# 试 验 报 告

## 实验名称： 智能机器人大作业

## 课程名称： 智能机器人原理

院系： 人工智能学院

班级： 2021 级本科 1 班

姓名： 刘元昊

学号： 21311223

日期： 2024 年 7 月 10 日

**中山大学**

1. 问题理解

**题目：**设计一个场景，包括起点，终点，障碍物，采用2种及以上的**规划方法**实现规划过程，采用2种及以上的**横向控制方法**实现路径跟踪过程，要求实现模拟移动机器人运动的过程，机器人本体的**运动控制模型**不限。

**要求：**

1. 植入代码语言不限，要求可运行，有readme，制作代码运行讲解视频文件；
2. 简单介绍不同方法的原理，优缺点；
3. 试验分析验证、在规划路径质量，规划效率2个方面对比不同规划方法；
4. 在跟踪误差方面对比不同的跟踪控制方法；
5. 要求有效果图、数据分析图、数据分析表；
6. 可以去网上寻找开源代码，但不允许同学之间抄袭，否则被确认后，2个人都是0分处理。

**问题理解：**

实现路径规划的**规划方法**的方法，是机器人导航中的关键步骤，主要目的是为机器人找到从起点到终点的可行路径，同时避开障碍物。常见的路径规划方法包括A\*算法和Dijkstra算法。

实现路径跟踪的**横向控制方法**是指机器人在已规划路径上行驶，确保其沿着预定路径前进。横向控制方法用于调整机器人的方向和姿态

在本问题中，考虑一个二维平面作为场景，人为确定起点、终点和障碍物，从而模拟一个典型的机器人路径规划场景。

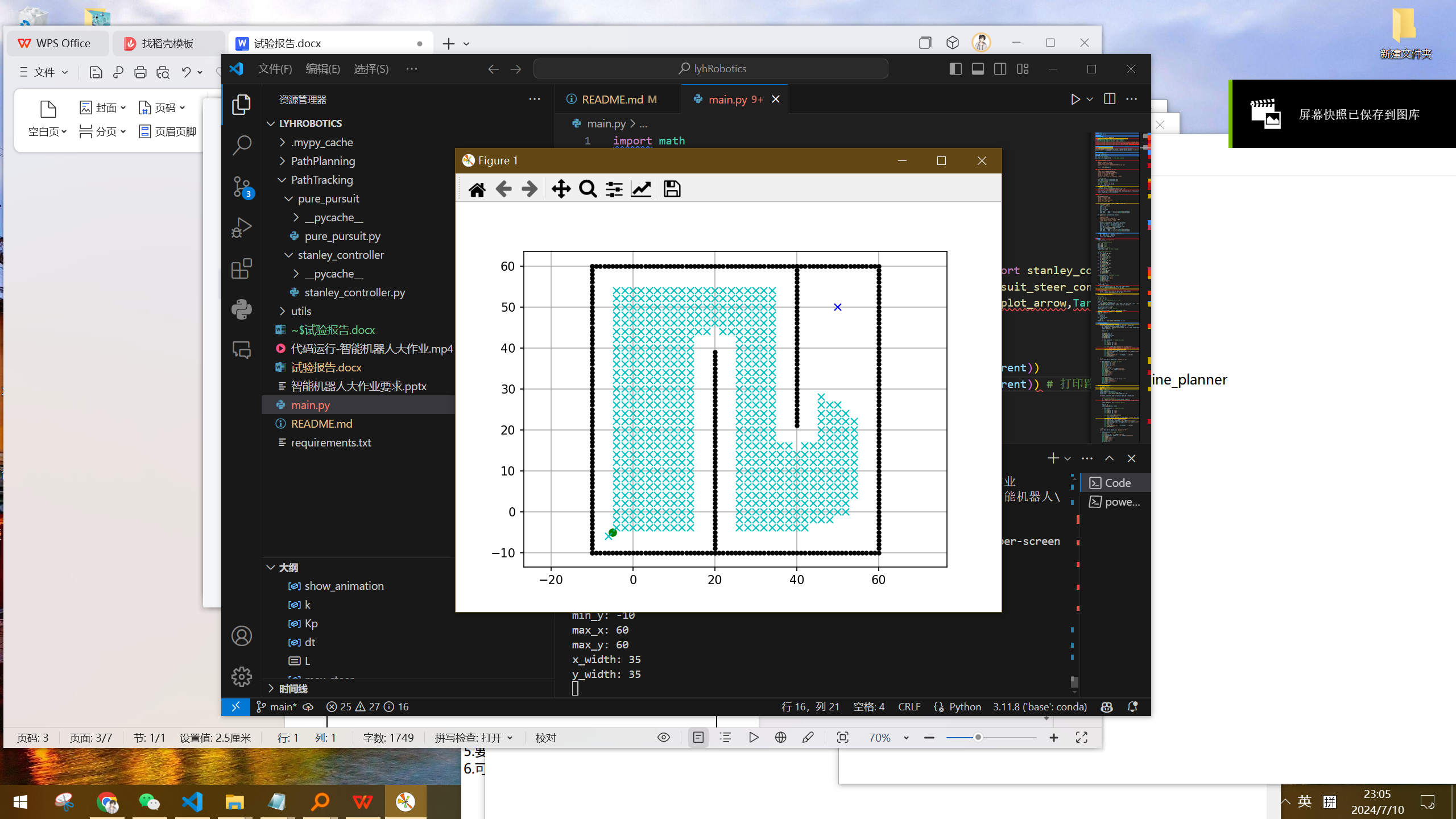


图 1 二维平面，包括起点、重点、障碍物

1. 实验环境

本次实验是在标准的Python编程环境中完成，具体依赖以下软件包和相关版本：

numpy == 2.0.0

scipy == 1.14.0

matplotlib == 3.9.1

cvxpy == 1.5.2

pytest == 8.2.2 # For unit test

pytest-xdist == 3.6.1 # For unit test

mypy == 1.10.1 # For unit test

ruff == 0.5.1 # For unit test

本次实验的代码保存在Github的FarmerJohnLYH/lyhRobotics。

链接：<https://github.com/FarmerJohnLYH/lyhRobotics>

1. 算法设计的简介

该系统集成了多种路径规划算法（如A\*算法和Dijkstra算法）和路径跟踪算法（如Stanley控制器和Pure Pursuit控制器），旨在实现高效、稳定的路径跟踪。主要组件包括路径规划模块和路径跟踪模块，通过这些组件的协同工作，实现自动驾驶车辆在复杂环境中的路径规划和稳定行驶。

1. 路径规划模块·A\*算法
2. 算法简介

A算法（A-star）是一种广泛应用于图搜索和路径规划的启发式搜索算法。它在计算机科学和人工智能领域中占有重要地位，特别适用于游戏开发、机器人导航以及地理信息系统等场景。A算法结合了Dijkstra算法的最短路径搜索和贪婪最佳优先搜索的优点，通过启发式函数有效地减少了搜索空间，从而提高了搜索效率。

1. 算法原理

A\*算法的基本原理是利用一个优先队列存储路径，并根据路径的总代价选择下一条路径进行扩展。路径的总代价由两个部分组成：

* g(n)：从起点到节点n的实际代价。
* h(n)：从节点n到目标节点的启发式估计代价。

1. A\*算法的评价函数f(n)定义为： f(n)=g(n)+h(n)

3. 算法优势和不足

### 优势

### **高效性**：A\*算法结合了实际代价和启发式估计，可以在很多情况下比传统的Dijkstra算法更快地找到最优路径。

### **灵活性**：通过调整启发式函数，A\*算法可以适应不同的搜索场景。启发式函数的设计可以根据具体问题进行优化，从而提升搜索效率。

### **最优性**：如果启发式函数是可接受的（即不超过实际代价），A\*算法能够保证找到全局最优路径。

### 不足

### **空间复杂度高：A\*算法需要维护开放列表和闭合列表，可能会占用大量内存，特别是在图规模较大时。**

### **依赖启发式函数：启发式函数的设计直接影响算法的性能和效果。如果启发式函数不准确，可能会导致搜索效率下降。**

### **扩展性有限：在某些复杂的动态环境中，A\*算法的适用性有限，可能需要结合其他算法或技术进行改进。**

1. 路径规划模块·Dijkstra算法

1. 算法简介

Dijkstra算法是一种经典的单源最短路径算法，由荷兰计算机科学家Edgar Dijkstra于1956年提出。该算法在加权图中，能够有效地找到从起点到其他所有节点的最短路径。由于其简单性和高效性，Dijkstra算法被广泛应用于路由算法、网络通信、地图导航等领域。

2. 算法原理

Dijkstra算法的核心思想是通过逐步扩展最短路径集合，利用贪心策略找到从起点到各个节点的最短路径。具体步骤如下：

① 初始化：设定起点节点的距离为0，其他节点的距离为无穷大。将所有节点标记为未访问。

② 选择当前节点：从未访问的节点中选择距离起点最近的节点作为当前节点。

③ 更新邻居节点：对于当前节点的每个邻居节点，计算从起点经过当前节点到达邻居节点的距离。如果该距离小于邻居节点的当前距离，则更新邻居节点的距离。

④ 标记访问：将当前节点标记为已访问。

⑤ 重复步骤2-4，直到所有节点都被访问。

3. 算法优势和不足

**优势**

1. 简单易实现：Dijkstra算法逻辑清晰，易于理解和实现。
2. 适用范围广：适用于各种加权图，广泛应用于路由、导航等领域。
3. 保证最优解：对于非负权图，Dijkstra算法能够保证找到从起点到其他所有节点的最短路径。

**不足**

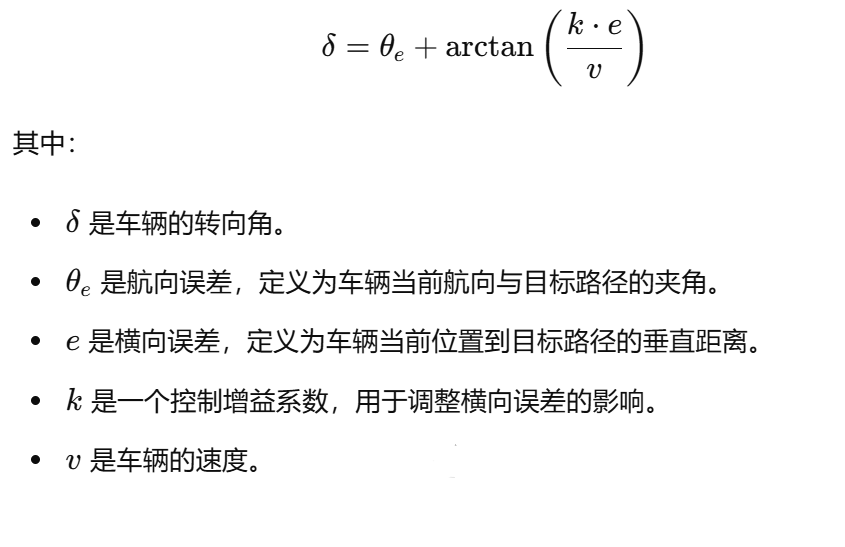
1. 不适用于负权图：Dijkstra算法不能处理包含负权边的图，这会导致错误的最短路径计算。
2. 空间复杂度高：需要维护一个优先队列和距离数组，对于大规模图，可能会占用大量内存。
3. 无法动态更新：在动态变化的图中，Dijkstra算法需要重新计算整个图的最短路径，效率较低。
4. 路径跟踪模块·Stanley控制器

1. 算法简介

Stanley控制器是一种经典的路径跟踪算法，最早由斯坦福大学在其自主车辆项目中提出。该算法广泛应用于自动驾驶车辆和移动机器人中，用于路径跟踪和路径规划。其核心思想是通过调整车辆的转向角度，使车辆的横向误差和航向误差尽可能小，从而实现平滑且准确的路径跟踪。

2. 算法原理

Stanley控制器的核心是通过调整车辆的转向角度来最小化车辆的横向误差（cross-track error, CTE）和航向误差（heading error）。具体来说，Stanley控制器根据以下公式计算转向角：



通过上述公式，Stanley控制器将横向误差和航向误差结合起来，生成一个转向角度，使车辆尽可能平滑地跟踪目标路径。

3. 算法优势和不足

**优势**

1. 实现简单：Stanley控制器的计算相对简单，易于实现和调试。
2. 实时性好：由于其计算复杂度低，Stanley控制器能够实时运行，适用于自动驾驶和实时路径跟踪场景。
3. 鲁棒性强：Stanley控制器对路径的平滑性和噪声具有较好的鲁棒性，能够处理一定程度的路径偏差和测量噪声。

**不足**

1. 对速度依赖性强：Stanley控制器的性能依赖于车辆的速度，当速度较低时，横向误差的影响较大，可能导致路径跟踪不稳定。
2. 在急转弯时表现不佳：由于Stanley控制器基于横向误差和航向误差的线性组合，在急转弯或曲率变化较大的路径上可能表现不佳，出现过冲或不足的情况。
3. 参数调节困难：控制增益系数需要根据具体应用场景进行调节，可能需要反复试验和调整，增加了调试的复杂性。
4. 路径跟踪模块·Pure Pursuit控制器

1. 算法简介

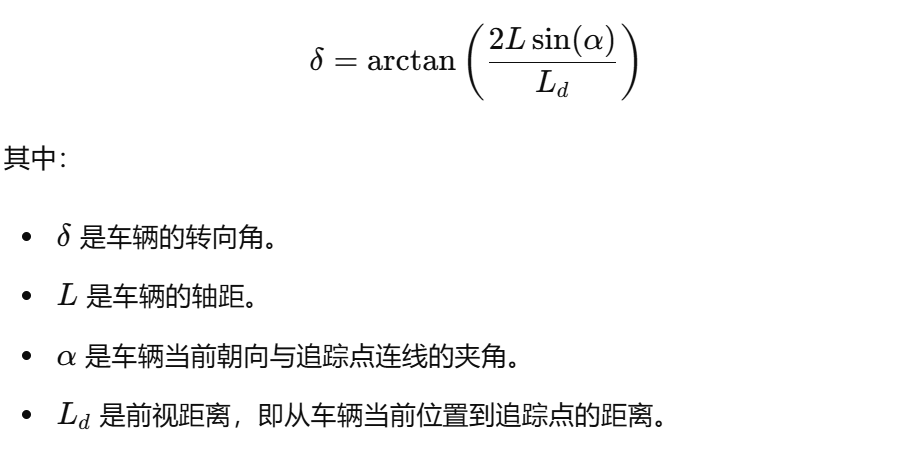
Pure Pursuit控制器是一种经典的路径跟踪算法，广泛应用于自动驾驶车辆和移动机器人。该算法的核心思想是通过将车辆的目标位置设置为一系列的“追踪点”，车辆不断调整方向，以追踪这些点，从而实现平滑且准确的路径跟踪。Pure Pursuit控制器由于其计算简单且效果显著，在许多实际应用中得到了广泛采用。

2. 算法原理

Pure Pursuit控制器的基本原理是通过计算车辆当前位置与目标路径上的一个追踪点之间的几何关系，确定合适的转向角度，使车辆沿着路径行驶。具体步骤如下：

1. 选择追踪点：从当前路径中选择一个距离车辆当前位置最近的追踪点，该点通常位于一个预设的前视距离内。
2. 计算转向角：根据车辆当前位置与追踪点的位置，计算需要的转向角度，使车辆朝向追踪点行驶。
3. 更新车辆状态：根据计算的转向角，调整车辆的转向，更新车辆的位置和朝向。
4. 重复上述步骤，直到车辆到达目标位置。

算法公式如下：



3. 算法优势和不足

**优势**

1. 实现简单：Pure Pursuit控制器的算法逻辑简单，易于理解和实现。
2. 实时性好：由于其计算复杂度低，Pure Pursuit控制器能够实时运行，适用于自动驾驶和实时路径跟踪场景。
3. 平滑路径跟踪：Pure Pursuit控制器通过不断调整追踪点，实现平滑的路径跟踪，适合各种复杂路径。

**不足**

1. 对速度依赖性强：Pure Pursuit控制器的性能依赖于车辆的速度，当速度较高时，前视距离的选择变得更加关键，可能导致路径跟踪不稳定。
2. 在急转弯时表现不佳：由于Pure Pursuit控制器基于几何关系计算转向角度，在急转弯或曲率变化较大的路径上可能表现不佳，出现过冲或不足的情况。
3. 前视距离选择困难：前视距离的选择需要根据具体应用场景进行调节，可能需要反复试验和调整，增加了调试的复杂性。

4. 算法改进

为了提升Pure Pursuit控制器的性能和适用性，可以从以下几个方面进行改进：

1. 动态前视距离调整

在实际应用中，可以引入动态前视距离调整机制，根据车辆速度和路径曲率动态调整前视距离。这样可以在低速时选择较短的前视距离，以提高转向精度；在高速时选择较长的前视距离，以确保路径跟踪的稳定性。

1. 结合PID控制器

将Pure Pursuit控制器与PID控制器相结合，利用PID控制器的反馈调整能力，进一步优化转向角度。PID控制器可以根据横向误差和角度误差进行调整，减少过冲和振荡，提高路径跟踪的精度和稳定性。

1. 试验分析

|  |  |
| --- | --- |
| a_star_stanley1  图 2 A\*和Stanley控制器 | a_star_pure_pursuit1  图 3 A\*和Pure Pursuit控制器 |
| dijkstra_stanley1  图 4 Dijkstra和Stanley控制器 | dijkstra_pure_pursuit1  图 5 Dijkstra和Pure Pursuit控制器 |

实验结果如图所示，展示了不同路径规划算法（A\*和Dijkstra）与不同控制算法（Stanley控制器和Pure Pursuit控制器）组合后的路径跟踪效果。

每个图中，红点线表示目标路径（course），蓝线表示实际跟踪路径（trajectory），红色叉表示目标点（target），设置车辆目标速度9.99 km/h。

1. 针对 A\* 和Stanley 控制器的组合

轨迹与目标路径基本吻合，特别是在直线路段和较缓的曲线路段。

在急转弯处有一定的偏差，但整体控制效果较好。A\*算法与Stanley控制器的组合在路径跟踪中表现较为稳定，能够有效地在复杂路径中跟随目标路径。

1. 针对A\*和Pure Pursuit 控制器的组合

实际轨迹与目标路径高度吻合，包括直线和曲线部分。

在急转弯处表现出色，偏差较小。A\*算法与Pure Pursuit控制器的组合在路径跟踪中表现非常出色。

1. 针对Dijkstra 和 Stanley 控制器的组合

轨迹与目标路径基本吻合，尤其在直线路段表现良好。

在急转弯处有一定的偏差，但总体控制效果不错。Dijkstra算法与Stanley控制器的组合能够实现稳定的路径跟踪。

1. 针对Dijkstra 和 Pure Pursuit 控制器的组合

实际轨迹与目标路径高度吻合，包括直线和曲线部分。

在急转弯处表现出色，偏差很小。Dijkstra算法与Pure Pursuit控制器的组合在路径跟踪中表现优异。

1. 结论

根据实验结果，比较A\*算法与Dijkstra算法可以发现，两者在路径规划中表现均较为稳定，均能找到合理的路径。

在复杂路径上，A\*算法由于其启发式搜索特性，可能会比Dijkstra算法在计算效率上有一定优势。

比较Stanley控制器与Pure Pursuit控制器发现，Stanley控制器在路径跟踪中表现更加稳定，特别是在处理横向误差和航向误差时效果良好。而Pure Pursuit控制器在路径跟踪中表现非常优异，特别是在急转弯和曲率变化较大的路径上，其平滑跟踪效果显著。

综合实验结果，在给定的算法中的最佳组合为A\*算法与Pure Pursuit控制器的组合，在路径跟踪中表现最优，适用于各种复杂路径，能够实现平滑且准确的路径跟踪。

通过对实验结果的分析，可以得出不同路径规划算法与控制算法的组合在路径跟踪中的效果和适用场景，为自动驾驶和移动机器人路径规划提供了有力的支持。