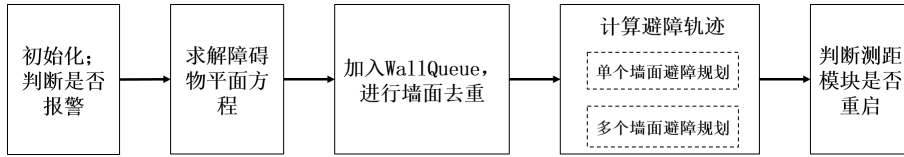


图 5-1 紧急避障规划算法整体流程

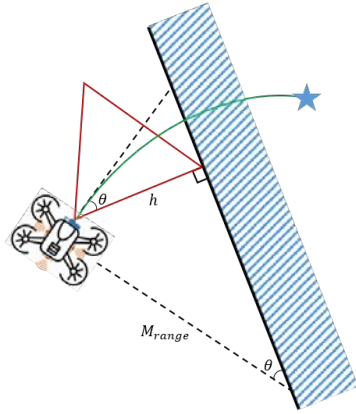


图 5-2  $M_{range}$  的计算原理

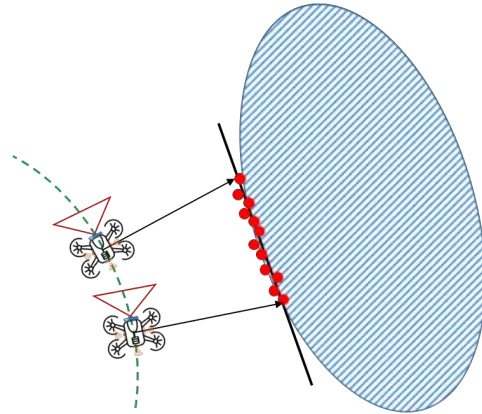


图 5-3 测距数据拟合墙面

### 5.2.1 算法整体流程

这一部分主要讲述如何使用测距模块的测量数据来建模障碍物、触发避障以及使用状态切换的算法整体过程，算法整体流程如图5-1所示，具体内容分为以下几步：

步骤一：首先为测距模块的测量数据设置一个报警阈值  $M_{range}$ ，给每个测距模块设置一个标识  $avaiFlag$ ，用来标记是否使用该模块，该标识默认为 1，表示使用该模块，当该标识为 0 时，则表示禁用当前测距模块。算法首先获得测距模块的测距数据，若测距数据小于该阈值并且此时  $avaiFlag = 1$ ，则触发报警，同时将  $avaiFlag$  置零，转入步骤二；若前述条件不满足，则忽略当前测距数据，当前流程终止。 $M_{range}$  的具体计算原理如图5-2所示，

假设无人机摄像头视场角为  $\theta$ ，当视场角边界与障碍物表面垂直时，测距模块的测量数据处于报警的临界值处，即小于这个临界值，无人机便距离障碍物过近，需要进行躲避。根据几何关系可得：

$$M_{range} = \frac{h}{\sin \theta} \quad (5-1)$$

其中， $h$  表示无人机如果接下来在垂直墙面方向以最大加速度  $a_{max}$  做匀减速运动时所飞过的距离，当无人机在垂直于墙面方向的分速度为最大速度  $v_{max}$  时，便可

求得:

$$h = \frac{v_{max}^2}{2a_{max}} \quad (5-2)$$

同时, 由于接下来无人机需要采集测距模块在一段时间  $\Delta t$  内的测量数据, 因此要将这段时间里的运动距离考虑在内, 于是得到  $M_{range}$  的最终计算公式如下:

$$M_{range} = \frac{v_{max}^2}{2a_{max} \sin \theta} + v_{max} \Delta t \quad (5-3)$$

步骤二: 当步骤一中的测距模块报警之后, 算法会记录该模块在接下来  $\Delta t$  时间内的扫描测量数据, 结合无人机当前状态信息解算出对应的障碍物表面点的坐标, 并将其保存到一个 **buffer** 中。保存完成后, 计算这个 **buffer** 中数据的平均值, 若平均值大于  $M_{range}$ , 说明无人机未靠近障碍物, 则报警解除, 但保持此时的 *avaiFlag* 不变; 若平均值小于  $M_{range}$ , 如图5-3所示, 使用最小二乘法进行平面拟合, 得到障碍物平面方程, 本论文将此平面当作阻挡无人机的障碍物的表面, 在二维地图上, 该墙面方程是一个直线方程, 然后转入步骤三。

步骤三: 位于无人机左边、右边和后边的三个测距模块会并行执行步骤一与步骤二, 从步骤二得到墙面后, 将其加入 *WallQueue* 队列中, 该队列是一个 **FIFO** 队列, 该队列中存放着目前会阻挡无人机飞行的障碍物墙面方程。接下来对新加入队列的墙面进行去重处理, 具体操作为: 计算新加入的墙面与队列中已有墙面之间的夹角, 若计算得到夹角小于  $30^\circ$ , 则认为新加入的墙面与旧墙面属于同一个障碍物表面, 从而将该新墙面从队列中剔除, 当前流程终止, 去重操作可以去除多余建模的障碍物墙面, 增加算法稳定性; 否则, 转入步骤四。除此之外, 在无人机飞行过程中, *WallQueue* 会根据无人机当前位置来计算无人机距离队列中墙面的距离, 若距离大于无人机与该墙面初始距离, 说明无人机已经成功躲避并远离该墙面, 则将该墙面方程从队列中删除。

步骤四: 算法会从 *WallQueue* 中取出需要躲避的墙面, 计算避障轨迹给无人机执行, 避障轨迹计算的具体内容在下一节进行详细讲述。

步骤五: 在无人机执行指定轨迹过程中, 同步执行该步骤。该步骤的作用是重启之前被禁用的测距模块。具体方法是维护一个 **minbuffer**, 用来维护测距模块的日常观测数据, 当 **minbuffer** 中的值大于  $M_{range}$ , 并且此时 *avaiFlag* = 0 的话, 将 *avaiFlag* 置 1, 表示该模块已经离开障碍物墙面范围, 可以重新使用了。

### 5.2.2 单个墙面避障规划

这一节主要讲述如何计算用于躲避单个障碍物墙面的避障轨迹, 具体包含两个步骤。

步骤一：判断无人机当前运动方向是否远离墙面。具体判断方法是采样无人机沿着当前运动方向到达的下一个位置点，分别计算当前位置与下一个位置到墙面的距离，若距离较小，则靠近墙面；否则就远离墙面。当无人机远离墙面时，说明障碍物对无人机运动没有威胁，当前流程终止；当无人机靠近墙面时，则需计算避障轨迹给无人机执行。

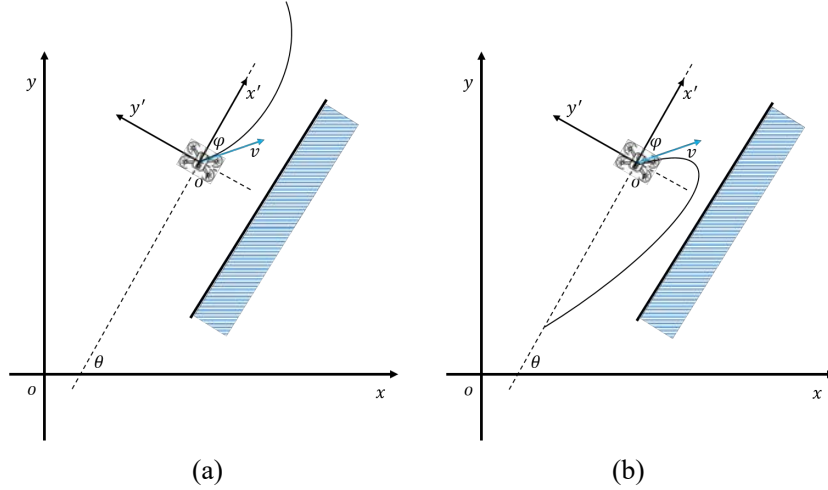


图 5-4 两种避障运动轨迹原理。(a) 第一种运动轨迹；(b) 第二种运动轨迹

步骤二：计算避障轨迹。在躲避一个墙面时，需要将要躲避的墙面作为一个硬约束，将无人机当前运动状态作为初始条件，计算得到安全的二阶运动轨迹。在这里，本论文设计了两种运动轨迹。图5-4(a)所示的是第一种运动轨迹，该轨迹要求无人机在  $y'$  轴方向做匀减速运动，在  $x'$  轴方向做匀速运动。其具体计算过程为：首先以无人机当前位置为机体坐标系原点，分别以平行于墙面的方向和垂直于墙面的方向做为机体坐标系的  $x'$  轴和  $y'$  轴， $\varphi$  表示当前无人机速度方向与  $x'$  轴正方向的夹角， $\theta$  表示墙面方向与地面坐标系  $x$  轴正方向的夹角，于是可得机体坐标系的方向向量为：

$$\begin{cases} \vec{o'x'} = (\cos \theta, \sin \theta) \\ \vec{o'y'} = (-\sin \theta, \cos \theta) \end{cases} \quad (5-4)$$

已知无人机在地面坐标系下的速度  $v$ ，则该速度在机体坐标系上的分量为：

$$\begin{cases} v_{x'} = v \cdot \vec{o'x'} \\ v_{y'} = v \cdot \vec{o'y'} \end{cases} \quad (5-5)$$

在  $y'$  轴方向所需加速度为：

$$a_{y'} = a_{max} \cdot (-1) \cdot \text{sign}(v_{y'}) \quad (5-6)$$

所以, 第一种轨迹在机体坐标系下的轨迹方程为:

$$\begin{cases} x' = v_{x'} \cdot t \\ y' = v_{y'} \cdot t + \frac{1}{2} a_{y'} t^2 \end{cases}, \quad t \in [0, t_{max}], \quad t_{max} = \frac{2 \|v_{y'}\|}{a_{max}} \quad (5-7)$$

设无人机在地面坐标系中的当前位置为  $[p_x, p_y]^T$ , 则机体坐标系与地面坐标系之间的转换关系为:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \quad (5-8)$$

因此第一种运动轨迹在地面坐标系下的方程为:

$$\begin{cases} x(t) = -\frac{\sin \theta}{2} a_{y'} t^2 + (\cos \theta \cdot v_{x'} - \sin \theta \cdot v_{y'}) t + p_x \\ y(t) = \frac{\cos \theta}{2} a_{y'} t^2 + (\sin \theta \cdot v_{x'} + \cos \theta \cdot v_{y'}) t + p_y \end{cases} \quad (5-9)$$

当只有一个墙面时, 往往采用第一种避障轨迹, 因为其运动形式简单, 运动幅度较小, 无人机较容易执行。

图5-4(b)所示的是第二种运动轨迹, 第二种运动轨迹往往用在无人机躲避多个墙面的情况下, 能够保证无人机不会陷入障碍物死角。在这种轨迹中, 无人机在  $x'$  轴方向做匀减速运动, 等无人机在该方向的分速度等大反向之后, 再做匀速运动, 而在  $y'$  轴方向一直做匀减速运动, 本论文设在  $x'$  轴方向的加速度为  $a_1$ , 在  $y'$  轴方向的加速度为  $a_2$ 。从上面描述可知, 第二种运动轨迹根据  $x$  轴的运动情况可以分为两个阶段, 设第一个阶段的飞行时间为  $t_1$ , 第二个阶段的飞行时间为  $t_2$ , 可以得到关于  $a_1$ 、 $a_2$ 、 $t_1$ 、 $t_2$  的关系式如下:

$$\begin{cases} a_1^2 + a_2^2 = a_{max}^2 \\ \frac{2|v_{y'}|}{a_2} = t_1 + t_2 \\ \frac{2|v_{x'}|}{a_1} = t_1 \end{cases} \quad (5-10)$$

同时, 容易发现:

$$\begin{cases} \frac{|v_{y'}|^2}{2a_2} < h = \frac{v_{max}^2}{2a_{max}} \\ t_1 + t_2 > t_1 \end{cases} \quad (5-11)$$

对上式进一步推导得到:

$$\frac{|v_{y'}|}{2a_2} < \frac{v_{max}^2}{2a_{max}} \Rightarrow a_2 > \frac{v_{y'}^2}{v_{max}^2} \cdot a_{max} \quad (5-12)$$

$$t_1 + t_2 > t_1 \Rightarrow \frac{2|v_{y'}|}{a_2} > \frac{2|v_{x'}|}{a_1} \Rightarrow \frac{a_2}{a_1} < \left| \frac{v_{y'}}{v_{x'}} \right| \quad (5-13)$$

根据式5-10.1，式5-13可进一步化简为：

$$a_2 < \left| \frac{v_{y'}}{v} \right| \cdot a_{max} \quad (5-14)$$

根据式5-12和式5-14得：

$$\frac{v_{y'}^2}{v_{max}^2} \cdot a_{max} < a_2 < \left| \frac{v_{y'}}{v} \right| \cdot a_{max} \quad (5-15)$$

于是，本论文令：

$$a_2 = \frac{1}{2} \left( \left| \frac{v_{y'}}{v} \right| + \frac{v_{y'}^2}{v_{max}^2} \right) \cdot a_{max} \quad , \quad a_1 = \sqrt{a_{max}^2 - a_2^2} \quad (5-16)$$

计算出加速度之后，第二种运动轨迹在机体坐标系下得轨迹方程为：

$$\begin{aligned} x' &= \begin{cases} v_{x'}t + \frac{1}{2}a_1t^2, t \in [0, t_1] \\ -v_{x'}(t - t_1), t \in [t_1, t_1 + t_2] \end{cases} \\ y' &= v_{y'}t + \frac{1}{2}a_2t^2, t \in [0, t_1 + t_2] \end{aligned} \quad (5-17)$$

在地面坐标系下的轨迹方程为：

$$\begin{aligned} x(t) &= \begin{cases} p_x + (\cos \theta \cdot v_{x'} - \sin \theta \cdot v_{y'})t + (\frac{\cos \theta}{2}a_1 - \frac{\sin \theta}{2}a_2)t^2, t \in [0, t_1] \\ (p_x + \cos \theta \cdot v_{x'} \cdot t_1) - (\cos \theta \cdot v_{x'} + \sin \theta \cdot v_{y'})t - \frac{\sin \theta}{2}a_2t^2, t \in [t_1, t_1 + t_2] \end{cases} \\ y(t) &= \begin{cases} p_y + (\sin \theta \cdot v_{x'} + \cos \theta \cdot v_{y'})t + (\frac{\sin \theta}{2}a_1 + \frac{\cos \theta}{2}a_2)t^2, t \in [0, t_1] \\ (p_y + \sin \theta \cdot v_{x'} \cdot t_1) + (\cos \theta \cdot v_{y'} - \sin \theta \cdot v_{x'})t + \frac{\cos \theta}{2}a_2t^2, t \in [t_1, t_1 + t_2] \end{cases} \end{aligned} \quad (5-18)$$

### 5.2.3 多个墙面避障规划

在无人机运动过程中，经常会出现多个测距模块同时报警并生成“墙面”，此时就需要无人机对多个墙面进行紧急避障规划，本论文将存储墙面方程的数据结构 *WallQueue* 定义为一个先入先出 (First In First Out, FIFO) 队列，经过分析，本论文发现对多个墙面的轨迹规划可以简化为对两个墙面的轨迹规划，整体避障过程往往是无人机在执行对旧墙面的避障轨迹过程中又发现了新墙面，然后需要对这两个墙面都进行躲避。因此，实现对两个墙面的轨迹规划包含两个步骤，具体内容如下所述。

步骤一：首先判断无人机当前运动方向相对于新加入 *WallQueue* 的墙面的关



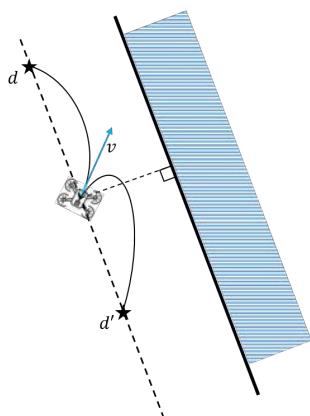


图 5-5 两种避障轨迹的终点位置

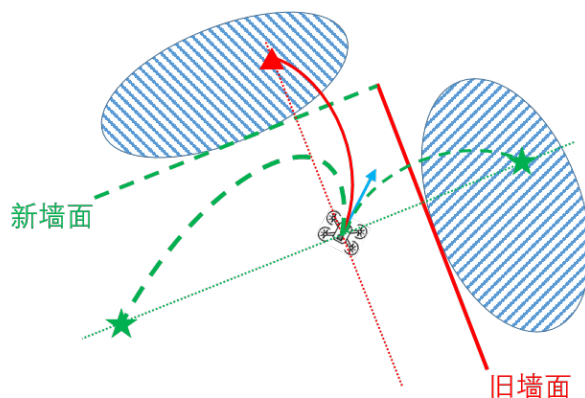


图 5-6 选择新墙面避障轨迹的原理示意图

系，即无人机是靠近墙面还是远离墙面。判断方法是采样无人机沿着当前运动方向到达的下一个位置点，分别计算当前位置与下一个位置到墙面的距离，若距离较小，则靠近墙面；否则就远离墙面。经过上述判断若发现无人机远离新墙面，说明无人机不会与新墙面发生碰撞，则放弃重新规划轨迹，继续执行对旧墙面的避障轨迹；若发现无人机靠近新墙面，则需要重新规划轨迹。

步骤二：根据两种候选轨迹的终点位置与旧墙面的几何关系来选择使用哪种轨迹。本论文在 5.2.2 节计算得到了两条候选轨迹，如图5-5所示，这两条候选轨迹的终点位置分为：

$$d = \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} t_{max} \cdot \cos \theta \cdot v_{x'} \\ t_{max} \cdot \sin \theta \cdot v_{x'} \end{bmatrix}, \quad d' = \begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} t_2 \cdot \cos \theta \cdot v_{x'} \\ t_2 \cdot \sin \theta \cdot v_{x'} \end{bmatrix} \quad (5-19)$$

当无人机检测到新墙面时，算法首先在在前一部分中判断是否需要对其规划避障轨迹，然后本部分根据  $d$  与  $d'$  相对于旧墙面的远近关系，来选择合适的运动轨迹对新墙面进行躲避，同时保证无人机不会与旧墙面发生碰撞。具体如图5-6所示，红色实线表示旧墙面，绿色虚线表示新墙面，无人机当前正在执行对实线墙面的避障轨迹，此时传感器感知获得一个新墙面并经过判断发现需要进行躲避。于是先计算出两个候选轨迹的终点位置，即图中的两个五角星，然后判断终点位置是否在旧墙面内，判断方法是计算两个终点位置与无人机当前位置是否在新墙面方程的同一区域内，若有一个在旧墙面内，则选择另一个终点位置对应的避障轨迹；若都不在旧墙面内，则选择距离旧墙面距离更远的那个终点位置对应的避障轨迹，即图中加粗绿色虚线对应的轨迹，交给无人机执行。

### 5.3 仿真实验与结果分析

本节在 MATLAB 中搭建二维仿真环境，并分别编程实现了本论文提出的紧急避障规划算法与论文 [54] 提出的改进人工势场法，最后设计仿真实验在多种地图中来检验算法效果。论文 [54] 在设计人工势场时引入了环绕障碍物的旋度势场以解决局部最小值与抖动问题。

实验中使用的算法相关参数设置为：无人机视场角  $\theta = 60^\circ$ ，无人机最大速度  $v_{max} = 2m/s$ ，最大加速度  $a_{max} = 2m/s^2$ ，采样时间  $\Delta t = 0.5s$ ，斥力有效范围  $d_0 = 0.15m$ 。

首先在三个简单环境中检验紧急避障规划算法对单个墙面的避障效果，如图5-7所示，红色轨迹表示无人机正常飞行轨迹，蓝色轨迹表示无人机执行紧急避障规划算法计算的避障轨迹，梅红色直线表示紧急避障算法建模得到的障碍物墙面。可以看到，当无人机靠近障碍物时，算法计算的轨迹会使无人机做出一个反弹的临时轨迹，能够实现无人机对障碍物的躲避远离，保证无人机安全性，也增大了无人机的路径搜索空间，有利于后续轨迹算法产生更优的轨迹。

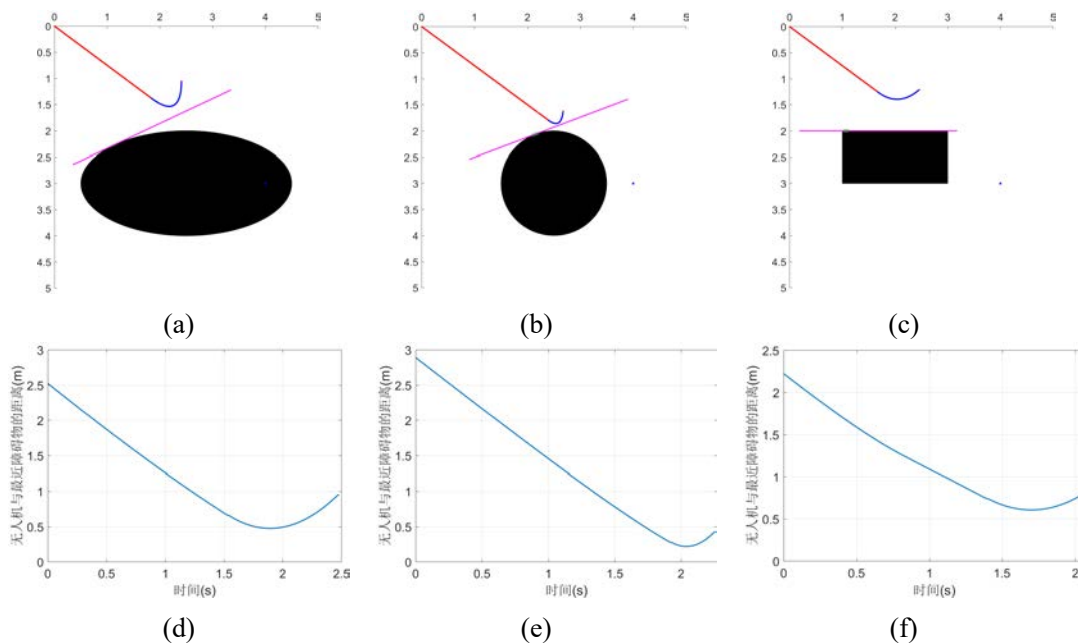


图 5-7 算法在三种简单场景下的避障效果。(a)(b)(c)：无人机在三个场景中的避障轨迹；(d)(e)(f)：无人机在三个场景中障碍物与无人机的距离

接着，本论文又设计了三个场景来验证算法对两个墙面的避障效果，避障轨迹如图5-8所示，可以看到，在面对多个障碍物形成的死角区域时，本论文提出的算法能够使无人机成功转向并跳出该区域，而改进的人工势场法依然陷入局部极小值，无法跳出该区域，阻碍后续任务执行。

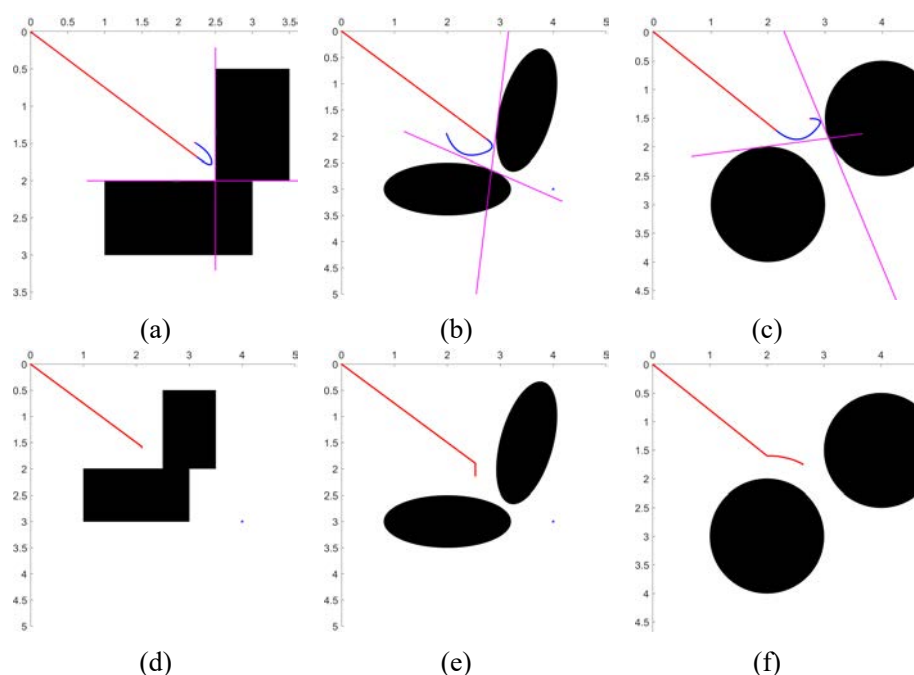


图 5-8 紧急避障规划算法与人工势场法的实验对比效果。(a)(b)(c): 本论文提出的紧急避障规划算法效果; (d)(e)(f): 人工势场法效果

在复杂地图环境中, 本论文设计了一个极端的无人机飞行策略, 即当没有障碍物时, 让无人机朝向目标点飞行; 当快碰到障碍物时, 使用本论文提出的紧急避障规划算法来躲避障碍物。本论文在实验中使用这样的飞行策略来检验算法对复杂环境的鲁棒性与有效性。实验中, 无人机起点位置位于左上角的 (0,0), 终点位置设置在 (9,9)。实验结果如图5-9所示,

在图5-9(a) 环境中, 无人机无论使用简单飞行策略还是人工势场法都安全到达目标点; 但在图5-9(b)(c) 环境中, 改进人工势场法使得无人机陷入了复杂环境带来的局部极小值的位置, 障碍物旋度势场不再发挥作用, 因此无人机无法进一步前进。而采用了紧急避障规划算法的无人机, 最终都顺利抵达目标点。因此可以看出, 在复杂环境中, 无人机即使使用最为简单的飞行策略, 本论文设计的多测距模块同时感知避障与多障碍物避障的算法, 能够使无人机适应更复杂的环境, 保证无人机飞行安全, 不会陷入障碍物构成的死角。

在图5-10实验中, 无人机飞行策略为沿着绿色虚线构成的折线路径飞行, 若遇到障碍物, 则使用紧急避障规划算法进行躲避。从实验结果可以看出, 飞行过程中紧急避障规划算法控制无人机顺利躲开障碍物, 而人工势场法再次陷入局部极小值位置。

通过上述实验结果可知, 人工势场法的成功率在 40%, 本论文提出的紧急避障规划算法的成功率为 90%, 因此, 该算法在多种环境中具有较好的鲁棒性。

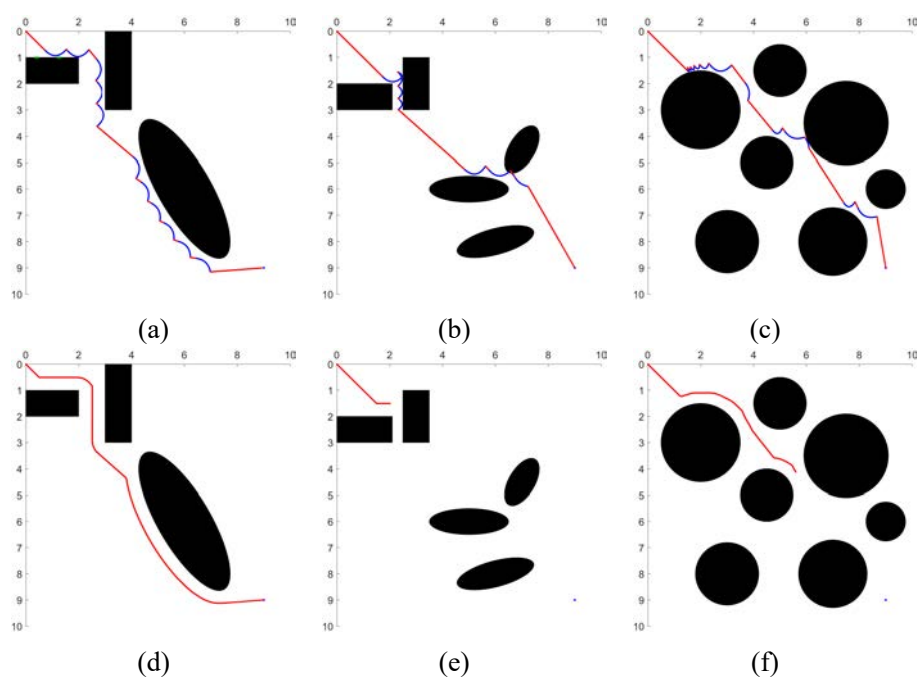


图 5-9 两种算法在复杂环境下的避障效果。(a)(b)(c): 紧急避障规划算法效果; (d)(e)(f): 人工势场法效果

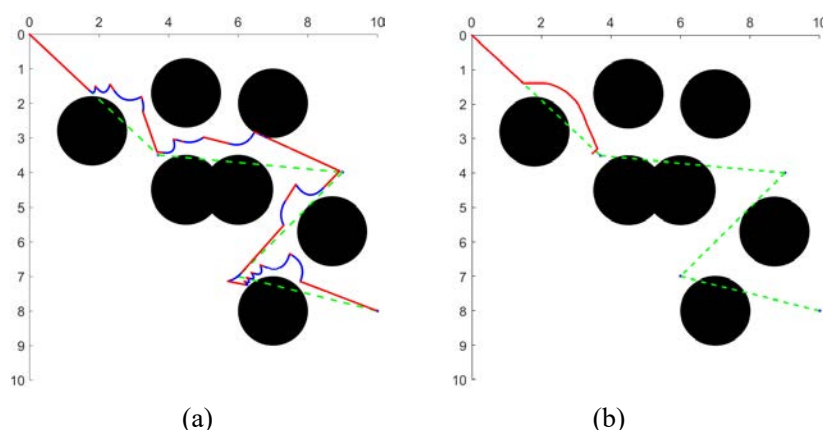


图 5-10 两种算法在沿折线路径运动的避障效果。(a) 紧急避障规划算法的效果; (b) 人工势场法的效果

## 5.4 本章小结

本章针对无人机在动态环境下紧急躲避障碍物的问题设计了一种动态环境下紧急避障规划算法，解决了现有算法存在的抖动和陷入局部极小值的不足。本章充分挖掘利用测距模块的测距能力和无人机自身的运动实现对障碍物的扫描建模，并设计了单个墙面与多个墙面的避障策略，设计了两种二阶的平滑避障轨迹，减小因轨迹突变带来的机械损坏。最后搭建仿真环境并设计实验来验证新提出的紧

急避障规划算法的有效性与在不同环境中鲁棒性，该避障算法能够使无人机在靠近障碍物时做出一个反弹动作，实现对物体的躲避远离，不会陷入障碍物构成的死角，为后续飞行提供更高安全性和更大运动空间。

## 第六章 全文总结与展望

### 6.1 全文工作总结

本论文针对无人机在包含静态障碍物与动态障碍物的复杂环境中的安全飞行问题进行了充分研究，在运动规划层面开展多项研究工作，综合运用几何理论、运筹学理论在路径搜索、飞行走廊生成、轨迹规划和紧急避障规划四个运动规划子层面针对现有的问题提出了具备创新性的解决办法。

在路径搜索方面，针对现有启发式函数未考虑环境中障碍物存在的问题，设计了考虑障碍物存在启发式函数，引入膨胀系数的概念，并设计了自适应策略获得具体地图对应的最佳膨胀系数。考虑到环境中的运动物体，本论文在搜索算法中添加了动态障碍物剪枝功能并设计出用于观测运动目标的遮挡惩罚项。实验证明这些创新性工作大幅提升了搜索效率，且提升了算法在不同任务场景中的适应性。

在飞行走廊生成方面，针对 **Bresenham** 画线算法的准确性与覆盖率较低以及现有飞行走廊生成算法速度慢的问题，设计出一种基于端点任意直线栅格计算方法的飞行走廊生成算法，首先给出了端点任意直线的栅格计算方法的判别项与递推公式，然后阐述飞行走廊生成算法的具体内容。实验结果证明端点任意直线的栅格计算方法对直线的覆盖率更高；飞行走廊生成算法能够在多种地图环境中快速生成有效安全的飞行走廊，为后续最优轨迹求解提供了有利条件。

在轨迹规划方面，运用几何理论设计出一种新的动态障碍物避障建模算法，根据动态障碍物与飞行走廊的相对位置将其分为三类来分别建模处理，并使用 **MINVO** 基函数表示无人机轨迹，建模出了以控制点为优化变量的包含动态避障约束的二次规划问题，最后在 **MATLAB** 中使用 **quadprog()** 函数得到全局最优解，实验结果表明该算法得到的最优轨迹能够保证无人机避开动态障碍物，同时保证了轨迹平滑性与较小的飞行代价，该算法在多个地图中均发挥出可靠的避障效果，具有较好的鲁棒性。

在紧急避障方面，本论文针对现有算法存在抖动与局部极小值的问题设计了一种紧急避障规划算法，充分挖掘利用测距模块的单点测距能力和无人机自身的运动实现对障碍物的扫描建模，并设计了两种更贴合无人机运动学特性的二阶的顺滑避障轨迹。实验结果表明，该避障算法能够使无人机在靠近障碍物时快速躲避远离，不会陷入障碍物构成的死角，为后续飞行提供更高安全性和更大运动空间。

## 6.2 未来研究展望

随着无人机技术的进步，无人机在诸多领域中发挥出越来越大的价值，无人机面临的环境也越来越复杂，因而如何实现无人机在复杂环境下的安全飞行成为一个备受关注的问题，本论文在运动规划层面开展了多项创新性工作，提升了无人机在动态复杂环境下的避障飞行能力。但在无人机在复杂环境中执行任务过程中，仍存在较多有价值的问题尚未解决，本论文接下来对一些值得进一步探索的方向进行阐述。

### （1）动态障碍物危险评估与筛选

无人机执行任务的一些复杂环境中会存在较多的运动物体，目前已有的无人机动态避障算法会对环境中感知到的所有运动物体考虑在内，即在建模优化问题过程中会添加对所有运动物体的避障约束。运动物体较多时，对应的避障约束数量较大，导致优化问题规模较大，求解过程复杂，消耗时间过长，甚至会出现无解的情况。可以设计一个评估动态障碍物对无人机飞行威胁程度的算法，根据无人机与动态障碍物的相对运动情况判断动态障碍物是否会影响无人机未来运动，然后根据评估结果设计一个筛选策略，筛选出那些真正威胁无人机飞行的运动物体。这可以极大减少无人机需要观测与躲避的动态障碍物数量，降低优化问题计算难度，实现更精准高效的避障。

### （2）基于真实物理引擎的无人机仿真平台

真实任务环境复杂，影响因素较多，设计的无人机相关算法如果直接在真实无人机上运行，容易发生安全事故导致昂贵的真实无人机坠毁，这会造成较大的经济损失。因此需要在仿真平台上的检验算法有效性，MATLAB 这类仿真平台只适用于数值仿真实验，无法模拟无人机真实飞行环境，只能验证算法的理论可行性，无法检验出算法在真实环境中的效果。因此，可以考虑搭建基于真实物理引擎的无人机仿真环境，开发能够直接部署在真实无人机上的算法软件，既能验证算法在真实物理环境中的效果，又能避免因实机飞行失败所造成的经济损失。具体可以在 Gazebo 物理引擎中搭建仿真的任务环境与无人机模型，在 ROS 平台上开发无人机算法，并使用 PX4 开源飞控来执行算法指令，控制无人机运动。

## 参考文献

- [1] 陶于金, 李沛峰. 无人机系统发展与关键技术综述 [J]. 航空制造技术, 2014, 464(20): 34-39.
- [2] 张迷霞, 栾俊, 游于陆, et al. 无人机航空摄影测量与遥感行业发展现状与趋势 [J]. 测绘与空间地理信息, 2023, 46(1): 38-41,46.
- [3] Gupta A, Afrin T, Scully E, et al. Advances of uavs toward future transportation: The state-of-the-art, challenges, and opportunities[J]. Future transportation, 2021, 1(2): 326-350.
- [4] 张欣, 黄郑, 孟悦, et al. 电力巡检四旋翼无人机的控制研究 [J]. 自动化仪表, 2022, 43(4): 38-44.
- [5] Oh D, Han J. Smart search system of autonomous flight uavs for disaster rescue[J]. Sensors, 2021, 21(20): 6810.
- [6] Gupta S G, Ghonge D, Jawandhiya P M, et al. Review of unmanned aircraft system (uas)[J]. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume, 2013, 2(12): 1646-1658.
- [7] Chung S J, Paranjape A A, Dames P, et al. A survey on aerial swarm robotics[J]. IEEE Transactions on Robotics, 2018, 34(4): 837-855.
- [8] Bekey G A, Ambrose R, Kumar V, et al. Robotics: state of the art and future challenges[M]. Hackensack, NJ: World Scientific, 2008, 18-23.
- [9] Quan L, Han L, Zhou B, et al. Survey of uav motion planning[J]. IET Cyber-systems and Robotics, 2020, 2(1): 14-21.
- [10] Guivant J, Nebot E, Nieto J, et al. Navigation and mapping in large unstructured environments[J]. The International Journal of Robotics Research, 2004, 23(4-5): 449-472.
- [11] Yan F, Liu Y S, Xiao J Z. Path planning in complex 3d environments using a probabilistic roadmap method[J]. International Journal of Automation and computing, 2013, 10: 525-533.
- [12] Lavalley S M. Rapidly-exploring random trees : A new tool for path planning[J]. Computer ence Dept. Oct, 1998, 98.
- [13] Karaman S, Frazzoli E. Incremental sampling-based algorithms for optimal motion planning[J]. Robotics Science and Systems VI, 2010, 104(2).
- [14] Gammell J D, Srinivasa S S, Barfoot T D. Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic[C]. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, 2014: 2997-3004.



- [15] Musliman I A, Rahman A A, Coors V, et al. Implementing 3d network analysis in 3d-gis[J]. Int. Arch. ISPRS, 2008, 37: 913-918.
- [16] De Filippis L, Guglieri G, Quagliotti F. Path planning strategies for uavs in 3d environments[J]. Journal of Intelligent & Robotic Systems, 2012, 65: 247-264.
- [17] Harabor D, Grastien A. Online graph pruning for pathfinding on grid maps[C]. Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, 2011: 1114-1119.
- [18] Pivtoraiko M, Kelly A. Kinodynamic motion planning with state lattice motion primitives[C]. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, 2011: 2172-2179.
- [19] Howard T M, Kelly A. Optimal rough terrain trajectory generation for wheeled mobile robots[J]. The International Journal of Robotics Research, 2007, 26(2): 141-166.
- [20] Dolgov D, Thrun S, Montemerlo M, et al. Path planning for autonomous vehicles in unknown semi-structured environments[J]. The international journal of robotics research, 2010, 29(5): 485-501.
- [21] Karaman S, Frazzoli E. Optimal kinodynamic motion planning using incremental sampling-based methods[C]. 49th IEEE conference on decision and control (CDC), Atlanta, 2010: 7681-7687.
- [22] Liu S, Atanasov N, Mohta K, et al. Search-based motion planning for quadrotors using linear quadratic minimum time control[C]. 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), Vancouver, 2017: 2872-2879.
- [23] Zhou B, Gao F, Wang L, et al. Robust and efficient quadrotor trajectory generation for fast autonomous flight[J]. IEEE Robotics and Automation Letters, 2019, 4(4): 3529-3536.
- [24] Mueller M W, Hehn M, D'Andrea R. A computationally efficient motion primitive for quadcopter trajectory generation[J]. IEEE transactions on robotics, 2015, 31(6): 1294-1310.
- [25] Deits R, Tedrake R. Computing large convex regions of obstacle-free space through semidefinite programming[C]. Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics, Istanbul, 2015: 109-124.
- [26] Deits R, Tedrake R. Efficient mixed-integer planning for uavs in cluttered environments[C]. 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, 2015: 42-49.

- [27] Liu S, Watterson M, Mohta K, et al. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments[J]. IEEE Robotics and Automation Letters, 2017, 2(3): 1688-1695.
- [28] Gao F, Wang L, Zhou B, et al. Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments[J]. IEEE Transactions on Robotics, 2020, 36(5): 1526-1545.
- [29] Gao F, Shen S. Online quadrotor trajectory generation and autonomous navigation on point clouds[C]. 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Lausanne, 2016: 139-146.
- [30] Gao F, Wu W, Gao W, et al. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments[J]. Journal of Field Robotics, 2019, 36(4): 710-733.
- [31] Gao F, Wang L, Wang K, et al. Optimal trajectory generation for quadrotor teach-and-repeat[J]. IEEE Robotics and Automation Letters, 2019, 4(2): 1493-1500.
- [32] Chen J, Liu T, Shen S. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments[C]. 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016: 1476-1483.
- [33] Zhou B, Zhang Y, Chen X, et al. Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning[J]. IEEE Robotics and Automation Letters, 2021, 6(2): 779-786.
- [34] Bresenham J E. Algorithm for computer control of a digital plotter[J]. IBM Systems journal, 1965, 4(1): 25-30.
- [35] Mellinger D, Kumar V. Minimum snap trajectory generation and control for quadrotors[C]. 2011 IEEE international conference on robotics and automation, Shanghai, 2011: 2520-2525.
- [36] Richter C, Bry A, Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments[C]. Robotics Research: The 16th International Symposium ISRR, Singapore, 2016: 649-666.
- [37] Gao F, Wu W, Lin Y, et al. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial[C]. 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, 2018: 344-351.
- [38] Campos-Macías L, Gómez-Gutiérrez D, Aldana-López R, et al. A hybrid method for online trajectory planning of mobile robots in cluttered environments[J]. IEEE Robotics and Automation Letters, 2017, 2(2): 935-942.

- [39] Gao F, Lin Y, Shen S. Gradient-based online safe trajectory generation for quadrotor flight in complex environments[C]. 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), Vancouver, 2017: 3681-3688.
- [40] Gao F, Shen S. Quadrotor trajectory generation in dynamic environments using semi-definite relaxation on nonconvex qcqp[C]. 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017: 6354-6361.
- [41] Wang Y, Ji J, Wang Q, et al. Autonomous flights in dynamic environments with onboard vision[C]. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021: 1966-1973.
- [42] Mellinger D, Kushleyev A, Kumar V. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams[C]. 2012 IEEE international conference on robotics and automation, Saint Paul, 2012: 477-483.
- [43] Chakravarthy A, Ghose D. Obstacle avoidance in a dynamic environment: A collision cone approach[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 1998, 28(5): 562-574.
- [44] Huang S, Teo R S H, Liu W. Distributed cooperative avoidance control for multi-unmanned aerial vehicles[J]. Actuators, 2018, 8(1): 1.
- [45] Jenie Y I, van Kampen E J, de Visser C C, et al. Three-dimensional velocity obstacle method for uncoordinated avoidance maneuvers of unmanned aerial vehicles[J]. Journal of Guidance, Control, and Dynamics, 2016, 39(10): 2312-2323.
- [46] Snape J, Van Den Berg J, Guy S J, et al. The hybrid reciprocal velocity obstacle[J]. IEEE Transactions on Robotics, 2011, 27(4): 696-706.
- [47] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots[J]. The international journal of robotics research, 1986, 5(1): 90-98.
- [48] Borenstein J, Koren Y. Real-time obstacle avoidance for fast mobile robots[J]. IEEE Transactions on systems, Man, and Cybernetics, 1989, 19(5): 1179-1187.
- [49] Panati S, Baasandorj B, Chong K T. Autonomous mobile robot navigation using harmonic potential field[J]. IOP Conference Series: Materials Science and Engineering, 2015, 83(1): 012018.
- [50] Bertsekas D. Dynamic programming and optimal control: Volume i[M]. Nashua: Athena scientific, 2012, 435-449.

- [51] Pan N, Zhang R, Yang T, et al. Fast-tracker 2.0: Improving autonomy of aerial tracking with active vision and human location regression[J]. IET Cyber-Systems and Robotics, 2021, 3(4): 292-301.
- [52] Tordesillas J, How J P. Minvo basis: Finding simplexes with minimum volume enclosing polynomial curves[J]. Computer-Aided Design, 2022, 151: 103341.
- [53] Zhou X, Zhu J, Zhou H, et al. Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments[C]. 2021 IEEE International Conference on Robotics and Automation (ICRA), Xian, 2021: 4101-4107.
- [54] Pan Z, Zhang C, Xia Y, et al. An improved artificial potential field method for path planning and formation control of the multi-uav systems[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 69(3): 1129-1133.