河 北 工 业 大 学

毕业设计说明书

作　　者：邢王源　　学　号：170738

学　　院：机械工程学院

系(专业)：车辆工程

题　　目：基于 CARLA 的无人赛车轨迹优化及控制仿真研究

指导者：王刚　　副教授
(姓 名)　　(专业技术职务)

指导者：许晟杰　　工程师
(姓 名)　　(专业技术职务)

评阅者：
(姓 名)　　(专业技术职务)

2021 年 5 月 24 日

毕 业 设 计 (论 文) 中 文 摘 要

基于 CARLA 的无人赛车轨迹优化及控制仿真研究

摘要：

目前对于智能驾驶汽车的研究多集中于城市交通环境，然而进行赛车方面的研究有助于提高驾驶速度，同时也便于了解高速工况下的安全性问题。因此本文使用雷山赛车场验证赛车优化轨迹生成算法，首先在 CARLA 中新建了雷山赛道，随后通过编写 Python 脚本计算轨迹生成算法所需的详细赛道数据。接下来以获取到的赛道信息为输入，以单轨车辆动力学模型为基础，使用顺序近似的方法依次求解速度分布及轨迹分布。该算法绕过了最快圈速优化时求解非凸成本函数的复杂性，从而以较低的计算时间生成车辆轨迹。最后在 CARLA 中应用基于车辆几何模型的控制算法，进行了轨迹跟踪的横向及纵向控制。本课题的研究内容提供了在虚拟环境下进行无人驾驶赛车算法开发以及车辆仿真的可行方法，在无人驾驶车辆路径规划方面也有可借鉴的部分。

关键词：赛车轨迹优化　车辆动力学模型　CARLA 模拟器仿真　PurePursuit 控制　Stanley 控制　PID 控制

毕业设计（论文）外文摘要

Title The Research on Trajectory Optimization and Control Simulation for Driverless Racecar Based on CARLA

Abstract

At present, the research on self-driving vehicles is mostly focused on the urban traffic environment. However, the research on racing cars is helpful to improve the driving speed and to understand the safety issues under high speed conditions. This paper uses Thunderhill raceway to verify the optimized trajectory generation algorithm for the racecar. The Thunderhill raceway is first created in CARLA, and then the detailed raceway data required by the trajectory generation algorithm is calculated by writing a Python script. The obtained raceway information is then used as input to solve for the speed profile and trajectory in turn, using a sequential approximation based on the bicycle model. The algorithm bypasses the complexity of solving a non-convex cost function for the fastest lap time and thus generates vehicle trajectories in a lower computational time. Finally, a vehicle geometry model-based control algorithm is applied in CARLA for lateral and longitudinal control of the trajectory tracking. This topic provides a feasible method for the development of driverless racing algorithm and simulation in the virtual environment, it also has a part to play in the path planning of driverless vehicles.

Keyword: trajectory optimization vehicle dynamic model CARLA simulation PurePursuit control Stanley control PID control

目　　录

1 引言

1.1 研究背景及意义

随着科技的迅速发展，无人驾驶技术也日趋成熟，这一技术主要涵盖了环境感知、路径规划与跟踪控制几个方面，其中路径规划与跟踪控制是保证无人驾驶车辆能够稳定行驶的关键技术。

无人驾驶赛车将赛车作为载体，而对这一方面的技术进行研究有助于充分利用赛车性能。为了在最短的时间内完成一场比赛，赛车驾驶员几乎百分之百地利用车辆轮胎和路面之间的摩擦力，专业驾驶员非常擅长协调制动、油门和转向输入，用以最大限度地提高车辆在赛道上的速度，同时保持车辆轮胎在摩擦极限内。因此，以无人驾驶赛车为研究对象进行路径规划与跟踪控制方面的探索有较大意义。

利用轨迹生成算法生成与专业赛车手较为贴合的赛车轨迹能够对实际的比赛过程进行预估，算法的结果也为赛车手的驾驶路线提供了参考。同时，以轨迹生成算法为基础可以研究车辆参数改变对于轨迹分布、速度分布以及圈速的影响，从而不断提高比赛成绩。由于轨迹生成算法得到的速度分布速度较高，因此可在此基础上研究基于高速极限情况下的控制算法，这对于普通车辆在危险情况下安全问题的解决也有借鉴意义。总的来说，本课题的研究的内容一方面可为无人驾驶方程式赛车开发提供理论基础和技术支持，从而发挥赛车动态性能取得优异成绩；另一方面，研究结果对无人驾驶赛车路径规划与跟踪控制部分技术发展有助力作用。

1.2 课题相关技术研究现状

路径规划是无人驾驶汽车的重要组成模块，也是汽车实现无人驾驶的基础，其功能是在当前工作环境中按照某种评价指标搜索出一条最优路径[1]。而路径跟踪问题的重点在于编写控制算法，从而使无人驾驶汽车具有基本无误差地沿目标路径行驶的能力。以上两方面都是无人驾驶汽车能够正常行驶的基础，因此研究人员们针对这两方面的问题做了大量的研究，也提出了很多解决方案。

1.1.1 无人车轨迹生成研究概述

无人车轨迹规划的核心技术主要包括：基于图搜索算法，基于随机搜索算法，基于曲线插值的方法，以及基于数值优化方法[2,3]，下文将对这四种技术的研究现状进行详细介绍。

图搜索方法最初应用在计算机领域，是主要用于处理“图”这种数据结构的算法，这一算法提供了一种分析连接数据最有效的途径，即在节点之间关系的基础上对复杂系统的体系和变化规律进行推断，最终得到的路径以最优评价为指标。全局路径规划系统主要使用图搜索算法，例如考虑时间维度的A*算法、Hybrid A*算法、D*算法及一系列衍生算法，此外在考虑时间、速度两个维度的情况下，可使用时空栅格进行局部路径规划[4]。在轨迹规划中较为经典的算法为Dijkstra算法，在这一算法的搜索方式中，广度被当做第一考虑项，接下来以路径长度依次增加为标准生成距离最短的轨迹。为了提高算法的速度，A*算法在 Dijkstra 的基础上进行了优化，即在 Dijkstra 的代价估算中引入节点到目标点的启发值[5]。但是由于车辆转向机构的限制，并不能实现沿曲率变化剧烈的路径行驶，进而导致算法生成的轨迹不能直接最为车辆的参考轨迹。为此，通过改进节点的扩展方式衍生的 Hybrid A*和改进节点连接方式衍生的 State Lattice 是对搜索算法的改进。其规划的结果为圆弧加直线片段的组合连接，因而依旧存在无法作为车辆参考轨迹的问题。由此可见，基于搜索方法得到的轨迹还需要进一步的后处理，使得其转变为曲率连续的路径。

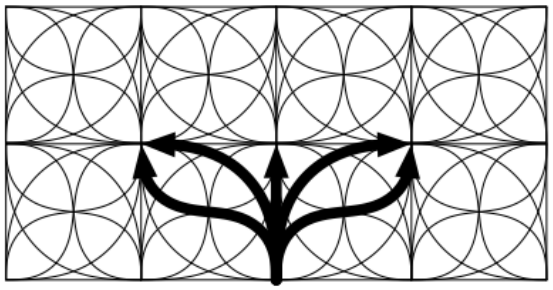


图 1-1 非结构化环境下的常规状态网格

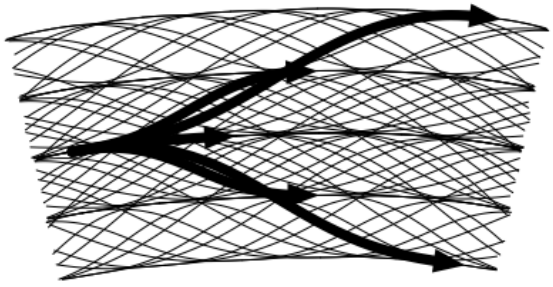


图 1-2 结构化环境的状态网格

与搜索类方法不同，LaValle 提出的快速搜索随机树（Rapidly-exploring Random Tree, RRT）算法主要采取了采样的思路，以在连续的特征空间里扩展的方式得到轨迹节点。

使用 RRT 方法进行路径规划时采样点是随机选取的，这一特点也使其减少了对状态空间进行预处理的过程，同时机器人在客观方面存在的约束包括非完整性约束、动

力学约束、运动学约束等也被考虑在搜索过程中[6]。为了使这一方法能够有更广泛的应用，很多学者也在不断改进该算法。为了提高搜索效率，Kuffner 和 LaValle 提出了 Bi-RRT（双向搜索树）[7]，双向的实质是在初始状态和目标状态并行生成两棵树，以此加快算法收敛；随后他们又提出了 RRT-connect[8]，改进后的方法能够提高节点扩展效率。

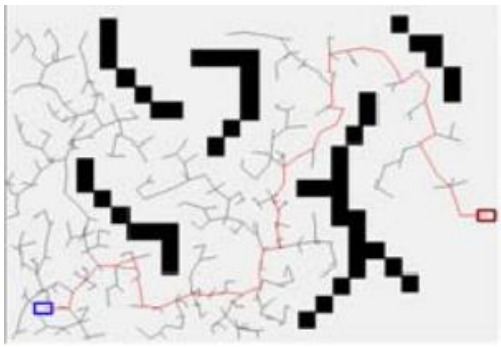


图 1-3 基本 RRT 构造的随机树

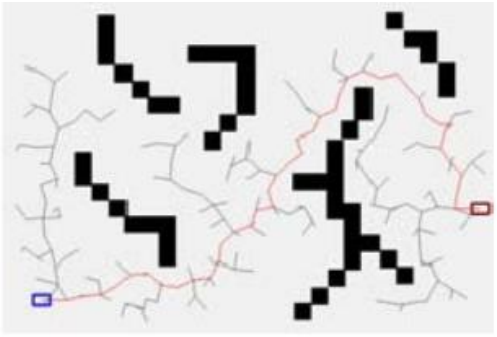


图 1-4 双向 RRT 构造的随机树

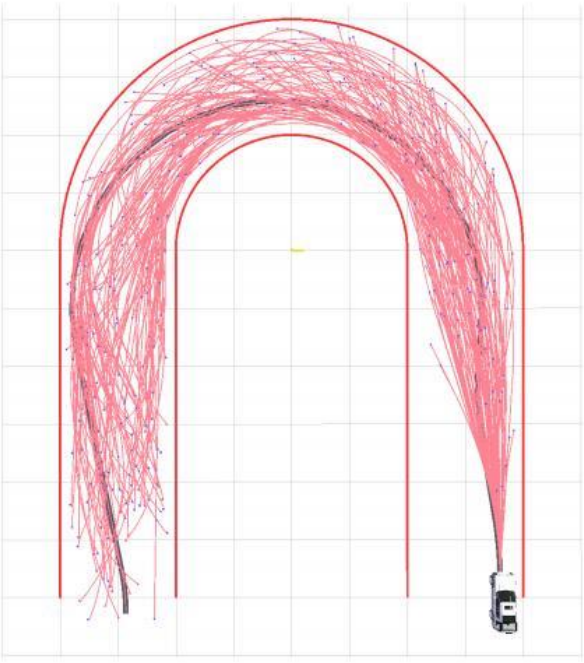


图 1-5 使用 RRT 进行道路转弯处局部路径规划

基于曲线插值的局部路径规划以前方道路上的参考点为基础，为了能够得到曲率连续的轨迹，其将路径点进行插值处理，但是这也会存在局限，即必须要先获取路径点的信息，因此在周围环境变化的情况会花费较大的计算成本，使得此方法只适用于常态化环境和某一部分的轨迹规划。为了解决 RRT 算法的随机性所造成的路径不平滑问题，Elbanhawi 等人[9]提出了简单有效的 3 次 B 样条平滑算法，能够同时保证平滑的

曲率和机器人的非完整性约束。为了使智能车辆拥有在大量无规则障碍物的复杂环境下进行运动规划的能力，杜明博[6]等人引入了目标偏向采样策略以及合理的度量函数，大大地提高了规划速度和质量，而且基于最大曲率约束的后处理方法能够生成平滑的且曲率连续的可执行轨迹。周维等人[10]采用目标导向、节点修剪、曲线拟合和最优路径选择等方法对基础 RRT 规划算法进行改进，利用 Taylor 线性化法建立基于 LTV-MPC 的轨迹跟踪控制器，其路径的获得主要基于启发式，同时使用五次多项式对规划出的局部路径进行拟合，最终得到智能车辆路径规划与跟踪控制算法的基础构架。

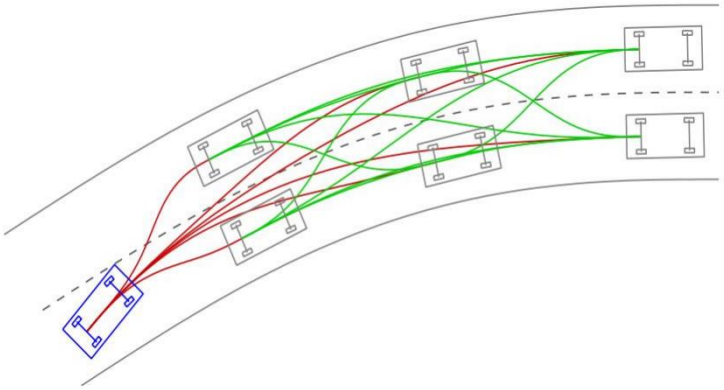


图 1-6 使用多项式进行局部路径规划

数值优化方法以道路、交通参与物等因素为约束和评价指标，接下来使用数值最优方法进行求解，进而获得最优轨迹。人工势场法、最优控制等方法都是基于数值优化进行局部路径规划常采用的一些方法。人工势场法以虚拟势场来表示构型空间，进而构建出一种基于斥力和引力的势场模型。类比于物理中的势场，物体会在势场中沿最低势能的方向移动，进而实现路径规划[11]。但是当周围环境较为复杂时，人工势场法可能会使智能车陷入局部最小值点，或在很小的区域内不停振荡。郭袅鹏[12]对人工势场进行了改进，在传统人工势场法的基础上加入相对距离以及调节系数，这一改进使得虚拟势场变得可以调整和控制，同时 RRT 算法也被加入其中，这些都能达到脱离局部最小值点的效果，进而避免了移动机器人陷入局部最小值点而无法移动的问题。Ren 等人针对人工势场法在存在障碍物的狭窄通道处存在的振荡问题，以牛顿法为基础进行了改进[13]，与标准的梯度下降不同，这一方法可以适用于定义了连续导航功能的任何位置，在规划速度和系统稳定性方面都有提升。最优控制问题通过计算得到一组序列，这一序列不仅满足约束条件且能够令评价指标目标函数达到最优。在最优控制问题中，模型预测控制算法是一种常用的方法，通过对智能车动力学模型进行建模并以此作为约束，能够得到更加符合智能车动力学的轨迹，对于后续轨迹跟踪时需要

输入系统的前馈值，也可由动力学模型计算得到。同时，由于模型预测控制算法需要用到滚动时域方法，因此实际施加到被控系统中的只是每一个周期内计算得到的输入序列的初始值，这也使得智能车能够实现动态的轨迹规划，能够以当前状态为基础进行调整。Xu 等人[14]使用模型预测控制完成了换道过程，并且在将减少能源消耗加入到目标中。Wang 等人[15]研究了基于凸二次规划的模型预测控制和智能车无碰撞导航，在轨迹生成区间将智能车的形状考虑成凸多边形区域，将具有车辆形状的模型预测控制问题作为基于凸二次规划的模型预测控制方案加以解决，控制序列可以快速求解能够改进系统的实时性。

在赛车运动中，圈速模拟通常被用作一种经济有效的工具来评估车辆设置的调整对整圈性能的影响，并利用这些信息进行优化，以找到理想的设置，从而开始实车的测试。但是不同赛道的速度曲线、弯道半径和其他条件的差异导致理想设置只适合单一赛道，因此需要针对每个赛道对车辆参数进行单独优化。赛事团队使用的大多数圈速模拟器都是基于准稳态方法（Quasi-Steady State, QSS），其中车辆模型通过迭代以尽可能高的速度遵循目标路径。在更高级的情况下，这种方法可以通过纳入瞬态车辆行为进一步扩展。QSS 方法需要预先掌握目标路径的信息，这通常是从专业赛车手驾驶的最快圈速的线路记录数据中获得的，而这要以车手完成了正确的轨迹为基础[16]。对于决策系统，其最好能够实时规划车辆应该采取的轨迹而不需要事先处理，这样便能够实现对不同的交通状况做出反应，而且在自主赛车中，能够为随后的每一圈规划更优的轨迹，或者对不断变化的赛道条件做出反应。为满足这些要求，需要找到在使用尽可能少的计算硬件情况下实现较短圈速路径规划的方法，这对于赛车的决策系统而言是一个巨大的挑战。

1.1.2 无人车轨迹跟踪控制研究概述

无人车轨迹跟踪控制是智能车辆研究领域中的核心问题之一，指对当前的周围环境和车体位移、姿态、车速等信息进行处理，并分别向油门、制动及转向等执行系统发出控制指令，从而使车辆能够沿决策层生成的轨迹运行。为实现车辆轨迹的跟踪需要对车辆进行纵向控制和横向控制。纵向控制是指调整车速使车辆间保持足够的空间，使用最少的制动保持相对恒定的车速，并在紧急情况下尽可能快的制动，横向控制通过自动转向控制使车辆始终沿着期望路径行驶，同时保证车辆的行驶安全性和乘坐舒适性。

目前，具有非完整约束的车辆的轨迹跟踪控制一直是学术研究中的关键问题[17]。为了实现非完整约束车辆的稳定性和轨迹跟踪控制，已经设计出了很多的控制器，如反馈线性化法、反步控制法、滑模控制以及神经网络等等方法。Duan 等人[18]提出了一种改进的 PID 控制器，其使用前馈控制的方法消除动态条件引起的航向误差。将神经网络与 PID 控制器结合起来能够实现更准确的跟踪，因为可以通过训练神经网络得到最合适的 PID 参数，Gaining 等人[19]基于车辆模型和转向系统模型建立了神经网络 PID 控制器，并将其应用于横向路径跟踪控制。Lie 等人[20]通过积分反步法推导了闭环控制结构的轨迹跟踪控制器，并利用李雅普诺夫理论分析了所提出的跟踪控制器的稳定性。Bingbing 等人[21]在二轮自行车模型的基础上，以碰撞中心为参考点建立控制模型，并结合线性二次调节器（Linear Quadratic Regulator, LQR）算法设计轨迹跟踪控制器，以实现对参考轨迹的跟踪。Kun 等人[22]为了提高智能车辆的自主路径跟踪控制精度，提出了一种基于模糊神经网络和神经网络预测控制的路径跟踪控制策略。

上述的方法对算法参数以及周围环境的依赖性很高，同时，车辆在运动过程中受到运动学、动力学和执行器等各方面的约束，而上述方法很难对这些进行考虑和处理。而基于模型预测控制的轨迹跟踪控制具有强大的预测能力，还具备滚动优化和处理多目标约束的能力，因此已广泛应用于车辆轨迹规划和跟踪控制。朱盈璇等人[23]提出了一种基于配点法的非线性模型预测控制实时优化算法，其采用配点法将最优控制问题转化为带约束的非线性规划问题，并运用序列二次规划法实现了快速求解，通过滚动优化和反馈校正达到了对智能车实时轨迹跟踪的效果。Falcone 等人[24,25]根据车辆的线性以及非线性动力学模型，分别设计了线性和非线性 MPC 控制器，实现了车辆的主动转向与主动制动，其稳定性也较好。Attia 等人[26]基于 MPC 提出了智能车的纵向和横向综合控制策略，设计的非线性控制器能够对期望轨迹进行稳定的跟踪。Gutjahr 等人[27]采用线性时变模型预测控制方法，结合系统的约束条件，滚动的进行优化。龚建伟等人[28]在模型预测控制理论的基础上，分别针对车辆的运动学和动力学模型建立了模型预测控制器，并将规划部分结合到其中。

1.3 本文研究内容

此次毕业设计课题主要结合 CARLA 模拟器进行了赛车快速轨迹生成算法的编写，并得到了优化轨迹的分布，同时也对在 CARLA 中搭建地图以及进行车辆控制进行了初步探索。本文的第一章对无人驾驶车辆的轨迹规划和轨迹跟踪现状进行了介绍，并

简述了进行无人驾驶赛车轨迹规划以及轨迹跟踪研究的意义；第二章进行了车辆单轨动力学模型以及轮胎模型的搭建，后续的赛车轨迹生成算法将以此为约束条件进行计算；第三章介绍了赛车优化轨迹生成算法的具体思路，并进行了算法编写，其中车辆基础参数对照奥迪 TTS 进行设置；第四章介绍了在 CARLA 中新建雷山赛道的详细方式，并通过编写程序获取新建赛道中心线样本点的位置坐标、朝向角、曲率以及距离起点的沿赛道距离等参数文件，将以上参数文件作为输入得到了优化轨迹以及速度分布结果；第五章介绍了此次轨迹跟踪使用的横向控制算法以及纵向控制算法原理，并通过编程实现了在CARLA模拟器中的轨迹跟踪仿真，做出本文研究路线如图1-7所示。

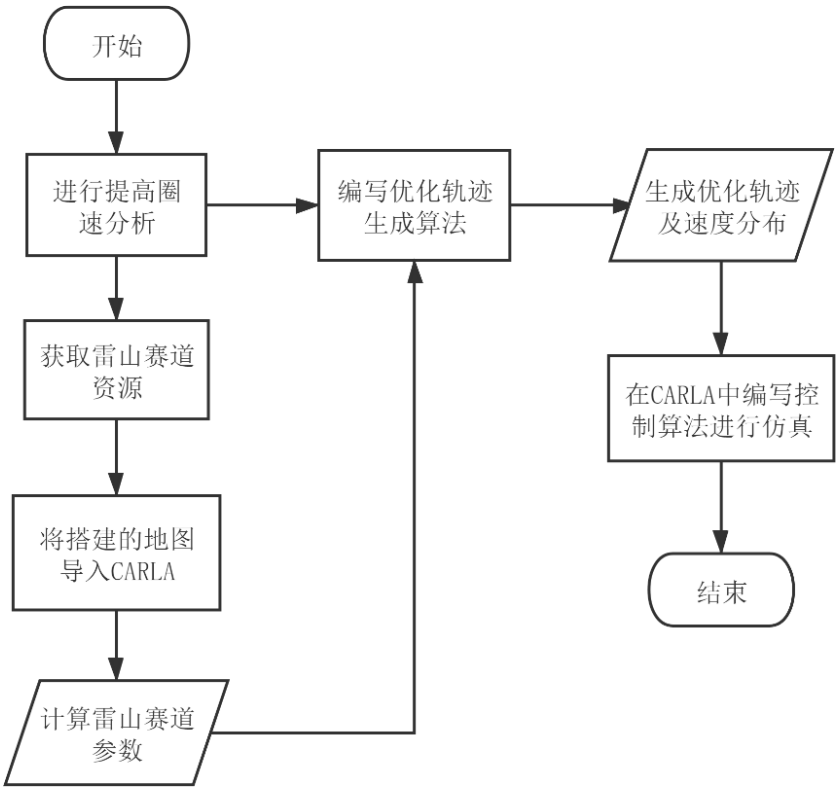


图 1-7 本文研究线路

2 车辆动力学建模

2.1 数学模型的意义

数学模型的建立历来是控制系统设计的基础，控制系统的数学模型是描述系统内部变量之间的数学表达式，无论是经典控制理论中通过传递函数建立的数学模型，还是现代控制理论中通过状态方程建立的数学模型，其目的都是要找出系统输入与输出之间的映射关系[29]。对车辆来说，模型建立的主要任务是寻找车辆系统的状态变量与控制量之间的关系，为设计出有效的跟踪控制律，实现车辆较小误差轨迹跟踪做好准备。智能车辆属于典型的非线性系统，它要求在无人控制的情况下，自主的识别环境、避障，完成规定的任务，这就要求车辆控制系统能对车辆的位姿、速度、位移等物理量实时检测和反馈。所以建立车辆非线性系统的数学模型，准确描述各物理量之间的数学关系成了当前研究的主要内容，它是设计智能车辆的跟踪控制器以及进行运动仿真的前提[30]。

汽车是一个由多个子系统共同组成的多自由度、高耦合度的非线性系统，因此，对于车辆整体的分析和精确建模非常困难。同时，过高的自由度所带来的复杂程度和计算量十分巨大，故一般在分析时会对车辆模型进行简化。在循迹控制研究中，一部分是要求汽车可以沿着期望的轨迹行驶，这部分属于横向控制的范畴，一般通过控制方向盘转角或转向轮转角来实现；另一部分是要求汽车的速度处于理想状态，这部分属于纵向控制的范围，一般由控制驱动系统或制动系统来完成[31]。

2.2 车辆单轨模型

20 世纪 40 年代，Riekert 和 Schunck 建立了最早的 2 自由度车辆动力学模型。他们将整车看作是一个刚体，考虑了前后轴侧偏刚度，对左右两侧的车轮不进行区分，车轮转角直接作为模型的输入，定义了汽车的不足转向和过多转向特性。文献[32]采用了单轨 2 自由度模型，模型忽略了车辆的垂向运动、俯仰运动、侧倾运动以及悬架运动，以及轴荷侧向转移、空气动力学载荷和转向几何，假设道路的坡度、整车侧偏角和横摆角偏差都很小。本文选用车辆单轨模型对车辆进行动力学建模，并做出以下假设：

（1）假设车辆是以前轮驱动并控制转向，也就是后轮的转角恒为零；

（2）假设车辆在平坦路面上行驶，车体只做与地面平行的水平运动，忽略了车辆的垂直、俯仰、侧倾等其他运动；

（3）假设悬架及车辆是刚性的，忽略悬架和轮胎之间耦合的影响；

（4）不考虑车辆前后左右的载荷转移；

（5）仅考虑车辆的纯侧偏轮胎特性，不考虑轮胎力的横纵向耦合；

（6）不考虑空气动力学的影响。

如图 2-1 为车辆单轨模型的示意图。示意图中的符号可在表 2-1 中对应。

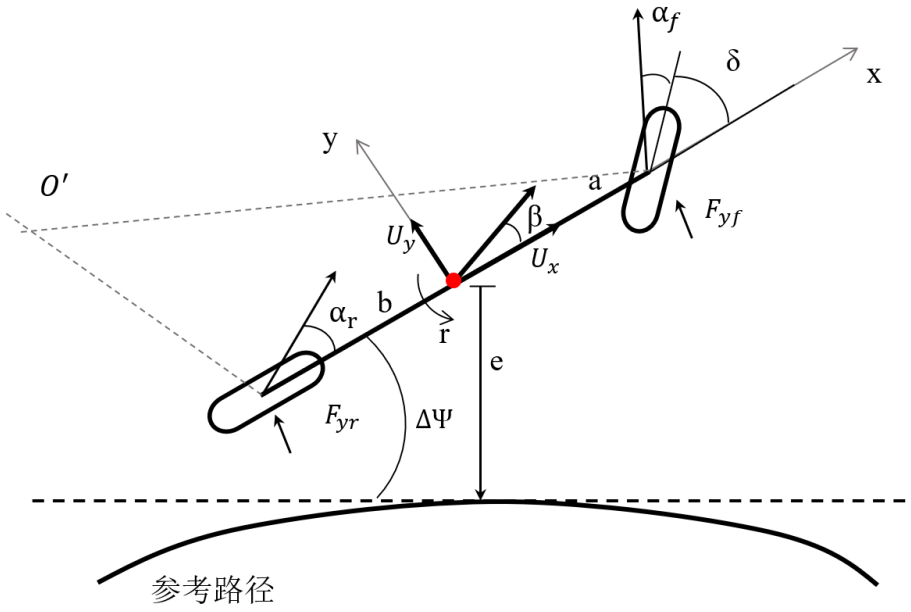


图 2-1 车辆单轨模型示意图

表 2-1 车辆单轨模型参数定义

参数	符号	单位
前轴到重心	a	m
后轴到重心	b	m
前轮横向力	F_{yf}	N
前轮侧偏角	α_f	rad
后轮横向力	F_{yr}	N
后轮侧偏角	α_r	rad
转向角	δ	rad

横摆角速度	r	rad/s
车身侧偏角	β	rad
车辆质心与期望路径的横向距离偏差	e	m
车辆航向与期望路径朝向的偏差角	$\Delta\Psi$	rad
纵向速度	U_x	m/s
横向速度	U_y	m/s
瞬时转向中心	O'	-

2.2.1 横向动力学

设V为车辆质心处的标量速度值，其与车辆的纵向速度U_x的关系由可用式（2-1）表示为

$$U_x = V\cos\beta \tag{2-1}$$

其中，β为车身侧偏角，即质心处的速度与车身实际朝向之间的夹角。

由于车身侧偏角β很小，因此可以假设$U_x = V$，因此有

$$\beta = \arctan\frac{U_y}{U_x} \approx \frac{U_y}{U_x} \tag{2-2}$$

在质心处汽车的矢量速度$\boldsymbol{V} = U_x\boldsymbol{i} + U_y\boldsymbol{j}$且$di/dt = r\boldsymbol{j}$，$dj/dt = -r\boldsymbol{i}$，因此车辆的纵向及横向加速度的推导如式（2-3）至式（2-5）所示

$$\boldsymbol{a} = \frac{d\boldsymbol{V}}{dt} = \left(\dot{U}_x - U_y r\right)\boldsymbol{i} + \left(\dot{U}_y + U_x r\right) = a_{\mathrm{x}}\boldsymbol{i} + a_{\mathrm{y}}\boldsymbol{j} \tag{2-3}$$

$$a_{\mathrm{x}} = \dot{U}_x - U_y r \tag{2-4}$$

$$a_{\mathrm{y}} = \dot{U}_y + U_x r \tag{2-5}$$

考虑车辆转向时y轴上的横向动力学，将牛顿第二定律应用于横向运动和圆周运动，可得出式（2-6）及式（2-7）

$$F_{yf} + F_{yr} = ma_y = m\left(\dot{U}_y + U_x\dot{\Psi}\right) = mU_x\left(\dot{\beta} + \dot{\Psi}\right) \tag{2-6}$$

$$aF_{yf} - bF_{yr} = I_z\ddot{\Psi} \tag{2-7}$$

其中，F_{yf}和F_{yr}分别是前后轮上的横向反作用力，m是车辆质量，$\dot{\Psi}$为对车辆的航向角求导，即车辆的横摆角速度r，a、b是前后轴与车辆质心之间的水平距离，I_z是车身的惯性矩。

前轮胎和后轮胎侧偏角可以从几何关系中获得，如式（2-8）和式（2-9）所示

$$\alpha_f = \arctan\left(\frac{U_y+a\Psi}{U_x}\right) - \delta \approx \beta + \frac{a\Psi}{U_x} - \delta \tag{2-8}$$

$$\alpha_r = \arctan\left(\frac{U_y-b\Psi}{U_x}\right) \approx \beta - \frac{b\Psi}{U_x} \tag{2-9}$$

2.3 轮胎模型

轮胎是车辆的重要组成部件，其非线性特性和结构参数影响着车辆的转向特性以及行驶的稳定性。在行驶过程中，轮胎受到侧向力、纵向力、垂直力以及回正力矩的作用，所以对车辆进行控制研究之前，选用合适的轮胎模型很重要。目前应用比较普遍的轮胎模型有刚性轮胎模型、经验轮胎模型（魔术公式）、物理轮胎模型以及有限元轮胎模型。为了更贴合实际同时简化计算，选择物理模型中的 Fiala tire model 作为轮胎的建模方式，Fiala tire model 基于 brush model 假设，只需要少量的系数。该轮胎模型在力和力矩计算中将轮胎的横向和纵向滑移状态结合起来，提供了更真实的综合滑移描述，其计算公式如式（2-10）所示

$$F_y = \begin{cases} -C\tan\alpha + \dfrac{C^2}{3\mu F_z}|\tan\alpha|\tan\alpha - \dfrac{C^3}{27\mu^2 F_z^2}\tan^3\alpha & ,|\alpha| < \arctan\left(\dfrac{3\mu F_z}{C}\right) \\ -uF_z sign\alpha & ,\text{其他} \end{cases} \tag{2-10}$$

可通过代入车辆单轨模型中相应的前后轮胎的垂向力F_{zf}和F_{zr}以及相应的侧偏刚度C_{zf}和C_{zr}、侧偏角α_{zf}和α_{zr}，来计算相应的F_{yf}和F_{yr}。

侧向力与轮胎侧偏角α_f和α_r有关，在轮胎使用 Fiala 模型的前提下这种关系是非线性的，在稳态转弯条件的假设下，轮胎模型可以在每个小段区域处近似线性化处理，得到式（2-11）和式（2-12）

$$F_y = \widetilde{F_y} - \tilde{C}(\alpha - \tilde{\alpha}) \tag{2-11}$$

$$\widetilde{F_y} = \frac{F_z}{g}U_x^2 K \tag{2-12}$$

其中，K为车辆轨迹某一点处的曲率，参数$\widetilde{F_y}$、$\tilde{C}$、$\tilde{\alpha}$如图 2-2 所示。在任一点处进行线性化近似可以得到式（2-13）和式（2-14）

$$F_{yf} = -\widetilde{C_f}\alpha_f = \widetilde{C_f}(\delta - \beta - \frac{a\Psi}{U_x}) \tag{2-13}$$

$$F_{yr} = -\widetilde{C_r}\alpha_r = \widetilde{C_r}(\frac{b\Psi}{U_x} - \beta) \tag{2-14}$$

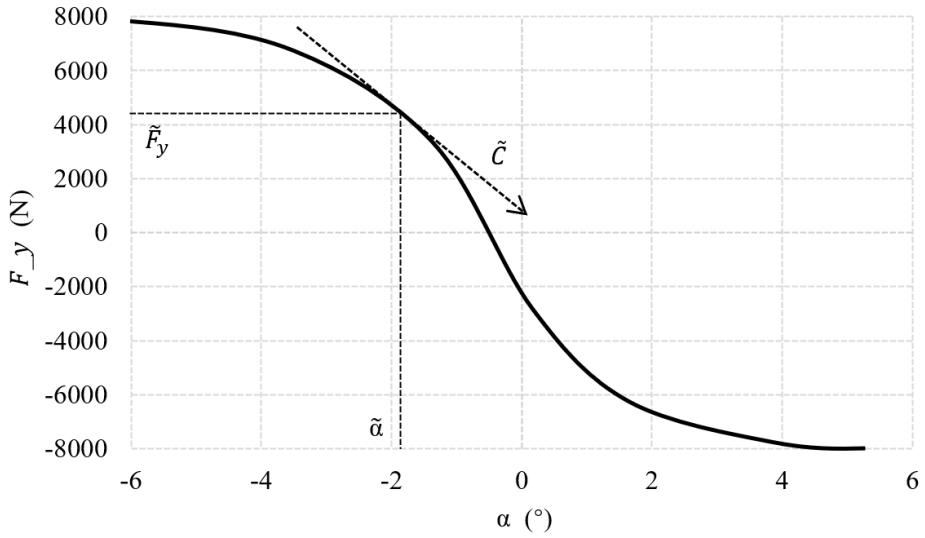


图 2-2 在$\alpha = \tilde{\alpha}$处线性化轮胎模型示意图

2.4 车辆动力学模型

将式（2-6）、式（2-7）中的F_{yf}和F_{yr}用式（2-13）、式（2-14）的形式代入，得到式（2-15）和式（2-16）

$$\dot{\beta} = \frac{-(\widetilde{C_r} + \widetilde{C_f})}{mU_x}\beta + \left(\frac{\widetilde{C_r} + \widetilde{C_f}}{mU_x^2} - 1\right)\dot{\Psi} + \frac{\widetilde{C_f}}{mU_x}\delta \tag{2-15}$$

$$\ddot{\Psi} = \frac{\widetilde{c_r}b - \widetilde{c_f}a}{I_z}\beta - \frac{\widetilde{c_r}b^2 - \widetilde{c_f}a^2}{I_z U_x}\dot{\Psi} + \frac{\widetilde{c_f}a}{I_z}\delta \tag{2-16}$$

车辆与期望路径之间的横向距离偏差e可以表示为式（2-17）的形式

$$\dot{e} = U_y \cos\Delta\Psi + U_x \sin\Delta\Psi = U_x(\beta + \Delta\Psi) \tag{2-17}$$

需要指出的是，式（2-17）的转换过程是基于$\Delta\Psi$小角度假设进行的。

由于$\Delta\Psi$为车辆航向与期望路径朝向的偏差角，即$\Delta\Psi = \Psi - \Psi_d$，对其求导即得式（2-18）

$$\dot{\Delta\Psi} = \dot{\Psi} - U_x K \tag{2-18}$$

将$X = [e \ \ \Delta\Psi \ \ r \ \ \beta \ \ \Psi]^T$作为车辆系统的状态向量，具有$\delta$转向输入的线性化连续车辆单轨模型可写为状态空间形式，即$\dot{X} = AX + B\delta + d$的形式，其中

$$A=\begin{bmatrix}0 & U_{\mathrm{x}} & 0 & U_{\mathrm{x}} & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & \dfrac{-a^2\widetilde{C_f}-b^2\widetilde{C_r}}{U_{\mathrm{x}}I_z} & \dfrac{\mathrm{b}\widetilde{C_r}-\mathrm{a}\widetilde{C_f}}{I_z} & 0\\ 0 & 0 & \dfrac{\mathrm{b}\widetilde{C_r}-\mathrm{a}\widetilde{C_f}}{mU_x^2}-1 & \dfrac{-\widetilde{C_f}-\widetilde{C_r}}{mU_{\mathrm{x}}} & 0\\ 0 & 0 & 1 & 0 & 0\end{bmatrix} \tag{2-19}$$

$$B=\begin{bmatrix}0 & 0 & \dfrac{a\widetilde{C_f}}{I_z} & \dfrac{\widetilde{C_f}}{\mathrm{m}U_{\mathrm{x}}} & 0\end{bmatrix}^T \tag{2-20}$$

$$d=\begin{bmatrix}0\\ -KU_{\mathrm{x}}\\ \dfrac{\mathrm{a}\widetilde{C_f}\widetilde{\alpha_f}-\mathrm{b}\widetilde{C_r}\widetilde{\alpha_r}+\mathrm{a}\widetilde{F_{yf}}-b\widetilde{F_{yr}}}{I_z}\\ \dfrac{\widetilde{C_f}\widetilde{\alpha_f}+\widetilde{C_r}\widetilde{\alpha_r}+\widetilde{F_{yf}}+\widetilde{F_{yr}}}{mU_{\mathrm{x}}}\\ 0\end{bmatrix} \tag{2-21}$$

其中，d为仿射项，得到车辆的仿射动力学模型。

2.5 模型编程

本文汽车参数的设置主要基于奥迪 TTS，其整车参数如表 2-2 所示。

表 2-2 车辆单轨模型参数定义

参数	单位	数值
前轴到重心的距离	m	1.0441
后轴到重心的距离	m	1.4248
车轮半径	m	0.34
车重	kg	1512.4
车辆绕 z 轴的转动惯量	kg·m²	2.25×10³
发动机输出功率极限	kW	210
前轮摩擦系数	-	0.97
后轮摩擦系数	-	1.02
车轮转角极限	（°）	70
前轮侧偏刚度	N/rad	160000
后轮侧偏刚度	N/rad	180000

上述车辆动力学模型将在优化轨迹生成算法中作为凸优化算法的约束项，在车辆行驶过程中通过运行写入以上车辆动力学模型的函数，可以得到车辆的状态参数。首

先根据车辆参数计算出 Fiala 轮胎模型的侧偏角临界值，在负数临界值与正数临界值之间进行插入 250 个点，得到侧偏角的列表。将得到的侧偏角列表作为实参，进行 Fiala 轮胎模型下轮胎的侧向力的计算。接下来将以线性化离散的形式计算车辆动力学模型。以式（2-12）计算目前速度分布下所需的侧向力，在前文应用 Fiala 模型得到的侧偏角列表与轮胎的侧向力列表之间的对应关系基础上，通过插值得到目前轮胎侧向力分布下对应的侧偏角，以此为基础求解小区间上的轮胎侧偏刚度，最后将数值作为参数赋给状态矩阵，从而完成了通过车辆动力学模型得到车辆的状态参数的过程。

将建立轮胎模型的过程以图 2-3 中伪代码的形式表示，将车辆动力学模型建立的过程以图 2-4 中伪代码的形式表示。

```
1: procedure 定义 Fiala 轮胎模型（侧偏刚度，摩擦系数，侧偏角列表，垂向力）
2:         侧向力列表←初始化为零列表
3:         for i←0 to 侧偏角列表元素个数
4:             if 侧偏角＜侧偏角临界值
5:                 侧向力←式（2-10）情况一
6:             else
7:                 侧向力←式（2-10）情况二
8:         end for
9:         return 侧向力列表
10: end procedure
```

图 2-3 建立轮胎模型伪代码

```
1: procedure 定义车辆动力学模型（车辆参数类，车速列表，曲率列表）
2:         A、B、D 状态矩阵列表←空列表
3:         侧偏角临界值←式（2-10）情况一的临界值
4:         侧偏角列表←在正负临界值之间插值
5:         侧向力列表←Fiala 轮胎模型函数
6:             for i←0 to 车速列表元素个数
7:                 期望侧偏力列表←（车辆参数，车速列表，曲率列表）
8:                 期望侧偏角列表←在侧偏力与侧偏角对应关系中插值
9:                 A、B、D 状态矩阵列表←（车速列表，曲率列表，车辆参数，
期望侧偏角列表）
10:            end for
11:            return A、B、D 状态矩阵列表
12: end procedure
```

图 2-4 建立车辆动力学模型伪代码

2.6 本章小结

本章将车辆简化为单轨动力学模型，并在这一模型的基础上进行横向动力学以及轮胎模型的公式推导，接下来将车辆单轨动力学模型表示为状态空间形式，最后将这一动力学模型编写为程序。在本次车辆动力学编程的过程中，车辆参数对照奥迪 TTS 进行设置，后续轨迹生成算法将以此车辆动力学模型为基础计算车辆的状态量。

3 轨迹生成算法

3.1 算法概述

对于最快圈速优化轨迹生成，其复杂性有两个方面。首先，必须确定车辆的纵向和横向输入，但是车辆的横向和纵向动力学存在耦合性和非线性。其次，直接最小化圈速需要最小化一个非凸成本函数。非凸优化问题不仅比凸优化问题求解成本更高，而且求解技术也只能保证收敛到局部最小值。

针对车辆动力学相互耦合且在求取最优值时为非凸优化问题的特点，可以通过顺序近似的进行求解，这会将其转化为更简单的凸优化问题[33]。同时借鉴期望最大化算法（Expectation Maximization Algorithm）的思路，其使用了迭代优化策略，它的计算方法中每一次迭代都分两步，其中一个为期望步（E 步），另一个为极大步（M 步），因此可以将根据固定轨迹生成速度分布以及根据固定速度生成轨迹的过程理解为在相互提供期望值但同时也因为存在限制而产生一个最大值，即先将数值初始化再通过迭代使其收敛。基于上述两种方法提出了最终的轨迹生成算法。该算法绕过了最快圈速优化的复杂性，从而以较低的计算时间生成车辆轨迹。为了避免耦合控制输入的问题，组合的横向、纵向最优控制问题被迭代求解的两个连续子问题替代。在第一个子问题中，给定固定的车辆路径，计算最小时间纵向速度输入。在第二子问题中，给定固定速度更新车辆路径。因此避免了非凸成本函数最小值的求解，而是通过求解路径最小曲率这一凸函数的方法来更新车辆路径。

3.2 路径描述

赛道的路径和道路边界可以在图 3-1 中通过笛卡尔$E-N$坐标直观地描述。但是在赛道中进行轨迹规划时，更方便的一种表示方法是将赛道中心线参数化为曲率轮廓K，该曲率轮廓是沿着赛道距离s的函数（图 3-3）。另外还可以将道路边界信息存储成两个以s为自变量的函数$w_{in}(\mathrm{s})$和$w_{out}(\mathrm{s})$，这两个函数分别对应于s处道路中心线到道路内侧和外侧边界的最短距离（图 3-2）。

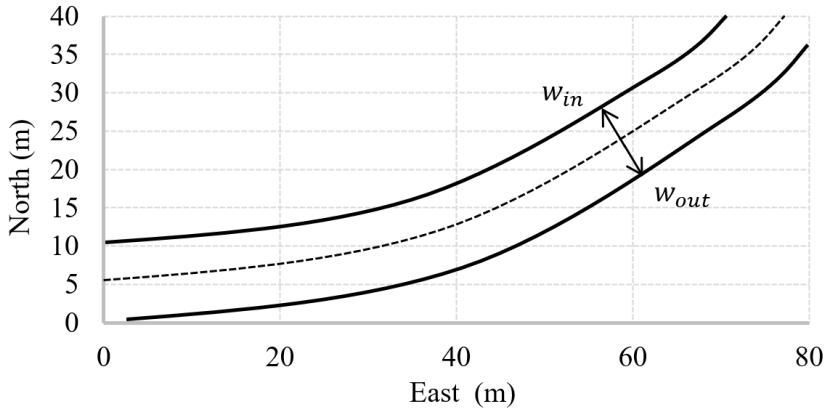


图 3-1 将赛道放在笛卡尔坐标中

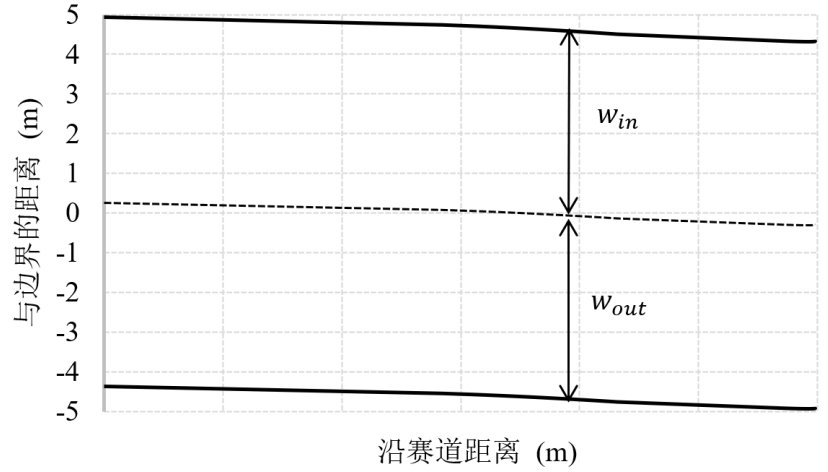


图 3-2 将赛道边界距中心线的横向距离表示为s的函数

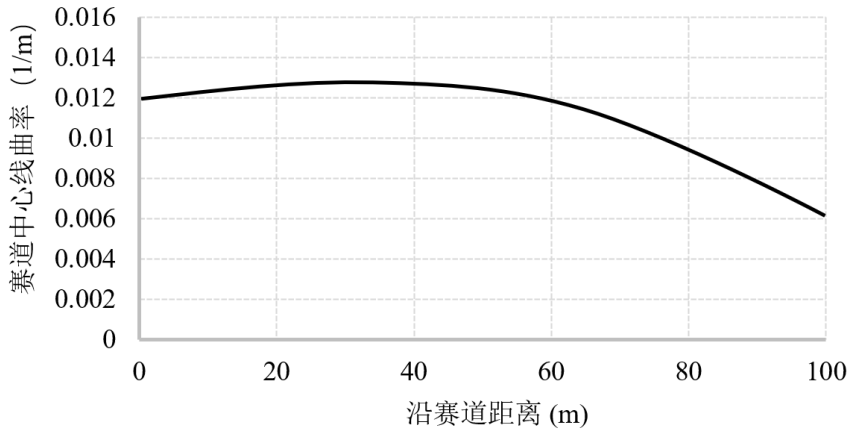


图 3-3 将赛道曲率表示为s的函数

3.2.1 程序编写

路径信息通过 csv 文件获取，这一文件中存储着赛道中心线样本点的位置坐标、朝向角、曲率以及赛道起点的沿赛道距离信息，第四章将会借助 CARLA 对赛道信息的获取过程进行介绍，将赛道信息读取过程以图 3-4 所示的伪代码形式表示。

1: **procedure** 定义赛道信息类（赛道信息 csv 文件）
2: E、N、朝向角列表、曲率列表、沿赛道距离列表←初始化为零列表
3: E、N、朝向角列表、曲率列表、沿赛道距离列表←赛道信息矩阵的对应列
4: **end procedure**

图 3-4 赛道信息读取过程伪代码

3.3 给定轨迹生成速度

3.3.1 公式推导

给定由s和K描述的固定参考路径，算法的第一步是找到车辆在不超过可用摩擦力的情况下可以达到的最短时间速度分布。本章中速度分布的计算方式直接使用了 Subosits 和 Gerdes[34]描述的三步速度生成方法。确定车辆的速度曲线需要将轮胎产生力的能力转化为车辆的加速度极限，然后转化为最大安全速度。对于单轨车辆模型的集中前后轮胎，每个车轮上可用的纵向和横向力F_x和F_y受摩擦圆约束，约束条件可由式（3-1）和式（3-2）表示

$$F_{xf}^2 + F_{yf}^2 = \left(\mu F_{zf}\right)^2 \tag{3-1}$$

$$F_{xr}^2 + F_{yr}^2 = (\mu F_{zr})^2 \tag{3-2}$$

其中，μ是轮胎与路面之间的摩擦系数，F_z是可用的法向力，$F_{zf} = mgb/(a+b)$，$F_{zr} = mga/(a+b)$。速度曲线生成的第一步是找出路径上每个点的最大速度。对于这种计算，纵向加速度可以固定为零，结合式（3-1）和式（3-2）忽略重量转移和地形影响可得到式（3-3）

$$U_x(s) = \sqrt{\frac{\mu g}{|K(s)|}} \tag{3-3}$$

经过计算式（3-3），得到图 3-5 中第二阶段所示的速度分布，在确定了逐点约束后实施微分约束。第二步依据给定点的速度由前一点的速度和可用的加速纵向力$F_{x,accel,max}$决定的原理进行向前积分计算。其中$F_{x,accel,max}$需要考虑车辆发动机力极限和由于道路曲率引起的所有轮胎上的横向力需求，假设加速在长度为Δs的每个小空间上是恒定的，则后一点的速度根据式（3-4）得到

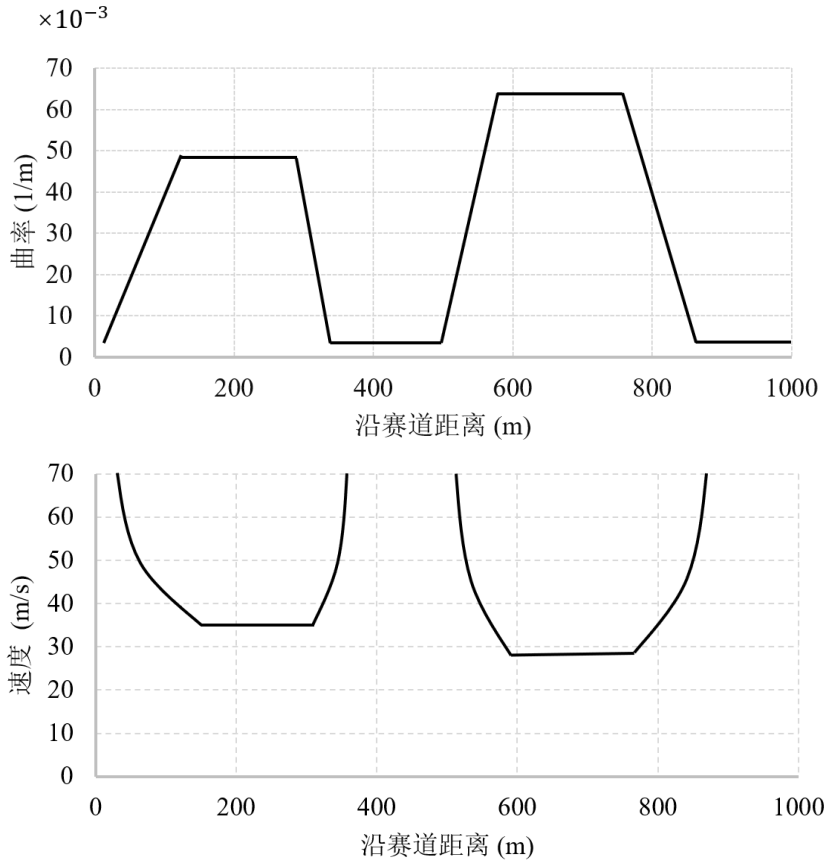
$$U_x(s+\Delta s) = \sqrt{U_x^2(s) + 2\frac{F_{x,accel,max}}{m}\Delta s} \tag{3-4}$$

从开放路径的起点或封闭路径上速度最慢的点开始向前执行积分。对于路径上的每个点，将经过向前积分计算得到的值与第一步中的相应值进行比较，并选择较小的值作为下一个积分步骤的初始条件，得到图 3-5 中第三阶段所示的速度分布。

第三步是向后积分步骤，为了减少时间，接近弯道时的减速必须尽可能大。此时可用于减速的纵向力$F_{x,decel,max}$受到所有轮胎上横向力需求的限制。假设减速在长度为Δs的每个小空间上是恒定的，则前一点的速度根据式（3-5）得到

$$U_x(s-\Delta s)=\sqrt{U_x^2(s)-2\frac{F_{x,decel,max}}{m}\Delta s} \tag{3-5}$$

对于封闭路径，积分从最小速度点开始。对于具有不同起点和终点的开放路径，从终点向后积分。在每一点，计算向后积分得到的值与第二步中得到的相应速度值进行比较，如果在任何一点上向后积分得到的速度大于第二步得到的速度值应取较小值，最终得到图 3-5 第四阶段实线所示的速度分布。



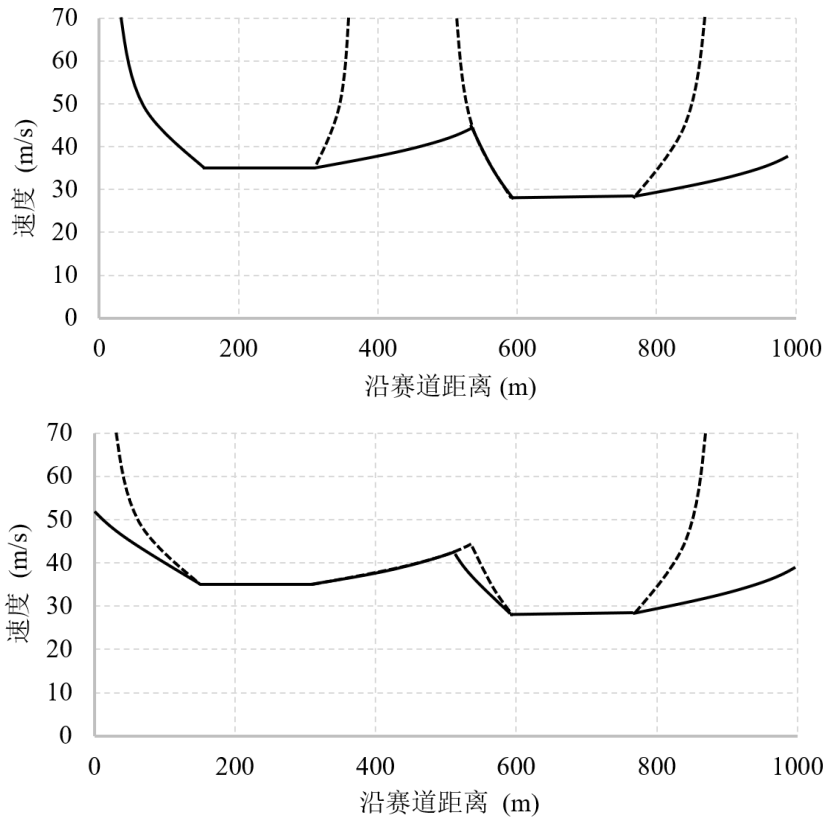


图 3-5 三步生成速度过程示意图

3.3.2 程序编写

首先通过计算式（3-3）找到赛道上速度最小的样本点，记录这一样本点的索引，通过 roll 函数将速度最小值对应的索引放在第一位，形成一个新的索引列表，将使用这一索引列表索引到的样本点曲率值值存放在一个新的列表。创建三个速度列表，在第一个列表存放计算式（3-3）得到的结果。对第二个速度列表初始化为 0 后将先前计算出的速度最小值放在第一位，而第三个速度在初始化为 0 后将先前计算出的速度最小值放在最后一位。以第二个速度列表中的元素个数为循环次数，每次循环中应用式（3-4）计算速度，并与第一个速度列表对应索引得到的速度进行比较，取较小值。最后以第三个速度列表中的元素个数为循环次数，每次循环中应用式（3-5）计算速度，并与第二个速度列表对应索引得到的速度进行比较，取较小值。以第三个速度列表中的速度作为最终结果，最后再使用 roll 函数，即还原到以赛道起点为开始，沿赛道的速度分布。将上述过程以图 3-6 所示的伪代码形式表示。

```
1: procedure 生成速度分布（纵向加速度极限，纵向减速极限，曲率列表，速度列表，沿赛道距离列表）
2:      速度最小值U_xs←（式 3-3）
3:      速度最小值索引 idx←np.argmin（U_xs）
4:      将U_xs放到第一位的索列表←np.roll（速度列表元素个数，-idx）
5:      变换顺序的曲率←（将U_xs放到第一位的索列表）
6:      U_x1列表←式（3-3）
7:      for i←0 to 速度列表元素个数
8:          temp←式（3-4）
9:          if temp＞U_x1
10:             temp←U_x1
11:         if temp≤U_x1
12:             temp←temp
13:         U_x2 =temp
14:     end for
15:     for i←0 to 速度列表元素个数
16:         temp←式（3-5）
17:         if temp＞U_x2
18:             temp←U_x2
19:         if temp≤U_x2
20:             temp←temp
21:         U_x3 =temp
22:     end for
23:     return U_x3
24:     速度分布←np.roll（U_x3，idx）
25: end procedure
```

图 3-6 计算速度分布过程伪代码

3.4 给定速度生成轨迹

3.4.1 整体方法概述

轨迹生成算法的第二步将原始参考路径$K(s)$和相应的速度分布$U_x(s)$作为输入，并修改参考路径以获得新的、更快的比赛路线。

给定赛道的圈速t可由式（3-6）计算

$$t = \int_0^l \frac{ds}{U_x(s)} \tag{3-6}$$

从式（3-6）中可以看出，要实现最快圈速需要同时最小化总路径长度l，同时最大

化车辆纵向速度U_x。然而这是存在矛盾的两个目标，因为当达到轮胎的侧向力极限时，较低的曲率会导致较长的路径长度，但可以得到较高的车速，如式（3-3）所示。由于优化式（3-6）会得到一个非凸成本函数，需要时间密集型非线性规划来实现这种曲率与距离权衡从而得到最快的圈速。

首先不考虑路面起伏，将赛道简化为一个平面的情况下，可以将赛道简单地分为弯道和直道两部分。而驾驶技术带来的影响主要体现在弯道上，因为赛道是连续的，即使两个弯的几何形状完全一样，如若入弯前和出弯后的路不同，那么便是两种走法，所以弯道需要足够的技术和经验才能驾驭。赛车靠轮胎与地面的摩擦力驱动，而这个摩擦力是有极限的。如果侧向力超过轮胎的极限就会发生侧滑，所以为了更快的过弯就需要增大过弯时行车线的半径。

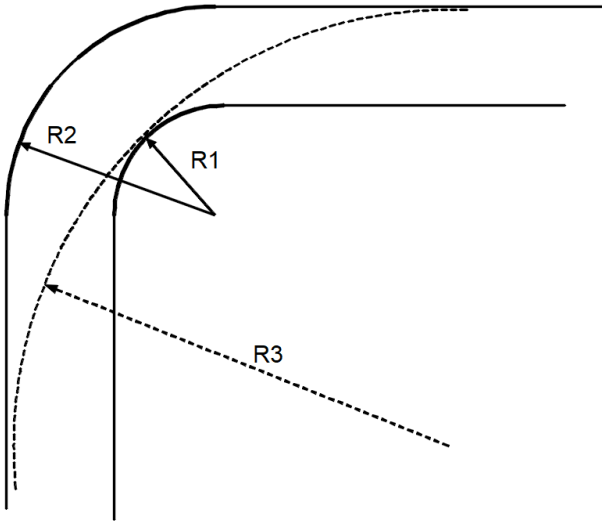


图 3-7 常规赛车过弯路线示意图

由图 3-7 中可以发现，由虚线表示的赛车线的转弯半径比赛道两侧半径R1、R2都大，由式（3-3）可知这会使得车辆能够以较高的速度运行，因此也是大多数情况下会选择的路线。图 3-7 只是个一个简化的示意图，但能够由此推及到过弯的要领，即采用“外-内-外”的走法。因此结合这一赛车的常规知识，我们暂时不考虑增加距离对圈速的影响，优先考虑提升车速，即尽量减小赛车行驶路线的曲率。因此，所提出的方法是通过在每个路径修改步骤中仅最小化车辆行驶曲率来简化成本函数，而路径曲率可以也可以很容易地表述为凸函数。从而实现在考虑车辆动力学约束的情况下，利用凸优化的计算速度来快速地完成路径修改步骤。然而，最小化曲率并不等于最短圈速，并且不能保证找到时间最优的解决方案。提出的成本函数依赖于这样的假设，即具有最小曲率的路径是最短时间赛道的良好近似，对于大多数赛道来说，降低赛道的

曲率比最小化路径长度更重要，因为相对狭窄的赛道宽度为缩短总路径长度提供了有限的空间。

3.4.2 凸优化问题

在以s和K定义赛道路径的前提下，需要车辆的动力学模型。我们假设车辆动力学由 2.4 章节中的平面单轨模型给出。以第二章的车辆动力学模型为基础，此时状态矩阵 A、B 中的参数应修改为时间 t 的函数 A(t)、B(t)，将具有转向输入δ的仿射、时变车辆单轨模型表示为状态空间的形式，如式（3-7）、式（3-8）所示

$$\dot{x}(t) = A(t)x + B(t)\delta + d(t) \tag{3-7}$$

$$X = [e\ \ \Delta\Psi\ \ r\ \ \beta\ \ \Psi]^T \tag{3-8}$$

其中，

$$A(t) = \begin{bmatrix} 0 & U_{\mathrm{x}}(t) & 0 & U_{\mathrm{x}}(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-a^2\widetilde{C_f}(\mathrm{t}) - b^2\widetilde{C_r}(\mathrm{t})}{U_{\mathrm{x}}(t)I_z} & \dfrac{\mathrm{b}\widetilde{C_r}(\mathrm{t}) - \mathrm{a}\widetilde{C_f}(\mathrm{t})}{I_z} & 0 \\ 0 & 0 & \dfrac{\mathrm{b}\widetilde{C_r}(\mathrm{t}) - \mathrm{a}\widetilde{C_f}(\mathrm{t})}{mU_x^2(t)} - 1 & \dfrac{-\widetilde{C_f}(\mathrm{t}) - \widetilde{C_r}(\mathrm{t})}{mU_{\mathrm{x}}(t)} & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-9}$$

$$B(t) = \begin{bmatrix} 0 & 0 & \dfrac{a\widetilde{C_f}(\mathrm{t})}{I_z} & \dfrac{\widetilde{C_f}(\mathrm{t})}{\mathrm{m}U_{\mathrm{x}}(t)} & 0 \end{bmatrix}^T \tag{3-10}$$

$$d(t) = \begin{bmatrix} 0 \\ -K(t)U_{\mathrm{x}}(t) \\ \dfrac{\mathrm{a}\widetilde{C_f}(\mathrm{t})\widetilde{\alpha_f}(\mathrm{t}) - \mathrm{b}\widetilde{C_r}(\mathrm{t})\widetilde{\alpha_r}(\mathrm{t}) + \mathrm{a}\widetilde{F_{yf}}(\mathrm{t}) - b\widetilde{F_{yr}}(\mathrm{t})}{I_z} \\ \dfrac{\widetilde{C_f}(\mathrm{t})\widetilde{\alpha_f}(\mathrm{t}) + \widetilde{C_r}(\mathrm{t})\widetilde{\alpha_r}(\mathrm{t}) + \widetilde{F_{yf}}(\mathrm{t}) + \widetilde{F_{yr}}(\mathrm{t})}{mU_{\mathrm{x}}(t)} \\ 0 \end{bmatrix} \tag{3-11}$$

现在，已将非线性的车辆动力学模型近似为线性时变模型，接下来将通过解决以下凸优化问题来完成路径更新。

$$\begin{cases} \text{Minimize} \sum_k \left(\dfrac{\Psi_k - \Psi_{k-1}}{\mathrm{s}_k - \mathrm{s}_{k-1}}\right)^2 + \lambda(\delta_k - \delta_{k-1})^2 & (3\text{-}12) \\ \text{subject to } x_{k+1} = A_k x_k + B_k \delta_k + d_k & (3\text{-}13) \\ w_k^{out} \le e_k \le w_k^{in} & (3\text{-}14) \\ x_1 = x_T & (3\text{-}15) \end{cases}$$

其中，k= 1…T 是离散化时间参数，A_k、B_k和d_k是式（3-7）中连续状态空间方程的离散化形式。目标函数式（3-12）求解最小化路径曲率，其中路径曲率是车辆航向角相对于路径s的导数。为了保持目标函数的凸性，项$s_k - s_{k-1}$是一个常数。此外，在成本函数中增加了一个权重为λ的项，以确保实验实施的平滑转向轮廓。等式约束（式 3-13）确保车辆遵循横向动力学。不等式约束（式 3-14）允许车辆横向偏离参考路径，以找到曲率较低的新路径，但仅限于道路边缘。可用的道路宽度w_{in}和w_{out}会有一个内置的偏移量，以考虑车辆的宽度。最后，对于完整的赛道，要求等式约束（式 3-15），以确保生成的赛道是一个连续的环。参考路径从道路中心线开始，优化会基于道路可用宽度来降低路径曲率，从而找到修改路径。

3.4.3 程序编写

首先需要调用前文构建的车辆动力学模型，以第一步得到的速度分布为基础，获得状态矩阵。创建一个列表存储每个样本点的转向角以及一个矩阵存储每个样本点处的状态向量，接下来表示出式（3-12）所示的目标函数，并将式（3-13）至式（3-15）作为约束，最后调用凸优化函数得到经优化后的轨迹参数及车辆状态参数，以图 3-8 中所示的伪代码表示进行凸优化的过程。

```
1: procedure 凸优化过程（车辆参数类，速度分布列表，曲率分布列表）
2:         目标函数←式（3-7）
3:         约束←空列表
4:         for i←0 to 速度分布列表元素个数
5:             约束←式（3-13）
6:         约束←式（3-14）
7:         约束←式（3-15）
8:         end for
9:         return 车辆状态量←cvxpy.Problem(目标函数，约束)
10: end procedure
```

图 3-8 凸优化过程伪代码

3.5 算法总结

该算法的输入是任何一个赛道的初始路径，将初始路径用沿路径的距离s、路径曲率$K(s)$以及车道边缘距离$w_{in}(s)$和$w_{out}(s)$的参数化形式表示。给定初始路径，使用章节 3.3 中的方法计算最小时间速度曲线。随后，通过章节 3.4 中求解最小曲率凸优化问题来修改路径。重复上述顺序求解的速度和轨迹的过程，直到圈速提高小于一个小的

常数ε。优化算法计算每一次的转向输入δ^*以及由此产生的车辆横向状态向量X^*。为了获得新路径的s和K，首先需要用式（3-16）和式（3-17）更新路径的$(E_k，N_k)$坐标

$$E_k = E_r - e_k^* \cos(\Psi_{r,k}) \tag{3-16}$$

$$N_k = N_r - e_k^* \sin(\Psi_{r,k}) \tag{3-17}$$

其中，$\Psi_{r,k}$是原始路径以及经过迭代后的期望路径的朝向角。接下来，通过包含在X^*内的车辆航向角Ψ^*和车辆质心与期望路径之间的横向偏差e^*更新每次迭代后路径，新路径由式（3-18）和式（3-19）所示的计算方式近似得到

$$s_k = s_{k-1} + \sqrt{(E_k - E_{k-1})^2 + (N_k - N_{k-1})^2} \tag{3-18}$$

$$\mathrm{K}_k = \frac{\Psi_k^* - \Psi_{k-1}^*}{\mathrm{s}_k - \mathrm{s}_{k-1}} \tag{3-19}$$

以图 3-9 所示的伪代码形式表示迭代生成车辆轨迹以及速度分布的整体算法。具体的优化轨迹生成算法见附录 A。

```
1: procedure 轨迹生成（沿赛道距离，赛道曲率，w_0^in，w_0^out）
2:         参考轨迹←（沿赛道距离，赛道曲率，w_0^in，w_0^out）
3:         while Δt*>ε do
4:             速度分布←计算速度分布（参考轨迹）
5:             参考轨迹←计算凸优化问题（速度分布, 参考轨迹）
6:             t*←计算圈速（速度分布, 参考轨迹）
7:         end while
8:         return 优化轨迹， 速度分布
9: end procedure
```

图 3-9 轨迹生成算法整体构成

3.6 本章小结

本章首先对赛车优化轨迹生成算法的提出思路进行概述，接下来对赛道进行参数化处理，详细说明了生成速度分布阶段的内容以及使用凸优化方法生成轨迹的原因，同时进行了计算公式推导和伪代码整理，最后对整个算法的整体思路进行总结，并通过编程的方式将此算法从理论转化为可运行的 Python 语言。

4 优化轨迹生成

4.1 CARLA 内置逻辑概述

CARLA 以 Unreal Engine 4 为基础来运行模拟，并使用 OpenDRIVE 标准来定义道路和城市设置，而模拟的控制权是通过 Python 和 C 语言处理的 API 授予的，CARLA 被设计成一个服务器-客户端系统。服务器负责与模拟本身相关的一切：传感器渲染、物理计算、世界状态及其参与者的更新等等。由于它的目标是获得实际的结果，最好的方法是使用专用的 GPU 运行服务器。客户端 API 用 Python 实现与服务器之间的交互，客户端向服务器发送命令和元命令，并接收传感器读数。命令控制车辆，包括转向、加速和制动。元命令控制服务器的行为，用于重置模拟、更改环境属性（天气条件、照明以及汽车和行人的密度等）和修改传感器套件。客户端是用户运行以请求模拟器中的信息或更改的模块，它们连接到服务器、检索信息和命令更改, 这是通过脚本完成的。客户明确自己的身份，并连接到世界，它通过终端与服务器通信。客户端的创建需要两个参数：标识它的 IP 地址，以及与服务器通信的两个 TCP 端口，可选的第三个参数为工作线程的数量，默认情况下该值设置全部为 0。一般情况下 CARLA 使用本地主机 IP 和端口 2000 进行连接，但这些可以随意更改。我们只需要设置第一个 TCP 端口的端口号，第二个端口为 N+1。基于以上逻辑可以认为服务器等同于模拟器，而客户端−服务器通信是通过 Python 应用编程接口控制[35]。

在开始控制算法之前，必须完成与服务器的连接以及模拟器和 actor 的所有设置。对模拟器和 actor 进行配置的基础逻辑如下：首先创造一个可以将命令发送给模拟器的客户端：client = carla.Client('127.0.0.1', 2000)创建客户端后，设置其超时（time-out）。这会限制所有网络操作，使它们不能一直阻塞客户端。如果连接失败，将返回错误。其表示形式为：client.set_timeout(10.0)。通过创建出的客户端可以检索想要运行的 world：world = client.get_world()。检索出的 world 中包含 blueprint_library： blueprint_library = world.get_blueprint_library()，我们可以在这里过滤出想要的 actor，也可以设置其属性以及更改位置。接下来可在此基础上进行客户端与服务器通信，这部分是 Python 脚本的主要内容，控制算法包含其中，也在这一部分进行模拟。模拟结束后销毁 actor，客

户端断开连接。

由于官方对 CARLA 软件的更新速度较快，且不同版本拥有的功能、与之相匹配的 Python API 环境、CARLA 的环境配置以及调用内置函数的语句有较大区别，因此此次毕设选用近期更新的 0.9.10 版本，其为目前较稳定的版本且功能较为全面，与其匹配的 API 环境为 Python3.7。CARLA 中的*. egg 文件使得所有的依赖文件以及模块能够在服务器端的 Unreal Engine 和客户端的 Python 之间兼容，即实现从独立的 Python 进程中调用 Unreal Engine 的函数，而这一文件可通过在 Python 脚本中加入系统路径的方式直接导入，图 4-1 表示了其间的相互关系。

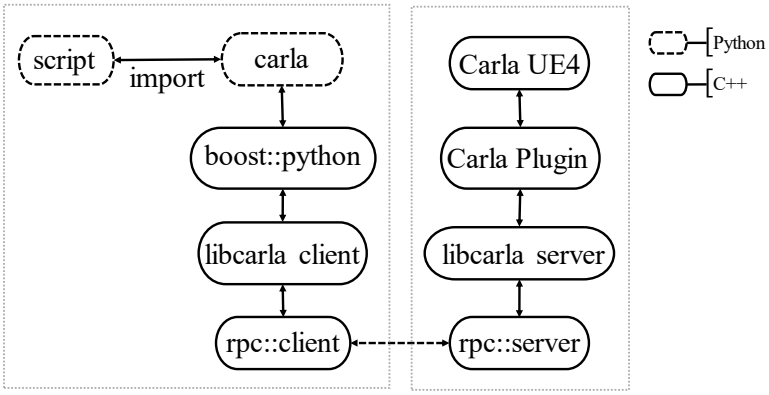


图 4-1 联合 CARLA 与 UE4 的方式

4.2 构建自定义地图

CARLA 模拟器可以重新创建场景，且数据收集可以依赖于几种传感器和摄像机，同时本次使用 CARLA 模拟器的目的是在某赛道上利用第三章给出的轨迹生成算法生成轨迹，再通过控制算法使车辆沿生成轨迹运动。因此首先需要创建可以在 CARLA 中运行的地图，并以此为基础进行轨迹生成及轨迹跟踪。

雷山赛道是位于美国加利福尼亚州威洛斯以西 7 英里处的萨克拉门托谷的一个赛车场，其为美国赛程最长的汽车比赛——雷山 25 小时耐力赛的举办地，在每年 12 月的第一个周末举行。雷山赛道由两部分组成，其中最初的 3 英里的赛道被称为雷山东部赛道，而另一个 2 英里的赛道被称为雷山西部赛道。这两条赛道也可以合并为一条 5 英里长的赛道，也是美国最长的公路赛道，此次便以雷山赛道为道路基础进行优化轨迹生成算法的验证。

4.2.1 获取赛道资源

CARLA 0.9.10 提供了导入新建地图的功能，其能够通过可用的 OpenDRIVE（.oxdr）

格式文件生成可在 CARLA 中运行的道路基础。为了获取赛道的基础地理信息，首先确定一个提供地理数据的开源数据库，其中的数据需要包含关于道路的几何形状及其特征。OpenStreetMap 是目前唯一有效的免费选择，它可以提供必要的数据用于地图的生成，其导出的.osm 文件格式也是广泛使用的标准。在 OpenStreetMap 的官方网站可以通过框选的方式导出所需区域的.osm 文件如图 4-2 所示。

图 4-2 选中的 Thunderhill 赛道部分

4.2.2 赛道格式转换

接下来根据 CARLA 官网的信息，需要将.osm 格式的文件转换为.oxdr 格式，进行格式转换并最终导入 CARLA 的方式很多。例如 CARLA 官方提供了用编写 Python 脚本的方式，但是实际用程序转换得到的.oxdr 文件在被 CARLA 提取后显示在虚幻引擎中会发生变形，查阅了很多资料也没有解决这一问题。后续在 RoadRunner 的官方网站教程中发现 RoadRunner 支持将.osm 文件转换为 CARLA 可识别的文件，但是显示在 RoadRunner 界面上的地图依旧是残缺的，也无法找到有效的方式。最终在论文[36]中再次找到一种转换的方式。Simulation of Urban Mobility（SUMO）模拟软件中的 netconvert 工具允许导入不同来源的道路网络，然后生成一个新文件作为输出，该文件的格式由命令行中所指定。SUMO 是一个开源的微观交通模拟器，允许模拟和管理道路交通，它以包的形式提供用于生成、验证和评估交通场景的工具。实现从. osm 到.oxdr 格式的

转换可在电脑终端中运行如下命令：netconvert--map.osm--opendrive-output map.xodr。

4.2.3 赛道修剪及配置

首次在 RoadRunner 中打开的.oxdr 格式地图文件如图 4-3 所示，可以明显看出应用 OpenStreetMap 获取的地图信息是框选范围内的所有道路，而所需的只有由雷山东部赛道和雷山西部赛道组合的闭环部分，因此需要将生成的.oxdr 文件导入到 RoadRunner 中进行调整修改。RoadRunner 是一个交互式编辑器，可以由使用者设计 3D 场景来模拟和测试自动驾驶系统。使用者可以通过创建特定区域的道路标志和交通标识来定制道路场景，例如插入交通信号、护栏、道路损坏以及树叶、建筑物和其他 3D 模型。首先需要使用 RoadRunner 对多余的道路、场地等部分去除，接下来需要对道路的属性进行定义。由图 4-2 可以看到，雷山赛道为单向行驶的赛道，因此在 RoadRunner 中需要赋予道路单行的属性，选取道路样式中的 oneway 属性并将赛道的方向设置为实际赛道规定的逆时针，如图 4-4 所示。最后使用 Road Plan 工具对转角位置进行平顺处理，以防导入 CARLA 模拟器后出现识别问题。注意到 RoadRunner 和 CARLA 自带的虚幻引擎 UE4 之间使用相同的坐标系，因此可以通过在 RoadRunner 中使用 World Settings 工具设置地图的坐标原点，此次将坐标原点设置到图 4-5 所示的位置。

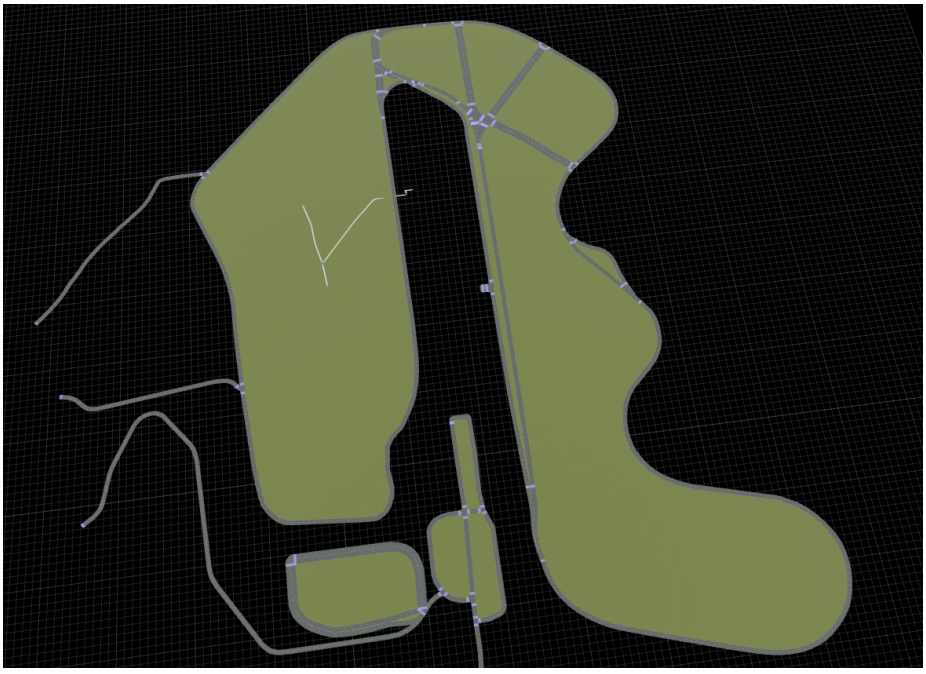


图 4-3 导入 RoadRunner 的原始赛道

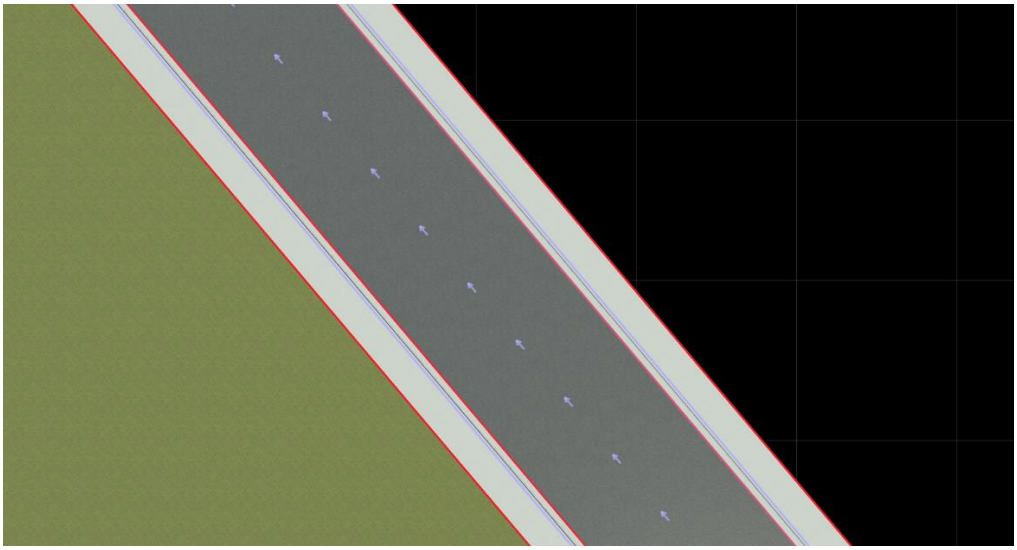


图 4-4 设置赛道逆时针单行属性

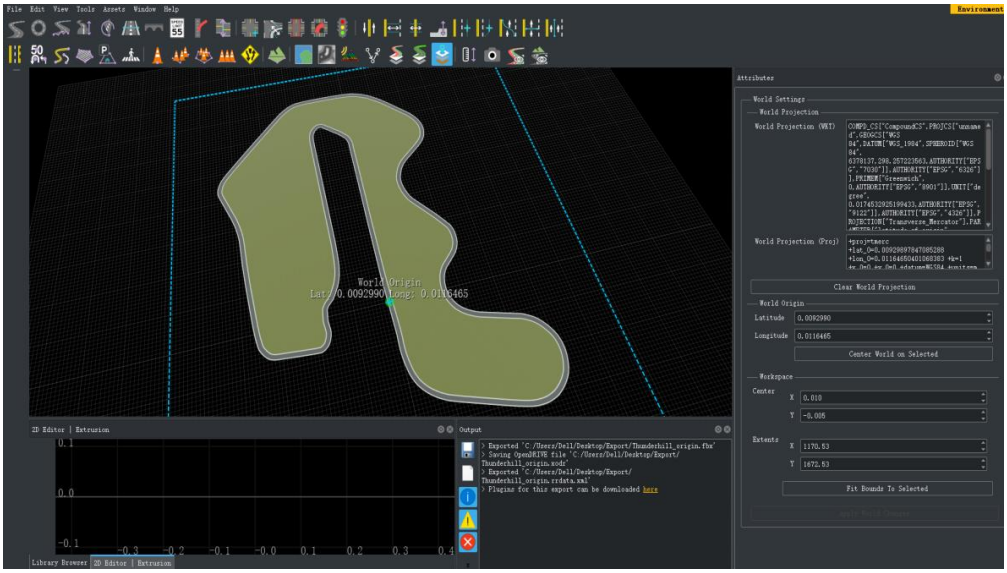


图 4-5 经过 RoadRunner 处理的赛道

经过修剪最终得到如图 4-5 所示的闭环赛道。导出调整好的.xodr 文件后便可以利用 CARLA 提供的 API 配置文件，将其提取到 CARLA 的服务器端，其在服务器端显示的形式如图 4-6 所示。另外，还可以在 CARLA 编译后得到的 UE4 编辑器中创建新地图，或者对创建的新地图进行修改。UE4 编辑器一般用来对 RoadRunner 绘制的地图进行进一步的处理，并在处理后通过自动开启 CARLA 和 UE4 的连接，直接将地图导入 CARLA 的服务器端。这一编辑器还具有其他更复杂的功能，但其操作也相应比较复杂且编译启动时间的时间很长，由于此次需要的道路并不复杂且只涉及对道路的修剪，因此没有采用这一方式编辑和导入地图。此外，还有一种方式是通过 CARLA 提供的 OpenDRIVE standalone mode 加载地图。此方法仅使用 OpenDRIVE 文件运行完整

模拟，而不需要任何其他几何图形或资源，为了实现这一点，模拟器将自动生成一个道路网格供 actor 导航。模拟器获取一个 OpenDRIVE 文件，并按程序创建一个临时三维网格来运行模拟，使用时需要在程序脚本中设置网格参数。由于 config.py 文件中包含了对 CARLA 固定的设置，参数不便于调整，因此选择 OpenDRIVE standalone mode 方法，调用 client.generate_opendrive_world 函数将自定义地图作为被获取的世界。总结以上方法，可由图 4-7 表示在 CARLA 中创建赛道的过程。

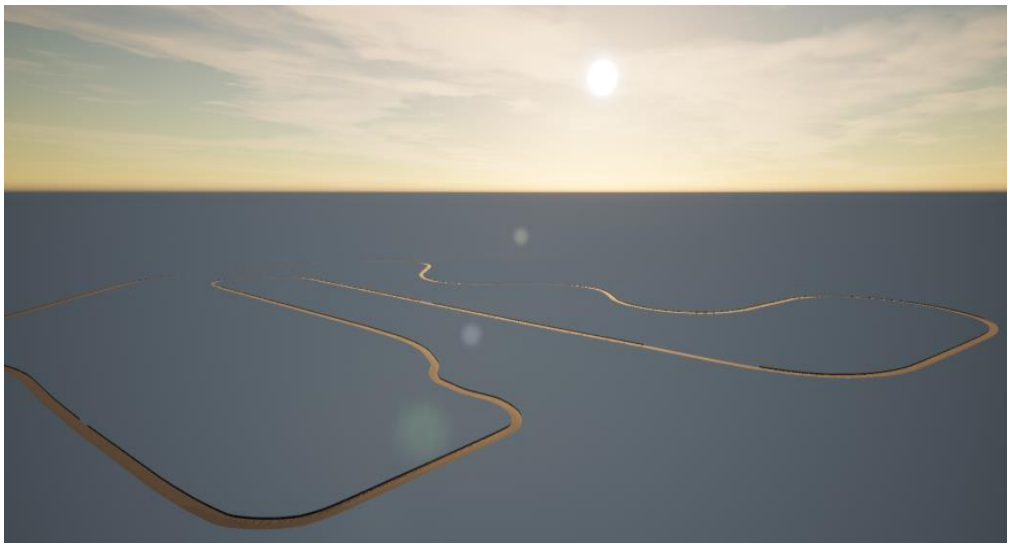


图 4-6 将赛道导入 CARLA

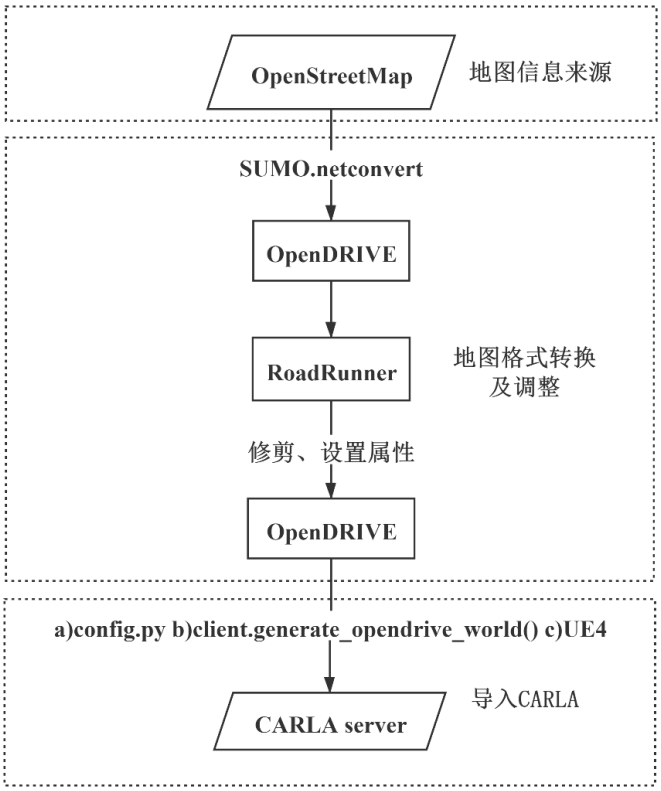


图 4-7 创建赛道操作流程图

4.3 基于 CARLA 客户端进行赛道参数计算

4.3.1 计算赛道中心线样本点的位置坐标

根据 CARLA 0.9.10 官方网站提供的信息，调用 map.generate_waypoints()函数可以在 CARLA 中根据导入的地图的网格信息自动生成 waypoints，为了能够清楚了解通过调用这一函数生成的 waypoints 的特性，进行了如下测试。首先使用 RoadRunner 绘制一个有五条车道的弯道地图，将其用前文所述的方式导入 CARLA 中，编写 Python 脚本输出生成 waypoints 的位置坐标到文本文件中并调用 CARLA 内置 DebugHelper 类中的 draw_point 函数在地图上显示 waypoints 的位置。通过文本文件中的坐标信息以及服务器端显示的点可以直观地说明其生成的原理是在每条路的中心线处等间距产生，其结果如图 4-8 所示。接下来将先前处理好的雷山赛道导入 CARLA，编写 Python 脚本将生成的 waypoints 按顺序输出为.csv 文件，但是在这一文件中插入连线散点图后得到图 4-9 所示的结果，这也说明调用这一函数生成 waypoints 存在顺序问题。

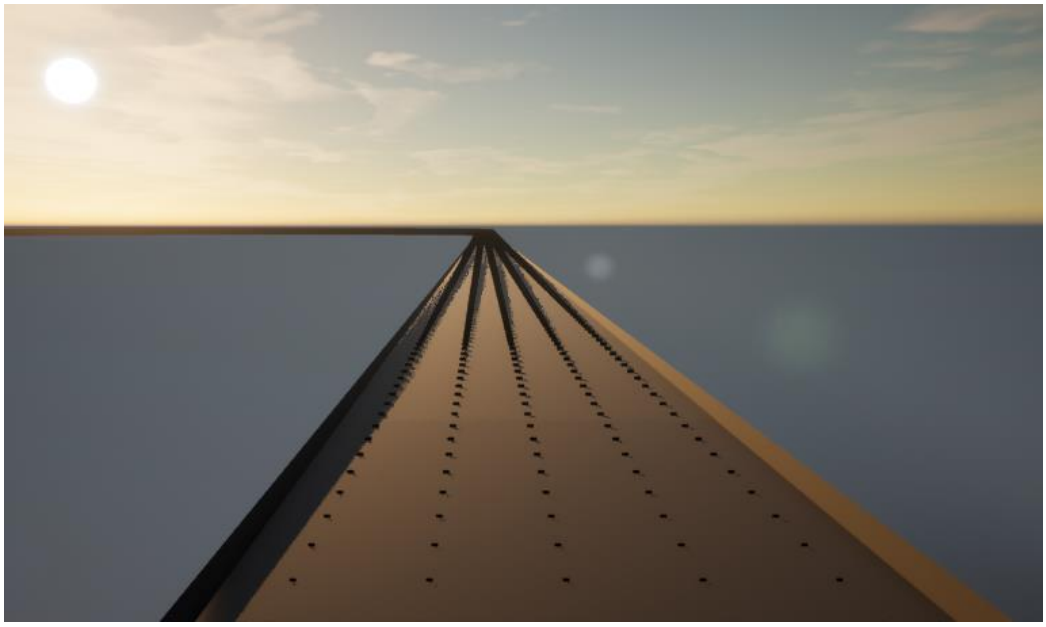


图 4-8 标记道路上的 waypoints

后续通过查找资料发现 scenario_runner 模块中的 interpolate_trajectory 函数可以实现沿道路的方向属性生成 waypoints 的功能。由于 scenario_runner 并不是 CARLA 的自带包，需要进行 git clone 以及 bashrc 配置才能使用。这一函数需要指定起点和终点的坐标以及中心线样本点的间距从而生成道路中心的路径，从定义这一函数的 route_manipulation.py 文件中也可以发现这一函数的特点和方法。在 Python 脚本中调用 map.get_waypoint(carla.Transform(carla.Location(x=, y=, z=)))函数可以获取距离这一坐

标最近的一个 waypoint 点，用此方式定义路径生成的起点。由于赛道的确切长度不能得到，同时将起点和终点设置为同一个 waypoint 时并不会生成路径，因此使用距离起始点一段距离的 waypoint 作为终点，可在起点坐标的基础上加上 carla.Vector3D(0, -30, 0)，将由此得到的所有中心线样本点。

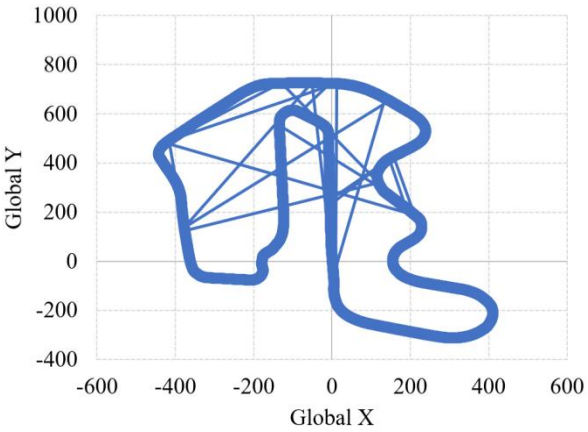


图 4-9 赛道中线折线散点图 1

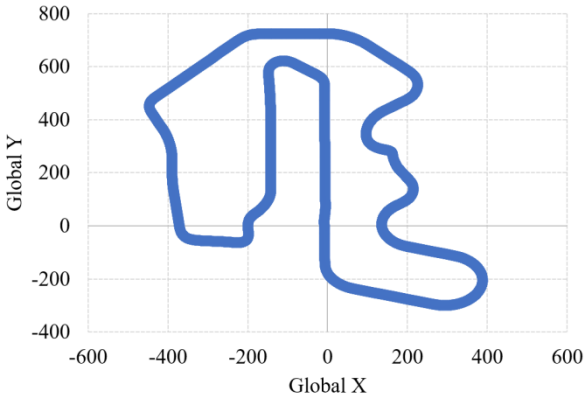


图 4-10 赛道中线折线散点图 2

由于样本点数量较多，将加大采样间距后的部分样本点表示在地图上如图 4-11 所示。接下来可通过互换起点和终点，得到剩余路径上的所有中心线样本点。将两次得到的中心线样本点坐标组合在一起，再次通过连线散点图验证如图 4-10 所示，于是得到了沿闭环赛道的中心线坐标。使用 CARLA 客户端程序，将获取到的部分样本点表示在模拟器端得到图 4-11。

图 4-11 引入 scenario_runner 模块生成的部分 waypoints

基于以上尝试，在 CARLA 客户端中获取赛道中心线样本点位置坐标文件的过程如图 4-12 所示。

```
1:procedure 获取中心线样本点
2：     import carla
3:      import srunner
4:      import interpolate_trajectory
5:      try
6:          连接客户端
7:          导入地图←（.xodr 文件）
8:          获取世界
9:          trajectory←interpolate_trajectory（world，起点，终点）
10:         for i←0 to len(trajectory)
11:             waypoints 数据←（i，trajectory）
12:         end for
13:         return waypoints 数据
14:         保存文件←（文件名，waypoints 数据）
15:     finally
16:         销毁 actor
17:end procedure
```

图 4-12 获取赛道中心线样本点位置坐标文件的伪代码

4.3.2 计算赛道中心线样本点的其他参数

为进行第三章所述的优化轨迹生成算法运算，还需要获取赛道的朝向、曲率、宽度以及中心线所有样本点距离起点的沿赛道距离。

由于生成中心线样本点时并没有使用 CARLA 自带的自动生成 waypoints 点的方法，因此无法通过调用 CARLA 语句计算样本点处的朝向。但是由于样本点间距很小，所以采用近似计算的方式可以得到所有样本点的朝向角、曲率以及距离起点的沿赛道距离。通过使用 atan2(y, x)函数计算每个样本点前后相邻样本点与横坐标轴之间的夹角，并通过如图 4-13 所示的几何关系，计算出每个样本点处的朝向角ψ_R，其计算公式如式（4-1）所示。曲率可通过获得每个样本点前后相邻样本点的位置坐标及朝向角以式（4-2）的方法计算。

$$\psi_R = \pi - \left(\frac{\pi}{2} - \left(\frac{\theta_1 - \theta_2}{2} + \theta_2\right)\right) = \frac{\pi + \theta_1 + \theta_2}{2} \tag{4-1}$$

$$K = \frac{2\sin\alpha}{d} \tag{4-2}$$

其中，α表示两样本点之间的朝向角差值，d 表示两样本点之间的距离。而通过按顺序累加相邻两样本点的方式，便可以获得每个样本点距离起点的沿赛道距离列表。由于获取的路径边缘信息并不精确，因此在利用 RoadRunner 修剪赛道时，将赛道宽度统一为定值。

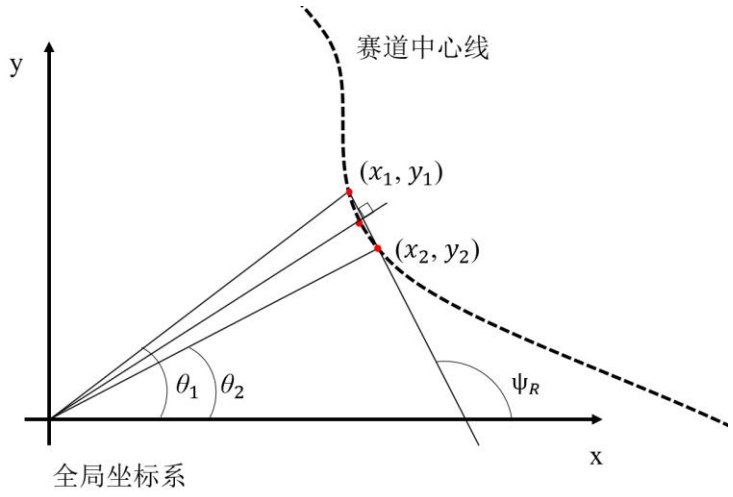


图 4-13 计算中心线样本点处朝向角示意图

对于样本点朝向角的计算，式（4-1）为主要运算过程，以伪代码的形式表示在 CARLA 客户端中的编程过程如图 4-14 所示。

```
1: procedure 获取赛道朝向角（waypoints 文件）
2:          赛道朝向角列表←空列表
3:          waypoints 矩阵←（waypoints 文件）
4:          for i 0 to（len(waypoints 矩阵)-1）
5:                  x1， y1， x2， y2←（waypoints 矩阵，i）
6:                  θ1， θ2←（x1， y1， x2， y2）
7:                  赛道朝向角←（θ1， θ2）
8:          end for
9:          return 赛道朝向角数据
10:         保存文件←（文件名，赛道朝向角数据）
11: end procedure
```

图 4-14 计算样本点朝向角的伪代码

对于样本点处曲率的计算，以式（4-2）为主要运算过程，以伪代码的形式表示在 CARLA 客户端中的编程过程如图 4-15 所示。

```
1: procedure 获取赛道曲率（waypoints 文件，赛道朝向角文件）
2:          赛道曲率列表←空列表
3:          waypoints 矩阵←（waypoints 文件）
4:          赛道朝向角矩阵←（赛道朝向角文件）
5:          for i 0 to（len(waypoints 矩阵)-1）
6:                  x1， y1， x2， y2←（waypoints 矩阵，i）
7:                  yaw1， yaw2←（赛道朝向角矩阵，i）
8:                  赛道曲率←（x1， y1， x2， y2， yaw1， yaw2）
9:          end for
10:         return 赛道曲率数据
11:        保存文件←（文件名，赛道曲率数据）
12: end procedure
```

图 4-15 计算样本点曲率的伪代码

为了获得每个样本点距离起点的沿赛道距离，以伪代码的形式表示在 CARLA 客户端中的编程过程如图 4-16 所示。获取赛道参数的具体程序见附录 B。

```
1: procedure 获取沿赛道距离列表（waypoints 文件）
2:          赛道沿赛道距离列表←空列表
3:          waypoints 矩阵←（waypoints 文件）
4:          for i 0 to（len(waypoints 矩阵)-1）
5:                  x1， y1， x2， y2←（waypoints 矩阵，i）
6:                  沿赛道距离←（x1， y1， x2， y2）
7:          end for
8:          return 沿赛道距离数据
9:         保存文件←（文件名，沿赛道距离数据）
10: end procedure
```

图 4-16 计算样本点沿赛道距离的伪代码

4.3.3 优化轨迹生成及分析

以获取到的赛道信息作为输入，运行优化轨迹生成算法，得到优化轨迹如图 4-17 所示，其上的速度分布如图 4-20 所示，计算得到赛车的圈速为 160.31s，在 Colab 中以随机存取存储器为 12.69G、磁盘为 107.77G 的配置运行该算法并绘制出生成轨迹及速度分布图所用的时间为 83s。

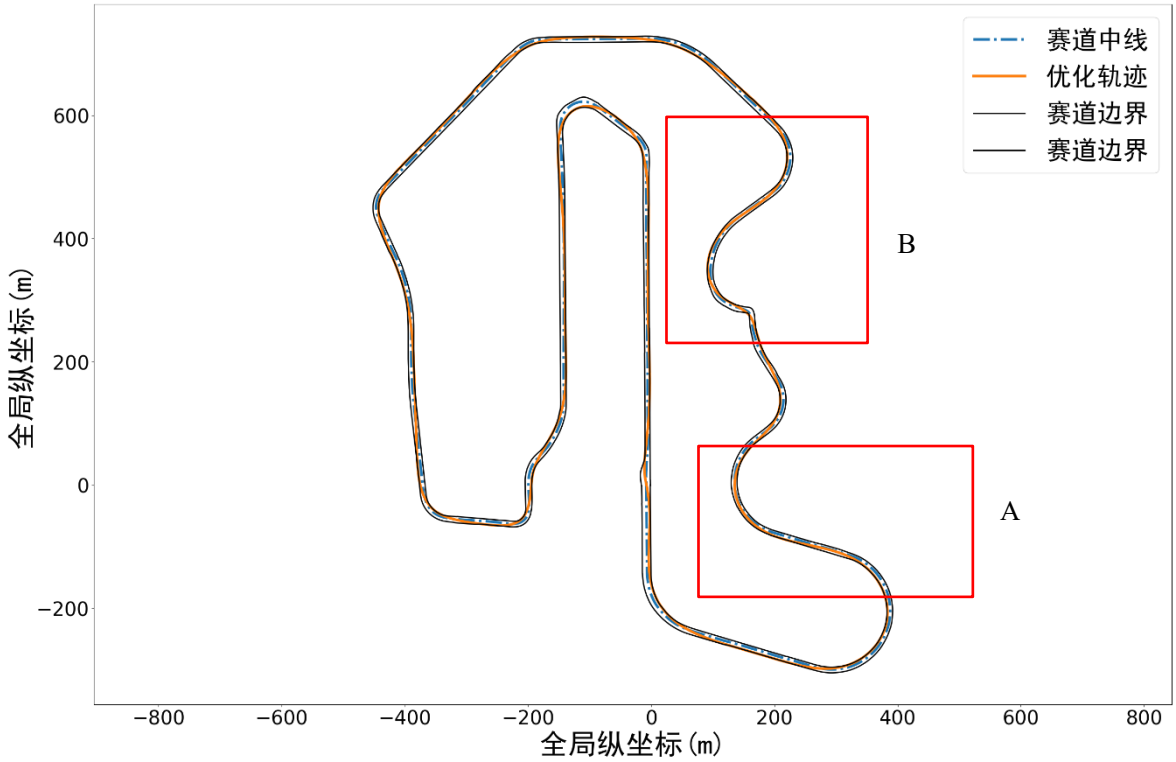


图 4-17 优化轨迹结果

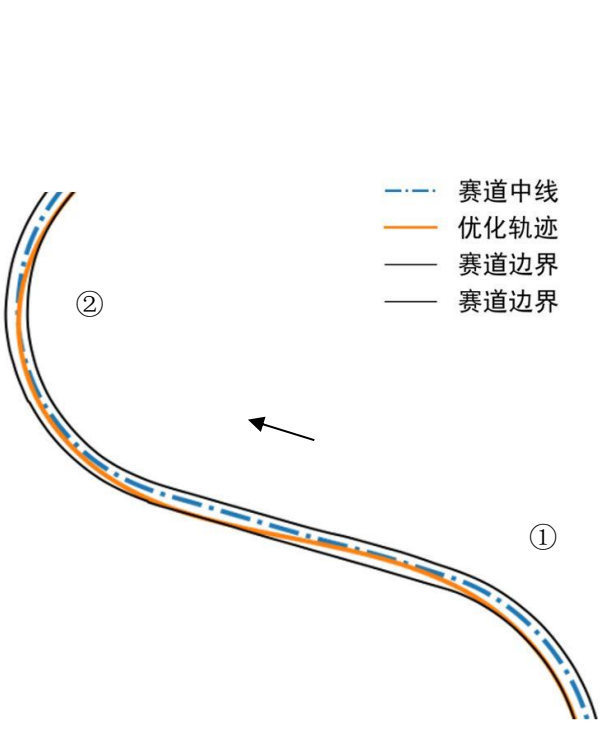


图 4-18 局部放大 A 处优化轨迹结果

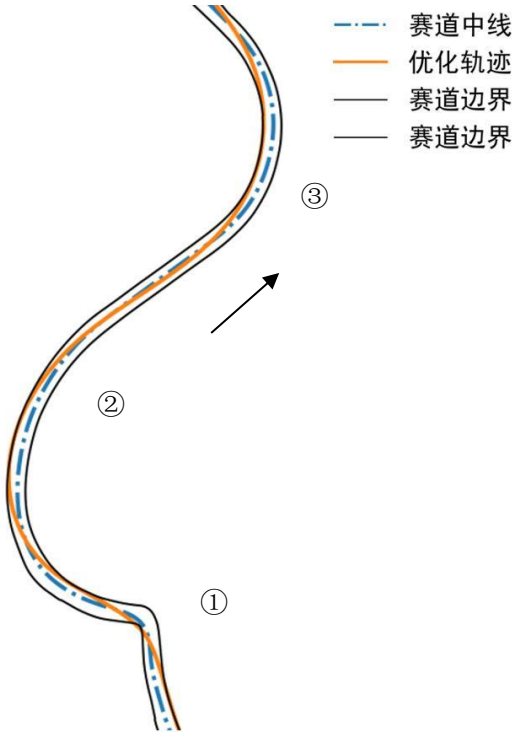


图 4-19 局部放大 B 处优化轨迹结果

由图 4-17 可见，生成轨迹与预期的较为相似，即在大部分入弯过程中算法会选择沿外侧赛道行驶，在弯道部分算法会选择贴近弯心的方式过弯，而出弯时算法会选择沿外侧赛道，以此使得过弯的曲率较小。而对于赛道中存在连续弯道的部分，生成的

轨迹也着重考虑到了缩短总体圈速，因此并不会完全遵循上述的过弯方式。将图 4-17 的优化轨迹结果的 A、B 部分局部放大，如图 4-18、图 4-19 所示，其中箭头指示的方向为赛车行驶的方向。如图 4-18 所示，第一次出弯过程中并没有很贴近外侧赛道，同时在与下一个弯道衔接的部分调整位置，以此保证第二次过弯的曲率不至于过大。如图 4-19 所示，由于弯道①和②之间连接较为紧密，为了以较快的速度通过且不偏离赛道，在第一个弯道算法会选择以较小的曲率通过，并将紧密连接到第二个弯道的弯心，而在向第三个弯道过渡的过程中开始偏向外侧，为在第三个弯道过弯准备。

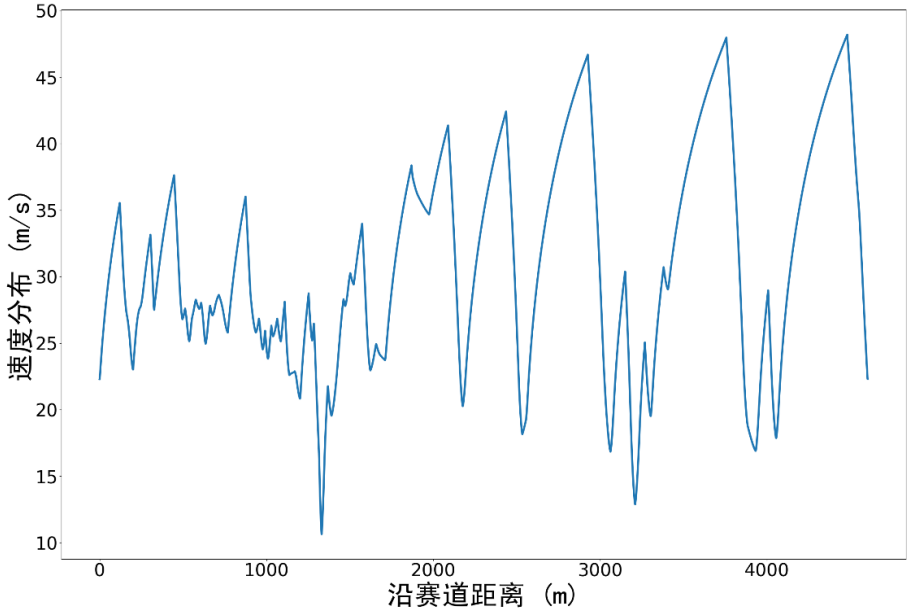


图 4-20 速度分布结果

4.4 本章小结

本章首先对 CARLA 模拟器的内部逻辑进行介绍，并以此为基础实现了雷山赛道地图的搭建。接下来探究了其中 waypoints 生成的机理，并在此基础上通过 CARLA 模拟器客户端获取了优化轨迹生成算法所需的赛道参数文件，包括赛道中心线样本点的坐标、朝向、曲率、宽度以及距离起点的沿赛道距离。以这一赛道参数文件作为输入运行优化轨迹生成的程序，最终得了优化轨迹以及其上的速度分布。

5 轨迹跟踪仿真

5.1 轨迹跟踪仿真简介

目前的自动驾驶联合仿真一般遵循的逻辑为，测试平台中的仿真车辆将传感器检测数据发送给自动驾驶软件，自动驾驶软件经过一系列感知、定位、决策、控制环节，将最终的控制信号传递回仿真车辆，驱动汽车安全行驶，其间的关系可由图 5-1 表示。对于自动驾驶研究，闭环仿真测试系统是必不可少的步骤，新的算法都应该经过初步的仿真测试才能进入实车验证阶段。

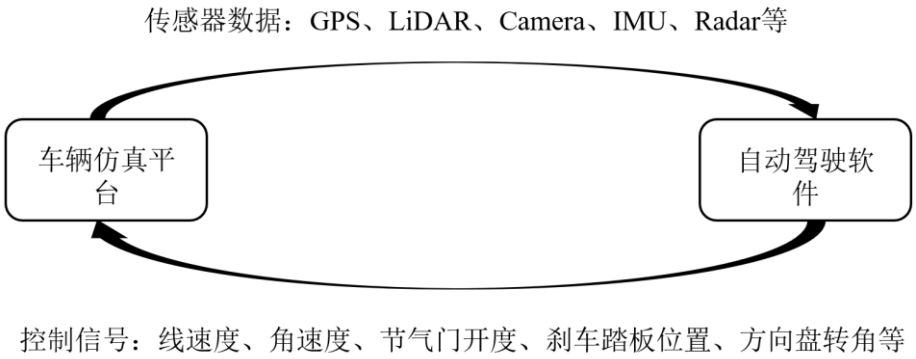


图 5-1 闭环仿真测试基本框架

市面上有很多与自动驾驶仿真相关的软件。第一类是专门为自动驾驶开发出的模拟仿真软件，如 Prescan、VTD、51sim-one、Panosim、GaiA 等等。第二类是基于游戏引擎做的自动驾驶仿真软件，主要代表是基于 Unity 的 LGSVL Simulator、Baidu-Unity 和基于 Unreal 的 CARLA、CarSim 等。第三类是基于一些机器人仿真软件做的自动驾驶仿真器，如基于 ROS 的 Gazebo、Rviz 开发的仿真平台、基于 blender 开发的平台等等。自动驾驶软件也有很多种。其中通过 Python 脚本的形式实现端到端自动驾驶的有 Udacity 的 Behavioral Cloning 和 CARLA 的 Conditional Mitation Learning；也可以通过一个非常庞大的软件套装来实现自动驾驶感知、定位、决策、控制等环节，而这一软件的实质就是感知、定位、决策、控制等环节的算法组合。目前来看，业内影响力比较大的自动驾驶软件主要是由名古屋大学研究人员开发，目前由 Tier Ⅳ公司负责维护的 Autoware 以及百度发布的 Apollo，并且两者都是开源的，也都有自己的开发社区。

在此次的仿真中 CARLA 模拟器作为车辆仿真平台，而编写的 Python 脚本起到自动驾驶软件中控制部分的作用。从而使 CARLA 模拟器与使用 Python 脚本编译的轨迹跟踪控制器形成闭合的信息流，在控制算法运行的过程中，可通过调用获取车辆目前状态的语句使 CARLA 车辆仿真平台将检测数据发送给控制器，控制器接受所需的参数，经过计算将最终的控制信号传递回仿真车辆，驱动汽车按照期望的路线行驶。

5.2 轨迹跟踪控制

5.2.1 横向控制理论——PurePursuit

横向控制为通过转向角缩小车辆与参考路径之间的横向误差，包括横向朝向误差和横向距离误差两方面。机器人学中最流行的路径跟踪方法之一是几何路径跟踪器，这些方法利用车辆和路径之间的几何关系找到解决路径跟踪问题的控制方法。这些方法通常利用前瞻距离来测量车辆前方的误差，也会涉及简单的圆弧计算甚至更复杂的螺旋理论的计算[37]，本文将介绍并使用其中的两种算法实现 CARLA 虚拟环境中车辆的轨迹跟踪控制。

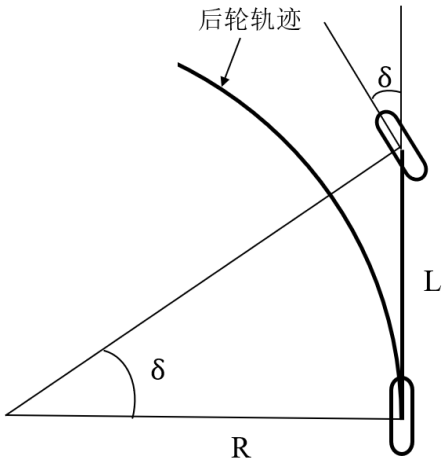


图 5-2 车辆单轨几何模型示意图

实施基于几何关系的路径跟踪控制时，通常将车辆简化为单轨模型，其通过将两个前轮和两个后轮结合在一起的方式实现了使用两轮模型来简化四轮车辆，同时假设车辆只能在平面上行驶。这些简化将导致前轮转向角与后轮轴行驶轨迹的曲率之间遵循简单的几何关系，如图 5-2 所示，这个简单的几何关系可以由式（5-1）表示

$$\tan(\delta) = \frac{L}{R} \tag{5-1}$$

其中，δ是前轮的转向角，L是前轴和后轴之间的距离，R是后轴在给定转向角下行驶的圆的半径。

PurePursuit 算法通过计算将车辆从当前位置移动到某个目标点的曲率来实现控制功能，其最重要的步骤在于选择一个在路径上领先车辆一定距离的目标位置，Pure Pursuit 这一名称也是对于该方法的一种描述类比。可以认为算法的整体思路是使车辆追赶路径上一个移动的目标点，而这一做法和人类的驾驶方式很相似，即驾驶员驾驶过程中会主动关注车前方一段距离处的并朝着这一位置驾驶车辆。

Shin[38]表明，如果车辆坐标系以后轴中点为坐标原点同时使后轴与 x 轴共线，那么前进与转向之间在几何关系上的解耦的，因而此算法以车辆后轴中心作为车辆的原点，表示车辆的位置。

将车辆坐标系、世界坐标系以及算法所需的参数表示在图 5-3 中，结合图 5-3 进行计算转向角公式的推导过程如下。

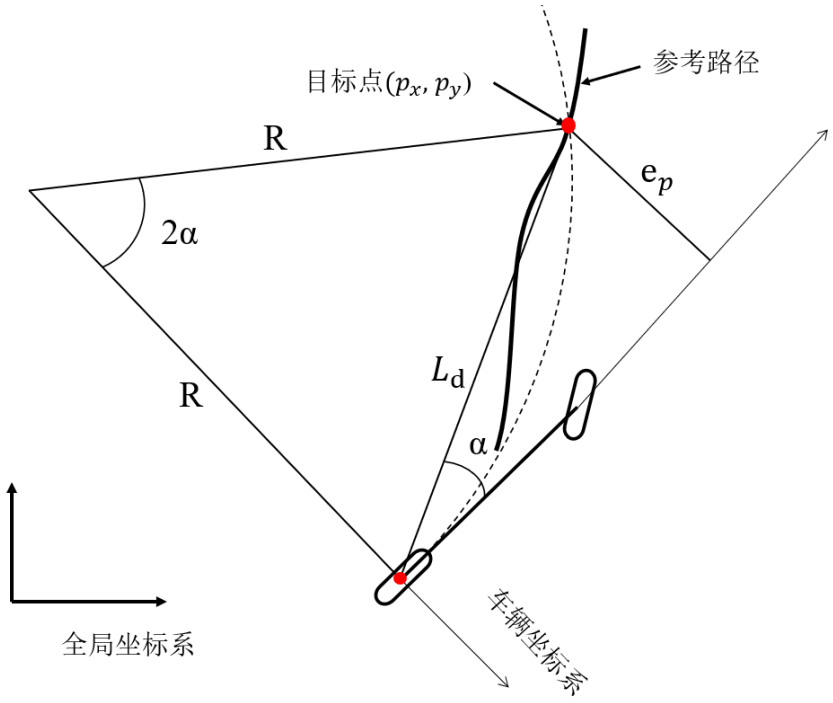


图 5-3 Pure Pursuit 几何模型示意图

结合图 5-3 中的等腰三角形，应用正弦定理如式（5-2）所示

$$\frac{L_d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)} \tag{5-2}$$

$$\frac{L_d}{2\sin(\alpha)cos(\alpha)} = \frac{R}{\cos(\alpha)} \tag{5-3}$$

$$\frac{L_d}{\sin(\alpha)} = 2R \tag{5-4}$$

其中，α为代表前瞻距离的弦与当前车身横摆角之间相差的角度。

也可以将式（5-4）表示为式（5-5）的形式

$$K = \frac{2\sin(\alpha)}{L_d} \tag{5-5}$$

其中，K为圆弧的曲率，将式（5-1）表示为式（5-6）的形式

$$\delta = \tan^{-1}(KL) \tag{5-6}$$

结合式（5-5）、式（5-6），将变量写为时间 t 的函数，得到转向角$\delta(t)$可由式（5-7）表示

$$\delta(t) = \tan^{-1}(\frac{2L\sin(\alpha(t))}{L_d}) \tag{5-7}$$

5.2.2 PurePursuit 控制算法分析及编程

定义e_p为目标点与车身之间的最短距离，即横向距离误差，结合图 5-3 中的几何关系得到式（5-8）

$$\sin(\alpha(t)) = \frac{e_p(t)}{L_d} \tag{5-8}$$

将式（5-8）带入式（5-5）得到式（5-9）

$$K = \frac{2}{L_d^2} e_p(t) \tag{5-9}$$

式（5-9）说明这一算法是基于横向距离误差的转向角比例控制器。

算法实施的整体过程为，先确定前瞻距离，即当前车辆原点与参考路径上目标点之间的直线距离。接下来，以实现车辆后轴中心沿连接当前车辆原点与参考路径上目标点的弧运动到目标点处为目的计算转向角。

确定L_d这一参数时需要考虑两方面的影响：第一是要能够收敛到参考路径，即如果车辆离路径有一个较大的距离，必须能够回到参考路径上；其次是能够保持参考路径，即如果车辆在参考路径上，则希望保持在路径上。对于第一个问题，通过类比人类驾驶员很容易得知改变参数的影响。较长的前瞻距离会实现逐渐地收敛到路径上，并且有较少的振荡。这一跟踪算法的响应类似于如图 5-4 所示的二阶动态系统的阶跃响应，而L_d的值有类似阻尼系数的效果。对于第二个问题，前瞻距离越长，参考路径上能够用来计算转向角的就越少。而该算法的主要部分就是计算转向角，从而使得车辆能够沿一条连接车辆原点和参考轨迹目标点的弧线运动，如果车辆和目标点之间的距离过长便无法计算出合适的转向角，难以使车辆保持沿参考轨迹行驶。因此此次将前瞻距离设置为速度的线性函数即L_d=$k_{vf}v + b$，以此考虑行驶速度的影响，同时由于

车辆的起始速度为零将导致此时分母为零，因此加上一常数 b。经过调试将k_{vf}设置为 0.1，将 b 设置为 2 时的效果较好，最终转向角可由式（5-10）表示

$$\delta(t) = \tan^{-1}(\frac{2L\sin(\alpha(t))}{k_{vf}v(t)+b}) \quad (5\text{-}10)$$

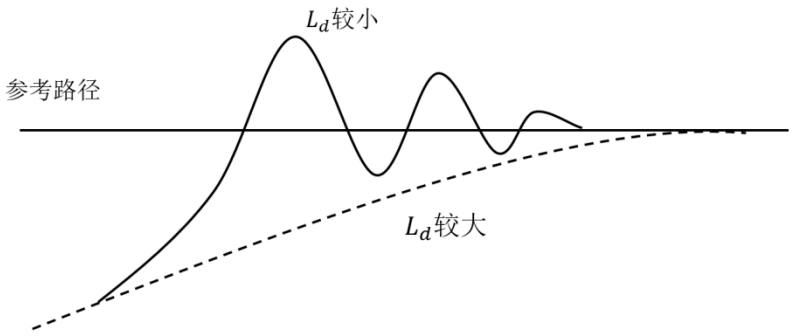


图 5-4 L_d对车辆行驶轨迹的影响

下文将对在 CARLA 中实现 PurePursuit 控制的编程逻辑进行简要的介绍。首先，在 CARLA 客户端程序中编写函数计算前瞻距离。接下来，需要获取使用上述前瞻距离寻找到的参考路径样本点，在将参考路径所有样本点的参数文本文件转换为矩阵的情况下，可以通过索引矩阵行数的形式，获取样本点的信息。在 PurePursuit 理论中，车辆的参考点为后轴中心，而在 CARLA 模拟器中调用获取车辆位置的函数，只能获取重心处的位置，因此需要对获取到的位置坐标进行转换。为实现 PurePursuit 理论推导出的转向角公式的计算，需要先计算出车辆参考点与目标点的连线与车身之间的夹角，其计算方法结合图 5-3 的几何关系，首先需要计算由前瞻距离L_d以及横向距离误差e_p构成的直角三角形的长直角边，在应用反余弦函数得到角 α，再根据公式计算转向角，最后依旧结合图 5-3 的几何关系考虑转角的方向问题。将上述过程以图 5-5 所示的伪代码表示。

```
1: procedure PurePursuit 横向控制方法（车辆速度，waypoints 列表，carla.vehicle）
2:         前瞻距离←（k_vf，车辆速度, b）
3:         参考轨迹上目标点的索引←（前瞻距离, waypoints 列表，carla.vehicle）
4:         车辆参考点←（carla.vehicle，前轴到重心的距离，后轴到重心的距离）
5:         转向方向←（参考轨迹上目标点的索引，waypoints 列表，carla.vehicle）
6:         转向角←（参考轨迹上目标点的索引，waypoints 列表，carla.vehicle）
7:         return 转向角
8: end procedure
```

图 5-5 PurePursuit 横向控制方法伪代码

5.2.3 横向控制理论——Stanley

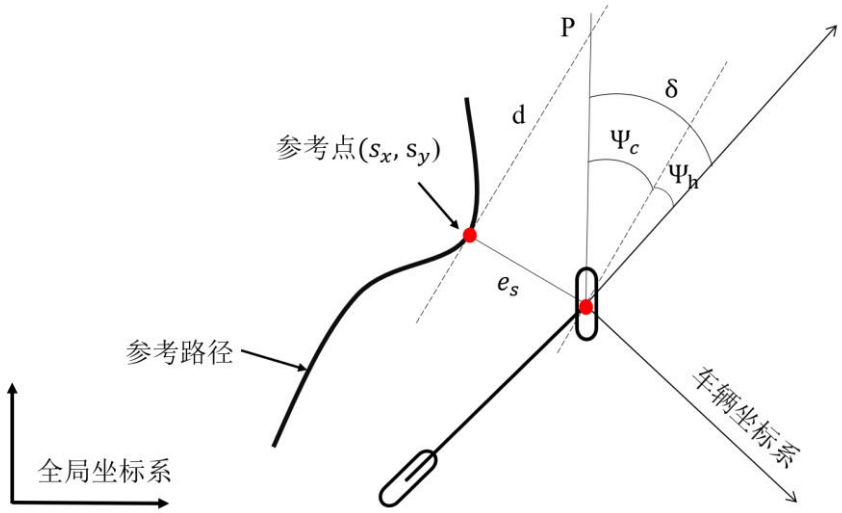


图 5-6 Stanley 几何模型示意图

Stanley 方法是斯坦福大学自动驾驶汽车斯坦利在 DARPA 挑战赛中使用的路径跟踪方法，图 5-6 列出了控制参数的几何关系。定义距离前轴中心最近的路径点（s_x，s_y）为参考点，将参考点处路径切线与前转向轮并排放置可直观地表示出控制算法第一项的给出原因，即第一项希望通过将转向角设置为等于航向误差的方式缩小横向朝向误差，其表示方法如式（5-11）所示

$$\Psi_{\mathrm{h}} = \Psi - \Psi_{\mathrm{r}} \tag{5-11}$$

其中，Ψ_{h}为转向角的第一项，Ψ为车辆当前的横摆角，Ψ_{r}为参考点处路径切线的朝向角。

用参数e_s表示前轴中心与参考点之间的距离，当e_s不为零时，将会通过添加第二项的方式来调整δ。假设车辆预期会在沿参考点处切线方向上与参考点距离 d 的位置与参考点处切线相交， 由图 5-6 所示的几何关系可得式（5-12）

$$\Psi_{\mathrm{c}} = \tan^{-1}\left(\frac{e_s}{d}\right) \tag{5-12}$$

将 d 与车速关联，定义一个增益参数k_{cte}，令$d = v/k_{cte}$，可将式（5-12）写为式（5-13）的形式

$$\Psi_{\mathrm{c}} = \tan^{-1}\left(\frac{k_{cte}e_s}{v}\right) \tag{5-13}$$

其中，v为车辆的速度，k_{cte}为增益参数。由此，这一项实现了同时考虑横向距离误差和车速对转向角的影响，即横向距离误差越大时转向角越大，而速度越大时转向角越小，同时应用正切函数控住大横向距离误差下的转向角。最后限制转向角的极限边界，

将变量写为时间 t 的函数，得到最终的转向角δ(t)的表达式如式（5-14）所示

$$\delta(t) = \Psi_h(t) + \tan^{-1}\left(\frac{k_{cte}e_s(t)}{v(t)}\right)，\ \delta(t) \in [\delta_{min}，\ \delta_{max}] \tag{5-14}$$

5.2.4 Stanley 控制算法分析及编程

根据图 5-6 所示的几何关系可得式（5-15）

$$\sin\Psi_c = \frac{e_s}{\sqrt{d^2 + {e_s}^2}} \tag{5-15}$$

将分子分母同时乘以增益系数k_{cte}，同时将前文定义的$d = v/k_{cte}$代入式（5-15）进行等量替换得到式（5-16）

$$\sin\Psi_c = \frac{k_{cte}e_s}{\sqrt{v^2 + (k_{cte}e_s)^2}} \tag{5-16}$$

由图 5-6 所示的几何关系还可得式（5-17）

$$\dot{e}_s = -v\sin\Psi_c \tag{5-17}$$

将式（5-16）代入式（5-17）中得到式（5-18）

$$\dot{e}_s = -\frac{k_{cte}e_s v}{\sqrt{v^2 + (k_{cte}e_s)^2}} = -\frac{k_{cte}e_s}{\sqrt{1 + (\frac{k_{cte}e_s}{v})^2}} \tag{5-18}$$

当e_s很小时，分母约等于 1，此时$\dot{e}_s \approx k_{cte}e_s$，即

$$\frac{de_s}{dt} = -k_{cte}e_s \tag{5-19}$$

将式（5-19）写为如下式（5-20）的形式

$$\frac{de_s}{e_s} = -k_{cte}dt \tag{5-20}$$

对式（5-20）进行积分运算，得到式（5-21），进而写为式（5-22）的形式

$$\ln e_s = -k_{cte}t + C_1 \tag{5-21}$$

$$e_s = e^{-k_{cte}t + C_1} = Ce^{-k_{cte}t} \tag{5-22}$$

因此 Stanley 方法是横向距离误差e_s的非线性反馈函数，对于任意横向误差e_s都可以指数收敛于 0，而参数k_{cte}决定了收敛速度。对于负指数函数，在横坐标为零时，其收敛的速度会发生变化，因此并不能在理论上找到这一数值的最优值，经过多次运行程序，将k_{cte}这一参数设置为 0.7 时的控制效果最好。

下文将对在 CARLA 中实现 Stanley 控制的编程逻辑进行简要的介绍。对于 Stanley 横向控制理论，其车辆的参考点为前轴的中点，因此还需要使用上文所述的函数进行

坐标转换，在 PurePursuit 控制中将 length 这一参数赋为负值，而在 Stanley 控制中将 length 这一参数赋为正值。在控制主函数中会选取一个样本点子集来导航车辆，这一子集以离车辆最近的样本点为起点，以向前一定距离的样本点为终点，因此可以第一个样本点的朝向作为参考计算朝向角误差。再计算 Stanley 理论中转角公式的第二项，同时也需要用上文所述的函数计算转角的方向，将两项的结果相加，得到最终的转向角。将上述过程表示为图 5-7 所示的伪代码形式。

1: **procedure** Stanley 横向控制方法（车辆速度，waypoints 列表，carla.vehicle, k_{cte}）
2: 朝向角误差项←（waypoints 列表，carla.vehicle）
3: 横向误差项←（车辆速度，waypoints 列表，carla.vehicle, k_{cte}）
4: 转向方向←（参考轨迹上目标点的索引，waypoints 列表，carla.vehicle）
5: 转向角←（朝向角误差项，横向误差项，waypoints 列表，carla.vehicle）
6: **return** 转向角
7: **end procedure**

图 5-7 Stanley 横向控制方法伪代码

5.2.5 纵向控制理论——PID

纵向控制为通过调节油门大小以及刹车踏板位置，缩小实际车速与参考速度之间的误差。其控制过程如图 5-8 所示，控制器获取到期望速度，通过首级控制输出所需的加速度，将这一输出作为次级控制的输入，进而输出节气门的开度以及刹车踏板位置等信号实现车速的调节。

对于首级控制，使用 PID 方法计算所需加速度，其计算公式如式（5-23）所示。

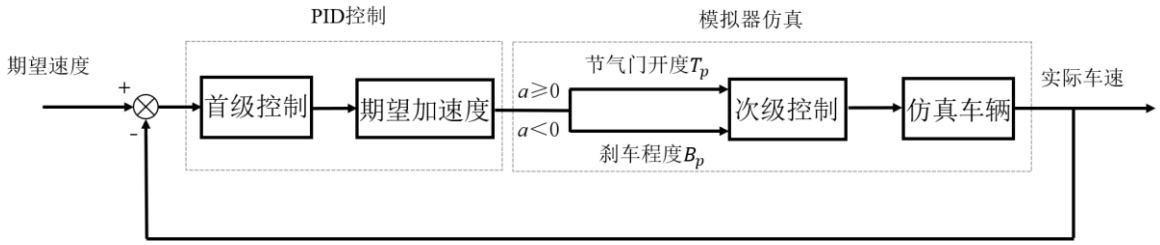


图 5-8 纵向控制过程示意图

$$\ddot{x}_{des} = K_P\left(\dot{x}_{ref} - \dot{x}\right) + K_I \int_0^t \left(\dot{x}_{ref} - \dot{x}\right) dt + K_D \frac{d\left(\dot{x}_{ref} - \dot{x}\right)}{dt} \tag{5-23}$$

其中，K_P、K_I、K_D分为比例项、积分项、微分项系数，$\ddot{x}_{des}$为所需的加速度，$\dot{x}_{ref}$为参考速度，$\dot{x}$为目前车辆的速度。

次级控制在 CARLA 车辆仿真平台中实现，以首级控制输入的参考加速度为基础，经由车辆纵向动力学模型计算出发动机转矩以及发动机的转速，结合发动机特性曲线得到此时的节气门开度，将其赋给仿真车辆。因此只需要计算首级控制的输出，作为 CARLA 服务器端的输入，若得到的数值为正值将其赋给油门，若得到的数值为负值则将其绝对值赋给刹车。

5.2.6 PID 控制算法分析及编程

由于轨迹生成算法得到的速度分布较高，因此在 PID 控制中将比例项的系数设置为较大的 0.95，以更快地跟踪速度，经过多次运行程序将积分项系数设置为 0.01，将微分项参数设置为 0.05 时运行效果较好。

在进行纵向控制时需要获取通过遍历计算样本点与车辆的距离，并得到距离车辆最近的样本点的行数索引，进而读取这一行第三列中存储的速度，将这一数值作为参考速度。接下来通过 PID 计算用以控制车辆节气门开度以及刹车程度的参数，若得到的数值为正，则将这一数值赋给节气门，若得到的数值为负数，将这一数值的绝对值赋给刹车，将上述过程表示为图 5-9 所示的伪代码形式。具体的横向控制算法以及纵向控制算法函数见附录 C。

```
1: procedure PID 纵向控制方法（参考速度列表，carla.vehicle，K_P，K_I，K_D，时间）
2:         参考速度←（参考速度列表，carla.vehicle）
3:         return 节气门开度，刹车程度←（参考速度，carla.vehicle，K_P，K_I，K_D，时间）
4: end procedure
```

图 5-9 PID 纵向控制方法伪代码

5.3 轨迹跟踪仿真

在 CARLA 中进行车辆轨迹和速度跟踪控制的基本逻辑如图 5-10 所示。图 4-17 所示的优化轨迹为高速极限工况下的运动，要实现轨迹的跟踪需要使用较复杂的控制算法并结合轨迹生成过程中使用的车辆动力学模型。由于目前无法得知和修改车辆仿真平台对车辆动力驱动模型以及动力学模型的建模方式，因此无法完全贴合以第二章中的动力学模型为基础的优化轨迹算法得到的参考速度分布以及参考轨迹位置的跟踪仿真。此次仅以生成速度的 90%作为参考速度，再考虑到优化轨迹的过弯过程距离外边界较近，上述控制算法难以实现准确控制，因此仅使用赛道的中线作为待跟踪的轨迹。

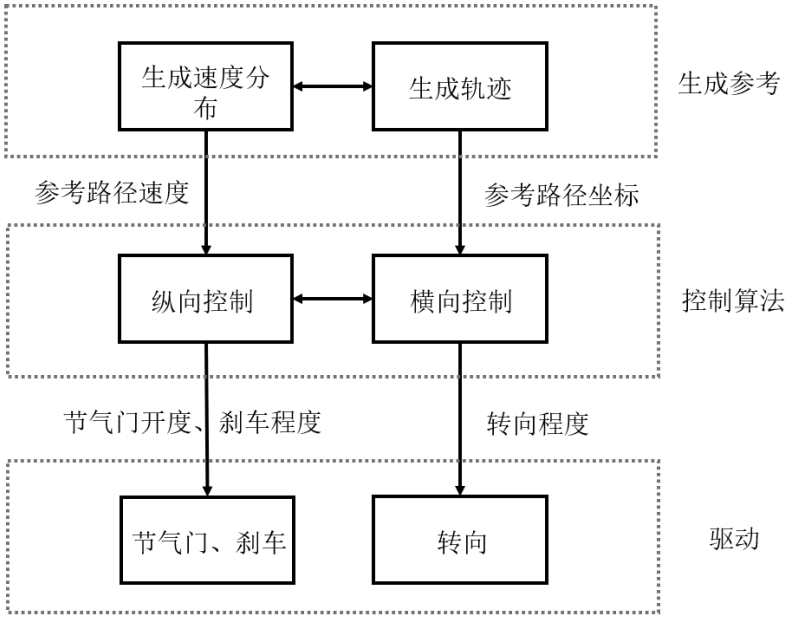


图 5-10 仿真车辆控制方法基本构架

5.3.1 定义参考路径及参考速度

进行轨迹跟踪控制的基础是得知参考轨迹的位置坐标以及每一位置坐标的参考速度，因此在运行 CARLA 模拟器的过程中应能够读取上述的参考轨迹信息。为此，将期望路径上样本点的信息储存到文本文件中，使得文本文件每行都储存着一个样本点的横坐标、纵坐标以及速度。之后在 Python 脚本中将文本文件转换为 N×3 的矩阵，其中 N 由样本点的个数决定，通过获取矩阵某一位置中的内容得到参考路径的信息，以此为基础进行导航。

5.3.2 仿真过程

在 CARLA 中对车辆进行转向控制时需输入转向角，其可识别的范围是[-1, 1]，因此需要对控制算法计算出的弧度值与最大转向角进行除法运算，以得到可传输给 CARLA 的数值。为了实现纵向控制，需要调节节气门开度以及刹车的程度，这两个参数可接受的范围都为[0, 1]。

由于车辆跟踪的效果与参考轨迹样本点密度有关，因此在仿真的过程中将样本点进行插值处理，同时引入哈希表以便后续进行索引。但是经过插值又会出现计算速度变慢的问题，于是在实际控制过程中会以距离车辆参考点最近的参考路径样本点为参照，选取其前方 20 米以内的部分样本点以及后方一个样本点作为样本点子集导航车辆。仿真程序的总体逻辑如图 5-11 所示，在 CARLA 客户端中运行的主函数见附录 C。

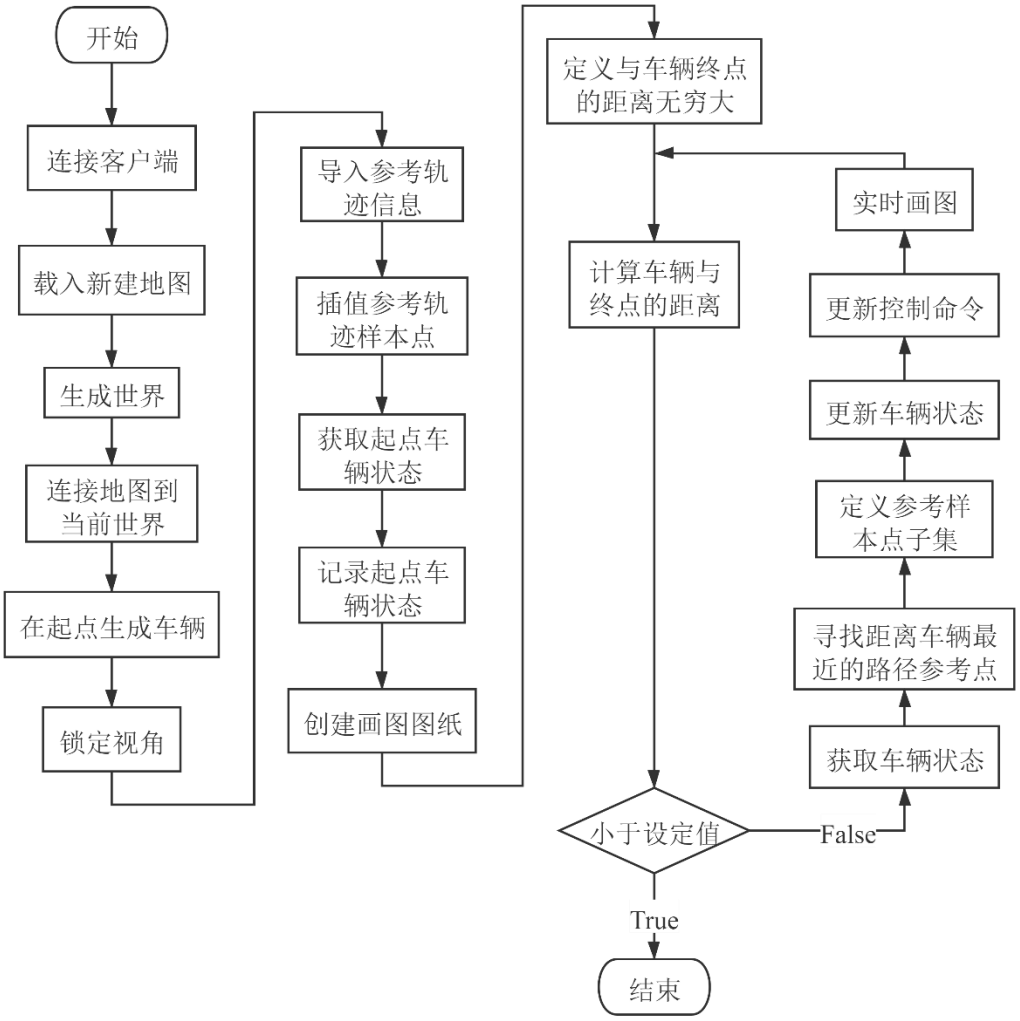


图 5-11 控制程序流程图

5.3.3 仿真结果及分析

在应用 PurePursuit 方法进行横向控制时，车辆并不能在赛道中一圈，而应用 Stanley 方法进行横向控制方法时车辆的跟踪比较准确，以下将列出结合 Stanley 方法横向控制和 PID 方法纵向控制的仿真运行结果。车辆在模拟器中运行的部分过程如图 5-12 所示，其中以黄色的圆圈表示参考路径上的所有样本点，以绿色圆圈标记车辆重心，以蓝色的叉号标记车辆的参考点，以红色的圆圈标记算法选取到的参考路径上的样本点。运行一周的结果如图 5-14 至图 5-18 所示，其中图 5-14 所示的轨迹跟踪曲线中黄色的线为实际运行的轨迹，绿色的线为参考轨迹；图 5-15 所示的速度跟踪曲线中蓝色为实际车辆的速度分布，黄色的线为参考速度分布；图 5-16 至图 5-18 为运行过程中节气门开度、刹车程度以及转向角的数值。

图 5-12 车辆控制仿真俯视画面

由图 5-14 所示的轨迹跟踪结果中可以发现，轨迹跟踪在直道部分的效果较好，而轨迹不重合的的位置多发生在弯道处，且转角越小重合的程度越高，这说明此控制器更适用于较小转向角的情况。同时考虑到此次的参考车速很高，因此在尝试以更低车速运行一周，其结果如图 5-13 所示，可以发现两条曲线几乎完全重合，这也说明过高的车速会影响控制的效果，因此为了能够实现高速情况下的车辆横向控制，需要更精准的控制算法，甚至需要对极限情况在的车辆动力学进行建模。

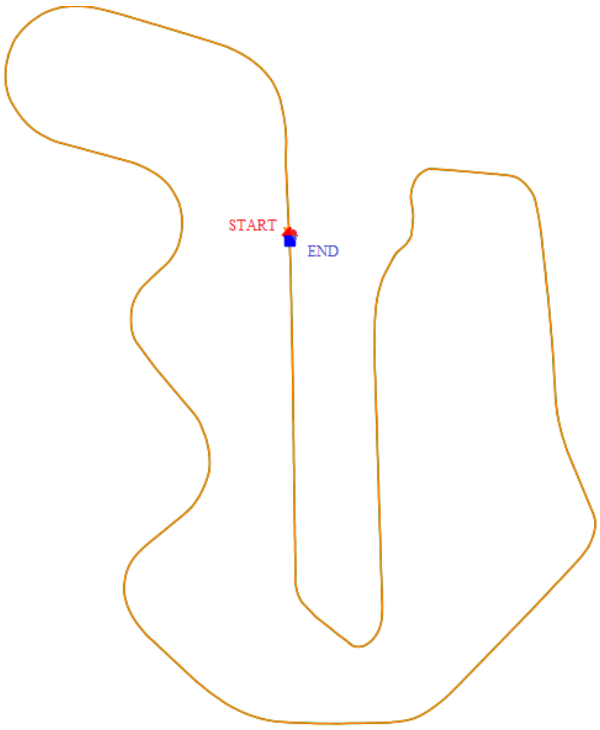


图 5-13 低速轨迹跟踪结果

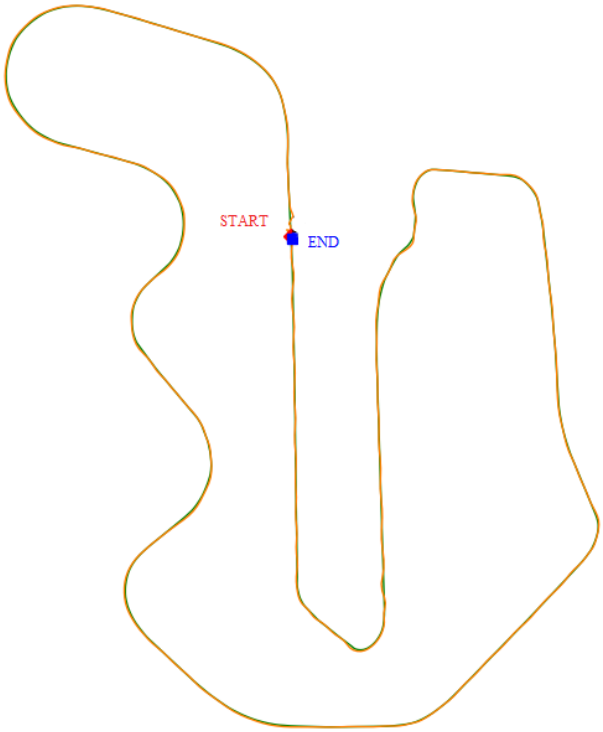


图 5-14 高速轨迹跟踪结果

从图 5-15 所示的速度跟踪结果图中可以发现，在赛道中间部分纵向控制算法的跟踪效果最好，而在运行的开始阶段实际速度与参考速度之间有较大差距。对照赛道的结构可以发现，开始阶段有较多弯道而跟踪效果最好的部分弯道较少，同时注意到轨迹生成算法得到的速度分布在过弯过程也会出现变化，因此开始导致阶段的速度变化较剧烈，使得跟踪的效果变差，这也说明 PID 控制算法更适用于速度波动较不频繁的情况。

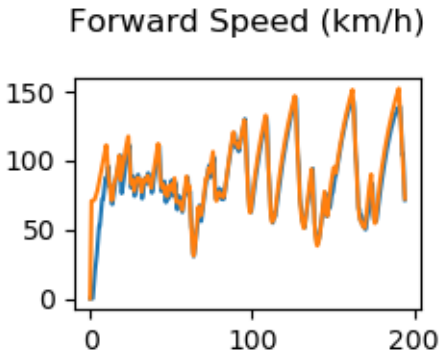


图 5-15 高速运行速度跟踪结果

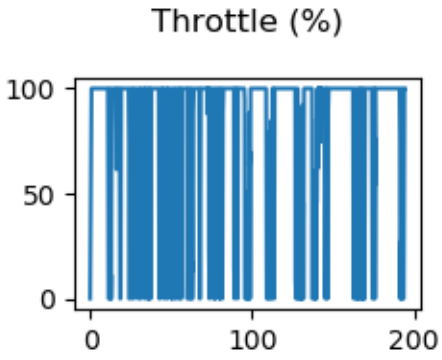


图 5-16 高速运行过程的节气门开度

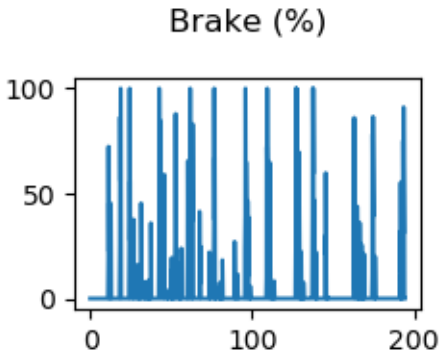


图 5-17 高速运行过程的刹车程度

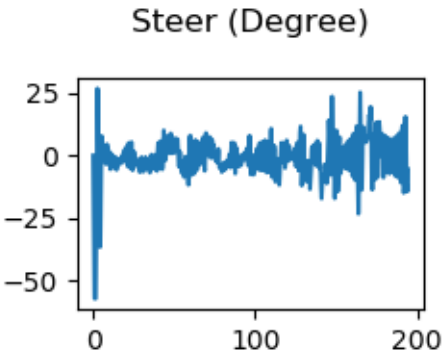


图 5-18 高速运行过程的转向角度

5.4 本章小结

本章对 PurePursuit 和 Stanley 横向控制算法以及 PID 纵向控制算法进行了介绍，并以此为基础进行了 CARLA 模拟器客户端的控制程序编写，最终通过 Stanley 横向控制算法以及 PID 纵向控制算法进行轨迹跟踪控制，使车辆在 CARLA 模拟器中沿雷山赛道中线以优化轨迹算法生成速度分布的 90%进行了跟踪控制，最后根据结果图对控制算法的适应范围和局限性进行了分析。

总 结 与 展 望

本文主要结合 CARLA 模拟器进行了赛车快速轨迹生成算法的编写，并得到了优化轨迹的分布，同时也对在 CARLA 中搭建地图以及进行车辆控制进行了初步探索。主要完成了以下几项工作：

1）以单轨车辆模型和 Fiala 轮胎模型为基础，建立了优化轨迹生成算法中的车辆动力学模型，以分步计算速度分布和优化轨迹分布的方式分解车辆行驶过程中横向与纵向之间的耦合关系。其中，速度分布以依次增加约束的方式获得，而在生成轨迹的过程中，应用凸优化方法将提高圈速这一多目标函数转化为凸优化问题，进而提高了计算速度。

2）以 OpenStreetMap 地图文件为基础，经过格式转换及 RoadRunner 软件的修剪最终在 CARLA 中实现了雷山赛道地图的搭建，在这一新建地图的基础上通过 Python 脚本获取到赛道的参数信息，并以此作为输入进行优化轨迹算法的计算，生成了优化轨迹分布以及速度分布。

3）以 Stanley 横向控制、PurePursuit 横向控制和 PID 纵向控制理论为基础编写控制算法进行轨迹及速度跟踪，在对不同控制方法的控制效果进行比较后，最终使用结合 Stanley 横向控制和 PID 纵向控制的方式，在 CARLA 模拟器中实现了优化轨迹算法生成的纵向车速的 90%的跟踪控制。

这些在无人驾驶赛车路径规划方面有可借鉴的部分，同时其提供的搭建地图以及基于模拟器获取道路信息的方法，为以现实世界的道路情况为基础进行数据搜集、算法开发以及车辆驾驶仿真提供了可行的方式。

但是目前阶段还有很多不足，由于还没实现对 CARLA 模拟器中车辆设定的动力驱动模型以及动力学模型进行修改和定义，会使得仿真车辆无法模拟实车情况下真实的效果。因此下一阶段可联合 CarSim、UE4 编辑器进行车辆动力学的定义和修改。同时为了实现优化轨迹的跟踪还需要编写以车辆动力学为基础且更精准的控制算法，或者联合 Autoware 自动驾驶软件，通过连接物理方向盘在道路上运行一周的方式用激光

雷达采集更精确的道路信息，并在此基础上修改或重写其中的控制算法。在这两方面得到改进后，便能对轨迹生成进行有效的仿真验证，同时也为后续在这一赛道及车辆模型下进行相关的开发配置了基础环境。

参 考 文 献

[1] 冀同涛. 无人驾驶电动赛车路径规划与跟踪控制研究[D]. [辽宁工业大学硕士学位论文]. 锦州: 辽宁工业大学, 2020.

[2] Paden B, Cap M, Yong S Z, et al. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2016, 1(1): 33-55.

[3] Gonzalez D，Perez J，Milanes V，et al. A Review of Motion Planning Techniques for Automated Vehicles[J]. 17(4): 1-11. IEEE Transactions on Intelligent Transportation Systems, 2015.

[4] 吕吉冬. 考虑车辆动力学的智能车局部路径规划方法研究[D]. [吉林大学硕士学位论文]. 长春: 吉林大学, 2020.

[5] KAMMEL S, ZIEGLER J, PITZER B，et al. Team AnnieWAY\″s Autonomous System for The 2007 DARPA Urban Challenge[J]. Journal of Field Robotics, 2008, 25(9): 615-639．

[6] 杜明博, 梅涛, 陈佳佳, 赵盼, 梁华为, 黄如林, 陶翔. 复杂环境下基于 RRT 的智能车辆运动规划算[J]. 机器人, 2015, 37(04): 443-450.

[7] LaValle S M, Kuffner J J. Rapidly-Exploring Random Trees: Progress and prospects[C]//4th International Workshop on Algorithmic Foundations of Robotics. Wellesley, USA: A K Peters, 2000: 293-308.

[8] Kuffner J J, LaValle S M. RRT Connect: An Efficient Approach to Single-Query Path Planning[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2000: 995-1001.

[9] Elbanhawi M, Simic M, Jazar R. Continuous-Curvature Bounded Trajectory Planning Using Parametric Splines[M]//Frontiers in Artificial Intelligence and Applications, vo1.262. Amsterdam, Netherlands: IOS Press, 2014: 513-522.

[10] 周维, 过学迅, 裴晓飞, 张震, 余嘉星. 基于 RRT 与 MPC 的智能车辆路径规划与跟踪控制研究[J]. 汽车工程, 2020, 42(09): 1151-1158.

[11] 耿以才. 基于动态人工势场法无人驾驶汽车路径规划研究[D]. [上海工程技术大学硕士学位论文]. 上海: 上海工程技术大学, 2016.

[12] 郭袅鹏. 基于改进人工势场法的路径规划算法研究[D].[哈尔滨工业大学硕士学位论文]. 哈尔滨: 哈尔滨工业大学，2017.

[13] Ren J, McIsaac K A, Patel R V. Modified Newton's Method Applied to Potential Field-Based Navigation for Mobile Robots[J]. IEEE Transactions on Robotics, 2006 22(2): 384-391.

[14] Xu L，Yin G，Li G，et al. Stable Trajectory Planning and Energy-Efficience Control Allocation of Lane Change Maneuver for Autonomous Electric Vehicle[J]. Journal of Intelligent and Connected Vehicles, 2018.

[15] Wang Z，Li G，Jiang H，et al. Collision-Free Navigation of Autonomous Vehicles Using Convex Quadratic Programming-Based Model Predictive Control[J]. IEEE/ASME Transactions on Mechatronics, 2018.

[16] S. Garlicka, A. Bradleyb*. Real-Time Optimal Trajectory Planning for Autonomous Vehicles and Lap Time Simulation Using Machine Learning.

[17] 黄梅. 基于模型预测控制的智能车辆轨迹规划及跟踪控制[D]. [重庆邮电大学硕士学位论文]. 重庆: 重庆邮电大学, 2020.

[18] Duan J, Yao J, Liu D, et al. A Path Tracking Control Algorithm with Speed Adjustment for Intelligent Vehicle[C]. IEEE International Conference on Robotics and Biomimetics (ROBIO). Shenzhen: IEEE, 2013: 2397-2402.

[19] Gaining H, Weiping F, Wen W, et al. The Lateral Tracking Control for The Intelligent Vehicle Based on Adaptive PID Neural Network[J]. Sensors, 2017, 17(6): 1244-1259.

[20] Lie G, Ping S G, Ming Y, et al. Lane Changing Trajectory Planning and Tracking Controller Design for Intelligent Vehicle Running on Curved Road[J]. Mathematical Problems in Engineering, 2014, 2014: 1-9.

[21] Bingbing T, Haiying Z, Sishan W. Design of LQR Trajectory Tracking Controller for Autonomous Vehicle[J]. Journal of Hubei University of Automotive Technology, 2017, 31(04): 1-11.

[22] Kun Z, Shengmin C, Wang J F. Path Tracking Control of Intelligent Vehicle Based on Fuzzy Neural Network[J]. Automotive Engineer, 2015, 37(1): 38-42.

[23] 朱盈璇, 赵克刚, 叶杰, 梁政焘. 基于配点法的智能车实时轨迹跟踪研究. 机床与液压. 2020, 48(11): 150-154.

[24] Falcone P, Borrelli F, Asgari J, et al. Predictive Active Steering Control for Autonomous Vehicle Systems[J]. IEEE Transactions on Control Systems Technology, 2007, 15(3): 566-580.

[25] Falcone P, Eric T H, Borrelfi F, et al. MPC-Based Yaw and Lateral Stabilisation via Active Front Steering and Braking[J]. Vehicle System Dynamics, 2008, 46(1): 611-628.

[26] Attia R, Orjuela R, Basset M. Combined Longitudinal and Lateral Control for Automated Vehicle Guidance[J]. Vehicle System Dynamics, 2014, 52(2): 261-279.

[27] Gutjahr B, Groll L, Werling M. Constrained Linear Time-Varying Lateral Vehicle MPC[J]. IEEE Trajectory Optimization Using Transactions on Intelligent Transportation Systems, 2016, 18(6): 1586-1595.

[28] 龚建伟，姜岩，徐威. 无人驾驶车辆模型预测控制[M]. 北京: 北京理工大学出版社, 2014.

[29] 胡寿松, 自动控制原理[M]，北京:国防工业出版社，1994: 80-128.

[30] 游峰. 智能车辆自动换道与自动超车控制方法的研究[D]. [吉林大学博士学位论文]. 长春: 吉林大学, 2005.
[31] 李子龙. 基于人工神经网络的智能汽车循迹控制研究[D]. [合肥工业大学硕士学位论文]. 合肥: 合肥工业大学, 2019.
[32] Smith D E, Starkey J M. Effects of Model Complexity on The Performance of Automated Vehicle Steering Controllers: Model Development, Validation and Comparison[J]. Vehicle System Dynamics, 1995, 24(2): 163－181.
[33] Kapania N R, Subosits J, Christian Gerades J. A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories[J]. Journal of Dynamic Systems, Measurement, and Control. 2016, 138(9).
[34] John K Subosits, John Christian Gerdes. Autonomous Vehicle Control for Emergency Maneuvers: The Effect of Topography. In American Control Conference (ACC), 2015, 1405–1410.
[35] Cattaruzza M. Design and Simulation of Autonomous Driving Algorithms[D]. Politecnico di Torino, 2019.
[36] Santonato S F. A Complete End-to-End Simulation Flow for Autonomous Driving Frameworks[D]. Politecnico di Torino, 2020.
[37] Jeffrey S. Wit. Vector Pursuit Path Tracking for Autonomous Ground Vehicles[D]. PHD Thesis, University of Florida, 2000.
[38] Shin, D.H. High Performance Tracking of Bxplicit Path by Roadworthy Mobile Robots[D]. Doctoral Thesis, Dept of Civil Engineering, CMU, May, 1990.

致　谢

这次毕设题目对我来说确实并不容易，几乎所有的东西都要从头学起，中途有过抱怨想过放弃，但是好在老师、同学的鼓励和帮助让我挺了下来，虽然做到现在的程度还有很多不足，但是我还是要祝贺自己坚持到了最后。

本次我的论文能够如期完成，首先要感谢王刚老师在整个毕设期间的付出，感谢王老师仔细阅读我的每一次报告，告诉我内容和格式上不正确的地方，提出明确的问题并给出改进的方向，为我能够写出规范且有质量的论文打下了基础。还要感谢学校和老师们为我们争取到去中汽研进行毕业设计的机会，让我们有机会体验企业生活与工作，这里的工作氛围让我深刻体会到了严谨的思维以及坚定的信念对生活以及工作的重要性，这些也正是我需要努力完善之处。

更要感谢我的企业导师许晟杰，许老师在毕业设计正式开始之前就给了我很多有用的学习资料，还细心地帮我整理出了可能用到的网站、软件以及各种高效的检索方式。开学后，许老师也在努力想办法让能够我们尽快来到中汽研实习。在中汽研实习期间，许老师更是毫无保留地将自己知道的东西教给我，总是耐心地回答我不懂的地方，会培养我借助可用资源自己解决问题的能力，也会重视我的想法，愿意陪我一起尝试新的方法。在我毕设卡在一处没有进展时，许老师会和我一起努力寻找解决的办法，还不断鼓励我要敢于面对挑战，相信自己。在和许老师一起学习的过程中，对我影响最大的还是他遇到问题时敢于面对，沉着冷静的魄力，还有最难得的积极乐观的心态。很幸运在做毕设期间能有机会认识这样一个温柔又优秀的老师，也祝您正在担心的事有一个好的结果吧！最后，还要感谢陪伴我度过大学时光的李卓飞同学，在我遇到困难的时候竭尽全力地帮我，在我焦虑难过的时候帮我缓解压力，在毕设的过程中也帮我解决了 RoadRunner 软件的大问题，这也是我的毕设能够进行下去的关键。

在河北工业大学的四年，很幸运能够遇见五个贴心的舍友，一群热心友善的同学，每个任真负责的老师，还有一次次难忘的经历，这其中有的曾让我激动不已，也不免有痛苦迷茫。还好，走到现在，这种种的如愿和遗憾早已经与我的大学时光融在一起，每一道印记都有重要的意义，每一次成长都算数，还要继续赶路，那就再见吧，祝你沿途好风景！

附 录

附录 A:优化轨迹生成算法

```
#轮胎模型
import numpy as np


def fiala(C, muP, muS, alpha, Fz):
    Fy = np.zeros(alpha.size)
    for i in range(alpha.size):
        if np.abs(alpha[i]) < alphaSlide:
            Fy[i] = -C * np.tan(alpha[i]) + C**2 / (3 * muP * Fz) * (2 - muS / muP) * np.tan(alpha[i]) * np.abs(np.tan(alpha[i])) - C**3/(9*muP**2*Fz**2)*np.tan(alpha[i])**3*(1-(2*muS)/(3*muP))
        else :
            Fy[i] = -muS * Fz * np.sign(alpha[i])

    return Fy
```

```
#车辆参数
import numpy as np
import tiremodels as tm


class Vehicle:
    def __init__(self, vehicleName = "shelley"):

        if vehicleName is "shelley":

            self.a = 1.0441
            self.b = 1.4248
            self.d = 1.50
            self.rW = 0.34
            self.m = 1512.4
            self.Cf = 160000.0
            self.Cr = 180000.0
            self.Iz  = 2.25E3
```

```
            self.muF = 0.97
            self.muR = 1.02
            self.g = 9.81
            self.L = self.a + self.b
            self.FzF = self.m*self.b*self.g/self.L
            self.FzR = self.m*self.a*self.g/self.L
            self.h = 0.55
            self.brakeTimeDelay = 0.25
            self.rollResistance = 255.0
            self.powerLimit = 160000.0
            self.dragCoeff = 0.3638
            self.deltaLim = 27. * np.pi / 180
            self.beta = 2.0
            self.brakeFactor = 0.95
        else:
            sys.exit("invalid vehicle specified")
```

```
#输入赛道信息
import numpy as np
from numpy import genfromtxt
import scipy.io as sio
from scipy.integrate import odeint
from scipy import interpolate


class Path:
    def __init__(self):
        self.roadIC = np.zeros((3,1))
        self.posE = [0]
        self.posN = [0]
        self.roadPsi = [0]
        self.curvature = [0]
        self.s = [0]
        self.isOpen = True
        self.roadIC =np.zeros((3,1))
        self.refPointName = None
        self.friction = None
        self.vMax = None


    def loadFromCSV(self, pathName):
        x = genfromtxt(pathName, delimiter =",")
        self.s = np.array(x[:,0])
```

```
        self.curvature = np.array(x[:,1])
        self.posE = np.array(x[:,2])
        self.posN = np.array(x[:,3])
        self.roadPsi = np.array(x[:,4])
        self.roadIC = np.array(x[0:3,5])


    def toDict(self):
        out = {}
        out["s"] = self.s
        out["K"] = self.curvature
        out["posE"] = self.posE
        out["posN"] = self.posN
        out["roadPsi"] = self.roadPsi

        return out
```

```
#车辆动力学模型
import numpy as np
import tiremodels as tm
import scipy
import pdb


def getAllSys(veh, Ux, K, ts):
    A = []
    B = []
    D = []
    N = len(Ux)

    aF = np.zeros((N,1))
    aR = np.zeros((N,1))

    numTableValues = 250

    alphaFslide = np.abs(np.arctan(3*veh.muF*veh.m*veh.b/veh.L*veh.g/veh.Cf))
    alphaRslide = np.abs(np.arctan(3*veh.muR*veh.m*veh.a/veh.L*veh.g/veh.Cr))

    alphaFtable = np.linspace(-alphaFslide, alphaFslide, numTableValues)
```

```
    alphaRtable = np.linspace(-alphaRslide, alphaRslide, numTableValues
)

    FyFtable = tm.fiala(veh.Cf, veh.muF, veh.muF, alphaFtable, veh.FzF)

    FyRtable = tm.fiala(veh.Cr, veh.muR, veh.muR, alphaRtable, veh.FzR)


    alphaFtable = np.flip(alphaFtable, 0)
    alphaRtable = np.flip(alphaRtable, 0)
    FyFtable = np.flip(FyFtable, 0)
    FyRtable = np.flip(FyRtable, 0)

    for i in range(N):

        FyFdes = veh.b / veh.L * veh.m * Ux[i]**2 * K[i]
        FyRdes = veh.a / veh.L * veh.m * Ux[i]**2 * K[i]

        aF[i] = _force2alpha(FyFtable, alphaFtable, FyFdes)
        aR[i] = _force2alpha(FyRtable, alphaRtable , FyRdes)

        a, b, d = getAffineModel(Ux[i], K[i], ts[i], veh, aF[i], aR[i])


        A.append(a)
        B.append(b)
        D.append(d)

    return A, B, D


def getAffineModel(Ux, K, ts, veh, aFhat, aRhat):

    a = veh.a
    b = veh.b
    m = veh.m
    Cf = veh.Cf
    Cr = veh.Cr
    g = veh.g
    L = a + b
    Iz = veh.Iz
    FzF = veh.FzF
    FzR = veh.FzR
    muF = veh.muF
```

```
    muR = veh.muR


    FyFhat = tm.fiala(Cf, muF, muF, aFhat, FzF)
    FyRhat = tm.fiala(Cr, muR, muR, aRhat, FzR)
    Cf = getLocalStiffness(aFhat, Cf, muF, muF, FzF)
    Cr = getLocalStiffness(aRhat, Cr, muR, muR, FzR)

    Ac = np.array([ [0,  Ux,  0,  Ux] , [0, 0, 1, 0] ,
        [0, 0, (-a**2*Cf - b**2*Cr)/(Ux*Iz), (b*Cr - a*Cf)/Iz] ,
        [0,  0,(b*Cr - a*Cf)/(m*Ux**2)-1   , -(Cf+Cr)/(m*Ux)] ])

    Bc = [[0], [0], [a*Cf/Iz], [Cf/(m*Ux)]]

    dc = [[0],
         [-K*Ux],
         [(a*Cf*aFhat - b*Cr*aRhat)/Iz + (a*FyFhat-b*FyRhat)/Iz ],
         [(Cf*aFhat + Cr*aRhat)/(m*Ux) + (FyFhat + FyRhat)/(m*Ux)]]

    A, B1 = myc2d(Ac, np.concatenate((Bc, dc), axis = 1), ts)
    B = B1[:,0]
    d = B1[:,1]

    return A, B, d


def _force2alpha(forceTable, alphaTable, Fdes):
        if Fdes > max(forceTable):
             Fdes = max(forceTable) - 1

        elif Fdes < min(forceTable):
             Fdes = min(forceTable) + 1

        alpha = np.interp(Fdes, forceTable ,alphaTable)

        return alpha


def myc2d(A, B, ts):
    m, n = A.shape
    _, nb = B.shape

    conc1 = ts * np.concatenate((A,B), axis = 1)
    conc2 = np.zeros((nb, n+nb))
```

```
    conc  = np.concatenate( (conc1, conc2), axis = 0)

    s = scipy.linalg.expm(conc)
    Ad = s[0:n, 0:n]
    Bd = s[0:n, n:n+nb]

    return Ad, Bd


def getLocalStiffness(alpha, C, muP, muS, Fz):
    delta = 1e-4
    alpha2 = alpha + delta
    alpha1 = alpha - delta
    Fy1 = tm.fiala(C, muP, muS, alpha1, Fz)
    Fy2 = tm.fiala(C, muP, muS, alpha2, Fz)

    Cbar = (Fy2 - Fy1) / (alpha2 - alpha1)
    Cbar = abs(Cbar)

    return Cbar
```

```
#生成速度分布
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
from numpy import cos
from numpy import sin
import pdb


class RecordedProfile():
    def __init__(self, matFileName):
        mfile = sio.loadmat(matFileName)
        self.s =  mfile["s"]
        self.Ux = mfile["Ux"]
        try:
            self.Ax = mfile["Ax"]
        except:
            self.Ax = np.zeros(self.s.shape)


class BasicProfile():
```

```
22.     def __init__(self, vehicle, path, friction = 0.3, vMax = 10., AxMax
    = 9.81):
23.         self.vehicle = vehicle
24.         self.path = path
25.         self.s = path.s
26.         self.Ux = np.zeros(self.s.shape)
27.         self.Ax = np.zeros(self.s.shape)
28.
29.         if isinstance(vMax, np.ndarray):
30.             self.vMax = vMax
31.         else:
32.             self.vMax = vMax * np.ones(self.s.shape)
33.
34.         if isinstance(friction, np.ndarray):
35.             self.mu = friction
36.         else:
37.             self.mu = friction * np.ones(self.s.shape)
38.
39.         if path.isOpen:
40.             self.generateBasicProfileOpen(AxMax)
41.
42.         else:
43.             self.generateBasicProfileClosed(AxMax)
44.
45.
46.     def toDict(self):
47.         out = {}
48.         out["s"] = self.s
49.         out["Ux"] = self.Ux
50.         out["Ax"] = self.Ax
51.
52.         return out
53.
54.     def generateBasicProfileClosed(self, AxMax):
55.         g = 9.81
56.         K = self.path.curvature
57.         s = self.s
58.         AyMax = self.mu* g
59.         AxMax = min( np.append(self.mu * g, abs(AxMax)))
60.
61.         UxSS = np.sqrt ( np.divide(AyMax, np.abs(K + 1e-8) ) )
62.         minUx = np.amin(UxSS)
63.         maxUx = self.vMax
64.         idx = np.argmin(UxSS)
```

```
65.
66.         inds = np.arange(len(UxSS))
67.         shiftedInds = np.roll(inds, -idx)
68.         kShifted = K[shiftedInds]
69.         maxUxShifted = maxUx[shiftedInds]
70.         AyMaxShifted = AyMax[shiftedInds]
71.
72.         UxShift, AxShift = self.genSpeed(kShifted, minUx, maxUxShifted,
    AxMax, AyMaxShifted)
73.
74.         self.Ux = np.roll(UxShift, idx)
75.         self.Ax = np.roll(AxShift, idx)
76.
77.         return
78.
79.
80.     def generateBasicProfileOpen(self, AxMax):
81.         g = 9.81
82.         K = self.path.curvature
83.         AyMax = self.mu* g
84.         AxMax = min( np.append(self.mu * g, abs(AxMax)))
85.         self.Ux, self.Ax = self.genSpeed(K, 0, self.vMax, AxMax, AyMax)

86.
87.     def genSpeed(self, K, minUx, maxUx, AxMax, AyMax):
88.         g = 9.81
89.         s = self.s
90.
91.         numSteps = s.size
92.
93.         UxInit1 = np.zeros(numSteps)
94.         UxInit2 = np.zeros(numSteps)
95.         UxInit2[0]  = minUx
96.         UxInit3 = np.zeros(numSteps)
97.         UxInit3[-1] = minUx
98.
99.         ax = np.zeros(numSteps)
100.         ay = np.zeros(numSteps)
101.
102.         UxInit1 = np.sqrt ( np.divide(AyMax, np.abs(K + 1e-8) ) )
103.
104.         for i in range(UxInit2.size-1):
105.             temp = np.sqrt( UxInit2[i]**2 + 2*AxMax*(s[i+1] - s[i]))

```

```
106.
107.             if temp > maxUx[i]:
108.                 temp = maxUx[i]
109.
110.             if temp > UxInit1[i+1]:
111.                 temp = UxInit1[i+1]
112.
113.             UxInit2[i+1] = temp
114.
115.         for i in reversed(range(1,UxInit3.size)):
116.             temp = np.sqrt( UxInit3[i]**2 + 2* AxMax * (s[i] - s[i-1])
    )
117.
118.             if temp > UxInit2[i-1]:
119.                 temp = UxInit2[i-1]
120.
121.             UxInit3[i-1] = temp
122.
123.         ax = np.divide( (np.roll(UxInit3,1)**2 - UxInit3**2) , (2 * (n
   p.roll(s,1) - s) ) )
124.         ax[0] = ax[1]
125.
126.         return UxInit3, ax
127.
128.
129.
130.def getLapTime(s, Ux):
131.
132.     ts = np.concatenate(( s[1, np.newaxis], np.diff(s) )) / Ux
133.     lapTime = np.sum(ts)
134.
135.     return lapTime, ts
```

```
1. #生成优化轨迹
2. import numpy as np
3. import cvxpy as cp
4. import utils
5. from velocityprofiles import *
6. from simulation import *
7. from control import *
8. from paths import *
9. import sys
10.from scipy import interpolate
```

```
import pdb


class OptimizationResults:
    def __init__(self, vpDict, logFile, optDict, pathDict, lapTime):
        self.vps = [vpDict]
        self.logFiles = [logFile]
        self.opt = [optDict]
        self.path = [pathDict]
        self.lapTimes= [lapTime]

    def append(self, vpDict, logFile, optDict, pathDict, lapTime):
        self.vps.append(vpDict)
        self.logFiles.append(logFile)
        self.opt.append(optDict)
        self.path.append(pathDict)
        self.lapTimes.append(lapTime)


class RapidPathGeneration:
    def __init__(self, vehicle, path, bounds, mu, NUM_ITERS = 5, buff =
    None):

        self.NUM_ITERS = NUM_ITERS
        self.NUM_POINTS = 1863
        self.lam1 = 1.0
        self.lam2 = np.zeros((self.NUM_POINTS, 1))

        self.vehicle = vehicle
        self.initialPath = path
        self.path = path
        self.bounds = bounds
        self.mu = mu

        if buff is None:
            self.buff = ([0, path.s[-1]], [0, 0])
        else:
            self.buff = buff

        self.widthLeft, self.widthRight  = self.getLaneWidth()
        self.vp = RacingProfile(vehicle, path, self.mu, vMax = 99)
        self.controller = LaneKeepingController(path, vehicle, self.vp)

```

```
52.         bikeSim = Simulation(vehicle, self.controller, path = self.path
    , profile = self.vp, mapMatchType = "closest")
53.         logFile0 = bikeSim.simulate()
54.         lapTime = bikeSim.getLapTime()
55.
56.         self.optResults = OptimizationResults(self.vp.toDict(), logFile
    0, None, self.initialPath.toDict(), lapTime)
57.
58.
59.     def optimize(self):
60.         print("Optimizing Path")
61.
62.         for i in range(self.NUM_ITERS):
63.
64.             opt = self.getRapidTrajectory()
65.             self.updateWorld(opt)
66.
67.             self.vp = RacingProfile(self.vehicle, self.path, self.mu)
68.
69.             bikeSim = Simulation(self.vehicle, self.controller, path =
    self.path, profile = self.vp, mapMatchType = "closest")
70.
71.             logFile = bikeSim.simulate()
72.             lapTime = bikeSim.getLapTime()
73.
74.             self.optResults.append(self.vp.toDict(), logFile, opt, self
    .path.toDict(), lapTime)
75.
76.         return self.optResults
77.
78.
79.     def getRapidTrajectory(self):
80.
81.         ds = self.path.s[-1] / (self.NUM_POINTS)
82.
83.         self.path.resample(ds)
84.         self.vp.resample(ds)
85.         self.resampleLaneWidth()
86.
87.         opt = self.getCurvatureProfile()
88.
89.         return opt
90.
91.
```

```
92.     def resampleLaneWidth(self):
93.
94.         n = len(self.path.s)
95.         N = len(self.widthLeft)
96.
97.         ind = np.round(np.linspace(0, N-1, n))
98.         ind = ind.astype(int)
99.
100.         self.widthLeft = self.widthLeft[ind]
101.         self.widthRight= self.widthRight[ind]
102.
103.         return
104.
105.
106.     def getCurvatureProfile(self):
107.
108.         n = len(self.path.s)
109.         assert n == self.NUM_POINTS
110.         s = self.path.s
111.         K = self.path.curvature
112.         Ux = self.vp.Ux
113.         psiR = self.path.roadPsi
114.         lam1 = self.lam1
115.         lam2 = self.lam2
116.
117.
118.         offset = np.linalg.norm( [self.path.posE[0] - self.path.posE[-
    1], self.path.posN[0] - self.path.posN[-1]] )
119.         ds = np.diff(s)
120.
121.         _, ts = getLapTime(self.vp.s, self.vp.Ux)
122.
123.         print('Generating Affine Tire Models ...')
124.         A, B, d = utils.getAllSys(self.vehicle, Ux, K, ts)
125.
126.         print('Solving Convex Problem ...')
127.
128.         delta = cp.Variable(n)
129.         x     = cp.Variable((4,n))
130.
131.         Kmat = np.diag(-K[0:n-1])
132.
133.         objective = cp.Minimize(cp.norm( 1/ds*(psiR[1:n] - psiR[0:n-1]
    + x[1,1:n].T - x[1,0:n-1].T), 2) +
```

```
134.         lam1*cp.norm(delta[1:n]-delta[0:n-1]) +
135.         cp.sum(  lam2[0:n-1]*(ds/Ux[0:n-1]* ( Kmat*x[0,0:n-1].T  )   +
136.         ds/Ux[0:n-1]*cp.square( x[3,0:n-1] + x[1,0:n-1]   ).T) ) )
137.
138.         constraints = []
139.
140.         for i in range(n-1):
141.             constraints += [ x[:,i+1] == A[i]*x[:,i] + B[i]*delta[i] +
     d[i] ]
142.
143.         constraints += [x[0,:] <= self.widthLeft.squeeze()]
144.         constraints += [x[0,:] >= self.widthRight.squeeze()]
145.         constraints += [x[0,n-1] == x[0,0]                          -
    offset]
146.         constraints += [x[1,n-1] == x[1,0]
         ]
147.         constraints += [x[2,n-1] == x[2,0]
         ]
148.         constraints += [x[3,n-1] == x[3,0]
         ]
149.
150.         constraints += [(x[0,n-1] - x[0,n-2])/ds[-1] == (x[0,1] - x[0,
    0])/ds[0] ]
151.
152.         prob = cp.Problem(objective, constraints)
153.         res =prob.solve()
154.
155.         x = x.value
156.         opt = {}
157.         opt["feasible"] = res != np.inf
158.         if not opt["feasible"]:
159.             sys.exit('No feasible solution found')
160.         opt["aF"] = x[3,:] + self.vehicle.a*x[2,:]/Ux - delta
161.         opt["aR"] = x[3,:] - self.vehicle.b*x[2,:]/Ux.T
162.         opt["e"] = x[0,:]
163.         opt["dPsi"] = x[1,:]
164.         opt["r"] = x[2,:]
165.         opt["roadPsi"] = opt["dPsi"] + psiR
166.         opt["beta"] = x[3,:].T
167.         opt["s"] = self.path.s
168.         opt["widthLeft"] = self.widthLeft
169.         opt["widthRight"] = self.widthRight
170.         opt["ts"] = ts
```

```
171.         opt["delta"] = delta
172.         opt["posE"], opt["posN"] = convertPathToGlobal(self.path, self
   .path.s, opt["e"])
173.         opt["K"] = 1/ds*np.diff(psiR + opt["dPsi"])
174.         opt["K"] = np.concatenate((opt["K"][0, np.newaxis], opt["K"]))
175.
176.         return opt
177.
178.
179.     def getLaneWidth(self):
180.         print('Getting Lane Boundaries')
181.
182.         path = self.path
183.         n = len(path.s)
184.         widthLeft = np.zeros((n,1))
185.         widthRight = np.zeros((n,1))
186.         buffS, buffVal = self.buff
187.
188.         buffer = np.interp(path.s, buffS, buffVal)
189.
190.         idxLeft = -1
191.         idxRight = -1
192.
193.         for i in range(n):
194.             point = [path.posE[i], path.posN[i]]
195.
196.             widthLeft[i], idxLeft = getMinDistance(point, self.bounds[
   "in"], idxLeft) - buffer[i]
197.
198.         for i in range(n):
199.             point = [path.posE[i], path.posN[i]]
200.             widthRight[i], idxRight = getMinDistance(point, self.bound
   s["out"], idxRight) - buffer[i]
201.
202.         self.widthLeft = widthLeft
203.         self.widthRight = -widthRight
204.
205.         return widthLeft, -widthRight
206.
207.
208.
209. def convertPathToGlobal(path, s, e):
210.
```

```
    n = len(s)
    E = np.zeros((n,1))
    N = np.zeros((n,1))

    centE = np.interp(s, path.s, path.posE)
    centN = np.interp(s, path.s, path.posN)
    theta = np.interp(s, path.s, path.roadPsi)

    for i in range(n):
        E[i] = centE[i] - e[i] * np.sin( np.pi / 2 - theta[i])
        N[i] = centN[i] - e[i] * np.cos( np.pi / 2 - theta[i])

    return E, N
```

```
#优化轨迹生成主函数
import sys
if '/content/Wolverine/utils' not in sys.path:
    sys.path.append('/content/Wolverine/utils')
    if '/content/Wolverine' not in sys.path:
        sys.path.append('/content/Wolverine')
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
params = {'legend.fontsize': 'x-large',
          'figure.figsize': (30, 20),
         'axes.labelsize': 'x-large',
         'axes.titlesize':'x-large',
         'xtick.labelsize':'x-large',
         'ytick.labelsize':'x-large'}
pylab.rcParams.update(params)
from vehicles import *
from paths import *
from pathgeneration import *
import scipy.io as sio
import numpy as np
!gdown --id 1fsKERl26TNTFIY25PhReoCujxwJvfyHn
import matplotlib as mpl
zhfont = mpl.font_manager.FontProperties(fname='SimHei .ttf',size=30)
zhfont1 = mpl.font_manager.FontProperties(fname='SimHei .ttf',size=20)


veh = Vehicle(vehicleName = "shelley")
track = Path()
```

```
track.loadFromMAT("/content/Wolverine/maps/THcenter.mat")
bounds = sio.loadmat('/content/Wolverine/maps/thunderhill_bounds_shifted.mat')
rpg = RapidPathGeneration(veh, track, bounds, mu = 0.9, NUM_ITERS = 3)

results = rpg.optimize()
vps = results.vps
opt = results.opt
logs = results.logFiles
paths = results.path
lapTimes = results.lapTimes
opt_e = opt[1]["e"]
opt_left = opt[1]["widthLeft"]
opt_right = opt[1]["widthRight"]
optPosE = opt[1]["posE"]
optPosN = opt[1]["posN"]
optS = opt[1]["s"]
optK = opt[1]["K"]

plt.figure(dpi=300)
A,=plt.plot(refPath["posE"], refPath["posN"], '-.', label = u"赛道中线", linewidth=2)
B,=plt.plot(optPath["posE"], optPath["posN"], label = u"优化轨迹", linewidth=2)
C,=plt.plot(bounds["in"][:,0], bounds["in"][:,1],'k', label = u"赛道边界", linewidth=1)
D,=plt.plot(bounds["out"][:,0], bounds["out"][:,1],'k', label = u"赛道边界", linewidth=1)
legend = plt.legend(handles=[A,B,C,D],prop=zhfont1)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.axis("equal")
plt.xlabel(u'全局纵坐标(m)',fontproperties=zhfont)
plt.ylabel(u"全局纵坐标(m)",fontproperties=zhfont)
plt.savefig('Trajectory.png', dpi=600)


plt.plot(vps[0]['s'] , vps[0]['Ux'] , linewidth = 4)
plt.xticks(fontsize=35)
plt.yticks(fontsize=35)
plt.xlabel(u'沿赛道距离 (m)',fontproperties=zhfont)
plt.ylabel(u"速度分布 (m/s)",fontproperties=zhfont)
plt.savefig('speed.png', dpi=600)
plt.show()
```

附录 B:赛道参数获取程序

```
#获取赛道中心线坐标
import glob
import os
import sys
from datetime import datetime
try:
    sys.path.append(glob.glob('C:\carla\CARLA_0.9.10\WindowsNoEditor\PythonAPI\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

import carla
import srunner
from carla import Transform, Location, Rotation
from carla import Map
from carla import Vector3D
from srunner.challenge.utils.route_manipulation import interpolate_trajectory

actor_list = []

try:
    client = carla.Client('localhost',2000)
    client.set_timeout(20)

    xodr_path = "Thunderhilll.xodr"
    od_file = open(xodr_path)
    data = od_file.read()

    vertex_distance = 2.0
    max_road_length = 50.0
    wall_height = 0.0
    extra_width = 0.6
    global waypoint_grid
    waypoint_grid = 5

    world = client.generate_opendrive_world(
        data, carla.OpendriveGenerationParameters(
```

```
        vertex_distance=vertex_distance,
        max_road_length=max_road_length,
        wall_height=wall_height,
        additional_width=extra_width,
        smooth_junctions=True,
        enable_mesh_visibility=True))

    weather = carla.WeatherParameters(
    cloudiness=0.0,
    precipitation=0.0,
    sun_altitude_angle=current_hours)
    world.set_weather(weather)
    spectator = world.get_spectator()

    map = world.get_map()
    transform = carla.Transform(carla.Location(x=16.867473602294922, y=191.2739715576172, z=0), carla.Rotation(pitch=0,yaw=180,roll=0))
    end_waypoint = map.get_waypoint(transform.location)
    start_waypoint = map.get_waypoint(transform.location + carla.Vector3D(0, -300, 0))

    waypoints = [start_waypoint.transform.location, end_waypoint.transform.location]
    gps_route, trajectory = interpolate_trajectory(world,waypoints,1)


    def _get_waypoints(trajectory):
        waypoints = []
        for index in range(len(trajectory)):
            waypoint = trajectory[index][0]
            waypoints.append([waypoint.location.x, waypoint.location.y, 20])

        return waypoints


    def text_save(filename, data):
        file = open(filename,'w')
        print(len(data))
        for i in range(len(data)):
            s = str(data[i]).replace('[','').replace(']','')
            s = s.replace("'",'').replace(" ",'') +'\n'
            file.write(s)
        file.close()
```

```
80.         print("保存文件成功")
81.
82.
83.     def draw_trajectory(trajectory, persistency=120, vertical_shift=0):

84.         for index in range(len(trajectory)):
85.             color_start = carla.Color(r=0, g=255, b=0, a =20)
86.             color = carla.Color(r=0, g=0, b=255, a =20)
87.             color_end = carla.Color(r=255, g=0, b=0, a =20)
88.             if index == 0:
89.                 waypoint = trajectory[index][0]
90.                 location = waypoint.location + carla.Location(z=vertica
    l_shift)
91.                 world.debug.draw_string(location, 'START', draw_shadow=
    False,
92.                                         color=color_start, life_tim
    e=persistency,
93.                                         persistent_lines=True)
94.             elif index == len(trajectory)-1:
95.                 waypoint = trajectory[index][0]
96.                 location = waypoint.location + carla.Location(z=vertica
    l_shift)
97.                 world.debug.draw_string(location, 'END', draw_shadow=Fa
    lse,
98.                                         color=color_end, life_time=
    persistency,
99.                                         persistent_lines=True)

100.             else:
101.                 waypoint = trajectory[index][0]
102.                 location = waypoint.location + carla.Location(z=vertic
    al_shift)
103.                 world.debug.draw_string(location, '.', draw_shadow=Fal
    se,
104.                                         color=color, life_time=per
    sistency,
105.                                         persistent_lines=True)
106.     waypoints_list = _get_waypoints(trajectory)
107.     filename = 'C:\yuan\center_waypoints.txt'
108.     text_save(filename, waypoints_list)
109.     draw_trajectory(trajectory,200,0.5)
110.
111. finally:
112.     for actor in actor_list:
```

```
        actor.destroy()
    print("All cleaned up!")
```

```
#计算赛道曲率
def calcu_k(WAYPOINTS_FILENAME , psi_list):
    k_list=[]

    waypoints_file = WAYPOINTS_FILENAME
    waypoints_np   = None
    with open(waypoints_file) as waypoints_file_handle:
        waypoints = list(csv.reader(waypoints_file_handle,
                                    delimiter=',',
                                    quoting=csv.QUOTE_NONNUMERIC))
        waypoints_np = np.array(waypoints)

    for i in arrange(len(waypoint_np)-1):
        if i!= len(waypoint_np)-1:
            x1 = waypoint_np[i-1][0]
            y1 = waypoint_np[i-1][1]
            yaw1 = psi_list[i-1]

            x2 = waypoint_np[i+1][0]
            y2 = waypoint_np[i+1][1]
            yaw2 = psi_list[i+1]
        else:
            x1 = waypoint_np[i-1][0]
            y1 = waypoint_np[i-1][1]
            yaw1 = psi_list[i-1]

            x2 = waypoint_np[0][0]
            y2 = waypoint_np[0][1]
            yaw2 = psi_list[0]

        k = 2*sin(radians((yaw2-yaw1)/2)) / sqrt((x2-x1)**2 + (y2-y1)**2)
        k_list = k_list.append(k)

        return k_list
psi_list =  calcu_psi('center_waypoints.txt' )
k_list =  calcu_k('center_waypoints.txt' , psi_list)
filename = 'C:\yuan\carla_indy\road_k.txt'
data =  k_list
file = open(filename,'w')
for i in range(len(data)):
```

```
    s = str(data[i]).replace('[','').replace(']','')
    s = s.replace("'",'').replace(" ",'') +'\n'
    file.write(s)
file.close()
```

```
#计算赛道朝向角
def calcu_psi(WAYPOINTS_FILENAME):
    psi_list=[]

    waypoints_file = WAYPOINTS_FILENAME
    waypoints_np   = None
    with open(waypoints_file) as waypoints_file_handle:
        waypoints = list(csv.reader(waypoints_file_handle,
                                    delimiter=',',
                                    quoting=csv.QUOTE_NONNUMERIC))
        waypoints_np = np.array(waypoints)

    for i in arrange(len(waypoint_np)-1):
        if i!= len(waypoint_np)-1:
            x1 = waypoint_np[i-1][0]
            y1 = waypoint_np[i-1][1]
            theta1=atan2(y1,x1)

            x2 = waypoint_np[i+1][0]
            y2 = waypoint_np[i+1][1]
            theta2=atan2(y2,x2)
        else:
            x1 = waypoint_np[i][0]
            y1 = waypoint_np[i][1]
            theta1=atan2(y1,x1)

            x2 = waypoint_np[0][0]
            y2 = waypoint_np[0][1]
            theta2=atan2(y2,x2)

        psi = (np.pi+theta1+theta2)/2
        psi_list= psi_list.append(psi)

    return psi_list
psi_list =  calcu_psi('center_waypoints.txt' )
filename = 'C:\yuan\carla_indy\road_psi.txt'
data =  psi_list
file = open(filename,'w')
```

```
for i in range(len(data)):
    s = str(data[i]).replace('[','').replace(']','')
    s = s.replace("'",'').replace(" ",'') +'\n'
    file.write(s)
file.close()
```

```
#计算沿赛道距离
def calcu_s(WAYPOINTS_FILENAME):
    s_list=[]
    s=0

    waypoints_file = WAYPOINTS_FILENAME
    waypoints_np   = None
    with open(waypoints_file) as waypoints_file_handle:
        waypoints = list(csv.reader(waypoints_file_handle,
                                    delimiter=',',
                                    quoting=csv.QUOTE_NONNUMERIC))
        waypoints_np = np.array(waypoints)

    for i in arrange(len(waypoint_np)-1):
        if i= 0:
            s_list = s_list.append(0)
        else:
            x1 = waypoint_np[i][0]
            y1 = waypoint_np[i][1]
            x2 = waypoint_np[i-1][0]
            y2 = waypoint_np[i-1][1]
            s+= sqrt((x2-x1)**2 + (y2-y1)**2)
            s_list = s_list.append(s)

    return s_list
s_list =  calcu_s('center_waypoints.txt' )
filename = 'C:\yuan\carla_indy\road_s.txt'
data =  s_list
file = open(filename,'w')
for i in range(len(data)):
    s = str(data[i]).replace('[','').replace(']','')
    s = s.replace("'",'').replace(" ",'') +'\n'
    file.write(s)
file.close()
```

附录 C：CARLA 模拟器中控制算法

```
#控制算法函数
from numpy.lib import ufunclike
import cutils
import numpy as np
import math
import sys
import os
sys.path.append(os.path.abspath(sys.path[0] + '/..'))
import carla


class Controller2D(object):
    def __init__(self, waypoints, control_method, world):
        self.vars                = cutils.CUtils()
        self._current_x          = 0
        self._current_y          = 0
        self._current_yaw        = 0
        self._current_speed      = 0
        self._desired_speed      = 0
        self._current_frame      = 0
        self._current_timestamp  = 0
        self._start_control_loop = False
        self._set_throttle       = 0
        self._set_brake          = 0
        self._set_steer          = 0
        self._waypoints          = waypoints
        self._conv_rad_to_steer  = 180.0 / 70.0 / np.pi
        self._pi                 = np.pi
        self._2pi                = 2.0 * np.pi
        self._kP                 = 0.95
        self._kI                 = 0.01
        self._kD                 = 0.05
        self._kB                 = 0.2
        self._Kpp                = 4.5
        self._Kvf                = 1.0
        self._Kcte               = 0.7
        self._Kmpc               = 0.3
        self._eps_lookahead      = 10e-3
        self._closest_distance   = 0
        self._wheelbase          = 2.4689
        self._control_method     = control_method
```

```
42.         self._steering_diff      = np.linspace(-5, 5, 21, endpoint = Tr
    ue)
43.         self.world               = world
44.         self._desired_speed_mult = 0.9
45.
46.
47.     def update_values(self, x, y, yaw, speed, timestamp, frame, closest
    _distance):
48.         self._current_x          = x
49.         self._current_y          = y
50.         self._current_yaw        = yaw
51.         self._current_speed      = speed
52.         self._current_timestamp = timestamp
53.         self._current_frame      = frame
54.         self._closest_distance   = closest_distance
55.         if self._current_frame:
56.             self._start_control_loop = True
57.
58.
59.     def update_desired_speed(self):
60.         min_idx       = 0
61.         min_dist      = float("inf")
62.         desired_speed = 0
63.         for i in range(len(self._waypoints)):
64.             dist = np.linalg.norm(np.array([
65.                     self._waypoints[i][0] - self._current_x,
66.                     self._waypoints[i][1] - self._current_y]))
67.             if dist < min_dist:
68.                 min_dist = dist
69.                 min_idx = i
70.         if min_idx < len(self._waypoints)-1:
71.             desired_speed = self._waypoints[min_idx][2]
72.         else:
73.             desired_speed = self._waypoints[-1][2]
74.         self._desired_speed = desired_speed * self._desired_speed_mult
    -  1
75.
76.
77.     def update_waypoints(self, new_waypoints):
78.         self._waypoints = new_waypoints
79.
80.
81.     def get_commands(self):
82.         return self._set_throttle, self._set_steer, self._set_brake
```

```
83.
84.
85.     def set_throttle(self, input_throttle):
86.         throttle           = np.fmax(np.fmin(input_throttle, 1.0), 0.0)

87.         self._set_throttle = throttle
88.
89.
90.     def set_steer(self, input_steer_in_rad):
91.         input_steer = self._conv_rad_to_steer * input_steer_in_rad
92.         steer           = np.fmax(np.fmin(input_steer, 1.0), -1.0)
93.         self._set_steer = steer
94.
95.
96.     def set_brake(self, input_brake):
97.         brake           = np.fmax(np.fmin(input_brake, 1.0), 0.0)
98.         self._set_brake = brake
99.
100.
101.     def calculate_throttle(self, t, v, v_desired):
102.         time_step = t - self.vars.t_previous
103.         speed_error = v_desired - v
104.         p_term = self._kP * speed_error
105.         i_term = self.vars.i_term_previous + self._kI * time_step * sp
    eed_error
106.         d_term = self._kD * speed_error / time_step
107.         self.vars.i_term_previous = i_term
108.         u = p_term + i_term + d_term
109.         if u>=0:
110.             Tp = u
111.             Bp = 0.0
112.         else:
113.             Tp = 0.0
114.             Bp = (-1)*u*self._kB
115.
116.         return Tp, Bp
117.
118.
119.     def get_shifted_coordinate(self, x, y, yaw, length):
120.         x_shifted = x + length*np.cos(yaw)
121.         y_shifted = y + length*np.sin(yaw)
122.
123.         return x_shifted, y_shifted
124.
```

```
125.
126.     def get_lookahead_dis(self, v):
127.         return self._Kvf*v
128.
129.
130.     def get_distance(self, x1, y1, x2, y2):
131.         return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
132.
133.
134.     def get_goal_waypoint_index(self, x, y, waypoints, lookahead_dis):

135.         for i in range(len(waypoints)):
136.             dis = self.get_distance(x, y, waypoints[i][0], waypoints[i
    ][1])
137.             if abs(dis - lookahead_dis) <= self._eps_lookahead:
138.                 return i
139.
140.         return len(waypoints)-1
141.
142.
143.     def get_alpha(self, v1, v2, lookahead_dis):
144.         inner_prod = v1[0]*v2[0] + v1[1]*v2[1]
145.
146.         return np.arccos(inner_prod/lookahead_dis)
147.
148.
149.     def get_steering_direction(self, v1, v2):
150.         corss_prod = v1[0]*v2[1] - v1[1]*v2[0]
151.         if corss_prod >= 0 :
152.             return -1
153.
154.         return 1
155.
156.
157.     def get_heading_error(self, waypoints, current_yaw):
158.         waypoint_delta_x = waypoints[1][0] - waypoints[0][0]
159.         waypoint_delta_y = waypoints[1][1] - waypoints[0][1]
160.         waypoint_heading = np.arctan(waypoint_delta_y/waypoint_delta_x
    )
161.         heading_error_mod = divmod((waypoint_heading - current_yaw), n
    p.pi)[1]
162.         if heading_error_mod > np.pi/2 and heading_error_mod < np.pi:

163.             heading_error_mod -= np.pi
```

```
164.
165.         return heading_error_mod
166.
167.
168.     def get_cte_heading_error(self, v):
169.         proportional_cte_error = self._Kcte * self._closest_distance
170.         cte_heading_error = np.arctan(proportional_cte_error/v)
171.         cte_heading_error_mod = divmod(cte_heading_error, np.pi)[1]
172.         if cte_heading_error_mod > np.pi/2 and cte_heading_error_mod <
    np.pi:
173.             cte_heading_error_mod -= np.pi
174.
175.         return cte_heading_error_mod
176.
177.
178.     def calculate_steering(self, x, y, yaw, waypoints, v):
179.         if self._control_method == 'PurePursuit':
180.             lookahead_dis = self.get_lookahead_dis(v)
181.             print(lookahead_dis)
182.             idx = self.get_goal_waypoint_index(x, y, waypoints, lookah
   ead_dis)
183.             world = self.world
184.
185.             world.debug.draw_string(carla.Location(x=waypoints[idx][0]
   ,y = waypoints[idx][1]), 'O', draw_shadow=False,
186.                                     color=carla.Color(r=255, g=0, b=0,
    a =20), life_time=0.001,
187.                                     persistent_lines=True)
188.
189.             world.debug.draw_string(carla.Location(x,y), 'O', draw_sha
   dow=False,
190.                                     color=carla.Color(r=0, g=255, b=0,
    a =20), life_time=0.001,
191.                                     persistent_lines=True)
192.
193.             v1 = [waypoints[idx][0] - x, waypoints[idx][1] - y]
194.             v2 = [np.cos(yaw), np.sin(yaw)]
195.             alpha = self.get_alpha(v1, v2, lookahead_dis)
196.             if math.isnan(alpha):
197.                 alpha = self.vars.alpha_previous
198.             if not math.isnan(alpha):
199.                 self.vars.alpha_previous = alpha
200.             steering = self.get_steering_direction(v1, v2)*np.arctan((
   2*self._wheelbase*np.sin(alpha))/(self._Kpp*v))
```

```
201.             if math.isnan(steering):
202.                 steering = self.vars.steering_previous
203.             if not math.isnan(steering):
204.                 self.vars.steering_previous = steering
205.             return steering
206.         elif self._control_method == 'Stanley':
207.             world = self.world
208.
209.             world.debug.draw_string(carla.Location(x=waypoints[300][0]
    ,y = waypoints[300][1]), 'O', draw_shadow=False,
210.                                     color=carla.Color(r=255, g=0, b=0,
     a =20), life_time=0.001,
211.                                     persistent_lines=True)
212.
213.             world.debug.draw_string(carla.Location(x,y), 'O', draw_sha
    dow=False,
214.                                     color=carla.Color(r=0, g=255, b=0,
     a =20), life_time=0.001,
215.                                     persistent_lines=True)
216.
217.             v1 = [waypoints[300][0] - x, waypoints[300][1] - y]
218.             v2 = [np.cos(yaw), np.sin(yaw)]
219.             heading_error = self.get_heading_error(waypoints, yaw)
220.             cte_error = self.get_steering_direction(v1, v2)*self.get_c
    te_heading_error(v)
221.             steering =  heading_error + cte_error
222.             return steering
223.
224.         else:
225.             return 0
226.
227.
228.     def update_controls(self):
229.
230.         x               = self._current_x
231.         y               = self._current_y
232.         yaw             = self._current_yaw
233.         v               = self._current_speed
234.         self.update_desired_speed()
235.         v_desired       = self._desired_speed
236.         t               = self._current_timestamp
237.         waypoints       = self._waypoints
238.         throttle_output = 0
239.         steer_output    = 0
```

```
        brake_output    = 0

        self.vars.create_var('v_previous', 0.0)
        self.vars.create_var('t_previous',0.0)
        self.vars.create_var('i_term_previous',0.0)
        self.vars.create_var('alpha_previous',0.0)
        self.vars.create_var('steering_previous',0.0)

        if self._start_control_loop:

            throttle_output, brake_output = self.calculate_throttle(t, v, v_desired)
            steer_output    = self.calculate_steering(x, y, yaw, waypoints, v)

            self.set_throttle(throttle_output)
            self.set_steer(steer_output)
            self.set_brake(brake_output)

        self.vars.v_previous = v
        self.vars.t_previous = t
```

```
#控制主程序
from __future__ import print_function
from __future__ import division

import sys
import os
import argparse
import logging
import time
import math
import numpy as np
import csv
import matplotlib.pyplot as plt
import controller2d
import configparser
os.environ['PYGAME_HIDE_SUPPORT_PROMPT'] = "hide"

sys.path.append(os.path.abspath(sys.path[0] + '/..'))
import live_plotter as lv
import carla
from datetime import datetime
```

```
22.import srunner
23.from srunner.challenge.utils.route_manipulation import interpolate_traj
   ectory
24.
25.
26.ITER_FOR_SIM_TIMESTEP  = 10
27.WAIT_TIME_BEFORE_START = 5.00
28.TOTAL_RUN_TIME         = 200.00
29.TOTAL_FRAME_BUFFER     = 300
30.
31.
32.FIGSIZE_X_INCHES   = 8
33.FIGSIZE_Y_INCHES   = 8
34.PLOT_LEFT          = 0.1
35.PLOT_BOT           = 0.1
36.PLOT_WIDTH         = 0.8
37.PLOT_HEIGHT        = 0.8
38.
39.
40.WAYPOINTS_FILENAME = 'stanford_waypoints_full.txt'
41.
42.
43.DIST_THRESHOLD_TO_LAST_WAYPOINT = 10.0
44.
45.INTERP_MAX_POINTS_PLOT    = 10
46.INTERP_LOOKAHEAD_DISTANCE = 20
47.INTERP_DISTANCE_RES       = 0.01
48.
49.CONTROLLER_OUTPUT_FOLDER = os.path.dirname(os.path.realpath(__file__))
   +\
50.                           '/controller_output/'
51.
52.
53.
54.class Timer(object):
55.
56.    def __init__(self, period):
57.        self.step = 0
58.        self._lap_step = 0
59.        self._lap_time = time.time()
60.        self._period_for_lap = period
61.
62.
63.    def tick(self):
```

```
64.         self.step += 1
65.
66.
67.     def has_exceeded_lap_period(self):
68.         if self.elapsed_seconds_since_lap() >= self._period_for_lap:
69.             return True
70.         else:
71.             return False
72.
73.
74.     def lap(self):
75.         self._lap_step = self.step
76.         self._lap_time = time.time()
77.
78.
79.     def ticks_per_second(self):
80.         return float(self.step - self._lap_step) /\
81.                       self.elapsed_seconds_since_lap()
82.
83.
84.     def elapsed_seconds_since_lap(self):
85.         return time.time() - self._lap_time
86.
87.
88.
89.
90.def send_control_command(vehicle,throttle, steer, brake,
91.                           hand_brake=False, reverse=False):
92.
93.
94.     steer = np.fmax(np.fmin(steer, 1.0), -1.0)
95.     throttle = np.fmax(np.fmin(throttle, 1.0), 0)
96.     brake = np.fmax(np.fmin(brake, 1.0), 0)
97.     control_para = carla.VehicleControl(throttle, steer, brake)
98.     control_para.hand_brake = hand_brake
99.     control_para.reverse = reverse
100.     vehicle.apply_control(control_para)
101.
102.
103.def create_controller_output_dir(output_folder):
104.     if not os.path.exists(output_folder):
105.         os.makedirs(output_folder)
106.
107.
```

```
108.def store_trajectory_plot(graph, fname):
109.    create_controller_output_dir(CONTROLLER_OUTPUT_FOLDER)
110.    file_name = os.path.join(CONTROLLER_OUTPUT_FOLDER, fname)
111.    graph.savefig(file_name)
112.
113.
114.def write_trajectory_file(x_list, y_list, v_list, t_list):
115.    create_controller_output_dir(CONTROLLER_OUTPUT_FOLDER)
116.    file_name = os.path.join(CONTROLLER_OUTPUT_FOLDER, 'trajectory.txt
   ')
117.
118.    with open(file_name, 'w') as trajectory_file:
119.        for i in range(len(x_list)):
120.            trajectory_file.write('%3.3f, %3.3f, %2.3f, %6.3f\n' %\
121.                                  (x_list[i], y_list[i], v_list[i], t_
   list[i]))
122.
123.
124.def make_carla_client(host,port):
125.
126.    client = carla.Client(host, port)
127.    client.set_timeout(30.0)
128.
129.    xodr_path = "Thunderhilll.xodr"
130.    od_file = open(xodr_path)
131.    data = od_file.read()
132.    vertex_distance = 2.0
133.    max_road_length = 50.0
134.    wall_height = 0.5
135.    extra_width = 0.6
136.
137.    world = client.generate_opendrive_world(
138.        data, carla.OpendriveGenerationParameters(
139.        vertex_distance=vertex_distance,
140.        max_road_length=max_road_length,
141.        wall_height=wall_height,
142.        additional_width=extra_width,
143.        smooth_junctions=True,
144.        enable_mesh_visibility=True))
145.
146.    world_snapshot = world.get_snapshot()
147.    world_snapshot = world.wait_for_tick()
148.
149.    world.on_tick(lambda world_snapshot: world_snapshot)
```

```
150.
151.     now = datetime.now()
152.     current_hours = int(now.strftime('%H'))
153.     sun_angle =  current_hours*(15)-90 if (current_hours <= 12 and cur
    rent_hours >= 0) else current_hours*(-15)+270
154.
155.     weather = carla.WeatherParameters(
156.     cloudiness=0.0,
157.     precipitation=0.0,
158.     sun_altitude_angle=sun_angle)
159.     world.set_weather(weather)
160.     spectator = world.get_spectator()
161.
162.     map = world.get_map()
163.     print('make_carla_client----Carla client connected.')
164.
165.     transform = carla.Transform(carla.Location(x=18.30674744, y=191.23
    94257, z=0), carla.Rotation(pitch=0,yaw=0,roll=0))
166.     start_waypoint = map.get_waypoint(transform.location)
167.
168.     waypoint = start_waypoint
169.     blueprint = world.get_blueprint_library().filter('vehicle.*tt*')[0
    ]
170.     blueprint.set_attribute('color', '255,255,246')
171.     location = waypoint.transform.location + carla.Vector3D(0, 0, 1.5)

172.     rotation = carla.Rotation(pitch=0,yaw=90,roll=0)
173.
174.     vehicle = world.spawn_actor(blueprint, carla.Transform(location, r
    otation))
175.     print("make_carla_client----SPAWNED!")
176.     vehicle.set_simulate_physics(True)
177.     vehicle.set_light_state(carla.VehicleLightState(carla.VehicleLight
    State.All | carla.VehicleLightState.LowBeam | carla.VehicleLightState.H
    ighBeam | carla.VehicleLightState.Interior))
178.
179.     camera_bp = world.get_blueprint_library().find('sensor.camera.rgb'
    )
180.     camera_transform = carla.Transform(carla.Location(x=-10,z=7), carl
    a.Rotation(-20,0,0))
181.     camera = world.spawn_actor(camera_bp, camera_transform, attach_to=
    vehicle)
182.     spectator.set_transform(camera.get_transform())
183.     spectator.set_transform(camera.get_transform())
```

```
184.     print("make_carla_client----Camera Done!")
185. 
186.     return client, world, spectator, map , vehicle, camera
187. 
188. def exec_waypoint_nav_demo(args):
189. 
190.     if args.control_method == 'PurePursuit':
191.         print("Control method selected : Pure Pursuit\n")
192.     elif args.control_method == 'Stanley':
193.         print("Control method selected : Stanley\n")
194.     else:
195.         print("Control method selected : Unknown\n")
196. 
197.     client, world, spectator, map, vehicle, camera = make_carla_client
    (args.host,args.port)
198. 
199.     print('Carla client connected.')
200. 
201. 
202.     config = configparser.ConfigParser()
203.     config.read(os.path.join(
204.             os.path.dirname(os.path.realpath(__file__)), 'options.cfg'
    ))
205.     demo_opt = config['Demo Parameters']
206. 
207.     enable_live_plot = demo_opt.get('live_plotting', 'true').capitaliz
    e()
208.     enable_live_plot = enable_live_plot == 'True'
209.     live_plot_period = float(demo_opt.get('live_plotting_period', 0))

210. 
211.     live_plot_timer = Timer(live_plot_period)
212. 
213. 
214.     waypoints_file = WAYPOINTS_FILENAME
215.     waypoints_np   = None
216.     with open(waypoints_file) as waypoints_file_handle:
217.         waypoints = list(csv.reader(waypoints_file_handle,
218.                                     delimiter=',',
219.                                     quoting=csv.QUOTE_NONNUMERIC))
220.         waypoints_np = np.array(waypoints)
221.     print('Waypoints loaded')
222. 
223. 
```

```
    for w in waypoints_np:
        world.debug.draw_string(carla.Location(x=w[0],y = w[1]), 'O', draw_shadow=False,
                                        color=carla.Color(r=255, g=255, b=0, a =200), life_time=500.0,
                                        persistent_lines=True)


    wp_distance = []
    for i in range(1, waypoints_np.shape[0]):
        wp_distance.append(
                np.sqrt((waypoints_np[i, 0] - waypoints_np[i-1, 0])**2 +
                        (waypoints_np[i, 1] - waypoints_np[i-1, 1])**2))
    wp_distance.append(0)

    wp_interp      = []
    wp_interp_hash = []
    interp_counter = 0
    print("numpy original: ",waypoints_np.shape[0])

    delete_items = []
    for i in range(waypoints_np.shape[0] - 1):

        wp_vector = waypoints_np[i+1] - waypoints_np[i]
        if wp_vector[0] == 0:
            delete_items.append(i+1)

    waypoints_np = np.delete(waypoints_np, delete_items, 0)
    print("numpy deleted: ", waypoints_np.shape[0])

    for i in range(waypoints_np.shape[0] - 1):

        wp_interp.append(list(waypoints_np[i]))
        wp_interp_hash.append(interp_counter)
        interp_counter+=1

        num_pts_to_interp = int(np.floor(wp_distance[i] /\
                                        float(INTERP_DISTANCE_RES)) - 1)
        wp_vector = waypoints_np[i+1] - waypoints_np[i]
        wp_uvector = wp_vector / np.linalg.norm(wp_vector)
        for j in range(num_pts_to_interp):
```

```
263.             next_wp_vector = INTERP_DISTANCE_RES * float(j+1) * wp_uve
    ctor
264.             wp_interp.append(list(waypoints_np[i] + next_wp_vector))
265.             interp_counter+=1
266.     print('Interpolate Finish')
267.
268.     wp_interp.append(list(waypoints_np[-1]))
269.     wp_interp_hash.append(interp_counter)
270.     interp_counter+=1
271.
272.
273.     controller = controller2d.Controller2D(waypoints, args.control_met
    hod, world)
274.     print('Controller loaded')
275.
276.     num_iterations = ITER_FOR_SIM_TIMESTEP
277.     if (ITER_FOR_SIM_TIMESTEP < 1):
278.         num_iterations = 1
279.     print("Closed")
280.
281.     timestamp = world.wait_for_tick()
282.     sim_start_stamp = timestamp.elapsed_seconds
283.     send_control_command(vehicle,throttle=0.0, steer=0, brake=1.0)
284.
285.     sim_duration = 0
286.     for i in range(num_iterations):
287.
288.         timestamp = world.wait_for_tick()
289.         now_stamp = timestamp.elapsed_seconds
290.         send_control_command(vehicle,throttle=0.0, steer=0, brake=1.0)

291.         if i == num_iterations - 1:
292.             print(now_stamp,sim_start_stamp)
293.             sim_duration = now_stamp - sim_start_stamp
294.     print(sim_duration)
295.
296.
297.     SIMULATION_TIME_STEP = sim_duration / float(num_iterations)
298.     print("SERVER SIMULATION STEP APPROXIMATION: " + \
299.             str(SIMULATION_TIME_STEP))
300.     TOTAL_EPISODE_FRAMES = int((TOTAL_RUN_TIME + WAIT_TIME_BEFORE_STAR
    T) /\
301.                                SIMULATION_TIME_STEP) + TOTAL_FRAME_BUFFER
```

```
302.     print("total frame: ",TOTAL_EPISODE_FRAMES)
303. 
304.     start_x = round(vehicle.get_transform().location.x,2)
305.     start_y = round(vehicle.get_transform().location.y,2)
306.     start_yaw = round(vehicle.get_transform().rotation.yaw,2)
307. 
308.     send_control_command(vehicle,throttle=0.0, steer=0, brake=1.0)
309.     x_history     = [start_x]
310.     y_history     = [start_y]
311.     yaw_history   = [start_yaw]
312.     time_history  = [0]
313.     speed_history = [0]
314. 
315.     lp_traj = lv.LivePlotter(tk_title="Trajectory Trace")
316.     lp_1d = lv.LivePlotter(tk_title="Controls Feedback")
317. 
318.     trajectory_fig = lp_traj.plot_new_dynamic_2d_figure(
319.             title='Vehicle Trajectory',
320.             figsize=(FIGSIZE_X_INCHES, FIGSIZE_Y_INCHES),
321.             edgecolor="black",
322.             rect=[PLOT_LEFT, PLOT_BOT, PLOT_WIDTH, PLOT_HEIGHT])
323. 
324.     trajectory_fig.set_invert_x_axis()
325.     trajectory_fig.set_axis_equal()
326. 
327.     trajectory_fig.add_graph("waypoints", window_size=waypoints_np.sha
    pe[0],
328.                              x0=waypoints_np[:,0], y0=waypoints_np[
    :,1],
329.                              linestyle="-", marker="", color='g')
330. 
331.     trajectory_fig.add_graph("trajectory", window_size=TOTAL_EPISODE_F
    RAMES,
332.                              x0=[start_x]*TOTAL_EPISODE_FRAMES,
333.                              y0=[start_y]*TOTAL_EPISODE_FRAMES,
334.                              color=[1, 0.5, 0])
335. 
336.     trajectory_fig.add_graph("lookahead_path",
337.                              window_size=INTERP_MAX_POINTS_PLOT,
338.                              x0=[start_x]*INTERP_MAX_POINTS_PLOT,
339.                              y0=[start_y]*INTERP_MAX_POINTS_PLOT,
340.                              color=[0, 0.7, 0.7],
341.                              linewidth=4)
342. 
```

```
343.     trajectory_fig.add_graph("start_pos", window_size=1,
344.                                  x0=[start_x], y0=[start_y],
345.                                  marker=11, color=[1, 0.5, 0],
346.                                  markertext="Start", marker_text_offset
   =1)
347.
348.     trajectory_fig.add_graph("end_pos", window_size=1,
349.                                  x0=[waypoints_np[-1, 0]],
350.                                  y0=[waypoints_np[-1, 1]],
351.                                  marker="D", color='r',
352.                                  markertext="End", marker_text_offset=1
   )
353.
354.     trajectory_fig.add_graph("car", window_size=1,
355.                                  marker="s", color='b', markertext="Car
   ",
356.                                  marker_text_offset=1)
357.
358.     forward_speed_fig =\
359.             lp_1d.plot_new_dynamic_figure(title="Forward Speed (km/h)"
   )
360.     forward_speed_fig.add_graph("forward_speed",
361.                                  label="forward_speed",
362.                                  window_size=TOTAL_EPISODE_FRAMES)
363.     forward_speed_fig.add_graph("reference_signal",
364.                                  label="reference_Signal",
365.                                  window_size=TOTAL_EPISODE_FRAMES)
366.
367.     throttle_fig = lp_1d.plot_new_dynamic_figure(title="Throttle (%)")

368.     throttle_fig.add_graph("throttle",
369.                            label="throttle",
370.                            window_size=TOTAL_EPISODE_FRAMES)
371.
372.     brake_fig = lp_1d.plot_new_dynamic_figure(title="Brake (%)")
373.     brake_fig.add_graph("brake",
374.                            label="brake",
375.                            window_size=TOTAL_EPISODE_FRAMES)
376.
377.
378.     steer_fig = lp_1d.plot_new_dynamic_figure(title="Steer (Degree)")

379.     steer_fig.add_graph("steer",
380.                            label="steer",
```

```
                                window_size=TOTAL_EPISODE_FRAMES)


    if not enable_live_plot:
        lp_traj._root.withdraw()
        lp_1d._root.withdraw()

    reached_the_end = False
    skip_first_frame = True
    closest_index    = 0
    closest_distance = 0


    for frame in range(TOTAL_EPISODE_FRAMES):

        current_x = vehicle.get_transform().location.x
        current_y = vehicle.get_transform().location.y
        current_yaw = math.radians(vehicle.get_transform().rotation.yaw)
        current_x, current_y = controller.get_shifted_coordinate(current_x, current_y, current_yaw, math.sqrt(4**2+0.5**2))

        current_speed = math.sqrt(vehicle.get_velocity().x**2+vehicle.get_velocity().y**2)
        timestamp = world.wait_for_tick()
        current_timestamp = timestamp.elapsed_seconds

        spectator = world.get_spectator()
        transform = vehicle.get_transform()

        spectator.set_transform(camera.get_transform())


        if args.control_method == 'PurePursuit':
            length = -1.2
        elif args.control_method == 'Stanley' :
            length = 1.2
        else:
            length = 0.0
        shifted_x, shifted_y = controller.get_shifted_coordinate(current_x, current_y, current_yaw, length)
        world.debug.draw_string(carla.Location(x=shifted_x,y = shifted_y), 'X', draw_shadow=False,
```

```
                                color=carla.Color(r=0, g=0, b=255, a =255), life_time=0.01,
                                persistent_lines=True)

        if current_timestamp <= WAIT_TIME_BEFORE_START:
            send_control_command(vehicle,throttle=0.0, steer=0, brake=1.0)
            continue
        else:
            current_timestamp = current_timestamp - WAIT_TIME_BEFORE_START


        x_history.append(current_x)
        y_history.append(current_y)
        yaw_history.append(current_yaw)
        speed_history.append(current_speed)
        time_history.append(current_timestamp)


        closest_distance = np.linalg.norm(np.array([
                waypoints_np[closest_index, 0] - current_x,
                waypoints_np[closest_index, 1] - current_y]))

        new_distance = closest_distance
        new_index = closest_index
        while new_distance <= closest_distance:
            closest_distance = new_distance
            closest_index = new_index
            new_index += 1
            if new_index >= waypoints_np.shape[0]:
                break
            new_distance = np.linalg.norm(np.array([
                    waypoints_np[new_index, 0] - current_x,
                    waypoints_np[new_index, 1] - current_y]))
        new_distance = closest_distance
        new_index = closest_index
        while new_distance <= closest_distance:
            closest_distance = new_distance
            closest_index = new_index
            new_index -= 1
            if new_index < 0:
                break
            new_distance = np.linalg.norm(np.array([
```

```
460.                     waypoints_np[new_index, 0] - current_x,
461.                     waypoints_np[new_index, 1] - current_y]))
462.
463.         waypoint_subset_first_index = closest_index - 1
464.         if waypoint_subset_first_index < 0:
465.             waypoint_subset_first_index = 0
466.
467.         waypoint_subset_last_index = closest_index
468.         total_distance_ahead = 0
469.         while total_distance_ahead < INTERP_LOOKAHEAD_DISTANCE:
470.             total_distance_ahead += wp_distance[waypoint_subset_last_i
    ndex]
471.             waypoint_subset_last_index += 1
472.             if waypoint_subset_last_index >= waypoints_np.shape[0]:
473.                 waypoint_subset_last_index = waypoints_np.shape[0] - 1

474.                 break
475.
476.         new_waypoints = \
477.                 wp_interp[wp_interp_hash[waypoint_subset_first_index]:
    \
478.                           wp_interp_hash[waypoint_subset_last_index]
     + 1]
479.         controller.update_waypoints(new_waypoints)
480.
481.         controller.update_values(current_x, current_y, current_yaw,
482.                                  current_speed,
483.                                  current_timestamp, frame, new_dist
    ance)
484.
485.         controller.update_controls()
486.         cmd_throttle, cmd_steer, cmd_brake = controller.get_commands()

487.
488.
489.         if skip_first_frame and frame == 0:
490.             pass
491.         else:
492.
493.             trajectory_fig.roll("trajectory", current_x, current_y)
494.             trajectory_fig.roll("car", current_x, current_y)
495.
496.             new_waypoints_np = np.array(new_waypoints)
497.             path_indices = np.floor(np.linspace(0,
```

```
                                                    new_waypoints_np.shape
[0]-1,
                                                    INTERP_MAX_POINTS_PLOT
))

            trajectory_fig.update("lookahead_path",
                    new_waypoints_np[path_indices.astype(int), 0],
                    new_waypoints_np[path_indices.astype(int), 1],
                    new_colour=[0, 0.7, 0.7])

            forward_speed_fig.roll("forward_speed",
                                   current_timestamp,
                                   current_speed*3.6)
            forward_speed_fig.roll("reference_signal",
                                   current_timestamp,
                                   controller._desired_speed*3.6)

            throttle_fig.roll("throttle", current_timestamp, cmd_throt
tle*100)
            brake_fig.roll("brake", current_timestamp, cmd_brake*100)

            steer_fig.roll("steer", current_timestamp, cmd_steer*180/n
p.pi)


            if enable_live_plot and \
                live_plot_timer.has_exceeded_lap_period():
                lp_traj.refresh()
                lp_1d.refresh()
                live_plot_timer.lap()

        send_control_command(vehicle,throttle=cmd_throttle,
                             steer=cmd_steer*0.5,
                             brake=cmd_brake)


        dist_to_last_waypoint = np.linalg.norm(np.array([
            waypoints[-1][0] - current_x,
            waypoints[-1][1] - current_y]))
        print("dist_to_last_waypoint:",dist_to_last_waypoint)
        if  dist_to_last_waypoint < DIST_THRESHOLD_TO_LAST_WAYPOINT:
            reached_the_end = True
            print("End")
            break
```

```
537.
538.     if reached_the_end:
539.         print("Reached the end of path. Writing to controller_output..
    .")
540.     else:
541.         print("Exceeded assessment time. Writing to controller_output.
    ..")
542.
543.     send_control_command(vehicle,throttle=0.0, steer=0.0, brake=1.0)
544.
545.     store_trajectory_plot(trajectory_fig.fig, 'trajectory.png')
546.     store_trajectory_plot(forward_speed_fig.fig, 'forward_speed.png')

547.     store_trajectory_plot(throttle_fig.fig, 'throttle_output.png')
548.     store_trajectory_plot(brake_fig.fig, 'brake_output.png')
549.     store_trajectory_plot(steer_fig.fig, 'steer_output.png')
550.     write_trajectory_file(x_history, y_history, speed_history, time_hi
    story)
551.
552.
553.def main():
554.
555.     argparser = argparse.ArgumentParser(description=__doc__)
556.     argparser.add_argument(
557.         '-v', '--verbose',
558.         action='store_true',
559.         dest='debug',
560.         help='print debug information')
561.     argparser.add_argument(
562.         '--host',
563.         metavar='H',
564.         default='127.0.0.1',
565.         help='IP of the host server (default: localhost)')
566.     argparser.add_argument(
567.         '-p', '--port',
568.         metavar='P',
569.         default=2000,
570.         type=int,
571.         help='TCP port to listen to (default: 2000)')
572.     argparser.add_argument(
573.         '-a', '--autopilot',
574.         action='store_true',
575.         help='enable autopilot')
576.     argparser.add_argument(
```

```
577.         '-q', '--quality-level',
578.         choices=['Low', 'Epic'],
579.         type=lambda s: s.title(),
580.         default='Low',
581.         help='graphics quality level.')
582.     argparser.add_argument(
583.         '-c', '--carla-settings',
584.         metavar='PATH',
585.         dest='settings_filepath',
586.         default=None,
587.         help='Path to a "CarlaSettings.ini" file')
588.     argparser.add_argument(
589.         '--control-method',
590.         metavar='CONTROL_METHOD',
591.         dest='control_method',
592.         choices = {'PurePursuit','Stanley'},
593.         default=' Stanley ',
594.         help='Select control method for Lane Keeping Assist : PurePurs
    uit or Stanley ')
595.     args = argparser.parse_args()
596.
597.
598.     log_level = logging.DEBUG if args.debug else logging.INFO
599.     logging.basicConfig(format='%(levelname)s: %(message)s', level=log
    _level)
600.     logging.info('listening to server %s:%s', args.host, args.port)
601.
602.     args.out_filename_format = '_out/episode_{:0>4d}/{:s}/{:0>6d}'
603.
604.     while True:
605.         try:
606.             exec_waypoint_nav_demo(args)
607.             print('mark2')
608.             print(args)
609.             print('Done.')
610.             return
611.
612.         except:
613.             print("break out")
614.             break
615.
616. if __name__ == '__main__':
617.
618.     try:
```

```
619.         main()
620.     except KeyboardInterrupt:
621.         print('\nCancelled by user.')
```