

多智能体车辆控制实验报告一

1. UDPClient 类的作用、构造过程及通信原理

1.1 UDPClient 类的作用

UDPClient 类负责车辆与仿真服务器之间的 UDP 通信，主要功能包括：

- 接收仿真服务器发送的车辆状态数据（如位置 x、y，航向角 yaw，速度 speed 等）
- 向仿真服务器发送车辆控制指令（线速度 v 和角速度 w）
- 解析 JSON 格式的通信数据

1.2 类的构造过程

1. 初始化参数：

- 服务器 IP 地址（如 172.25.73.124）
- 本地接收端口（9858）
- 服务器接收端口（9859）
- 车辆名称标识（"cyc"）

2. 创建 UDP Socket：

- 绑定本地端口用于接收数据
- 设置目标端口用于发送指令

3. 启动数据接收线程：

- 使用 threading 模块创建后台线程
- 持续监听服务器发送的数据

1.3 UDP 通信基本原理

- 无连接协议：通信前不需要建立连接
- 不可靠但高效：不保证数据包的顺序和可达性
- 适合实时控制：延迟低，吞吐量高
- 使用 JSON 格式传输数据

1.4 net 参数详解

net 字符串格式："vehicle_name,server_ip,udp_port,udp_send_port"

net = "jHjs2augsl dwR5fNGJFT4DtFBYiO,172.25.73.110,6424,6425"

各参数说明：

| 参数名 | 示例值 | 作用 |

参数名	示例值	作用
vehicle_key	"jHjs2augsl dwR5fNGJFT4DtFBYiO"	车辆加密标识符
server_ip	"172.25.73.110"	更新后的服务器 IP 地址

参数名	示例值	作用
udp_port	6424	新的本地数据接收端口
udp_send_port	6425	新的服务器指令接收端口

1.5 代码补全过程

1. 解析 net 字符串
2. 初始化 UDPClient 实例
3. 启动通信线程
4. 设置控制频率 (10Hz)
5. 实现控制循环

2. get_vehicle_state()实现逻辑及实验验证

2.1 方法实现逻辑

1. 通过 UDP 接收原始数据
2. 解析 JSON 格式数据
3. 匹配对应 vehicle_name 的车辆
4. 返回 VehicleData 对象

2.2 实验验证

实验 1: 直线运动测试

- 控制参数: $v=10$, $w=0$
- 预期结果: 车辆沿直线匀速运动
- 实际效果: 成功实现直线运动
- 问题记录: 初始速度设置过大导致运动不稳定

```
v, w = 10, 0 # 这里可以替换为你的控制算法
self.udp_client.send_control_command(v, w)
```

实验 2: 圆周运动测试

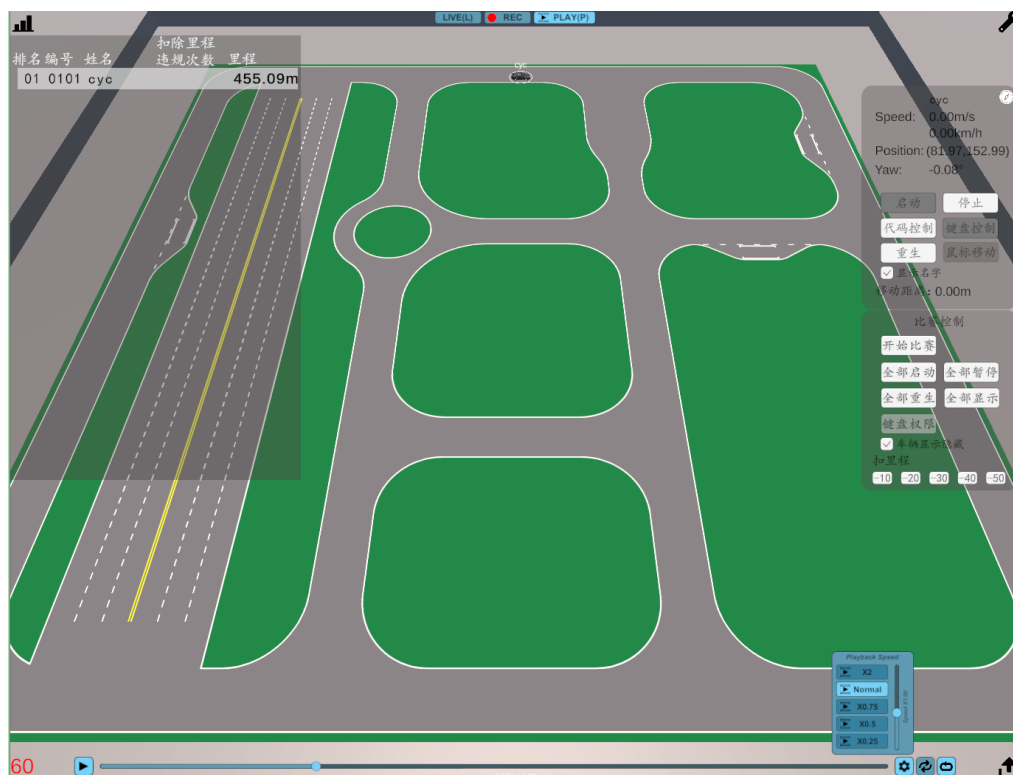
- 控制参数: $v=10$, $w=0.55$
- 预期结果: 车辆做圆周运动
- 实际效果: 成功实现圆周运动
- 问题记录: 转向半径略大于预期

```

start_time = time.time()
while True:
    vehicle_data = self.udp_client.get_vehicle_state()
    self.m_x = vehicle_data.x
    self.m_y = vehicle_data.y
    self.m_yaw = vehicle_data.yaw / 180 * math.pi

    v, w = 10, 0.55 # 这里可以替换为你的控制算法
    self.udp_client.send_control_command(v, w)

```



3. 交互式控制逻辑设计

3.1 原始代码问题

- 仅支持固定参数控制
- 缺乏实时交互能力
- 无法动态调整控制参数

3.2 改进方案

1. 增加键盘控制模块
2. 实现动态参数调整
3. 添加安全限制

改进后的控制逻辑:

```

python
import keyboard

```

```

v = 0.0
w = 0.0

while True:
    if keyboard.is_pressed('w'):
        v += 0.1
    if keyboard.is_pressed('s'):
        v -= 0.1
    if keyboard.is_pressed('a'):
        w += 0.1
    if keyboard.is_pressed('d'):
        w -= 0.1
    # 发送控制指令
    udp_client.send_control_command(v, w)
    ...

```

3.3 物理现象分析

现象	原因	解决方案
车辆抖动	控制频率不稳定	优化控制循环
响应延迟	网络通信延迟	降低数据量
运动不连续	指令变化过大	添加平滑处理

4. 实验问题及总结

4.1 遇到的问题

- 初始连接失败
 - 原因：端口号配置错误
 - 解决：核对实验平台文档
- 车辆不移动
 - 原因：控制指令超出范围
 - 解决：调整 v, w 参数范围
- 数据接收不稳定
 - 原因：网络丢包
 - 解决：增加数据校验

4.2 实验收获

- 掌握了 UDP 通信原理
- 理解了实时控制系统设计
- 熟悉了多智能体协同控制基础

4.3 不足之处

1. 控制算法有待优化
2. 异常处理不够完善
3. 可视化程度不足

5. 改进方向

1. 增加 PID 控制算法
2. 完善异常处理机制
3. 添加实时可视化界面
4. 支持多车协同控制
