

四旋翼室内飞行系统仿真与研究

刘顺财¹
LIU Shuncai

摘要

为更好地了解不同算法下的路径规划效果,在路径规划方面,通过仿真对比了D*、A*、粒子群优化(PSO)以及遗传算法与粒子群优化结合(GA+PSO)4种算法的性能,实验表明D*与A*算法在静态环境下能快速锁定最短路径。为让四旋翼更好地在2D激光雷达实现飞行器的立体室内飞行能力。探索在无3D激光雷达条件下,单目摄像头与2D激光雷达结合的飞行策略。实验证明,该策略能精确遍历预设巡航点,误差控制在0.1 m以内,满足题目要求的0.25 m以内。同时,揭示了2D激光雷达在特定应用场景下的局限性,并提出解决方案,配备两个单目摄像头可以避免飞机旋转导致的地图旋转问题。这一设计有效克服地图的偏差问题,为复杂环境下的精准导航提供了可靠的技术支持。

关键词

路径规划;单目摄像头;2D激光雷达;立体

doi: 10.3969/j.issn.1672-9528.2025.07.014

0 引言

为让四旋翼飞行器更好地实现自动路径规划,在系统中引入D*算法^[1]、A*算法^[2]、粒子群优化(PSO)算法^[3]以及遗传算法与粒子群优化结合(GA+PSO)算法^[4]进行仿真对比,了解这些算法的优势和适用场景。

在只有2D的激光雷达硬件下,四旋翼飞行器进行立体飞行时,面临较大挑战。尤其当飞行器机身旋转时,此前通过2D激光雷达SLAM建图^[5]的地图也会发生旋转,导致飞机定位不准确。为解决这个问题,在保留2D激光雷达下,在机身前后各增加一个单目摄像头。这样可以提高飞行器的定位精度,扫描立体货架双面二维码时无需旋转,避免地图

旋转,从而使飞机飞行更加高效和精准。

1 软件算法

为研究四旋翼飞行器在静态室内环境的最短路径和最短时间。在系统中分别加入D*算法、A*算法、粒子群优化(PSO)算法,以及遗传算法与粒子群优化相结合(GA+PSO)的算法,实验使用Python编写程序,进行仿真对比分析。

1.1 A* 算法

在静态室内环境路径规划中,A*算法是最有效的方法。估价函数公式表示为:

$$f(n)=g(n)+h(n) \quad (1)$$

式中:估价函数 $f(n)$ 表示从初始点到目标点经由节点 n ;实际代价 $g(n)$ 表示从初始节点到节点 n 在状态空间中;估计代价 $h(n)$ 表示从节点 n 到目标点最佳路径。

为找到最短路径,关键在于估价函数 $h(n)$:当

1. 福州理工学院电子工程学院 福建福州 350506

[基金项目] 2023年度福建省中青年教师教育科研项目(科技类)-重点项目(JZ230068)

参考文献:

- [1] 钟意.基于粗粒化处理的计算机网络安全态势预测研究[J].信息与电脑(理论版),2023,35(13):239-241.
- [2] 富坤,刘琪,嵇佳明,等.基于空间尺度粗粒化的异常检测方法[J].计算机应用研究,2022,39(7):2068-2075.
- [3] 王莹.基于连接图的复杂多层有向网络粗粒度化及其同步性能研究[D].天津:天津理工大学,2022.
- [4] 贾贞.复合聚类网络同步能力的尺度可变性及其粗粒化方

法研究[D].桂林:桂林理工大学,2020.

- [5] 杨青林.复杂网络同步和可控性的粗粒化简约方法[D].沈阳:东北大学,2020:26.

【作者简介】

韩向阳(1992—),男,甘肃白银人,本科,助理实验师,研究方向:计算机网络、人工智能。

(收稿日期:2025-03-17 修回日期:2025-07-10)

$h(n)=d(n)$, 表示距离最短, 沿着最短路径进行搜索, 此时搜索效率是最高的; 当估价值 $h(n) \leq n$ 时, 表示到目标节点需要搜索更多的节点数, 导致搜索范围太大并且效率低。

1.2 D* 算法

D* 算法主要应用在动态变化的环境, 是通过 A* 算法拓展而来。(1) 比如在环境中增加障碍物; (2) 无需重新计算全部路径, 只需更新受环境影响的这部分路径。D* 算法的核心在于其估价函数:

$$f(n) = g(n) + h(n) \quad (2)$$

因为 D* 算法由 A* 算法拓展来的, 所以估价函数 $f(n)$ 、实际代价 $g(n)$ 、估计代价 $h(n)$ 和 A* 算法表示相同。

1.3 PSO 算法

PSO 算法基本思想^[6]是通过群体中信息共享和个体协作来实现最短距离。通过搜索空间中的每个粒子最优解, 将粒子记为当前个体极值, 并把这一值共享所有粒子。找到最优的个体极值, 从而完成所有粒子群中的当前全局最优解。每个粒子根据自身当前个体极值以及所有粒子共享的当前最优解来调整自己的速度和位置。粒子通过跟踪两个极值 (pbest, gbest) 进行更新, 用公式表示为:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1(pbest_i - x_i(t)) + c_2 \cdot r_2(gbest - x_i(t)) \quad (3)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

式中: $v_i(t)$ 和 $x_i(t)$ 分别表示粒子 i 在时间 t 的速度和位置; w 表示惯性权重, 控制之前速度对当前速度的影响程度; c_1 和 c_2 表示学习因子或加速系数, 通常设置为 2; r_1 和 r_2 表示两个随机数, 在 $[0,1]$ 区间内均匀分布, 以增加搜索的随机性。

pbest_{*i*} 是粒子 i 迄今为止找到的最佳位置。gbest 是整个群体迄今为止找到的最佳位置。

1.4 GA+PSO 算法

将 GA 和 PSO 结合^[7]目的是利用 GA 的全局搜索能力和 PSO 的快速收敛性。GA 通过交叉和变异进行全局搜索, 避免陷入局部最优。PSO 通过群体协作快速收敛找到较优解。结合后算法对复杂问题的适应性更强, 此算法主要应用于路径规划。

2 算法软件仿真

为进行数据分析, 需对场地进行数学建模。场地大小为 $500 \text{ cm} \times 500 \text{ cm}$, 以 50 cm 为最小单位划分, 可分割为 10×10 的网格。场地内有两个货架, 以 1 表示障碍物, 0 表示无障碍区域^[8]。因此, 可将场地建模成一个数组, 这个

数组是一个二维数组。由 10 行和 10 列组成, 每个元素都是一个整数, 表示对应网格位置上的状态 (0 表示无障碍, 1 表示有障碍物)。在数据分析和编程中, 二维数组常用于表示平面上的网格或矩阵结构。处理后的数组为:

```
grid = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

3 硬件模型

四旋翼飞行器的硬件由单片机^[9]、RK3588S 计算机、2 个单目摄像头、N10 激光雷达以及电调、电机和螺旋桨等硬件组成, 如图 1 所示。

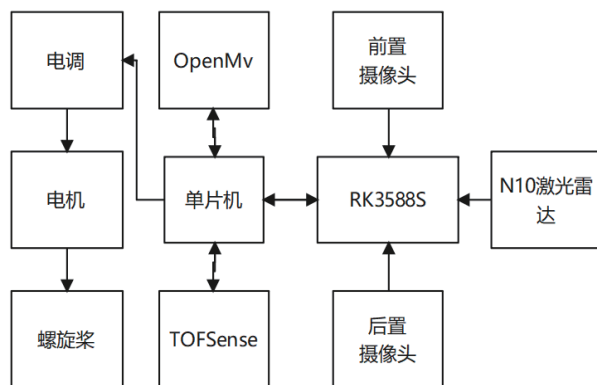


图 1 系统硬件框架图

4 软件模型

为模拟四旋翼飞行器飞行立体场地, 系统测试场地采用 2024 年全国大学生电子设计大赛设计中立体货架盘点无人机系统仿真环境, 从场地的正方形小方块 ($50 \text{ cm} \times 50 \text{ cm}$) 飞到场地的圆形 (直径 50 cm), 场地如图 2 所示。图 2 为一个大小为 $500 \text{ cm} \times 400 \text{ cm}$ 的仓库俯视图, 黑色方形区域为起飞点、黑色圆形区域为降落点, 另有 2 个双面立体货架, 每个货架两端为红色圆杆, 立体货架的 4 个面编号为 A 面、B 面、C 面、D 面, 每个面上有 6 个固定坐标位置, 用于张贴包含货物编号信息的二维码 (共 24 个二维码), 6 个固定坐标位置上面 3 个二维码中心点高为 140 cm , 下面 3 个二维码中心点为 100 cm

。系统的软件方面设计使用 C 和 Python 语言进行编程，在单片机核心板中烧写进以编译好的代码进行运行，在 RK3588S 中安装 ROS 框架，并使用 Python 编写 SLAM 动态建图代码并运行。

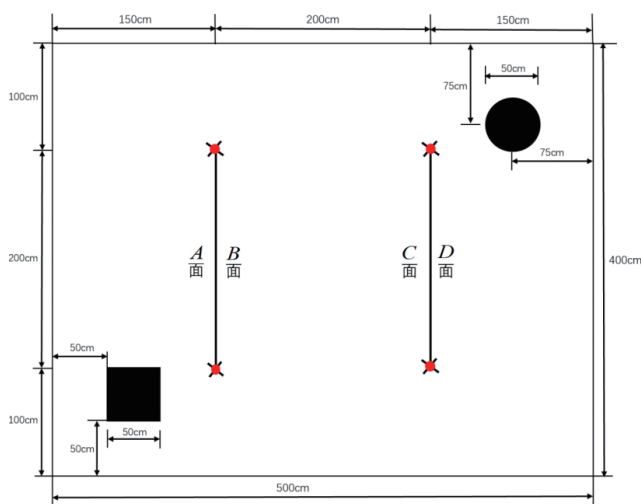


图2 立体货架仓库俯视图

4.1 巡航场地建模

为了更好地通过 Python 编写程序实现四旋翼导航与 SLAM 动态建图^[10]，需要对场地建模，按照每个格子尺寸都是 50 cm，根据图 3 模拟场地尺寸分割设置起飞时正方形坐标为原点 (0, 0)，最终是圆形坐标为 (7, 5)。

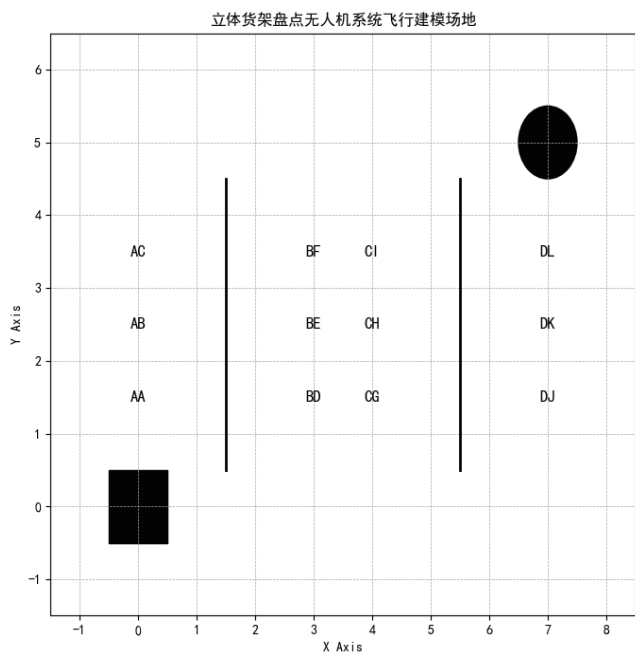


图3 模拟场地图

平面上的巡航点坐标分别为原点 $0(0, 0) \rightarrow$ 原点 $0(0, 0) \rightarrow AC(0, 3.5 \times 50) \rightarrow AC(0, 3.5 \times 50) \rightarrow AB(0, 2.5 \times 50) \rightarrow AB(0, 2.5 \times 50) \rightarrow AA(0, 1.5 \times 50) \rightarrow AA(0, 1.5 \times 50)$

$\rightarrow NC(0, -0.5 \times 50) \rightarrow NC(3 \times 50, -0.5 \times 50) \rightarrow BD(3 \times 50, 1.5 \times 50) \rightarrow BD(3 \times 50, 1.5 \times 50) \rightarrow BE(3 \times 50, 2.5 \times 50) \rightarrow BE(3 \times 50, 2.5 \times 50) \rightarrow BF(3 \times 50, 3.5 \times 50) \rightarrow BF(3 \times 50, 3.5 \times 50) \rightarrow CI(4 \times 50, 3.5 \times 50) \rightarrow CI(4 \times 50, 3.5 \times 50) \rightarrow CH(4 \times 50, 2.5 \times 50) \rightarrow CH(4 \times 50, 2.5 \times 50) \rightarrow CG(4 \times 50, 1.5 \times 50) \rightarrow CG(4 \times 50, 1.5 \times 50) \rightarrow NC(4 \times 50, 0.5 \times 50) \rightarrow NC(7 \times 50, 0.5 \times 50) \rightarrow DJ(7 \times 50, 1.5 \times 50) \rightarrow DJ(7 \times 50, 1.5 \times 50) \rightarrow DK(7 \times 50, 2.5 \times 50) \rightarrow DK(7 \times 50, 2.5 \times 50) \rightarrow DL(7 \times 50, 3.5 \times 50) \rightarrow DL(7 \times 50, 3.5 \times 50) \rightarrow$ 终点 $(7 \times 50, 5 \times 50) \rightarrow$ 终点 $(7 \times 50, 5 \times 50)$ ，总计 32 个点。因为每个巡航点都是 $n \times 50$ ，为了避免出现小数点，分解成 $n \times 10 \times 5$ ，结合 24 点扫描二维码在相同巡航点需要扫描上下具有不同的高度 100 和 140 的二维码，提取不同位置的特征坐标值，生成巡航点三维数组 `const int16_t cruise_point_tab[3][32]` = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 30, 30, 30, 30, 30, 30, 30, 40, 40, 40, 40, 40, 40, 70, 70, 70, 70, 70, 70, 70, 70}, {0, 0, 35, 35, 25, 25, 15, 15, -5, -5, 15, 15, 25, 25, 35, 35, 35, 35, 25, 25, 15, 15, -5, -5, 15, 15, 25, 25, 35, 35, 50, 50}, {0, 150, 140, 100, 100, 140, 140, 100, 150, 150, 140, 100, 100, 140, 140, 100, 100, 140, 150, 150, 140, 100, 100, 140, 150, 150, 140, 100, 100, 140, 150, 0}};，至此建模完成。

4.2 软件流程图

在无人机送货系统中，整个程序流程可以分为几个关键步骤，实现快速、高效自主巡检、盘点任务。

首先，程序开始时进行初始化，对无人机各项传感器以及单片机底层硬件进行校准，确保数据准确性。完成初始化后，无人机等待起飞按键指令。当接收到起飞指令后，发布起始点巡航点 $0(0, 0, 150)$ 。无人机起飞至预定巡航点高度 150cm，判断是否在巡航点原点 $(0, 0, 150)$ 上方，是则发布起飞巡航点 $AC1(0, 35, 140)$ 。

接下来无人机根据设定巡航点，依次发布巡航点 $AC1(0, 35, 140) \rightarrow AC2(0, 35, 100) \rightarrow AB2(0, 25, 100) \rightarrow AB1(0, 25, 140) \rightarrow AA1(0, 15, 140) \rightarrow AA2(0, 15, 100) \rightarrow NC(0, -5, 150) \rightarrow NC(30, -5, 150) \rightarrow BD1(30, 15, 140) \rightarrow BD2(30, 15, 100) \rightarrow BE2(30, 25, 100) \rightarrow BE1(30, 25, 140) \rightarrow BF1(30, 35, 140) \rightarrow BF2(30, 35, 100) \rightarrow CI2(40, 35, 100) \rightarrow CI1(40, 35, 140) \rightarrow CH1(40, 25, 140) \rightarrow CH2(40, 25, 100) \rightarrow CG1(40, 15, 100) \rightarrow CG2(40, 15, 140) \rightarrow NC(40, -5, 150) \rightarrow NC(70, 5, 150) \rightarrow DJ1(70, 15, 140) \rightarrow DJ2(70, 15, 100) \rightarrow DK2(70, 25, 100) \rightarrow DK1(70, 25, 140) \rightarrow DL1(70, 35, 140) \rightarrow DL2(70, 35, 100) \rightarrow$ 终点上方 $(70, 50, 150) \rightarrow$ 终点 $(70, 50, 0)$ 。同时依次飞至上述巡航点上。在位于巡航点上时，无人机在 A 面和 C 面通过前面摄像头判断是否正对着二维

码的圆形面，有则通过发出激光笔指示灯操作完成巡检、盘点任务。无人机在 *B* 面和 *D* 面通过后面摄像头判断是否正对着二维码的圆形面，有时通过发出激光笔指示灯操作完成巡检、盘点任务。

在巡检完 *D* 面巡航点任务后，无人机发布终点飞巡航点位置信息，无人机飞至巡航终点上方。当无人机返回到起终点上方时，执行定点降落操作，确保降落的准确性和稳定性。

整个程序流程中，无人机需要不断地检测自身位置发布下一个巡航点目标位置信息，实时调整飞行状态以适应任务要求。在巡检巡航点任务过程中，无人机需保持稳定的悬停，确保巡航点任务操作的顺利进行。通过上述流程，无人机可在规定时间内完成巡航点任务。整个过程四旋翼飞行器都不用机身都不用旋转，最后无人机飞至巡航终点上方并降落结束本次任务。系统软件工作具体流程图如图 4 所示。

5 实验分析

5.1 实验数据

四旋翼室内飞行系统仿真与研究专注于两大领域：对比不同算法下的路径规划效果以及利用前后 2 个单目摄像头与 2D 激光雷达实现飞行器的立体室内飞行能力。

(1) 对比分析四旋翼飞行器在 4 种算法下的自动路径

规划仿真，尤其聚焦于优化规划时间及寻求最短路径。

(2) 研究在无 3D 雷达下，利用前后 2 个单目摄像头与 2D 激光雷达对立体空间上的 30 个巡航点进行精准遍历。通过测试四旋翼飞行器遍历 2 个立体货架 4 面 (*A*、*B*、*C*、*D* 面板上的 24 个二维码) 的巡航点，并记录飞机悬停在这 24 个巡航点后，激光笔与二维码中心位置的偏离值。从而验证单目摄像头和 2D 激光雷达的遍历立体货架的可行性。

5.2 实验结果分析

基于建模好的场地 *grid*，利用 Python 实现 4 种算法。这些算法将模拟从场地正方形小方块 (50 cm×50 cm) 绕过两个货架，自主规划路径至圆形目标区域 (直径 50 cm)，并计算最短路径与最短时间，50 cm 在建模中表示 1 个距离单位最终模拟测试结果如表 1 和图 5 所示。

表 1 D*、A*、PSO 和 GA+PSO4 种算法仿真表

算法	从起点到终点规划路径	
	所需步数 / 步	规划路径长度 /cm
D*	11	13.9×50
A*	11	13.9×50
PSO	87	137.9×50
GA+PSO	6	12.7×50

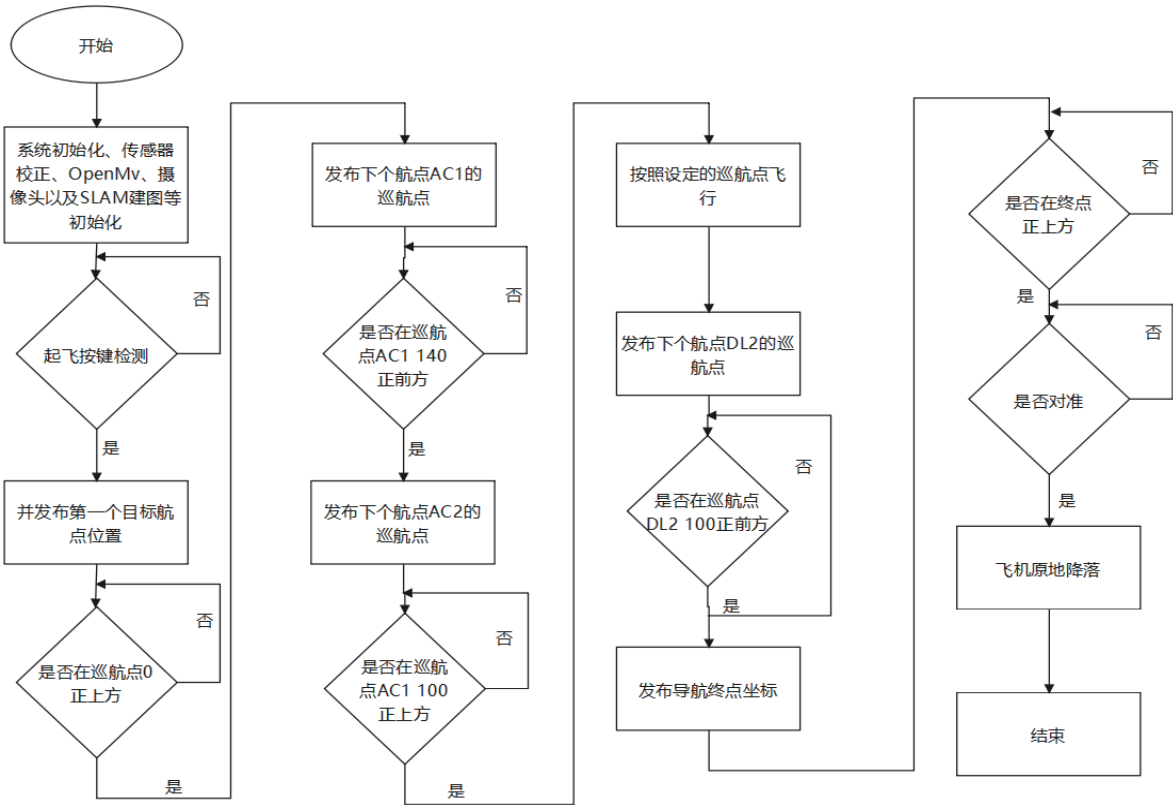


图 4 系统软件工作具体流程图

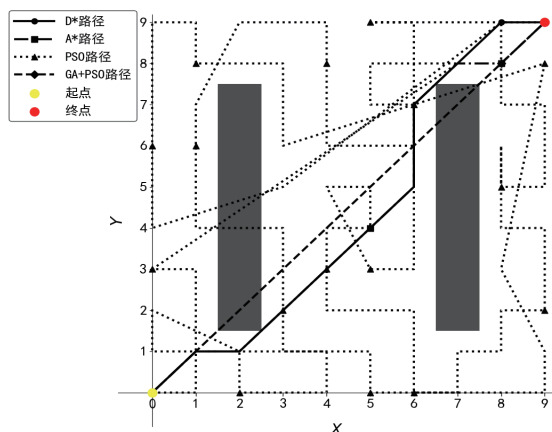


图5 算法路径规划对比图

根据表1和图5的数据可知，在静态环境中，D*算法和A*算法需要经过11步的时间就可以准确地找到最短路径（13.9×50 cm）。而粒子群优化（PSO）算法及其与遗传算法的结合（GA+PSO）用时分别为87步和6步，规划的路径分别达到了137.9×50 cm和12.7×50 cm。

实验表明：在静态环境中，可以优先使用D*与A*算法，它们都可以较快地规划出最短路径。而PSO算法未能快速完成，而经过改进的算法GA+PSO算法可以快速完成，并寻找出最短路径。实验过程中也进行了多次算法优化，但结果并非完美。由于仿真的路径具有不确定性，所以暂时无法用于真实的四旋翼飞行器中。

实验通过飞行记录2D激光雷达、2个单目摄像头和2D激光雷达数据对比。记录原点(0, 0, 150)时，激光打点距离下面正方形中心点；记录AC1(0, 35, 140)，BD2(30, 15, 100)，CG1(40, 15, 100)，DL1(70, 35, 140)，四点激光打点距离圆形二维码的中心点距离；记录终点(70, 50, 150)时，激光打点距离填充圆形的中心点距离。因为巡航点数量比较多，选取具有代表性的巡航点，为了测试过程中存在单次误差，这边每个巡航点测试6次，然后取平均值，单位为m，结果如表2所示。

表2 巡航点记录对比表

巡航点	单位: m	
	2D LiDAR	ours
原点 (0, 0, 150)	0.082	0.083
AC1(0, 35, 140)	0.072	0.071
BD2(30, 15, 100)	0.121	0.082
CG1(40, 15, 100)	1.250	0.081
DL1(70, 35, 140)	1.340	0.086
终点 (70, 50, 150)	2.130	0.082

实验证明，采用单目摄像头结合2D激光雷达，误差基本控制在0.1 m以内，符合题目要求的0.25 m以内。然而，

在测试A、B、C、D面及降落点时，仅使用2D激光雷达的误差较大，未能达标。原因是2D激光建图后，机身旋转会导致地图同步旋转，进而造成建图不准确。这一发现验证了2D激光雷达在特定应用场景下的局限性。

6 结语

对比四旋翼飞行器在4算法下的自动规划路径，实验表明在静态场景下，D*与A*算法能快速锁定最短路径。然而，PSO及其与GA的结合算法，由于算法特性和大规模地图建模的复杂性，寻优效果不佳，难以找到最短路径，且容易陷入局部最优。

实验中解决在无3D激光雷达条件下利用2个单目摄像头与2D激光雷达下，SLAM建图在立体建图时飞机旋转导致的地图转变与精度定位问题。系统配备了两个单目摄像头（前后各一）与2D激光雷达，确保飞行器在飞行过程中无需旋转。实验验证显示，该设计使飞行器能够精确遍历各个巡航点，显著提升飞行准确性与稳定性。

参考文献：

- [1] 乔道迹, 张艳兵. 多障碍环境下巡检机器人路径规划优化研究[J]. 现代电子技术, 2025,48(1):130-134.
- [2] 曾宪阳, 张加旺. 改进A*算法融合DWA机器人路径规划研究[J]. 电子测量技术, 2025,48(6):20-27.
- [3] 吴鹏, 叶宝林, 吴维敏, 等. 基于改进混沌粒子群算法的交通信号控制[J]. 计量学报, 2024,45(12):1876-1884.
- [4] 王慧. 结合遗传算法的粒子群优化模型及其应用研究[D]. 济南: 山东师范大学, 2008.
- [5] 任泉, 王思宇, 张慧, 等. 基于2D激光SLAM的室内移动机器人定位与建图的设计及实现[J]. 信息技术与信息化, 2024(11): 182-185.
- [6] 王芷馨. 基于改进粒子群算法的模块化系统辨识[D]. 青岛: 青岛大学, 2021.
- [7] 刘能现, 俞建家. 基于GA与PSO并行的模糊聚类算法[J]. 福州大学学报(自然科学版), 2009,37(3):336-339.
- [8] 蒋睿. 基于改进混合粒子群算法无人机室内路径规划研究[D]. 太原: 中北大学, 2023.
- [9] 胥鑫磊, 李世豪, 董丁铭, 等. 基于STM32F407的植保飞行器设计研究[J]. 科技资讯, 2024,22(11):134-137.
- [10] 武冬雨. 基于多传感器融合的四旋翼无人机室内自主飞行技术[D]. 西安: 西安电子科技大学, 2022.

【作者简介】

刘顺财(1986—), 男, 福建漳州人, 硕士, 副教授, 研究方向: 嵌入式AI。

(收稿日期: 2025-02-23 修回日期: 2025-07-01)