重庆邮电大学

学生课程设计报告册

学年学期：2023-2024学年 ■春□秋学期

课程名称：智能车辆决策与控制技术

学生学院：自动化学院

专业班级：08902101

学生学号：2021212923

学生姓名：陈亦杨

学生成绩：

指导教师：黄鑫

重庆邮电大学教务处制

智能车辆决策与控制技术课程设计报告
成绩评定参考说明

A. 是否根据要解决的问题进行了理论分析。

B. 是否根据所要解决的问题建立数学模型。

C. 是否有具体的设计过程，是否对设计方案进行分析比较。

D. 是否有具体的仿真平台搭建过程及软件编程过程。

E. 是否有具体的仿真参数选定过程或实物参数调整过程。

F. 仿真结果或硬件效果是否丰富。

G. 是否有对仿真结果或硬件输出结果的具体分析。

H. 是否很好地解决了本次课程设计提出的问题。

I. 是否有对本次课程设计的心得体会以及对相关问题的展望。

J. 整个设计报告格式是否规范，目录要自动链接，公式编号及引用要自动链接。

K. 在指导教师出题及设计要求的大框架下，根据自己的具体设计情况（扩展、就某一方面的独特研究、就某一方面的创新），自己拟定设计题目、设计内容及要求的学生有加分奖励。

L. 课程设计最终成绩按等级制计算并提交，百分制与等级制的换算见附表。

附表：成绩与等级的换算关系

成绩	≥90	80~89	70~79	60~69	＜60
等级	A（优）控制在 15%左右，每组可不止一个	B+（良）	B-（中）	C（及格）	F（不及格）

一、 课程设计任务及要求

设计题目	基于 Carla 的自动驾驶车辆路径跟踪仿真

设计任务：

1. 安装 Carla 仿真软件，并配置好相关依赖项。
2. 配置摄像头、GPS、激光雷达等相关的传感器（至少一种）。
3. 自定义 Carla 自带的地图作为仿真场景。
4. 学习一种路径规划算法（强制），可实现简易的避障功能（可选）。
5. 学习至少一种路径跟踪控制算法，与上面的路径规划算法一起加载到 Carla 里面进行仿真测试。
6. 在 Carla 里面对至少两种典型交通场景（直行、变道等）进行测试。

提取车辆速度、加速度、航向角等相关数据进行分析。

设计要求：

. 1.自动驾驶车辆路径跟踪的平均横向误差应该小于 0.1m;

. 2.车辆的加速度的绝对值不大于 6。

二、 人员及分工

姓名	学号	班级	设计分工	联系电话
卞胜强	2021213376	08902102	仿真环境搭建 路径规划、结果展示	13356501640
陈亦杨	2021212923	08902101	仿真运动控制与 PID 调参	15171318216
杨宇	2021213341	08902102	设计概述与仿真结果分析	18270075250
曾万琪	2021213267	08902101	原理分析	17783699532
李金祥	2021213064	08902102	仿真建模与设计总结	15086618159

三、课程设计说明书

目　录

摘　要

本次实验旨在通过 **Carla 仿真平台**，对自动驾驶车辆的路径跟踪性能进行深入研究与仿真测试。实验的主要研究内容包括 Carla 仿真软件的安装与配置、相关传感器的配置、**自定义地图**的创建、**路径规划算法**的学习与实现、路径跟踪控制算法的学习与测试，以及在不同交通场景下的仿真实验。

首先，安装 Carla 仿真软件，并配置好所有必要的依赖项，以确保仿真环境的稳定运行。随后，将配置至少一种传感器，以模拟自动驾驶车辆的感知能力。接下来，利用 Carla 自带的地图编辑器，自定义至少两种典型的交通场景，并在这些场景中设置静态和动态障碍物，以模拟真实道路环境。

在**路径规划**方面，本次实验使用 Carla 的**路径规划器**来实现路径规划，本次实验选用 A*算法，该算法能够基于传感器感知到的环境信息，动态生成从起始点到目标点的无碰撞路径。

在路径跟踪控制方面，本次实验使用 **PID 控制算法**作为路径跟踪控制算法，并将其与**路径规划算法**相结合，以实现自动驾驶车辆的路径跟踪功能。然后，将这两种算法加载到 Carla 仿真环境中，进行集成测试。

最后，在自定义的交通场景中对自动驾驶车辆进行仿真测试，并提取车辆的速度、加速度、航向角等相关数据进行分析。通过数据分析，项目将评估自动驾驶车辆的**路径跟踪性能**，并验证其是否满足主要技术要求指标。

通过本次实验，为自动驾驶车辆路径跟踪技术的研究提供有价值的参考和依据，进一步推动自动驾驶技术的发展和应用。

关键词：Carla，自动驾驶仿真，路径规划，路径跟踪，PID

1 设计说明

1.1 设计概述

本实验旨在通过 Carla 仿真平台，实现自动驾驶车辆的路径跟踪仿真测试。路径跟踪作为自动驾驶系统的核心功能之一，对于保证车辆安全、高效地行驶至关重要。本实验将通过配置传感器、自定义地图、使用路径规划算法和路径跟踪控制算法等步骤，对自动驾驶车辆在不同交通场景下的路径跟踪性能进行深入研究。

1.1.1 实验目标

1. 自动驾驶车辆路径跟踪性能目标

(1) 确保自动驾驶车辆在路径跟踪过程中的平均横向误差小于 0.1m，以提高行驶精度和安全性。

(2) 在不同交通场景和初始条件下，保持稳定的路径跟踪性能。

2. 车辆动力学性能目标

(1) 控制车辆的加速度绝对值不大于 $6m/s^2$，以确保行驶过程中的舒适性和安全性。

(2) 在紧急制动或避障等情况下，确保车辆的制动距离和制动时间满足安全要求。

1.1.2 实验步骤

1. Carla 仿真环境搭建

(1) 安装 Carla 仿真软件，并配置好相关依赖项，确保 Carla 能够正常运行。

(2) 了解和熟悉 Carla 的基本操作和功能，为后续工作奠定基础。

2. 传感器配置

(1) 至少配置一种传感器，如摄像头、GPS 或激光雷达，以模拟自动驾驶车辆的感知能力，本次实验选用的传感器为 camera。

(2) 根据实验需求，调整传感器的参数和位置，确保其在仿真中的有效性。

3. 地图自定义

(1) 利用 Carla 自带的地图编辑器，自定义仿真场景。

(2) 设计并构建至少两种典型交通场景，如直行、变道等，并设置相应的道路标志、交通信号灯等。

(3) 在自定义地图中设置静态和动态障碍物，以模拟真实道路中的复杂交通环境。

4. 路径规划算法学习与实现

(1) 学习一种路径规划算法，如 A*、Dijkstra 或 RRT 等，并实现其代码。

(2) 根据传感器感知到的环境信息，动态生成从起始点到目标点的无碰撞路径。

(3) 实现简易的避障功能，使自动驾驶车辆能够避开障碍物并安全行驶。

5. 路径跟踪控制算法学习与实现

(1) 学习至少一种路径跟踪控制算法，如 PID、MPC（模型预测控制）或 LQR（线性二次型调节器）等，并实现其代码。

(2) 将路径规划算法与路径跟踪控制算法结合使用，实现自动驾驶车辆的路径跟踪功能。

(3) 将上述算法加载到 Carla 仿真环境中，进行集成测试。

6. 仿真测试

(1)在自定义的两种典型交通场景中，对自动驾驶车辆进行仿真测试。

(2)设置不同的初始条件（如起始位置、速度等）和交通环境（如车流量、障碍物密度等），以模拟真实世界的复杂性。

(3)记录并保存仿真过程中的车辆状态数据，如速度、加速度、航向角等。

7. 数据分析

(1)提取并整理仿真测试过程中记录的数据。

(2)对车辆速度、加速度、航向角等相关数据进行分析，评估自动驾驶车辆路径跟踪的性能。

(3)绘制相关图表，展示关键指标的变化趋势和分布情况。

1.1.3 实验意义

1. 技术验证与评估：通过基于 Carla 的自动驾驶车辆路径跟踪仿真，可以对自动驾驶技术的关键组成部分——路径规划算法和路径跟踪控制算法进行验

证和评估。这种仿真测试有助于实验人员在实际部署之前发现和修复潜在的问题，从而提高自动驾驶系统的性能和安全性。

2. 场景适应性测试：实验要求在 Carla 中测试至少两种典型交通场景（直行、变道等），这有助于评估自动驾驶系统在不同驾驶场景下的适应性和鲁棒性。通过模拟各种复杂场景，实验人员可以更好地了解系统的局限性和改进方向。
3. 数据收集与分析：实验要求提取车辆速度、加速度、航向角等相关数据进行分析，这些数据对于理解自动驾驶系统的行为和改进算法至关重要。通过对这些数据的分析，实验人员可以找出系统中的瓶颈和优化点，为后续的研究和开发提供有力支持。
4. 教育与研究：本次实验不仅是一个技术验证项目，还是一个很好的教育和研究平台。它可以让学生深入了解自动驾驶技术的核心概念和算法实现，提高实践能力和创新能力。
5. 推动自动驾驶技术的发展：通过本次实验，可以推动自动驾驶技术在路径规划和路径跟踪控制方面的进步。这些技术的进步不仅可以提高自动驾驶系统的性能和安全性，还可以降低其制造成本和维护成本，推动自动驾驶技术的商业化和普及化。
6. 促进交通安全：自动驾驶技术有望显著提高道路交通的安全性。通过本次实验，可以验证自动驾驶车辆在路径跟踪方面的性能，为未来的自动驾驶车辆在实际道路上的应用提供技术支持和保障。这对于减少交通事故、保护人民生命财产安全具有重要意义。

1.2 原理分析

1.2.1 车辆动力学模型

当车辆速度很高时，单车模型中前后轮的速度矢量不再与轮子方向一致。此时运动学模型就不能准确地描述车辆的运动状态，这就需要使用动力学模型对车辆进行建模。

车辆单车模型中需要考虑两个维度的信息，这两个维度分别指代表车辆横向位置信息的 y 和表示车辆偏航角信息的 。下面分析过程中，先不考虑路堤角度的影响。

受力分析

平动

首先假设车辆为一个质点，对该质点进行受力分析，并根据牛顿第二定律得

$$ma_y = F_{yf} + F_{yr} \tag{1}$$

其中，a_y为车辆重心处 y 轴方向的惯性加速度，F_{yf}和F_{yr}为前后轮横向受到的力。平动过程中，有两种力共同作用于加速度a_y：车辆延 y 轴产生的惯性加速度$\ddot{y}$和车辆绕旋转中心 O 旋转产生的向心加速度$a_c = V_X\dot{\psi}$。

$$a_y = \ddot{y} + V_X\dot{\psi} \tag{2}$$

将公式(2)带入公式(1)得

$$m\left(\ddot{y} + V_X\dot{\psi}\right) = F_{yf} + F_{yr} \tag{3}$$

转动

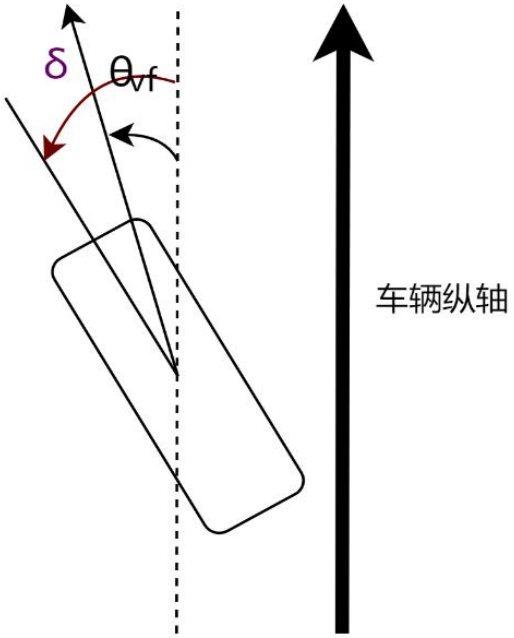
假设车辆为刚体，刚体绕重心转动，该运动过程使用力矩和转动惯量进行描述。车辆绕 z 轴旋转产生的力矩平衡，对应的偏航动力学方程为

$$I_Z\ddot{\psi} = I_f F_{yf} - I_r F_{yr} \tag{4}$$

其中I_f和I_r代表前后轮胎到重心的距离。

受力计算

上述等式(3)和(4)中都用到了轮胎横向受力情况F_{yf}和F_{yr}，根据实验结果知，轮胎的横向力与小的滑移角存在正比例的关系,滑移角是轮胎方向与车轮速度矢量之间的夹角。



根据上图可知

$$\alpha_f = \delta - \theta_{Vf} \tag{5}$$

其中，θ_{Vf}代表速度矢量与车辆纵轴的夹角，δ \delta δ代表前轮转向角。

同理，由于后轮转向角δ为0，故后轮滑移角为

$$\alpha_r = -\theta_{Vf} \tag{6}$$

车辆前轮的横向力可以表示为

$$F_{yf} = 2C_{\alpha f}(\delta - \theta_{Vf}) \tag{7}$$

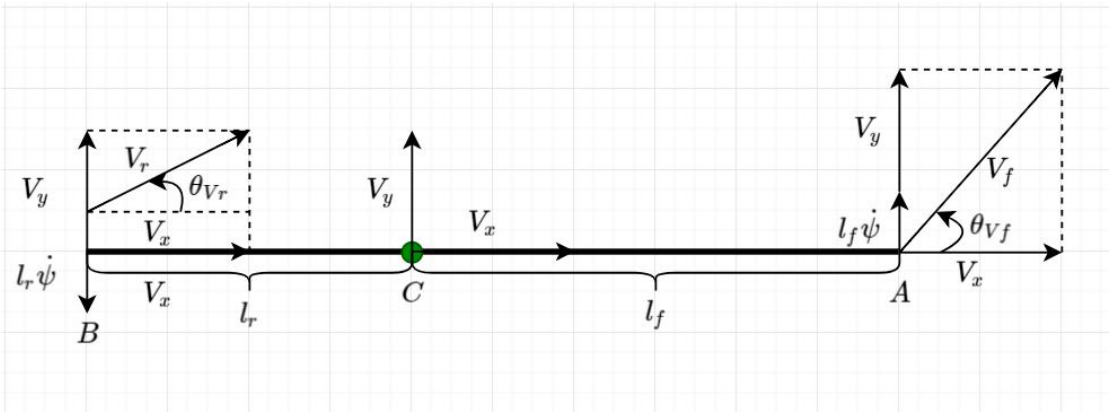
其中，比例常数$C_{\alpha r}$代表每个前轮的侧偏刚度。

同理后轮的横向力可以写为

$$F_{yr} = 2C_{\alpha r}(-\theta_{Vr}) \tag{8}$$

其中，比例常数$C_{\alpha r}$代表每个后轮的侧偏刚度。

速度方向



车辆平动产生的速度分量V_x和V_y，以及绕点 C 转动产生的线速度$I_f\dot{\psi}$和$I_r\dot{\psi}$组成。根据上图得

$$\tan(\theta_{Vf}) = \frac{V_y + I_f\dot{\psi}}{V_x} \tag{9}$$

$$\tan(\theta_{Vr}) = \frac{V_y - I_r\dot{\psi}}{V_x} \tag{10}$$

由于通常情况下速度矢量的夹角很小，可以使用小角度近似原理得

$$\theta_{Vf} = \frac{\dot{y} + I_f\dot{\psi}}{V_x} \tag{11}$$

$$\theta_{Vr} = \frac{\dot{y} - I_r\dot{\psi}}{V_x} \tag{12}$$

微分方程

将等式(7)、(8)、(9)和(10)代入等式(3)中得

$$m\left(\ddot{y} + V_x\dot{\psi}\right) = 2C_{\alpha f}\left(\delta - \frac{\dot{y} + I_f\dot{\psi}}{V_x}\right) + 2C\alpha r\left(-\frac{\dot{y} - I_r\dot{\psi}}{V_x}\right) \tag{13}$$

等式(13)左右两边同时除以 m,分别提取$\ddot{y}$、$\dot{y}$、$\dot{\psi}$和δ项得

$$\ddot{y} = -\frac{2C_{\alpha f} + 2C_{\alpha r}}{mV_x}\dot{y} - \left(V_x + \frac{2C_{\alpha f}I_f - 2C_{\alpha r}I_t}{mV_x}\right)\dot{\psi} + \frac{2C_{\alpha f}}{m}\delta \tag{14}$$

转成矩阵形式如下

$$\frac{d}{dt}\dot{y}=\begin{bmatrix}0 & -\dfrac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -\left(V_x+\dfrac{2C_{\alpha f}I_f-2C_{\alpha r}I_r}{mV_x}\right)\end{bmatrix}\begin{bmatrix}y\\ \dot{y}\\ \psi\\ \dot{\psi}\end{bmatrix}+\frac{2C_{\alpha f}}{m}\delta \tag{15}$$

同理，将等式(7)、(8)、(9)和(10)代入等式(4)中得

$$I_z\ddot{\psi}=2I_fC_{\alpha f}\left(\delta-\frac{\dot{y}+I_f\dot{\psi}}{V_x}\right)-2I_rC_{\alpha r}\left(-\frac{\dot{y}-I_r\dot{\psi}}{V_x}\right) \tag{16}$$

等式(13)左右两边同时除以I_z，分别提取$\dot{y}$、$\ddot{\psi}$、$\dot{\psi}$和δ项可得

$$\ddot{\psi}=-\frac{2I_fC_{\alpha f}-2I_rC_{\alpha r}}{I_zV_x}\dot{y}-\frac{2I_f{}^2C_{\alpha f}+2I_r{}^2C_{\alpha r}}{I_zV_x}\dot{\psi}+\frac{2I_fC_{\alpha f}}{I_z}\delta \tag{17}$$

等效的矩阵形式为

$$\frac{d}{dt}\dot{\psi}=\begin{bmatrix}0 & -\dfrac{2I_fC_{\alpha f}-2I_rC_{\alpha r}}{I_zV_x} & 0 & -\left(\dfrac{2I_f{}^2C_{\alpha f}+2I_r{}^2C_{\alpha r}}{I_zV_x}\right)\end{bmatrix}\begin{bmatrix}y\\ \dot{y}\\ \psi\\ \dot{\psi}\end{bmatrix}+\frac{2I_fC_{\alpha f}}{I_z}\delta \tag{18}$$

根据等式(15)和(18)得

$$\frac{d}{dt}\begin{bmatrix}y\\ \dot{y}\\ \psi\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & -\dfrac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -\left(V_x+\dfrac{2C_{\alpha f}I_f-2C_{\alpha r}I_r}{mV_x}\right)\\ 0 & 0 & 0 & 1\\ 0 & -\dfrac{2I_fC_{\alpha f}-2I_rC_{\alpha r}}{I_zV_x} & 0 & -\dfrac{2I_f{}^2C_{\alpha f}-2I_r{}^2C_{\alpha r}}{I_zV_x}\end{bmatrix}\begin{bmatrix}y\\ \dot{y}\\ \psi\\ \dot{\psi}\end{bmatrix}+\begin{bmatrix}0\\ \dfrac{2C_{\alpha f}}{m}\\ 0\\ \dfrac{2I_fC_{\alpha f}}{I_z}\end{bmatrix} \tag{19}$$

上述动力学方程的推导建立在车辆滑移角很小的情况下，这时的轮胎作用力与滑移角可以近似为线性关系。当滑移角很大时，轮胎作用力与滑移角就不再是线性关系。

1.2.2 路径规划

路径规划对于一辆无人驾驶车辆来说就是根据给定的环境模型，在一定的约束条件下规划出一条连接当前位置和目标位置的无碰撞路径。自动驾驶汽车的路径规划从功能上可以分为**全局路径规划**、**行为决策规划**和**局部运动规划**。

全局路径规划可以理解为实现自动驾驶汽车软件系统内部的导航功能，即在宏观层面上指导自动驾驶汽车软件系统的控制规划模块大致按照什么方向的道路走，从而引导车辆从起始点到达目标点。

行为决策规划可以简单地理解为自动驾驶汽车的“大脑”。全局路径规划模块产生的路径信息，直接被下游的行为决策规划模块所使用。行为决策规划模块接收到全局路径规划的结果，同时也接收感知、预测和地图信息，综合这些输入信息，行为决策规划模块决定车辆该如何行驶，比如正常跟车或者变道、在遇见红绿灯或者行人时等待避让以及在路口和其他车辆的交互等。

局部运动规划对无人车辆的行驶起着精确导航的作用，其任务就是在寻找到的全局最佳路径和最优行为策略的前提下，根据车辆当前的几何形状和动力学模型、实时所处环境的分布情况以及一个目标状态集，找到一系列的控制输入，驱动汽车从初始状态运动到目标状态集中的某一状态，并且在运动过程中避免和障碍物发生碰撞同时满足车辆的动力学约束。

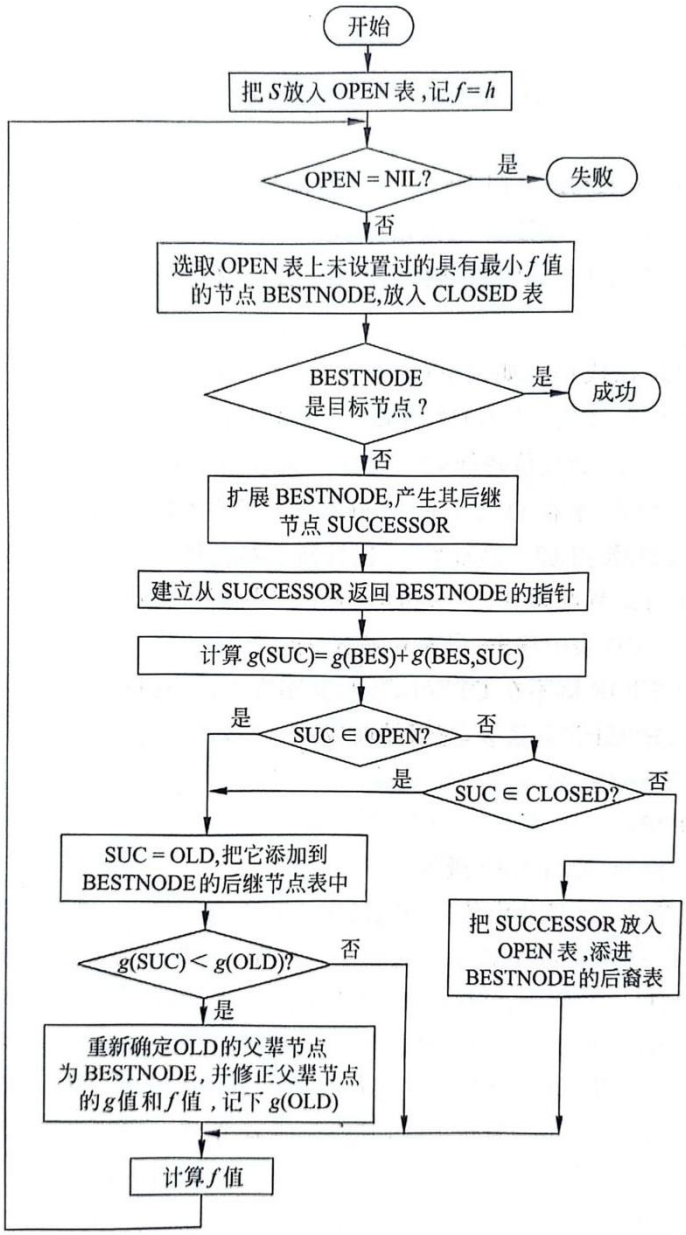
全局路径规划——A*算法

Astar 算法是一种启发式搜索算法，用于在图中寻找从起点到终点的最短路径。它的流程大致如下：

1. 将起点加入开放列表，该列表存储待扩展的节点。
2. 从开放列表中选出评价函数 f(n) 最小的节点 n，将其移出开放列表并加入关闭列表，该列表存储已经扩展过的节点。
3. 对 n 的所有邻居节点 m，进行如下操作：
   - 如果 m 是不可达的或者已经在关闭列表中，忽略它。
   - 如果 m 不在开放列表中，将其加入开放列表，并记录其父节点为 n，计算其评价函数 f(m)=g(m)+h(m)，其中 g(m)是从起点到 m 的实际代价，h(m)是从 m 到终点的估计代价。
   - 如果 m 已经在开放列表中，检查是否存在一条更优的路径经过 n 到达 m，即比较 g(n)+c(n,m)和 g(m)，其中 c(n,m)是从 n 到 m 的移动代价。如果前者更小，说明找到了更优的路径，更新 m 的父节点为 n，重新计算 f(m)和 g(m)。
4. 如果找到终点，沿着父节点回溯，得到最短路径。如果开放列表为空，说明没有可行的路径。
5. 重复步骤 2~4，直到找到终点或开放列表为空。

Astar 算法的关键在于选择合适的估计函数 h(n)，它可以影响算法的效率和最优性。常见的 h(n)有欧几里得距离、曼哈顿距离和对角线距离等。

A*算法流程框图如下图所示：



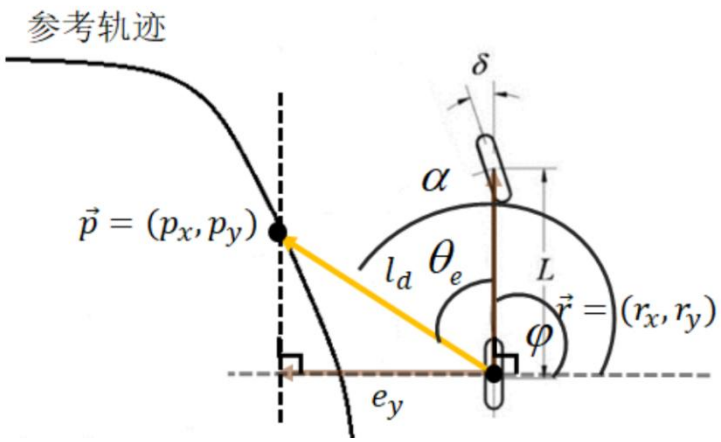
（A*算法相关代码见附录）

1.2.3 路径跟踪控制器——PID 控制器

该模块主要逻辑就是基于 PID 控制建立一个简单的规划。

横向跟踪误差

横向跟踪误差（cross track error，简称 CTE）为前轴中心点(r_x, r_y)到最近路径点(p_x, p_y)的距离，具体如下图所示。



其中，$\theta_e = \alpha - \varphi$

φ：无人车航向角

α：向量$(\vec{p}-\vec{r})$的角度

e_y：横向跟踪误差

l_d：前视距离

以上图为基础进行简略分析，如果参考轨迹点在无人车的左边$\theta_e \in [0,\pi]$，则应该向左打方向盘；反之$\theta_e \in [0,-\pi]$，则向右打方向盘。

经分析可得横向跟踪误差（CTE）计算公式如下：

$$e_y = l_d \sin\theta_e \tag{20}$$

其中，$l_d = \|\vec{p}-\vec{r}\|_2$为无人车后轴中心离当前路点的距离，也被称为前视距离。

传感器模拟——Carma

蓝图：sensor.camera.depth

输出：carla.Image

相机提供场景的原始数据，将每个像素到相机的距离（也称为深度缓冲区或 z 缓冲区）编码为创建元素的深度图。

该图像使用 RGB 色彩空间的 3 个通道（从较小到较高的字节）对每个像素的深度值进行编码：R -> G -> B。以米为单位的实际距离可以是解码方式：

```
normalized = (R + G * 256 + B * 256 * 256) / (256 * 256 * 256 - 1)
```

```
in_meters = 1000 * normalized
```

输出 carla.Image 应使用 carla.colorConverter 将图像保存到磁盘，该转换器会将存储在 RGB 通道中的距离转换为包含距离的[0,1]浮点数，然后将其转换为灰度。carla.colorConverter 中有两个选项可以获取深度视图：深度和对数深度。两者的精度都是毫米级的，但对数方法为更近的物体提供了更好的结果。

```
raw_image.save_to_disk("path/to/save/converted/image",carla.Depth
)
```

基本相机属性

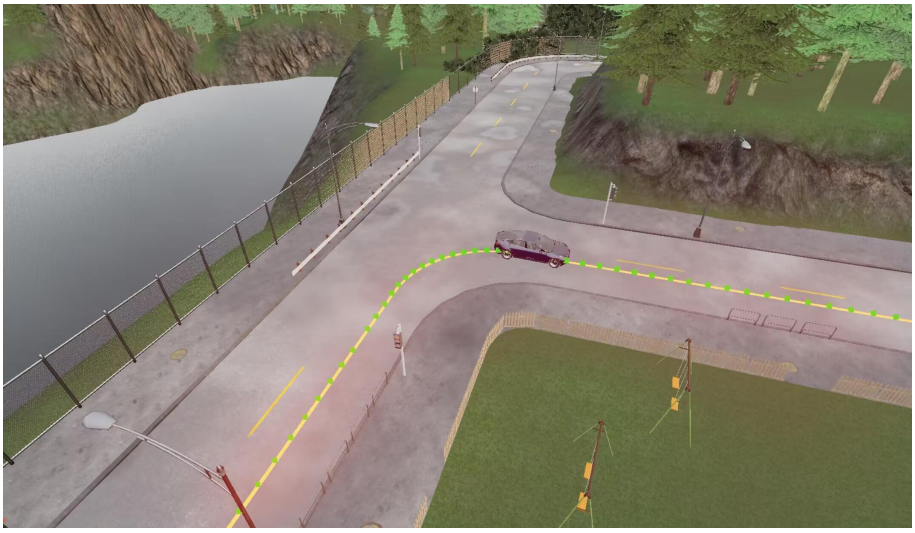
Blueprint 属性	类型	默认	描述
image_size_x	int	800	图像宽度（以像素为单位）
image_size_y	int	600	图像高度（以像素为单位）
fov	float	90.0	以度为单位的水平视野
sensor_tick	float	0.0	传感器捕获之间的模拟描述（刻度）

输出属性

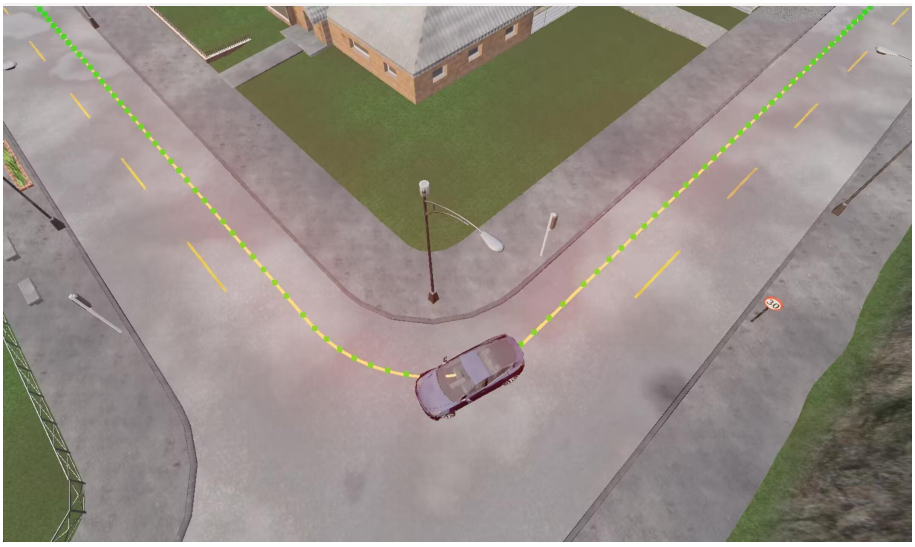
传感器数据属性	类型	描述
frame	int	测量时的帧号
timestamp	double	自剧集开始以来测量的模拟时间（以秒为单位）
transform	carla.Transform	测量时传感器在世界坐标中的位置和旋转
width	int	图像宽度（以像素为单位）
height	int	图像高度（以像素为单位）
fov	float	以度为单位的水平视野
raw_data	bytes	GRA 32 位像素数组

1.3 仿真建模

Carla 仿真小车直线行驶



Carla 仿真小车弯道行驶



Carla 仿真小车弯道行驶

1.4 仿真结果分析

本次实验基于 Carla 仿真软件进行了自动驾驶车辆路径跟踪的仿真测试。我们在 Carla 环境中对自动驾驶车辆在不同交通场景下的性能进行了评估，以验证项目的主要研究目标是否达成。

1.4.1 仿真环境设置

1. Carla 仿真软件安装及配置：成功安装 Carla 仿真软件，并配置了所有必要的依赖项，确保仿真环境稳定运行。
2. 传感器配置：根据项目要求，我们配置了摄像头（camera）作为主要的传感器，用于车辆的环境感知和定位。
3. 自定义地图：我们根据仿真需求，自定义了 Carla 自带的地图，并设置了多种交通场景，包括直行和变道等。

1.4.2 算法学习与实现

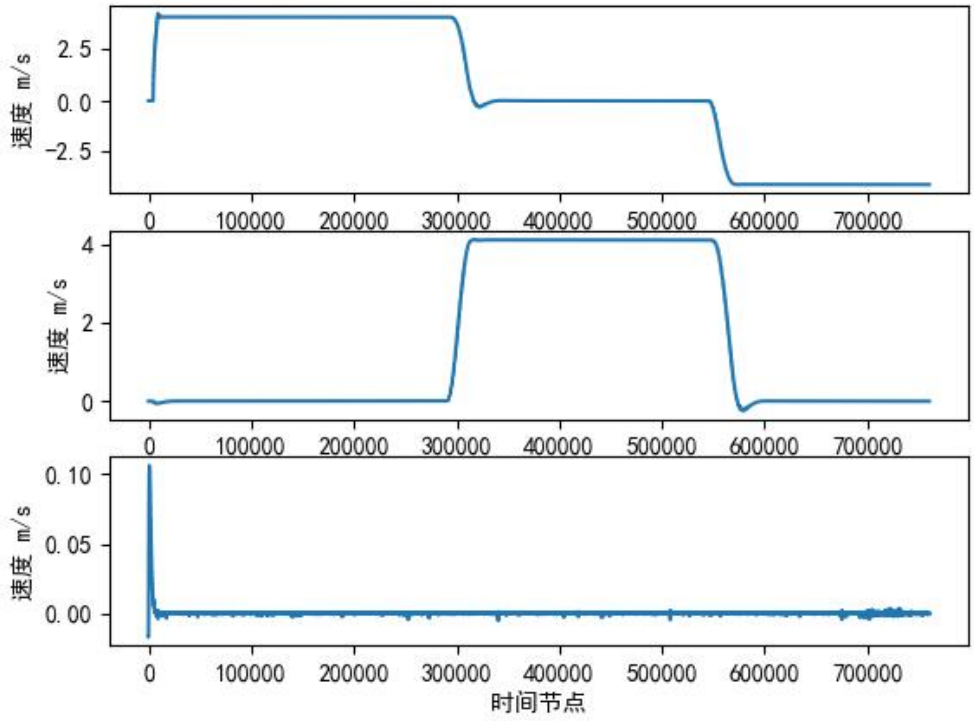
1. 路径规划算法：我们使用了 A*（A Star）算法，并实现了其简易的避障功能。A*算法在仿真中表现出良好的路径规划性能，能够有效地避开障碍物并规划出合理的行驶路径。
2. 路径跟踪控制算法：我们选择了 PID（Proportional-Integral-Derivative）控制算法作为路径跟踪控制算法。PID 算法在 Carla 仿真环境中实现了对规划路径的准确跟踪，保证了车辆的稳定性和安全性。

1.4.3 仿真测试与数据分析

1. 典型交通场景测试：我们在 Carla 仿真环境中对直行和变道两种典型交通场景进行了测试。在直行场景中，自动驾驶车辆能够稳定地沿着规划路径行驶，并保持较小的横向误差。在变道场景中，车辆能够准确地识别出变道指令，并在保证安全的前提下顺利完成变道操作。
2. 数据提取与分析：

   我们提取了仿真测试过程中车辆的速度、加速度、航向角等相关数据，并进行了详细的分析。

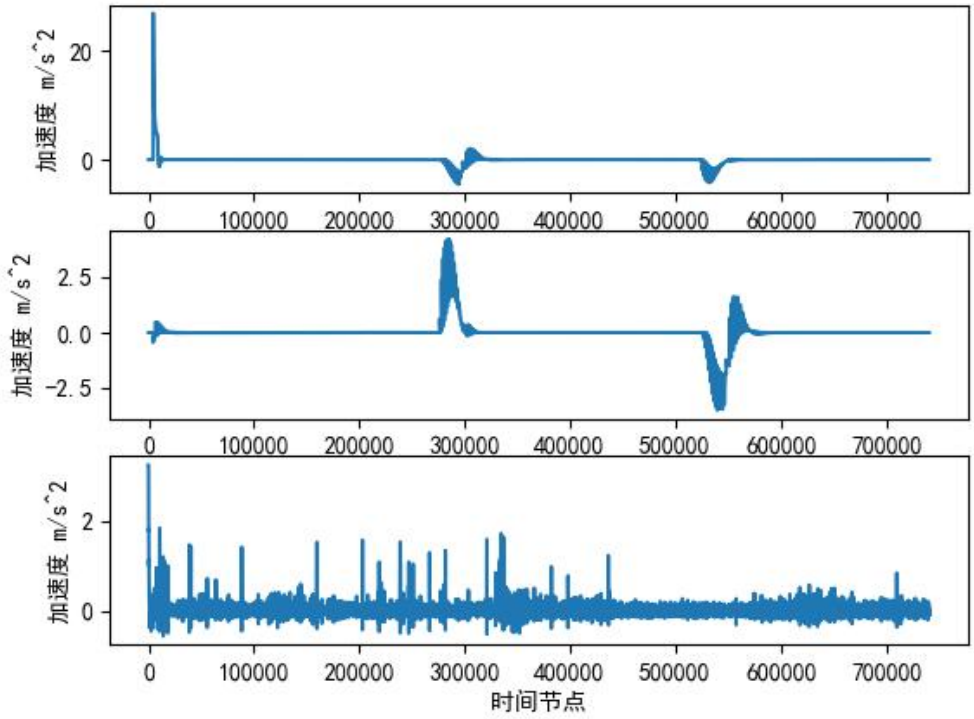
   (1)速度分析：



在直行场景中，自动驾驶车辆能够保持稳定的行驶速度，并在需要时适当调整速度以保持行驶安全。

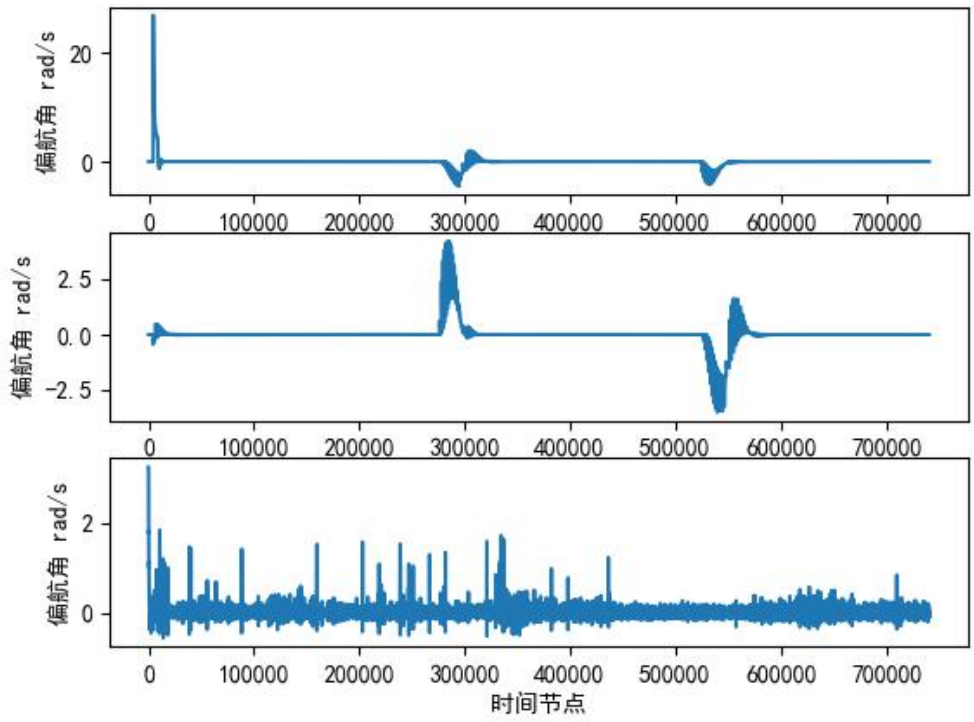
在变道场景中，车辆能够在确保安全的前提下，适当加速或减速以完成变道操作。总体来看，车辆的速度变化平稳，未出现剧烈波动或超速现象。

(2)加速度分析：



仿真汽车加速度的绝对值始终不超过 6/cm^3，并且在需要加速或减速的情况下，车辆的加速度变化迅速，反应灵敏。

(3)航向角分析：



车辆在行驶过程中，航向角能够准确反映车辆的行驶方向，与规划路径保持一致。

在变道或避障时，车辆能够及时调整航向角，确保行驶轨迹的平滑性和安全性。

(4)路径跟踪误差分析：

在直行场景中，自动驾驶车辆的路径跟踪误差较小，平均横向误差小于 0.1m，达到了项目的主要研究目标。

在变道场景中，由于需要调整行驶轨迹，路径跟踪误差会相对增大，但仍保持在可接受范围内。

结果表明，自动驾驶车辆在路径跟踪过程中，平均横向误差小于 0.1m，满足了项目的主要研究目标之一。同时，车辆的加速度绝对值始终保持在 $6m/s^2$ 以内，符合技术经济指标的要求。

1.4.4 结论

通过本次仿真测试，我们验证了基于 Carla 的自动驾驶车辆路径跟踪系统的性能。仿真结果显示，自动驾驶车辆在多种交通场景下均能够稳定地跟踪规划路径，并保持较小的横向误差和加速度。这表明我们学习并实现的路径规划算法和路径跟踪控制算法在 Carla 仿真环境中具有良好的表现。因此，我们可以认为本项目的主要研究目标已经达成，为自动驾驶技术的进一步研究提供了有力的支持。

2 设计总结

本次实验通过 Carla 仿真平台，系统地研究并测试了自动驾驶车辆的路径跟踪性能。实验涵盖了从仿真环境的搭建、传感器配置、地图自定义到路径规划和路径跟踪控制算法的学习与实现，以及在不同交通场景下的综合仿真实验和数据分析。实验的第一步是安装 Carla 仿真软件，并配置所有必要的依赖项，以确保仿真环境能够稳定运行。这一步为后续工作打下了坚实的基础。通过熟悉 Carla 的基本操作和功能，我们确保了整个实验过程的顺利进行。在传感器配置阶段，我们选择了摄像头（camera）作为实验传感器。该传感器能够模拟自动驾驶车辆的感知能力，实时捕捉周围环境的信息。根据实验需求，我们调整了摄像头的参数和位置，以确保其在仿真中的有效性。这为后续的路径规划和路径跟踪提供了可靠的数据支持。

利用 Carla 自带的地图编辑器，我们创建了至少两种典型的交通场景，并设置了静态和动态障碍物，以模拟真实道路环境。通过设计直行、变道等交通场景，并配置相应的道路标志和交通信号灯，我们为路径规划和路径跟踪控制算法的测试提供了多样化的测试环境。在路径规划方面，我们选择了 A 算法。这是一种基于图搜索的经典算法，能够基于传感器感知到的环境信息，动态生成从起始点到目标点的无碰撞路径。通过学习和实现 A 算法，我们实现了简易的避障功能，使自动驾驶车辆能够避开障碍物并安全行驶。这为车辆的路径跟踪奠定了基础。为了实现路径跟踪功能，我们学习并实现了 PID 控制算法。PID 控制算法以其简单、高效的特点，广泛应用于工业控制领域。在本次实验中，我们将路径规划算法与 PID 控制算法结合使用，实现了自动驾驶车辆的路径跟踪功能。随后，我们将这两种算法加载到 Carla 仿真环境中，进行了集成测试，验证了算法的实际效果。

在自定义的交通场景中，我们对自动驾驶车辆进行了多次仿真测试。通过设置不同的初始条件（如起始位置、速度等）和交通环境（如车流量、障碍物密度等），我们模拟了真实世界的复杂性。在测试过程中，我们记录并保存了车辆的速度、加速度、航向角等状态数据。随后，通过数据分析，我们评估了自动驾驶车辆的路径跟踪性能，并验证了其是否满足主要技术要求指标。

本次实验通过 Carla 仿真平台，对自动驾驶车辆的路径跟踪性能进行了系统研究与测试，涵盖了仿真环境搭建、传感器配置、地图自定义、路径规划和路径跟踪控制算法的学习与实现。通过在不同交通场景下的综合仿真实验和数据分析，我们验证了路径规划与 PID 控制算法的有效性，评估了车辆的路径跟踪性能，并发现了潜在问题和优化点。实验不仅验证了关键技术，还为教育、研究和推动自动驾驶技术的发展提供了宝贵的经验和数据支持，对未来实际应用具有重要意义。

3 程序设计

```
#-*- coding: utf-8 -*-
from __future__ import print_function

import argparse
import collections
import datetime
import glob
import logging
import math
import os
import numpy.random as random
import re
import sys
import weakref

try:
    import pygame
    from pygame.locals import KMOD_CTRL
    from pygame.locals import K_ESCAPE
    from pygame.locals import K_q
except ImportError:
    raise RuntimeError('cannot import pygame, make sure pygame package is installed')

try:
    import numpy as np
except ImportError:
    raise RuntimeError(
        'cannot import numpy, make sure numpy package is installed')

# ==============================================================================
# -- Find CARLA module ---------------------------------------------------------
# ==============================================================================
try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
```

```
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])

except IndexError:
    pass

# ==============================================================================
# -- Add PythonAPI for release mode --------------------------------------------
# ==============================================================================
try:
    sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))) + '/carla')
except IndexError:
    pass

import carla
from carla import ColorConverter as cc

import sys
sys.path.append("I:\\Tools\\carla\\WindowsNoEditor\\PythonAPI\\carla")
sys.path.append("I:\\Tools\\carla\\WindowsNoEditor\\PythonAPI\\carla\\agents")

from agents.navigation.global_route_planner import GlobalRoutePlanner
from agents.navigation.controller import VehiclePIDController
from agents.tools.misc import draw_waypoints, distance_vehicle, vector, is_within_distance, get_speed
from agents.navigation.basic_agent import BasicAgent
from agents.navigation.controller import VehiclePIDController, PIDLongitudinalController

# 这里是连接 Carla 服务器
client = carla.Client('localhost', 2000)
client.set_timeout(100.0)  # seconds

client.load_world('Town01')
world = client.get_world()

#dt = 0.05
```

```
70.  #settings = world.get_settings()
71.  #settings.fixed_delta_seconds = dt # Set a variable time-step
72.  #world.apply_settings(settings)
73.
74.  m = world.get_map()
75.  spawn_points = m.get_spawn_points()   #生成点
76.
77.  # global path planner
78.  origin = carla.Location(spawn_points[150].location)  #定义起始点
79.  destination = carla.Location(spawn_points[50].location)  #定义目标
     点
80.
81.  # spawn ego vehicle 生成主车，定义生成的点
82.  blueprint_library = world.get_blueprint_library()   #蓝图库
83.  ego_bp = blueprint_library.find('vehicle.tesla.model3')
84.  ego = world.spawn_actor(ego_bp, spawn_points[150])
85.
86.  # 定义观察视角
87.  spectator = world.get_spectator()    #设置观察视角
88.  ego_transform = ego.get_transform()
89.  #asshole spectator
90.  '''''transform1 = carla.Transform(ego_transform.transform(carla.L
     ocation(x=-6, z=2.5)),
91.                                    ego_transform.rotation)'''
92.  #overlooking spectator
93.  transform1 = carla.Transform(ego_transform.location + carla.Locat
     ion( z = 20),
94.        carla.Rotation(pitch = -90) )
95.  spectator.set_transform(transform1)
96.
97.  distance = 1
98.  grp = GlobalRoutePlanner(m, distance)   #全局路径规划算法
99.
100. """
101. This method returns list of (carla.Waypoint, RoadOption)
102. from origin to destination
103. """
104. route = grp.trace_route(origin, destination)
105.
106. wps = []    # waypoint 的集合
107. for i in range(len(route)):
108.     wps.append(route[i][0])
109.
110. """
```

```
    Draw a list of waypoints at a certain height given in z.
        :param world: carla.world object
        :param waypoints: list or iterable container with the waypoints to draw
        :param z: height in meters
    """
    draw_waypoints(world, wps)

    #绘制点和曲线
    for pi, pj in zip(route[:-1], route[1:]):      #[:-1]从第一个元素开始到倒数第二个，[1: ]从第二个开始到最后，pi 和 pj 刚好成连续的前后两个点
        #zip 函数将两个数组打包为元组，((route[0],route[1]),(route[1],route[2]),...,)
        pi_location = pi[0].transform.location
        pj_location = pj[0].transform.location
        pi_location.z = 0.5
        pj_location.z = 0.5
        world.debug.draw_line(pi_location, pj_location, thickness=0.1, color=carla.Color(r=255))  #life_time=T, 绘制线条
        pi_location.z = 0.6
        world.debug.draw_point(pi_location, color=carla.Color(g=255))  #, life_time=T  绘制点

def condistance(location1,location2):
    dx = location1.x - location2.x
    dy = location1.y - location2.y
    dz = location1.z - location2.z
    return math.sqrt(dx*dx + dy*dy + dz*dz)


#ego.apply_control(carla.VehicleControl(throttle=5.0, steer=0))
agent = BasicAgent(ego)
agent.set_destination(destination)

k = 0
epoch = 5000
acceleration = []
velocity = []
angular_velocity = []
while k < len(route):
    if agent.done():
        print("The target has been reached, stopping the simulation")
        break
```

```
149.
150. #PID 参数设置
151.     initspd_pid = PIDLongitudinalController(ego, K_P=3, K_I=0.1,
     K_D=2.0)
152.     # 设置目标车的 cut_in 的横纵向控制 PID
153.     args_lateral_dict = {'K_P': 1.0, 'K_D': 2.0, 'K_I': 0.0, "dt":
     1.0/1.0}
154.     args_long_dict = {'K_P': 1.0, 'K_D': 2.0, 'K_I': 0.0, "dt": 1.
     0/1.0}
155.     PID = VehiclePIDController(ego, args_lateral_dict, args_long_
     dict)
156.     target_speed = 15
157.
158.     control = PID.run_step(target_speed, wps[k])
159.     ego.apply_control(control)
160.
161.     velocity.append(ego.get_velocity())
162.     acceleration.append(ego.get_acceleration())
163.     angular_velocity.append(ego.get_angular_velocity())
164.
165.     target_location = wps[k].transform.location
166.     if condistance(ego.get_transform().location, target_location)
     <= 2.0:
167.         k += 1
168.
169.     #epoch -= 1
170.     #if epoch == 0:
171.     #    break
172.
173. # 画图
174. import matplotlib.pyplot as plt
175. plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
176. plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
177. v_x = []
178. v_y = []
179. v_z = []
180. a_x = []
181. a_y = []
182. a_z = []
183. ywd_x = []
184. ywd_y = []
185. ywd_z = []
186. for data in velocity:
187.     v_x.append(data.x)
```

```
    v_y.append(data.y)
    v_z.append(data.z)

for data in acceleration:
    a_x.append(data.x)
    a_y.append(data.y)
    a_z.append(data.z)

for data in acceleration:
    ywd_x.append(data.x)
    ywd_y.append(data.y)
    ywd_z.append(data.z)

velocity_xyz = [v_x, v_y, v_z]
plt.figure(1)
for i in range(3):
    plt.subplot(3, 1, i+1)
    plt.plot(np.array(velocity_xyz[i]))
    plt.xlabel('时间节点')
    plt.ylabel('速度 m/s')
plt.show()


plt.figure(2)
acceleration_xyz = [a_x, a_y, a_z]
for i in range(3):
    plt.subplot(3,1, i+1)
    plt.plot(np.array(acceleration_xyz[i]))
    plt.xlabel('时间节点')
    plt.ylabel('加速度 m/s^2')
plt.show()

plt.figure(3)
angular_velocity_xyz = []
for i in range(3):
    plt.subplot(3, 1, i+1)
    plt.plot(np.array(acceleration_xyz[i]))
    plt.xlabel('时间节点')
    plt.ylabel('偏航角 rad/s')
plt.show()
```