

结构化环境中无人车辆的轨迹规划研究

邬杨明

北京科技大学

论文题目: 结构化环境中无人车辆的轨迹规划研究

学 号: S20170512

作 者: 邬杨明

专业名称: 机械工程

2019 年 11 月 18 日

结构化环境中无人车辆的轨迹规划研究

Research on trajectory planning of autonomous vehicles in structured environments

研究生姓名：邬杨明

指导教师姓名：孟宇

北京科技大学机械学院

北京 100083，中国

Master Degree Candidate: Wu Yangming

Supervisor: Meng Yu

School of Mechanical Engineering

University of Science and Technology Beijing

30 Xueyuan Road, Haidian District

Beijing 100083, P.R.CHINA

分类号: TP242

密 级: 公开

U D C:

单位代码: 10008

北京科技大学硕士学位论文

论文题目: 结构化环境中无人车辆的轨迹规划研究

作者: 邬杨明

指导教师: 孟宇 副教授 单位: 北京科技大学

指导小组成员: 顾青 讲师 单位: 北京科技大学

刘立 教授 单位: 北京科技大学

论文提交日期: 2019年11月18日

学位授予单位: 北京科技大学

致 谢

六年半的北科大生活已经进入了终章，回忆起来有许多感慨，大学一年级学的是工业设计，学了大半年之后发现并不适合我，大学二年级转专业来到了车辆工程系，开始学习很多关于工程学的知识，推免后选择了车辆电子实验室，进入研究生阶段决定认真做一点科研，出一点成果，弥补自己本科期间缺少竞赛经历的遗憾，因为转专业后确实是太忙了，我记得我一年刷了60多学分，人都累傻了。进入管庄校区，开始了解自动驾驶相关的知识，经过一年的了解，当时觉得决策（decision-making）才是真正的人工智能，可以适合自己大展拳脚，觉得自己可以大干一场了，后来经过一些调研和学习，发现这一行的入门资料太少，学习工具太匮乏，入行门槛太高，并且整个领域也处于研究的起步阶段，没有一个比较有说服力的框架，也没有一套适应性较广的算法，并且也缺乏有经验的老师和师兄的指导，最后算是放弃决策这一块的研究，退而求其次，选择了运动规划这一课题的研究。在这一课题的研究中，也出了一篇SCI文章，接下来在这里我要进行一一道谢。

首先感谢我的恩师孟宇副教授对我的谆谆教诲和悉心关怀，孟老师以全面系统的专业知识和高超的研究水平对我倾囊相授，并在论文的选题、设计和实施过程中给予具体的指导，在生活上，孟老师也经常对我促膝长谈，帮我排忧解难，使我更加积极地去面对生活。在每一周的组会上，孟老师会详细谈到论文和项目上的问题，详细了解困难，提出解决方案。在期刊论文的发表上，孟老师花了大量精力，进行论文整体架构的调整，关于论文的综述和贡献部分，也提出了大量漂亮的意见，关于写作的方法论，孟老师也进行了大量指导，孟老师高屋建瓴地指出了论文中存在的许多问题。最后，在论文的回复信中，孟老师也提出了大量的修改意见，使得我能够理解审稿人提出的问题，并且使得我做出了合理的修改以及对审稿人的意见有一个完美的回复。

接下来继续感谢顾青老师和刘立老师，感谢顾老师在我本科毕业设计期间对我的课题指导，在我研究生期间每一次组会上的课题指导。感谢刘老师在我开题答辩、中期答辩和预答辩中对我课题上的指导。两位老师使得我在运动规划领域，一步一步走得更深，走得更扎实，使得我发现该领域中的问题所在，使得我确定要解决的问题，最终我能够完成小论文和大论文，这离不开各位老师的贡献。

最后，感谢课题组的师兄师姐，同级同学，以及师弟师妹，大家能够一

起学习，进步，以及成长，实所幸甚！感谢家人，对我二十多年来的热情鼓励和无私奉献！感谢深蓝学院，慧拓无限科技有限公司，泡泡机器人，在这里我学到了很多运动规划领域的专业知识，学习到了最新的理论，提高了工程实践能力，也充实了我研究生期间的生活。感谢所有！

摘要

本文基于 lattice 搜索和凸优化，提出了一个适用于结构化道路自动驾驶车辆的解耦轨迹规划框架，对于一个包含时间信息的三维轨迹规划问题，由于多种硬约束和软约束的存在，规划空间的可行域通常是非凸的，所以对于轨迹规划算法来说很容易陷入局部最优，并且对于非完备的算法来说很可能不能找到一个可行解。并且轨迹规划问题的解空间，不管是连续的还是离散的，解空间都是巨大的，所以一个算法很难在满足实时性的同时生成一个最优解。为了应对状态空间的非凸问题，以及改善轨迹优化的收敛速度，本文考虑到 lattice 搜索能够离散行驶环境和减小解空间，选择了 lattice 搜索作为轨迹规划的前端算法。Lattice 搜索生成的路径或者轨迹会位于全局最优解的领域内，所以它生成的解是接近全局最优的，可以成为次优解。然而这个次优解无论在空间上还是时间上往往都不够平滑，经常会包含曲率、曲率导数、加速度以及加加速度的跳变点，因为这个原因，一个随后的非线性优化方法被引入来改善这个次优解的性质。通常将这个随后的非线性优化步叫做轨迹规划的后端，后端算法的目的就是使得轨迹更加的光滑和安全，满足车辆运动的内部约束和外部约束，并且轨迹规划前端生成的次优解被作为后端算法的初始解，通过后端算法的迭代，这个初始解逐渐变得完美。这个提出的框架，分别开发了 Python3 语言版本和 ROS 平台下 C++语言版本，仿真结果显示该框架能够实时地生成满足运动学、动力学约束地能够躲避障碍物的轨迹。

关键词： 轨迹规划，路径规划，速度规划，非线性优化，图搜索，自动驾驶

Research on trajectory planning of autonomous vehicles in structured environments

Abstract

This thesis presents a decoupled trajectory planning framework based on the integration of lattice searching and convex optimization for autonomous driving in structured environments. For a 3D trajectory planning problem with timestamps information, due to the presence of multiple kinds of constraints, the feasible domain is non-convex, so it is easy to fall into local optimum for trajectory planning. And the solution space of this problem is so enormous that it is difficult to identify an optimal solution in polynomial time. To address this non-convex problem, and to improve the convergence speed of an optimization process, the approach based on lattice searching is adopted in consideration of the ability to discretize driving environments and reduce the solution space. And the resulting path generated by lattice searching typically lies in the neighborhood of the global optimum. But this solution is neither spatiotemporally smooth nor globally optimal, so it is generally called the rough solution. For this reason, a subsequent nonlinear optimization process is introduced to refine the rough trajectory (combined by path and speed). In this framework, lattice search is used as the front-end algorithm for trajectory search, and nonlinear optimization is used as the back-end algorithm for trajectory generation. This strategy can not only accelerate the generation of trajectories, but also ensures that the trajectories have a better quality. The proposed framework is implemented and evaluated through simulations in various challenging scenarios in this thesis. The simulation results verify that the trajectory planner can generate high-quality trajectories, and the execution time is also acceptable. Moreover, this thesis analyzes and compares state-of-the-art planning algorithms, enumerates the advantages and disadvantages of various algorithms, and finally points out the scenarios in which various algorithms can be applied. This thesis has a detailed explanation of the planning field, and some innovative ideas can be found in this thesis.

Key Words: Trajectory planning, path planning, speed profile planning, lattice searching, Dijkstra's algorithm, nonlinear optimization, dynamic programming, autonomous driving.

目 录

致 谢	I
摘 要	III
Abstract	V
1 绪论	1
1.1 课题背景与研究意义	1
1.1.1 背景	1
1.1.2 研究意义	2
1.2 相关工作	3
1.2.1 图搜索	6
1.2.2 采样类算法	8
1.2.3 最优化类算法	14
1.2.4 采样与优化的结合	17
1.3 研究内容	22
1.4 章节安排	24
2 规划基础	26
2.1 道路模型	26
2.1.1 OpenDRIVE 道路模型	26
2.1.2 Lanelets 道路模型	28
2.1.3 改进的 OpenDRIVE 格式	30
2.2 笛卡尔坐标与 Frenet 坐标的相互投影	34
2.2.1 笛卡尔坐标映射到 Frenet 坐标	34
2.2.2 Frenet 坐标到笛卡尔坐标的转换	37
2.3 碰撞检测	39
2.3.1 基于代价地图的碰撞检测	39
2.3.2 基于车体圆的碰撞检测	41
2.4 运动基元	43
2.4.1 螺旋曲线	44
2.4.2 多项式曲线	48
3 路径规划	51
3.1 路径搜索	52
3.1.1 路径 lattice 图	52
3.1.2 路径 lattice 权重	55

3.1.3 Lattice 图的搜索算法	61
3.2 路径优化	63
3.2.1 目标函数	64
3.2.2 约束函数	65
3.2.3 非线性优化求解器	68
4 速度规划	71
4.1 速度曲线搜索	71
4.1.1 路程-速度空间采样	71
4.1.2 路程-时间空间采样	74
4.1.3 速度 lattice 权重设计与速度搜索	76
4.2 速度曲线优化	78
4.2.1 目标函数	79
4.2.2 约束函数	82
4.2.3 二次规划求解器	86
5 仿真与实验	88
5.1 案例研究	88
5.1.1 案例一：静态障碍物避免	90
5.1.2 案例二：自主换道场景	92
5.1.3 案例三：停车场景	94
5.1.4 案例四：超车场景	95
5.2 自动驾驶平台中的仿真测试	98
5.2.1 基于 ROS 的仿真测试	98
5.2.2 基于 autoware 的仿真测试	99
5.3 算法性能分析与比较	100
6 结论	103
6.1 全文总结	103
6.2 论文的贡献	103
6.3 研究展望	104
参考文献	107
作者简历及在学研究成果	112
独创性说明	113
关于论文使用授权的说明	113
学位论文数据集	115

1 绪论

1.1 课题背景与研究意义

1.1.1 背景

近年来，自动驾驶技术的研究一直是学术界，工业界和军方关注的焦点^[1,2]。2014 年，有 32675 次交通死亡事故，230 万人受伤以及 610 万次报告的碰撞事件，其中估计有 94% 是由驾驶员错误造成的，其中 31% 涉及合法醉酒司机，10% 来自分心驾驶员^[3,4]，而自动驾驶车辆能够获得比驾驶员更多的视野和环境信息，能够大量减少驾驶员失误和交通事故发生率，同时减轻驾驶员疲劳，减少燃料消耗并提高道路容量^[5,6]，所以自动驾驶技术的普及有望使得车辆运行比人类驾驶员驾驶获得更高的安全性及其它效益。美国国防先进研究项目局（Defense Advanced Research Projects Agency, DARPA）在 2004-2007 年举办了 3 届自动驾驶挑战赛，激发了该领域的研究兴趣。此外，Google 无人驾驶汽车已经在超过 1,600,000 英里的美国道路上进行了测试，特斯拉为其 MODEL S 汽车配备了自动驾驶系统。虽然谷歌和特斯拉针对 DARPA 挑战而开发的汽车已显示出良好的自动驾驶性能，但在技术和价格方面仍然存在自动驾驶汽车商业化的重大挑战^[7,8]，在实际应用中存在与自动驾驶系统相关的许多问题^[9]，Gartner 新技术成熟度曲线 2018 也强调了 L4/L5 级自动驾驶的发展动态，特别是 L4 级处于膨胀期与幻灭期的交点，预计十年以后才能实现真正的全自动驾驶^[10]。

智能化是汽车工业发展的必然趋势，而自动驾驶技术是汽车智能化的重要研究内容，也因此受到了相关国家政策的支持鼓励，在国务院发布的《中国制造 2025》^[11]中，明确要求“到 2025 年，掌握自动驾驶总体技术及各项关键技术”，我国工业和信息化部也印发了《促进新一代人工智能产业发展三年行动计划（2018-2020 年）》的通知，强调到 2020 年建立可靠、安全、实时性强的智能网联汽车智能化平台，形成平台相关标准，支撑高度自动驾驶（HA 级）^[12]。现代汽车配备各种传感器和电子系统，提供紧急协助（ABS，牵引力控制，稳定性控制），ADAS（自适应巡航，车道保持，侧风辅助，盲点检测）和导航辅助（行程规划，路线选择，交通规则更新），而在下一代智能车辆中将具有更强的功能，允许在各种驾驶场景中自主操作^[13]。

目前关于自动驾驶的研究涉及不同的领域，包括感知，定位、决策规划和控制^[14]。感知和定位是利用传感器技术和通信技术分析车辆所处的周围环

境信息^[15]，了解车辆自身状态，为底层的决策控制模块提供决策和控制数据，根据这些数据进行车辆的行为规划和轨迹规划^[16]，而控制的目标是确定转向系统，油门系统或制动系统的适当参数，以使车辆跟踪规划轨迹。规划是一个消耗内存，计算频次较高的模块，与感知、定位、控制和数据融合等模块并行运行，规划模块的输入和输出会和这些模块交互，所以可靠性高、适应性强的规划模块至关重要。

轨迹规划是自动驾驶系统的核心模块之一，涉及车辆的横向时空轨迹和纵向时空轨迹，同时避免与移动的环境参与者发生碰撞^[17]。由于不同的环境条件，比如交通规则、能见度、天气，以及交通参与者的多样性比如小型汽车、公共汽车、卡车等，而增加了超车决策规划的复杂性^[18]。目前国内外对轨迹规划进行了大量的研究，提出了大量的决策模型和规划算法，但自动驾驶车辆在城市道路环境下的自主行驶仍然存在较多问题^[19]，由于城市环境中结构化道路和交通规则的复杂性，交通参与者的多样性，感知信息的不确定性，以及环境信息的部分可观性，给行为决策和轨迹规划的研究带来了严峻的挑战，使得决策模型和规划算法并不能适用所有驾驶场景。

1.1.2 研究意义

自动驾驶场景中公认的比较难处理的场景包括交叉路口，转向掉头，超车等模式，传感器感知的误差、盲区以及延时的特点，还有环境的动态性和不确定性，使得决策与规划的研究变得尤为困难，也是攻克自动驾驶技术的一道坎，所以我们现在研究特定驾驶情景下的车辆自主行驶技术也变得尤为必要。决策规划模块是自动驾驶车辆的“大脑”，提高自动驾驶车辆在复杂、动态、不确定环境中的输出的轨迹性能，能够提高车辆的智能化水平，使得自动驾驶水平真正迈向 L4/L5 这个等级，具有十分重要的研究意义和极具潜力的应用前景。一般地，可以将行驶环境划分为自由空间和结构化空间，结构化空间或者结构化道路，是指具有显式约束的行驶环境，显式约束包括车道线、道路边缘、墙壁、走廊等硬约束，这种约束限制了车辆的形式范围，自由空间就是指不具有这种显式的、规则的约束，车辆只要不碰撞就可以自由行驶。对于这两种空间也是分别开发了一些算法。对于不同的行驶场景（如跟车，换道，超车，停车，缓慢行驶等场景），针对性地开发决策模块和规划模块，优点是在不同地场景调用不同的模块，互不干扰，同时这些模块能够独立并行开发，调整参数，对于简单少量的场景很有效；如果场景过多，使得算法结构复杂、冗余，特别是一些场景之间有覆盖，有交叉，这就需要额外的逻辑来处理场景之间的切换，基于场景的决策规划并不能覆盖所有场景，

使得算法太重决策，并没有很好的泛化能力，这在交通复杂的情况下是很不适用的。所以本文从结构化道路入手，着眼于轨迹规划问题的研究。

无人车辆在结构化道路中行驶是一种最常见的场景，然而，无人车辆的行驶速度较高，运行环境复杂，一些机器人领域较成熟的算法，如图搜索算法 A*，控制空间采样算法动态窗口法（Dynamic Window Approach, DWA），状态空间随机采样算法快速拓展随机树（RRT），等等已不再适合无人车的轨迹规划的性能要求。无人车的轨迹规划，通常需要处理大量的感知部分数据，预测模块对周围环境的预测，主车辆对自身当前状态的估计和预测，这些充满不确定性的数据我们需要在 100ms 内处理完毕并输出一条满足非完整性约束、运动学、动力学约束、时间和空间光滑的可行无碰撞的轨迹。一般地，在含时间戳信息的三维状态空间中规划，因为障碍物的存在，这个 3D 状态空间是非凸的，也就意味着轨迹规划算法极容易陷入局部最优，甚至根本不会输出一个可行的解。同时，轨迹规划的解空间是巨大的，也就意味着很难在多项式时间内找到一个最优解。本文将目前的研究难点给总结为以下几条：

- 1) 车辆行驶速度较高，必须要做速度规划，轨迹必须要严格时空光滑，算法必须保证实时性，鲁棒性。
- 2) 传感器感知的误差、盲区以及延时。
- 3) 环境的动态性，部分可观性，不确定性。
- 4) 结构化道路和交通规则的复杂性，需要作为约束加入到规划算法中，不像自由空间一样自由。
- 5) 交通参与者的多样性和交互性。
- 6) 无论是只包含位置的二维状态空间，还是包含有时间的三维状态空间，都是非凸的。
- 7) 解空间巨大，无论离散与否。
- 8) 自动驾驶涉及到的场景纷杂，算法是否具备泛化能力。

对于自动驾驶车辆轨迹规划的研究，特别是对于以上难点的攻克，是一个十分有意义的研究任务。

1.2 相关工作

在 2007 年 DARPA 举办的无人驾驶车辆挑战赛中，麻省理工学院的“Talos”无人车与康奈尔大学的“Skynet”无人车相碰撞，人们意识到行为决策和对其他交通参与者意图预测的重要性^[20]。在车辆自主行驶过程中给定了全局路径之后，行为决策层会基于交通规则和感知的环境信息，以及驾驶者偏好和对

其他交通参与者的预测，会做出安全评价最优或者时间最小的决策，转变到合适的驾驶模式，再去跟踪由运动规划层给出的轨迹。KATRAKAZAS^[16]将运动规划分为三个层次：路线规划，行为规划和轨迹规划。路线规划基于最短距离或最小预期行驶时间或其他标准来给定道路网络下的两个地理位置之间的最佳路线，现在的许多地图网站都具备路线规划功能，路线规划提供更具战略性，全局性和固定的路径信息，给出的路线能够行驶几分钟或几小时，然而通常不考虑更精细的时间尺度和道路交通信息流的交互，路线规划所需的信息通常是数字地图、GPS 和关于交通状况和道路建设的信息，通常，路线规划也叫做全局规划。

全局规划和局部规划的区分正是由于传感器感知范围的有限性、计算力的限制以及运动的不确定性，才将其分为两个层次。如果传感器能够感知无限大的范围，且计算机的计算力是无限的，并且物体的运动是确定性的，那么也就不必将规划分为两个层次。轨迹规划与路线规划相比，地理尺度更小，时间尺度更精细，规划时间通常以毫秒为单位，轨迹规划通常强调的是局部规划，而路径规划通常是全局规划，轨迹规划层需要传感器的实时环境信息，高频次地给出安全轨迹以执行驾驶任务并且会实时地更新规划轨迹，这一过程又叫滚动优化过程，给出的轨迹需要满足车辆行驶地平顺性和稳定性、交通规则约束等其他约束条件。行为规划也可以叫做行为决策，行为决策层动态地产生一系列车辆驾驶行为，如停车，转向掉头，超车换道，跟随和避让等等，快速响应道路交通环境和驾驶场景，时间尺度通常以秒为单位，以执行由路线规划层给定的总体驾驶任务，行为决策层是自动驾驶技术核心中的核心，是最接近人类智能的一个模块。PADEN^[14]提出当代自动驾驶系统中的决策通常分层为路线规划，行为决策，局部运动规划和反馈控制。然而这些层次的划分相当模糊，不同的文献中出现了不同版本的划分，这取决于各个研究机构自己开发的自动驾驶车辆的系统架构，但是在核心的算法层面各家又是大同小异。本节对行为决策和运动规划的研究现状进行总结，图 1-1 是自动驾驶车辆的技术框架图，图 1-2 是宝马集团开发的自动驾驶车辆驾驶策略和控制的系统结构^[21]。

对于车辆的轨迹规划，将车当作质点处理，规划从起点到终点的路径够不够完成任务呢？显然不够，车辆和障碍物都是有体积的，车辆运动过程中，需要进行碰撞检查，显然需要将其当作刚体处理。仅当作刚体处理够不够呢？也不够，刚体受到完整性约束，就是说可控自由度等于全部的自由度，每个自由度都可控，受到几个几何约束自由度就相应地减少几个，而车辆还和刚

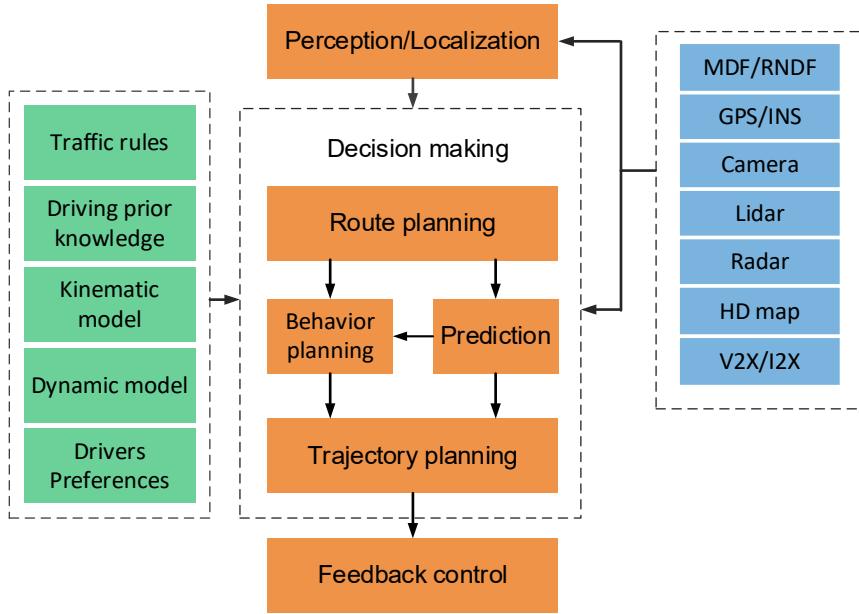


图 1-1 自动驾驶车辆技术框架

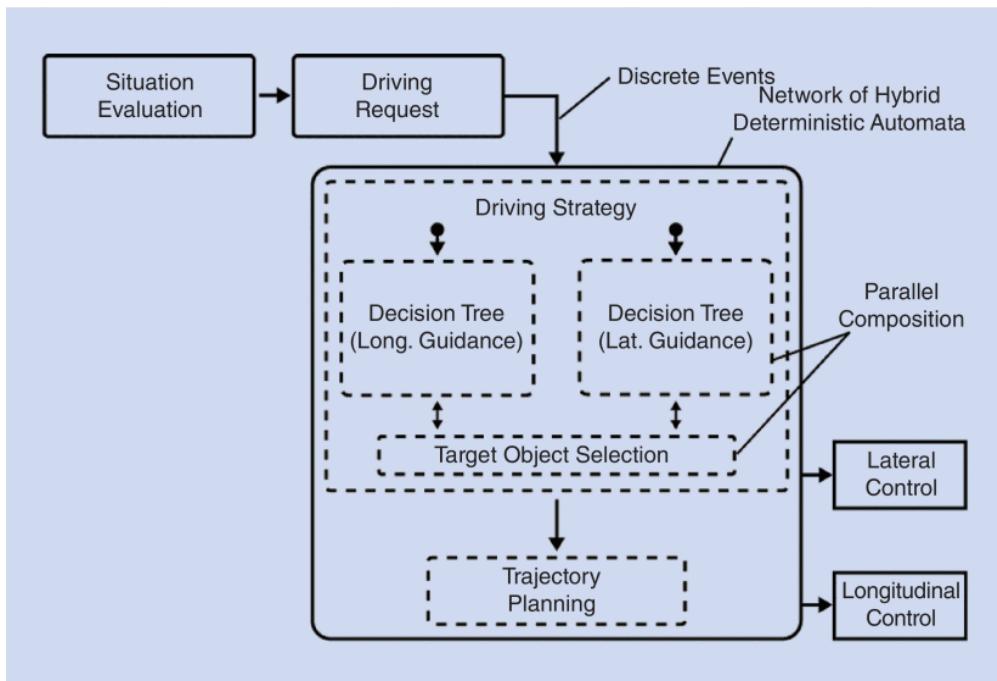


图 1-2 宝马集团开发的自动驾驶车辆驾驶策略和控制的系统结构

体不同，受到非完整性约束，就是说车辆受到了一个速度方向上的约束和车轮转角上的约束，但是车辆在二维平面空间内依然具有 4 个自由度，即笛卡尔坐标系方向的两个自由度、车体航向角、车轮转角。总之就是，车辆不能够像刚体一样任意旋转、平移，我们不能够将车辆完全按照刚体处理，神奇的是，虽然车辆受到速度方向上的约束，但是依然能够到达，二维空间内不

带约束的刚体能够到达的地方。那么考虑车辆的非完整性约束够了吗？依然不够，车辆的运动是与真实世界实时交互的，环境是动态的，可能在做碰撞检查时，前一秒会在某个地方发生碰撞，下一秒就不会了，也就是说此时我们从二维的欧式空间进入到了带时间向量的三维空间，在这个三维空间做的规划，就叫做轨迹规划。

从上世纪七十年代开始，人们就从事运动规划的研究，大多数的研究对象都是机器人领域^[22]，因为机器人运动的低速性，早期人们从事的基本都是路径规划的研究，但是车辆的速度较快，所以规划出的路径包含时间这一元素尤为必要，这时人们提出了轨迹规划的概念，进行运动规划的车辆和机器人之间的主要区别在于，前者涉及必须遵守交通规则的道路网络，而后者必须应对开放的环境，没有太多具体的规则要遵循，只要能到达最终目的地就行^[23]，除此之外，自动驾驶车辆轨迹规划的约束还包括安全性和舒适性等等。文献^[16]对路径规划和轨迹规划的定义作了阐述，路径是起始构型空间^[24]到终点构型空间的连续变换序列，路径规划就是发现这样一个满足避障条件、运动学约束和交通规则的序列。轨迹是由时间、位置、速度参数化了的状态空间序列，轨迹规划就是发现满足车辆运动学、动力学约束、避障条件、交通规则约束和舒适性约束的状态空间序列。由于轨迹是在路径的基础上加了时间约束，本文并不对路径规划算法和轨迹规划算法做出严格的区分，在路径的基础上明确考虑规划和执行时间^[25]，路径也就成了轨迹，可以说轨迹是路径的超集，所以两者之间的算法是相通的。本文将基于路径或轨迹的搜索方式或生成原理的不同将常见的算法分为四类，如图 1-19 所示，下面也会对这四类算法进行详细的介绍。

1.2.1 图搜索

在图搜索算法中，机器人或者车辆的运动的构型空间会近似为栅格（grid）或状态格(lattice)。Jonathan Bohren 使用 Dijkstra 算法根据交通状况自适应地计算航路点序列^[26]，A*算法是代表性的启发式图搜索算法，由于其距离函数的定义具有很好的启发性，因此该算法在栅格地图搜索中表现出的性能相当不错^[27]，但是在实际应用中，其总体性能仍局限于地图的分辨率，为此很多专家学者对其进行了各种改进。2008 年 DAARPA 比赛中来自斯坦福大学的 junior 无人车基于比赛环境的需要，采用了混合 A*算法，并最终取得了比赛的亚军^[1]，后来 Stentz 提出了一种可在动态环境中完成最短路径搜索的方法 D*算法^[28]，其他的改进 A*算法 Field D*^[29] ,Theta*^[30] , Anytime repairing A* (ARA*) and Anytime D*(AD*)^[31]。

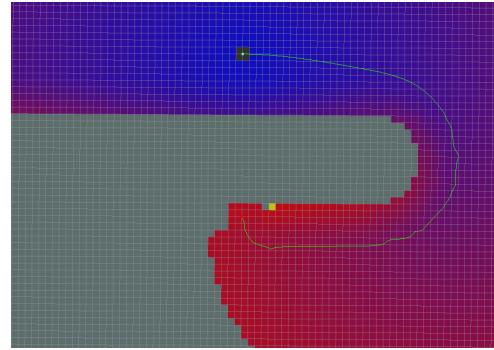


图 1-3 图搜索中的栅格图

后来考虑到路径的连续性和车辆的运动学约束，lattice 搜索算法被开发了出来，如图 1-4 所示^[32]，通过构造 lattice 空间，Howard and Kelly 得到了不平地形上的全局规划和局部规划有潜力的结果^[33]。然而，搜索算法的增量性质导致计算复杂性的指数增长问题，基于网格的方法在低速应用中的路径规划中表现良好，但它们不适合高速驾驶。通过 lattice 构造的行驶轨迹为特定类型的几何曲线，这种规划算法限制了车辆的运动潜力，同时也减弱算法泛化能力。在局部规划，也就是轨迹规划过程中，我们并不能确定一个临时的目标点具体在什么位置，为了有良好的避障能力，我们往往会构造多个运动基元，通过赋予每一个运动基元以代价值，然后通过图搜索算法，寻找出一条存在于 lattice 图中的最优解，如不存在一条可行的路径，算法就应当输出无有效解的提示。

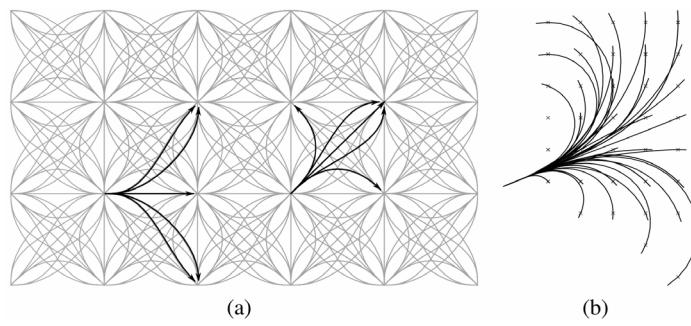


图 1-4 图搜索中的 lattice 图

栅格图和 lattice 图这两种图只是离散环境的方式不同，在构造图的过程中，栅格图的构造速度较快，而 lattice 图的较慢，为了增加算法的完备性，可以适当增加图的分辨率，然而，太高的分辨率又使得计算复杂度呈指数增长。为了在图中找到最优路径，常见的图搜索算法有 A*家族算法，Dijkstra 算法和动态规划算法，当然，在上面提到的两种图里需要使用不同的搜索算法，至于具体的原因和分析，后面的章节会有详细的分析。lattice 将状态空间离

散化，并保证离散化后的状态空间的连通性，状态格采样考虑运动学、动力学、曲率等约束，构建的边的代价可以任意配置，该代价不必与工作空间中的度量相关，最后以动态规划算法去计算以能量最小或者安全性要求最高的目的来计算代价函数。但是，lattice 是不完备的，可能在有解的情况下给不出解，而自动驾驶算法需要很好的稳定性，这种情况下会导致隐患。同时，降低了车辆的运动潜力，车辆的状态变换是连续的，但是 lattice 将连续的状态空间离散化了，也降低了算法的泛化能力。同时，在分辨率增加的情况下，维度诅咒问题，增加了计算开销。

1.2.2 采样类算法

为了解决高维空间中的规划问题，采样算法被开发了出来，确定性算法因为分辨率或者维度问题，导致规划的时间太长，这就是著名的“维度诅咒”。采样类算法是在状态空间中采样离散的状态，将代价值赋予状态或者连接状态的边，从而求得可行的轨迹。其实采样的目的，就是为了提高搜索效率，但是与图搜索类算法相比，算法的完备性有所损失。讲究采样的有效性、离散度，是否能增加算法的相对完备性。根据这个采样机制的不同，又有随机性采样和确定性采样。

(一) 随机性采样

顾名思义，随机采样就是在构型空间中用随机函数生成的状态，并用碰撞检测算法去除有碰撞风险的采样点。随机采样方法包括概率路图法（Probabilistic Roadmap method, PRM）以及快速搜索随机树算法（Rapidly-exploring random tree method, RRT）。PRM 算法首先在位形空间中进行充分离线采样，以此构造可反复使用的路径网络图，随后根据给定的起始位置和目标位置在路径网络图中通过查询（而非搜索）得到行驶路径。RRT 算法通过随机拓展无碰撞的子节点向前规划，直道拓展到目标位姿或达到每一步的最大拓展次数，输出可行解或无解的结论。

随机采样在自动驾驶里面，用的车辆平台并不是很多，早期 MIT 用过，而且可以玩的花样并不多，如今在特定的空间里面进行确定性采样并用代价函数来评价的规划算法已经盖过了随机采样算法的风头。因为随机采样生成的轨迹并不是最优的，并且总是频繁地改变路径的航向，所以很影响车辆跟踪行驶时候的平顺性。

文献^[34]和文献^[35]回顾了采样算法的应用和改进，Yoshiaki Kuwata 将 RRT 算法应用于路径的实时增量式搜索^[37]，然而采样算法产生的路径不是最优的，

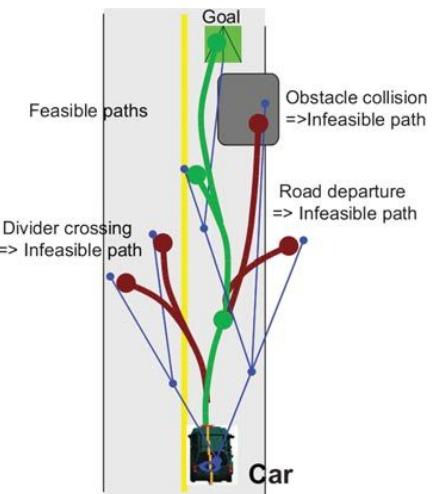
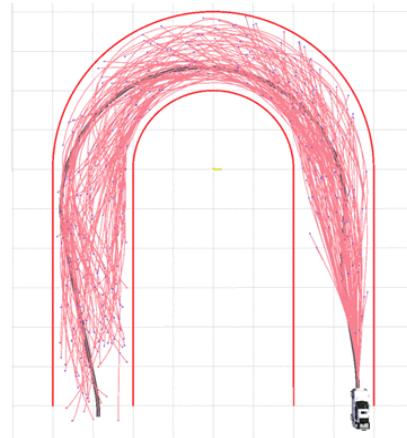


图 1-5 MIT 挑战赛中用的 RRT

图 1-6 RRT*路径搜索图^[36]

是不稳定的并且不是曲率连续的，后来 Karaman 等人开发了 RRT*算法虽然能够收敛于最优路径，但难以解决复杂的多约束和曲率难以控制的问题使得直接生成的轨迹对乘员的舒适性产生影响^[38]，且 RRT*在短时间内不一定能够收敛到最优解也使得不满足自动驾驶规划算法的实时性要求。同时，随机采样难以解决狭窄通道问题，RRT 对最近邻域度量的选择高度敏感，由于目标函数与实际的最小邻域度量完全不同，比如能量型函数，无法满足要求。

(二) 确定性采样

确定性采样就是在状态空间中，设置某种采样规则来进行采样，比如在行车道里面可以沿着道路中心线的法向量方向采样位姿，采样点的航向角、曲率和道路中心线的切向量有关。然后根据采样空间的不同，可以分为控制空间采样和状态空间采样。

[1] 控制空间采样

控制空间采样是基于参数化控制输入空间，生成有限的控制输入子集。一般地，基于车辆的运动学模型，有了车辆运动学微分方程，在速度空间，曲率空间和角速度空间里面均匀采样然后前向模拟，这些根据控制空间采样而生成的轨迹能够满足非完整性约束，显然是可行的。由于这个方法的简单和有效，在局部规划中被广泛使用。最早使用数值优化思想的是 Dieter Fox 提出的动态窗口法（Dynamic Window Approach, DWA）解决移动机器人的避障问题^[39]，DWA 是在控制空间的速度空间和角速度空间中均匀采样，然后向前模拟一段轨迹，根据代价函数选择最好的速度值和角速度值作为本次规划的结果，如图 1-8 所示。

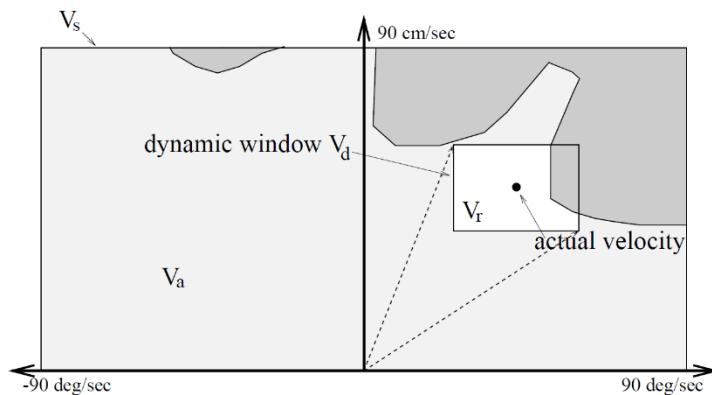


图 1-7 DWA 的采样空间 $C = [v, w]^T$

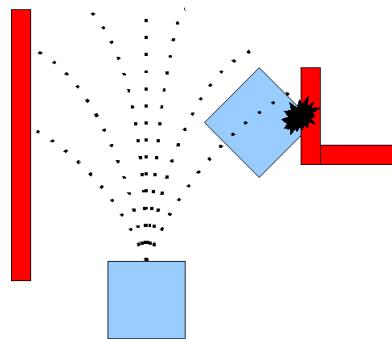


图 1-8 DWA 在控制空间中采样后对应的状态空间中的轨迹（虚线所示）

然而，控制空间采样的缺点在于，在采样过程中无法充分利用环境信息和参考路径的信息，并且，由于向前模拟运动的时域很短，造成了规划算法的短视，所以不适用于自动驾驶车辆中高速场景下的使用。除此之外，由于在控制空间中采样，生成的轨迹并不能均匀分布，对于结构化道路而言，很多采样轨迹是无效的，所以降低了采样的有效性和完备性。在采样过程中，

我们希望采样的搜索空间尽可能充分利用计算资源，轨迹之间的间距很重要，两条彼此接近的轨迹可能同时撞上障碍物而同时是无效的，所以需要设计采样的轨迹集尽可能地覆盖可达的解空间。

[2] 状态空间采样

因为控制空间采样难以对环境约束有较好地处理，并且离散度问题和相对完备性问题不好处理，所以状态空间采样开始流行起来。状态空间采样，就是在车辆的状态量空间采样一组终端状态，然后使用某种方式连接采样的状态终端，也就是连接状态的轨迹需要满足车辆运动学约束和终端约束。状态空间采样过程能够显式地考虑道路环境信息和参考线信息，也就是说，在结构化道路中，状态空间的采样可能更有效，减少了不必要的采样轨迹。

Howard 等人^[33,40]将轨迹生成问题建模成两点边值问题，然后将开发的这套技术叫做模型预测轨迹生成，主要是通过采样一组位姿 ($\mathbf{p} = [x, y, \theta, \kappa]^T$)，基于车辆的运动学模型，然后通过打靶法和牛顿法求解出螺旋曲线的参数，那么轨迹也就确定了。在这里，需要明确几个术语，位置 (position)：表示笛卡尔坐标，记为 $[x, y]$ ；位姿 (pose)：表示坐标与航向，记为 $[x, y, \theta]$ ，通常一般在二维平面内规划，所以有三个状态量，当在三维笛卡尔空间中时，就有了六个状态量；位姿 (posture)：表示二维平面内车辆的位姿 (pose) 和前轮转向的车轮转角，在这里车轮转角一般不用真实的转角表示，而是为了方便计算换算成曲率 (curvature) 表示，记为 $[x, y, \theta, \kappa]$ 。模型预测轨迹生成，最早可见于 2001 年、2003 年 Nagy 和 Kelly^[41,42]两人的工作，是基于最优控制，使用多项式螺旋曲线表示控制量，曲线的系数作为控制量的参数。不过 2007 年 Howard^[33]在研究火星车不平地形的运动规划时，在 kelly 的基础上又加上了速度控制（包括线速度和角速度），并且将参数向量通过约束进行了降维，这就是模型预测轨迹生成的由来。2011 年，Matthew^[43]也使用这一套算法构建 lattice，并将 lattice 加了时间维度，称作保形时空 lattice (conformal spatiotemporal lattice)。2012 年，xu wenda^[44]又在前人的基础上，使用四次螺旋曲线来连接采样状态，验证了四次螺旋曲线比三次的更平滑，特别是在连接点处能够保证曲率的连续。2017 年，国防科大的 li xiaohui^[45,46]也引入了这一套算法应用在自己的自动驾驶平台上。

模型预测轨迹生成，说白了是一套简单的最优控制算法。当是完全约束时，就是带约束的轨迹生成问题，将雅可比矩阵参与牛顿迭代即可；当是欠约束或者说约束不足时，就是带约束的最优化轨迹生成问题，构造哈密顿函数求变分即可。这套算法也有很大的缺点，计算结果对控制量参数化后的参数向量变化十分敏感，给以微小扰动就使结果发生剧烈变化，也就是对初值，

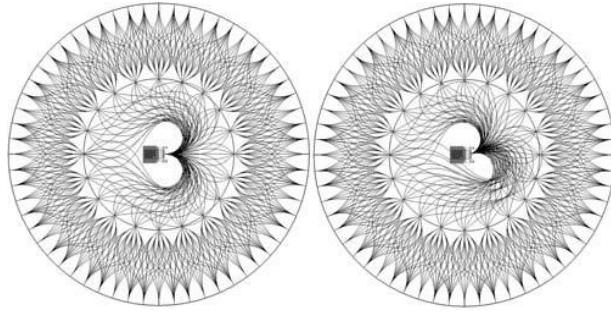


图 1-9 开放环境中的状态空间采样

且对差分过程中的小扰动差值十分敏感，该问题是一个病态问题。同时，又因为参数向量的数量级并不一致，所以迭代过程之中很容易引起“大数吃小数”问题，导致算法的不稳定性。

本质上，控制空间采样是一个求车辆运动的正解问题，状态空间采样是一个求车辆运动的逆解问题，求逆解是困难的，并且使用打靶法结合牛顿法求解也不一定稳定，所以也有了多项式曲线，贝塞尔曲线等曲线连接采样位姿^[16,23]，然而采样生成的曲线不一定满足非完整性约束，所以需要后续的曲率检查或者优化。为了保证生成的曲线满足非完整性约束，Dubins 曲线^[47]，Reeds-Shepp 曲线^[48]被开发了出来，尽管这两种曲线保证了路径的最短，也满足非完整性约束，但是却不够灵活，在结构化道路上行驶的车辆不可能会固定一个车轮转角值进行转向。

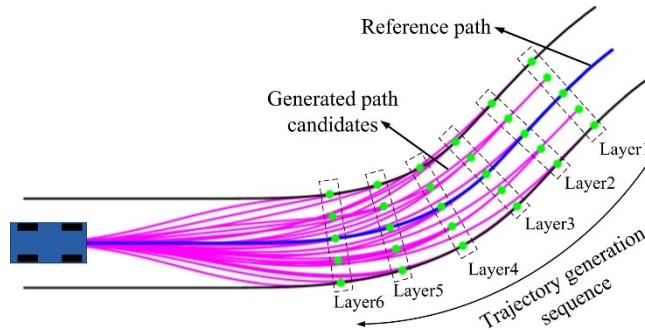


图 1-10 状态空间采样的模型预测轨迹生成算法

基于 Frenet 坐标系的采样也很流行。首先是基于预定义的路点建立车道的参考路径，然后沿着参考路径采样一系列候选路径，候选路径的生成方式比较广受欢迎的是 Frenet 三次多项式曲线^[49-51]，最后使用静态安全代价项，舒适度代价项和动态安全代价项的线性组合作为总的代价函数，来评估一条代价值最小的路径最为可行驶的路径。这些使用 Frenet 坐标系中的三次多项式曲线作为运动基元的算法，虽然计算复杂度很低，但是不够平滑，特别是不能保证曲率的连续性，另外该种类算法也比较短视，同时对速度规划也没

有很好的处理，所以对于低速机器人的使用是没有太大问题的，但对于自动驾驶车辆该类算法的特性则远远达不到使用需求。

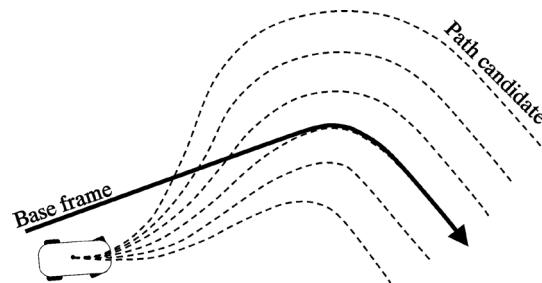


图 1-11 Frenet 坐标系的三次多项式曲线采样

Moritz Werling^[25,52]开发了一套流行的规划算法框架，通过在两个二维空间中采样，使用 jerk 最优的五次多项式连接采样状态，采样的两组轨迹，一组横向轨迹集，一组纵向轨迹集，每一条轨迹都有安全性和光滑性等代价值进行评价，这种将三维空间 ($\mathbf{s} = [s, \rho, t]^T$) 解耦的做法大大降低了计算复杂度，并且轨迹也足够光滑，无论是在空间上还是时间上。然而这个算法会导致车轮频繁地转向，并且在全局意义上并不是最优的。还有，这种完全依据 Frenet 坐标系的规划算法，在实际的应用中会出现路点的映射矛盾问题。

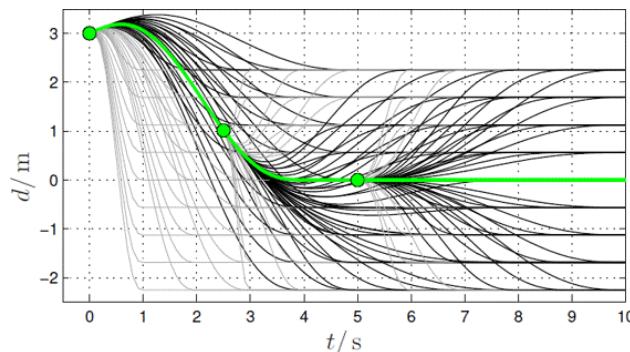


图 1-12 $d-t$ 状态空间采样的横向轨迹图

常见的采样状态空间包括欧氏空间，速度空间，加速度空间，里程-时间空间等等，采样之后，使用某种形式的边连接采样终端，这一过程也可叫做求运动学逆解的过程。这种对行驶环境的有效离散化，对解空间的离散化，对车辆运动状态的离散化，都能够大大降低计算复杂度，能够在极短的时间内找到一个可行解。

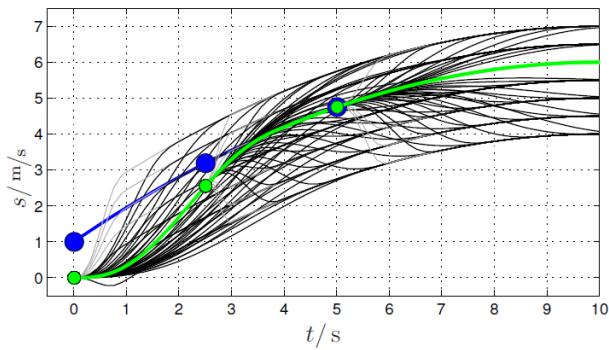
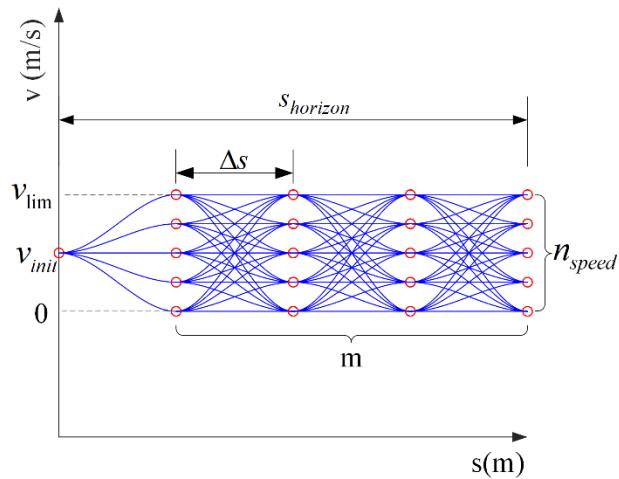
图 1-13 $s - t$ 状态空间采样的纵向轨迹图

图 1-14 速度空间采样

1.2.3 最优化类算法

随着自动驾驶技术的兴起，基于数值优化的方法近年来取得了巨大成功^[51]，旨在最小化或最大化受不同变量约束的目标函数。数值优化法的目标函数中，一般会考虑碰撞风险代价项，光滑性代价项，在约束函数中，一般会包含运动学约束，动力学约束，交通规则约束，避障约束。根据驾驶者偏好分别给不同的代价项赋予不同的权重来计算轨迹集的代价函数。根据状态空间是否解耦来区分，优化类方法可以分为直接优化法和解耦优化法。

(一) 直接优化法

直接优化法试图在笛卡尔坐标系或 Frenet 坐标系中通过非线性优化得到一条可行的轨迹。一般而言，此类方法在时空状态空间（spatiotemporal state space）中执行优化，这意味着这类方法可以直接处理时空障碍物（动态障碍物）。Ziegler 等人^[2,53]他们放弃组合式方法，转而采用一种不必离散工作空间

的连续的优化方法，因而离散方法带来的次优性、概率完备性或分辨率完备性这种采样类算法固有的缺陷也就去除了，此外，这个方法的计算复杂度并不随着状态空间维数的增加而成指数性增长。他们将一个非凸的非线性优化问题转换成一个凸的非线性优化问题，并精心设计了一个目标函数和约束函数。并且引入行驶通道，将障碍物处理成凸包，并凸化整个工作空间，使得算法在不给定好的初值的情况下也能够快速收敛到工作空间最优。他们引进了序列二次规划(sequential quadratic programming, SQP)算法来求解非线性优化问题。目前 SQP 通常被视作非线性优化的最新的技术^[53,54]，但是对于一个中等规模的优化问题，不给定一个好的初值作为热启动，算法的迭代次数并不少，所以并不能很快地收敛到最优解。

一个快速规划算法，凸可行域（convex feasible set, CFS）算法^[55-57]被提出，用来解决具有凸目标函数的，非凸约束的运动规划问题。CFS 算法的主要思想是，将一个原始问题，通过在非凸的可行域里获得凸的可行域，转换成一系列凸的子问题，然后迭代地求解凸的子问题直到收敛，迭代过程如图 1-15 所示。该算法和 SQP 的思想很相似，都是求解一系列凸子问题。不同之处在于获得凸子问题的方式，在 CFS 中，原始问题的几何结构会被充分考虑，这个策略使得算法比传统 SQP 或者内点法更快。在^[58,59]中，约束迭代线性二次形（constrained iterative LQR, CILQR）算法被提出，来解决具有非线性动力学和广义形式约束的最优控制问题。作者证明 CILQR 的计算复杂度比标准 SQP 的更小。但是这些方法的缺陷之处就是容易陷入一个局部最优，整个规划缺乏全局性的考虑。

轨迹是时间域和空间域的一到一的映射组成的，所以上述这些优化类方法通常以相等的时间间隔采样一组时间戳，然后与这些时间戳具有映射关系的空间坐标就是优化的决策变量。根据性能指标和约束函数，一个合适的算法被选择出来去优化这些决策变量。然而，在复杂环境中，无论是时间域上还是空间域上，状态空间都是高度非凸的，并且可行域的凸化并不是那么简单，所以优化类算法的实时性和最优化需要被重点关注。

(二)解耦优化法

解耦优化法通常是对一个三维优化问题 ($\mathbf{X}=[x, y, t]^T$)，给解耦成路径-速度分别优化问题。对于路径优化，Dolgov^[60]使用梯度下降法来优化路径，Xu^[44]使用单纯形法来优化路径，Fan^[61]使用二次规划算法来优化路径，这些算法都是接受了一条粗糙的解作为初值来进入优化过程，优化的目的是使得生成的路径更加的平滑。对于速度优化，通常是基于 ST 图的优化，将主车辆和障碍物信息都投影到 ST 图中。然后因为纵向速度曲线是时间戳与纵向里程的一

到一映射而形成的，所以目前主要存在两种优化策略。第一种优化策略^[62,63]

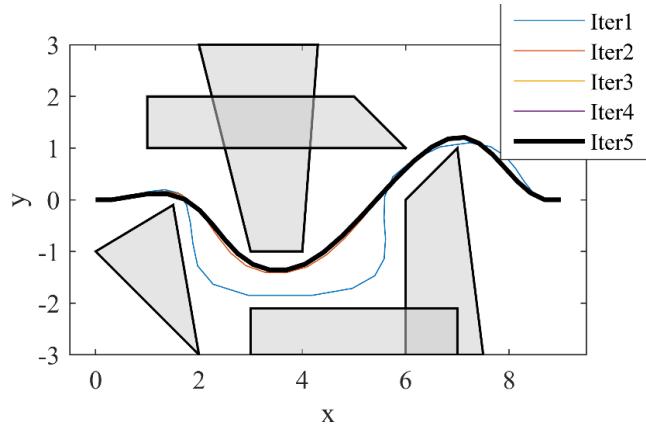


图 1-15 轨迹的直接优化过程

是将与时间戳相映射的纵向里程 $s = [s_1, s_2, \dots, s_n]^T$ 作为决策变量，第二种优化策略^[64]是将与纵向里程相映射的时间戳 $t = [t_1, t_2, \dots, t_n]^T$ 作为优化的决策变量，如图 1-16 所示。

解耦优化法与直接优化法相比，具有更快的优化速度，因为通过解耦将一个中等规模的优化问题变成两个稍小的中等规模的优化问题。然而，通过解耦，引入了次优性。因为直接法能够直接处理动态障碍物，而解耦法，在路径优化时忽略了时间信息，这使得优化的结果可能偏离了期望的结果，在速度优化时，会忽略一些空间上的信息。解耦法在优化时忽略了一些信息，从而导致优化结果偏离了最优解，甚至无效。为了处理这个问题，后来出现 EM 算法，这个算法会在下一节介绍。

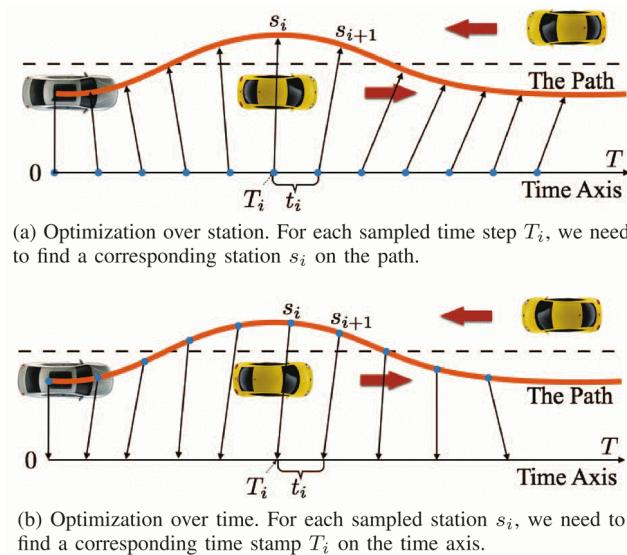


图 1-16 速度优化策略

1.2.4 采样与优化的结合

相对而言，采样类算法具有更高的计算效率，并且不容易陷入局部最优，同时灵活性也比较好，能够适应多种场景。然而，采样类算法限制了车轮的运动潜力，因为只采样几组位姿很明显会降低车辆的运动能力，同时算法的完备性也是一个问题，再有，算法生成的轨迹也不够光滑。而对于优化类的算法来说，算法容易陷入局部最优，计算复杂度较高，灵活性较差，而优点就是没有限制车辆的运动潜力，完备性也较好，并且生成的轨迹是足够光滑的。经过这种对比，发现采样类算法和优化类算法的很多特性是互补的关系，所以联合采样类算法和优化类算法的策略被提了出来，自从无人驾驶变得火热以来，这种组合类的算法近年来被大量提出，经过实际的车辆测试也取得了很好的效果。

Wenda Xu^[44]开发了一个有效的实时自动驾驶运动规划器，规划器首先离散化状态空间，然后基于代价函数搜索出最优的轨迹，接着对于路径和速度进行迭代优化。后优化过程具有较低的计算复杂度且在几次迭代后就能够收敛到高质量的解。与不带优化的规划器相比，Wenda Xu 提出的规划器能够降低 52% 的规划时间并改善轨迹质量，Xu 提出的算法框架如图 1-17 所示。

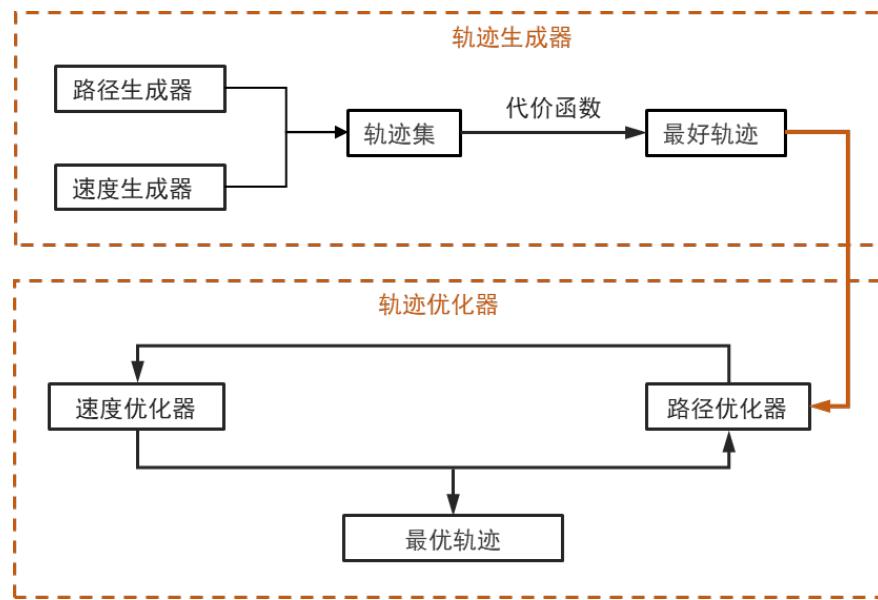


图 1-17 Wenda Xu 提出的两阶段轨迹规划框架

樊浩阳^[61]基于百度 Apollo 自动驾驶平台，开发了 EM 运动规划器。该规划器考虑安全性，舒适度和可拓展性，致力于 L4 级的自动驾驶应用。该规划器的顶层是一个多车道策略，通过并行计算不同车道的轨迹来负责换道任务；在车道级的轨迹生成部分，该规划器基于 Frenet 坐标系迭代地计算路径和速

度；对于路径优化和速度优化，组合动态规划和样条二次规划来建立可调的，可拓展的框架，去同时处理交通规则，障碍物决策和平滑性问题。该规划期得到了高速和城市低速场景的验证，算法框架如图 1-18 所示。

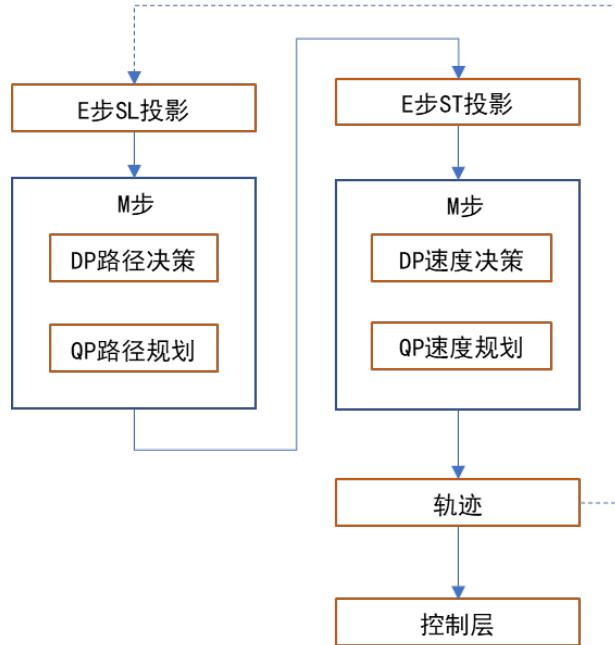


图 1-18 Apollo EM 运动规划器

Lim^[65]等人开发了一个适用于城市环境中自动驾驶的联合采样和优化的分层轨迹规划器。为了适应复杂的行驶环境，上层的决策层使用环境模型搜索出决定驾驶行为的微尺度轨迹，决策层推理出合理的行为并推送给轨迹规划器。为了规划行为，采样类方法离散化行驶环境和简化搜索区域，然后底层轨迹规划器会基于上层规划出的行为，根据车辆动力学约束进行微尺度轨迹的优化。因此底层规划器的规划空间是有限的，改善了最优化方法找到最好轨迹的效率。根据文章的思想，最优化方法被使用，是由于其对行驶环境建出连续的数学模型和能够以较低的计算复杂度收敛到最优值的特点。具体地，他们在 Frenet 坐标系和 ST 坐标系中分别采样一些状态，然后使用动态规划搜索出一条粗糙的路径，使用 Hybrid A* 算法搜索出粗糙的速度曲线，然后使用 SQP 算法去优化粗糙的轨迹，和百度 Apollo EM 算法不同，Lim 等人是直接优化轨迹，而不是像 EM 一样分别优化路径和速度。

上面这些解耦类的算法因为对解空间的离散化和动态规划算法的使用，能够使得最终解跳出驻点，随后的优化过程使得解是全局最优的，并且也让生成的轨迹更加的平滑。然而，这些算法在优化过程中忽略了非完整性约束，也就使得生成的轨迹不一定是可行驶的。

为了清晰地对比各类算法的优缺点，本文制作了两个表格显示这些算法

的特性，如表 1-1、表 1-2 所示。我们比较的规划算法的特性包括：最优化，移动能力，完备性，空间光滑性，时间光滑性，实时性，车辆的行驶环境，灵活性，非完整性约束和状态空间是离散的还是连续的。最优化表示规划算法在非凸的解空间中找到最优解的能力，通常对于最优化类的算法而言生成的解往往会陷入驻点，所以是局部最优的，而对于采样类算法来说，只要分辨率合适，生成的解往往位于最优解的领域，所以采样类算法往往是次优的。移动能力表示规划算法是否限制了车辆的运动潜力，一些算法^[32,33,40,46]会指定车辆的运动学和动力学参数，例如固定的车轮转角、速度值和加速度值，显然这种预定义某种移动规则的策略会严重降低车辆的运动潜力，所以移动能力也是规划算法的一项考量指标。完备性^[24,66]表示规划算法在解空间中找到一个可行解的能力，如果存在解，需要在有限的时间内给出可行解，如果不存在解，就也需要在有限的时间内给出无解的结论。像随机采样类算法 RRT，因为采样的随机性，在即使有解的情况下也可能找不到可行解，所以该算法是概率完备的。对于确定性采样类算法而言，找到解的可能性和算法的分辨率有关，即分辨率越高，找到解的可能性越大，并且解的质量也越高，反之，分辨率越低则找到解的概率越低，所以常将此类算法叫做分辨率完备^[40]的算法。对于最优化类算法而言，只要能够跳出障碍物驻点，该算法就是完备的，也就是说，因为障碍物的非凸性，我们往往需要精心处理障碍物，才能使得最优化类的算法找到可行解。空间光滑性表示规划算法生成路径的航向角曲线和曲率曲线是否是连续而光滑的，如果解是在曲率曲线上是光滑的，我们就说该算法的空间光滑性比较好。时间光滑性表征一个规划算法生成轨迹的速度曲线，加速度曲线和加加速度曲线是否是连续而光滑的。车辆的行驶环境主要可以分为结构化空间 (structured space, S-space) 和自由空间 (open space, O-space)，不同的算法在不同的行驶环境中有着不一样的表现。灵活性表征一个算法适应不同驾驶场景的能力，比如跟车场景，停车场景和换道场景。考虑到优化过程中的计算复杂度，一些算法^[44,61,65]在轨迹优化过程中忽视了非完整性约束，来改善轨迹迭代过程中的收敛速度。如果一个机器人的可控自由度等于总共自由度，则它受完整性约束。对于角轮 (castor wheels) 或者是全向轮 (Omni-wheels) 的机器人，是完整 (holonomic) 的，因为它能够朝各个方向移动，并且机器人总自由度与可控自由度是相等的。非完整性约束是指，可控自由度小于车辆的总自由度，通常二维平面内车辆的自由度为 4，包括 $[x, y]$ ，车身朝向与车轮转角，但是车辆这种结构，其可控的自由度为 2，包括油门/刹车和方向盘转角，使得它难以满足任何方向的行驶（除非车辆发生打滑或侧滑），所以是非完整性约束^[16,66]。本质上，非完整性约束

是由车辆的阿克曼转向机构带来的，所以我们也可以转换成曲率约束，因为阿克曼转角的存在，车辆存在最小转弯半径，所以曲率约束在优化过程中是不可忽略的。最后一些规划算法的规划空间是离散的，而一些算法的规划空间是连续的，离散是为了更快地找出接近全局最优的解，状态空间的连续是为了保证解的质量。

表 1-1 规划算法的特征比较

	Ziegler ^[2] [55]	CFS ^[56] , [55]	CILQR ^[58] , [59]	Howard 等 ^[40]	Werlin g 等 ^[52]	Li 等 ^[46]
最优化	局部最优	局部最优	局部最优	次优	次优	次优
移动性	好	好	好	差	差	差
完备性	√	√	√	分辨率完备	分辨率完备	分辨率完备
空间光滑性	√	√	√	√	√	√
时间光滑性	√	√	√	×	√	×
实时性	√	√	√	√	√	√
行驶环境	S-space, O-space	S-space, O-space	S-space, O- space	S-space, O-space	S- space	S-space, O- space
灵活性	×	×	×	√	√	√
曲率约束	√	√	√	√	√	√
状态空间	连续	连续	连续	离散	离散	离散
所属算法分 类	最优化类	最优化类	最优化类	采样类	采样类	采样类

表 1-2 规划算法的特征比较

	Lim 等 ^[65]	Xu 等 ^[44]	Fan 等 ^[61]	Dolgov	时 空 lattice ^[32] , [43]	Zhan 等 ^[62]
最优化	最优	最优	最优	最优	次优	最优
移动性	好	好	好	差	差	较好
完备性	√	√	√	分辨率完备	分辨率完备	√
空间光滑性	√	√	√	√	√	√
时间光滑性	√	√	√	×	×	√

性						
实时性	√	√	√	√	×	√
行驶环境	S-space	S-space, O-space	S-space	O-space	S-space, O-space	S-space
灵活性	√	√	√	×	√	√
曲率约束	×	×	×	×	√	×
状态空间	离散和连 续	离散和连 续	离散和连 续	离散	离散	离散 和 连 续
所属算法 分类	联合采样 与优化	联合采样 与优化	联合采样 与优化	图 搜索	图搜索类 类	联合采样 与优化

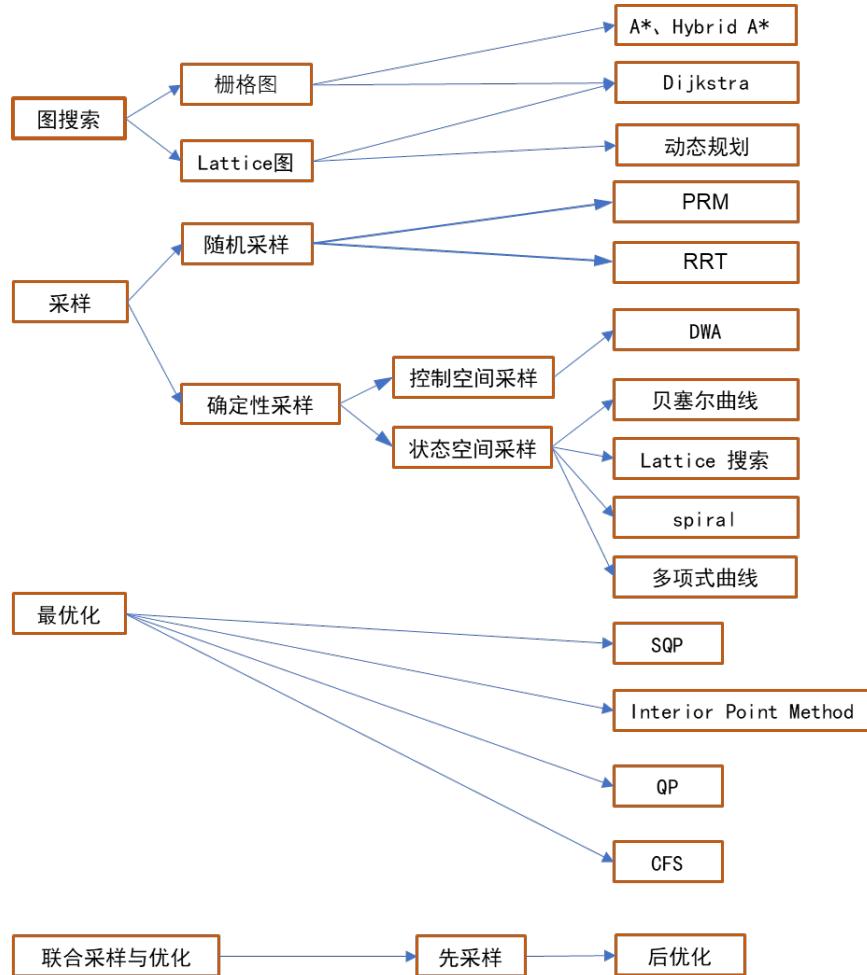


图 1-19 轨迹规划算法的流派

总之，现有的应用在自动驾驶领域的规划算法基本上都是在最优性和规划效率之间寻找一个平衡，为了实时性的考虑工程师也宁愿得到一个次优的轨迹，但在学术研究中，我们尽量希望得到最优的解，所以在研究最优解的

同时也在着力于解决实时性问题。除了环境不确定性和多约束之外，规划问题还有一个性质导致它的复杂性，那就是非凸性，不像凸优化中局部最优点一定是全局最优点，规划问题是一个非凸优化问题，我们并不能保证轨迹生成的可行性和唯一性，所以我们往往会在轨迹评价的代价函数里面加入一些驾驶者偏好约束，如算法是激进的还是保守的，轨迹规划模块的可调性也是自动驾驶技术研究中不可忽视的一点。现在流行的是，人们更喜欢使用采样类方法来离散化解空间，使用优化类方法来生成连续而平滑的解。对于只使用优化类的规划算法来说，能够在时空空间中生成一个可行解，同时能够考虑静态障碍物和动态障碍物，然而，如果不给定一个好的初值，或者约束函数没有得到很好的处理，那么算法就很容易陷入局部最优，并且收敛速度也不令人满意。现在的结论是，目前还没有一套统一的算法，在满足各种约束和需求的同时，能够同时适应结构化环境和自由空间，和能够同时适应多种行驶场景。短视的，局部最优的，不完备的，不灵活的缺点或多或少地分布于当前地一些算法当中，所以开发一个通用的，实时的，灵活的，可调的，最优的算法是一个重要的研究课题。

1.3 研究内容

本文的研究内容着眼于自动驾驶技术在民用方向的发展，旨在提高自动驾驶车辆的自主通行能力，以无人乘用车为研究对象，解决自动驾驶车辆在结构道路上行驶的实时轨迹规划问题，为智能车辆迈向 L4/L5 级别进行一些积极有益的探索，为我国自动驾驶技术的落地提供现实的技术储备和支撑。车辆自主行驶过程中，行为层会做出是否应该超车，何时超车，怎么超车的决策，但是本文主要关注轨迹规划的研究，也就是说上层的决策已经确定，我们只需专注于轨迹规划的算法开发。由于驾驶环境的动态性质，车辆行驶不是一个标准化的操作，在实际场景中每一个工作都是独特的，这种独特性源于周围车辆数量的变化，某一个动作的持续时间，车辆之间的相对速度，车辆之间的距离等，而我们的轨迹规划研究会适应这种独特性。本文提出了一种解耦的轨迹规划框架，如图 1-20 所示，并使用 Python 语言和 ROS 平台对这个算法框架进行了仿真验证，使用了模型预测控制路径跟踪控制器对算法框架输出的路径进行了跟踪，取得了良好的效果。

本文的研究工作主要是：对于结构化道路，本文参考了 OPENDRIVE 的地图处理格式和文献^[1-3]的工作，使用了三次多项式曲线来连接路点，然后将路点曲线弧长参数化，具体工作是在路点曲线上等间隔采样，然后使用弧长参数对路点曲线进行参数化，形成微小分段的弧长参数化的三次多项式组成

的路点曲线，通常将这条曲线叫做道路的参考线。然后，基于这条参考线，

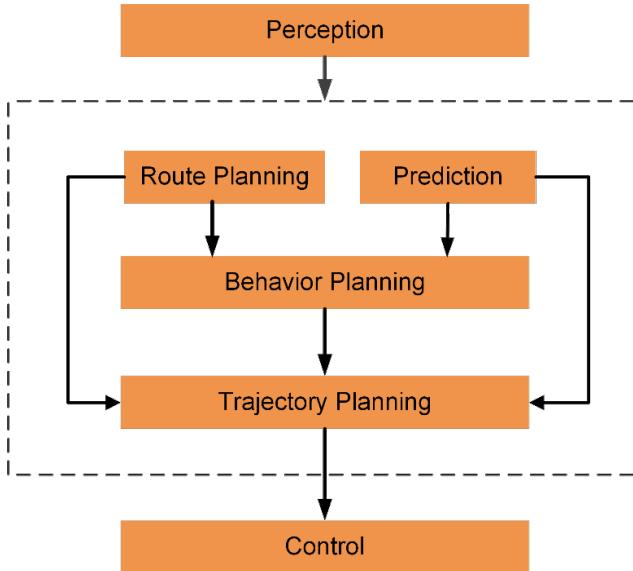


图 1-20 自动驾驶框架

我们生成驾驶通道的边界，也就是道路的边界，和生成车道的参考线。给出了结构化环境的数学表达和程序实现，为后续的规划工作创造了仿真环境。

本文从构型空间是否连续的角度，充分对比了采样类算法和优化类算法关于轨迹规划输出结果的性能，从轨迹规划是否解耦的角度，充分对比了解耦类算法和降维类算法的性能，并对主要的算法做了复现的工作，这对于非凸可行域中快速找到可行解有重要意义。

对于结构化道路而言，在 Frenet 坐标系中规划能够带来很多计算上的方便。本文基于道路的参考线，推导了笛卡尔坐标系和 Frenet 坐标系的映射方程，能够做到将车辆的状态信息和障碍物的运动信息投影到 Frenet 坐标系。后续的所有的工作都是在 Frenet 坐标中进行的，对于提高算法的运行效率有重要意义。

本文充分考虑到最优性和规划效率之间的平衡，所以放弃 A*这种能够搜索到最优解但生成的路径并不平滑且有维度爆炸的问题，和 RRT 这种虽然满足实时性但是搜索结果不稳定的算法，根据前人的工作，提出了在 Frenet 坐标系中采样并使用三次样条曲线连接位姿，最终构建一个路径 lattice，并使用 Dijkstra 算法搜索出一条粗糙路径的算法思想。这种解耦的思想是算法效率得到提高的重要基础。

本文结合了采样类算法和优化类算法的优点，首先将一个三维规划问题转换为两个二维规划问题，分为构建了路径 lattice 和速度 lattice，并根据路径和速度的平滑性，碰撞风险和车辆的运动学约束设计了图中边的权重，并

使用穷举类算法搜索出一条粗糙的路径和速度曲线。本文将路径优化问题建模为一个具有二次目标函数和非线性约束的中等规模的非线性规划问题，并使用 CASADI 优化库的 IPOPT 里面的内点法解这个 KKT 条件。将速度优化问题建模为一个中等规模的二次规划问题，并使用 CVXOPT 的内点法解这个二次规划问题。结合两类算法优点的思想是本文提出的规划框架的基石，结合前人的相关工作，这种思想可以在大量的解空间中快速找到一条较优轨迹。然后本文对无人车轨迹规划算法的性能要求总结为：

- [1] 满足实时性，达到 10Hz 的运行频率。这里需要考虑怎么去降低规划算法的计算复杂度，能够在非凸的、巨大的解空间中找到一个还不错的解；
- [2] 时空光滑性。即在空间上，路径的航向角和曲率是连续变化的；在时间上，速度、加速度和加加速度是连续变化的；
- [3] 适用的场景要广，高速场景、低速场景，换道、跟车、超车、停车等场景需要尽量使用一个规划算法，如果对各自的场景都开发一个对应的算法会极大地增加整体程序的复杂度，需要具备良好的泛化能力；
- [4] 充分考虑不确定性，能够对感知误差和物体运动的不确定性留有一定弹性，即规划一般需要是保守的，提高安全性能。
- [5] 怎么在三维空间中处理非凸的时空障碍物，怎么让解收敛到全局最优。

1.4 章节安排

第一章对课题背景和研究意义进行了阐述，并且对目前的相关工作进行了详细的分析与整理，对现有的轨迹规划方法进行了分类，并总结了最新算法各自的优缺点，并在最后概括了本文的研究内容。第二章详细介绍了规划领域使用到了基础知识，包括使用的地图格式，对参考线的处理，并给出了目前最新算法中常常使用到的 Frenet 坐标与笛卡尔坐标的转换方程，并对碰撞检查和运动基元有一个统筹性的整理和完备的说明。第三章介绍了路径规划中的前端和后端算法，包括前端算法对路径的搜索和后端算法对路径的优化，对路径权重的设计、图搜索算法的选择、路径优化的建模、求解器的选择都有一个比较详细的介绍。第四章介绍了速度规划中的前端和后端算法，分别是速度的搜索和优化，该章节总结并给出了两种速度搜索的方法，并给出了速度优化的详细建模方法。第五章是对本文提出的算法框架的仿真验证，分别展示了 Python 语言版本的仿真结果和 ROS 平台下的仿真结果，结果显示本文的轨迹规划方法完全能够在满足实时性约束、运动学约束、动力学约束、避障约束以及道路边界约束的情况下，输出一条理想的轨迹，并对

比了目前主流算法的时间性能。第六章是对全文的总结，总结了本文的贡献，提出了研究展望。

2 规划基础

2.1 道路模型

2.1.1 OpenDRIVE 道路模型

对于设计运动基元和建模道路的几何形状来说，参数化曲线被频繁地使用。结构化道路的规划和自由空间的规划是不同的，在自由空间里一般满足避障要求就可以了，而对于结构化道路的规划，往往需要满足车道线的约束和道路边界的约束，而又因为道路是弯弯曲曲的，为了自动驾驶中运动规划算法的考虑，现在主要开发了两种的道路模型。这一节首先介绍 OpenDRIVE^[67]道路模型。OpenDRIVE 是一个开源的文件格式，用于道路网络的逻辑描述。OpenDRIVE 道路是基于参考线形成的，如图 2-1 所示，参考线是一条没有宽度的，为了适应道路形状的曲线。沿着这条参考线，每一个点处都有一个 Frenet 标架，标架的两个轴分别是点在参考线的切向方向和法向方向，沿着参考线的纵向方向，会积累里程数据，一般记为 s ，点的法向方向的距离也表示与参考线的偏移量，在这里可以记为 t ，然后参考线的左侧偏移量的值为正，右侧偏移量的值为负值。

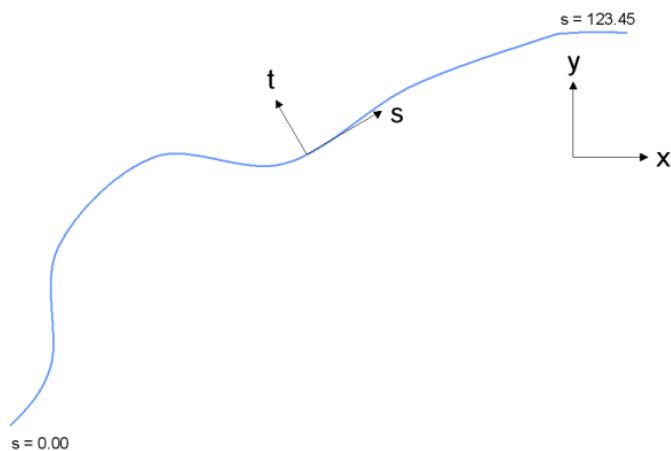


图 2-1 OpenDRIVE 道路参考线

在 OpenDRIVE 中路网结构中的道路，道路由三部分组成，参考线，车道，车道，和车道其他特征，比如限速特征。对于道路而言，参考线是最重要的特征，包含了道路的位置，朝向和曲率，然后道路的其他特征，大部分都是基于参考线来构建的，如图 2-2、图 2-3 所示，基于道路参考线建立了回环道路和十字路口。

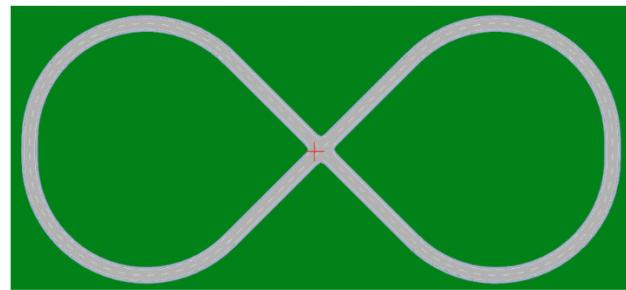


图 2-2 OpenDRIVE 格式的 8 字形道路图



图 2-3 OpenDRIVE 道路格式的十字路口图

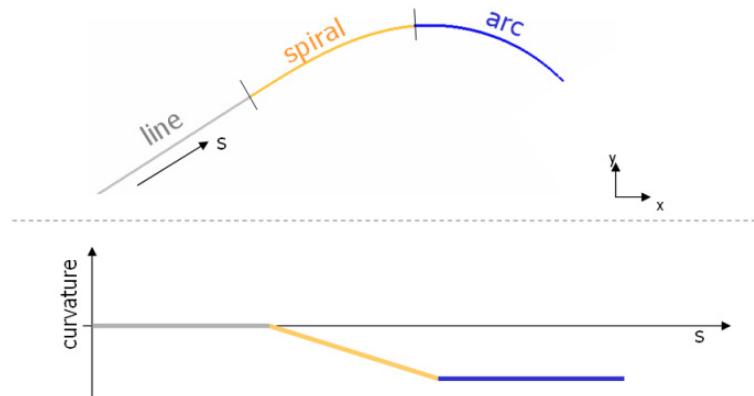


图 2-4 OpenDRIVE 道路参考线的基本曲线构成

对于参考线的构造，OpenDRIVE 提供了三种曲线，分别是直线，螺旋曲线和圆弧，如图 2-4 所示。然后以参考线为基准线来构建道路中的车道，对于每一段道路基元，在参考线左侧的车道编号为正，右侧车道编号为负，再根据车道的宽度，以及车道线的虚实，来构建多条车道。一般地，车道线都是基于与参考线的偏移而制作的，如图 2-5 所示。最后为了构建整个道路网

络, OpenDRIVE 将多个道路片段编号, 通过指针记录道路片段的前驱与后继, 对于缺少车道线的道路, 为了方便车辆决策和规划, 会建立虚拟参考线, 作为算法的指引线, 如图 2-6 所示。

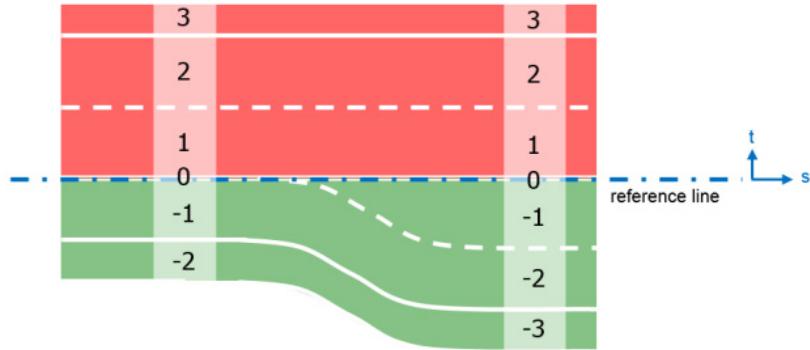


图 2-5 OpenDRIVE 道路中车道的建立

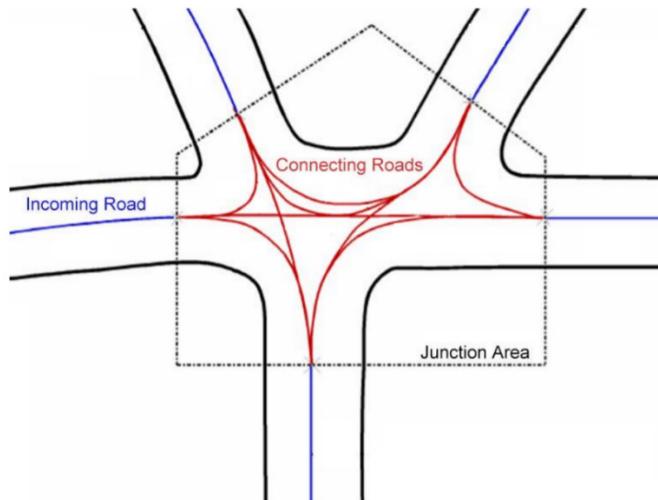


图 2-6 OpenDRIVE 格式中不含车道线的道路区域的处理方式图

2.1.2 Lanelets 道路模型

Bender 等^[68]为自动驾驶领域提出了详细的地图格式 lanelets, lanelets 应用于 2013 年自动驾驶汽车伯莎, 经过他们的实验验证, 该地图能够在结构化道路的行驶中取得良好的效果。

Lanelets 也是一种拓扑形式的自动驾驶地图, 它通过线和点来构建整个道路网络, 通过线的颜色, 可以区分道路的行驶方向, 左侧车道线为红色, 右侧车道线为绿色, 通过对线的使用也能够区分道路中的不同车道。对于整个道路网络, 可以叫做 lanelets, 对于道路的基元, 比如一个单一的车道,

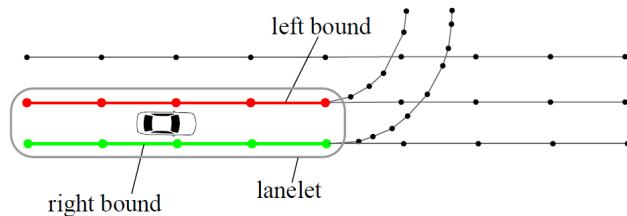


图 2-7 lanelets 道路模型。高亮的是一个单独的 lanelet，行驶方向由车道边界指定，该 lanelet 的行驶方向是自左至右

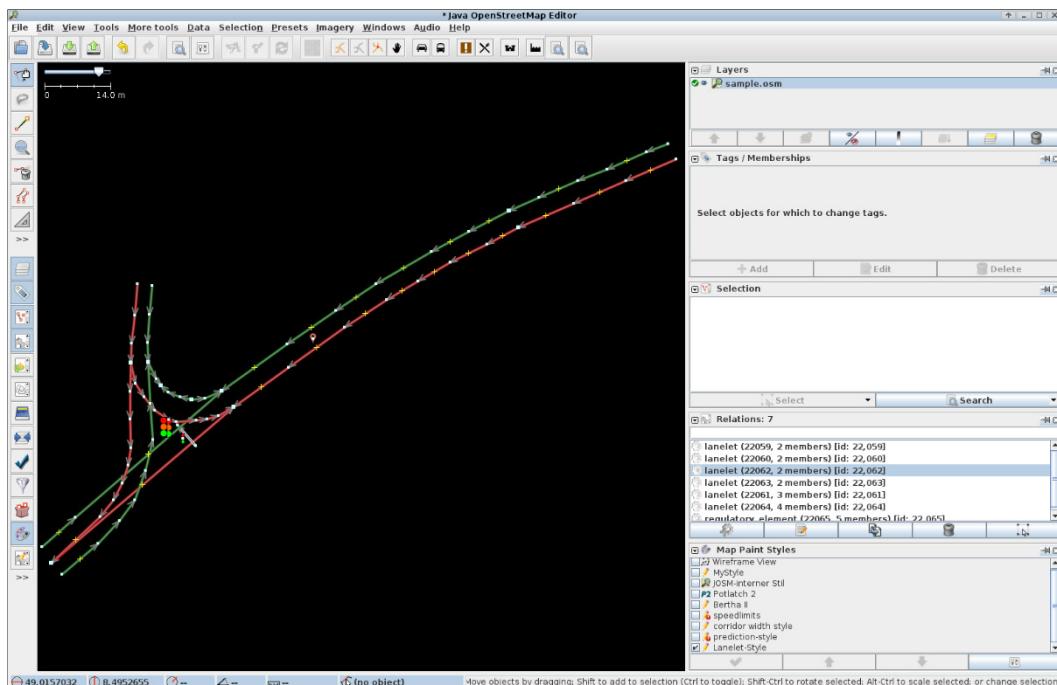


图 2-8 lanelets 的构建工具

可以叫做 lanelet，如图 2-7 所示。然后对于岔路口的地方，通过指针来保证每一个 lanelet 的连接。整个 lanelets 的构建思想是，基于离散的路点，使用线段连接路点构建整个网络，这种策略能够基本表示地图的大致形状，预处理起来速度很快，在规划算法的使用中也能够降低道路约束的复杂度，但是缺乏道路曲率信息，并且精度比不上 OpenDRIVE 地图，如图 2-8 所示，显示的是 lanelets 的构建工具。

lanelets 具有用于路由，运动计划和其他车辆预测的接口。但是，该格式具有某些缺点，因为它设计用于特定的，已知的路线，例如仅在预定义的点才可以更改车道，因此也不可能超车，此外，不支持许多特殊操作。2018 年，Fabian Poggenhans 等人^[69]使用 C++ 开发了开源的 Lanelet2。Lanelet2 是在 lanelets 基础上完善的产品，如图 2-9 所示。

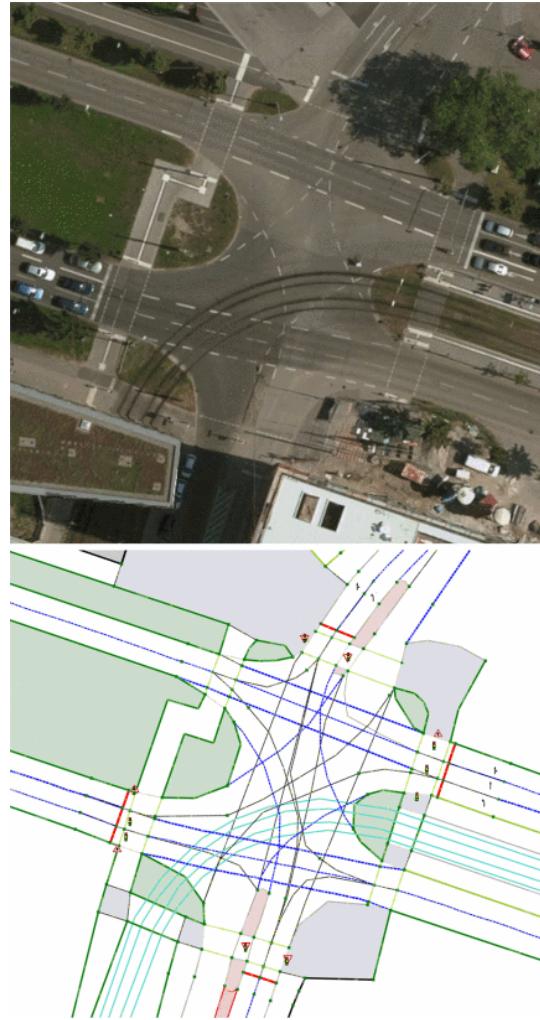


图 2-9 Lanelet2 道路模型

2.1.3 改进的 OpenDRIVE 格式

根据开源地图格式 OPENDRIVE 和文献[1-4]等人的工作，我们首先测试了使用多项式曲线，线段，圆弧和螺旋曲线来连接路点构造出笛卡尔坐标系中的参数化曲线。然而，我们发现，识别一段路径是应该使用直线，圆弧还是螺旋曲线来表示很麻烦，我们需要人为决策和标注，这极大地降低了地图创建的效率，同时，螺旋曲线的系数计算并不稳定，所以最终参考 wang 等人^[70]的工作，使用三次多项式曲线来表征整个参考线。

$$\begin{cases} x(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \\ y(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \end{cases} \quad (2-1)$$

根据这条参数化曲线，可以很容易地计算出曲线上任何一个点的姿态，包括位置，方向和曲率，然而曲线的参数 t 并没有物理意义，只是一个单纯的数学变量，对于自动驾驶车辆的运动规划模块来说，我们很难直接利用这

条曲线方程。如果我们引入微分几何中的弧长参数 s 的概念，那么很多问题都能够得到简化。因为参考线用弧长参数 s 表示，也就是用里程表示，那么所有障碍物的位姿信息和速度信息都能够投影到这条弧长参数化曲线，这样能够充分利用里程和与道路的偏移信息，这比直接利用笛卡尔坐标系更方便。因为道路是弯曲的，直接给出笛卡尔坐标并不能直接反映位置是否在车道内，也不能反映车辆朝向与道路走向的关系。

为了方便参考线的弧长参数化，首先使用三次样条曲线插值，来对一系列离散的路点进行插值，这样本文就得到了连续的样条曲线形式的参考线。接下来，本文使用数值积分计算累积的弧长参数 s 的值，然后在参考线中采样，关于这个采样的技巧文献^[70]有过证明，如果采样太稀疏，那么近似弧长参数化的样条曲线会与真实的参考线有较大的匹配误差，这也就意味着弧长参数化曲线并不与真实的路点曲线近似，失去了参数化的意义；如果采样的点太密集，即使能够做到匹配误差极小，但是会增加计算复杂度，并且每一段近似弧长参数样条曲线的系数是要存储在内存中，这给地图预处理带来困难，并且也给随后的实时查询系数表带来较大的挑战。根据本文的试验，采样间隔保持在 1 米左右，这样既能兼顾匹配误差，又能兼顾程序实时运行中的查询效率。因为弧长参数化的曲线和样条曲线参考线是在保证在一定精度下的匹配，所以将这个过程叫做参考线的近似弧长参数化，然后通常将这个近似弧长参数化后的曲线也叫做道路的参考线，只不过和原来的参考线相比会有微量的误差，这个误差的精度取决于采样的步长。如图图 2-10 所示，近似弧长参数化的三次多项式曲线表示为

$$\begin{cases} x(s) = b_0 + b_1 s + b_2 s^2 + b_3 s^3 \\ y(s) = c_0 + c_1 s + c_2 s^2 + c_3 s^3 \end{cases} \quad (2-2)$$

只要边值条件已知就可以计算出曲线的系数，我们也就可以根据上式计算出道路参考线上的任意一点的位置。方向和曲率也很容易地使用关于弧长参数的一阶导数和二阶导数计算出来。

$$\begin{aligned} \frac{dx}{ds} &= x' = \cos \theta, \quad \frac{dy}{ds} = y' = \sin \theta \\ \frac{d^2x}{ds^2} &= x'', \quad \frac{d^2y}{ds^2} = y'' \\ \kappa &= \frac{x'y'' - x''y'}{\sqrt{(x' + y')^3}} \end{aligned} \quad (2-3)$$

从曲线方程可以看出来，参考线是二阶连续的，也就是道路的曲率是连续的，这一点保证了本文后续轨迹规划的严格连续性，并且基于路点和参考

线，可以来构建道路，如图 2-11 所示。

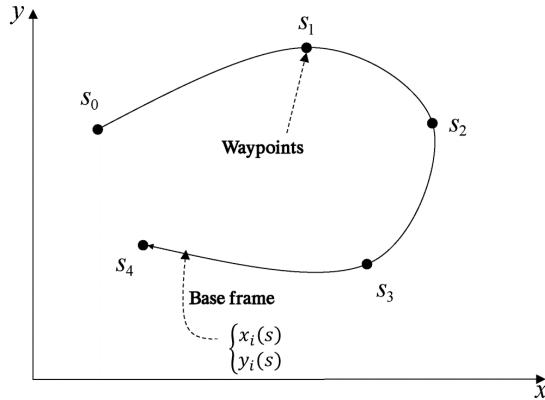


图 2-10 参考线的采样于弧长参数化曲线

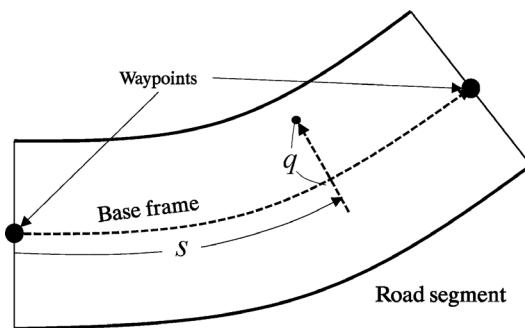


图 2-11 基于路点和参考线形成的道路片段

具体地，关于方程系数的计算，我们以三次多项式曲线为例，设样条曲线方程为

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \quad (2-4)$$

边值状态条件的起点状态和终点状态为下式

$$\mathbf{x}_0 = [x_0 \quad y_0 \quad \theta_0]^T, \mathbf{x}_f = [x_f \quad y_f \quad \theta_f]^T \quad (2-5)$$

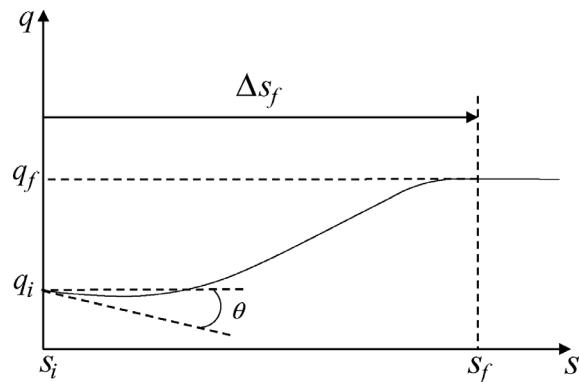


图 2-12 Frenet 坐标系中的运动基元

其中

$$\tan \theta = y' = a_1 + 2a_2x + 3a_3x^2 \quad (2-6)$$

由边值条件组成 4 个约束

$$\begin{aligned} y(x_0) &= y_0, \tan \theta_0 = y'(x_0) \\ y(x_f) &= y_f, \tan \theta_f = y'(x_f) \end{aligned} \quad (2-7)$$

将约束写成矩阵形式

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_f & x_f^2 & x_f^3 \\ 0 & 1 & 2x_0 & 3x_0^2 \\ 0 & 1 & 2x_f & 3x_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_f \\ \tan \theta_0 \\ \tan \theta_f \end{bmatrix} \quad (2-8)$$

弧长的数值积分公式为

$$s_f = \int_{x_0}^{x_f} \sqrt{1+y'^2} dx \quad (2-9)$$

然后使用弧长参数曲线式 (2-2) 来近似参数化样条曲线。三次多项式曲线的一阶导数和二阶导数为

$$\begin{aligned} x'(s) &= b_1 + 2b_2s + 3b_3s^2, x''(s) = 2b_2 + 6b_3s \\ y'(s) &= c_1 + 2c_2s + 3c_3s^2, y''(s) = 2c_2 + 6c_3s \end{aligned} \quad (2-10)$$

边值约束式 (2-7) 可以写为

$$\begin{aligned} x(s_0) &= x_0, x'(s_0) = \cos \theta_0 \\ y(s_0) &= y_0, y'(s_0) = \sin \theta_0 \\ x(s_f) &= x_f, x'(s_f) = \cos \theta_f \\ y(s_f) &= y_f, y'(s_f) = \sin \theta_f \end{aligned} \quad (2-11)$$

写成矩阵的形式

$$\begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 \\ 1 & s_f & s_f^2 & s_f^3 \\ 0 & 1 & 2s_0 & 3s_0^2 \\ 0 & 1 & 2s_f & 3s_f^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_f \\ \cos \theta_0 \\ \cos \theta_f \end{bmatrix} \quad (2-12)$$

$$\begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 \\ 1 & s_f & s_f^2 & s_f^3 \\ 0 & 1 & 2s_0 & 3s_0^2 \\ 0 & 1 & 2s_f & 3s_f^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_f \\ \sin \theta_0 \\ \sin \theta_f \end{bmatrix} \quad (2-13)$$

这样很容易建立了道路参考线的方程，一般而言，Frenet 坐标系会附着于最右侧车道的参考线上。在这里需要区分道路参考线和车道参考线，根据

OPENDRIVE 格式，将行驶方向相反的两条车道交界的实线作为道路的参考线，如果道路的车道都是一个行驶方向，即单向道路，就使用道路最左边的边界作为道路参考线；而车道参考线一般就是每一条车道的中心线，这条中心线是根据道路的参考线进行偏移构造的。有了车道的参考线，将参考线进行一定宽度的偏移就可以构造车道线和道路边沿，然后整个规划算法也是基于这条参考线进行的。

2.2 笛卡尔坐标与 Frenet 坐标的相互投影

2.2.1 笛卡尔坐标映射到 Frenet 坐标

为了得到障碍物在 Frenet 坐标系中的航向角和位置信息，需要给出转换方程。在物理意义上，笛卡尔坐标中的点到 Frenet 坐标系中的映射点就是笛卡尔坐标点到参考线距离最短的点，本文使用 kd 树和牛顿法来计算这个映射点，如图 2-13、图 2-14 所示。

kd 树是包含多维向量信息的二项树数据结构，这是一个用来计算 k 近邻的十分常用的工具，为了将笛卡尔坐标点映射到 Frenet 坐标点，本文的做法是先将参考线离散化成一系列离散的路点，为了保证映射点的求解效率，需要对路点间的距离选取合适的值，根据经验，选取步长 $s = 1.0m$ ，然后将离散路点存储到 kd 树，这样就能够快速查找到一个笛卡尔点到参考线上的距离比较小的点，能够大大减小使用优化类算法的迭代次数。对于映射点的求解，先进行 kd 树的构造，再进行 kd 树的查询，然后使用非线性优化方法求得最近距离，最后使用向量的外积求得方向，也就是确定笛卡尔坐标点是在参考线的左侧还是右侧，以此就确定了映射点的 Frenet 坐标和 Frenet 航向角。

具体地，本文会对 kd 树在规划中的应用做一个详细地介绍。给定一个样本数据集合 $S \in R^n$ 和切分轴 r ，然后递归地构建一颗 kd 树，并且每一次循环构造一个节点。其中， $S = [X_0, X_1, \dots, X_M]$ ， $X_0 = [x_0, y_0]^T$ ，M 是参考线上离散路点数据集的容量。如果 S 中的样本量为 1，就将 S 中唯一的一个向量作为当前节点，并不再添加左右子节点；如样本量大于 1，就将 S 内的元素按照切分轴 r 排序，选出 r 排序的中位向量作为当前节点，此时 S 被分割为 S_L, S_M, S_R 三部分，第一个记号表示中位向量之前的向量，第二个记号表示中位向量，第三个记号表示排列在中位向量之后的向量，这三个部分构成了两个分区。此时如果切分轴 r 位于一个向量的最后一维，将切分轴加 1 就进入向量的第一维，若切分轴不位于向量的最后一维，直接将切分轴加 1。这样再接着分别

分割 S_L, S_R , 这样数据集就分成了四个分区, 这样递归地分割, 直至数据集中只剩 1 个元素或 0 个元素为止。

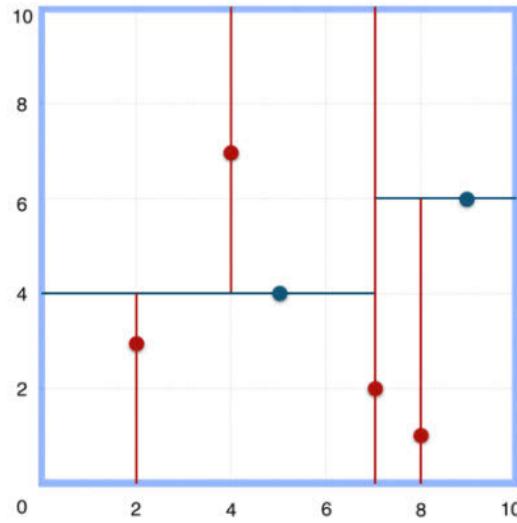


图 2-13 构建 kd 树过程中元素的切分

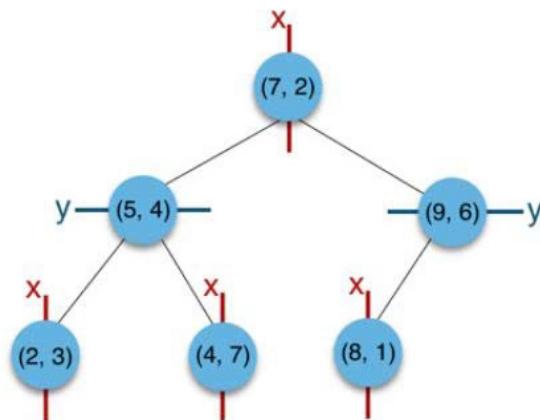


图 2-14 6 个二维节点构建的 kd 树

在参考线 kd 树构造完成之后, 然后再使用二分法的思想去搜索最近点, 这样算法的时间复杂度为 $O(\log(M))$, 相比于基于遍历算法的时间复杂度 $O(M)$, kd 树的最近点搜索大大提高了映射点求解效率。对于一个笛卡尔坐标点 $\mathbf{p} = [x_p, y_p, \theta_p]^T$, 本文通过 kd 树搜索, 找到参考线上的近似的最近点 $\mathbf{p}_r = [x_r, y_r]^T$, 如图 2-15 所示, 红色点是笛卡尔坐标点, 绿色点是通过 kd 树搜索的到参考线的近似映射点, 蓝色点是构建参考线的路点, 然后为了求解映射点的精确坐标, 提取近似映射点所在的两条近似弧长参数化曲线 c_1, c_2 的系数, 然后进入非线性优化步。

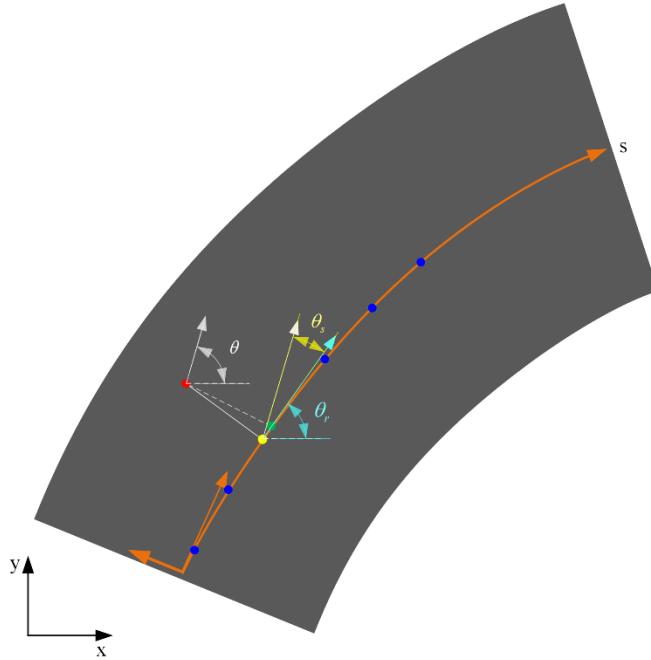


图 2-15 笛卡尔坐标点到参考线的近似映射点和精确映射点

曲线 c_1, c_2 的方程都是如下的这种形式，两条曲线的系数是从上一节存储的参考线系数表中查询得到的。

$$c_1 = \begin{cases} x_1(s) = b_0 + b_1s + b_2s^2 + b_3s^3 \\ y_1(s) = c_0 + c_1s + c_2s^2 + c_3s^3 \end{cases}, \quad c_2 = \begin{cases} x_2(s) = b_0 + b_1s + b_2s^2 + b_3s^3 \\ y_2(s) = c_0 + c_1s + c_2s^2 + c_3s^3 \end{cases} \quad (2-14)$$

对于曲线 c_1 ，构造的非线性优化的目标函数和约束函数为，

$$\begin{aligned} \min \quad & J = (x_p - x_1(s))^2 + (y_p - y_1(s))^2 \\ \text{s.t.} \quad & s_r - \Delta s \leq s \leq s_r \end{aligned} \quad (2-15)$$

对于曲线 c_2 ，构造的非线性优化的目标函数和约束函数为，

$$\begin{aligned} \min \quad & J = (x_p - x_2(s))^2 + (y_p - y_2(s))^2 \\ \text{s.t.} \quad & s_r \leq s \leq s_r + \Delta s \end{aligned} \quad (2-16)$$

对于这两个带约束的非线性优化问题，并且该问题是一个只包含单个决策变量的凸优化问题，本文使用序列二次规划算法计算最优解 s_{opt} ，初值选为 s_r ，有 $x_1(s_r) = x_{lr}$ ， $y_1(s_r) = y_{lr}$ ，这样只通过几次迭代就能够求得最优解 J_{opt} 。

对于两条曲线，可以求得两个最优解 J_{opt1} 和 J_{opt2} ，也就是在两条曲线上分别求得一个距离最近点，比较两个最近点之间的最优解，选取包含最小值的点作为真实的笛卡尔坐标点到参考线的映射点，此时 $s = s_{opt}$ ，接着将 s_{opt} 带入曲线方程可以求得映射点的笛卡尔坐标点 $[x_{opt}, y_{opt}]$ 。笛卡尔坐标点到映射点两点之间的距离，也就是笛卡尔坐标系点到 Frenet 坐标系的偏移量的绝对值

$|\rho_{opt}| = J_{opt}$ 。并且本文规定，笛卡尔坐标点在参考线左侧，那么 Frenet 坐标 ρ 为正值，反之则为负值。为了判断左右，这里引入了向量的外积概念，

$$\begin{aligned}\vec{a} &= [1, \tan \theta_{rop}] \\ \vec{b} &= [x_p - x_{opt}, y_p - y_{opt}]\end{aligned}\quad (2-17)$$

两个向量的外积为

$$L = y_p - y_{opt} - (x_p - x_{opt}) \tan \theta_{rop} \quad (2-18)$$

根据右手法则，若 $L > 0$ ，则点在参考线的左侧；若 $L = 0$ ，则点在参考线上；若 $L < 0$ ，则点在参考线右侧。其中， θ_{rop} 为参考线上映射点的航向角，本文给出它的求解方程为

$$\theta_{rop} = \arctan(dy(s_{opt}) / dx(s_{opt})) \quad (2-19)$$

最终，给出 ρ_{opt} 的求解方程

$$\rho_{opt} = \begin{cases} \frac{L}{|L|} J_{opt}, & |L| > 0 \\ 0, & L = 0 \end{cases} \quad (2-20)$$

这样，本文就得到了 $[x_p, y_p]^T \Leftrightarrow [s_{opt}, \rho_{opt}]^T$ 之间的转换关系，然而笛卡尔坐标点的航向角 θ_p 还有待转换到 Frenet 航向角 θ_{opt} 。对于笛卡尔航向角与 Frenet 航向角的转换，有如下公式

$$\theta_s = \theta - \theta_r \quad (2-21)$$

其中， θ 是笛卡尔坐标系的航向角， θ_r 是参考线上映射点的航向角， θ_s 就是 Frenet 航向角，如图 2-12 所示，可以得到红色坐标点的 Frenet 航向，有

$$\theta_{sop} = \theta_p - \theta_{rop} \quad (2-22)$$

本文通过 kd 树和 SQP 的联合求解，能够快速得到一个笛卡尔位姿到 Frenet 位姿的转换关系，如图 2-15 所示，黄色点是红色点到参考线的精确映射点。

2.2.2 Frenet 坐标到笛卡尔坐标的转换

在结构化道路中的采样文献[5]有过研究，我们根据车道参考线去定义采样点向量 $\vec{p}(s, \rho) = [x(s, \rho), y(s, \rho), \theta(s), \kappa(s, \rho)]^T$ ， s 是纵向里程， ρ 是横向偏移量，为了符合交通法规和满足安全要求，采样端点的航向角会和道路中心线的航向角一致，这一节给出采样点的 Frenet 坐标到笛卡尔坐标的转换公式。

$$\begin{cases} x(s, \rho) = x_r(s) + \rho \cos(\theta_r(s) + \pi/2) \\ y(s, \rho) = y_r(s) + \rho \sin(\theta_r(s) + \pi/2) \\ \theta(s, \rho) = \theta_r(s) \end{cases} \quad (2-23)$$

其中，根据参考线方程 $\vec{r} = [x_r(s), y_r(s), \theta_r(s), \kappa_r(s)]^T$ ，只要已知 Frenet 坐标 $[s, \rho]$ ，然后希望采样的位姿平行于参考线，采样位姿和对应的参考线投影点的曲率圆是共圆心的，所以我们就能够得到转换到笛卡尔坐标系的坐标值。在这里，需要注意的是，参考线左侧的偏移 ρ 为正，反之则为负。但是此时并不能够得到采样点的曲率，一般而言，我们采样点的曲率圆和参考线上投影点处的曲率圆是共圆心的，如图 2-16 所示。

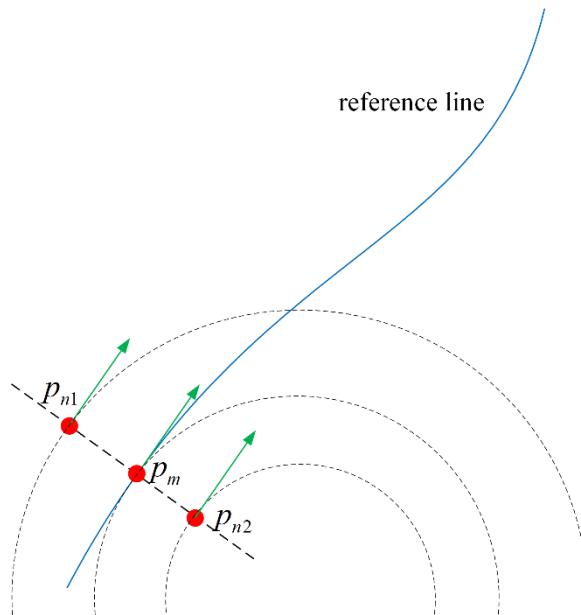


图 2-16 参考线法线方向上采样点的曲率圆

所以能够给出采样点曲率计算方程为

$$\kappa_{spl} = \frac{\kappa_r}{|\kappa_r|} \left(\frac{1}{|\kappa_r|} + \rho \right)^{-1} \quad (2-24)$$

参考线投影点的曲率 κ_r 的计算公式为

$$\kappa_r = \frac{x'y'' - x''y'}{\sqrt{(x'^2 + y'^2)^3}} \quad (2-25)$$

我们通过分析，若 $\kappa_r > 0$ ，即曲线是上凸的，那么曲线走势对应的物理意义是车辆的车轮向右偏，那么无论 ρ 是正值还是负值（对应着曲线的左侧和右侧），采样点曲率方程都是正确的。然而当 $\kappa_r < 0$ 时，曲线是下凹的，采样点在曲线左侧那么曲率圆半径会减小，在曲线右侧则曲率圆半径会增加，所以方程 (2-24) 并不正确，但是添加一个符号可修正上式，即

$$\kappa(s, \rho) = \frac{\kappa_r(s)}{|\kappa_r(s)|} \left(|\kappa_r(s)|^{-1} + \frac{\kappa_r(s)}{|\kappa_r(s)|} \rho \right)^{-1} \quad (2-26)$$

通过枚举四种情形而论证，上式确实能够表达出采样点的曲率。其实上面对曲率正负和曲线凹凸关系的描述并不准则，在这里我们严格定义一下，我们规定参考线是有方向的，即沿着参考线的方向，参考线左侧点的偏移是正的，右侧点的偏移是负的，然后参考线是弯曲的，弯曲程度可以使用曲率的绝对值表示，弯曲的方向使用正负表示，即曲线如果是顺时针旋转，则曲率符号为正，如果曲线是逆时针旋转，则曲率符号为负。所以有采样点的方程

$$\begin{cases} x(s, \rho) = x_r(s) + \rho \cos(\theta_r(s) + \pi / 2) \\ y(s, \rho) = y_r(s) + \rho \sin(\theta_r(s) + \pi / 2) \\ \theta(s, \rho) = \theta_r(s) \\ \kappa(s, \rho) = \frac{\kappa_r(s)}{|\kappa_r(s)|} \left(|\kappa_r(s)|^{-1} + \frac{\kappa_r(s)}{|\kappa_r(s)|} \rho \right)^{-1} \end{cases} \quad (2-27)$$

2.3 碰撞检测

碰撞检测是游戏领域、机器人领域、工程仿真领域内一项基本的几何运算^[71]，碰撞检测是计算两个目标是否，什么时候，以及什么位置发生碰撞，碰撞检测运算模块是规划模块耗时、耗资源的一个子模块。目前大部分的规划算法都是做离散的碰撞检查，也就是说算法将轨迹离散成一系列的栅格或某种状态来进行碰撞运算。并且一些算法只做静态的碰撞检查，这意味着碰撞运算忽略时间因素，这会带来潜在的碰撞风险。

规划算法生成的轨迹 T 需要满足碰撞避免约束，这种约束是车辆的外部约束^[2]，也是一种硬约束^[72]，因此有 $T \subset C_{free}$ 。每当环境变化，或者时间戳在增长，以及规划算法再次运行，都需要再次调用碰撞检测算法，所以一个设计优良的碰撞检测算法至关重要。目前流行的碰撞检测算法中，基于代价地图的碰撞检测和基于车体圆的碰撞检测比较流行。

2.3.1 基于代价地图的碰撞检测

代价地图是将传感器感知到的环境信息处理成栅格的形式，每一个栅格的值可以使用代价值 0-255 表示，通过值来表示栅格的被占用，自由区域和未知区域这三种状态信息。

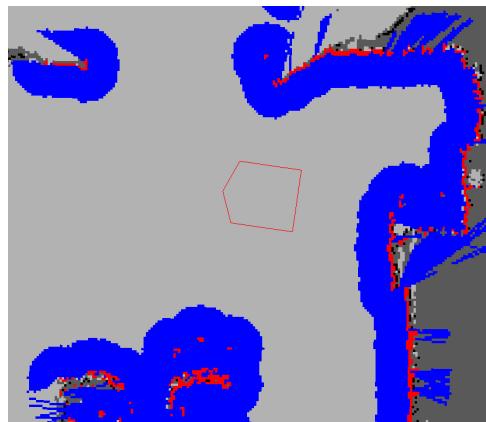


图 2-17 代价地图中的碰撞检查

在图 2-17 中，红色格子（图中红色、蓝色区域都是一系列栅格累计出来的）代表的是代价地图中的障碍，蓝色格子代表的是通过机器人内切圆半径计算的障碍物膨胀，红色多边形代表的是机器人轮廓(机器人轮廓的垂直投影)。为了使机器人不碰到障碍物，机器人的轮廓绝对不允许与红色格子相交，机器人的中心绝对不允许与蓝色格子相交。

为了对车辆的轨迹进行碰撞检测，需要将车辆的每一个状态投影到栅格图中，如图图 2-18 所示。对于一条车辆轨迹而言，每一个状态都被分解成栅格的形式，将车辆轨迹的栅格都存储到一个轨迹表 *TrajList* 中。

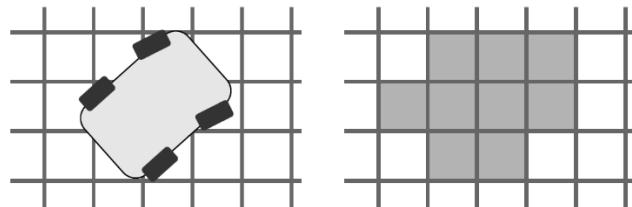


图 2-18 代价地图中车辆的栅格化投影

对环境空间和车辆状态空间的栅格分解是规划算法中一个很流行的技术，不仅方便搜索出可行的路径，而且能够快速地进行碰撞检查。然而选择合适的栅格分辨率需要丰富的经验，分辨率是否合适影响着传感器信息的保真度和车体的近似度。如果栅格的分辨率太高，规划算法搜索解的时间复杂度呈指数增长，所以搜索出解的效率较低，并且会增加碰撞检查的时间复杂度；如果分辨率太小，那么自由空间的区域面积就会减少，使得碰撞检查的结果相对保守，并且因为基于栅格的规划算法是分辨率完备的，所以搜索出可行解的可能性也会降低。

有了包含障碍物信息的代价地图和车辆轨迹信息的轨迹表，此时遍历轨

迹表进行简单的数值计算，就能够快速完成碰撞检查。碰撞检查的时间复杂度取决于栅格的分辨率大小。

2.3.2 基于车体圆的碰撞检测

另一种流行的碰撞检测方式是基于车体包络圆的检测技术^[2,65,73,74]，在碰撞检查中，车辆和其他对象都被建模成三自由度的在二维平面空间内可以平移和旋转的对象。车体可以被矩形近似，并且也可以被一组重叠圆近似，以一种最优的方式使用半径相同的圆盘来覆盖矩形或任意多边形，在许多文献中有过研究，并且被证明是 NP 难的^[75]。

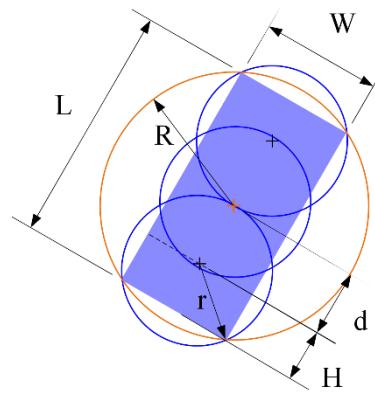


图 2-19 车体包络圆

关于上图的参数， L 表示车体纵向长度， W 表示车体横向宽度， R 表示车体外接圆半径， r 表示车体小包络圆半径， H 表示后轴中心点到车体后边界距离， d 表示两个小包络圆圆心的距离。大外接圆的圆心位于车体的中心点，最下方的一个小圆的圆心在后轴中心点，每一个小圆的半径都是相同的，接下来给出各个参数的数学关系

$$r = \sqrt{\frac{W^2}{4} + H^2}, d = \frac{L}{2} - H, R = \sqrt{\frac{W^2}{4} + \frac{L^2}{4}} \quad (2-28)$$

本文使用三个磁盘覆盖车体，上面也给出了各个参数的数学关系。接下来，在进行碰撞检查的时候，已知了后轴中心点的位姿，需要给出四个磁盘的位置。

$$\begin{aligned} x_2 &= x + d \cos \theta, x_3 = x + 2d \cos \theta \\ y_2 &= y + d \sin \theta, y_3 = y + 2d \sin \theta \end{aligned} \quad (2-29)$$

其中，沿着车辆的纵轴朝向，后轴中点位姿 $[x, y, \theta]^T$ ，第二个和第三个磁盘圆心坐标是 $[x_2, y_2]^T$ 和 $[x_3, y_3]^T$ 。如图 2-19 所示，是车体包络圆形成的轨迹。

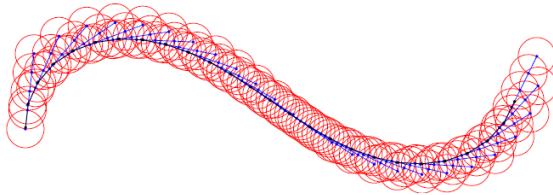


图 2-20 车体圆轨迹

在自动驾驶中，广义上的障碍物包括车道的边界，车道线，以及道路交通参与者等障碍物。对于轨迹和道路边界以及车道线的检查方式是，基于 Frenet 坐标系，有了道路边界或车道线的左右边界 Frenet 坐标值 $\rho_{left}, \rho_{right}$ ，需要满足磁盘圆心 Frenet 横坐标值 ρ_i 与 ρ_{left} 或 ρ_{right} 的差值小于磁盘圆半径 r 。即有：

$$\begin{aligned} \rho_i &< \rho_{left} - r \\ \rho_i &> \rho_{right} + r \end{aligned} \quad (2-30)$$

如果不满足上式，说明路径与道路边界或车道线发生碰撞。对于一条路径，设采样 N 个点进行碰撞检查，那么车体圆就有 $4N$ 个，假设障碍物有 M 个，那么最好情况下的平均时间复杂度达到 $O(NM)$ ，最坏情况下的平均时间复杂度达到 $O(4NM)$ 。对道路边界或者车道线进行碰撞检测之后，就需要对道路中的障碍物进行碰撞检查，目前，障碍物的表达形式主要是以离散的点进行的，在车辆行进过程中，将周围的障碍物点先处理成凸包的形式，然后采样凸包的边界，得到一系列离散的点存储到 kd 树中。车辆运动过程中，车体的小包围圆和外接圆也会形成轨迹，图中显示了车体包围圆的轨迹。然后对于一个磁盘，找到 kd 树中的最近障碍物点，再检查碰撞关系。

具体的碰撞检查策略是，对于一条轨迹，先使用第一个路点处的车体的外接圆进行碰撞检测，如果没有碰撞，说明车体不会与障碍物进行碰撞，如果有碰撞，则继续使用车体的三个包围圆分别与障碍物进行碰撞检查，如果没有碰撞，则检查下一个路点，如果有碰撞，则退出碰撞检查，并给出轨迹发生碰撞的标志信息。

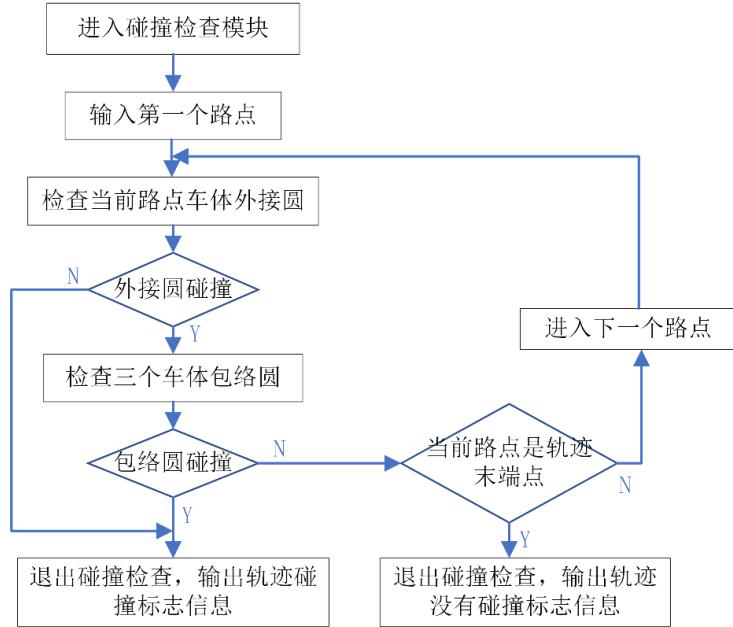


图 2-21 车体圆碰撞检测策略流程图

2.4 运动基元

对于采样类的算法来说，通常是采样几组位姿，通过某种曲线来连接采样的状态，常见的曲线包括多项式曲线，Dubins 曲线，Reeds-Shepp 曲线，螺旋曲线，为了减少采样类算法的计算复杂度，一般像多项式曲线和螺旋曲线常被选中，作为连接两个状态的曲线。在图论中，常把这条曲线叫做边(Edge)，在规划算法中，常把这条曲线叫做运动基元，也就是作为机器人或者车辆的运动轨迹片段。在本节中，文中会描述表示车辆运动的运动基元家族中比较代表性的曲线。本文会详细介绍两种比较常见的运动基元螺旋曲线和多项式曲线，并阐述他们怎么被表示，以及怎么样获得特殊的路径，并分析与比较各自的优缺点。

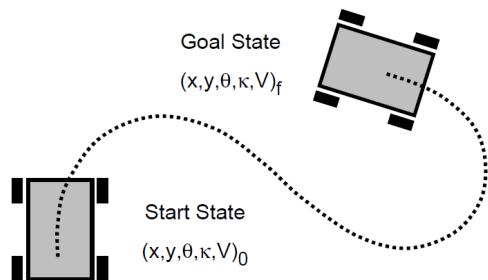


图 2-22 连接起点状态和终点状态的运动基元

2.4.1 螺旋曲线

运动基元的定义是，一个连续的函数 f 将区间 $[0,1]$ 映射到机器人构型空间，例如 $C = \{(x, y, \theta, \kappa)\}$ ：

$$f : [0,1] \rightarrow C \quad (2-31)$$

机器人的起点构型是 $f(0) = q_{init} \in C$ ，终点构型是 $f(1) = q_{goal} \in C$ 。我们致力于寻找一条满足相关约束的路径（运动基元）。对于车辆的自行车运动学模型，有下面这个方程

$$\begin{aligned}\dot{x}(t) &= v(t) \cos \theta(t) \\ \dot{y}(t) &= v(t) \sin \theta(t) \\ \dot{\theta}(t) &= \kappa(t) v(t) \\ \dot{\kappa}(t) &= u(t)\end{aligned} \quad (2-32)$$

其中，

$$\kappa(t) = \frac{1}{r(t)} = \frac{\tan \delta(t)}{L} \quad (2-33)$$

车辆的状态向量为 $[x, y, \theta, \kappa]^T$ ，车辆运动的输入量或者控制量为 $[v, u]^T$ ， κ 是车辆轨迹的曲率，根据上述方程计算而来， L 是前轴后轴之间的轴距， δ 是前轮转向车辆的前轮转角，本质上，车辆通过控制速度和车轮转角这两个输入量来完成运动。为了消除时间的影响，并将轨迹解耦成路径-速度这两个组成成分，为了减少车辆运动轨迹中速度这个维度，也就是只有轨迹的形状，忽略速度信息，我们可以令

$$\frac{ds}{dt} = 1 \quad (2-34)$$

这也就意味着车辆的速度是 1m/s，这样我们就得到了去掉速度维度的车辆运动学方程，

$$\begin{aligned}\frac{d}{ds} x(s) &= \cos \theta(s) \\ \frac{d}{ds} y(s) &= \sin \theta(s) \\ \frac{d}{ds} \theta(s) &= \kappa(s)\end{aligned} \quad (2-35)$$

这样车辆的状态向量为 $[x, y, \theta]$ ，控制量为 κ ，这样通过控制曲率 κ 就可以实现路径到达指定的位姿。上面的这种形式对于轨迹生成而言是十分有用的，因为它允许轨迹的几何形状独立于速度。通过这种转换，忽略速度信息后，端点约束可以表示为

$$\begin{aligned}\mathbf{x}(s_0) &= [x_0, y_0, \theta_0, \kappa_{init}]^T \\ \mathbf{x}(s_f) &= [x_f, y_f, \theta_f, \kappa_f]^T\end{aligned}\quad (2-36)$$

根据 Kelly 和 Nagy^[42], 以及 Howard^[33]等人的工作, 我们也使用多项式螺旋曲线表示运动基元。螺旋曲线的定义是, 曲率是关于弧长 s 的多项式函数而形成的曲线, 所以三次多项式螺旋曲线的形式是:

$$\kappa(s) = \kappa_0 + \kappa_1 s + \kappa_2 s^2 + \kappa_3 s^3 \quad (2-37)$$

五次多项式螺旋曲线的形式是:

$$\kappa(s) = \kappa_0 + \kappa_1 s + \kappa_2 s^2 + \kappa_3 s^3 + \kappa_4 s^4 + \kappa_5 s^5 \quad (2-38)$$

我们在规划的不同阶段和不同的车辆行驶速度时使用三次或五次曲线, 这取决于曲率导数的高阶连续性的需要。三次多项式曲线可以保证运动基元之间曲率的连续性, 但是不能保证曲率导数的连续性, 在低速场景中, 生成路径的跟踪误差可以被忽视, 但在高速场景中, 这种不连续性就不能被忽略了。使用五次曲线则可以保证曲率导数的连续性, 在多种文献中有过研究五次曲线会导致更光滑的机器人运动, 如图 2-23 所示。



图 2-23 三次多项式螺旋曲线（点画线）与五次多项式螺旋曲线（实线）。虚线是起
点和终点状态的曲率圆

在上图的例子中, 五次多项式曲线基元的起点约束包括 $\frac{d\kappa(0)}{ds} = 0, \frac{d^2\kappa(0)}{ds^2} = 0$ 。对于三次多项式螺旋曲线运动基元来说, 如果加上上面这两种边值约束就会过约束了, 所以三次多项式运动基元就会需要去掉一种约束。产生的结果就是从起点位姿到达终点位姿, 五次多项式螺旋曲线更加平滑, 但是路径的长度也更长。多项式螺旋曲线使用梯度下降法能够快速可靠地收敛, 同时能够充分表征路径空间, 所以是对运动基元的一个很方便的建模方式。

将控制量曲率写成关于弧长 s 的多项式形式, 可以拥有许多必要的自由度来满足任意数量的约束, n 次螺旋曲线就是关于弧长 s 的 n 次曲率多项式,

$$\kappa(s) = \kappa_0 + \kappa_1 s + \kappa_2 s^2 + \kappa_3 s^3 + \dots \quad (2-39)$$

在复平面中, 这些曲线都叫做广义角螺旋 (generalized Cornu spiral) 曲线, 三次螺旋曲线是一种代表性的角螺旋曲线, 如图 2-24 所示,

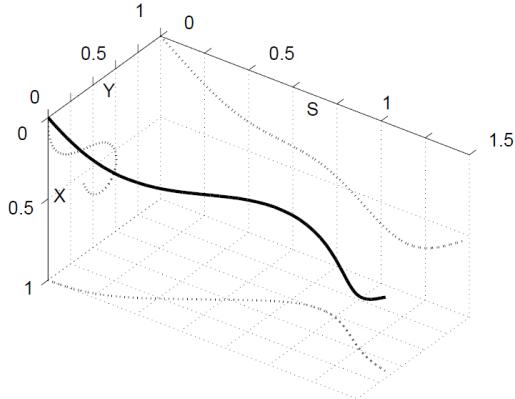


图 2-24 三次螺旋曲线在三维笛卡尔空间的投影。在这个图的 xy 平面上，投影后的曲线在起点和终点处的航向角发生了 180 度的改变，并且曲率的符号也发生了改变
(曲率符号可以参见 2.2.2 节)

螺旋曲线运动基元具有许多优点，可以总结为，使用少量的参数就可以表示车辆的任意可行的运动。三次多项式螺旋曲线航向角和位置的闭式解可以被表示为

$$\begin{aligned}\kappa(s) &= \kappa_0 + \kappa_1 s + \kappa_2 s^2 + \kappa_3 s^3 \\ \theta(s) &= \kappa_0 s + \frac{\kappa_1}{2} s^2 + \frac{\kappa_2}{3} s^3 + \frac{\kappa_3}{4} s^4 \\ x(s) &= \int_0^s \cos \theta(s) ds \\ y(s) &= \int_0^s \sin \theta(s) ds\end{aligned}\tag{2-40}$$

对于上面关于 x, y 求解的方程，一般叫做广义菲涅尔积分，关于菲涅尔积分和他们梯度的计算会成为轨迹生成的主要计算负担。根据轨迹的起点约束和终端约束，我们可以以三次多项式螺旋为例，写出下面的约束方程

$$\begin{aligned}\kappa_0 &= \kappa_{init} \\ \kappa_f &= \kappa_0 + \kappa_1 s_f + \kappa_2 s_f^2 + \kappa_3 s_f^3 \\ \theta_f &= \kappa_0 s_f + \frac{\kappa_1}{2} s_f^2 + \frac{\kappa_2}{3} s_f^3 + \frac{\kappa_3}{4} s_f^4 + \theta_0 \\ x_f &= \int_0^{s_f} \cos \theta(s) ds + x_0 \\ y_f &= \int_0^{s_f} \sin \theta(s) ds + y_0\end{aligned}\tag{2-41}$$

根据边值条件约束 $\mathbf{x}(s_0), \mathbf{x}(s_f)$ 和曲率函数 $k(s)$ 的方程，已知 $\kappa_0, \kappa_1, \kappa_2, \kappa_3, s_f$ 共五个未知的参数，正好能够根据上面五个约束方程求出五个参数。所以我们写出三次多项式螺旋曲线的参数向量为

$$\mathbf{p} = [\kappa_0, \kappa_1, \kappa_2, \kappa_3, s_f]^T \quad (2-42)$$

又因为 $\kappa_0 = \kappa_{init}$ ，为了简化后续的计算，我们将参数向量写成四维的形式， $\mathbf{q} = [\kappa_1, \kappa_2, \kappa_3, s_f]^T$ 。为了引入打靶法和牛顿法来求解这四个参数，我们假设给参数向量一个初值，然后根据方程（2-41）向后模拟，得到曲线的状态向量 $\mathbf{x}_g = [x_g, y_g, \theta_g, \kappa_g]^T$ ，此时模拟得到的终端状态是关于参数向量 \mathbf{q} 的方程，所以可以写成 $\mathbf{h}(\mathbf{q}) = \mathbf{x}_g(\mathbf{q})$ 的形式，这样我们得到一个差分方程

$$\mathbf{g}(\mathbf{q}) = \mathbf{x}_f - \mathbf{h}(\mathbf{q}) \quad (2-43)$$

具体地，

$$\begin{aligned} g_1(\mathbf{q}) &= x_f - h_x(\mathbf{q}) \\ g_2(\mathbf{q}) &= y_f - h_y(\mathbf{q}) \\ g_3(\mathbf{q}) &= \theta_f - h_\theta(\mathbf{q}) \\ g_4(\mathbf{q}) &= \kappa_f - h_\kappa(\mathbf{q}) \end{aligned} \quad (2-44)$$

为了使用牛顿法求解系数，我们给出牛顿迭代方程

$$\frac{\partial \mathbf{g}}{\partial \mathbf{q}} \Delta \mathbf{q} = \mathbf{g}(\mathbf{q}) \quad (2-45)$$

其中， $\frac{\partial \mathbf{g}}{\partial \mathbf{q}}$ 是差分方程的雅可比矩阵， $\Delta \mathbf{q}$ 是参数向量的增量，即 $\mathbf{q}_{k+1} - \mathbf{q}_k$ ，所以上式可以写作

$$\Delta \mathbf{q} = \mathbf{J}(\mathbf{q})^{-1} \mathbf{g}(\mathbf{q}) \quad (2-46)$$

雅可比矩阵 \mathbf{J} 的表达形式为

$$\mathbf{J} = - \begin{bmatrix} \frac{\partial x}{\partial \kappa_1} & \frac{\partial x}{\partial \kappa_2} & \frac{\partial x}{\partial \kappa_3} & \frac{\partial x}{\partial s_f} \\ \frac{\partial y}{\partial \kappa_1} & \frac{\partial y}{\partial \kappa_2} & \frac{\partial y}{\partial \kappa_3} & \frac{\partial y}{\partial s_f} \\ \frac{s_f^2}{2} & \frac{s_f^3}{3} & \frac{s_f^4}{4} & \kappa_f \\ s_f & s_f^2 & s_f^2 & \kappa_1 + 2\kappa_2 s_f + 3\kappa_3 s_f^2 \end{bmatrix} \quad (2-47)$$

求解上面这个雅可比矩阵是奇异的，所以我们通过使用数值方法，一个小扰动来估计雅可比矩阵。

$$\frac{\partial g_{i,j}(\mathbf{q})}{\partial q_j} = \frac{g_{i,j}(q_j + e_j, \mathbf{q}) - g_{i,j}(\mathbf{q})}{e_j} \quad (2-48)$$

微小扰动向量 $\mathbf{e} = [e_1, e_2, e_3, e_4]^T$ ，通过前向差分代替求解雅可比矩阵的数值解，能够牛顿迭代的效率。和其他非线性优化方法一样，解的初始猜测值会影响最终解的收敛速度，参数向量 \mathbf{q} 的一个好的初值会显著降低算法的迭代次数，在实际中，我们会预计算出一个解的查询表用于规划算法的在线使用。

对于构造查询表，我们根据 Nagy 的研究给出参数向量的初值设计

$$\begin{aligned}
 d &= \sqrt{x_f^2 + y_f^2} \\
 \Delta\theta &= \theta_f - \theta_0 \\
 s_f &= d\left(\frac{\Delta\theta^2}{5} + 1\right) + \frac{2}{5}\Delta\theta \\
 \kappa_3 &= 0 \\
 \kappa_1 &= \frac{6\theta_f}{s_f^2} - \frac{2\kappa_0}{s_f} + \frac{4\kappa_f}{s_f} \\
 \kappa_2 &= \frac{3}{s_f^2}(\kappa_0 + \kappa_f) + \frac{6\theta_f}{s_f^3}
 \end{aligned} \tag{2-49}$$

并设计算法的终止条件为

$$\begin{aligned}
 x_{err} &= 0.001m \\
 y_{err} &= 0.001m \\
 \theta_{err} &= 0.1rad \\
 \kappa_{err} &= 0.005m^{-1}
 \end{aligned} \tag{2-50}$$

并且设置了最大迭代次数为 50 次，一旦算法在迭代过程中满足的终止条件或者达到了最大迭代次数，就退出迭代输出结果。在构建查询表的过程中，我们通过均匀地采样多组终端状态，然后使用螺旋曲线生成运动基元，这样把生成的查询表存储成离线的形式，在车辆运动过程中通过搜索查询表来给螺旋曲线寻找较好的参数向量初值，如图 2-25 所示。

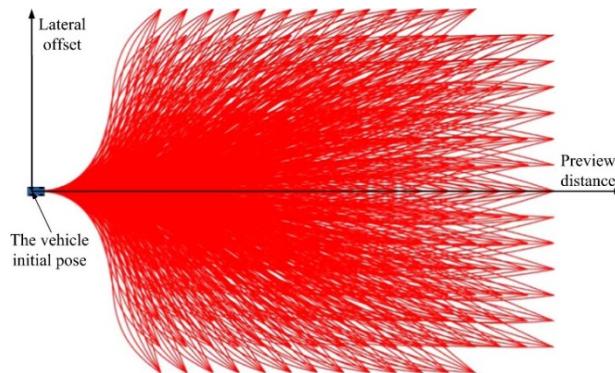


图 2-25 为三次螺旋曲线构建的查询表

2.4.2 多项式曲线

另一种比较常用的运动基元是多项式曲线，相比于螺旋曲线，不用计算菲涅尔积分，不用求逆解，也就是不再使用打靶法和牛顿法，会能更稳定地生成解，但是需要完成 Frenet 坐标到笛卡尔坐标的转换。在笛卡尔坐标系中

常见的多项式曲线可以表示为

$$y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots \text{ 或 } \begin{cases} x(t) = b_0 + b_1t + b_2t^2 + b_3t^3 + \dots \\ y(t) = c_0 + c_1t + c_2t^2 + c_3t^3 + \dots \end{cases} \quad (2-51)$$

然而对于边值条件

$$\mathbf{x}_0 = [x_0, y_0, \theta_0, \kappa_{init}]^T, \mathbf{x}_f = [x_f, y_f, \theta_f, \kappa_f]^T \quad (2-52)$$

当 $\theta_0/\theta_f \rightarrow \pm\frac{\pi}{2}$, 有 $\tan\theta_0/\tan\theta_f \rightarrow \pm\infty$, 此时 $dy/dx \rightarrow \pm\infty$, 无法有效得到方程的系数, 但实际中这样的多项式曲线是存在的, 如图 2-26 (b) 所示, 所以为了使用曲线连接图 b 这种采样位姿, 可以将多项式曲线变形为

$$x(y) = a_0 + a_1y + a_2y^2 + a_3y^3 + \dots \quad (2-53)$$

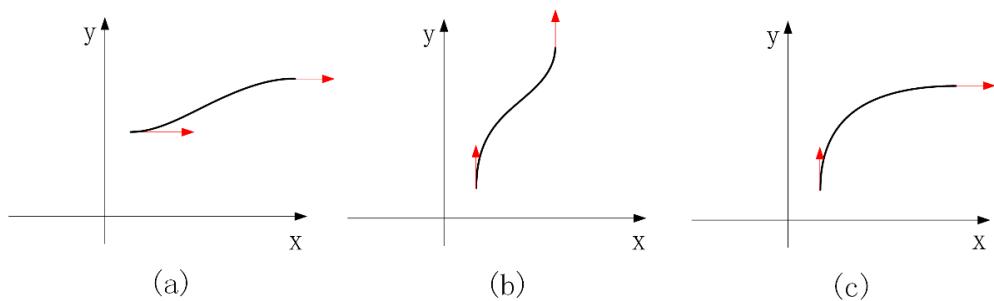


图 2-26 笛卡尔空间中使用三次多项式连接采样状态

这样在特殊位姿 $dy/dx \rightarrow \pm\infty$ 时, 有 $dx = a_1 + 2a_2y + 3a_3y^2 + \dots = 0$, 即 $a_1 = 0$, 这样能够求得曲线的系数。然而, 在更特殊的采样位姿下, 如图 c 所示, 这样的曲线我们无法有效使用笛卡尔坐标系中的多项式曲线来表示, 所以, 在笛卡尔坐标系中的任意两个位姿我们是不能使用多项式曲线来表示的, 当然根据上一节的知识, 螺旋曲线是完全能够表征各种特殊位姿的, 然而计算复杂度与多项式曲线比较起来太高。在实际中, 针对不同的行驶场景, 比如在自由空间中, 或者在低速场景下, 我们完全可以忽略螺旋曲线的计算复杂度而选择使用它。但是在结构化环境中, 由于参考线的存在, 此时我们更乐意选择多项式曲线, 因为我们此时不再使用笛卡尔坐标系中的方法来表征多项式曲线了, 同时在 Frenet 坐标系中, 本文选择多项式曲线连接采样状态, 多项式曲线表征的是 Frenet 坐标系中的多项式曲线, 其中自变量为纵向里程 s, 因变量为参考线的横向偏移量 ρ , 如式 (2-55) 所示, 这样能够克服特殊位姿情形下多项式曲线系数难以求解的问题。如图 2-27 所示, 显示的是三次多项式运动基元在 Frenet 坐标系中的位置关系。

$$\rho(x) = a_0 + a_1x + a_2x^2 + a_3x^3 \quad (2-54)$$

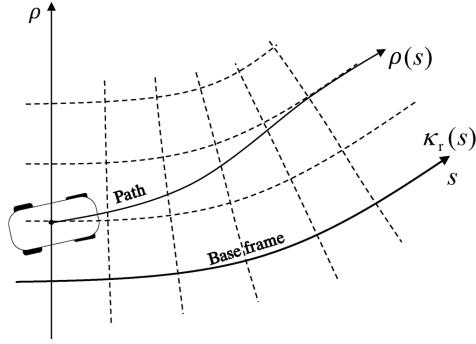


图 2-27 Frenet 多项式运动基元

Frenet 坐标系三次多项式运动基元的边值状态条件

$$\mathbf{x}_0 = [s_0, \rho_0, \theta_{s0}]^T, \mathbf{x}_f = [s_f, \rho_f, \theta_{sf}]^T \quad (2-55)$$

根据 Frenet 坐标系的多项式方程得,

$$\tan \theta_s = \rho' = a_1 + 2a_2 s + 3a_3 s^2 \quad (2-56)$$

这样由边值条件组成 4 个约束,

$$\begin{aligned} \rho(s_0) &= \rho_0, \tan \theta_{s0} = \rho'(s_0) \\ \rho(s_f) &= \rho_f, \tan \theta_{sf} = \rho'(s_f) \end{aligned} \quad (2-57)$$

将约束函数写成矩阵形式

$$\begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 \\ 1 & s_f & s_f^2 & s_f^3 \\ 0 & 1 & 2s_0 & 3s_0^2 \\ 0 & 1 & 2s_f & 3s_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \rho_0 \\ \rho_f \\ \tan \theta_{s0} \\ \tan \theta_{sf} \end{bmatrix} \quad (2-58)$$

这样能够稳定地求解多项式曲线的系数, 求解方程为

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & s_0 & s_0^2 & s_0^3 \\ 1 & s_f & s_f^2 & s_f^3 \\ 0 & 1 & 2s_0 & 3s_0^2 \\ 0 & 1 & 2s_f & 3s_f^2 \end{bmatrix}^{-1} \begin{bmatrix} \rho_0 \\ \rho_f \\ \tan \theta_{s0} \\ \tan \theta_{sf} \end{bmatrix} \quad (2-59)$$

3 路径规划

轨迹规划是一个包含[x,y,t]三个维度的规划问题，同时规划可行域一般来说都是非凸的，这使得解空间是巨大的并且非凸，大到在多项式时间内难以搜索到一个较好的解，同时也存在多个局部最优解，导致基于梯度的优化算法很容易收敛到局部最优而无法跳出这个局部最优点。为了处理解空间爆炸问题，需要降低维度，一般将[x,y]空间和时间维度解耦；为了解决非凸问题，通过采样的方式去搜索一个粗糙解，然后直接优化，或者通过精心设计目标函数，恰当处理约束，使得目标函数和可行域构成一个凸问题，然后优化。

本文结合了 Xu^[44] 和 Lim^[65] 文章中的思想，将轨迹规划解耦处理，并构建了一套解耦轨迹规划框架，如图 3 - 1 所示。

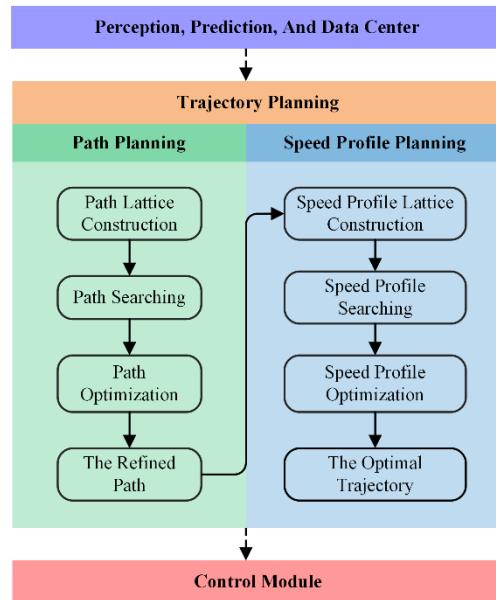


图 3 - 1 本文开发的解耦轨迹规划框架

框架的核心部分包括路径 lattice 搜索（路径规划的前端），路径优化（路径规划的后端），速度 lattice 搜索（速度规划的前端），速度优化（速度规划的后端）。在预处理中，我们将车辆的状态信息（位姿，速度，加速度，加加速度）和行驶环境信息（道路，障碍物）都投影到 Frenet 坐标系，并且道路的中心线被近似弧长参数化，作为规划模块的参考线。在路径搜索步，我们沿着车道采样，采样的位姿组成了有向无环图的节点，然后有向无环图中的边使用螺旋曲线或者多项式曲线连接。运动学约束，动力学约束，障碍物避让约束以及平滑性组成了有向无环图中的边的权重，然后使用动态规划或者

迪杰斯特拉算法在这个非凸的空间中搜索出代价值最小的路径。为了使得生成的路径满足空间光滑性要求，我们将路径优化问题建模成一个非线性规划（nonlinear programming, NLP）问题，这个 NLP 问题的目标函数是平滑性和相对参考线横向偏移的线性组合，约束函数包括偏移量边界约束，曲率约束和避障约束。在速度搜索步，主车辆的轨迹和障碍物预测轨迹都被投影到 ST (station-time graph) 图中，ST 图是一个非凸的二维空间。首先在 ST 图中构建速度 lattice，然后使用时间光滑度，碰撞风险代价项赋予到图中边的权重值之中。然后使用动态规划或迪杰斯特拉算法在图中搜索出分段的速度曲线，此时的速度曲线是由分段线段首尾连接构成的，所以该速度曲线在时间上并不是光滑的，为了改善其光滑性，我们将速度优化问题建模成一个标准的二次规划问题。关于这套框架，会在本章和下一章进行详细的说明。

3.1 路径搜索

路径搜索是，将车辆运动的连续构型空间，给离散化成由运动基元构成的有向无环图，通过图搜索算法来找到代价值最小的路径。这里面比较核心的地方是，怎么样去离散化构型空间，也就是通过怎样的采样方式来提高解的搜索效率；第二个核心的地方是，有向无环图的边，该怎么表征，是用线段，多项式曲线还是螺旋曲线？使得搜索出的解相对比较光滑，并且算法的时间复杂度也能够使得自动驾驶系统可以接受；第三个核心的地方在于，选择什么样的图搜索算法，能够高效地进行图搜索。

3.1.1 路径 lattice 图

图结构是解决路径规划问题的一个很有效的工具，图可以看作是节点集合与边集合的集合 $G = (V, E)$ ，其中，集合 V 中的元素称作节点，集合 E 中的元素称作边，是连接两个节点而形成的路径。图搜索，本质上是对图中的节点和边的遍历，完成对特定节点的识别和查找。对于路径搜索而言，我们采样的位姿 $\mathbf{p}_{pose} = [x, y, \theta, \kappa]^T$ 就构成图中的节点，连接节点之间的边就叫做运动基元或者路径，构成的路径搜索图可以叫做 lattice 图。

为了构建路径 lattice 图，我们需要设计一定的采样规则。合适的采样能够保证车辆的状态空间有良好的离散度和相对高的完备性，本文采取^[44]的采样思想，即沿着参考线的方向，在车辆前方一定距离的状态空间内均匀采样。然而在车辆前方的第一列采样点的位置，^[44]并没有做出说明，如果第一列采样点距离车辆的位置过近，降低了采样的有效性，增加了算法的时间复杂度，

如果过远会降低车辆的避障能力和运动潜力，文献^[61]提出可以对第一列的采样点做出速度自适应采样。根据文献^[40]对控制空间和状态空间采样的研究，我们可以保守地给出第一列采样点在 s 上的位置，并设计了关于最大采样距离的有限状态机模型，总共分为两个场景，一个是视野能够达到纵向方向 100 米且其间没有障碍物，另一个是视野不能达到纵向方向 100 米。关于这两个场景的采样总距离的设计原则是，一旦视野不受阻挡，那么采样总距离为 100 米，如果视野在 100 米受阻挡，那么采样总距离会随着可视距离而改变。对于第二种情形，会有两种可能的子场景，一种子场景纵向方向 100 米以内有障碍物车辆，如图 3-2 所示，另一种子场景是纵向方向上不存在障碍物，但是由于附近障碍物的存在而影响了自己的视野，所以此时向前采样的总距离都不能够达到 100 米，如图 3-3 所示。

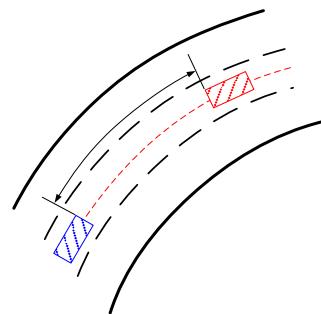


图 3-2 纵向方向上有障碍物时的最大采样距离

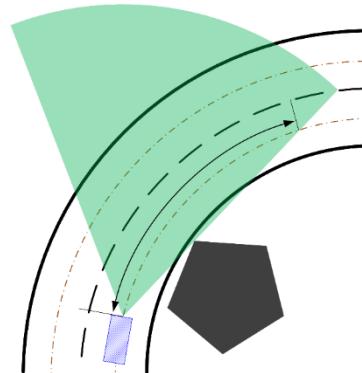


图 3-3 周围障碍物导致的视野限制时的最大采样距离

视野不受限的采样距离 $s_d = s_{\max} = 100.0m$ ；纵向方向上有障碍物的采样距离 $s_d = s_{obs}$ ， s_{obs} 为当前车辆到障碍物的纵向距离；周围障碍物导致的视野限制时的采样距离 $s_d = s_{vis}$ 。综上分析，向前采样的总距离 s_d 函数为

$$s_d = \min\{s_{\max}, s_{obs}, s_{vis}\} \quad (3-1)$$

另外，采样点之间的横向间隔和纵向间隔通常和车辆速度、道路结构、是否

变道等等相关。不妨设纵向采样点数为 m , 横向采样点数目为 n , m 与 n 的值需要调整到一个合适的值。分辨率太高使得维度爆炸增加计算代价, 太低则失去了最优性和降低完备性。所以我们可以给出横向采样步长 Δs 和纵向采样步长 $\Delta \rho$ 的方程

$$\Delta s = \frac{s_d}{m}, \quad \Delta \rho = \frac{w_{lane}}{n} \quad (3-2)$$

其中, w_{lane} 为道路的宽度, 因为每一次采样都是分布在一条车道内的, 当执行换道操作时, 可以同时在三条车道内采样。在这里给出了采样步长, 然而当主车辆和前方车辆过近时, 比如间隔距离为 10 米或 20 米时, 纵向采样点密度 m 的值设置为 5 或者 10 就会浪费大量的计算力, 因为此时 m 的值设置为 1 或者 2 就比较合理。根据经验, 将 m 的值设置成阶梯函数的形式

$$m = \begin{cases} 1, & s_d \leq 10 \\ 2, & s_d \leq 20 \\ 3, & s_d \leq 30 \\ 4, & s_d \leq 40 \\ 5, & s_d \leq 100 \end{cases} \quad (3-3)$$

根据经验, n 的值一般设置为 8~10 比较合适。在 Frenet 坐标系中采样的节点 (位姿) 如图 3 - 4 所示, 然后通过坐标转换可以转变成如图 3 - 5 所示的路径 lattice 图。

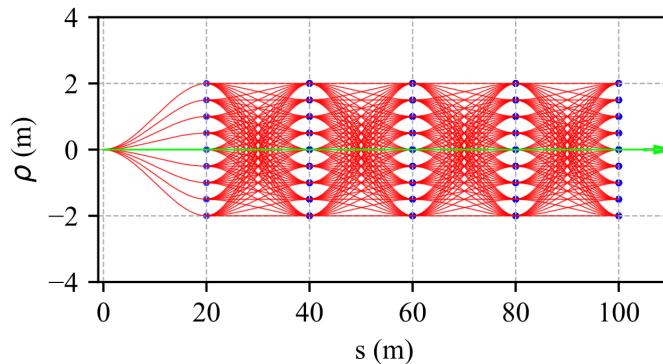


图 3 - 4 Frenet 坐标系中的路径 lattice。蓝色的节点表示采样的位姿, 红色的曲线表示连接节点的三次多项式, 绿色曲线表示车道的参考线。

在 Frenet 坐标系中, lattice 图的边, 也就是运动基元, 是使用三次多项式连接而成的, 采样节点的位姿是 $\mathbf{x} = [s, \rho, \theta_r]^T$, 这里需要强调的是, 采样节点的航向角与对应的参考线上的投影点的航向角是相同的, 也就是说我们尽可能地让行驶路径平行于车道的参考线。当然也可以在 Frenet 坐标系中采样完毕后, 使用 2.2.2 节的坐标转换方程将 $\mathbf{x} = [s, \rho, \theta_r, \kappa]^T$ 转换成 $[x, y, \theta, \kappa]^T$ 的形

式，然后再使用螺旋曲线连接采样的节点，最后也形成笛卡尔坐标系中的路径 lattice 图，然而根据我们的经验，在 Frenet 坐标系中构建 lattice 比在笛卡尔坐标系中构建 lattice 的效率提高了一个数量级。

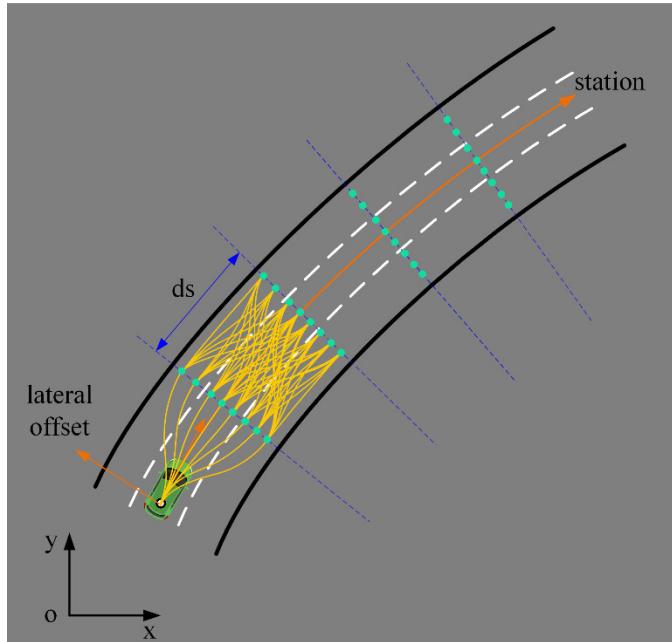


图 3-5 笛卡尔坐标系中的路径 lattice。绿色车辆表示主车辆的当前位姿，绿色节点表示采样的位姿， ds 表示纵向采样步长。

3.1.2 路径 lattice 权重

在构建完路径 lattice 后，需要赋予图中边的权重值。对于图中每一条边的权重，由四方面组成，空间平滑性，跟随车道参考线，静态障碍物碰撞风险，动态障碍物碰撞风险。对于车辆的内部约束来说，比如由车辆的阿克曼转角带来的曲率约束、轮胎的摩擦性能带来的动力学约束以及侧滑约束，这些约束条件并不加在边的权重之中，即使生成的路径不够光滑，甚至不满足运动学约束和动力学约束，此时也忽略这些特殊状态，这也是为了计算效率的考虑，对这些内部约束的计算会耗费大量计算力。并且即使加上这些车辆的内部约束作为 lattice 图的边的权重，生成的路径也是不够光滑的，最后会把 lattice 图搜索生成的解作为路径非线性优化步的初值，最后优化出的轨迹会满足运动学、动力学约束。所以在 lattice 图构建时暂时忽略掉车辆的内部约束是值得的。每一条边的权重为

$$C_{path} = w_{smooth} C_{smooth} + w_{ref} C_{ref} + w_s C_{static} + w_d C_{dyn} \quad (3-4)$$

对于路径权重的计算，我们计算路径上的离散点的代价值，然后对离散

点的代价值进行求和，这样完成对于一条边的代价值的计算，路径的离散点如图 3 - 6 所示。

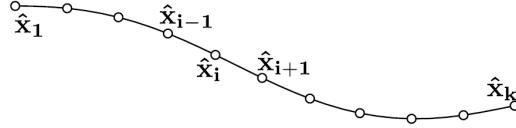


图 3 - 6 运动基元上的离散路点。通过计算这些离散路点的属性，然后将其求和计算出一条运动基元的属性，因为离散路点累计的属性会组成运动基元的属性

对于一条边，均匀采样 N 个点，求出曲率平方的均值，用来表征路径的空间平滑性

$$c_{smooth} = \frac{\sum_{i=0}^N \kappa(s_i)^2}{N} \quad (3-5)$$

在上式中，我们很难显式地表达出曲率关于弧长 s 的方程，为了方便计算，我们使用数值方法来计算曲率。曲率的常规计算方程为

$$\kappa = \frac{x'y'' - y'x''}{\sqrt{(x'^2 + y'^2)^3}} \quad (3-6)$$

为了数值计算曲率，我们使用离散的路点来近似计算曲率，对于离散路点的选取，我们根据路径上采样的路点 $[x_i, y_i]^T$ ，和该路点所在的曲线方程，求出该路点的领域路点 $[x_{i+1}, y_{i+1}]^T, [x_{i+2}, y_{i+2}]^T$ ，这样曲率的数值计算方式为

$$\kappa_i = \frac{(x_{i+1} - x_i)(y_{i+2} - 2y_{i+1} + y_i) - (y_{i+1} - y_i)(x_{i+2} - 2x_{i+1} + x_i)}{\sqrt{((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)^3}}$$

其中，

$$\begin{aligned} x'_i &\approx \frac{x_{i+1} - x_i}{e}, & x''_i &\approx \frac{x_{i+2} - 2x_{i+1} + x_i}{e^2} \\ y'_i &\approx \frac{y_{i+1} - y_i}{e}, & y''_i &\approx \frac{y_{i+2} - 2y_{i+1} + y_i}{e^2} \end{aligned} \quad (3-7)$$

对于车道中的运动基元，我们希望他们尽可能接近车道的参考线，也就是尽可能接近车道的中心线，本文使用路径与车道参考线的偏差平方的均值，用来表征对道路参考线的跟随性能

$$c_{ref} = \frac{1}{N} \sum_{i=0}^N (\rho(s_i) - r(s_i))^2 \quad (3-8)$$

其中， $r(s_i)$ 表示参考线在 Frenet 坐标系的 ρ 值，一般来说， $r(s_i) = 0.0$ 。

对静态的障碍物进行碰撞检查，当障碍物数量比较少时，使用运动基元上离散的路点和障碍物集进行遍历碰撞检查，这种算法的时间复杂度为 $O(NM)$ ， N 是运动基元离散路点的数量， M 是障碍物点的数量；当障碍物比较密集时，时间复杂度就太高了，此时就需要另一种算法了。对于碰撞检查代价值的设计，一种做法是使用布尔值，将 0 或者 1 赋予给一条路径的检查结果，0 表示无碰撞，1 表示碰撞。然而对于都没有发生碰撞的路径集来说，显然距离障碍物更近的路径，碰撞风险的值就越高，所以使用浮点数表示碰撞风险比布尔值表示更合理，此时的重点就是碰撞风险值的计算。因此，本文引入一种碰撞风险指标，使用高斯卷积来计算碰撞风险，这个指标在文章 [49,51] 已经成功应用。

为了防止碰撞，每一条候选路径都要被检查是否通过道路边界，车道线以及障碍物。我们设置候选路径跨越障碍物和道路边界的碰撞检查结果为 1.0，跨越实线的检查结果为 0.5，跨越虚线的检查结果为 0.1，既没有碰撞也没有跨越车道线的检查结果为 0.0，如图 3-7 所示为采样的运动基元。

$$r_{static}[i] = \begin{cases} 1, & \text{与静态障碍物碰撞} \\ 0.5, & \text{跨越实线} \\ 0.2, & \text{跨越虚线} \\ 0, & \text{无碰撞} \end{cases} \quad (3-9)$$

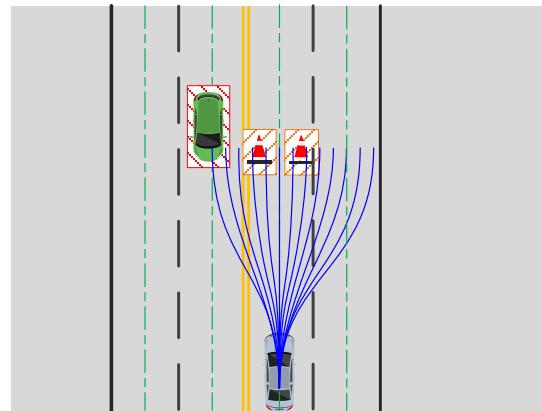


图 3-7 静态障碍物的碰撞检查

碰撞检查的结果对于路径选择来说是最重要的代价项，如果暂时使用上面这些数值作为碰撞风险代价项，根据我们的检查结果，6 号路径的碰撞风险为 0，那么根据总的代价，在下图中可能会选择 6 号候选路径最为最优路径，然而事实上 6 号路径距离障碍物非常近，这会带来非常高的碰撞风险。所以设计一个函数来计算风险值是必要的，根据常识，越靠近障碍物或道路边界的路径的碰撞风险越大。根据 Hu^[51] 的研究，使用离散高斯卷积来计算每

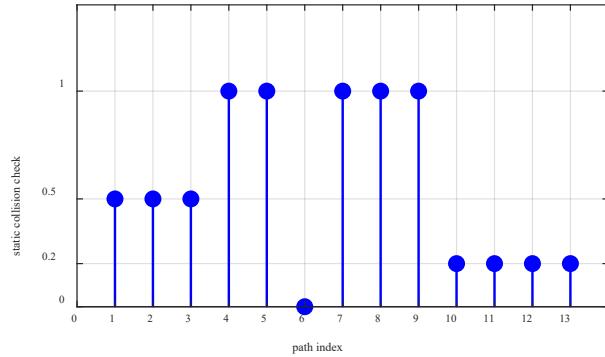


图 3-8 静态障碍物碰撞检查结果

一个候选路径的风险值。卷积在这里的物理意义是，一条路径如果发生碰撞，那么离这条路径越近的路径的碰撞风险也就越大，也就是说，碰撞会波及到附近的路径，带来隐性的风险。我们设计的高斯卷积核为

$$g[i] = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{i^2}{2\sigma^2}\right) \quad (3-10)$$

σ 的大小表示其中一条路径发生碰撞对附近路径的影响程度，该值越大，对附近影响程度越大。离散卷积方程

$$c_{static}[i] = \sum_{k=0}^N r_{static}[k] g[i-k] \quad (3-11)$$

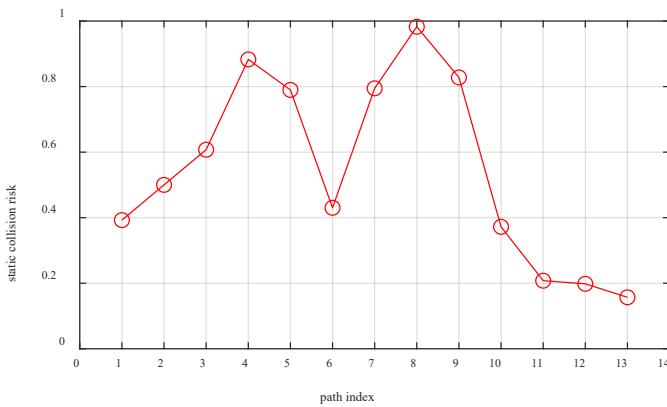


图 3-9 卷积后的路径碰撞风险

根据碰撞风险卷积公式，显示 8 号路径的碰撞风险最大，13 号的最小但也与相邻的 11, 12 号路径的风险值相差无几。在具体的碰撞检查过程中，我们使用 2.3.2 节的知识完成碰撞检查的计算，使用卷积计算的碰撞风险如图 3-9 所示。

对于动态障碍物的碰撞检查，需要将障碍物和主车辆的运动预测方程表

示为 $f(s, \rho, t)$ 的形式，为了方便对动态障碍物引起的碰撞风险进行评估，障碍物的离散时间序列记为 $f_o = [s(t_i), \rho(t_i)]^T$ ，当前车辆的离散时间序列记为 $f_{veh} = [s(t_i), \rho(t_i)]^T$ ，如图 3 - 10 所示。因为本文是解耦的方式做轨迹规划，对于这里的路径，我们使用上一轮规划循环输出的速度曲线，匹配到本次采样的路径上，这样能够得到主车辆采样得到的离散时间序列。

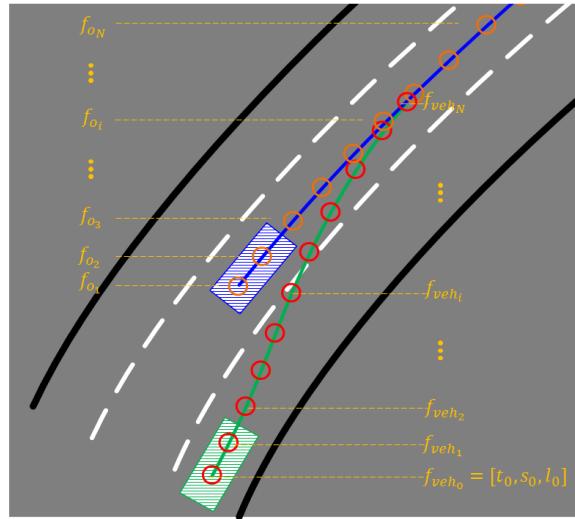


图 3 - 10 动态障碍物离散时间序列

设时间序列 $t_i = t_0 + i\varepsilon, i = 0, 1, \dots, n$ ，障碍物与主车辆与时间对应的位置间的差分需要被计算，差分表征了序列点之间的空间位置关系。时间序列差分的二范数为

$$e_i = \|f_{o_i} - f_{veh_i}\| \quad (3-12)$$

上式详细的计算方式为

$$e_i = \sqrt{(s_{o_i} - s_{veh_i})^2 + (\rho_{o_i} - \rho_{veh_i})^2} \quad (3-13)$$

图 3 - 11 是动态障碍物碰撞检查图，以主车辆后轴中点（蓝色点）为圆心，半径 $R_1 = 2r$ 的圆（ r 为 2.3.2 节中车体包络圆的半径），当时间序列差分的二范数小于 R_1 时，即障碍物时间序列点此时位于绿色区域时，则将主车辆在此时的动态碰撞风险代价值设置为 1.0， r_{dyn} 表示碰撞风险代价值；以主车辆后轴中点（蓝色点）为圆心，半径 $R_2 = 2R$ 的圆（ R 为 2.3.2 节中车体外接圆的半径），当时间序列差分的二范数小于 R_2 时，即障碍物时间序列点此时位于黄色区域时，则将主车辆在此时的动态碰撞风险代价值设置为 0.7；当时间序列差分的二范数大于 R_2 时，即障碍物时间序列点此时位于剩下的空白区域时，

则将主车辆在此时的动态碰撞风险代价值设置为 0.0。关于图 3-11 的动态障碍物碰撞代价值 r_{dyn} 和临界半径 R_1, R_2 的设置，只是一种近似的碰撞检查方式，

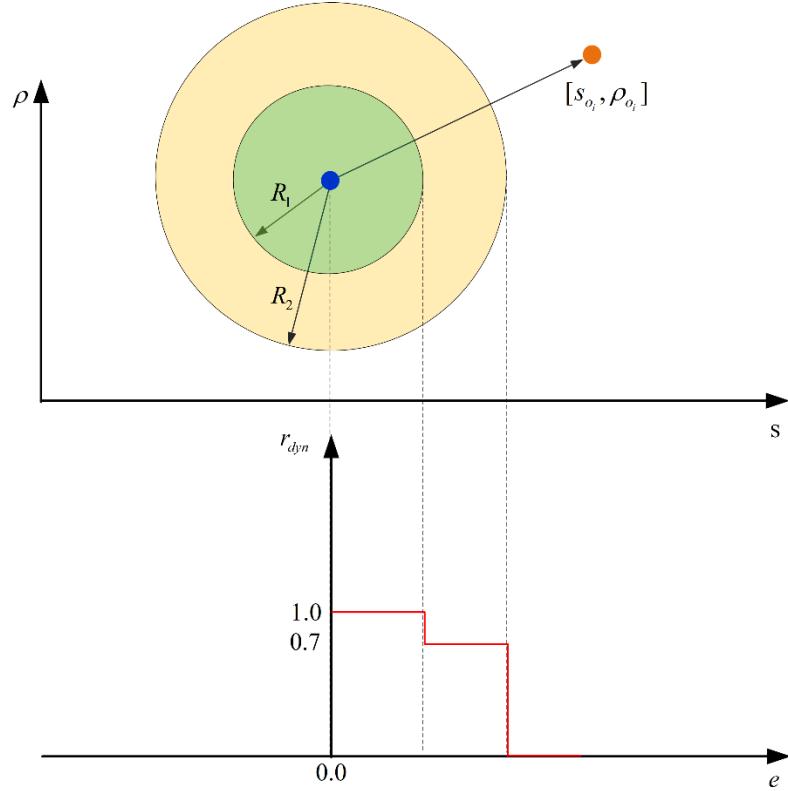


图 3-11 动态障碍物的碰撞检查

因为碰撞距离的计算使用的只是 Frenet 坐标值，而不是真实的笛卡尔坐标值，所以上面计算的时间序列差分的二范数并不是真实的欧式距离，只是一种伪欧式距离，但是根据经验这种伪距离也足够有效。我们给出动态碰撞检查代价值的函数

$$r_{dyn}(e) = \begin{cases} 1.0, & e \leq R_1 \\ 0.7, & e \leq R_2 \\ 0.0, & e > R_2 \end{cases} \quad (3-14)$$

随着时间的升序，算出运动基元累计的动态碰撞检查代价值，如图 3-12 所示。累计的动态碰撞检查代价值的函数为

$$r_{dynamic} = \sum_{k=0}^N r_{dyn}(e_k) \quad (3-15)$$

上式中的 e_k 表示时间戳为 k 时的时间序列差分的二范数，使用式(3-12)计算。同样地，也将动态障碍物碰撞风险进行卷积处理

$$c_{dyn}[i] = \sum_{k=0}^N r_{dynamic}[k]g[i-k] \quad (3-16)$$

上式中, i 为第 i 个采样的运动基元, $c_{dyn}[i]$ 表示第 i 个运动基元的动态碰撞风险指标值, 那么一条轨迹的累计动态碰撞风险计算的示意图如图 3 - 12 所示。

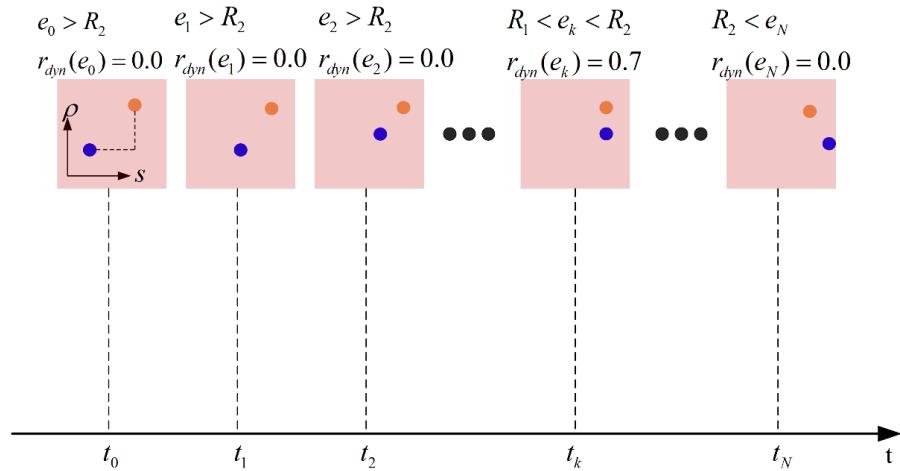


图 3 - 12 动态障碍物碰撞风险计算

3.1.3 Lattice 图的搜索算法

为了在图中搜索出路径, 目前主要有两种主流的算法, 启发式搜索算法和穷举类搜索算法。启发式搜索算法主要包括 A*家族算法, 穷举类搜索算法主要包括动态规划算法 (Dynamic Programming, DP) 和迪杰斯特拉 (Dijkstra's algorithm) 算法。启发式搜索算法主要是在代价函数之中加入启发值, 以加快对节点的搜索, 穷举类算法是对节点的遍历, 直到满足终止条件。对于代价函数中只包含欧氏距离或者曼哈顿距离这种代价项, 上述两类搜索算法都能够在图中搜索出最短路径, 只不过启发式搜索算法有更快的计算效率。对于路径 lattice 搜索, 代价函数是各种代价项的组合, 由于平滑度代价项, 与参考线偏移代价项, 和碰撞风险代价项的存在, 我们很难为 lattice 搜索去设计启发函数。因此, 在构建了 lattice 图, 并且也给图中的边赋予了权重之后, 对于这样的带权网络, 在路径规划领域主要使用穷举类搜索算法。

本文首先介绍动态规划进行路径 lattice 搜索, 动态规划算法是解决最优化问题的一个很优秀的工具, 对于一个大规模问题, 通过分解成多个小规模问题或多阶段问题, 如果子问题具有马尔科夫性, 即未来状态之只和当前状态有关, 而和过去状态无关, 那么我们就可以通过子问题最优, 并且通过向前求解或向后求解达到累计最优解, 那么能够达到累计最优的解就是一个大规模问题的最优解。对于路径搜索而言, 我们可以将一个 lattice 图看作多组节点的组合, 因为 lattice 图是沿着参考线方向采样, 采样 n 组, 那么该优化

问题就可以分为 n 个阶段，然后在每一阶段中有 m 个状态，即参考线法向方向采样 m 个节点。每一对可以连接的节点之间的边会有一个权重，将这个权重添加到动态规划的目标函数中，然后从起点节点开始向前拓展，当拓展到最后一个阶段，找到该阶段中累计代价值最低的节点作为路径规划的终点，最后就可以找到最优路径。具体地，本文给出动态规划在 lattice 图搜索中的贝尔曼方程，

$$T(n) = \min_{\substack{1 \leq j \leq m \\ 1 \leq i \leq m}} (T_j(n-1) + t_{j,i}) \quad (3-17)$$

式中， $T(n)$ 表示 n 阶段的最优解， $T(n-1)$ 表示 $n-1$ 阶段的最优解，然后 j 表示 $n-1$ 阶段的第 j 个状态， i 表示第 n 阶段的第 i 个状态，我们根据贝尔曼递推方程，能够很快地求出最优解。lattice 图是一个非线性的搜索结构，而不是常见的线性结构（向量与链表）和半线性结构（二叉树），所以图中存在很多重复的子问题，对于动态规划算法而言，能够减少重复子问题的计算，大大降低算法的时间复杂度，因为根据马尔科夫性，未来状态只和当前状态有关，所以过去的状态的累计代价不用重复计算。

另一个常用的 lattice 图搜索算法是迪杰斯特拉算法，本质上是一种广度优先搜索算法。算法流程是，从起始节点向外拓展，计算拓展节点的累计代价值 $g(n)$ ，最后拓展到目标节点，然后再进行路径回溯输出最优路径。因为 lattice 图搜索并没有一个明确的目标节点，为了使之适应 lattice 图搜索，我们设置了一个 *TARGET* 表，用于存储沿着参考线采样的最后一组节点，当最后一组节点都被拓展完成并被加入 *TARGET* 表中后，搜索出该表中代价值最低的节点作为目标节点，也就是路径规划的终点，然后继续进行路径回溯。下面给出算法的伪代码，

表 3-1 图搜索的迪杰斯特拉算法伪代码

```

// 维护一个优先级队列OPEN表，用来存储所有拓展到的节点
OPEN表使用起始节点  $n_{start}$  初始化；
分配权重  $g(n_{start}) = 0$ ，其他节点  $g(n) = \infty$  ;
// 进入主循环

while(OPEN) {
    移除OPEN表中代价值最小的节点n;
    if (节点n属于最后一组节点) {
        TARGET.push_back(N);
    }
    从节点n向其邻居拓展;
    对于节点n的一个邻居m,
    if (  $g(m) = \infty$  ) {

```

```

g(m)=g(n)+cost(n,m) ;
push 节点 m 到 OPEN 表;
}
else if (g(m)>g(n)+cost(n,m)) {
    g(m)=g(n)+cost(n,m) ;
}
return false;
// 退出该函数

```

上表中的节点的权重 $g(n)$ 的计算使用上一节中的路径 lattice 图权重的计算方法。实际上，可以将迪杰斯特拉的代价函数写成下面这种形式，

$$f(n) = g(n-1) + cost(n, m) \quad (3-18)$$

由于迪杰斯特拉的贪心策略，每一次选择代价值最低的节点进行拓展，所以对比上式方程和动态规划贝尔曼方程，发现两种算法在本质上是相同的，可以说迪杰斯特拉算法是一种特殊的动态规划类算法，并且可以套用同一种算法框架。为了加速搜索，OPEN 表可以使用优先级队列实现，这样实现时优先级队列会按照某种排序规则自动地将内部元素排序，方便弹出代价值最低的元素，这样就可以免去自己实现时的排序、查找操作，而自己只需要实现插入和删除操作即可。

3.2 路径优化

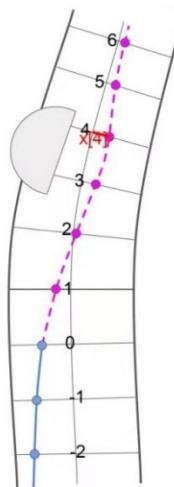


图 3 - 13 路径的横向优化

对于 3.1 节路径搜索给出的路径，在空间上并不是光滑的，如图 3 - 15、图 3 - 16 所示，在每一个运动基元的连接点处，往往会发生曲率的突变，这种曲率不连续的局部路径，在高速行驶时带来了很大的安全隐患，如果依赖

随后的路径跟踪控制器对路径进行修正,这也会带来不小的计算负担。此外,粗糙路径在路径长度项,碰撞风险项,跟随车道参考线项来说并不是最优的,所以为了保证良好的光滑性以及安全性,对路径的优化是必不可少的。

本文设计的路径优化策略是,在避障时我们可以通过横向偏移来完成避障,所以在参考线上均匀地采样路点,使用与路点相互映射的横向偏移量作为优化过程的决策变量,如图 3 - 13 所示,然后将期望的优化性能加入到目标函数中,将路径优化过程中的内部约束和外部约束添加到约束函数中,3.1 节给出的粗糙路径位于最优解的领域,所以通过非线性优化算法迭代几次就能够收敛到最优解。

3.2.1 目标函数

本文将弧长参数,也就是里程 s ,给离散化成步长为 e 的序列,

$$s_i = s_0 + ie, i = 0, 1, \dots, n \quad (3-19)$$

决策变量为参考线的偏移量 $\mathbf{p} = [\rho_0 \quad \rho_1 \quad \rho_2 \quad \dots \quad \rho_{n-1} \quad \rho_n]^T$,此时有性能指标

$$j(\mathbf{p}(s)) = \int_{s_0}^{s_f} L(\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}) ds \quad (3-20)$$

其中,

$$L = w_1 \rho'(s)^2 + w_2 \rho''(s)^2 + w_3 \rho'''(s)^2 + w_4 (\rho(s) - \rho_r(s))^2 \quad (3-21)$$

将上式写成离散求和的形式,

$$j = \sum (w_1 j_{di} + w_2 j_{ddi} + w_3 j_{dddi} + w_4 j_{refi}) e \quad (3-22)$$

设粗糙路径的偏移序列 $\mathbf{p}_r = [\rho_{r0} \quad \rho_{r1} \quad \rho_{r2} \quad \dots \quad \rho_{r(n-1)} \quad \rho_{rn}]^T$,在路径横向优化时,我们希望最后收敛的路径与开始的粗糙路径不要偏离太远,所以将决策变量与粗糙路径的偏差作为一个惩罚项

$$j_{refi} = (\rho_i - \rho_{ri})^2 \quad (3-23)$$

优化路径的平滑性作为一个惩罚项,使用决策变量的一阶导数,二阶导数和三阶导数组成路径的平滑性能指标,一般而言,我们希望路径越平滑越好,此时决策变量的导数的阶数越高,那么路径就越平滑,那么这就意味着路径的航向角、曲率、曲率的导数、曲率导数的导数等等都是连续的,但是阶数越高,决策变量的数量就越大,优化起来的收敛速度就越慢,根据经验,一般目标函数的平滑性项取到决策变量的三阶导数就足够了,此时能够保证

曲率的连续和光滑，并且收敛速度也比较令人满意。下面给出目标函数中平滑性代价项的求解方程，

$$\begin{aligned}\rho_i' &\approx \frac{\rho_{i+1} - \rho_i}{e}, j_{di} = (\rho_i')^2 \\ \rho_i'' &\approx \frac{\rho_{i+2} - 2\rho_{i+1} + \rho_i}{e^2}, j_{ddi} = (\rho_i'')^2 \\ \rho_i''' &\approx \frac{\rho_{i+3} - 3\rho_{i+2} + 3\rho_{i+1} - \rho_i}{e^3}, j_{dddi} = (\rho_i''')^2\end{aligned}\quad (3-24)$$

上式中第一项表示偏移量 ρ 的增量代价项，物理意义是 Frenet 航向角，即希望 Frenet 航向角越小越好，那么路径就能够平行于车道参考线，此时 Frenet 航向角为 0；第二项表征 Frenet 航向角的变化率，第三项表征 Frenet 航向角变化率的变化率，添加这两项到目标函数里即希望 Frenet 航向角的变化程度更小，即对应着笛卡尔坐标系中曲率和曲率的变化率更小，这样优化出的路径就十分平滑。

3.2.2 约束函数

路径优化的约束应该包括内部约束和外部约束，内部约束由车轮的运动学约束、动力学约束带来，通常包括由阿克曼转角几何带来的曲率约束，动力学限制带来的最大速度、最大加速度、最大减速度、最大加加速度（jerk）约束。外部约束通常包括行驶通道和障碍物避免约束，是由行驶环境和行驶规则带来的约束就成为外部约束。内部约束的曲率约束方程为

$$|\kappa_i| \leq \kappa_{\text{lim}} \quad (3-25)$$

因为曲率的方程不好获得，本文通过数值求解的方式得到某一个路点处的曲率，如式 (3-26) 所示，

$$\kappa_i = \frac{(x_{i+1} - x_i)(y_{i+2} - 2y_{i+1} + y_i) - (y_{i+1} - y_i)(x_{i+2} - 2x_{i+1} + x_i)}{\sqrt{((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)^3}} \quad (3-26)$$

外部约束的障碍物避免约束可以写为

$$\|\mathbf{X}_i - \mathbf{X}_o\| > r \quad (3-27)$$

上式中， \mathbf{X}_i 表示车体某一个磁盘圆心的 Frenet 坐标 $[s_i, \rho_i]^T$ ， \mathbf{X}_o 表示某一个障碍物的 Frenet 坐标。

对于一个障碍物，靠近车辆后轴的磁盘圆心和障碍物避免的约束方程为

$$(s_i - s_j)^2 + (\rho_i - \rho_j)^2 > r^2 \quad (3-28)$$

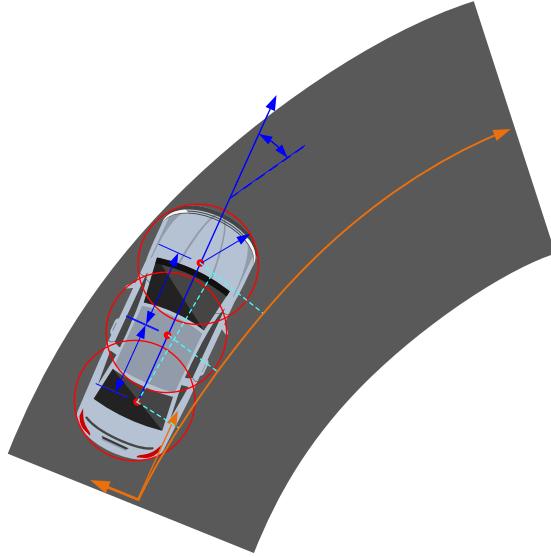


图 3 - 14 车体包围圆与在参考线上的投影

需要进行剩余两个磁盘圆的约束计算，后轴中心圆的笛卡尔坐标系坐标

$$\begin{aligned}x_i(s_i, \rho_i) &= x_r(s_i) + \rho_i \cos(\theta_{ri} + \frac{\pi}{2}) \\y_i(s_i, \rho_i) &= y_r(s_i) + \rho_i \sin(\theta_{ri} + \frac{\pi}{2})\end{aligned}\quad (3-29)$$

剩余两个圆的圆心坐标

$$\begin{aligned}x_{2i} &= x_i + d \cos \theta_i, \quad y_{2i} = y_i + d \sin \theta_i \\x_{3i} &= x_i + 2d \cos \theta_i, \quad y_{3i} = y_i + 2d \sin \theta_i\end{aligned}\quad (3-30)$$

此时需要投影到 frenet 坐标系，使用 2.2 节中的公式来计算，然而这种坐标之间的转换效率并不高，对于磁盘坐标的频繁地转换会给规划模块带来大量的计算负担。所以本文开发了一种近似的磁盘 Frenet 坐标计算方法，如下所示，

$$\begin{aligned}s_{2i} &= s_i + d \cos(\theta(s_i) - \theta_r(s_i)), \quad \rho_{2i} = \rho_i + d \sin(\theta(s_i) - \theta_r(s_i)) \\s_{3i} &= s_{2i} + d \cos(\theta(s_i) - \theta_r(s_{2i})), \quad \rho_{3i} = \rho_{2i} + d \sin(\theta(s_i) - \theta_r(s_{2i}))\end{aligned}\quad (3-31)$$

那么三圆碰撞避免约束就可以写为

$$\begin{aligned}(s_i - s_j)^2 + (\rho_i - \rho_j)^2 &> r^2 \\(s_{2i} - s_j)^2 + (\rho_{2i} - \rho_j)^2 &> r^2 \\(s_{3i} - s_j)^2 + (\rho_{3i} - \rho_j)^2 &> r^2\end{aligned}\quad (3-32)$$

上式中， $[s_j, \rho_j]^T$ 表示第 j 个障碍物的 Frenet 坐标。如果障碍物的数量为 n_o ，那么障碍物避免形成的约束方程的数量为 $3n_o(n+1)$ 。外部约束的行驶通道边界约束可以表达为

$$\rho_{\min} \leq \rho_i \leq \rho_{\max} \quad (3-33)$$

上式中, ρ_{\min} 表示车道的右边界约束, ρ_{\max} 为车道的左边界约束, 这样车道边界组成的不等式约束方程的数量为 $2(n+1)$ 。对于端点约束, 起点条件 $[x_0 \ y_0 \ s_0 \ \rho_0 \ \theta_0 \ \kappa_0]$, 所以形成约束为

$$\begin{aligned} \tan \theta_{s_0} &= \frac{\rho_1 - \rho_0}{s_1 - s_0} \\ \kappa_0 &= \frac{(x_1 - x_0)(y_2 - 2y_1 + y_0) - (y_1 - y_0)(x_2 - 2x_1 + x_0)}{\sqrt{((x_1 - x_0)^2 + (y_1 - y_0)^2)^3}} \end{aligned} \quad (3-34)$$

根据起始条件, 有

$$\begin{aligned} \rho_1 &= e \tan(\theta_0 - \theta_{r0}) + \rho_0 \\ \rho_2 &= \frac{\kappa_0 \sqrt{((x_1 - x_0)^2 + (y_1 - y_0)^2)^3} - (x_1 - x_0)(y_r(s_2) - 2y_1 + y_0) + (y_1 - y_0)(x_r(s_2) - 2x_1 + x_0)}{(x_1 - x_0) \sin(\theta_r(s_2) + \frac{\pi}{2}) - (y_1 - y_0) \cos(\theta_r(s_2) + \frac{\pi}{2})} \end{aligned} \quad (3-35)$$

最终, 优化粗糙路径的问题被转换成一个带约束的非线性优化问题, 并且约束包含强非线性的等式约束和不等式约束。为了表达上的方便, 我们将上面的目标函数和约束函数写成下面的形式,

$$\begin{aligned} \min \ j(\boldsymbol{\rho}) &= \sum_{i=1}^{n-3} L(\rho_i, \dot{\rho}_i, \ddot{\rho}_i) e \\ \text{s.t. } \mathbf{Aeq} \cdot \boldsymbol{\rho} &= \mathbf{Beq} \\ \mathbf{C}(\boldsymbol{\rho}) &\leq \mathbf{0} \\ \mathbf{LB} &\leq \boldsymbol{\rho} \leq \mathbf{UB} \end{aligned} \quad (3-36)$$

上式中约束函数中的等式约束是由端点条件形成的, 并且只有起始端点条件, 在本文中终点的位姿无法确定, 即使我们希望终点处的航向角平行于参考线, 曲率也和参考线有 2.2.2 节中给出的函数关系, 也不必将这个条件强加到约束函数里, 因为这种期望是可满足也可不满足的。 $\mathbf{C}(\boldsymbol{\rho}) \leq \mathbf{0}$ 是由曲率约束和避障约束形成的, $\mathbf{LB} \leq \boldsymbol{\rho} \leq \mathbf{UB}$ 是由决策变量的边界约束形成的, 决策变量的方程为 $\boldsymbol{\rho} = [\rho_0, \dots, \rho_n]^T$ 。

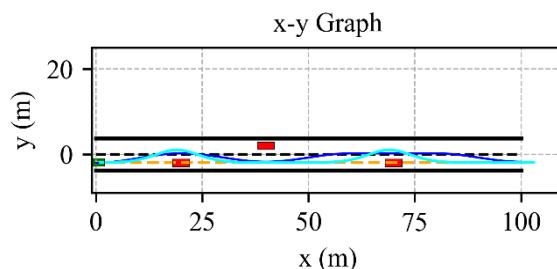


图 3-15 结构化道路上路径优化。蓝色曲线是 lattice 搜索出的粗糙路径, 青色曲线

表示优化器生成的路径，橙色虚线表示右侧车道的参考线，三个红色矩形表示障碍物

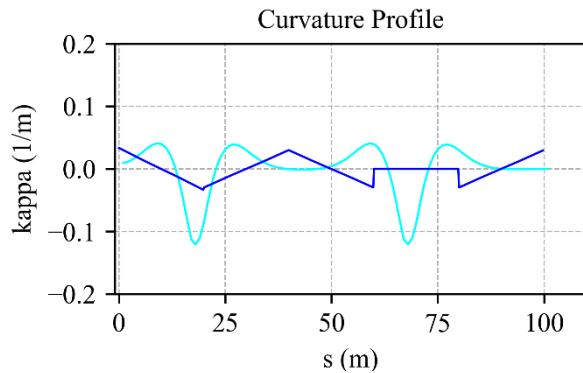


图 3 - 16 路径曲率图。蓝色曲线表示粗糙路径曲率，青色曲线表示优化路径曲率

在式 (3-36) 中，目标函数是二次可微分的关于决策变量平方的形式，约束函数是由强非线性的，所以选择一个优化算法来求解这个 NLP 问题，图 3 - 15、图 3 - 16 显示了有三个障碍物的直线路段的路径搜索和路径优化结果，看起来两条路径都是连续而光滑的，然而根据曲率图，粗糙路径曲率出现了三个跳变点，而优化路径的曲率是光滑的，说明经过优化，路径的平滑性会得到良好的改善。

3.2.3 非线性优化求解器

上一节将路径优化问题建模成了一个 NLP 问题，为了求解这个问题，本节会对优化做了基本的阐述。面对无约束优化问题，一般采用基于梯度的方法来求解，而对于带约束的优化问题，一般会引入一种函数，将带约束优化问题转换成无约束优化问题，从而进入迭代求解步。将惩罚项加入到目标函数中，组成一个新的目标函数，那么原问题就构成了一个新问题，惩罚项用来描述一个解对约束的破坏程度，破坏越严重，惩罚项越大，那么越偏离原问题的可行域，同时也越不能称为新问题的最优解，这种思想就是内点法的体现。通过内点法，可以比较快速地找到 NLP 问题的局部最优解。

另一种 NLP 问题的求解思路是序列二次规划 (sequential quadratic programming, SQP) 方法，该方法使用一系列二次规划问题的解来逼近原问题的解，如今，SQP 有着众多的改进版本和多样的实现方式，但是使用一系列二次规划子问题来逼近原始问题的思想不变，被公认为是求解 NLP 问题的最优秀的算法之一，在工业界和学术界有着广泛的使用。为了求解路径优化问题，我们将优化方程写成下面这种形式，

$$\begin{aligned}
 & \min f(\mathbf{x}) \\
 & s.t. \quad \mathbf{c}_E(\mathbf{x}) = 0 \\
 & \quad \mathbf{c}_I(\mathbf{x}) \geq 0
 \end{aligned} \tag{3-37}$$

$\mathbf{c}_E(\mathbf{x})$ 是等式约束, $\mathbf{c}_I(\mathbf{x})$ 是不等式约束, 写出上面优化问题的拉格朗日函数

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}_E(\mathbf{x}) - \boldsymbol{\mu}^T \mathbf{c}_I(\mathbf{x}) \tag{3-38}$$

再分别求出拉格朗日函数的雅可比矩阵和海塞矩阵, 然后对雅可比矩阵中的部分元素和约束函数进行一阶泰勒展开, 得到线性化的函数, 这时可以使用线性化的 Karush-Kuhn-Tucker 条件, 简称 KKT 条件, 去逼近真实的 KKT 条件, 并且解线性化的 KKT 条件相当于解二次规划问题, 这样迭代求解线性化 KKT 条件直到非线性规划问题的解收敛。本文给出路径优化过程中的 SQP 方法的计算步骤,

- [1] 已知目标函数 $f(\mathbf{x})$, 约束函数 $\mathbf{c}_E(\mathbf{x}), \mathbf{c}_I(\mathbf{x})$, 设置初值 $\mathbf{z}^{(0)} = (\mathbf{x}^{(0)}, \boldsymbol{\lambda}^{(0)}, \boldsymbol{\mu}^{(0)})$, $\mathbf{x}^{(0)}$ = 粗糙路径, 设定解的精度 tol, 设 $k=0$;
- [2] 计算目标函数梯度值 $\mathbf{g}(\mathbf{x}^{(k)})$ 、约束函数值 $\mathbf{c}_E(\mathbf{x}^{(k)})$ 、 $\mathbf{c}_I(\mathbf{x}^{(k)})$, 并计算约束函数的雅可比矩阵 $J_E(\mathbf{x}^{(k)}), J_I(\mathbf{x}^{(k)})$, 计算拉格朗日函数的海塞矩阵值;
- [3] 解二次规划子问题, 获得牛顿方向 $\boldsymbol{\delta}_x^{(k)}$, 以及下一个迭代点 $\mathbf{x}^{(k+1)}$ 的拉格朗日乘子向量 $\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\mu}^{(k+1)}$;
- [4] 令 $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}_x^{(k)}$ 并计算 $f(\mathbf{x}^{(k+1)})$, 若满足 $\|\boldsymbol{\delta}_x^{(k)}\| < tol$, 则停止迭代, 输出 $\mathbf{x}^{(k+1)}$ 作为问题的最优解, 否则, $k++$, 转到步骤[2]。

由于本文只是使用非线性优化求解器来解决路径优化问题, 故不会详细地介绍求解器的算法原理以及收敛性, 最后我们测试了一些非线性优化求解器在求解路径优化这种中等规模问题上的性能。下面给出各个求解器的时间性能表现,

表 3-2 路径优化步中的各个优化算法的时间性能

算法	运行时间 (s)	迭代次数
SQP	4.5036	48
SQP-LEGACY	3.2448	48
积极集法	21.0768	58
内点法	0.0982	12

在测试中，测试道路为 100 米的直道，包含三个静态障碍物，我们设定了步长 e 为 0.01 米，所以有 100 个决策变量，大约 2200 个约束函数，算法的时间性能如表 3-2 所示，由于内点法解决这个中等规模优化问题的效率，本文选择了内点法来优化粗糙路径。本文在试验了 SQP, SQP-LEGACY, 有效集，内点法，并且也测试了多种优化库之后，选中了 CasADi^[76]中的 IPOPT^[77]作为本文的内点法求解器。CasADi 是一种开源软件工具，用于常见的和最优控制中的数值优化，尤其是涉及到微分方程的优化，该项目由 Joel Andersson 和 Joris Gills 启动。使用该软件，可以快速生成函数的梯度信息，快速解决非线性方程寻根问题，解决非线性规划问题、二次规划问题以及最优控制问题。CasADi 提供了丰富的编程接口，也支持多种编程语言，包括 MATLAB, Python, C++，其中 Python 接口更稳定且文档更丰富。CasADi 留有一个接口用来调用多种 NLP 求解器，包含 SQP, IPOPT, SNOPT, BlockSQP, WORHP, WNITRO 等求解器，其中最受欢迎的是 IPOPT 求解器，这是 CasADi 中包含的一种开源的原始对偶内点法，只要根据 CasADi 提供的目标函数接口和约束函数接口，选择一个求解器进行迭代求解即可。IPOPT¹是用于大规模非线性优化的软件库，它可以很方便地解决本文中如式（3-36）所示的有强非线性约束的路径优化问题，其源代码使用 C++语言编写，发行版可用于生成可链接到自己的 C++, C, Fortran 或 Java 代码的库，并可以跨平台使用。本文由作者所写的路径优化源代码可以在作者个人 Github 账户²上下载。

¹ <https://github.com/coin-or/Ipopt>

² <https://github.com/yangmingustb/PTPSim>

4 速度规划

对于结构化道路上的轨迹规划而言，其实路径规划就是轨迹的横向规划，而速度规划就是轨迹的纵向规划，横向规划主要考虑的是在满足运动学、动力学约束的情况下在空间上躲避障碍物，纵向规划是在满足运动学、动力学约束的情况下在时间上躲避障碍物，本质上，一个是在 spatial 空间上做搜索和优化，一个是在 temporal 空间上做搜索和优化。和路径规划一样，速度规划也包括前端和后端，分别是速度曲线搜索和速度曲线优化，速度曲线搜索是在非凸的解空间快速生成一个粗糙的速度解，速度曲线优化是让粗糙速度解变得更加平滑，满足运动学、动力学约束。

4.1 速度曲线搜索

对于无人车而言，速度规划和路径规划一样重要，上一秒在一个位置存在障碍物，下一秒可能就不存在了，并且车辆的速度较快，一秒之间就能前进十米甚至数十米，所以无人车辆的安全性能和速度规划器的性能息息相关。一般而言，规划的速度决定于道路速度限制，道路结构，以及其他交通参与者（车，人，障碍物）等信息，同时在 Station-Time 图或者 Station-Speed 图中，障碍物的存在往往会使可行域是一个非凸可行域，所以实时找到最优解是比较困难的。现在在速度规划中有两个常用的空间，就是上面提到的里程-时间（Station-Time, ST）空间和里程-速度（Station-Speed, SS）空间，针对这两个速度搜索空间，目前有两种速度规划的前端方法，下两小节会分别加以介绍。

4.1.1 里程-速度空间采样

对于里程-速度空间，在该空间中均匀地采样一些状态，然后使用最优控制方法来连接这些节点，这本质上是将主车辆的速度空间给离散化。在采样时有较低的分辨率，能够搜索出粗糙解，但是可能会导致大幅度地加速减速，会影响轨迹的平滑性，同时也会更可能带来次优性；较密集地采样则会平缓地加速减速，但在计算效率上来说则是一个挑战。为了计算效率上的考虑，同时为了显式地考虑障碍物速度和道路速度限制信息，需要在速度空间和纵向里程空间里合适地采样，如图 4-1 所示，前面纵向里程间距为 Δs ，采样节点数为 m 列， n_{speed} 行， $s_{horizon}$ 为纵向采样总里程， v_{lim} 为速度限制

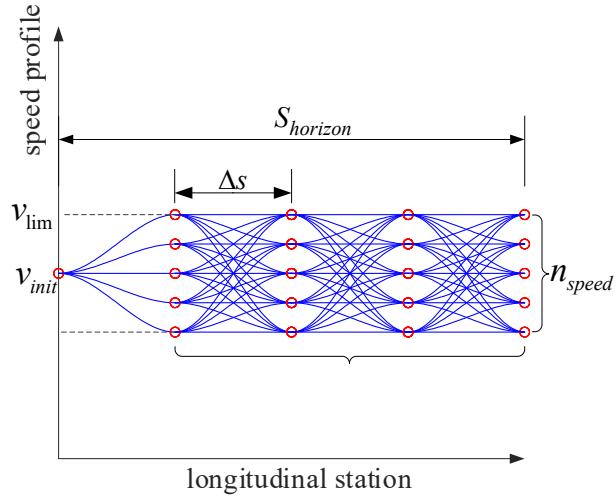


图 4-1 Station-Speed 空间采样

纵向里程采样点的坐标为

$$s_i = s_0 + i\Delta s, i = 0, 1, 2, \dots, m \quad (4-1)$$

速度采样点的坐标设计为

$$v_i = \frac{v_{\text{lim}} - v_{\text{init}}}{n_{\text{speed}} - 1} i, i = 0, 1, 2, \dots, n_{\text{speed}} - 1 \quad (4-2)$$

车辆运动的纵向运动学关系地状态转移方程为

$$\begin{aligned} \dot{s} &= v(t) \\ \dot{v} &= a(t) \end{aligned} \quad (4-3)$$

可以写成标准形式,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \quad (4-4)$$

所以可以将系统的控制量设定为加速度, 将控制量用二阶多项式方程表示

$$\mathbf{u} = a(t) = \rho_1 + 2\rho_2(t - t_0) + 3\rho_3(t - t_0)^2 \quad (4-5)$$

系统的初始和终端状态约束为

$$\begin{aligned} \mathbf{x}_0 &= (s_0, v_0, a_0)^T \\ \mathbf{x}_b &= (s_b, v_b)^T \end{aligned} \quad (4-6)$$

将控制量使用参数化方程表示

$$\mathbf{u}(t) = \mathbf{u}(\mathbf{p}, t) \quad (4-7)$$

$$\mathbf{p} = [\rho_1, \rho_2, \rho_3]^T \quad (4-8)$$

那么状态转移方程可以写作

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, \mathbf{u}, t) = f(\mathbf{p}, t) \quad (4-9)$$

为了满足边值条件约束，系统根据所给的参数向前模拟得到的终端条件为

$$\mathbf{x}_f = \mathbf{x}(t_0) + \int_{t_0}^{t_f} f(\mathbf{p}, t) dt \quad (4-10)$$

模拟得到的终端条件需要与真实端点条件相同

$$\mathbf{x}_f = \mathbf{x}_b \quad (4-11)$$

因为 t_f 的值也是未知的，所以也将其加入到参数向量中，有

$$\mathbf{q} = [\mathbf{p}^T, t_f]^T \quad (4-12)$$

如果模拟得到的终端状态与真实终端状态不同，我们使用牛顿法来迭代修正这个误差 $\mathbf{g}(\mathbf{q})$ ，此时可以写作

$$\mathbf{g}(\mathbf{q}) = \mathbf{x}_f - \mathbf{x}_b = 0 \quad (4-13)$$

根据参数向量，我们可以写出系统的状态表达方程

$$\begin{aligned} v(t) &= \rho_0 + \rho_1(t - t_0) + \rho_2(t - t_0)^2 + \rho_3(t - t_0)^3 \\ s &= s_0 + \int_{t_0}^t v(t) dt \end{aligned} \quad (4-14)$$

其中， s 可以写成多项式表达的解析解的形式

$$s = v_0(t - t_0) + \frac{1}{2}\rho_1(t - t_0)^2 + \frac{1}{3}\rho_2(t - t_0)^3 + \frac{1}{4}\rho_3(t - t_0)^4 + s_0 \quad (4-15)$$

由初始条件可得

$$\rho_0 = v_0, \quad \rho_1 = a_0 \quad (4-16)$$

因为控制量的两个参数是已知的，所以此时多项式方程的参数向量可以写为

$$\mathbf{q} = [\rho_2, \rho_3, t_f]^T \quad (4-17)$$

参数向量含有三个未知数，系统的状态转移和参数向量有关，有模拟的终端状态，

$$\begin{aligned} a_f &= \rho_1 + 2\rho_2(t_f - t_0) + 3\rho_3(t_f - t_0)^2 \\ v_f &= \rho_0 + \rho_1(t_f - t_0) + \rho_2(t_f - t_0)^2 + \rho_3(t_f - t_0)^3 \\ s_f &= s_0 + \int_{t_0}^{t_f} v(t) dt \end{aligned} \quad (4-18)$$

系统的状态量和控制量完全由参数向量 \mathbf{q} 决定，写出边值约束状态方程，

$$\begin{aligned} g_1(\mathbf{q}) - s_f &= 0 \\ g_2(\mathbf{q}) - v_f &= 0 \\ g_3(\mathbf{q}) - a_f &= 0 \end{aligned} \quad (4-19)$$

端点条件是关于参数向量的多项式函数，我们可以求出这个非线性系统的边值条件约束方程关于参数向量的雅可比矩阵

$$\mathbf{J} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial s}{\partial \rho_2} & \frac{\partial s}{\partial \rho_3} & \frac{\partial s}{\partial t_f} \\ \frac{\partial v}{\partial \rho_2} & \frac{\partial v}{\partial \rho_3} & \frac{\partial v}{\partial t_f} \\ \frac{\partial a}{\partial \rho_2} & \frac{\partial a}{\partial \rho_3} & \frac{\partial a}{\partial t_f} \end{bmatrix} \quad (4-20)$$

端点误差的雅可比矩阵可以使用解析解表达为

$$\mathbf{J} = \begin{bmatrix} \frac{1}{3}(t_f - t_0)^3 & \frac{1}{4}(t_f - t_0)^4 & v_f \\ (t_f - t_0)^2 & (t_f - t_0)^3 & a_f \\ 2(t_f - t_0) & 3(t_f - t_0)^2 & 2\rho_2 t_f + 6\rho_3(t_f - t_0) \end{bmatrix} \quad (4-21)$$

根据牛顿法，写出参数向量的迭代方程

$$\mathbf{J}\Delta\mathbf{q} = \mathbf{x}_b - \mathbf{x}_f \quad (4-22)$$

上式可以转换为

$$\mathbf{q}_{k+1} - \mathbf{q}_k = \Delta\mathbf{q} = -\mathbf{J}^{-1}\mathbf{g} \quad (4-23)$$

本文设计的参数向量的初值为，

$$\mathbf{q}_{init} = [0, 0, \frac{2(s_b - s_0)}{v_0 + v_b} + t_0]^T \quad (4-24)$$

通过在里程-速度坐标系中采样状态 $[s_i, v_i]$ ，然后使用最优控制的方法来连接采样状态，并且使用牛顿法来迭代控制量的参数，最终能够快速收敛，使得速度曲线能够从起点采样状态连接到终点采样状态。

4.1.2 里程-时间空间采样

速度规划属于一个非凸优化问题，为了防止陷入局部最小，通常也会在 ST 状态空间中构建 lattice 图用于搜索，这样也能够大大提高计算效率。Werling^[25,65]在 ST 空间中，采样 ST 值，再以 T 为自变量，使用某种曲线连接采样节点，作为速度基元；百度^[61]也在 ST 空间中均匀采样，使用线段连接

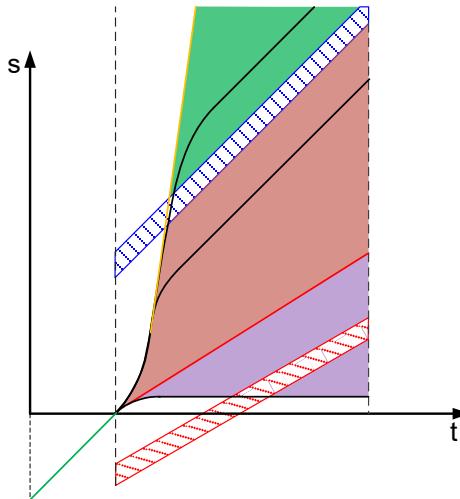


图 4-2 速度空间非凸可行域

采样节点。在 ST 空间采样比较方便的地方在于，能够将障碍物信息映射到 ST 图，能够显式地看到障碍物纵向里程和时间的关系，规划结果也显式地反映了车辆和移动障碍物在 ST 图中的相对位置关系，能够看出是超车、跟随还是停车行为。如图 4-2 所示，是由里程 s -时间 t 组成的速度空间，图中绿色曲线表示车辆在过去的轨迹，然后和时间轴的交点是当前时刻，红色四边形是后方车辆，蓝色四边形是前方车辆，紫色区域的下界，表示主车辆以最大减速速度减速至停止形成的曲线，红色区域的左边界表示车辆以最大加速度加速，直到达到最大速度时车辆进入稳定速度行驶而形成的曲线。

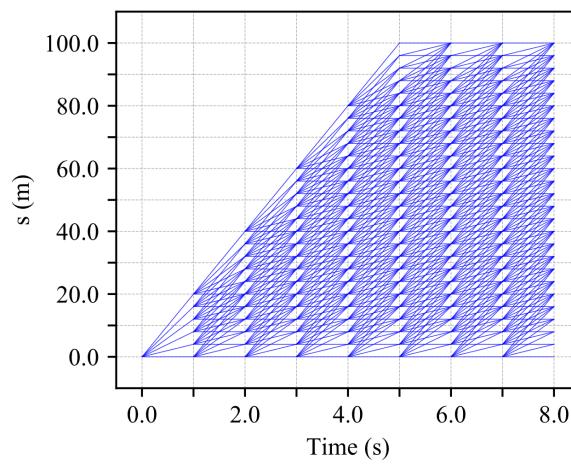


图 4-3 ST 图中的速度 lattice

为了在 ST 图中规划并且保证有足够的规划时域，所以会在图中采样多组状态点，和 Werling 的方法不同，本文并不会采样 s', s'' 的状态，只采样 $[s, t]$ ，这种通过多组状态之间的组合可以保证有足够长的规划时域。设置采

样的时间间隔为 h ，里程间隔为 Δs ，图 4-3 所示的采用特征为 $h = 1.0s, \Delta s = 4.0m$ ，对于图 4-4 所示的采样特征为 $h = 0.5s, \Delta s = 2.0m$ 。

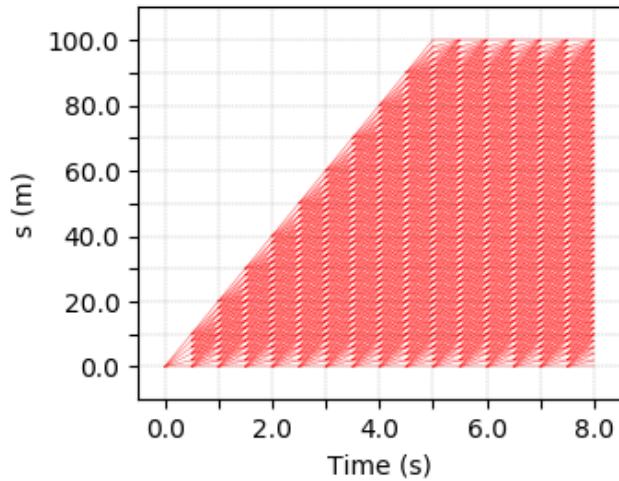


图 4-4 ST 图中的速度 lattice

采样之后，通过直线段来连接两个状态，这样就会在整个 ST 空间中生成一个有向无环图，然后将图的边赋予一定的权重，再通过一个图搜索算法生成粗糙的速度曲线，这样生成的曲线尽可能的平滑，且能够躲避障碍物。

4.1.3 速度 lattice 权重设计与速度搜索

为了能够在 lattice 图中生成粗糙的速度解，需要在设计速度 lattice 中的边的权重时，考虑运动学、动力学约束、安全性、合法性、舒适性等因素。本文设计的权重由四个代价项线性组合而成，分别是与参考速度的偏差，加速度，加加速度，障碍物碰撞风险。参考速度一般和前车速度，交通规则、道路结构相关，一般而言在弯道，路口等地方设置较低的参考速度，在决定跟车时以前车的速度作为参考速度，关于参考速度的设置一般和决策层有关，取决于决策层识别出的当前行驶环境和需要切换的行为模式。加速度和加加速度代价项代表的是速度曲线的平滑性，即我们认为一条轨迹，如果加速度和加加速度越小，那么该轨迹就越平滑，所以加速度和加加速度越大的轨迹就会有更大的代价项。最后增加了一个障碍物碰撞风险代价项来保证车辆的安全性，即我们认为，两个物体在时空上的距离越近，那么产生的碰撞风险越高。速度、加速度和加加速度可以由里程相对时间的差分来近似计算，

$$s'_i = v_i \approx \frac{s_i - s_{i-1}}{h}, \quad s''_i = a_i \approx \frac{s_i - 2s_{i-1} + s_{i-2}}{h^2}, \quad s'''_i = j_i \approx \frac{s_i - 3s_{i-1} + 3s_{i-2} - s_{i-3}}{h^3}$$

(4-25)

给出速度 lattice 的权重函数为

$$c_{speed} = c_{ref} + c_{acc} + c_{jerk} + c_{obs} \quad (4-26)$$

与参考速度的偏差的惩罚项 c_{ref} 表示为

$$c_{ref} = w_{ref} (s'_i - v_{ref})^2 \quad (4-27)$$

有上面这个代价项，意味着我们希望生成的速度曲线不要偏离参考速度太远， s'_i 表示 lattice 图中速度曲线上第 i 点处的速度， v_{ref} 表示当前的参考速度。加速度 c_{acc} 的惩罚项为

$$c_{acc} = w_{acc} (s''_i)^2 \quad (4-28)$$

加加速度 c_{jerk} 的惩罚项表示为

$$c_{jerk} = w_{jerk} (s'''_i)^2 \quad (4-29)$$

至于碰撞风险惩罚项 c_{obs} ，在动态场景中，碰撞风险指标通常包括 Time-To-Collision(TTC)^[78]，Distance-To-Collision(DTC)^[16] 和 Time-To-React(TTR)^[79]。本文联合使用车辆的物理模型和曲线车道模型，来预测车辆的短期轨迹，基于自行车模型和道路结构，向前模拟的短期轨迹被投影到 s-t 图中，尽管这个预测过程没有考虑到车辆运动的不确定性和车辆模型误差，但是在短期内依然比较有效，并且相比其他的预测模型，基于物理的预测模型有良好的计算效率，适合实时计算。基于投影的轨迹，本文使用 DTC 作为我们的风险指标，在同一时间戳，我们计算当前车辆到障碍物的最小距离作为 DTC 的值， c_{obs} 的表达式为

$$c_{obs} = \frac{w_{obs}}{DTC + \delta} \quad (4-30)$$

其中 δ 是非常小的量，通过可以设置为 0.001，主要是防止 DTC 的值为 0 的时候导致的 c_{obs} 为无穷大，这对计算机计算来说是一个麻烦，所以引入了这个极小量。

如图 4-5 所示，蓝色圆圈表示采样节点，红色线段表示连接节点的边，从起始节点采样 6 个子节点后，根据权重会选择一个节点，然后再拓展当前节点，搜索当前节点的子节点，当整个速度 lattice 图遍历完成后，也就能够搜索出在 lattice 图中的最优解。如图 4-6 所示，红色曲线表示当前搜索的最优化速度曲线。对速度 lattice 的搜索，也因为一些代价项的存在而无法设计启发函数，所以也选择穷举类算法(动态规划或迪杰斯特拉算法)作为速度 lattice 图的搜索算法。对于 SS 空间采样和 ST 空间采样，前者生成的速度曲线更加平滑，而后者生成速度曲线的计算复杂度更低，一般地，都是用 ST 空间搜索

作为速度规划的前端算法。

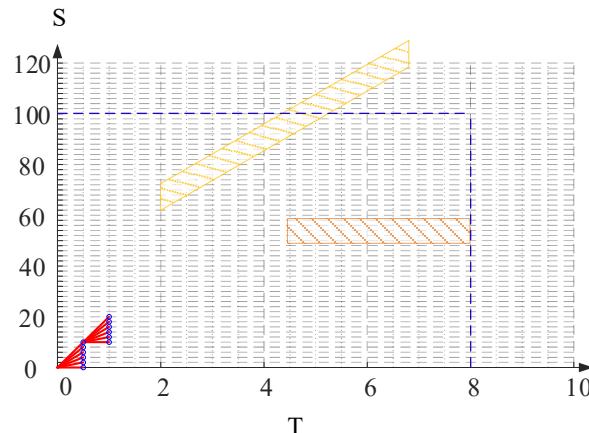


图 4-5 Station-Time 图中的速度采样过程

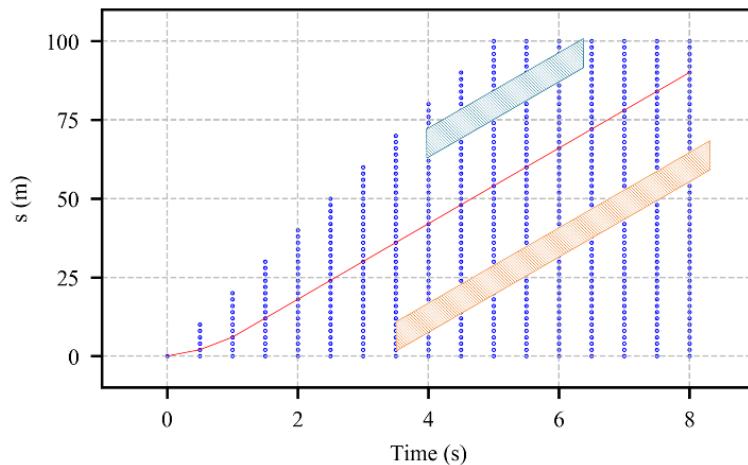


图 4-6 ST 图中生成的粗糙速度曲线，蓝色点表示采样节点，两个平行四边形是障碍物轨迹，红色曲线是搜索的速度曲线

4.2 速度曲线优化

对于速度规划的前端生成的速度曲线，其中的加速度以及加加速度曲线往往不够平滑，甚至是不连续的，为了改善轨迹的时间平滑性，我们往往会将速度优化问题建模成一个非线性优化问题，这一步作为速度规划的后端来生成满足各种约束的足够平滑的轨迹。对于速度优化来说，因为纵向速度是里程 s 和时间戳 t 一对一的映射而生成的，所以通常存在两种优化策略，一种是固定离散的时间戳 t ，优化决策变量 s ，这种策略在文献^[62,63]中有讨论；一种是固定离散的里程 s ，优化决策变量时间戳 t ，这种粗略在文献^[64,80]中有

过讨论。关于这两种优化策略，也可以参见图 1-16。本文为了方便加速度和加加速度的计算，选择了第一种优化策略，大概的策略是，本文将速度优化问题转换成一个标准的二次规划问题，详细推导过程可以参见以下两小节。

4.2.1 目标函数

我们将时间离散成固定步长的序列，

$$t_i = t_0 + i\epsilon, i = 0, 1, \dots, n \quad (4-31)$$

在这里， ϵ 比 4.1.2 节中的 h 更精细，因为需要保证速度曲线的足够光滑，不妨设步长为 0.1s，优化的决策变量为 $s = [s_1, s_2, \dots, s_n]^T$ ，最优速度被定义为最小化以下几项的有限和的形式，如下式

$$j = \sum_{i=1}^n (w_{vel}(v_i - v_{ri})^2 + w_{acc}a_i^2 + w_{jerk}jerk_i^2)\epsilon \quad (4-32)$$

s 关于时间的导数可以使用坐标的有限差分来近似计算，

$$\begin{aligned} s'_i &= v_i \approx \frac{s_{i+1} - s_i}{\epsilon} \\ s''_i &= a_i \approx \frac{s_{i+2} - 2s_{i+1} + s_i}{\epsilon^2} \\ s'''_i &= jerk_i \approx \frac{s_{i+3} - 3s_{i+2} + 3s_{i+1} - s_i}{\epsilon^3} \end{aligned} \quad (4-33)$$

显然，目标函数是一个二次函数，第一项是与粗糙速度曲线偏移的惩罚项，即我们希望优化的结果依然能够接近粗糙曲线，因为粗糙速度曲线位于最优解的领域，所以对速度的优化不能偏离太远。第二项和第三项分别是加速度惩罚项和加加速度惩罚项，因为这两种惩罚项的存在，所以速度曲线的加速度和加加速度值不会太大，这样生成的速度曲线就比较光滑。路径优化的起点状态为 $[s_0, t_0, v_0, a_0, jerk_0]^T$ ，可以使用坐标的差分进行近似计算，

$$v_0 \approx \frac{s_1 - s_0}{\epsilon}, \quad a_0 \approx \frac{s_2 - 2s_1 + s_0}{\epsilon^2}, \quad jerk_0 \approx \frac{s_3 - 3s_2 + 3s_1 - s_0}{\epsilon^3} \quad (4-34)$$

这样由初值条件可以确定 s_0, s_1, s_2, s_3 的值，那么剩余需要优化的 s 的数量为 $n-2$ ，假如步长为 0.1s，规划时域为 8s，那么需要优化的向量 s 的长度为 78。为了保守起见，我们希望在优化的末端能够使加速度、加加速度为零，速度保持一个恒定速度，所以有下式

$$\begin{aligned} v_n &\approx \frac{s_{n+1} - s_n}{\varepsilon} = v_{const}, \quad a_n \approx \frac{s_{n+2} - 2s_{n+1} + s_n}{\varepsilon^2} = 0 \\ jerk_n &\approx \frac{s_{n+3} - 3s_{n+2} + 3s_{n+1} - s_n}{\varepsilon^3} = 0 \end{aligned} \quad (4-35)$$

由上式可以得到

$$s_{n+1} = s_n + v_n \varepsilon, \quad s_{n+2} = s_n + 2v_n \varepsilon, \quad s_{n+3} = s_n + 3v_n \varepsilon \quad (4-36)$$

因为

$$v_n = v_{n-1} + a_{n-1} \varepsilon \quad (4-37)$$

因 ε 和 a_{n-1} 都是一个极小量，所以可以令

$$v_n \approx v_{n-1} \approx \frac{s_n - s_{n-1}}{\varepsilon} \quad (4-38)$$

此时，目标函数 (4-32) 可以被转换成

$$j = \mathcal{E} w_{vel} \sum_{i=1}^n j_{refi} + \mathcal{E} w_{acc} \sum_{i=1}^n j_{acci} + \mathcal{E} w_{jerk} \sum_{i=1}^n j_{jerki} \quad (4-39)$$

对于上式的第一项，可以被写为

$$\sum_{i=1}^n j_{refi} = \frac{1}{\varepsilon^2} \mathbf{s}_{n+1}^T \mathbf{H}_{vel} \mathbf{s}_{n+1} + \sum_{i=1}^n v_{ri}^2 - \frac{2}{\varepsilon} \mathbf{q}_{vel}^T \mathbf{s}_{n+1} \quad (4-40)$$

其中，

$$\left\{ \begin{array}{l} \mathbf{s}_{n+1} = [s_1, s_2, s_3, \dots, s_{n+1}]^T \\ \mathbf{H}_{vel} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \\ \mathbf{q}_{vel}^T = [-v_{r1}, v_{r1} - v_{r2}, v_{r2} - v_{r3}, \dots, v_{r(n-1)} - v_{rn}, v_{rn}] \end{array} \right. \quad (4-41)$$

加速度项可以被表达为

$$\sum_{i=1}^n j_{acci} = \frac{1}{\varepsilon^4} \mathbf{s}_{n+2}^T \mathbf{H}_{acc} \mathbf{s}_{n+2} \quad (4-42)$$

其中，

$$\left\{ \begin{array}{l} \mathbf{s}_{n+2} = [s_1, s_2, \dots, s_n, s_{n+1}, s_{n+2}]^T \\ \mathbf{H}_{acc} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & \cdots & 0 \\ -2 & 5 & -4 & \ddots & \ddots & \ddots & \vdots \\ 1 & -4 & 6 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 6 & -4 & 1 \\ \vdots & \ddots & \ddots & \ddots & -4 & 5 & -2 \\ 0 & \cdots & \cdots & 0 & 1 & -2 & 1 \end{bmatrix} \end{array} \right. \quad (4-43)$$

加加速度项可以被表达为

$$\sum_{i=1}^n j_{jerki} = \frac{1}{\varepsilon^6} \mathbf{s}^T \mathbf{H}_{jerk} \mathbf{s} \quad (4-44)$$

其中，

$$\left\{ \begin{array}{l} \mathbf{s} = [s_1, s_2, \dots, s_n, s_{n+1}, s_{n+2}, s_{n+3}]^T \\ \mathbf{H}_{jerk} = \begin{bmatrix} 1 & -3 & 3 & -1 & 0 & \cdots & \cdots & \cdots & 0 \\ -3 & 10 & -12 & 6 & \ddots & \ddots & \ddots & \ddots & \vdots \\ 3 & -12 & 19 & -15 & \ddots & \ddots & \ddots & \ddots & \vdots \\ -1 & 6 & -15 & 20 & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 20 & -15 & 6 & -1 \\ \vdots & \ddots & \ddots & \ddots & \ddots & -15 & 19 & -12 & 3 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 6 & -12 & 10 & -3 \\ 0 & \cdots & \cdots & \cdots & 0 & -1 & 3 & -3 & 1 \end{bmatrix} \end{array} \right. \quad (4-45)$$

为了使各个矩阵维度与式 (4-45) 保持一致，整理得到

$$\left\{ \begin{array}{l} \sum_{i=1}^n j_{refi} = \frac{1}{\varepsilon^2} [\mathbf{s}^T, s_{n+2}, s_{n+3}] \begin{bmatrix} \mathbf{H}_{vel}, \mathbf{0}, \mathbf{0} \\ \mathbf{0}, 0, 0 \\ \mathbf{0}, 0, 0 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ s_{n+2} \\ s_{n+3} \end{bmatrix} + \sum_{i=1}^n v_{ri}^2 - \frac{2}{\varepsilon} [\mathbf{q}_{vel}^T, 0, 0] \begin{bmatrix} \mathbf{s} \\ s_{n+2} \\ s_{n+3} \end{bmatrix} \\ \sum_{i=1}^n j_{acci} = \frac{1}{\varepsilon^4} [\mathbf{s}^T, s_{n+3}] \begin{bmatrix} \mathbf{H}_{acc} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ s_{n+3} \end{bmatrix} \end{array} \right.$$

目标函数整理成二次规划问题的标准形式，

$$\min j(\mathbf{s}) = \frac{1}{2} \mathbf{s}^T \mathbf{H} \mathbf{s} + \mathbf{q}^T \mathbf{s} + p \quad (4-46)$$

其中，

$$\mathbf{H} = 2 \frac{w_{vel}}{\varepsilon} \begin{bmatrix} \mathbf{H}_{vel} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 0 & 0 \\ \mathbf{0} & 0 & 0 \end{bmatrix} + 2 \frac{w_{acc}}{\varepsilon^3} \begin{bmatrix} \mathbf{H}_{acc} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + 2 \frac{w_{jerk}}{\varepsilon^5} \mathbf{H}_{jerk}$$

$$\mathbf{q}^T = -2w_{vel} \begin{bmatrix} \mathbf{q}_{vel}^T & 0 & 0 \end{bmatrix}$$

$$p = w_{vel} \varepsilon \sum_{i=1}^n v_{ri}^2$$

4.2.2 约束函数

对于速度优化，常常也包括内部约束和外部约束，内部约束是由车辆自身的动力学性能和轮胎的摩擦性能带来的，一般会包括最大速度约束、加速度约束、加加速度约束。外部约束由车辆的行驶环境带来的约束，包括交通规则约束，避障约束。内部约束中的加速度约束可以被表达为

$$|s_i''| \leq a_{lim} \quad (4-47)$$

写成矩阵的形式为，

$$\begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n+3} \end{bmatrix} \leq \mathbf{h}_{acc}$$

$$- \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n+3} \end{bmatrix} \leq \mathbf{h}_{acc} \quad (4-48)$$

其中，

$$\mathbf{h}_{acc} = [\varepsilon^2 a_{lim}, \dots, \varepsilon^2 a_{lim}]^T \quad (4-49)$$

内部约束的加加速度约束为

$$|s_i'''| \leq jerk_{lim} \quad (4-50)$$

写成矩阵的形式

$$\begin{aligned}
 & \left[\begin{array}{cccccc} -1 & 3 & -3 & 1 & & \\ & -1 & 3 & -3 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots \\ & & & -1 & 3 & -3 & 1 \end{array} \right] \mathbf{s} \leq \mathbf{h}_{jerk} \\
 & - \left[\begin{array}{cccccc} -1 & 3 & -3 & 1 & & \\ & -1 & 3 & -3 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots \\ & & & -1 & 3 & -3 & 1 \end{array} \right] \mathbf{s} \leq \mathbf{h}_{jerk} \\
 & \mathbf{h}_{jerk} = [\varepsilon^3 jerk_{\lim} \quad \dots \quad \varepsilon^3 jerk_{\lim}]^T
 \end{aligned} \tag{4-51}$$

一般而言，速度约束包括车道限速约束和车辆最大速度约束，然而通常意义下车道限速的速度会小于车辆的最大可行驶速度，所以为了减少约束的数量，只在约束函数中添加外部约束的车道速度约束

$$|s'| \leq v_{\lim} \tag{4-52}$$

将上式写成矩阵的形式

$$\begin{aligned}
 & \left[\begin{array}{cccccc} -1 & 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & -1 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 & 0 & 0 \end{array} \right] \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n+3} \end{bmatrix} \leq \begin{bmatrix} \varepsilon v_{\lim} \\ \varepsilon v_{\lim} \\ \vdots \\ \varepsilon v_{\lim} \end{bmatrix} \\
 & - \left[\begin{array}{cccccc} -1 & 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & -1 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 1 & 0 & 0 \end{array} \right] \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{n+3} \end{bmatrix} \leq \begin{bmatrix} \varepsilon v_{\lim} \\ \varepsilon v_{\lim} \\ \vdots \\ \varepsilon v_{\lim} \end{bmatrix}
 \end{aligned} \tag{4-53}$$

然后决策变量 s 会有取值范围约束

$$s_{init} \leq s_i \leq s_{\max} \tag{4-54}$$

写成矩阵形式

$$\begin{aligned}
 & [\mathbf{I} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0}] \mathbf{s} \leq \mathbf{h}_{\max} \\
 & -[\mathbf{I} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0}] \mathbf{s} \leq \mathbf{h}_{init}
 \end{aligned} \tag{4-55}$$

其中，

$$\begin{aligned}
 \mathbf{h}_{\max} &= [s_{\max} \quad \dots \quad s_{\max}]^T \\
 \mathbf{h}_{init} &= [s_{init} \quad s_{init} \quad s_{init}]^T
 \end{aligned} \tag{4-56}$$

对于障碍物避免约束，有

$$|s_{i,j} - s_i| > r_{veh} + r_{dyn}, \quad |s_{i,j} - s_{2i}| > r_{veh} + r_{dyn}, \quad |s_{i,j} - s_{3i}| > r_{veh} + r_{dyn} \tag{4-57}$$

其中 $s_{i,j}$ 表示第 j 个障碍物在 i 时刻的位置， r_{dyn} 表示动态障碍物的膨胀

半径， r 表示车体包络圆的半径。因为 s_{2i}, s_{3i} 的存在，每一个包络圆都有避障约束，所以障碍物约束形成了非线性约束，我们对 s_{2i}, s_{3i} 的表示为，

$$\begin{aligned} s_{2i} &= s_i + d \cos(\theta(s_i) - \theta_r(s_i)) \leq s_i + d \\ s_{3i} &= s_{2i} + d \cos(\theta(s_i) - \theta_r(s_{2i})) \leq s_i + 2d \end{aligned} \quad (4-58)$$

上式对 s_{2i}, s_{3i} 进行了放大，这种缩放是有物理意义的，因为这种处理增加了车辆到障碍物的安全距离。观察障碍物避免约束，发现可以使用两个方程来代替冗余的约束，

$$s_{i,j} - s_{3i} > r + r_{dyn}, \quad s_i - s_{i,j} > r + r_{dyn} \quad (4-59)$$

将上式写成矩阵的形式，

$$\begin{aligned} &\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_{n+3} \end{bmatrix} < \begin{bmatrix} s_{1,j} - r - r_{dyn} - 2d \\ \vdots \\ s_{n+3,j} - r - r_{dyn} - 2d \end{bmatrix} \\ &- \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_{n+3} \end{bmatrix} < \begin{bmatrix} -s_{1,j} - r - r_{dyn} \\ \vdots \\ -s_{n+3,j} - r - r_{dyn} \end{bmatrix} \end{aligned} \quad (4-60)$$

上式中的右边可以被整理成

$$\mathbf{O}_j = \begin{bmatrix} s_{1,j} - r - r_{dyn} - 2d \\ \vdots \\ s_{n+3,j} - r - r_{dyn} - 2d \end{bmatrix}, \mathbf{D}_j = \begin{bmatrix} -s_{1,j} - r - r_{dyn} \\ \vdots \\ -s_{n+3,j} - r - r_{dyn} \end{bmatrix} \quad (4-61)$$

如果障碍物的数量为 n_o 个， j 表示第 j 个障碍物，那么有下面方程，

$$\mathbf{G}_{obs} \mathbf{s} < \mathbf{h}_{obs} \quad (4-62)$$

其中

$$\begin{aligned} \mathbf{G}_{obs} &= [\mathbf{I}, -\mathbf{I}, \dots, \mathbf{I}, -\mathbf{I}]^T \\ \mathbf{h}_{obs} &= [\mathbf{O}_1 \quad \mathbf{D}_1 \quad \cdots \quad \mathbf{O}_j \quad \mathbf{D}_j \quad \cdots \quad \mathbf{O}_{n_o} \quad \mathbf{D}_{n_o}]^T \end{aligned} \quad (4-63)$$

边值条件形成等式约束

$$\begin{aligned} s_0 &= s_0 \\ s_1 &= \varepsilon v_0 + s_0 \\ s_2 &= a_0 \varepsilon^2 + s_0 + 2\varepsilon v_0 \\ s_3 &= jerk_0 \varepsilon^3 + 3\varepsilon v_0 + 3a_0 \varepsilon^2 + s_0 \\ s_{n+1} &= 2s_n - s_{n-1} \\ s_{n+2} &= 3s_n - 2s_{n-1} \\ s_{n+3} &= 4s_n - 3s_{n-1} \end{aligned} \quad (4-64)$$

写成矩阵形式

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & -3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & -4 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_{n-1} \\ s_n \\ s_{n+1} \\ s_{n+2} \\ s_{n+3} \end{bmatrix} = \begin{bmatrix} \varepsilon v_0 + s_0 \\ a_0 \varepsilon^2 + s_0 + 2\varepsilon v_0 \\ jerk_0 \varepsilon^3 + 3\varepsilon v_0 + 3a_0 \varepsilon^2 + s_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4-65)$$

对上面的约束矩阵加以整理，有

$$\begin{aligned} \mathbf{G} &= [\mathbf{G}_{acc}, -\mathbf{G}_{acc}, \mathbf{G}_{jerk}, -\mathbf{G}_{jerk}, \mathbf{G}_{vel}, \mathbf{G}_{max}, \mathbf{G}_{init}, \mathbf{G}_{obs}]^T \\ \mathbf{h} &= [\mathbf{h}_{acc}, \mathbf{h}_{acc}, \mathbf{h}_{jerk}, \mathbf{h}_{jerk}, \mathbf{h}_{vel}, \mathbf{h}_{max}, \mathbf{h}_{init}, \mathbf{h}_{obs}]^T \end{aligned} \quad (4-66)$$

等式约束矩阵 (4-66) 的左边可以被改写为

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & \cdots & 0 & 2 & -3 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 3 & -4 & 0 & 0 & 1 \end{bmatrix} \quad (4-67)$$

在目标函数和约束函数都被转换成矩阵表达的形式后，我们就可以写出二次规划的标准形式

$$\begin{aligned} \min \quad & j(\mathbf{s}) = \frac{1}{2} \mathbf{s}^T \mathbf{H} \mathbf{s} + \mathbf{q}^T \mathbf{s} + p \\ \text{s.t.} \quad & \mathbf{G} \mathbf{s} \leq \mathbf{h} \\ & \mathbf{A} \mathbf{s} = \mathbf{b} \end{aligned} \quad (4-68)$$

如式 (4-69) 所示，速度优化问题就变成了一个标准的二次规划问题，目标函数是二次的，约束函数是线性的，而速度优化的初值，就是速度规划的前端算法 lattice 搜索生成的粗糙速度曲线。此时，所有的约束都是线性的，所以构成的可行域是 \mathbb{R}^{n+3} 中的一个凸空间，构造一个拉格朗日函数

$$L(\mathbf{s}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = j(\mathbf{s}) - \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{s} - \mathbf{b}) - \boldsymbol{\mu}^T (\mathbf{h} - \mathbf{G}\mathbf{s}) \quad (4-69)$$

我们可以利用拉格朗日函数的 KKT 条件，快速求解出决策变量 \mathbf{S} ，关于二次规划的求解，可以见下一节 4.2.3。

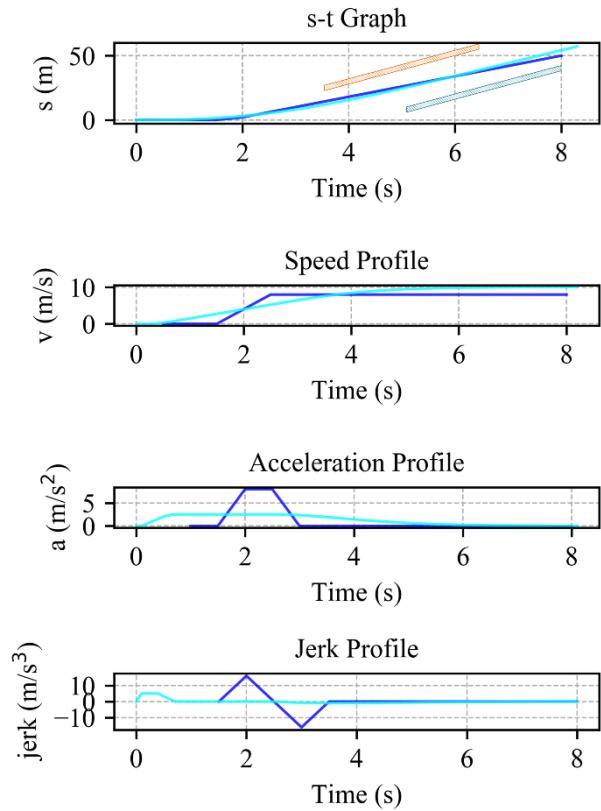


图 4-7 速度规划中的后端优化。蓝色曲线表示前端算法的粗糙速度解，青色曲线表示后端算法生成的最优速度解

图 4-7 是一次速度优化的结果图，st 图中，蓝色曲线和青色曲线几乎重叠，可以看出后端速度优化是以前端生成的粗糙速度解作为参考速度的。在 vt 图中，蓝色曲线较为陡峭，青色曲线比较光滑，且两条曲线都是连续的。在 at 图中，蓝色曲线超过了加速度约束值，因为在进行速度搜索时，不满足加速度约束的曲线并不会被舍弃，而是保留下来，根据我们的经验，如果前端算法在进行搜索时过滤掉太多不满足约束的解，那么最后生成出可行解的效果并不好，因为这样做往往无法搜索出一条可行解。青色曲线能够满足加速度约束，这说明前端算法只需要尽可能快的找出一个比较好的解，至于解的特性可以交给后端算法来优化和改善。在 jerk-t 图中，蓝色曲线的加加速度值也超过了阈值，通过优化后能够满足加加速度约束。

4.2.3 二次规划求解器

二次规划是指，目标函数是关于决策变量的二次函数、约束函数是关于决策变量的线性函数的优化问题。二次规划的标准形式为式（4-69）， \mathbf{s} 是决策变量， \mathbf{H} 为正定矩阵， \mathbf{q} 为目标函数一次项的系数向量， p 为常数， \mathbf{A} 为

等式约束的雅可比矩阵， \mathbf{b} 为等式约束的右端向量。在几何上，二次规划问题的目标函数是超椭球面，其可行域构成了一个凸多面体，二次规划问题的最优解可位于可行域的内部或边界上。在本文中， \mathbf{H} 为正定矩阵，所以速度优化问题成为一个凸优化问题，所以速度优化问题的局部极小点就是全局极小点，并且局部极小点是唯一的。求解凸二次规划问题的主流方法是积极集法和内点法，本文使用 CVXOPT³作为二次规划（Quadratic Programming, QP）的求解器，CVXOPT 开发的 QP 求解器使用的是内点法。内点法会将不等式约束添加上松弛变量使之变为等式约束，这样原始问题成为一个新问题，然后写出新问题的拉格朗日对偶问题，再写出新问题的扰动 KKT 条件，并给出新问题与对偶问题的对偶间隔。通过对扰动 KKT 条件进行线性化并求解，再计算搜索方向与步长，计算新的对偶间隔并更新迭代次数，通过这样迭代求解直到算法收敛。

本文测试了 CVXOPT 求解路径优化问题的解算效率，对于一个总时域为 8.0 秒的速度规划曲线，我们设置步长为 0.1 秒，这样就产生了 84 个决策变量和将近 1000 个约束方程，通过记录 CVXOPT 的解算效率，对于这个中等规模的 QP 问题，算法收敛的平均迭代次数是 6 次，运行时间是 0.0352 秒，所以在时间上来说该优化器有着良好的时间性能，并且本文也分析了优化器生成的速度曲线的性质，速度曲线是连续而光滑的，并且会满足车辆运动的内部约束和外部约束。关于使用 Python 语言进行速度优化的建模和优化的源代码，可以参见个人 GitHub 账户⁴。

³ <http://cvxopt.org/documentation/index.html>

⁴ <https://github.com/yangmingustb/PTPSim>

5 仿真与实验

为了验证本文提出的轨迹规划框架的可行性，分别在 Python 环境下，ROS⁵环境下做了仿真测试，在 Autoware⁶下测试了一些常见轨迹规划算法的性能。本章会对本文的轨迹规划算法和当前主流的算法做一些详细的对比与分析。

5.1 案例研究

为了评估本文提出的规划框架的性能，建立了一条带有多个挑战性场景的虚拟道路，道路的形状和特性如图 5-1、图 5-2 和图 5-3 所示。仿真环境包含了静态和动态障碍物，车辆在行驶的过程遇到的场景包括避免静态和动态障碍物，自主换道，停车，超车等场景。在本文的仿真环境中，车辆使用矩形表示，时间戳信息用矩形颜色的深浅表示，越浅的颜色表示时间过去得越久，越深的颜色表示时间更靠近当前时间，轨迹规划器使用了 Python3.6 语言开发，处理器信息是 2.6GHz Intel Core i7-6500U，内存是 8GB，操作系统是 Ubuntu 16.04.6 LTS。路径优化器使用了开源的非线性优化工具 CasADi，内点法求解器使用的是开源优化库 IPOPT，然后速度优化器和 MPC 优化器分别使用了开源优化库 CVXOPT 和 CVXPY。

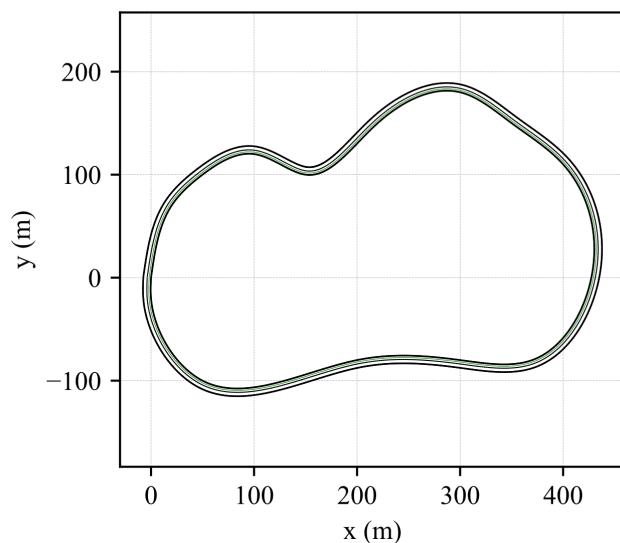


图 5-1 双车道仿真环境，全长 1163.8 米，黑色实线表示车道线，绿色虚线表示车道的参考线

⁵ <https://www.ros.org/>

⁶ <https://www.autoware.ai/>

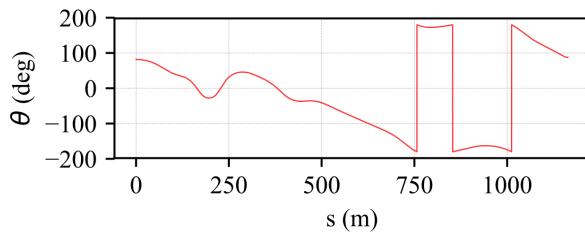


图 5-2 仿真道路的车道参考线的朝向角

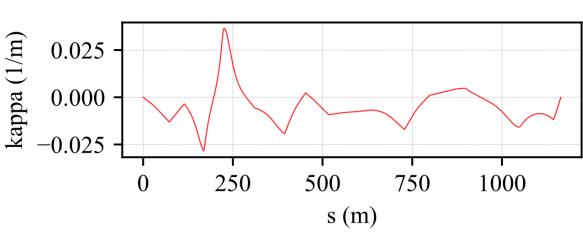


图 5-3 仿真道路的车道参考线的曲率

在我们的仿真器中，因为环境信息和车辆状态信息可以直接从仿真器获得，所以我们移除了感知模块。对于决策层，基于预定义的行为和场景，使用有限状态机^[1,15]来做决策。在自动驾驶中，现在主流的决策模型包括有限状态机（finite state machine, FSM）模型和部分可观马尔可夫决策过程（partially observable Markov decision process, POMDP）模型。对于有限状态机模型，根据人类知识，多个行驶状态被预定义，并且一个车辆的行驶状态又被分为多个子状态来适应不同场景的切换。为了完全考虑感知的不确定性，POMDP 模型提供了一个标准的决策框架，能够摆脱基于规则的决策模型的限制。在 DARPA 挑战赛期间，多数车辆使用的是有限状态机模型来做决策，并且取得了比较好的效果。并且由于清晰的逻辑和较低的计算复杂度，基于规则的决策模型完全能够处理简单的场景，而对于 POMDP，计算复杂度太高使得其不适用于实时的自动驾驶。同时，有限状态机的参数很容易调整，因为能够预测到每一步调参的后果。

对于预测层来说，本文联合车辆运动学模型和曲线道路模型来预测轨迹，在简单的场景和短期内，预测的轨迹还是比较有效的。对于控制层来说，由于非完整性约束和动力学约束的存在，我们使用模型预测控制（model predictive control, MPC）^[81]来跟踪轨迹，与纯跟踪（pure pursuit controller）^[14]控制器和后轴反馈控制器（rear wheel based feedback controller）^[14]相比，MPC 控制器能够保证合适的跟踪精度，并在高速场景下也有良好的控制效果。

5.1.1 案例一：静态障碍物避免

静态障碍物避免是轨迹规划最基本的功能之一，通常期望轨迹规划器能够输出时空光滑的轨迹，绕过障碍物且不跨越道路边界，本文设置了三个静态障碍物，主车辆此时从静止开始启动，试图在右侧车道中前行，图 5-4 中紫红色曲线显示路径搜索器能够搜索出一条较好的路径，能够避免静态障碍物，且看起来很光滑，蓝色的图表示由运动基元构成的路径 lattice。图 5-5 显示了路径优化的结果，图 5-6 显示了粗糙路径和优化路径的曲率，图中显示粗糙路径的曲率存在跳变点，经过优化后，曲率跳变被去除了，路径的空间平滑性得到了较好的改善。

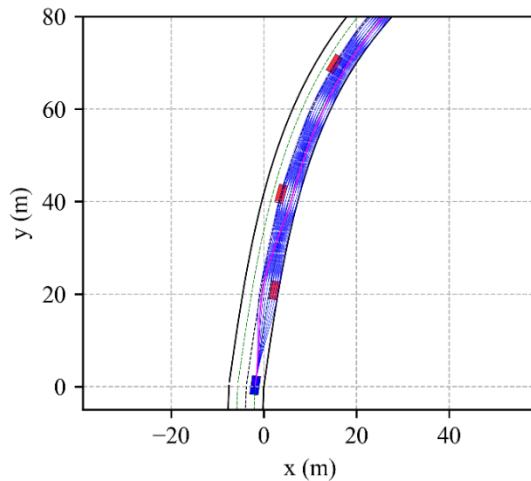


图 5-4 路径 lattice 构造与搜索，蓝色曲线表示路径 lattice 图，紫红色曲线表示粗糙路径，蓝色矩形表示主车辆，红色矩形表示障碍物

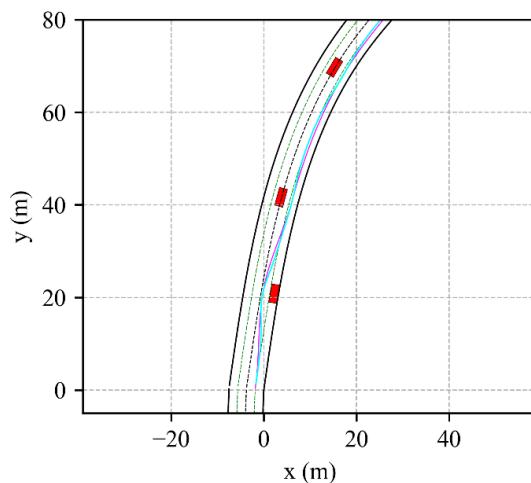


图 5-5 路径优化图。紫红色曲线表示粗糙路径，青色曲线表示优化的路径，绿色虚线表示车道参考线

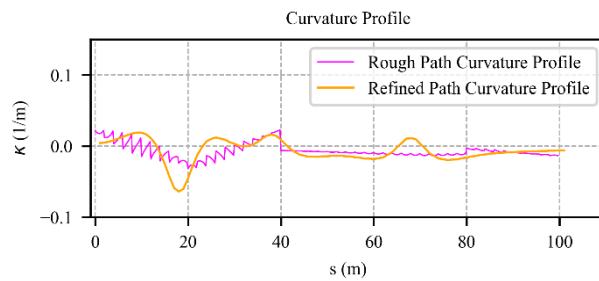


图 5-6 粗糙路径与优化路径曲率

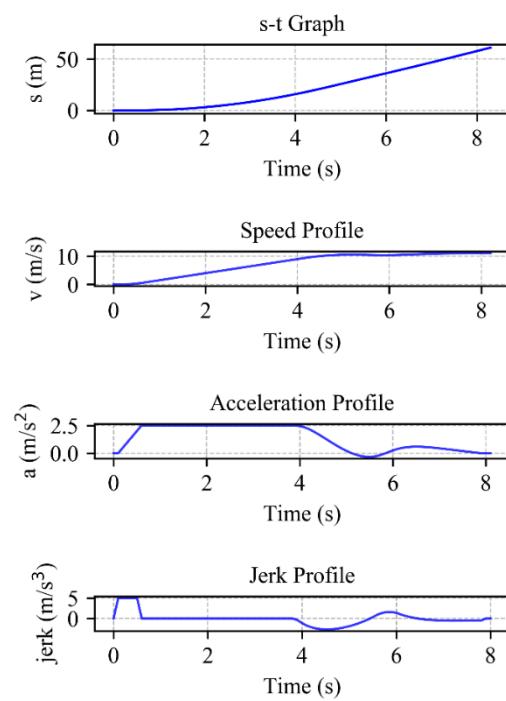


图 5-7 速度、加速度、加加速度曲线

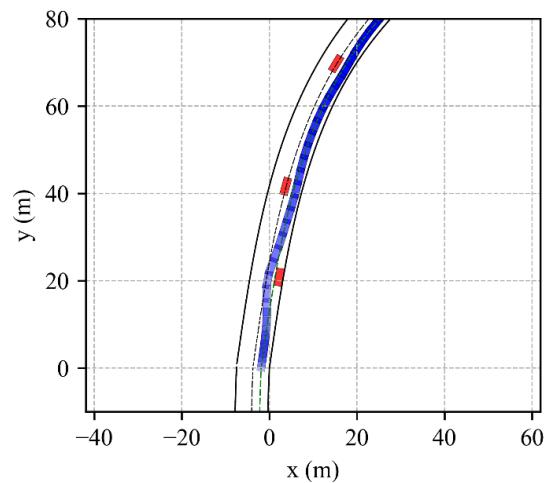


图 5-8 主车辆和静态障碍物的轨迹

图 5-7 记录了静态障碍物避免中的规划轨迹的速度、加速度、加加速度曲线图，各个曲线在时间上都是连续的，速度、加速度曲线更是光滑的。图 5-8 记录了通过 MPC 控制器跟踪规划轨迹的实际轨迹图。静态障碍物避免场景验证了我们的算法可以输出一条时空光滑的并且可行的轨迹。

5.1.2 案例二：自主换道场景

换道是自动驾驶车辆在结构化道路上行驶时的最常见的操作，根据我们的预测模型，障碍物会以当前的状态沿着车道向前行驶。我们在主车辆前方设置了一个缓慢移动的障碍物，障碍物速度为 3m/s，从而验证规划器的换道性能。图 5-9 显示了换道场景中生成的粗糙路径，图 5-10 显示了粗糙路径

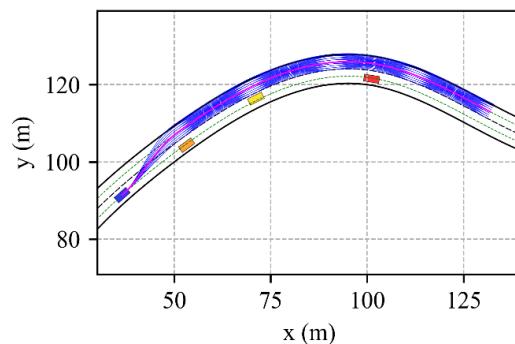


图 5-9 换道场景中粗糙路径搜索

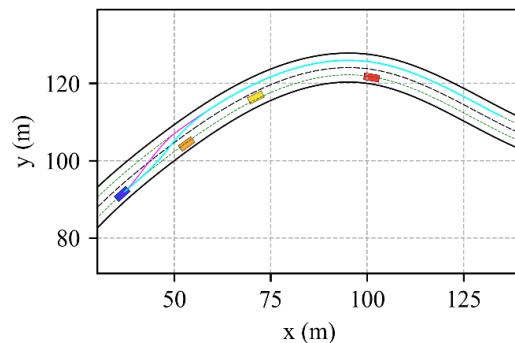


图 5-10 粗糙路径与优化路径

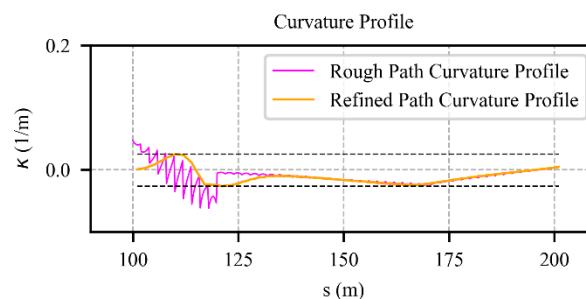


图 5-11 粗糙路径与优化路径的曲率

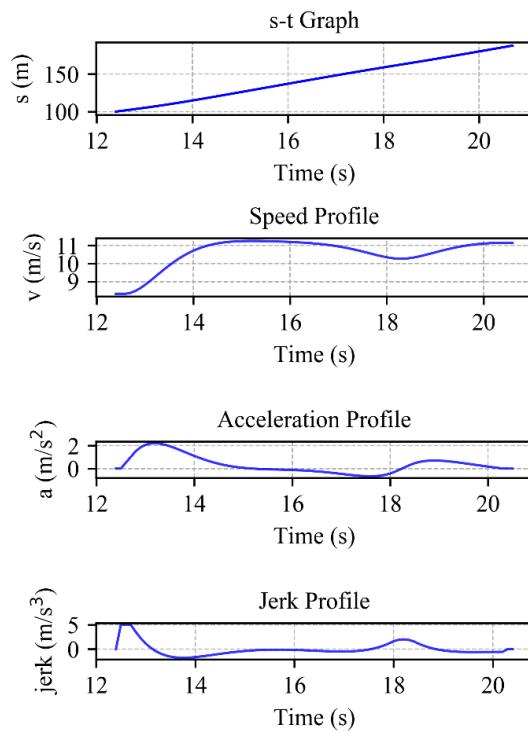


图 5-12 速度优化曲线图

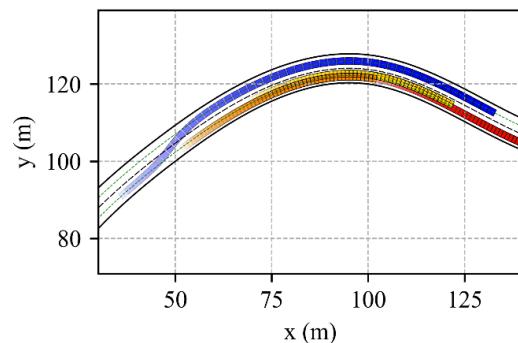


图 5-13 变道场景中的车辆轨迹跟踪图

和优化的路径，图 5-11 显示了粗糙路径和优化路径的曲率，优化路径的最大曲率为 $0.01m^{-1}$ ，图 5-12 显示了主车辆的速度应当在时刻 $t = 12.4s$ 从 $8m/s$ 加速度到 $t=15.2s$ 的 $11.3m/s$ 。我们设定的最大加速度值和最大减速度值分别为 $2.5m/s^2$ 、 $-2.5m/s^2$ ，并且设定最大加加速度值和最大减减速度值分别为 $5m/s^3$ 、 $-5m/s^3$ 。通过速度图，我们可以看出速度曲线是光滑的，并且没有超过设定的速度阈值。在这一节设置了自主换道场景，结果显示，轨迹规划器能够较好地完成换道的轨迹规划任务。

5.1.3 案例三：停车场景

在停车场景，我们在左侧车道设置了一个静态障碍物，如图 5-14 中的黄色矩形所示，并且在这里决策层并不会发出换道的指令，所以当主车辆的前方有静态障碍物时车辆应当停车。图 5-14 显示了向前采样的总里程为 80m，在纵向方向上采样四组位姿，每一组位姿的纵向间隔为 20m，迪杰斯特拉算法可以使用 0.08 秒就能搜索出一条最优路径。图 5-16 显示了粗糙路径和优化路径的曲率，并且粗糙路径的曲率并不够光滑。当 $t=24.2s$, $s=200.0m$ 时，主车辆检测到前方障碍物，如图 5-17 所示，当 $t=24.2s$ 时，主车速度为 15.1m/s，

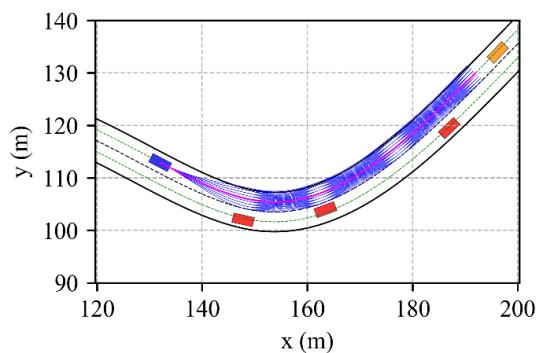


图 5-14 粗糙路径搜索

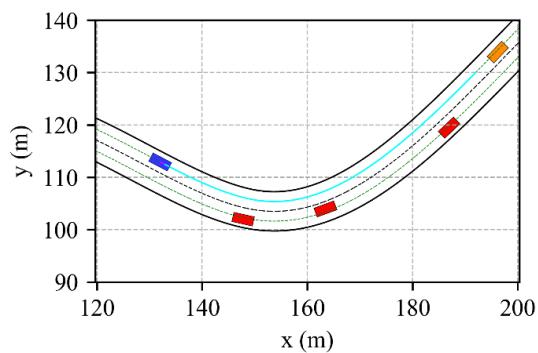


图 5-15 粗糙路径与优化路径

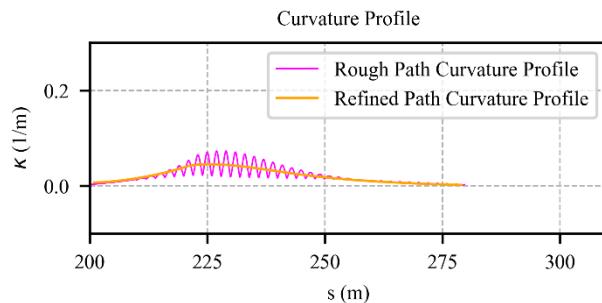


图 5-16 粗糙路径与优化路径的曲率

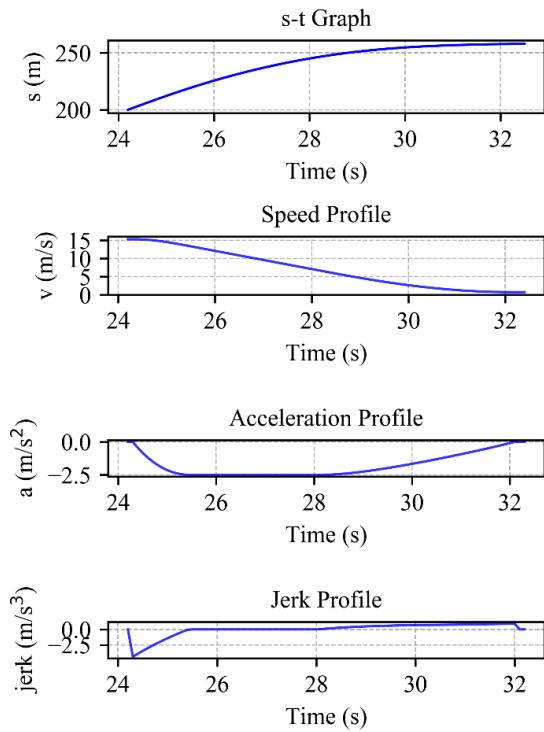


图 5-17 停车场景的速度优化

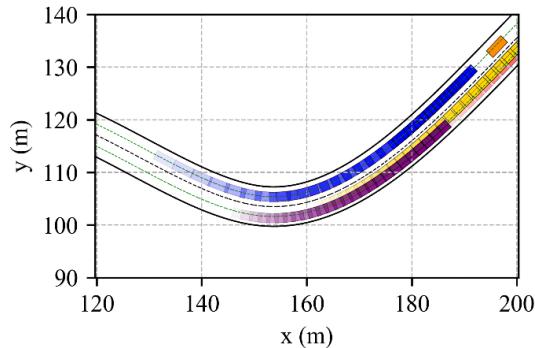


图 5-18 停车场景的轨迹跟踪图

在 $t=32.0$ s 时，主车速度降低到 0.0m/s ，如图所示，车辆在停车规划时也能够输出良好的速度曲线，并能做到合适的停车行为。这一节设置了车辆行驶过程中出现需要停车的场景，显示规划器能够输出光滑的、可行的、满足动力学约束的轨迹。

5.1.4 案例四：超车场景

轨迹规划中最有挑战性的场景之一就是超车，由于主车辆和周围车辆的速度都比较高，所以规划器给出的轨迹需要有合理的速度、加速度以及曲率值。当 $s=400\text{m}$, $t=39.2\text{s}$ 时，决策层发出了一个超车的指令，图 5-19 显示搜

索出的粗糙路径能够超越障碍物，图 5-20 显示了粗糙路径和优化路径，图 5-21 显示了粗糙路径和优化路径的曲率，粗糙路径的曲率曲线包含三个跳变点，这表明粗糙路径的曲率并不是连续的，根据我们的经验，曲率跳变点一般发生在运动基元的连接处。在超车过程中，当前车道中存在有两个缓慢移动的障碍物，相邻车道有一个速度较快的障碍物。如图 5-22 所示，在 $t=39.2s$ 时，主车辆从速度为 $6.0m/s$ 开始加速直到 $t=44.3s$ ，速度加速到 $15.1m/s$ 。在超越

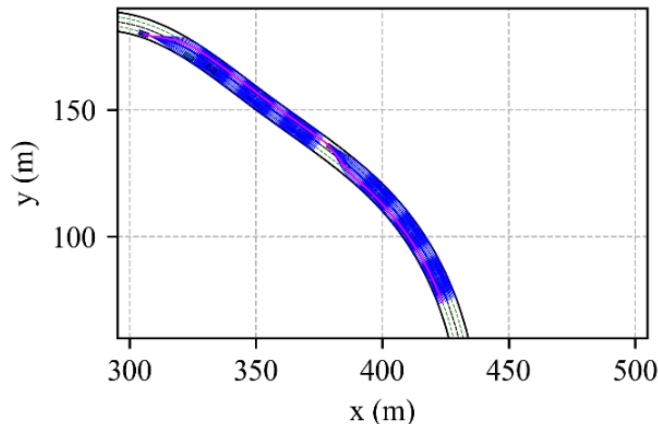


图 5-19 超车场景的路径搜索

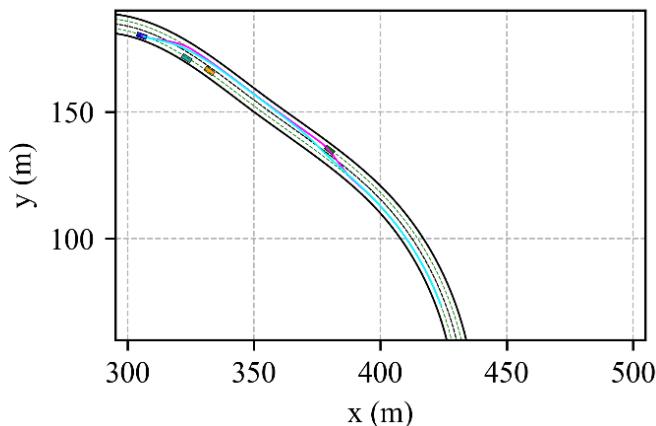


图 5-20 粗糙路径和优化路径

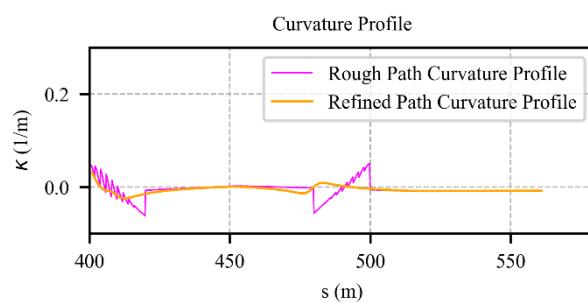
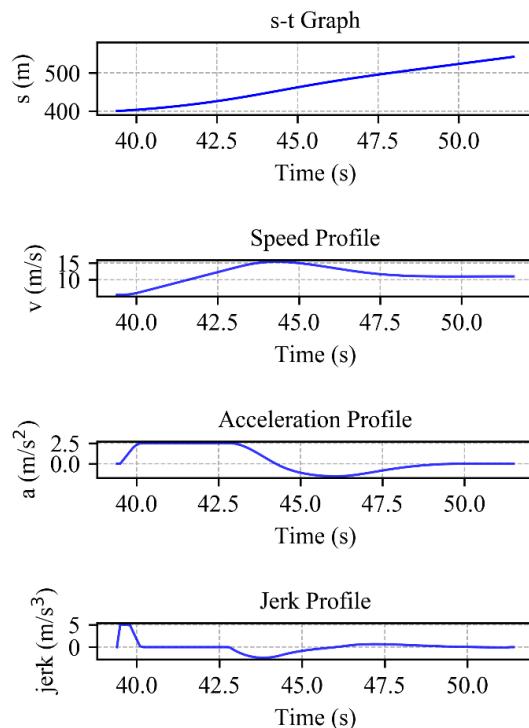
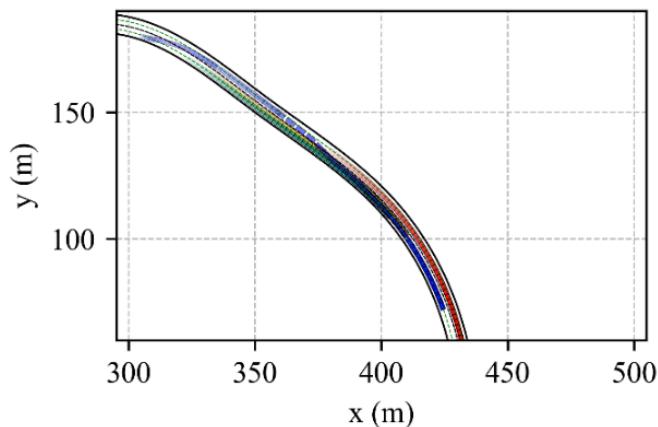


图 5-21 粗糙路径与优化路径的曲率**图 5-22 超车场景的速度优化****图 5-23 超车场景的轨迹跟踪图**

了右侧的车辆后，主车辆进入左侧车道，然后开始缓慢减速。在本文中我们设置的参考速度为 40km/h，当 $t=50s$ 时，主车速度又返回到参考速度。图 5-23 记录了主车辆和周围车辆的轨迹，蓝色轨迹属于主车辆，从图中可以看出，蓝色轨迹的矩形间隔在进入左侧车道后开始变大，这表明车辆在加速，在主车辆进入右侧车道后，矩形间隔开始缩小，这意味着车辆开始缓慢减速。

5.2 自动驾驶平台中的仿真测试

5.2.1 基于 ROS 的仿真测试

机器人操作系统（Robot Operating System, ROS）提供了库和工具来帮助机器人开发人员创建机器人应用程序，ROS 提供了硬件抽象、设备驱动程序、可视化工具、通信机制以及功能丰富的程序包，ROS 是一个开源的系统，广泛运用于科研和工业界，ROS 社区含有丰富的说明文档，十分容易上手，大量学者和工程师能够使用 ROS 快速搭建起自己的算法框架，加速了机器人相关算法的研发。

本文基于 ROS 平台也开发了文章中提出的解耦轨迹规划框架，项目命名为 localPlanner⁷，关于该项目的源代码我们也撰写了丰富的文档，可以在该项目中下载。图 5-24 是 RVIZ 中显示的近似弧长参数化后的道路参考线，我们设计的道路全场将近 2 公里，使用本文 2.1 节中的算法来构建道路参考线。

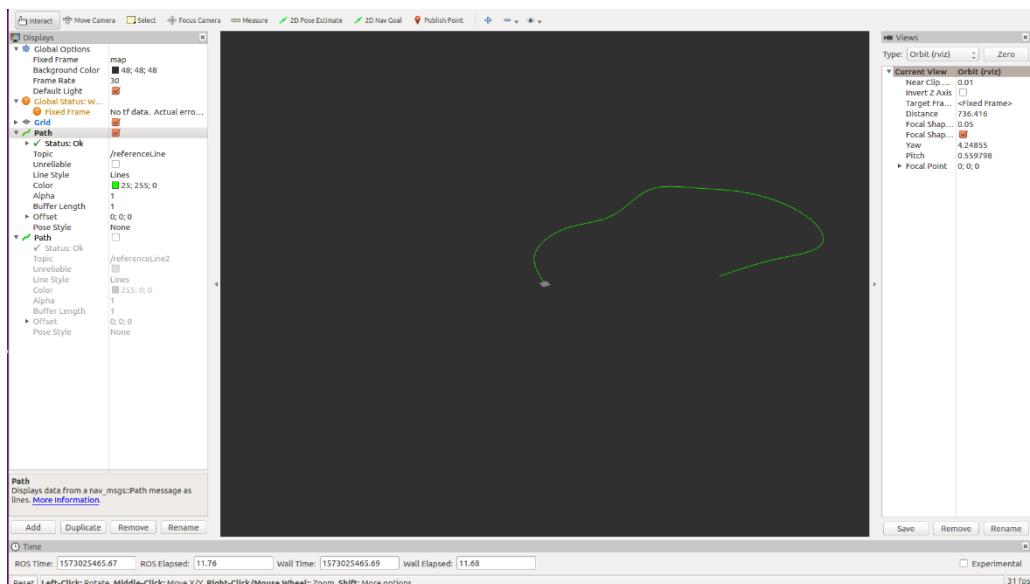


图 5-24 ROS 下构建的弧长参数化后的道路参考线

图 5-25 是 RVIZ 中显示的某一时刻的轨迹规划的结果，紫色曲线为道路的参考线，两侧的白点表示道路边缘，绿色方块表示障碍物，红色网络表示构建的路径 lattice，黄色的长方体表示车辆的轨迹，该图显示，本文的轨迹规划模块能够输出一条规划时域足够长、可避免障碍物、满足运动学动力学约束的可行轨迹，并且通过记录平均运行时间为 0.082 秒，

⁷ <https://github.com/yangmingustb/localPlanner>

可以得出该轨迹规划器具有良好实时性的结论。

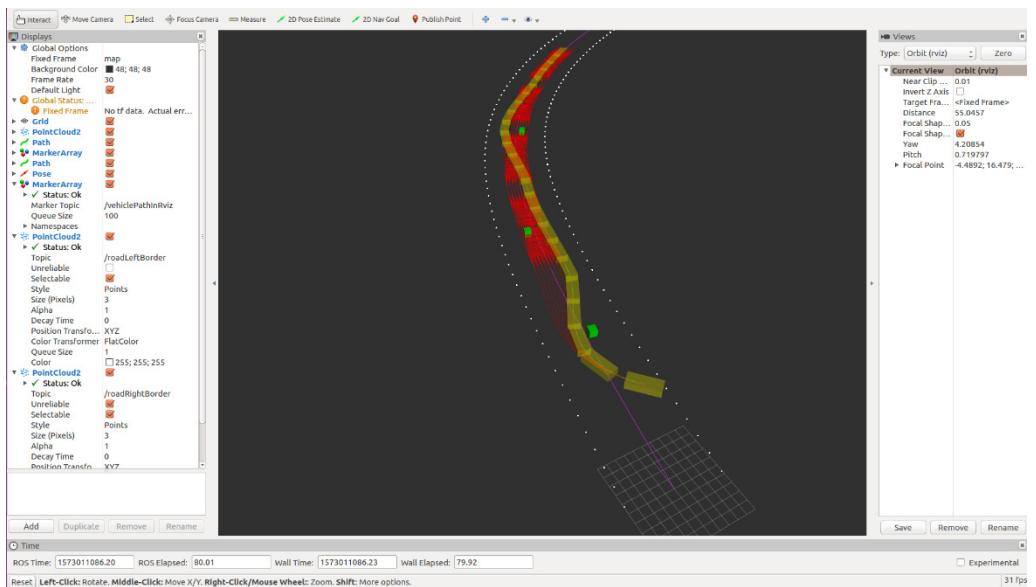


图 5-25 ROS 环境下进行轨迹规划算法测试图

5.2.2 基于 autoware 的仿真测试

Autoware 是完全基于 ROS 开发的开源软件，是专门针对自动驾驶车辆这一套硬件系统而开发的自动驾驶软件平台，定位模块依赖于高精地图和 NDT 算法，从 CAN 消息和 GNSS/IMU 传感器获得里程计信息，检测模块是使用相机、激光雷达与高精地图数据，采用深度学习和传感器融合算法完成对环境的感知，跟踪与预测模块使用卡尔曼滤波算法和高精地图数据完成自我功能的实现，规划模块基于概率机器人技术和基于规则的知识进行运算，控制模块通过控制车辆速度和车轮转角来实现车辆的运动。得益于开发者的贡献，Autoware 每一个模块或多或少都集成了几种当前流行的算法，Autoware 软件

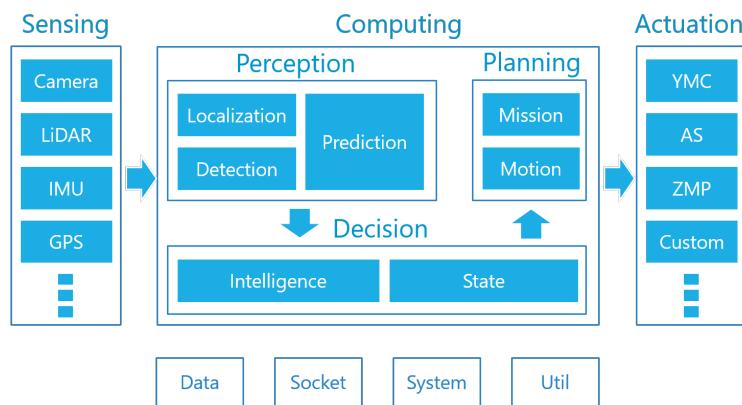


图 5-26 Autoware 软件架构

架构如图 5-26 所示，在线行驶时的可视化界面如图 5-27 所示。Autoware 的局部规划模块集成了三个规划器：分别是 `astar_planner`，实现的是 Hybrid A* 算法，主要用于自由空间环境中的局部规划；`adas_lattice_planner`，这个规划器实现的是 lattice 规划算法，该规划器既可用于结构化道路行驶也可以用于自由空间行驶；`op_planner`，实现的是先采样再使用三次多项式曲线作为运动基元的规划算法，该规划器只能用于结构化道路行驶。

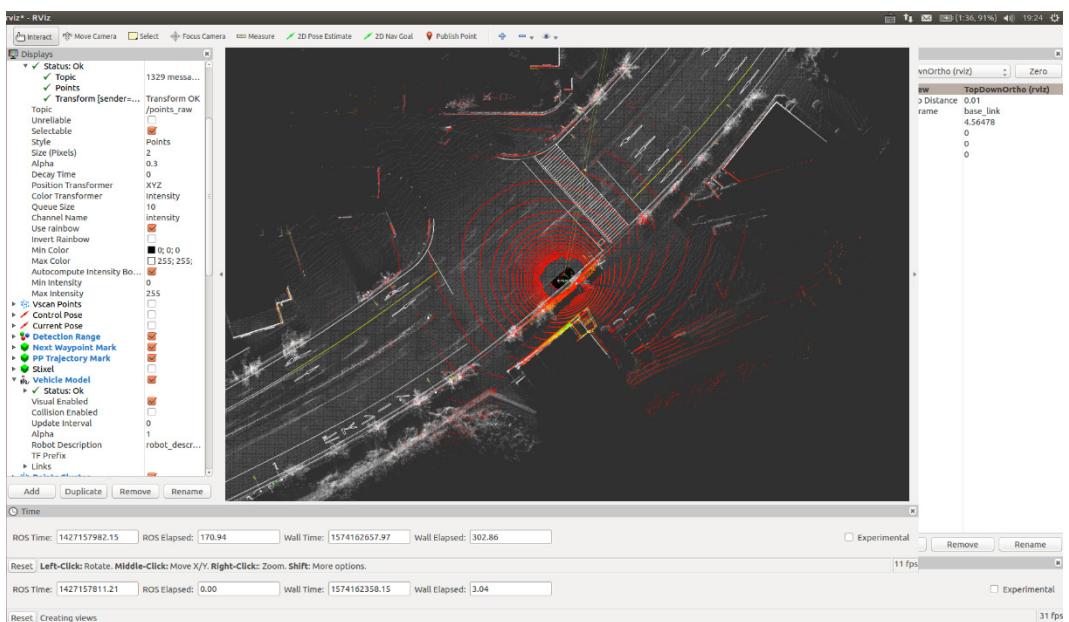


图 5-27 Autoware 系统的仿真车辆在结构化道路上运行

Autoware 作为自动驾驶的仿真平台，上手简单，完全基于 ROS，能够很快搭出算法的原型，并且在虚拟环境中进行验证和测试，本文 5.2.1 节中的项目也将会移植到 Autoware 中进行仿真测试。

5.3 算法性能分析与比较

本文在这里会以 Python 语言版本的算法性能进行分析，我们测试了路径规划前端和速度规划前端算法的运行时间，对纵向采样步长为 20 米、采样个数为 5，横向采样步长为 0.5 米、采样数量为 9 的路径 lattice，平均搜索时间是 0.12 秒。对于时间步长为 1 秒、采样数量为 8 组，里程 s 步长为 4 米、采样数量为 5 的速度 lattice，平均搜索时间是 0.04 秒。对于路径规划和速度规划后端算法的时间性能，本文在 3.2.3 节和 4.2.3 节有过说明。

为了与最新的规划算法进行性能上的比较，我们针对静态障碍物避免场景测试了多种算法的时间性能，在比较中，我们放置了四个静态障碍物，并且设置总的规划视野为 100 米。通过 5.1 节中四个案例的验证，并且比较表 1-1、表 1-2 中的各项算法特征，与 Werling^[25]的方法相比，我们的方法能够充分利用车辆的运动潜力。对于 Li^[46]的方法，时间性能是吸引人的，然而，算法具有较短的规划时域、不够平滑的轨迹以及车辆有限的运动潜力这三个缺点，这在高速情况下会带来潜在的碰撞风险，也会给跟踪模块带来一定的计算负担，并且这种方法的规划距离通常是二十米或者三十米，这取决于车辆当前速度和道路结构，一般而言，车速越快，道路越直，那么规划距离就越大，所以在高速场景下一般会放弃短视的算法。对于 Hybrid A* 算法以及随后的基于梯度下降的路径平滑算法，这个算法可以在拥挤的环境中生成一条光滑的路径，但是它缺少速度规划步，同时这一套算法不适合在结构化道路上的实时运行，因为不容易确定短期的临时目标点，所以启发函数难以设计故会导致搜索难以进行，并且目标点也在频繁地改变，这使得算法的时间复杂度太高。相反，Hybrid A* 算法更适合自由空间的规划。并且，我们的算法充分利用了采样类算法和优化类算法的优点，且试图降低两类算法各自的缺点。而且，我们提出的新颖的、全面的算法对各种约束满足和需求满足之间有一个很好的平衡。

与 Ziegler^[2]的方法相比，我们的方法能够跳出局部最优，而不必精心处理障碍物使其凸化，并且可以通过调整参数来适应不同的行驶场景。还有，轨迹规划的后端算法都是对轨迹的优化，然后优化的初值有不同的设置方式，我们的方法使用轨迹规划的前端算法来提供初值，所以只需迭代几次就能够

表 5-1 主流轨迹规划算法的性能比较

算法	迭代次数	运行时间 (s)	规划时域	速度规划	规划策略
本文	7(路径)+6 (速度)	0.55	长	有	路径-速度解耦，独立进行搜索与优化
Ziegler ^[2]	20	0.89	长	有	轨迹直接优化
Werling ^[25]		0.11	短	有	横向轨迹-纵向轨迹解耦，反应式轨迹生成
Li ^[46]	4	0.09	短	有	路径-速度解耦，路径搜索和速度生成
Hybrid A*		0.27	长	无	路径搜索

Hybrid A*+CG ^[60]	7	0.42	长	无	路径搜索与优化
---------------------------------	---	------	---	---	---------

收敛到最优解。关于各个轨迹规划算法的性能，本文测试的结果可以参见表 5-1。

6 结论

6.1 全文总结

本文在绪论中对于当前主流的轨迹规划算法有一个全面而系统的总结，体现了作者对于规划领域学习两年来的认识和心得，并对运动规划中使用到的基础知识，包括地图、坐标转换、碰撞检测和运动基元都有一个详实的归纳与总结，并且也做了详细的推导或者阐述。对于轨迹规划的前端，本文介绍了采样的规则和图搜索算法，对于轨迹规划的后端，本文给出了详细的建模过程。

在本文中，一个新的解耦轨迹规划框架被提出和实现，来解决非凸的时空规划问题。规划的前端算法可以在高度非凸的状态空间中搜索出一条接近全局最优的解，规划的后端算法可以保证生成的解是连续的、时空光滑的和最优的。并且通过解耦也使得我们算法的实时性也得到提高。

6.2 论文的贡献

根据相关的工作，并且为了实时地解决自动驾驶中结构化道路地轨迹规划问题，本文提出了一个解耦的轨迹规划框架。本文将一个三维的规划问题分解成两个二维的规划问题，首先执行路径搜索和路径后优化，然后基于优化的路径，进行速度搜索和速度优化。如此，离散构型空间的可行解被 lattice 搜索步发现，并作为初值传给优化步。有了 lattice 搜索，非线性优化步能够只迭代几次就收敛到最优的、连续的解。本文的主要贡献总结如下：

- [1] 本文提出了一个新颖的轨迹规划框架，该框架能够组合优化类方法和采样类方法的优点。对于文献^[2,55,56,59]中的优化类方法，他们能够保证足够的光滑性；对于文献^[45,46,49,51]中的采样类方法，他们比较关注实时性和算法的灵活性。本文提出的算法高亮的地方在于：在约束满足（非完整性约束，碰撞避免约束，交通规则约束）和需求满足（最优性，光滑性，实时性，灵活性）之间达到了一种很好地平衡。与文献^[2,55,56,59]中的方法相比，本文中的组合类方法可以跳出驻点，降低算法的迭代次数，提高算法的灵活性。与文献^[45,46,49,51]中的方法相比，组合类方法改善了光滑性和灵活性。因此，轨迹规划器能够快速生成一条时空光滑的、满足运动学约束的可行解。
- [2] 本文引进了一种局部的、连续的方法来优化 lattice 搜索生成的粗糙路径。文献^[61]将路径优化问题转换成一个二次规划问题，在文献^[44]中，单纯形

法被用来优化路径，在文献^[65]中，SQP 被用来优化粗糙的轨迹，这些方法在避免碰撞的同时，也能够平滑路径或轨迹。然而，这些方法在轨迹优化的过程中没有加入非完整性约束，这意味着优化后的轨迹不一定满足非完整性约束。因此，本文将路径优化问题考虑成一个非线性优化问题（nonlinear programming problem, NLP Problem），并且在 Frenet 坐标的优化中加入了非完整性约束。与文献^[2,55,58]中的工作不同，本文将粗糙解作为初值传给 NLP 问题，作为优化过程的热启动，这一过程能够保证快速地收敛和防止陷入局部最优。另外的一个新颖之处在于本文将与参考线的横向偏移作为优化的决策变量，并且使用路点的有限差分去计算轨迹的曲率。

- [3] 本文将速度优化问题建模成一个标准的二次规划问题，来优化路线上与时间戳具有一对一映射关系的纵向里程。与文献^[61]中的样条二次规划速度优化器，文献^[44]中的非导数单纯形速度优化器不同，本文将里程 $s = [s_1, s_2, \dots, s_n]^T$ 作为优化的决策变量，约束函数能够保证生成的速度曲线满足运动学、动力学约束，目标函数能够保证生成的速度曲线足够的时间光滑性。文献^[64]直接将速度优化问题建模成一个 NLP 问题，但是并没有提供一个良好的初值给优化过程，而本文的工作会提供一个较好的初值给二次规划，所以能够大大降低算法的迭代次数。并且由于速度 lattice 搜索的存在，本文的算法能够保证全局最优。与文献^[33]中的梯形速度曲线，文献^[45]中的光滑梯形速度曲线相比，本文并不指定一个明确的速度值，加速度值和加加速度值，而是根据约束函数和目标函数来优化这些参数，这一过程保证了速度优化器能够适应不同的驾驶场景，具有较好的灵活性。

6.3 研究展望

结合自己对运动规划领域的认知和实际的仿真测试，对于未来的研究有以下几点展望：

- 1) 本文的轨迹规划是基于参考线的规划，在实际使用中，发现 Frenet 坐标与笛卡尔坐标的相互转换，特别是频繁地发生坐标转换，会大大降低计算效率，为了进一步提高轨迹规划算法的实时性，规划空间不再依赖于 Frenet 坐标系，转而直接使用笛卡尔坐标系，这样去掉了坐标间的频繁转换。然后为了继续利用道路信息，我们依然使用道路参考线，但不再使用 Frenet 坐标系，根据我的经验，这样能够加速轨

迹前端算法和后端算法的收敛。

- 2) 着手设计采样算法, 而不是在 Frenet 坐标系中均匀采样, 因为 Frenet 均匀采样的有效性很低。转而只在遇到障碍物时才发生密集采样, 否则进行稀疏采样, 这样不仅提高采样效率, 也大大降低前端算法的搜索时间。

参考文献

- [1] MONTEMERLO M, BECKER J, BHAT S, 等. Junior: The Stanford entry in the Urban Challenge[J]. Journal of Field Robotics, 2008, 25(9): 569–597.
- [2] ZIEGLER J, BENDER P, DANG T, 等. Trajectory planning for Bertha — A local, continuous method[C]//2014 IEEE Intelligent Vehicles Symposium Proceedings. MI, USA: IEEE, 2014: 450–457.
- [3] SINGH S. Critical reasons for crashes investigated in the national motor vehicle crash causation survey[R]. 2015.
- [4] DINGUS T A, GUO F, LEE S, 等. Driver crash risk factors and prevalence evaluation using naturalistic driving data[J]. Proceedings of the National Academy of Sciences, 2016, 113(10): 2636–2641.
- [5] PETROV P, NASHASHIBI F. Modeling and Nonlinear Adaptive Control for Autonomous Vehicle Overtaking[J]. IEEE Transactions on Intelligent Transportation Systems, 2014, 15(4): 1643–1656.
- [6] GEIGER A, LENZ P, URTASUN R. Are we ready for autonomous driving? The KITTI vision benchmark suite[C]//2012 IEEE Conference on Computer Vision and Pattern Recognition. 2012: 3354–3361.
- [7] The DARPA Urban Challenge: autonomous vehicles in city traffic[M]. BUEHLER M, IAGNEMMA K, SINGH S. Berlin: Springer, 2009.
- [8] BURNS L D. Sustainable mobility: A vision of our transport future[EB/OL]. Nature, 2013-05-08. (2013-05-08)[2018-09-13]. doi:10.1038/497181a.
- [9] SCHWARTING W, ALONSO-MORA J, RUS D. Planning and Decision-Making for Autonomous Vehicles[J]. Annual Review of Control, Robotics, and Autonomous Systems, 2018, 1(1): 187–210.
- [10] 5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018[EB/OL]. [2018-10-06]. <https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/>.
- [11] 国务院关于印发《中国制造 2025》的通知_政府信息公开专栏[EB/OL]. [2018-10-06]. http://www.gov.cn/zhengce/content/2015-05/19/content_9784.htm.
- [12] 工业和信息化部关于印发《促进新一代人工智能产业发展三年行动计划（2018-2020 年）》的通知[EB/OL]. [2018-10-06]. <http://www.miit.gov.cn/n1146295/n1652858/n1652930/n3757016/c5960820/content.html>.
- [13] ESKANDARIAN A. Handbook of intelligent vehicles[M]. Springer, 2012, 2.
- [14] PADEN B, CAP M, YONG S Z, 等. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2016, 1(1): 33–55.
- [15] URMSON C, ANHALT J, BAGNELL D, 等. Autonomous driving in urban environments: Boss and the Urban Challenge[J]. Journal of Field Robotics, 2008, 25(8): 425–466.
- [16] KATRAKAZAS C, QUDDUS M, CHEN W-H, 等. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions[J]. Transportation Research Part C: Emerging Technologies, 2015, 60: 416–442.

- [17] MILANÉS V, LLORCA D F, VILLAGRÁ J, 等. Intelligent automatic overtaking system using vision for vehicle detection[J]. Expert Systems with Applications, 2012, 39(3): 3362–3373.
- [18] VANHOLME B, GRUYER D, LUSETTI B, 等. Highly Automated Driving on Highways Based on Legal Safety[J]. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(1): 333–347.
- [19] OKUMURA B, JAMES M R, KANZAWA Y, 等. Challenges in Perception and Decision Making for Intelligent Automotive Vehicles: A Case Study[J]. IEEE Transactions on Intelligent Vehicles, 2016, 1(1): 20–32.
- [20] FLETCHER L, TELLER S, OLSON E, 等. The MIT–Cornell collision and why it happened[J]. Journal of Field Robotics, 2008, 25(10): 775–807.
- [21] AEBERHARD M, RAUCH S, BAHRAM M, 等. Experience, Results and Lessons Learned from Automated Driving on Germany’s Highways[J]. IEEE Intelligent Transportation Systems Magazine, 2015, 7(1): 42–57.
- [22] VALLE S M L. Motion Planning[J]. IEEE Robotics Automation Magazine, 2011, 18(2): 108–118.
- [23] GONZALEZ D, PEREZ J, MILANES V, 等. A Review of Motion Planning Techniques for Automated Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 17(4): 1135–1145.
- [24] CHOSET H M, HUTCHINSON S, LYNCH K M, 等. Principles of robot motion: theory, algorithms, and implementation[M]. MIT press, 2005.
- [25] WERLING M, ZIEGLER J, KAMMEL S, 等. Optimal trajectory generation for dynamic street scenarios in a Frenet Frame[C]//2010 IEEE International Conference on Robotics and Automation. Anchorage, AK: IEEE, 2010: 987–993.
- [26] BOHREN J, FOOTE T, KELLER J, 等. Little Ben: The Ben Franklin Racing Team’s entry in the 2007 DARPA Urban Challenge[J]. Journal of Field Robotics, 2008, 25(9): 598–614.
- [27] HUYN N, DECHTER R, PEARL J. Probabilistic analysis of the complexity of A*[J]. Artificial Intelligence, 1980, 15(3): 241–254.
- [28] STENTZ A. Optimal and efficient path planning for partially-known environments[C]//Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. IEEE, 1994: 3310–3317.
- [29] FERGUSON D, STENTZ A. Using interpolation to improve path planning: The Field D* algorithm[J]. Journal of Field Robotics, 2006, 23(2): 79–101.
- [30] DANIEL K, NASH A, KOENIG S, 等. Theta*: Any-angle path planning on grids[J]. Journal of Artificial Intelligence Research, 2010, 39: 533–579.
- [31] LIKHACHEV M, FERGUSON D, GORDON G, 等. Anytime search in dynamic graphs[J]. Artificial Intelligence, 2008, 172(14): 1613–1643.
- [32] ZIEGLER J, STILLER C. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios[C]//2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. St. Louis, MO, USA: IEEE, 2009: 1879–1884.
- [33] HOWARD T M, KELLY A. Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots[J]. The International Journal of Robotics Research, 2007, 26(2): 141–166.
- [34] ELBANHAWI M, SIMIC M. Sampling-Based Robot Motion Planning: A Review[J]. IEEE Access, 2014, 2: 56–77.

- [35] KINGSTON Z, MOLL M, KAVRAKI L E. Sampling-Based Methods for Motion Planning with Constraints[J]. Annual Review of Control, Robotics, and Autonomous Systems, 2018, 1(1): 159–185.
- [36] JEON J hwan, COWLAGI R V, PETERS S C, 等. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving[C]//2013 American Control Conference. 2013: 188–193.
- [37] KUWATA Y, TEO J, FIORE G, 等. Real-Time Motion Planning With Applications to Autonomous Urban Driving[J]. IEEE Transactions on Control Systems Technology, 2009, 17(5): 1105–1118.
- [38] KARAMAN S, FRAZZOLI E. Optimal kinodynamic motion planning using incremental sampling-based methods[C]//49th IEEE Conference on Decision and Control (CDC). Atlanta, GA, USA: IEEE, 2010: 7681–7687.
- [39] FOX D, BURGARD W, THRUN S. The dynamic window approach to collision avoidance[J]. IEEE Robotics & Automation Magazine, 1997, 4(1): 23–33.
- [40] HOWARD T M, GREEN C J, KELLY A, 等. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments[J]. Journal of Field Robotics, 2008, 25(6–7): 325–345.
- [41] NAGY B, KELLY A. TRAJECTORY GENERATION FOR CAR-LIKE ROBOTS USING CUBIC CURVATURE POLYNOMIALS[J]. : 6.
- [42] KELLY A, NAGY B. Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control[J]. The International Journal of Robotics Research, 2003, 22(7–8): 583–601.
- [43] MCNAUGHTON M, URMSON C, DOLAN J M, 等. Motion planning for autonomous driving with a conformal spatiotemporal lattice[C]//2011 IEEE International Conference on Robotics and Automation. Shanghai, China: IEEE, 2011: 4889–4895.
- [44] XU W, WEI J, DOLAN J M, 等. A real-time motion planner with trajectory optimization for autonomous vehicles[C]//2012 IEEE International Conference on Robotics and Automation. 2012: 2061–2067.
- [45] LI X, SUN Z, CAO D, 等. Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles[J]. Mechanical Systems and Signal Processing, 2017, 87: 118–137.
- [46] LI X, SUN Z, CAO D, 等. Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications[J]. IEEE/ASME Transactions on Mechatronics, 2016, 21(2): 740–753.
- [47] DUBINS L E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents[J]. American Journal of mathematics, 1957, 79(3): 497–516.
- [48] REEDS J, SHEPP L. Optimal paths for a car that goes both forwards and backwards[J]. Pacific journal of mathematics, 1990, 145(2): 367–393.
- [49] CHU K, LEE M, SUNWOO M. Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles[J]. IEEE Transactions on Intelligent Transportation Systems, 2012, 13(4): 1599–1616.
- [50] KIM J, JO K, LIM W, 等. Curvilinear-Coordinate-Based Object and Situation Assessment for Highly Automated Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 16(3): 1559–1575.
- [51] HU X, CHEN L, TANG B, 等. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles[J].

- Mechanical Systems and Signal Processing, 2018, 100: 482–500.
- [52] WERLING M, KAMMEL S, ZIEGLER J, 等. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds[J]. The International Journal of Robotics Research, 2012, 31(3): 346–359.
- [53] ZIEGLER J, BENDER P, SCHREIBER M, 等. Making Bertha Drive—An Autonomous Journey on a Historic Route[J]. IEEE Intelligent Transportation Systems Magazine, 2014, 6(2): 8–20.
- [54] TAŞ Ö S, SALSCHEIDER N O, POGGENHANS F, 等. Making Bertha Cooperate—Team AnnieWAY’s Entry to the 2016 Grand Cooperative Driving Challenge[J]. IEEE Transactions on Intelligent Transportation Systems, 2018, 19(4): 1262–1276.
- [55] LIU C, LIN C-Y, TOMIZUKA M. The Convex Feasible Set Algorithm for Real Time Optimization in Motion Planning[J]. SIAM Journal on Control and Optimization, 2018, 56(4): 2712–2733.
- [56] LIU C, LIN C, WANG Y, 等. Convex feasible set algorithm for constrained trajectory smoothing[C]//2017 American Control Conference (ACC). 2017: 4177–4182.
- [57] LIU C, TOMIZUKA M. Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification[J]. Systems & Control Letters, 2017, 108: 56–63.
- [58] CHEN J, ZHAN W, TOMIZUKA M. Constrained iterative LQR for on-road autonomous driving motion planning[C]//2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). 2017: 1–7.
- [59] CHEN J, ZHAN W, TOMIZUKA M. Autonomous Driving Motion Planning with Constrained Iterative LQR[J]. IEEE Transactions on Intelligent Vehicles, 2019: 1–1.
- [60] DOLGOV D, THRUN S, MONTEMERLO M, 等. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments[J]. The International Journal of Robotics Research, 2010, 29(5): 485–501.
- [61] FAN H, ZHU F, LIU C, 等. Baidu Apollo EM Motion Planner[J]. arXiv preprint arXiv:1807.08048, 2018.
- [62] ZHAN W, CHEN J, CHAN C, 等. Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving[C]//2017 IEEE Intelligent Vehicles Symposium (IV). 2017: 632–639.
- [63] GU T, ATWOOD J, DONG C, 等. Tunable and stable real-time trajectory planning for urban autonomous driving[C]//2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2015: 250–256.
- [64] LIU C, ZHAN W, TOMIZUKA M. Speed profile planning in dynamic environments via temporal optimization[C]//2017 IEEE Intelligent Vehicles Symposium (IV). 2017: 154–159.
- [65] LIM W, LEE S, SUNWOO M, 等. Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method[J]. IEEE Transactions on Intelligent Transportation Systems, 2018, 19(2): 613–626.
- [66] LAVALLE S M. Planning algorithms[M]. Cambridge university press, 2006.
- [67] OpenDRIVE 2010 and beyond – status and future of... - Google 学术搜索 [EB/OL]. [2019-05-28]. https://scholar.google.com/scholar?hl=zh-CN&as_sdt=0%2C5&q=OpenDRIVE+2010+and+beyond+%E2%80%93+status+and+future+of+the+de+facto+standard+for+the+description+of+road+

- networks%2C&btnG=.
- [68] BENDER P, ZIEGLER J, STILLER C. Lanelets: Efficient map representation for autonomous driving[C]//2014 IEEE Intelligent Vehicles Symposium Proceedings. 2014: 420–425.
- [69] POGGENHANS F, PAULS J, JANOSOVITS J, 等. Lanelet2: A high-definition map framework for the future of automated driving[C]//2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018: 1672–1679.
- [70] WANG H, KEARNEY J, ATKINSON K. Arc-Length Parameterized Spline Curves for Real-Time Simulation[J]. : 10.
- [71] PONAMGI M, MANOCHA D, LIN M C. Incremental algorithms for collision detection between solid models[C]//Proceedings of the third ACM symposium on Solid modeling and applications - SMA '95. Salt Lake City, Utah, United States: ACM Press, 1995: 293–304.
- [72] ZHANG Y, CHEN H, WASLANDER S L, 等. Speed Planning for Autonomous Driving via Convex Optimization[C]//2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018: 1089–1094.
- [73] GUTJAHR B, GRÖLL L, WERLING M. Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(6): 1586–1595.
- [74] CHEN J, LIU C, TOMIZUKA M. FOAD: Fast Optimization-based Autonomous Driving Motion Planner[C]//2018 Annual American Control Conference (ACC). 2018: 4725–4732.
- [75] MASUYAMA S, IBARAKI T, HASEGAWA T. The Computational Complexity of the $\langle I \rangle_m \langle /I \rangle$ -Center Problems on the Plane[J]. IEICE TRANSACTIONS (1976-1990), 1981, E64-E(2): 57–64.
- [76] ANDERSSON J A E, GILLIS J, HORN G, 等. CasADI: a software framework for nonlinear optimization and optimal control[J]. Mathematical Programming Computation, 2019, 11(1): 1–36.
- [77] On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming | SpringerLink [EB/OL]. [2019-11-15]. <https://link.springer.com/article/10.1007%2Fs10107-004-0559-y>.
- [78] KIM J, KUM D. Collision Risk Assessment Algorithm via Lane-Based Probabilistic Motion Prediction of Surrounding Vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2018, 19(9): 2965–2976.
- [79] HILLENBRAND J, SPIEKER A M, KROSCHEL K. A Multilevel Collision Mitigation Approach amp;mdash;Its Situation Assessment, Decision Making, and Performance Tradeoffs[J]. IEEE Transactions on Intelligent Transportation Systems, 2006, 7(4): 528–540.
- [80] LIU C, LIN C, SHIRAISHI S, 等. Distributed Conflict Resolution for Connected Autonomous Vehicles[J]. IEEE Transactions on Intelligent Vehicles, 2018, 3(1): 18–29.
- [81] JI J, KHAJEPOUR A, MELEK W W, 等. Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints[J]. IEEE Transactions on Vehicular Technology, 2017, 66(2): 952–964.

作者简历及在学研究成果

一、 作者入学前简历

起止年月	学习或工作单位	备注
2013 年 09 月至 2017 年 06 月	在北京科技大学机械工程专业攻读学士学位	

二、 在学期间从事的科研工作

无。

三、 在学期间所获的科研奖励

无。

四、 在学期间发表的论文

- [1] Y. Meng, Y. Wu, Q. Gu and L. Liu, "A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization," in IEEE Access, vol. 7, pp. 130530-130551, 2019. doi: 10.1109/ACCESS.2019.2940271. (SCI 检索, 已发表, 检索号: 000487543400006。)

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签名：_____ 导师签名：_____ 日期：_____

学位论文数据集

关键词*	密级*	中图分类号*	UDC	论文资助
学位授予单位名称*		学位授予单位 代码*	学位类别*	学位级别*
北京科技大学		10008	工学	硕士
论文题名*		并列题名		论文语种*
作者姓名*			学号*	s20170512
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京科技大学		10008	北京市海淀区 学院路 30 号	100083
学科专业*		研究方向*	学制*	学位授予年*
论文提交日期 *				
导师姓名*			职称*	
评阅人	答辩委员会主席*		答辩委员会成员	
电子版论文提交格式 文本() 图像() 视频() 音频() 多媒体() 其他()				
推荐格式: application/msword; application/pdf				
电子版论文出版(发布)者		电子版论文出版(发布)地		权限声明
论文总页数*				
共 33 项，其中带*为必填数据，为 22 项。				

