

Self-Driving Cars

Coursera Note

collated by decem17

Coursera: Self-driving Cars <https://www.coursera.org/specializations/self-driving-cars>

Glossary of Terms

ACC: Adaptive Cruise Control

A cruise control system for vehicles which controls longitudinal speed. ACC can maintain a desired reference speed or adjust its speed accordingly to maintain safe driving distances to other vehicles.

Ego

A term to express the notion of self, which is used to refer to the vehicle being controlled autonomously, as opposed to other vehicles or objects in the scene. It is most often used in the form ego-vehicle, meaning the self-vehicle.

FMEA: Failure Mode and Effects Analysis

A bottom-up approach of failure analysis which examines individual causes and determines their effects on the higher-level system.

GNSS: Global Navigation Satellite System

A generic term for all satellite systems which provide position estimation. The Global Positioning System (GPS) made by the United States is a type of GNSS. Another example is the Russian made GLONASS.

HAZOP: Hazard and Operability Study

A variation of FMEA (Failure Mode and Effects Analysis) which uses guide words to brainstorm over sets of possible failures that can arise.

IMU: Inertial Measurement Unit

A sensor device consisting of an accelerometer and a gyroscope. The IMU is used to measure vehicle acceleration and angular velocity, and its data can be fused with other sensors for state estimation.

LIDAR: Light Detection and Ranging

A type of sensor which detects range by transmitting light and measuring return time and shifts of the reflected signal.

LTI: Linear Time Invariant

A linear system whose dynamics do not change with time. For example, a car using the unicycle model is a LTI system. If the model includes the tires degrading over time (and changing the vehicle dynamics), then the system would no longer be LTI.

LQR: Linear Quadratic Regulation

A method of control utilizing full state feedback. The method seeks to optimize a quadratic cost function dependent on the state and control input.

MPC: Model Predictive Control

A method of control whose control input optimizes a user defined cost function over a finite time horizon. A common form of MPC is finite horizon LQR (linear quadratic regulation).

NHTSA: National Highway Traffic Safety Administration

An agency of the Executive Branch of the U.S. government who has developed a 12-part framework to structure safety assessment for autonomous driving. The framework can be found here.
https://www.safercar.gov/sites/ntsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf

ODD: Operational Design Domain

The set of conditions under which a given system is designed to function. For example, a self driving car can have a control system designed for driving in urban environments, and another for driving on the highway.

OEDR: Object and Event Detection and Response

The ability to detect objects and events that immediately affect the driving task, and to react to them appropriately.

PID: Proportional Integral Derivative Control

A common method of control defined by 3 gains.

- 1) A proportional gain which scales the control output based on the amount of the error
- 2) An integral gain which scales the control output based on the amount of accumulated error
- 3) A derivative gain which scales the control output based on the error rate of change

RADAR: Radio Detection And Ranging

A type of sensor which detects range and movement by transmitting radio waves and measuring return time and shifts of the reflected signal.

SONAR: Sound Navigation And Ranging

A type of sensor which detects range and movement by transmitting sound waves and measuring return time and shifts of the reflected signal.

Lesson 1: Taxonomy of Driving

驾驶任务由三个子任务构成 第一个子任务是感知 感知我们驾驶的时候周围的环境 这涵盖了追踪车辆的移动和 识别周围环境中的不同元素 比如路面 交通标志 车辆和行人等 我们还需要追踪所有移

动的物体 并预测它们接下来的行动 这样我们才能安全而准确地驾驶 第二个子任务是**运动规划** 这让我们能成功地到达目的地 比如说 你想从你的家驾驶到你的办公室 你需要考虑你会经过哪些路 什么时候应该变道或通过路口 怎么转弯绕过地上的坑 最后 我们需要用车辆**控制**来操作车辆 我们需要做出正确的转向 刹车和加速决定 来控制车辆在路上的位置和速度 这三个子任务构成了 驾驶任务的主体 它们需要在驾驶过程中被不断地执行

设计适用域(ODD) ODD 指明了给定的系统可以在什么样的操作环境下工作 它包括了环境 时间段 路面条件和其它驾驶过程所需要依赖的因素 在设计环节清晰地定义自动驾驶车辆的操作条件是确保系统安全性所必需的 所以 ODD 需要预先仔细设计

物体和事件检测响应(OEDR) OEDR 检测对驾驶任务有即时影响的物体和事件 并正确地对它们做出反应 OEDR 包含了自动驾驶的很大一部分 所以它和 ODD 一起被用来分类现在的自动驾驶系统

规划 Planning 规划是驾驶的另一个重要方面 即时反应已经是 OEDR 的一个部分 而规划任务主要关心的是为了到达目的地或者进行类似变道和通过路口的机动所需要的**长短期规划**

目前通行的自动驾驶分级 (the SAE Standard J3016)

L0 让我们从完全的人类感知 规划和控制开始 把它称作 0 级 在这个级别上 完全没有自动驾驶涉及驾驶员完成所有任务

L1 如果一个自动驾驶系统能协助驾驶员 完成侧向控制和纵向控制中的其中一个 我们就有了 1 级自动驾驶 自适应巡航控制是 1 级自动驾驶的一个好例子 在自适应巡航控制 (ACC) 中 系统可以控制车辆的速度 但需要驾驶员来控制方向 所以它可以完成纵向控制 但需要人来完成侧向控制 类似的车道保持辅助系统也属于 1 级 在车道保持辅助中 系统可以帮助你保持在 你的车道上 并在你滑向边界 (车道线) 的时候给出警告 今天的系统依赖于对车道线的视觉检测 和保持在车道正中的侧向控制

L2 让我们来看下一个级别 部分自动驾驶级别 在 2 级中 系统可以在特定场景下 同时完成侧向和纵向控制 一些拥有级别 2 特征的简单例子是 通用汽车超级巡航和尼桑专业驾驶辅助 它们可以控制你在侧向和正向上的运动 但驾驶员对系统的监视是始终需要的 现在 许多车辆生产商提供 2 级自动驾驶产品 包括梅赛德斯 奥迪 特斯拉和现代

L3 接下来是 3 级 在 3 级 或者叫有条件的自动驾驶中 除了控制任务之外 系统还可以在一定程度上进行 物体和事件检测响应 OEDR 然而 在 (系统) 故障时 驾驶员必须接管 2 级和 3 级的关键区别在于 (在 3 级中) 驾驶员在某些场景下不需要关注 (驾驶过程) 因为车辆会在需要介入时警示司机 这是一个有争议的级别 因为车辆并不总能知道它正在遭遇故障 3 级自动驾驶的一个例子是 奥迪 A8 Sedan 它是一个在低速环境下可以不需要监控的自动驾驶系统

L4 下一个级别是高度自动驾驶 系统可以做出使危险最小的决策 从而让驾驶员不用在紧急情况下介入 4 级系统可以自行处理紧急事件 但仍有可能 需要驾驶员介入来避免不必要的路边停靠 有了这种程度的自动化 乘客可以使用手机或观看电影 (因为) 知道系统可以处理紧急情况并确保乘客安全 然而 4 级自动驾驶依然把系统限制在有限的设计适用域 ODD 内 截止 2018 年秋天 只有 Waymo 部署了这一级别的用于公共交通的车辆 Waymo 车队可以在固定的地理区域内 在一系列操作条件下处理驾驶任务 而不需要人类驾驶员

L5 最后 在 5 级自动驾驶中系统是完全自动化的 设计适用域不再有限制 这意味着它可以在任何必须的条件下运行 5 级自动驾驶是我们的社会经历变革的节点 无人驾驶的车辆可以在任何我们需要的时候运输乘客和货物 我们还没有 5 级自动驾驶的例子

Lesson 2: Requirements for Perception

每个驾驶任务都可以被分解为两个部分 首先 我们需要理解周围在发生什么以及我们在哪里 所以我们需要感受自己的周边 其次 我们需要做出**驾驶决策** 比如 面对一个正要进入路面的行人 我们应该加速还是停下？ 回忆前一节课上对 OEDR 即物体和时间检测响应的定义 所有驾驶任务都需要某种 OEDR 也就是说 我们需要找到一种方式来识别自己周围的物体 识别附近发生的事件 并对它们做出响应 回忆上一节课对自动驾驶系统的分类 OEDR 是其中的一个标准 换句话说 如果我们要制造一辆自动驾驶汽车 我们需要有进行 OEDR 的能力

在感知任务中 我们需要识别的不同元素 首先 我们需要识别静止元素 比如车道标志 斑马线这样分割路面的东西 和（写在地上的）重要信息 比如“前方有学校” 它们都在**路面区域上** 然后是**路面之外的元素** 比如划分可驾驶区域的路缘 在路上还有一些会周期性变化的信号灯 和指示你是否可以直行 左转 右转 或停止的指示牌 还有各种交通标志 告诉你速度限制 指示方向 前方是否有医院 是否有学校等等 这些都是路面之外的元素 最后 还有**路障** 这些橙色锥体告诉你前方正在 修路 或者前方道路封闭等 这些同样是路面上的元素 然后 我们来讨论一下需要识别的动态元素 对它们运动的预测 是我们做出合理驾驶决策的前提 我们需要识别路上的**其它车辆** 四轮车 比如卡车 巴士和轿车等等 我们也需要识别并预测两轮车 比如摩托车 自行车等等 它们都是比四轮车心动更加自由的系统 所以它们更难预测 最后 我们还需要能预测 **周围行人的运动** 行人的行动和机动车非常不同 因为行人被认为 比机动车不稳定得多 这是源自 人的行动方式更加自由

感知的另一个重要目标是**自我定位** 我们需要时刻估计自身的位置和运动方式 知道我们的位置和在环境中如何移动 对做出明智 安全的驾驶决策至关重要 用于自我估计的数据来自 GPS IMU (惯性测量单元) 和测距传感器 它们需要被 组合起来提供对自身位置连续 清晰的认知

感知的困难之处 首先 进行**鲁棒的感知**是一个巨大的挑战 目标检测和分割可以 由现代机器学习方法完成 但还需要很多研究 来提升它们的可靠性和表现 来和人类级别的能力相提并论 大规模数据集对这项工作非常重要 有了更多的训练数据 我们的分割和检测模型 会表现得更加出色且鲁棒 但对不同可能的机动车类型 天气条件和路面情况 收集和标记数据是一个 非常昂贵且耗时的过程 其次 感知并不对**传感器的不稳定性**免疫 在很多情况下 可见性不能得到保证 GPS 数据可能会失效 雷达和激光雷达的数据会带有噪音 每个依赖于这些传感器的系统 都必须把这些不确定性纳入考虑范围 这就是为什么为每一个感知任务设计 可以接受传感器不稳定 和数据无效的系统如此重要 对相机和激光雷达还有遮蔽和反射的影响 它们会用有歧义的信息干扰感知模型 影响它们对物体位置的准确估计 还有一些其它的影响 比如剧烈的光照变化 镜头光晕 或者 GPS 在隧道中丢失信号 使得一些传感器的数据完全不可用 感知算法需要多重冗余的数据来源 来解决传感器数据丢失 最后，**天气和降水**会影响传感器的输入数据 所以我们至少需要有一些 在不同的天气条件下都可用的传感

器 比如说，雷达

Lesson 3: Driving Decisions and Actions

制定驾驶决策是属于规划这一大范围的 当我们制定驾驶决策时 我们通常要做出三种类型的决策 第一种是**长期决策规划** 例如这样的问题 我如何从纽约导航至洛杉矶或是从家至工作地点？ 通过回答这个问题 我们有一个任务计划 整个驾驶任务的高级计划 你今天使用的地图 APP 能够给你 一些驾驶说明：走那条路 要进入哪个车道 等等 但是驾驶不仅仅需要这些 第二类是**短期的驾驶决策** 例如现在换车道安全吗？ 或者我应该在什么时候在交叉路口左转？ 驾驶还需要一些**即时决策或反应** 这些决策包括控制和轨迹规划 比如 我如何在这条弯曲的道路上保持车道？ 我应该使用什么转向输入？ 我应该加速还是刹车？ 如果是这样,加减速多少？

解决多层次决策规划挑战的一种方法是**被动规划** 在被动规划中 我们定义一系列规则 需要考虑 本车辆和其他环境中物体 的现有状态并采取即时操作 所以这些规则只考虑当下状态 而不考虑对未来的预测 这类规则的一些例子可以是 如果道路上有行人 请停下来 或者如果速度限制发生变化 调整速度以满足限制 在这两个规则中 我们仅仅观察现在发生的事情 并基于即时可用信息制定决策

在**预测性规划**中 我们会对环境中的其他因素进行预测 例如车辆和行人 随时间怎么移动 我们使用当前状态和**预测的信息**来定义我们所有的决策 一些预测性规划中的规则示例包括 那辆车在过去的十秒钟一直停止不动 在接下来的几秒钟内它可能仍会停止不动 所以 也许我可以安全地通过它 或有行人在马路上闲逛 当我的车辆接近他们时 他们将进入我们的车道 让我减速 让他们有机会通过我前方的路 如你所见 这是一种更自然的思考方式 并与人如何操作车辆密切相关 我们预测道路上其他物体 在未来的位置 在我们做出决策前 但是 这种类型的规划 依靠于对环境中其他参与者行为的精确预测 这极大地增加了感知任务的复杂性 尽管如此 **预测性规划是自动驾驶汽车的主要规划方法** 因为它极大地扩展了车辆可以安全处理的场景

Lesson 1: Sensors and Computing Hardware

传感器就是任何可以测量 或检测环境某些属性的装置 或能够随时间改变环境属性的装置 传感器大致分为 两种类型 取决于他们所记录的环境属性 如果他们记录环境的属性,他们是**外部感应传感器 exteroceptive** 另一方面 如果传感器 记录了本身车辆的属性 他们是**本体感应传感器 proprioceptive**

外部感应传感器

相机是一种被动式 光线收集的 传感器 非常适合拍摄 场景内的丰富细节信息 事实上 一些团体认为 相机是唯一真正需要的传感器 对于自动驾驶来说 但是,目前技术表现 仍不可能仅使用视觉信息来完成 在谈论相机时 我们通常谈论三个重要的比较指标 我们挑选相机 基于他们的**分辨率** 分辨率是创建图像的像素数 因此,这是判定图像质量 的一种方法 我们还可以基于**视线范围** 选择相机 视线范围由相机的 水平和垂直角度范围所定义 并且可以通过选择镜头和变焦将其调整 另一个重要指

标 是动态视觉范围 摄像机的动态视觉范围是 图像中最暗 和最亮色调的差异 高动态视觉范围对于自驾车至关重要 因为 会遇到多变的光线条件 尤其是在夜间行驶时 在相机和镜头选择之间 有一个重要的权衡 取决于视线范围和分辨率的选择 较宽的视野允许 在环境中有较大的可视区域 但从一个特定对象吸收 光的像素更少 随着视线范围的增加 我们需要将分辨率提高到 仍然能够以相同质量感知到 可能遇到的 多样的环境信息 相机的其他一些属性也 会影响感知 例如焦距 景深和帧频

LIDAR(光学雷达)代表光线检测和测距传感器 光学雷达感知涉及 向环境中发射光束 并测量反射回的信号 通过测量返回的 光量和光束的飞行时间 范围内的**物体密度和反射物体**都可以被估计 光学雷达通常包括一个 带有多个光源的旋转原件 并输出三维点云图 这对于 评估场景几何十分有用 由于它是一个具有自身光源的 有源传感器 光学雷达不受环境照明情况 的影响 因此,光学雷达并不会面临 如同相机在 恶劣或多变光线条件下所受到的挑战 让我们来讨论挑选光学雷达的 重要比较指标 第一个指标是**光源数量** 常见尺寸包含的 8 16 32 和 64 个光源 第二个指标是 **每秒可以收集的点数** 点数收集越快 3D 点云信息可以更详细 另一个指标 是**转速** 转速越高 3D 点云的更新速度越快 检测范围也很重要 并取决于光源的 输出功率 最后我们有**视野指标** 是指光学雷达传感器的可视角度范围

RADAR 雷达传感器已经 可以超过光学雷达 测距更长并能可靠检测环境中的大型物体 它们在恶劣的天气尤其有用 因为它们大多不受 大降雨的影响 让我们讨论一些选择雷达 的比较指标 选择雷达可基于 **检测范围 视野范围 位置和速度的测量精确性** 雷达通常也可以被认为是 具有宽视野角度 但视距范围较短(WFOV, short range) 或是视野狭窄 但视距范围更大(NFOV, long range)

Ultrasonic / SONAR 声纳 最初以 声音导航和测距命名 使用声波测量范围 声纳是在廉价测距设备中 测量短距离的传感器 这使它们适合 **停车场景** 其中本体车辆需要 靠近其他车辆的情况下移动 声纳另一个优点是 它们的成本很低 而且,就像雷达和光学雷达一样 **声纳不受照明和降雨条件影响** 声纳可基于一些关键指标进行选择: **可测量的最大范围 检测视野及其成本**

本体感应传感器

最常见的是 **GNSS 和 IMU**

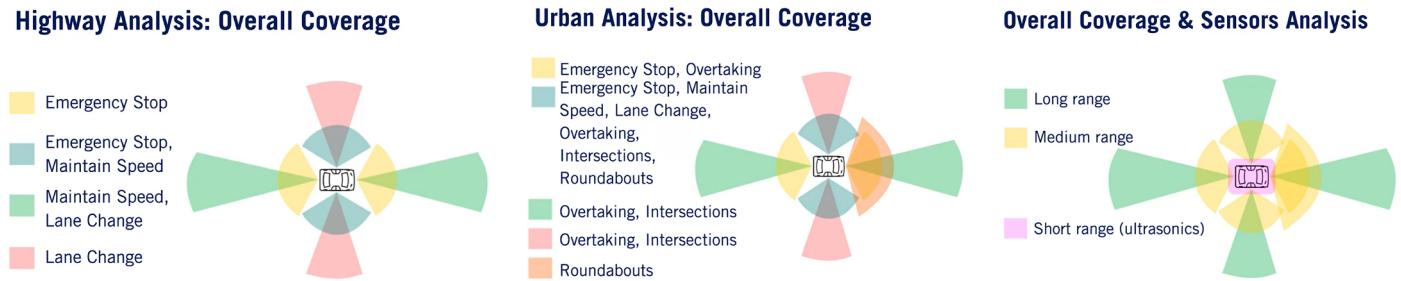
GNSS 接收器用于测量 **本体车辆的位置,速度 有时测量车辆前进方向** 精度很大程度上取决于 实际定位方法 和使用的校正方法 除此之外, **IMU** 还测量 **旋转角速度 本体车辆的加速度** 以及组合的测量可用来 **估计车辆的 3D 方向 前进方向 heading** 对于 车辆控制是最重要的 最后, 我们有 **车轮里程传感器 wheel odometry** 该传感器跟踪车轮旋转速度 使用这些来估计**本体车辆速度和前进方向的变化率** 这同样是车辆上记录行驶里程的传感器

计算硬件

接收所有传感器数据 并输出驾驶车辆所需的命令 大多数公司更喜欢设计自己的 计算系统来匹配特定的 传感器和算法需求 传感器的测量必须 确保时间一致 以便传感器融合能正常运行

Lesson 2: Hardware Configuration Design

如何放置这些传感器来生成环境的完整视图

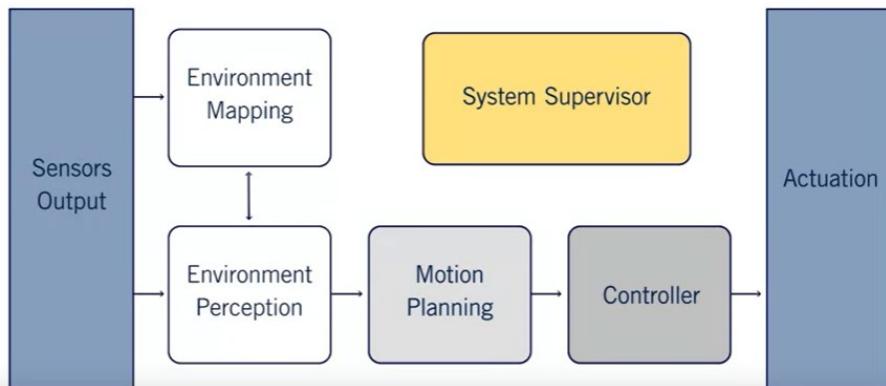


总之，我们对传感器的选择应该由我们想要执行的操作的要求驱动。它应该包括远程传感器以应对纵向的危险 和用于全向感知的宽视场传感器。配置的最终选择还取决于我们对运行条件的要求、由于故障导致的传感器冗余 以及预算。

Lesson 3: Software Architecture

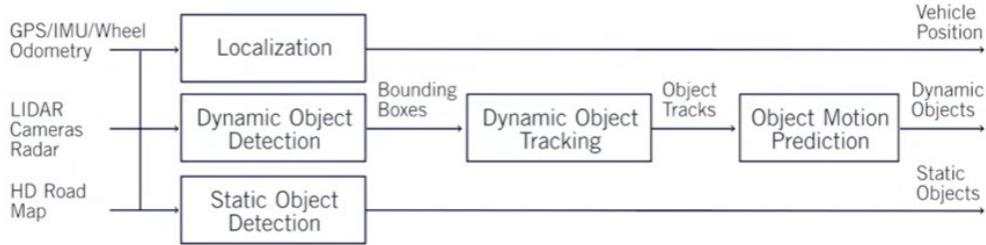
针对自动驾驶汽车软件栈的高级软件架构

Software Architecture | High-level



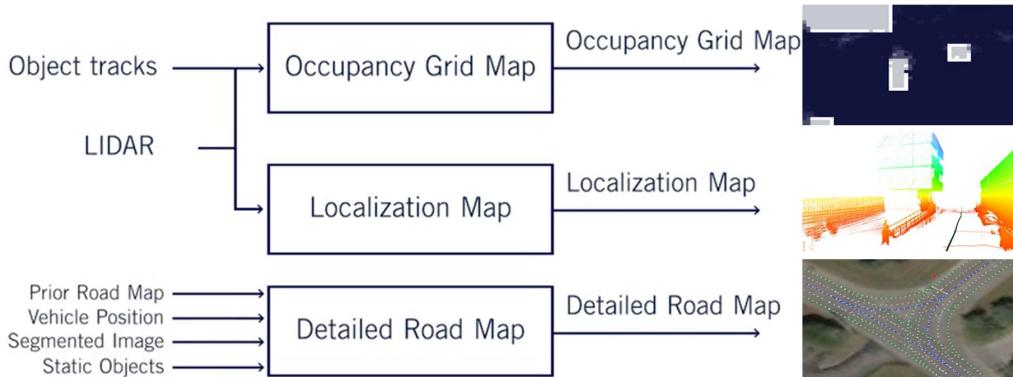
汽车使用各种传感器观察周围的环境。原始传感器测量结果被传递到两组模块中，专用于了解汽车周围的环境。**环境感知模块**具有两个关键功能：第一，识别在空间上无人驾驶车当前所处的位置；第二，分类和定位用于驾驶任务环境的重要元素。这些元素的例子包括其他汽车、自行车、行人、道路、道路标记和道路标志以及任何直接影响驾驶行为的物体。**环境地图模块**创建一组地图，其定位无人驾驶车周围环境中的对象用于一系列不同用途。用途包括从避免碰撞到运动跟踪和运动计划。第三个模块称为**运动规划**，基于由感知模块和绘图模块提供的所有信息，运动规划模块做出采取什么动作以及在何处驾驶的所有决定。运动规划模块的主要输出是安全、高效和舒适的规划路径，使车辆朝向其目标移动。然后，规划路径由第四个模块即**控制器**执行。控制器模块得到路径并确定最佳转向角、油门位置、制动踏板位置和档位设置，以精确地遵循规划路径。第五个也是最后一个模块是**系统监控**。系统监控器监控软件堆栈的所有部分，以及硬件输出，以确保所有系统都能按预期工作。系统监控还负责向安全驾驶员通知驾驶系统中发现的任何问题。

环境感知



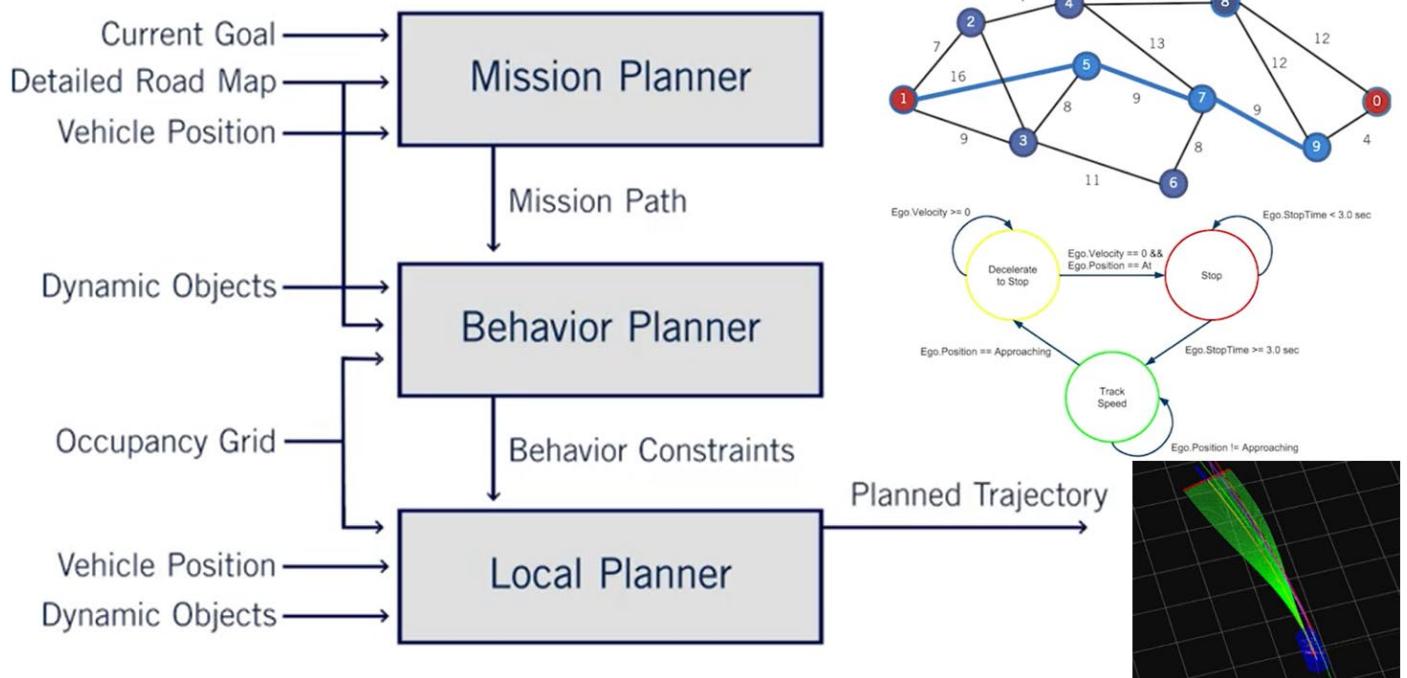
如前所述 感知栈有两个重要部分 即在空间中自我定位 以及 分类和定位环境的重要元素 定位模块接收多个信息流 如当前 GPS 位置 IMU 测量单元和车轮里程计 然后定位模块将它们组合起来输出准确的车辆位置 为了获得更高的准确性 一些定位模块还包含 LIDAR 和相机的数据 分类和定位的问题是 在环境元素中可以分为两个部分： 第一部分 检测环境的动态对象 第二部分 检测环境中的静态对象 动态对象检测模块使用一组摄像机输入 以及 LIDAR 点云来创建场景中动态对象周围的 3D 边界框 3D 边界编码物体的 class 或 type 以及物体的确切位置 方向和大小 一旦检测到 跟踪模块会随时间跟踪动态对象 跟踪器模块不仅提供动态物体的当前位置 还提供其通过环境的路径的历史 路径的历史记录与路线图一起使用以便预测所有动态物体的未来路径 这通常由预测模块处理 预测模块组合关于动态物体和当前环境的所有信息 从而预测所有动态物体的路径 静态对象检测模块还依赖于相机输入和 LIDAR 数据的组合来识别场景中的重要静态对象 这些重要数据包括无人驾驶车辆的当前车道 以及如信号和交通信号灯等监管要素的位置

环境绘图



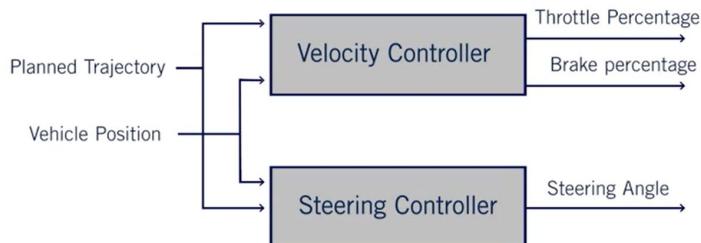
环境地图创建了在无人驾驶车周围当前环境下的不同表示类型 我们简要讨论了三种类型的地图 占据栅格图 **occupancy grid map** 定位图 **localization map** 以及细节道路图 **detailed road map** 占据网格图是所有静态对象的地图 在车辆周围环境中 LIDAR 主要用于构建占据栅格图 首先将一组过滤器应用于 LIDAR 数据以使得可以用于占据栅格 例如，移除掉可以移动的表面点和动态对象点 占据栅格图将环境表示为一组网格单元并将每个单元被占用的概率相关联 这使我们可以处理测量数据中的不确定性 并随时间改善地图 由 LIDAR 或相机数据构建的定位图由定位模块使用 用以改进状态估计 在驾驶中 将传感器数据与该地图进行比较以确定 汽车相对于定位图的运动 然后将该运动与其他本体感受式传感器的信息组合 以准确地定位车辆 细节道路图提供了路段的地图 其表示 无人驾驶车当前驾驶的驾驶环境 该地图捕捉信号和车道标记 可以用于运动规划中 这个地图传统上是预先记录的数据和感知栈从当前静态环境收集的传入信息的组合 环境绘图和感知模块相互作用以显著提高这两个模块的性能 例如 感知模块提供静态环境信息 可以用于更新细节道路图 细节道路图也能用于给预测模块创建更加准确的动态物体预测

运动规划



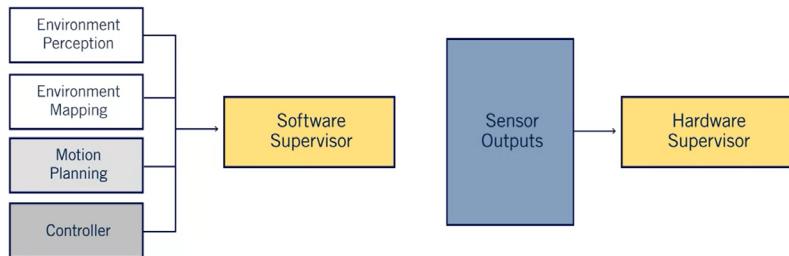
运动规划模块组合并使用来自环境绘图和感知模块的输出创建穿过环境的路径。自动驾驶汽车的运动规划是一项具有挑战性的任务，很难在单一的集成过程中解决。相反，现在大多数自动驾驶汽车使用的是一种**分解**，它将问题分为以下几个抽象层。在顶层，**任务规划器 mission planner** 处理**长期规划**，并在驾驶任务的整个范围内定义任务，从当前位置，通过道路网络到最终目的地。为了找到完整的任务，任务规划器确定连接起点和终点的最佳路段序列，然后将其传递到下一层。**行为规划器 behavior planner** 是下一个抽象层次，解决**短期规划**问题。行为规划器负责建立一套安全行动或策略，以便在沿任务路径行驶时执行。行为规划器决策的一个例子是，给定所需速度和附近车辆的预测行为，车辆是否应并入相邻车道。随着决策的机械化，行为规划器还提供了一组执行每个动作的约束，例如在切换前在当前车道上停留多长时间。最后，**局部规划器 local planner** 执行**即时或反应式规划**，并负责定义要驱动的特定路径和速度剖面。考虑到当前环境和机动所施加的所有限制，局部规划必须是平稳、安全和高效的。为了创建这样的计划，局部规划器将行为规划器、占据栅格、车辆运行限制和环境中其他动态对象提供的信息结合起来。局部规划器的输出是一个规划的轨迹，它是未来短时间内的期望轨迹和速度剖面的组合。

控制



一个典型的车辆控制器获取给定的轨迹，并将其转化为一组精确的驱动命令，以便车辆应用。典型的控制器将控制问题分为**纵向控制**和**横向控制**。横向控制器输出保持计划轨迹所需的**转向角**，而纵向控制器调节油门 **throttle**、档位 **gears** 和制动 **braking** 系统以达到正确的速度。两个控制器都计算当前误差和局部规划的跟踪性能，并调整当前驱动命令以使误差最小化。

系统监控



系统监控模块持续监控自动驾驶汽车的各个方面，并在发生子系统故障时发出相应的警告。有两个部分，**硬件监控器**和**软件监控器**。硬件监控器持续监视所有硬件组件，以检查是否存在任何故障，例如传感器损坏、测量值丢失或信息降级。硬件监控器的另一个职责是持续分析硬件输出，找出与自动驾驶汽车编程执行的域不匹配的任何输出。例如，如果其中一个摄像头传感器被纸袋挡住，或者正在下雪并损坏了激光雷达点云数据。软件监控器有责任验证此软件栈，以确保所有元件以正确的频率按预期运行，并提供完整的输出。软件主管还负责分析所有模块输出之间的不一致性。

Lesson 4: Environment Representation

定位图。此地图是当汽车在环境中移动时，使用一组连续的 **LIDAR** 点或相机图像特征创建。然后，该地图通过定位模块结合 GPS、IMU 和车轮里程计使用，以便始终准确地估计车辆的精确位置。

定位地图使用记录的 LIDAR 点或图像，这些点或图像组合在一起，形成环境的点云表示。当接收到新的 LIDAR 相机数据时，将其与定位地图进行比较，并通过将新数据与现有地图对齐来创建车辆位置的测量。然后将该测量值与其他传感器相结合，以估计本身的运动，并最终用于控制车辆。定位图可能相当大，并且存在许多方法来压缩它的内容并只保留定位所需的那些特征。

占据栅格图。占据栅格图还是使用一组连续的 LIDAR 点来构建环境地图，该地图指示所有静态或静止障碍物的位置。该地图用于为无人驾驶车规划安全无碰撞的路径。

占据栅格是车辆周围环境中静态物体的二维或三维离散化地图。创建此地图是为了识别自动驾驶汽车周围的所有静态对象，使用点云作为输入。分类为静态的对象包括树、建筑物、路缘和所有其他不可驾驶的表面。由于占用网格仅表示环境中的静态对象，因此必须首先删除所有动态对象。这是通过移除所有 LIDAR 点来完成的，这些 LIDAR 点位于由感知栈识别的检测到的动态对象的边界框内。接下来，不会干扰车辆的静态物体也会被移除。例如悬垂的树枝。作为这些步骤的结果，只有来自环境中静态对象的相关 LIDAR 点仍然存在。过滤过程并不完美，因此不可能盲目相信剩下的点是障碍物。因此，占据栅格通过跟踪栅格单元在一段时间内被占据的可能性，概率地表示环境。然后依赖此地图为车辆创建无碰撞的路径。

细节路线图。它包含详细的位置，所有监管的要素，监管的属性以及和车道标记。此地图用于规划从当前位置到最终目标的路径。

细节路线图是一张可以由自动驾驶汽车驾驶的完整道路网的地图。此地图包含有关道路车道的信息，

以及可能影响车道的任何交通管制要素。细节路线图被用来规划一条安全有效的自驾车路线。细节路线图可以通过以下三种方式之一创建。完全在线 **fully online**、完全离线 **fully offline** 或离线创建在线更新 **created offline and updated online**。

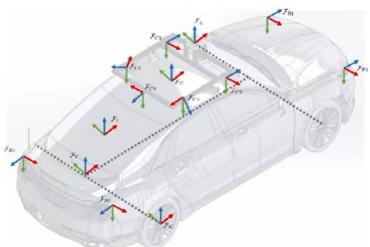
完全在线创建的地图在很大程度上依赖于感知栈中的静态对象比例来准确标记和正确定位所有相关的静态对象以创建地图。这包括当前驾驶环境中的所有车道边界、任何规则元素（如红绿灯或交通标志）、车道的任何规则属性（如右转标志或人行横道）。由于实时创建此类地图的复杂性，很少使用这种地图创建方法。

完全离线创建的地图通常是通过多次收集给定道路的数据来完成的。配备高精度传感器的专用车辆定期沿道路行驶，以构建离线地图。一旦收集完成，这些信息就可以通过使用静态物体感知和人类注释和校正的自动标记来进行标记。这种地图创建方法虽然可以生成非常详细和精确的地图，但无法对不断变化的环境做出反应或适应。

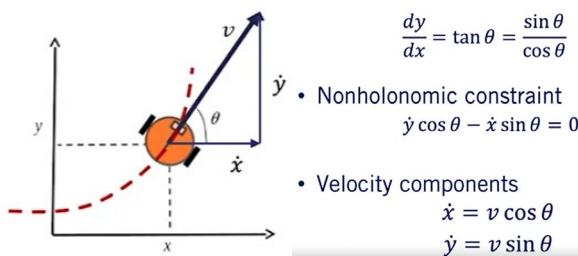
第三种是离线创建路线图，然后用新的相关信息在线更新路线图。这种地图创建方法利用了这两种方法，创建了一个高度精确的路线图，可以在驾驶时进行更新。

Lesson 1: Kinematic Modeling in 2D

坐标系: 全局坐标系 inertial frame (fixed, usually relative to earth), 车身坐标系 body frame (origin at vehicle centre of gravity, or centre of rotation), sensor frame (attached to sensor)

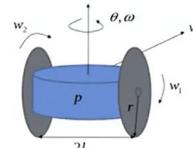


非完整约束 the kinematic nonholonomic constraint: 汽车的运动由于车轮的方向受到限制(可以在滚动时向前滚动和转动，但不能直接横向移动)，这个约束被称为非完整约束，这意味着它限制了汽车位置的变化速度。



- Continuous time model:

$$\begin{aligned}\dot{x} &= \left[\left(\frac{rw_1 + rw_2}{2} \right) \cos \theta \right] \\ \dot{y} &= \left[\left(\frac{rw_1 + rw_2}{2} \right) \sin \theta \right] \\ \dot{\theta} &= \left(\frac{rw_1 - rw_2}{2l} \right)\end{aligned}$$



- Discrete time model:

$$\begin{aligned}x_{k+1} &= x_k + \left[\left(\frac{rw_{1,k} + rw_{2,k}}{2} \right) \cos \theta_k \right] \Delta t \\ y_{k+1} &= y_k + \left[\left(\frac{rw_{1,k} + rw_{2,k}}{2} \right) \sin \theta_k \right] \Delta t \\ \theta_{k+1} &= \theta_k + \left(\frac{rw_{1,k} - rw_{2,k}}{2l} \right) \Delta t\end{aligned}$$

瞬时旋转中心 instantaneous centre of rotation (ICR)

Lesson 2: The Kinematic Bicycle Model

无滑动条件 no slip condition: 要求车轮不能横向移动或纵向滑动

- If the desired point is at the center of the rear axle
- If the desired point is at the center of the front axle
- If the desired point is at the center of the gravity (cg)

$$\begin{aligned}\dot{x}_r &= v \cos \theta \\ \dot{y}_r &= v \sin \theta \\ \dot{\theta} &= \frac{v \tan \delta}{L}\end{aligned}$$

$$\begin{aligned}\dot{x}_f &= v \cos(\theta + \delta) \\ \dot{y}_f &= v \sin(\theta + \delta) \\ \dot{\theta} &= \frac{v \sin \delta}{L}\end{aligned}$$

$$\begin{aligned}\dot{x}_c &= v \cos(\theta + \beta) \\ \dot{y}_c &= v \sin(\theta + \beta) \\ \dot{\theta} &= \frac{v \cos \beta \tan \delta}{L} \\ \beta &= \tan^{-1} \left(\frac{l_f \tan \delta}{L} \right)\end{aligned}$$

State-space Representation

- Modify CG kinematic bicycle model to use steering rate input

$$\begin{aligned}\text{State: } &[x, y, \theta, \delta]^T & \text{Inputs: } &[v, \varphi]^T \\ \dot{x}_c &= v \cos(\theta + \beta) & & \\ \dot{y}_c &= v \sin(\theta + \beta) & & \\ \dot{\theta} &= \frac{v \cos \beta \tan \delta}{L} & & \\ \dot{\delta} &= \varphi & \text{Modified Input: rate of change of steering angle} &\end{aligned}$$

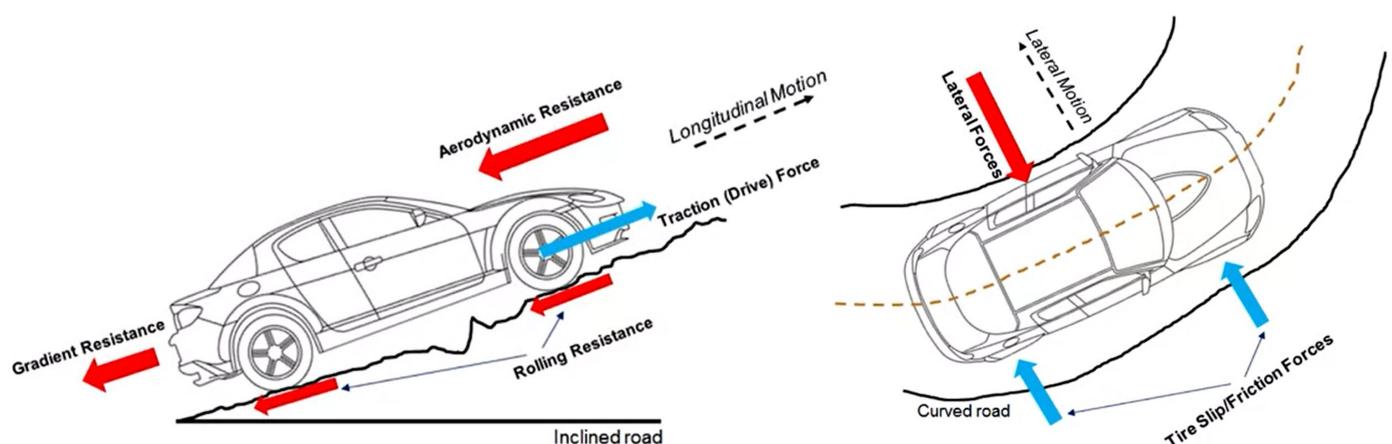
Lesson 3: Dynamic Modeling in 2D

在车辆建模中，当车辆以高速转向或道路很滑时无滑条件定义的假设可能不再存在。

考虑车身横滚和俯仰的完整三维车辆模型。由于任意倾斜的道路，每个轮胎上不同的力和力矩形成了一个非常复杂的模型。因此可以将我们的模型分为两个 2D 模型，将我们的车辆控制分割为转向控制和油门/制动控制问题。基于这些原因，我们将为我们的自动驾驶汽车建立一个单独的纵向 longitudinal 和横向 lateral 动力学模型。

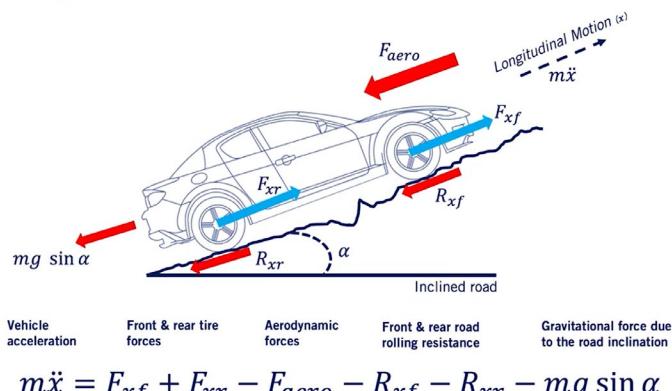
纵向模型考虑车辆在倾斜道路上行驶。我们限制车辆在 XZ 平面上运动。作用在车身和轮胎上的力包括牵引力 Traction force、滚动阻力 Rolling resistance、空气动力 Aerodynamic force 和重力引起的坡度阻力 Gradient resistance force due to gravity。

类似地，横向车辆动力学模型可以从 XY 平面上的运动发展而来。在这个 2D 模型中，还有一些横向力作用在车辆上，如滑移力 slip forces 和离心力 centrifugal forces。

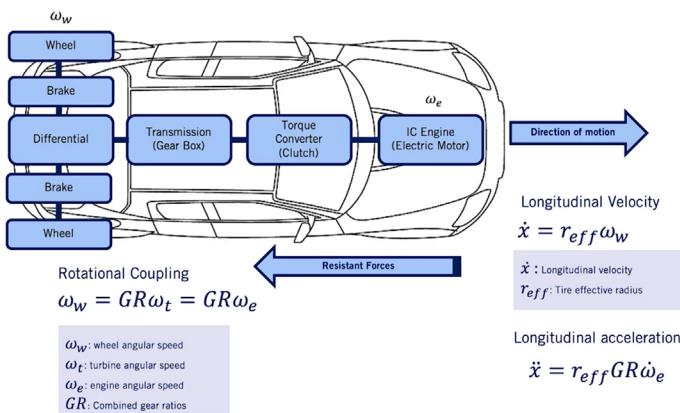


Lesson 4: Longitudinal Vehicle Modeling

Longitudinal Vehicle Model



Powertrain Modeling



Simplified Longitudinal Dynamics

- The full longitudinal dynamics

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg \sin \alpha$$

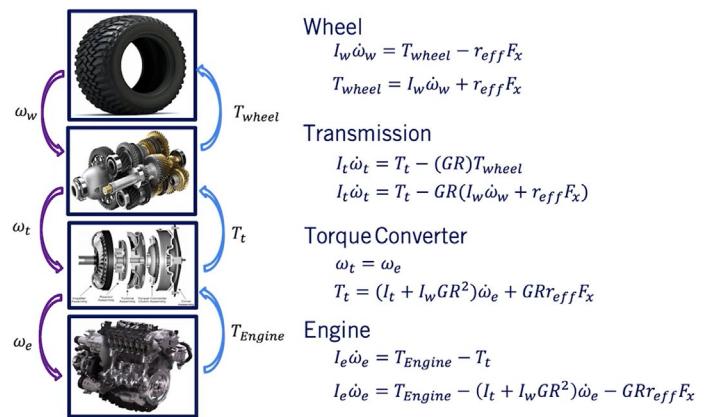
- Let F_x - total longitudinal force: $F_x = F_{xf} + F_{xr}$
- Let R_x - total rolling resistance: $R_x = R_{xf} + R_{xr}$
- Assume α is a small angle: $\sin \alpha = \alpha$

- Then the simplified longitudinal dynamics become

$$m\ddot{x} = F_x - F_{aero} - R_x - m g \alpha$$

Inertial Term Traction Force Total Resistant Forces (F_{Load})

Power Flow in Powertrain



Engine Dynamics

- Tire force in terms of inertia and load force:

$$F_x = m\ddot{x} + F_{load} = m r_{eff} G R \dot{\omega}_e + F_{load}$$

- Combining with our engine dynamics model yields:

$$(I_e + I_t + I_w GR^2 + m(GR^2)r_{eff}^2)\dot{\omega}_e = T_{Engine} - (GR)(r_{eff}F_{load})$$

I_e

- Finally, the engine dynamic model simplifies to

$$J_e \dot{\omega}_e = T_{Engine} - (GR)(r_{eff}F_{load})$$

Total Load Torque (T_{load})

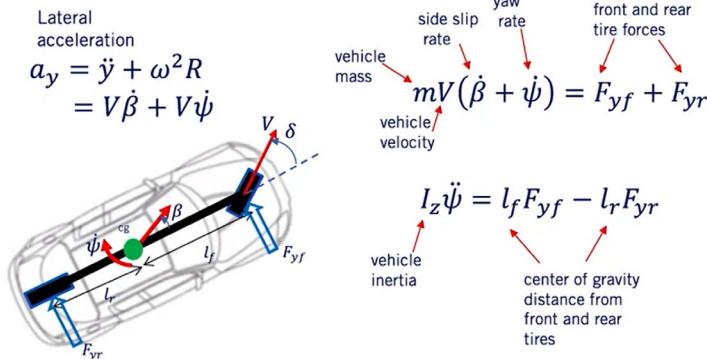
Lesson 5: Lateral Dynamics of Bicycle Model

为开始建立自行车模型的横向动力学模型，将做出以下假设：首先，假定前向纵向速度恒定。这样做是为了解耦我们的横向和纵向动态模型，这大大简化了我们的建模任务，但当加速或减速为曲线时会导致建模不准确。第二，与自行车运动学模型一样，前轴和后轴的左右车轮都集中在一个单独的车轮上。所以，这个假设把四轮转换成两轮自行车模型。最后，假设其他非线性效应，如悬架运动、路面倾斜和空气动力，可以忽略不计。在实践中，这些效应会对轮胎力产生重大影响。所以，在某些情况

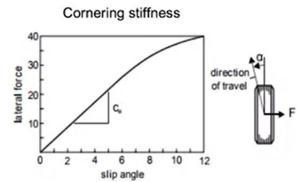
下，这是一个限制性的假设，但对于我们的目的来说已经足够了。使用车辆重心作为动力学模型的参考点。

Lateral Dynamics

- Lateral dynamics can be written as



Front and Rear Tire Forces



- C_f : linearized cornering stiffness of the front wheel

$$F_{yf} = C_f \alpha_f = C_f \left(\delta - \beta - \frac{l_f \dot{\psi}}{V} \right)$$
- C_r : linearized cornering stiffness of the rear wheel

$$F_{yr} = C_r \alpha_r = C_r \left(-\beta + \frac{l_r \dot{\psi}}{V} \right)$$

Standard State Space Representation

- State Vector: $X_{lat} = [y \quad \beta \quad \psi \quad \dot{\psi}]^T$

$$A_{lat} = \begin{bmatrix} 0 & V & 0 & 0 \\ 0 & -\frac{C_r + C_f}{mV} & 0 & \frac{C_r l_r - C_f l_f}{mV^2} - 1 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{C_r l_r - C_f l_f}{I_z} & 0 & -\frac{C_r l_r^2 + C_f l_f^2}{I_z V} \end{bmatrix}$$

$$B_{lat} = \begin{bmatrix} 0 \\ C_f \\ \frac{mV}{I_z} \\ 0 \\ C_f l_f \\ I_z \end{bmatrix}$$

$$\dot{X}_{lat} = A_{lat}X_{lat} + B_{lat}\delta$$

Motion planning for self-driving cars

Welcome to the Course

在一个非常高的层次上，运动规划问题的任务是生成从 A 点到 B 点的自动驾驶汽车所需的路径和速度。为此，我们需要我们在前面的课程中描述的感知和感知信息。因为运动规划需要了解周围的汽车、行人和环境中的其他障碍物。由于问题的复杂性，通常将规划问题分解为更小的子问题。

任务规划的三阶段层次结构，其中包括地图级导航 map level navigation，您可能熟悉使用在线地图工具规划行程，行为规划 behaviour planning 确保我们的驾驶行为遵循道路规则，以及局部规划 local planning，确保我们的路径平滑无碰撞。

Lesson 1: Driving Missions, Scenarios, and Behaviour

运动规划问题的任务是在遵循道路规则的同时，以安全和舒适的方式将车辆导航到目的地。

在高层次上，自主驾驶任务 autonomous driving mission 是从地图上的 A 点到 B 点。

mission planning 是一个 higher-level planning. 为了简化任务规划过程，自主驾驶任务规划将许多重要的低级变量从问题中抽象出来。然而，这些较低层次的变量，如道路结构、障碍物和道路上的其他主体，对自主驾驶运动规划问题至关重要。这些低级变量定义了不同的驾驶场景。

目标是找到一条 most efficient path in term of time or distance.

首先，让我们介绍一些与道路结构相关的常见场景。所谓道路结构 road structure，我们指的是车道边界 lane boundaries 和与驾驶员相关的监管要素 regulatory elements。最简单的情况是在车道上行驶，这通常称为 lane maintenance。在这种情况下，我们的目标是尽量减少偏离路线中心线的距离，并达到我们的参考速度 reference speed，这通常是速度限制 speed limit，以确保有效地行驶到我们的目的地。

更复杂的情况是，汽车必须执行车道变换操作。即使在没有动态障碍物存在的情况下，我们还需要在横向和纵向加速度约束，道路的速度限制和执行机动的时间范围内优化车道变换轨迹的形状。显然，所有这些参数都会影响轨迹的形状，从而导致要么缓慢的被动 passive 换道，要么更激进 aggressive 更不舒适的换道。

第三种常见情况是，汽车需要左转或右转。当自动驾驶汽车必须处理交叉口时，这通常是必需的。随着车道的改变，转弯 turn 的形状和激进性 aggressiveness 将根据情况而变化。此外，自主车辆可以采取的行动的可行性是受周围环境的状态影响的。例如，即使十字路口没有障碍物，自动驾驶车辆也不能在红灯处左转。

最后一个例子，U-turn。这对于汽车需要有效改变方向的特定导航场景非常重要。与车道变换以及右转弯和左转弯一样，U 形转弯的形状取决于影响轨迹形状的参数(车速和加速度限制)。掉头还很大程度上取决于周围环境的状况，因为在十字路口掉头并不总是合法的，这取决于您所在国家的法律。

道路结构并不是道路上情况的唯一因素。静态和动态障碍将极大地改变驾驶场景 driving scenario 的结构以及确定场景所需行为的难度。

静态障碍物限制了我们的路径可以占据的位置 static obstacles restrict which locations our path can occupy.

最重要的动态障碍物经常是前一个车辆 leading vehicle，需要保持安全的时间间隔 time gap. 即有两个相互竞争的利益 competing interests: 我们希望保持尽可能接近我们的参考速度，同时为了安全起见，我们的车和领头车之间保持一个时间间隔。

动态障碍物也会影响我们的转弯和换道方案。根据其位置和速度，可能有不同的执行时间窗口 time windows of execution。这些窗口将是估计值，因为它们将基于环境中所有其他代理的预测。例如，当我们在执行左转之前等待一个交叉口清空时，自主车辆决定执行的行为在很大程度上取决于迎面而来的车辆的行为。如果附近有迎面而来的车辆快速向我们驶来，那么唯一安全的行为就是暂停转弯。

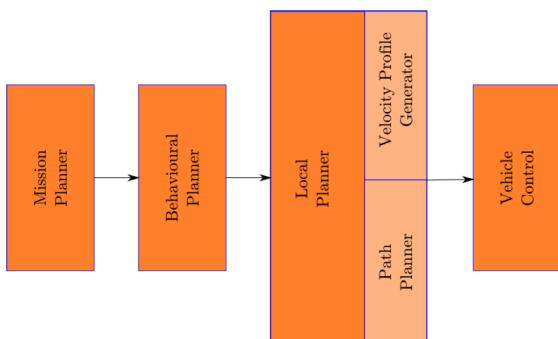
但是，如果自动驾驶车辆与迎面而来的车辆之间的距离较长，则自动驾驶车辆将有机会转弯。需要使用估计 estimation 和预测 prediction 来计算这些机会窗口。

现在，并不是所有的动态障碍物都是汽车，因为还有其他类型的障碍物，如自行车、卡车和行人。他们每个人都有自己的行为，在路上都有自己的规则。尽管如此复杂，这些场景的大多数行为都可以被认为是高级动作 actions 或机动策略 maneuvers 的简单组合。这些高级机动的一个例子是速度跟踪 speed tracking、减速停车 decelerate to stop、保持停车 stay stopped、让步 yield 和紧急停车 emergency stop。

速度跟踪是正常的驾驶行为。我们有一个参考速度或速度限制，我们在车道上前进时保持这个速度。减速停车是很自然的。如果前面有停车标志，我们需要平稳地减速停车以保持舒适。某些监管要素需要保持停止。如果我们在红灯处，那么我们需要停下来直到红灯变绿。一些监管要素也需要屈服。最值得注意的是，如果我们在一个让行标志处，并且有比我们优先级更高的交通流，我们需要减速并等待，直到我们可以继续前进。最后，当自动驾驶汽车检测到问题，车辆需要立即停车并靠边停车时，会发生紧急停车。这些高级行为还可以通过导航行为来增强，例如执行车道变换和转弯。通过将所有这些结合起来，我们现在可以涵盖最基本的驾驶场景。虽然这组策略覆盖范围很广，但需要注意的是，这组行为列表并非详尽无遗，而且有许多方法可以增加行为复杂性，以处理更多有趣的场景。

就一系列驾驶场景而言，我们真正开始触及表面。有许多不寻常的例子，使自动驾驶的问题和极具挑战性的任务。例如，假设你有一个乱穿马路 jaywalking 的行人，那么我们会有一个代理违反他们的道路规则，这使得他们的行为从运动规划的角度无法预测。另一个例子是，当一个摩托车手执行车道分割 lane splitting，这可能是或可能不是合法的，取决于您的国家。这种行为可能会让自动驾驶汽车感到困惑，因为它经常使用车道边界来通知道路上其他代理的预测。

解决运动规划问题是一项复杂的任务。在没有任何形式的选择性抽象和简化的情况下，求解最优运动方案是很困难的。为了解决这个问题，我们将任务分解为一系列优化问题。通过这样做，我们可以将每个优化问题的输入和输出调整到正确的抽象级别，从而允许我们实时执行运动规划。在这个层次结构中，层次结构中的越高意味着优化问题处于更高的抽象层次。在这个层次结构的顶部是任务规划 mission planner，它集中于解决导航到我们的目的地在地图级别的自主驾驶任务。接下来是行为规划 behavioural planner。这决定了自动驾驶车辆应该采取的行为取决于其当前的驾驶场景。在此基础上，我们使用局部规划器 local planner 来计算到达所需目标状态的无碰撞路径和速度分布。在我们的例子中，我们将把这个过程解耦为路径规划 path planner 和速度剖面生成 velocity profile generator，以提高性能。最后，我们计算出的运动规划将提供给控制器。



Lesson 2: Motion Planning Constraints

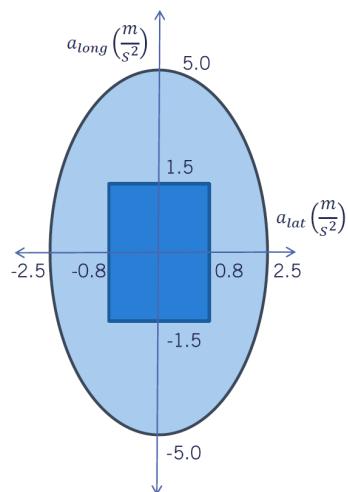
运动规划中涉及的一些最重要的约束。这些约束通常对于保持车辆内的稳定性和舒适性，以及在给定驾驶场景中保持所有代理的安全至关重要。

我们将处理的第一个运动规划约束与车辆运动学 kinematics 有关。在自动驾驶的运动规划中，车辆本身的运动学通常被简化为我们所知的运动学自行车模型。选择这种型号的一个原因是自行车有一系列可接受的转向角值，类似于汽车。对于固定速度 V ，转向角值 δ 的范围与车辆可遵循的容许曲率 κ 范围相对应。这意味着，在使用自行车模型执行运动规划时，在通过 κ_{\max} 表示的给定路径时，可以执行最大曲率幅度。对于自动驾驶汽车也是如此。这意味着我们规划的任何路径的曲率都需要注意这个最大曲率。不幸的是，这就是所谓的非完整约束 non-holonomic constraints。直观地说，这意味着约束不仅取决于机器人的状态，还取决于机器人如何达到其当前状态。非完整约束减少了机器人在其工作空间的任意给定点上可以采取的方向数量。一般来说，非完整约束使得规划问题更加复杂。

对曲率 κ 的直观解释是，它是圆的半径的倒数，沿着曲线拟合到那个特定的点。虽然这种圆拟合方法是有用的几何解释，一个更精确的数学公式给出了这里的曲率。它可以使给定路径的 x 和 y 分量相对于弧长的一阶和二阶导数来计算。

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

下一个将要讨论的约束来自车辆动力学 dynamics。这些动态约束的重点是保持汽车处于稳定的安全状态。第一个动力学约束是由汽车的摩擦椭圆 friction ellipse 施加的。如果您还记得的话，我们在课程一讨论了自行车模型中的轮胎打滑和摩擦椭圆。摩擦椭圆表示汽车轮胎和路面之间产生的摩擦力的最大值。如果汽车发动机的作用力超过轮胎的摩擦力，轮胎就会在路面上打滑。车辆的转向功能依赖于轮胎抓住路面。所以要保持操控性和稳定性，汽车必须保持在摩擦圈内。归根结底，这归结为横向和纵向加速度约束。



一般来说，尽管加速度限制通常比车内乘客的身体舒适度要高得多。因此，在非紧急情况下，我们通常关注加速度的舒适度。这些值限制了横向和纵向加速度在舒适范围内，表示为沿纵轴的纵向加速度和横向加速度。如图所示，深蓝色舒适矩形 comfort rectangle 位于浅蓝色摩擦椭圆内。这导致对运动计划的可行加速度的约束比摩擦椭圆要求的更严格。

从横向加速度约束，以及路径的曲率，我们现在间接地约束了汽车的速度。

$$a_{lat} = \frac{v^2}{r}, \quad a_{lat} \leq a_{lat_{max}} \rightarrow v^2 \leq \frac{a_{lat_{max}}}{\kappa}$$

我们可以看到速度的平方受到最大横向加速度($a_{lat_{max}}$ 是一个常数)的限制，还有路径的曲率 κ (它在路径的每一点上都是变化的)。

因此，很明显，当我们为我们的自动驾驶车辆生成速度剖面时，我们必须考虑路径的曲率以及车辆的最大横向加速度。

静态障碍也限制了我们的路径规划过程。由于静态障碍物（如停放的汽车或建筑塔架）具有不随时间变化的固定位置，因此通常通过遮挡车辆本身工作空间的某些部分来对其进行建模。

在这里我们将讨论占用率网格图。从本质上说，静态障碍物限制了汽车在其路径上可以占据的位置。有许多方法可以执行静态碰撞检查 static collision checking。

一种方法是在车辆沿给定路径行驶时检查车辆的线束 swath。当车辆本身穿过路径时，这个线束是车身占据的所有位置的联合体。如果线束与静态障碍物重叠，则该路径不可行。

快速碰撞检查的另一个选择是用一组圆来近似车身，并在车辆沿其路径移动时计算圆的位置。如果静态障碍物位于任一圆内，则认为该路径不可行。这些圆的并集通常大于车身，确保了保守近似。

另一方面，动态障碍物为运动规划问题提供了一些最具挑战性的约束条件。不同种类的动态障碍物，如汽车、自行车和行人，都会有不同的行为和运动模型。基于这些其他代理的行为来约束我们的运动计划通常会涉及预测，这是不确定的。然而，如果我们采取保守的方法，将自己限制在所有代理的所有可能行为上，我们的运动规划问题很快就会变得过度约束，无法有意义地解决。

我们在稳健性的安全性和前进所需的进取性之间取得平衡的程度，是自动驾驶研究的一个活跃领域。举个简单的例子，动态障碍物将约束我们的运动的情况是在交叉点处。如果两辆车相互正交进入交叉口，那么就有可能发生车祸。检查是否会发生碰撞的一种方法是跟踪车辆本身行驶方向形成的角度以及从车辆本身位置到另一个代理位置的矢量。如果这个角度是不变的，随着时间的推移，那么自我车辆将与其他代理碰撞，我们的运动规划将需要减速，以防止这一点。因此，动态障碍迫使我们根据障碍物在驾驶场景中的行进方式来修改我们的行为。

我们在上一课中讨论的另一个例子是，当一辆领先的汽车出现在车辆本身前面时。这辆领先的车对车辆本身的纵向速度施加了一个上限，因为如果我们在同一车道上行驶时超过它们的速度，我们最终会撞车。

动态障碍将约束我们的行为规划过程（我们在其中做出机动决策），以及我们的局部规划过程（它将影响我们的速度剖面规划）。

我们将讨论的最后一个约束包括道路规则以及车辆本身工作区中的监管元素。虽然道路规则为规划问题提供了一些约束条件，但它们也有助于我们对环境中其他主体的行为做出明智的决策。例如，迎面而来的车辆很有可能停留在车道上，而不是试图与我们的车辆本身正面相撞。这有助于在尝试预测其他代理将做什么时减少搜索空间。

道路规则施加的最常见约束之一是车道约束。简单地说，车道限制是为了防止我们的运动计划离开当

前车道，除非这样做是合法的。车道限制也会告知车辆本身在哪里可以安全地执行转弯操作。我们还需要遵守其他软性道路规则，例如在车道上保持车辆本身和领先车辆之间的时间间隔。时间间隔是指当以车辆本身当前速度行驶时，车辆本身达到领先车辆当前位置所需的时间量。保持一个相当大的时间间隔有助于确保运动规划过程中的安全，通过给车辆本身足够的反应时间来处理工作区中的事件。

工作区中的监管因素也会影响我们的驾驶行为。车辆本身显然需要尊重红灯和停车标志，以确保安全，但也需要了解不同区域的限速和其他动态监管要素，例如建筑工地提出的限速。一般来说，在执行运动规划时，监管因素对我们可用的行为有很大影响。

Lesson 3: Objective Functions for Autonomous Driving

运动规划问题的目标函数为我们提供了一种对当前运动计划进行评分的方法，并允许我们优化运动计划，使其具有理想的特性。

在执行路径规划时，最简单和最直观的目标是我们希望尽可能高效地到达目的地。在路径规划中，这通常转化为**最小化我们正在规划的路径的弧长**。直观地说，路径的弧长是汽车在通过路径时将行驶的总累积距离。

在许多情况下，路径将由弧长 arc length 参数化。可以表示为与弧长成比例的惩罚项 penalty term。在其他情况下，我们将不得不通过数值求解积分来计算路径的弧长。在弧长积分中，其中 s_f 是总累积弧长， x_i 是路径的起始 x 坐标， x_f 是路径的结束 x 坐标。通过最小化弧长，我们计算从当前位置到所需目的地的最短路径。

$$s_f = \int_{x_i}^{x_f} \sqrt{1 + \left(\frac{dy}{dx}\right)^2} dx \quad T_f = \int_0^{s_f} \frac{1}{v(s)} ds$$

在优化速度剖面时，**到达目的地的时间**通常是我们希望最小化的目标。直观地说，这相当于将总距离除以沿曲线在每个点上行驶的速度。其中 T_f 是剖面的总时间。通过最小化这个积分，我们在遵循给定的计划路径的同时最小化了到达目的地的时间。

现在，在自动驾驶的路径规划中，我们可能会得到一个我们想要遵循的参考路径。在许多情况下，这将由我们的地图模块提供，作为长度的中心线，或交叉口转弯所需的路径。为了确保我们尽可能接近**参考路径**，我们加入了差分项积分 integral of difference term (IOD)。本质上，它会惩罚与输入参考路径 x_{ref} 的偏差。类似地，在执行速度剖面优化时，我们通常希望将速度剖面之间的差异最小化，即在整个路径上的速度与某个参考速度 V_{ref} 之间，无论是速度限制还是以其他方式确定。

$$\int_0^{s_f} \|x(s) - x_{ref}(s)\| ds \quad \int_0^{s_f} \|v(s) - v_{ref}(s)\| ds \quad \int_0^{s_f} (v(s) - v_{ref}(s))_+ ds$$

虽然这可以表述为绝对差额处罚，但通常情况下，我们希望对超过限速的处罚比保持在限速以下的处罚更严厉。所以我们可以把所谓的铰链损失项 hinge loss 加到第三个等式中的目标上。本质上，**这个项只有在速度剖面超过速度极限时才有效**，即当我们的速度和速度极限之间的差值为正，否则这个项为零。这使得我们能够更严厉地处罚违反限速的行为，同时仍然允许我们鼓励自动驾驶车辆达到其要求的参考速度。

我们要解决的下一个目标是**乘坐平稳性**。回想上一课，为了保持稳定性和舒适性，我们限制了最大加速度。当在我们的速度剖面的目标函数中优化舒适度时，我们将注意力转向最小化沿着我们的轨迹的颠簸。

Jerk 是加速度相对于时间的变化率，或位置的三阶导数。车辆在行驶过程中的颠簸会极大地影响使用者在车内的舒适性。所以在规划我们的速度剖面时，我们希望把 jerk 的累积绝对值保持得尽可能小，这里用这个积分表示。

$$\int_0^{s_f} \|\ddot{x}(s)\|^2 ds$$

由于舒适度的差异，在可能的情况下，选择能够减少乘客感觉到的颠簸的外形是很重要的。

在规划路径时，我们必须注意**路径的结构也会影响速度曲线的舒适性**。回想上一课，高曲率路径将沿路径的最大速度限制为较低的值，以便保持在车辆的摩擦椭圆内。为了确保我们避免沿着路径的高曲率点，我们需要为大的绝对曲率值制定一些惩罚，通常用于表示这种惩罚的目标函数称为路径的弯曲能量 bending energy。

$$\int_0^{s_f} \|\kappa(s)\|^2 ds$$

基本上，它是沿着路径积分的曲率的平方。此目标使曲率沿路径更均匀地分布，从而防止沿路径的任何一个曲率值达到过高的总曲率值。

总体曲率分布更广，这将导致一个更平稳更舒适的条件，并将允许我们保持一个合理的速度，同时当优化我们的速度剖面时仍然保持在舒适矩形中。

运动规划的各种目标函数: 与效率相关的目标函数，并讨论了它们如何减少路径长度和运输时间。如何鼓励轨迹跟踪参考路径和速度剖面，以及如何通过惩罚高路径曲率和 jerk 来改善给定轨迹的舒适性。

当设计一个总体目标函数时，这些项中的每一项都需要组合成一个函数，通常作为一个加权和。

Lesson 4: Hierarchical Motion Planning

我们将层次规划器 hierarchical planner 的每一层描述为定义自己的优化问题，层次的每一层对于相关的目标函数和约束具有特定的抽象程度。

任务规划 mission planning 是一个最高级的优化问题，我们关注的是从车辆当前位置到给定目的地的地图级导航。另一方面，行为规划 behavioural planning 将侧重于当前的驾驶场景，并将根据工作区中的其他代理和道路规则来决定要执行的操作。最后，局部规划 local planning，它将着重于以解耦 decouple 的方式生成运动上可行的、无碰撞的路径 path 以及舒适的速度剖面 velocity profile。这将输出到车辆控制器。

通过将规划问题分解为子问题，我们限制了运动规划过程中每个部分所需的知识和计算量。这样可以进行更有效的计算。然而，这需要权衡，因为当我们分解问题时，每个子问题都会丢失一些域信息。与同时考虑规划过程所有方面的运动规划相比，我们最终得到一个次优的 **sub-optimal** 运动规划。对

于自动驾驶，实时运行我们的运动规划器的能力是至关重要的，因此这种权衡几乎总是值得的。

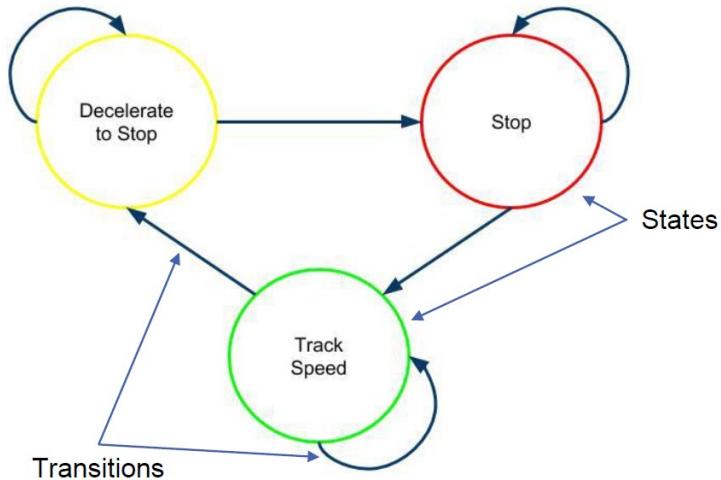
任务规划器的重点是从当前位置到所需目的地的地图级导航 map-level navigation。因为任务规划的规模通常是以公里为单位的，所以我们需要一个高层次的抽象以使问题易于处理。我们往往忽略了运动规划问题的各个方面，如障碍物和监管因素，而把重点放在宏观方面的问题，如交通和道路连接。这样，我们可以将问题简化到可以在大空间范围内规划出所需的驾驶任务。

这可以通过多种方式实现。如果我们关注道路网中道路的空间长度，我们可以从道路网中构造一个图 graph-based methods，然后使用 Dijkstra 算法或 A*搜索来寻找最短路径。如果把到目的地的时间作为最小化的目标，问题会变得更复杂，但仍然可以用相同的算法来解决。

我们的分层规划器的下一步是行为规划器。行为规划器是运动规划器的一部分，它专注于遵循道路规则所需的高层次决策，并识别在给定驾驶场景中哪些策略是安全的。例如，回想一下我们讨论的场景之一是在十字路口转弯。举个例子来说，ego 车辆进入一个十字路口左转，灯变绿。行为规划器应向路径规划器和速度发生器说明，只有当车辆位置与交叉口迎面而来的车辆和行人之间有足够的间隔时，才能安全行驶。按照我们在第一课中讨论的策略分类法，行为规划器应该向运动规划器的其余部分输出一个屈服 yield 策略，这将确保它们只有在被清除时才能继续。当然，我们的分类法并不是通用的，当决定你的自动驾驶汽车应该表现出什么样的操控类型和行为范围时，有很多可能性。

现在，你可能想知道，行为规划器应该如何吸收有关自动驾驶车辆周围环境的所有信息，并计算出它应该以有效的方式执行哪些策略。这是一个活跃的研究领域，没有固定的最佳方法来实现这一点。一般来说，有三种不同的主流架构来解决行为规划问题。

第一组是使用有限状态机 finite state machine (FSM) 的方法。



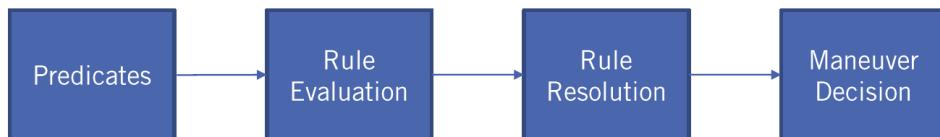
有限状态机背后的关键概念是，它们由以下部分组成：表示所需策略的**状态 states**(基于周围环境的感知)，以及指示状态应如何根据输入进行演化的**转换 transitions**(基于驾驶场景的输入)。这些可能是这样的：车辆位置，红绿灯转换，或在我们目前的驾驶情况下的任何其他因素的利益。

例如，可以想象一个处理停车标志的状态机有一种状态，比如减速停车、停止和跟随速度。当一辆车遇到停车标志时，它会首先进入减速到停车状态，然后行为规划器会发出信号，表示该减速了。一旦在停止位置停止，行为规划器将在设置的时间内保持停止状态。一旦超过时间且交叉口畅通，行为规划器将进入跟随速度状态，这意味着继续行驶是安全的。

需要注意的是，在决定在有限状态机的每一步执行哪个转换时，我们完全关注当前状态和有限状态机

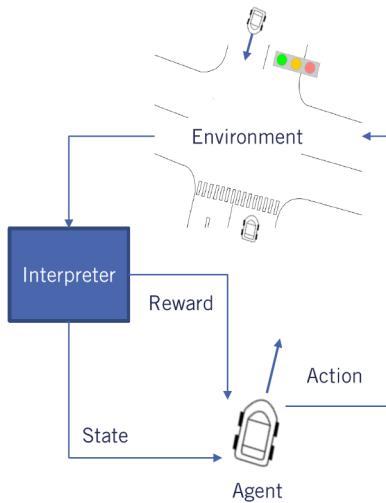
的输入。本质上，这意味着有限状态机是无记忆的 **memoryless**: 它过去的状态不会影响当前的转换或输出。这是一个有用的属性，它使有限状态机易于实现，但对于操作序列非常重要的系统，它会导致许多类似的状态。

下一种常见的行为规划器是基于规则的系统 **rule-based system**。本质上，这些类型的系统由规则层次 **hierarchy of rules** 结构组成以决定输出的行为，其中层次结构表示每个规则的相对重要性。每个规则可能对应于**道路规则**，例如红灯停车，或者可能对应于**驾驶最佳实践**，例如在 ego 车辆和领先车辆之间保持两秒钟的间隙。ego 车辆及其周围环境的状态将以**逻辑谓词 logical predicates** 的形式输入到基于规则的系统中。然后根据这些谓词评估每个规则，并将较高优先级的规则相对应的动作输出到运动规划器的其余部分。



例如，假设我们有以下两条规则。第一条规则是检查前方是否有绿灯，并决定车辆继续行驶。而第二条规则检查车道上是否有行人，并发布执行紧急停车操作的决定。在我们的系统中，为前面的行人停车比闯绿灯优先。现在假设我们开车行驶，在即将到来的十字路口看到一个绿灯，同时在十字路口前面看到一个非法的行人过街。在这种情况下，这两个规则都会触发，但由于紧急停止具有更高的优先级，因此基于规则的系统将输出机动策略。可以想象，确保基于规则的系统在逻辑上是一致的既重要又具有挑战性。否则，它们会表现出不稳定的行为。

最后一组行为规划方法是基于机器学习的方法。特别是，这种方法的一个有趣的例子是使用强化学习 **reinforcement learning**。强化学习是一个确定最优决策策略的过程，该策略使某个奖励函数 R 最大化。该奖励函数在所有时间步长上都对给定的行为链的质量进行了评估，而对未来状态的折扣比现在更大。强化学习过程要求 **agent** 在通常由仿真给出的环境中执行动作。然后，该代理将根据其与环境的交互而得到奖励。这使得它能够通过连续的交互作用收敛到一个最优策略。这组行为规划方法正在迅速扩展研究领域，不幸的是，深入研究超出了本课程的范围。

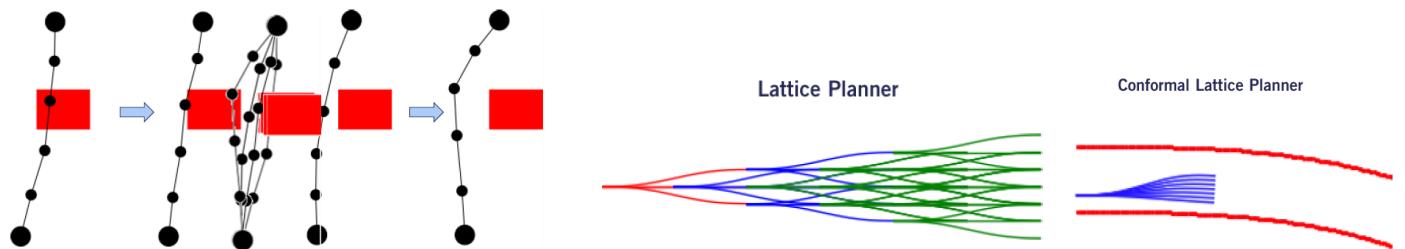


我们的分层规划器的下一个也是最后一个模块是局部规划器。如前所述，本地规划者的目标是为 ego 车辆生成运动上可行且无碰撞的**路径**以及舒适且安全的**速度剖面**。我们将局部规划问题分解为两个子问题：路径规划 **path planning** 和速度剖面生成 **velocity profile generation**。

开发一个好的路径规划算法的关键在于减少优化的搜索空间。主要有三类路径规划器：基于采样的规划器 sampling-based planner、变分规划器 variational planner 和格点规划器 lattice planner。

基于抽样的规划器对车辆的控制输入进行均匀随机抽样，以生成车辆的潜在路径。基于抽样的最具代表性的算法之一是快速探索的随机树（RRT）及其变体。这些算法通过在随机抽样的位置生成点并规划一条从树中最近点到该点的路径来构造路径树的分支。如果路径没有与任何静态障碍物碰撞，则该路径将添加到树中。此树使用许多潜在路径快速探索工作区，当到达目标区域时，将返回终止于该区域的路径。基于采样的算法通常速度极快，但在短周期内运行时，可能会产生质量较差的不稳定路径。

变分规划器依赖于代价泛函 **cost functional** 来优化轨迹函数，该函数根据考虑障碍物和机器人动力学的代价函数，将时间点映射到工作空间中的位置。变分规划器通常是轨迹规划器，这意味着它们将路径规划和速度规划结合到一个步骤中。因此，他们不必解耦的路径规划和速度规划问题。作为一个例子，假设我们从空间中一条路径的给定离散采样开始。一开始，它和一个障碍物也发生了碰撞。一个变分规划器将迭代地移动离散点，使得它们相对于机器人的动力学是平滑的，并且保持它们无碰撞。最后，轨迹将是无碰撞的，并且相对于机器人的动力学更加平滑。然而，变分方法往往速度慢，复杂度高，收敛到可行路径对初始条件敏感。变分法的一个例子是 CHOMP 算法。变分规划器超出了本课程的范围。



最后一组路径规划器是晶格规划器。晶格规划器通过限制 ego 车辆在工作空间中任何一点可以采取的行动来约束搜索空间。这组操作称为晶格规划器的控制集。此控件集与工作区的离散化配对时，隐式地定义了一个图。然后可以使用 Dijkstra 或 A*之类的图搜索算法来搜索该图，从而快速计算路径。障碍物会给跨越它们的人带来无限的代价。因此，图形搜索也允许我们执行碰撞检查。虽然晶格规划器通常非常快，但路径的质量对所选的控制集非常敏感。晶格规划器上的一个常见变体称为共形格规划器 conformal lattice planner。在车辆前方一定距离处选择一个目标点，相对于道路方向彼此横向偏移，并对每个目标点的路径进行优化。然后选择最能满足某个目标同时保持无碰撞的路径作为执行路径。

- P. Polack, F. Altche, B. Dandrea-Novel, and A. D. L. Fortelle, “[The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles](#)” 2017 IEEE Intelligent Vehicles Symposium (IV), 2017. Gives an overview of the kinematic bicycle model.
- S. Karaman and E. Frazzoli, “[Sampling-based optimal motion planning for non-holonomic dynamical systems](#),” 2013 IEEE International Conference on Robotics and Automation, 2013. Introduces the RRT* algorithm as an example of sampling-based planning.

- N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “[CHOMP: Gradient optimization techniques for efficient motion planning](#),” 2009 IEEE International Conference on Robotics and Automation, 2009. Introduces the CHOMP algorithm as an example of applying calculus of variations to planning.
- M. Pivtoraiko, R. A. Knepper, and A. Kelly, “[Differentially constrained mobile robot motion planning in state lattices](#),” Journal of Field Robotics, vol. 26, no. 3, pp. 308-333, 2009. Introduces the state lattice planning method.

Lesson 1: Occupancy Grids

我们将讨论两个环境地图的创建：占用栅格地图 occupancy grid map 和高清路线地图 HD road map。

占用网格是包围当前车辆位置的离散网格。这种离散化 discretized 可以在二维或三维中进行。我们讨论的方法可以应用于二维和三维问题。然而，为了简化本模块中的解释和计算要求，我们将只关注 2D 版本。占用栅格的每个栅格方格表示该栅格位置中是否存在**静态或静止对象**。如果是，则该栅格位置被分类为已占用。分类为占用栅格单元的静态对象的示例可以包括树、建筑物、路标和灯杆。在自动驾驶车辆领域，您可能不认为是障碍物的其他物体也应归类为占用空间，包括所有不可行驶的表面，如草坪或人行道。由 m^i 表示的占用网格的每个方块都映射到一个二进制值，其中 1 表示该方块被静态对象占用 occupied，0 表示它没有被静态对象占用 free。网格中所有被占用的正方形都是紫色的，与可驾驶曲面相对应的地图的其余部分保持透明。

我们现在来看看为创建准确的占用网格所做的一组假设。首先，**当前为创建此占用网格而测量的环境仅与静态对象相对应**。也就是说，在将传感器数据用于占用栅格映射之前，必须从传感器数据中删除所有动态对象或移动对象。第二，**每个网格单元都是独立的**。此假设是为了简化创建占用网格所需的更新函数。最后，**当前车辆状态在每一个时间步都与占用栅格地图相关**。

在自动驾驶汽车领域中，观察汽车与世界的当前状态的距离，LIDAR 是最常用的传感器。LIDAR 使用光脉冲来测量到汽车周围所有物体的距离，并返回整个视野的测量点云。LIDAR 的几个组成部分需要过滤掉，然后这些数据才能用于构建占用网格。第一步是过滤构成**地平面 ground plane** 的所有激光雷达点。在这种情况下，地平面是自动驾驶汽车可以安全行驶的路面。接下来，所有出现在**车辆最高点上方的点 objects above car height** 也被过滤掉。这组激光雷达点可以忽略，因为它们不会阻碍自动驾驶车辆的前进。最后，需要移除激光雷达捕获的所有**非静态物体 dynamic objects**。这包括所有车辆、行人、自行车和动物。

一旦所有的激光雷达数据过滤完成，三维激光雷达数据将需要**投影**到一个二维平面，用于构建我们的占用网格。激光雷达数据的过滤和压缩，为自动驾驶汽车创建精确的占用网格。

经过过滤和压缩的激光雷达数据类似于高清二维距离传感器的数据，该传感器可以精确测量在平面上车辆周围所有静止物体的距离。然而在完成所有滤波之后，由于**对数据使用的滤波方法、手头数据的复杂性，以及最主要的环境和传感器噪声**，仍然存在主要的地图不确定性。

为了处理这种噪声，占用网格将被修改为概率的 probabilistic。现在每个单元格 i 将存储一个介于 0 和 1 之间的概率，对应于给定的方块被占用的确定性，而不是存储一个二进制值。存储的值越高，给定的方块被占用的概率就越高。占用网格现在可以表示为由术语 bel 表示的**信念图 belief map**。为了保

持符号简单， $m^i \in \{0, 1\}$ 表示占用网格的一个正方形，其中 i 可以由测量值 Y 和车辆位置 X 构成。 m^i 上的置信度等于当前单元格 m^i 被占用的概率，给定为该单元格位置收集的传感器测量值。为了转换回一个二进制映射，可以建立一个阈值，在该阈值处给定的信念有足够的信心被归类为已占用。任何低于设置阈值的值都将被设置为自由。

$$bel_t(m^i) = p(m^i | y, x)$$

Current map cell Sensor measurement
for given cell

多组测量数据可以从一个时间点到另一个时间点进行组合，以获得更准确的占用率。事实上，我们可以递归地更新信念，这样在每一个时间步 t ，我们使用从时间 1 开始的所有先验信息来定义我们的信念。地图单元 m^i 上时间 t 的置信度定义为给定时间 1 到 t 的所有测量值和车辆位置， m^i 被占用的概率。

$$bel_t(m^i) = p(m^i | (y, x)_{1:t})$$

Normalizer constant

$$bel_t(m^i) = np(y_t | m^i) bel_{t-1}(m^i)$$

↓
Current measurement Previous belief map

为了将多个测量值组合成一个置信度地图，可以应用 **Bayes 定理**。在占用网格的例子中，我们得到一个贝叶斯更新步骤，其形式如下。给定 m^i 的 y_t 的分布 p 是给定一个单元格 m^i 被占用时获得特定测量值的概率。这被称为测量模型。时间 $t-1$ 除以 m^i 的置信度对应于从上一时间步存储在占用网格中的先前置信度。我们依赖于马尔可夫假设，即在每个时间步，在信念图中捕获估计单元格占有率所需的所有信息。所以在单元格更新方程中不需要考虑早期的历史。最后，在这种情况下， η 对应于应用于信念映射的规范化常数 normalizer constant。这需要成规模的 results 以确保它仍然是一个概率分布。

若一个物体被分类为障碍物所需的置信阈值被设置为非常高，因此只有大的静态物体被识别为被占用。降低该阈值将导致更多的单元被标记为已占用，但也将导致噪声更大的地图。

Lesson 2: Populating Occupancy Grids from LIDAR Scan Data (Part 1)

正如我们在前面的视频中所看到的，我们可以应用 Bayes 定理将之前的信念图与当前的测量信息结合起来，在每个时间步创建一个高度精确的占用率网格。

然而，使用这个简单的贝叶斯更新有一个问题。在计算机上进行浮点数乘法时，小数字相乘会导致显著的舍入误差，这反过来又会导致概率估计的不稳定性。此外，概率的乘法被证明是执行信念更新的低效方法。

然而，有一个解决办法。我们可以使用 logit 函数将我们的信念转换为对数概率，而不是用 0-1 的值存储信念图。**logit 函数设为 $logit(p) = \log [p / (1-p)]$** ，因此它取 0-1 的概率值，并将它们映射到整个实轴。也可以从对数几率域转换回概率域。通过 $p = e^{logit(p)} / [1 + e^{logit(p)}]$ 。

我们现在有了一个单元格概率的替代表示。让我们看看这是如何影响我们的贝叶斯更新方程的。

应注意，测量 y_t 与 $y_{1:t-1}$ 的其余测量分开。这是因为我们只想用最新的传感器测量值来更新占用率网格，而不是存储所有测量值并每次应用它们。

- Applying Bayes' rule:

$$p(m^i|y_{1:t}) = \frac{p(y_t|y_{1:t-1}, m^i)p(m^i|y_{1:t-1})}{p(y_t|y_{1:t-1})}$$

Current map cell Sensor measurement for given cell Pulling out current measurement y_t from past measurements $y_{1:t-1}$

Pulling out current measurement y_t from past measurements $y_{1:t-1}$

(推导...)

$$\begin{array}{c} \text{Inverse Measurement Model} \quad \text{Previous belief} \quad \text{Initial belief} \\ \swarrow \qquad \searrow \qquad \nearrow \\ l_{t,i} = \text{logit}\left(p(m^i|y_t)\right) + l_{t-1,i} - l_{0,i} \end{array}$$

这是我们的最终更新方程，它具有一个很好的特性，即当需要一个新的度量值时，只需要加上去。我们现在得到了方便的对数几率更新规则 log odds update rule，用于在占用网格图上进行贝叶斯推断。它由基于最新测量数据在每个时间步组合的三个项组成。

逆测量值模型 inverse measurement model，给定测量值 y_t 的占据栅格状态 m^i 概率的第一项 logit 是使用新的测量信息形成的 logit。但目前我们只能有 $p(y_t|m^i)$

$l_{t-1,i}$ 是单元格 i 在时间 $t-1$ 被占据的信念 logit 函数。

$l_{0,i}$ 是时间 0 时对同一单元格的初始信念 logit 函数。

初始信念表示网格单元被占用的基准 baseline 信念，该信念通常统一设置为 50%，因为我们不希望有改进该值的先验信息。它出现在这个方程的每一个时间步。

与直接更新概率相比，贝叶斯对数概率有两个强大的优势。由于 0-1 概率到整个实轴的 logit 映射，更新在数值上是稳定的，并且在计算上，它也显著地更有效，因为它完全依赖于加法来完成占用网格的所有更新。

Lesson 2: Populating Occupancy Grids from LIDAR Scan Data (Part 2)

tips: probability 和 odds 的区别

probability 是你期望在许多试验中看到该事件的次数。probability 总是在 0 和 1 之间。

odds 定义为事件发生的概率除以事件不发生的概率，即 $p/(1-p)$

回想一下之前的视频，为了对我们的占用率网格执行贝叶斯更新，我们需要能够评估对数几率更新规则。为了做到这一点，我们需要计算给定 y_t 的 m^i 的 p ， p 表示占用网格中的单元格 m^i 被占用的概率， y_t 表示给定激光雷达获得的测量值。

然而，到目前为止，我们所看到的测量模型的形式是给定 m^i 的 y 的概率 $p(y|m^i)$ ，在映射的情况下，它表示在占用网格中的一个单元被占用的情况下，获得某个激光雷达测量的概率。因此，对于占用率网格更新，我们需要翻转这个测量模型。也就是说，我们必须构造一个逆测量模型，该模型为每一个新的测量值构造 $p(m^i|y_t)$ ，给定一个测量值，观察单元格的状态。

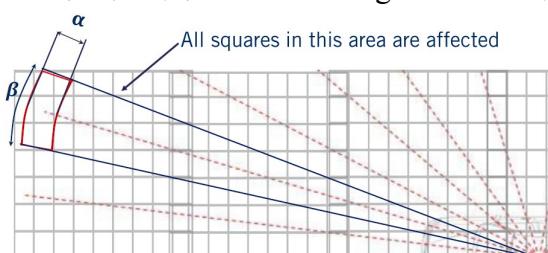
为简单起见，我们将仅将激光雷达数据视为包含两组信息的二维范围数据。每束光束发射的方位 scanner bearing $\phi^s = [-\phi_{\max}^s, \dots, \phi_{\max}^s]$, $\phi_j^s \in \phi^s$ 和 射程 scanner ranges $r^s = [r_1^s, \dots, r_j^s]$, $r_j^s \in [0, r_{\max}^s]$ 。通过添加仰角，可以很容易地将其扩展到三维。每个光束的方位由一组角度 ϕ 表示，这些角度 ϕ 均匀分布在由传感器的最小和最大视角定义的范围内。典型的旋转激光雷达能够捕获车辆周围所有方向的数据，我们假设本课程中可用的方位将覆盖所有 360 度。每个光束的射程对应于它在撞击 r_1 到 r_j 表示的物体之前所走过的距离。范围是在最小和最大范围 r_{\min} 和 r_{\max} 之间测量的，为了简单起见，我们假设 r_{\min} 为零。今天的大多数激光雷达也会返回无回波信号，如果某一特定光束不返回回波，则表明该方向范围内没有任何物体。当我们在这段视频中继续时，我们将在这里做出一个假设，即从传感器的单个旋转开始的整个激光雷达扫描测量是在同一时刻测量的。这不是一个精确的模型，因为我们必须实际校正车辆的运动。

因此，逆测量模型将采用以下形式。假设一辆装有激光雷达传感器的车辆，在下列环境中采集不同方位的距离测量值。我们构建了一个临时占用网格，该网格包含了所有方向上的电波 beams 的最大范围。该测量网格的坐标框架使用占用网格地图框架，因此我们在占用网格框架中定义传感器的位置 $x_{1,t}$ 和 $x_{2,t}$ 以及方向 $x_{3,t}$ 。在实践中，这个占用网格框架被设置为车辆框架，地图在每一步根据我们的状态估计进行转换。我们现在准备用占用概率填充临时测量网格。基于当前的激光雷达数据，我们可以定义测量网格的三个区域。有一个没有任何激光雷达光束能够到达的信息区域 no information。目前的激光雷达扫描没有提供有关该地区环境的新信息。一个物体概率很低的区域 low probability，因为所有的光束都穿过这个区域而没有遇到任何东西。最后，有一个目标概率很高的区域 high probability，在该区域中，激光雷达已与目标接触并返回非最大距离值。

为了将这些区域转换到测量网格上，每个网格正方形将被指定一个相对于车辆当前位置的方位范围。每个单元的相对范围只是传感器到单元的欧氏距离，定义如下，其中 r_i 是到网格单元 i 的范围， m_{xi} 和 m_{yi} 是网格单元中心的 x 和 y 坐标。最后， $x_{1,t}$ 和 $x_{2,t}$ 是当前时间 t 的传感器位置。使用 \tan^{-1} 反函数计算每个单元的相对方位。在这里， ϕ 表示相对于传感器坐标系的给定单元的方位。对于每个单元，我们将最相关的激光雷达光束关联起来，方法是找到光束角度 beam angle 和单元方位 cell bearing 之间误差最小的测量值。

$$\text{相对范围 } r^i = \sqrt{(m_x^i - x_{1,t})^2 + (m_y^i - x_{2,t})^2} \quad \text{相对方位 } \phi^i = \tan^{-1}\left(\frac{m_y^i - x_{2,t}}{m_x^i - x_{1,t}}\right) - x_{3,t} \quad \text{最接近的相对方位 } k = \operatorname{argmin}(|\phi^i - \phi_j^s|)$$

然后我们定义两个参数 Alpha 和 Beta，这两个参数定义了每个波束周围的扇区，在该扇区中，应根据波束范围确定单元格占用率。这实质上是在每个光束周围创建一个区域，该区域将被指定该特定光束的测量信息。Alpha 控制波束范围内的范围单元 affected range，该范围单元将被标记为高概率。Beta 控制光束的角度 affected angle，对于该角度，单元格将被标记为低概率或高概率。



我们现在准备分配任何单元格被占用的概率，给定基于这三种类型的单元格接收到的激光雷达测量值。

无信息区对应于范围大于测量范围的所有单元格，或与之相关测量的 β 角大小的圆锥体之外的所有单元格。我们指定一个障碍物的概率等于一个单元被占用的先验概率，通常为 0.5。

高概率区定义了在两个距离测量值的 α 范围内和在两个与之相关角度测量值的 β 范围内的单元格。我们分配了大于 0.5 的占用概率。

低概率区是由范围小于测量范围 $-\alpha/2$ 的单元格定义的，并且位于关于测量的 β 大小的圆锥体内。我们给这些单元分配了小于 0.5 的占用概率。

例如，假设一个激光雷达扫描返回一个范围到一个物体在一个红色的 X 标记的位置。受影响的区域将被标记为高概率是红色的。增大 Alpha 值将增大受影响区域的范围，增大 Beta 值将影响受影响区域的角度。

- No Information

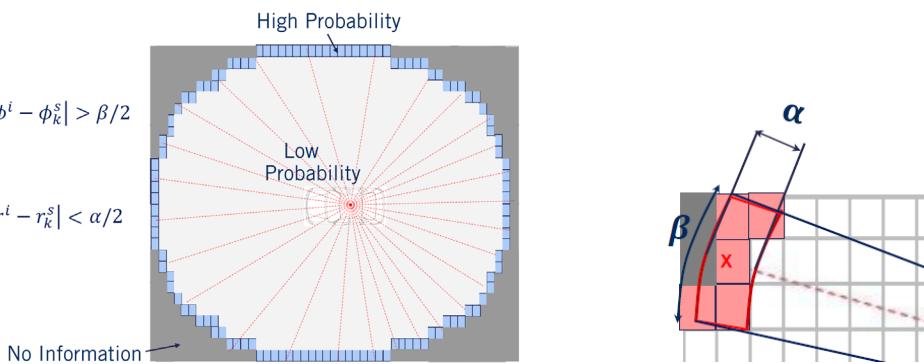
if $r^i > \min(r_{max}^s)$ or $|\phi^i - \phi_k^s| > \beta/2$

- High probability

if $r_k^s < r_{max}^s$ and $|r^i - r_k^s| < \alpha/2$

- Low probability

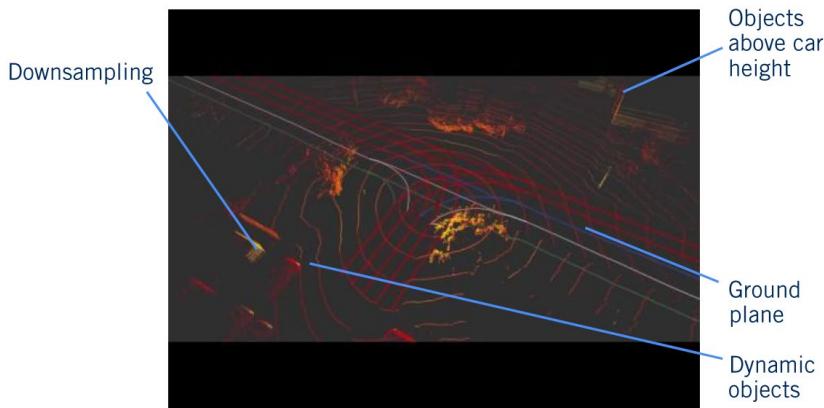
if $r^i < r_k^s$



现在，我们可以构建完整的反向测量模型，用于我们的日志更新。然而，这种简单的逆测量模型在计算上很昂贵。它需要对度量映射中的每个单元格进行完全更新，并依赖于多个浮点操作来确定哪些度量对应于哪些单元格。另一种方法是使用光线跟踪算法 ray tracing，如 Bresenham's line algorithm。最初设计于 20 世纪 60 年代初，旨在有效地解决当前可用硬件上显示和打印的线条绘制问题。通过沿激光雷达扫描光束的光线跟踪，我们减少了需要处理的单元数量，并通过整数加减和位移沿激光雷达光束在网格中移动来更快地识别它们。有趣的是，许多光束通过汽车附近的相同单元，大大增加了附近测量的可信度。

Lesson 3: Occupancy Grid Updates for Autonomous Driving

在这里，我们有一个典型的激光雷达扫描获得的自动驾驶车辆，因为它沿着当地道路行驶。我们将重点介绍几个必须应用于激光雷达扫描的过滤器，然后才能用于填充占用网格。首先，为了使更新操作实时运行，通常需要将激光雷达扫描的点数减少到较小的数量。其次，为了去除不影响驾驶的物体，我们过滤掉了所有位于自动驾驶汽车上方的激光雷达点。第三，我们不想给占用的可驾驶表面贴上标签。因此，我们删除所有的激光雷达点击中可驾驶表面或地面。最后，我们移除所有动态或移动的物体，例如正在运动的汽车或行人，再次依赖感知堆栈来识别它们的位置。



下采样 **downsampling** 是通过删除或忽略多余的 LIDAR 点来减少要考虑的激光雷达点的数量的过程。用于自动驾驶的普通激光雷达每秒可以产生 120 万个点，为车辆周围的所有物体提供非常丰富的几何描述。但是由于有许多点围绕它们捕获相同的对象信息，许多生成的点实际上是冗余的。在本例中，我们看到一个道路标志密集地覆盖在激光雷达点上，其中一小部分点足以指示障碍物的位置，以便进行占用栅格地图绘制。最重要的是，处理 120 万个点在计算上是不切实际的。第二，一些点必须被删除，以改善未来所有操作的计算。

下采样可以以多种方式执行。其中最简单的方法是使用一个系统滤波器 systematic filter，使激光雷达扫描环上的每 n 个点都保持不变。也可以在距离图像中应用图像下采样技术，并在 3D 网格中空间搜索，用单个占用方法 single occupancy measurement 代替点集合。在每种情况下，这些方法都可以在开源点云库（如 PCL）或计算机视觉库（如 OpenCV）中使用。

我们的第二个过滤器是一个很小的过滤器，它只是删除车辆高度(如 2.4m)以上的点。然而，这个过滤器通常假定一个平坦的接地层，因此您应该意识到盲目地应用这个假设是危险的。消除悬垂的树木、电线、桥梁和标志的价值是重要的，因此值得在本讨论中包括。

下一个过滤器包括移除掉在可驾驶表面上的点。由于激光雷达扫描的性质，在这张图片中看到的许多同心圆是由于激光雷达扫描击中了可驾驶表面。这些点不应与占用网格图中的占用单元格混淆。然而，这被证明是一项具有挑战性的任务，因为出现了一些复杂情况。

首先，所有道路都有不同的道路几何结构，包括排水的可变凹度、不同的坡度和坡度、曲率等。这种变化也很难预测。但是，如果不移除地平面，占用网格可能会出现伪影 artifacts，从而导致死锁 deadlock，汽车无法继续行驶，因为它认为道路被阻塞。

第二个问题是，路缘在不同的位置有不同的高度，并且道路边界在激光雷达数据中并不总是明确定义的。这些变化可能导致部分路缘或不可行驶区域作为地平面的一部分被移除。

最后，需要检测道路上的小物体，如足球或海龟，这意味着在点云上使用简单的几何方法很难解析前方道路的真实状态。处理这类问题的最佳方法是利用视觉和通过语义分割的深层神经网络。这可以在分割图像中看到，其中地平面显示为深紫色。然后，该任务就变成了将视觉数据中检测到的可驾驶表面映射到 Lidar 点云，从而掩盖掉位于可驾驶表面投影边界内的所有点。

所需的最后一个过滤器是移除所有动态对象，例如移动的汽车或行人。这可以再次通过依赖于感知堆栈来完成，感知堆栈必须检测和跟踪场景中的所有动态对象。检测到的动态对象的三维边界框用于删除受影响区域中的所有点。在边界框的大小上增加了一个小的阈值，用于解释感知算法中的任何小错误，提高了点移除滤波器的鲁棒性。然而，很多时候，这是不令人满意的两个原因。首先，并非所有检测到的动态对象类实例都在实际移动。一些车辆可能停在路边，因此可以被视为占用网格的一部分，因为它们确实是静态的。为了处理这个问题，感知器需要使用动态对象轨迹来识别那些当前是静态的

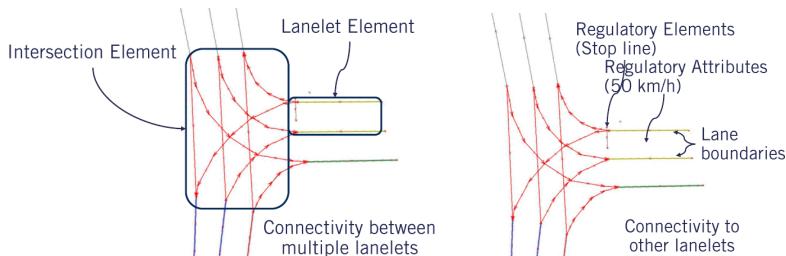
对象，这样占用映射器就可以避免删除它们。第二，由于感知堆栈通常需要计算时间，因此动态目标的检测需要经过一定的延迟。这导致使用过期对象位置更新占用栅格，从而导致边界框丢失动态车辆上的大部分激光雷达点。相反，我们可以根据运动物体的轨迹来预测它们的运动。边界框沿预测路径向前移动，导致从最近的激光雷达扫描中移除更多动态对象的点。

在所有这些过滤之后，激光雷达数据终于可以从 3D 投影到 2D 了。现在让我们来看看这最后一步的简单而有效的技术。首先，将存储激光雷达点高度的 Z 值设置为零，从而将激光雷达栅格折叠到一个平面上。2D 扫描平面被划分为与占用网格相同的网格图案。对于占用网格的每个单元，都会计算其内的所有激光雷达点。然后，该值将用作占用率的度量。单元中的点越多，占用栅格的该单元中存在静态对象测量的可能性就越大。

Lesson 4: High Definition Road Maps

高清路线图类似于传统的纸质地图或在线地图。但是，高清地图或高清地图中包含的信息要详细得多。传统地图存储的是道路的大致位置，而高清地图存储的是精确的道路位置，包括所有车道，精确到几厘米。同时，高清道路地图存储了可能影响自动驾驶车辆的所有道路标志和信号的位置。由于数据的详细性和相互关联性，需要一种有效的方法来存储地图中包含的所有信息。"Lanelets:自动驾驶的高效地图表示"中提出了 lanelet 概念。由于该方法能有效地存储和传输高清地图所需的复杂信息集，因此得到了广泛的应用。特别是在我们的行为规划方法中，它们将发挥关键作用。

lanelet 地图有两个主要组件。一种车道要素 lanelet element，它存储与它所代表的道路上一条车道的一个小的纵向段相连的所有信息。一种交集元素 intersection element，用于存储作为单个交集一部分的所有 lanelet 元素，以便在运动规划任务期间进行简单检索。



接下来将解释所有不同 lanelet 元素之间的连接。lanelet 元素具有多组信息，这些信息存储了它所表示的路段的相关信息。首先，lanelet 存储给定车道的左右边界。其次，lanelet 存储任何可能出现在 lanelet 元素末尾的监管元素，例如停止标志线或静态标志线。请注意，我们只为任何监管元素存储一条线，因为这是自主车辆视为该监管元素的活动位置的点。第三，lanelet 存储所有可能影响该路段的监管属性，例如速度限制。第四，lanelet 存储自身与周围其他 lanelet 元素的连接。这使得通过 HD 地图中的 lanelets 集创建的图进行简单的遍历和计算成为可能。

当新的监管元素出现或终止时，创建一个新的 lanelet.

每个 lanelet 以一个监管元素结束，或者遇到调控属性的变化。这意味着，对于作为交叉口一部分的车道，车道元素可以短到只有几米，或者对于高速公路路段，车道元素可以长到几百米。lanelet 的边界表示为 lanelet 的边。这些边界捕捉标记的车道边界或禁止驾驶的曲线。

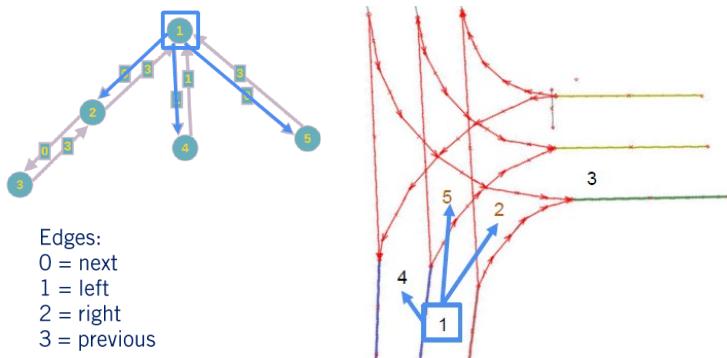
车道边界存储为一组创建连续多边形线的点。每个点存储其 x、y 和 z GPS 坐标。点之间的距离可以精确到几厘米，也可以精确到几米，这取决于所讨论的多段线的平滑度。使用这种多边形线结构便于收集运动规划任务所需的数据。点的顺序确定了行驶方向和 lanelet 的朝向 heading。车道的任何部分的道路曲率 curvature 也可以从该车道预计算。两个边界之间的中心线 center line 可以插值 interpolated，它可以作为自动驾驶车辆在该车道上的期望行驶路径。

有两种类型的 lanelet 规则：位于 lanelet 末尾的规则元素和影响 lanelet 整体的规则属性。规则元素表示为由一组共线点定义的线。监管要素通常要求采取行动或作出决定，例如在交通信号灯的情况下，或必须根据该交通信号灯的当前状态作出继续行驶的决定。整个 lanelet 都具有监管属性。例如限速，或者这条车道是否在交叉口或合流处与另一条车道交叉。

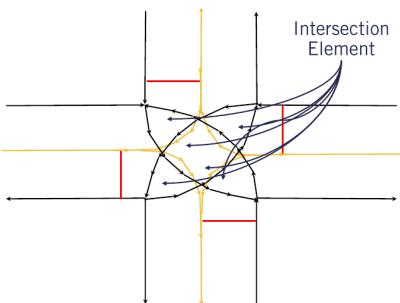
每个 lanelet 有四个可能的连接：lanelet 直接到左边 lanelets directly to the left, lanelet 直接到右边 lanelets directly to the right, lanelet 在它前面 lanelet preceding it, lanelet 在它后面 lanelet following it。整个 lanelet 结构连接在一个有向图 **directed graph** 中，这是 HD 图的基本结构。应该注意的是，在当前 lanelet 之前可能有多个 lanelet，或者在交叉口的情况下，在 lanelet 之后可能有多个 lanelet。

为了更好地理解这个概念，让我们看一个简单的例子，在三向交叉口使用这组特殊的车道。如您所见，一些 lanelet 被编号并表示为图上的节点，因此这些节点仅表示具有此交集的完整 lanelet 图的一部分。节点或顶点在该图形所在的 lanelet 上标记为 1 到 5，并形成 lanelet 图的顶点。该图的所有边都是有向的，并被标记为表示边所连接的两个顶点之间的关系。标记为 0 的边表示到下一个 lanelet 的过渡。1 表示连接到左 lanelet。标记为 2 的边表示连接到右 lanelet，标记为 3 的边表示以前的 lanelet。

在这张图片中由数字 1 表示的 lanelet 元素，实际上有两个可能的下一个元素：Lanelet 元素 5 和 Lanelet element 2，在离开 Lanelet 1 后可以继续行驶，因为它们是通过此交叉口的两条路。类似地，lanelet 元素 4 与 lanelet 元素 1 的边相连，因为 lanelet 元素 4 在 lanelet 元素 1 的左边。



交集元素 intersection element 只是持有一个指向所有监管元素的指针，这些元素构成了感兴趣的交集 intersection of interest。作为交集一部分的所有 lanelet 元素也指向该交集元素。这个结构的作用很像一个容器，在分配行为时简化了整个 lanelet 结构的查找。



lanelet 数据结构的一大优点是，它使部分运动规划过程更简单，计算效率更高。通过复杂的多车道道

路网络的路径规划过程，需要在转弯之前进行多个连接点的更改，由于每个车道被视为一个单独的顶点，因此可以使用此数据结构进行路径规划。将动态对象定位到已知地图可以改进路径预测。这种定位能力还通过提供一种简单的方法来规划自动驾驶汽车通过繁忙十字路口的行为，从而改善与动态对象的交互。

创建这样的地图可以通过三种方式来完成：第一种是离线创建地图，通过多次驱动路网并采集信息，然后融合分割信息和定位信息，提高地图的精度。在这种方法中，如果从算法中发现任何错误，也可以进行手动校正。第二种方法是在第一次驾驶道路网时在线创建 lanelet 地图。利用现有的传统地图，在很大程度上依赖于分割和目标检测，可以在自动驾驶车辆首次通过道路网络时创建地图。该方法在运动规划任务和感知任务方面都具有很高的计算开销。更不用说极易出错，因此很少部署。创建高清地图的第三种方法是离线创建地图，如果检测到更改，则在线更新地图。一个这样的变化可能是一个新的建设区导致了一个新的监管元素，这是在地图创建过程中无法预测的。事实上，所有这些方法都使用相同的底层计算来识别 lanelet 元素、属性和交集。在实践中，带有实时确认和更新的离线地图构建抓住了这两个方法的优点，确保了对不经常变化的静态元素的高精度，同时仍然捕获了环境中的新颖性并允许车辆适应它。

- S. Thrun, W. Burgard, and D. Fox, [Probabilistic robotics](#). Cambridge, MA: MIT Press, 2010. Read Chapter 9 - Occupancy Grid Mapping for an overview of how occupancy grids are generated.
- P. Bender, J. Ziegler, and C. Stiller, “[Lanelets: Efficient map representation for autonomous driving](#),” 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014. Introduces the concepts of lanelets used in mapping.

Lesson 1: Creating a Road Network Graph

在本单元中，我们将讨论自动驾驶中的任务规划问题以及如何解决该问题。自动驾驶任务是我们运动规划任务的最高级别部分，对于自动驾驶汽车导航到目的地至关重要。

首先，让我们回顾一下自动驾驶任务。自动驾驶任务的目标是通过导航道路网络，同时提取出道路规则和驾驶场景中存在的其他代理等较低层次的细节，为自我车辆找到从当前位置到给定目的地的最优路径。在本模块中，我们将从汽车到达目的地所需的时间或距离的角度来考虑最佳性。

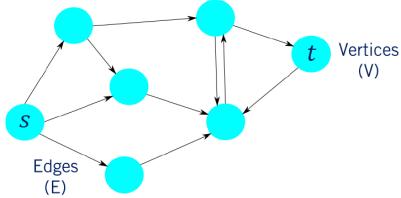
对于自动驾驶，任务规划被认为是最高层次的规划问题。这是因为任务规划器的空间规划规模是以公里为单位的，并且任务规划器不关注诸如障碍物或动力学之类的低级规划约束。相反，任务规划器在规划时将重点放在道路网的各个方面，例如限速和道路长度、交通流量和道路封闭。基于地图给我们带来的这些约束，任务规划人员需要找到到达我们所需目的地的最佳路径。关于道路网，需要注意的一点是，它是高度结构化的，我们可以在规划过程中利用它来简化问题。利用该结构，我们可以根据给定的地图有效地找到到达目的地的最优路径。要做到这一点，我们需要使用一个被称为图 graph 的数学结构，我们在这里将其覆盖到我们的道路网络上。那么什么是图呢？图是由一组表示为 V 的顶

点 vertex 和一组表示为 E 的边 edge 组成的离散结构 $G = (V, E)$ 。对于任务规划器，V 中的每个顶点将对应于道路网络上的给定点，并且每个边 E 将对应于连接道路网络中任意两点的路段。从这个意义上说，图中连续边的序列对应于通过道路网从一点到另一点的路径。

请注意，一般情况下，仅因为点 A 与使用路段的点 B 相邻，并不意味着可以从相同路段的点 B 到达点 A。这是因为在许多情况下，只有一个方向的路段可以合法通过。在这个意义上，图的边是有方向的，因为边只能在一个方向上遍历。我们在图中用箭头表示边来显示它们的方向性。

Graphs

Graph: $G = (V, E)$



既然有了有向图，我们如何找到到达目的地的最优路径？首先，我们定位图中的顶点，这些顶点对应于我们当前的自我车辆位置，我们将其表示为 s 和我们期望的目的地，我们将其表示为 t。一旦我们有了这两个顶点，我们就可以使用一种高效的图搜索算法来寻找到达目的地的最优或最短路径。由于我们的图公式目前是未加权的，一个很好的候选算法是广度优先搜索 **breadth first search (BFS)**。它是一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。

队列 queue 是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作，和栈一样，队列是一种操作受限制的线性表。进行插入操作的端称为队尾，进行删除操作的端称为队头。队列中没有元素时，称为空队列。

在较高的层次上，BFS 可以被认为是遍历图中的所有顶点，但这样做的方式是，在深入到图中之前，首先计算所有相邻顶点。从这个意义上说，图形搜索就像一个移动的波前 *wavefront*。

让我们看看 BFS 算法的步骤。我们构造了三个数据结构来帮助我们的搜索：一个开放的待评估顶点队列 open queue of vertices，一个由搜索算法评估的闭合顶点集 closed set of vertices 和一个存储搜索结果的前任词典 dictionary of predecessors。

队列是一种先进先出（FIFO - first in, first out）的数据结构，因此推送或添加到队列中的第一个顶点是从队列中弹出或返回的第一个顶点。

字典是一组无序的键值对，对于闭集中的每个节点，存储一个将立即标识的前任顶点。

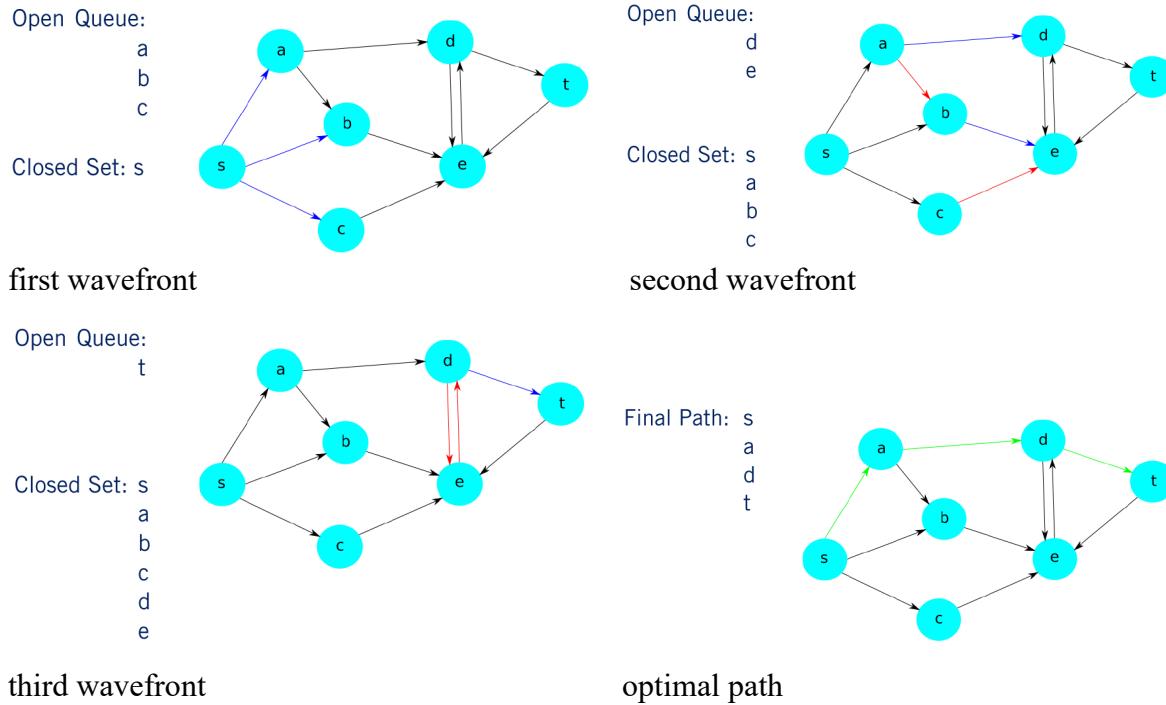
该算法首先将起始顶点添加到开放队列中。然后，当开放队列包含顶点时，我们从开放队列中获取第一个元素并检查它是否是目标位置。如果是这样，我们找到了最短的路。

如果没有，则将所有不在开放队列或封闭集中的相邻顶点添加到开放队列。这可以防止我们在图形搜索过程中陷入循环。请注意，“相邻”是指从当前顶点可以到达的所有顶点。因为我们使用队列来存储打开的顶点，所以我们确保在深入到图中之前处理搜索中当前深度处的所有相邻顶点(即广度优先)。因此，在移动到距起始顶点一步远的顶点之前，将对距起始顶点一步远的所有顶点进行处理。

当一个顶点被添加到开放队列中时，我们将其前面的顶点存储在前置字典中。一旦找到目标，这将帮

助我们重建最优路径。最后，我们将当前活动的顶点添加到封闭集，并返回到开放队列的下一个元素进行处理。

由于 BFS 算法的广度优先特性，当我们到达目标顶点时，我们已经处理了所有可能的目标顶点的前任顶点，这些顶点比目标顶点更接近起始顶点。这意味着当我们到达目标顶点时，我们已经找到了到达目标顶点的最短路径，我们可以终止算法。



与我们在图中的最优路径对应的边序列可以使用我们的地图转化为道路网络上的根，然后可以使用地图来管理我们规划层次结构的后续层中更详细的运动规划。

还有一个密切相关的深度优先搜索算法 depth first search。深度优先搜索使用后进先出堆栈 last in, first out stack，而不是开放集的队列 queue for the open set。此更改意味着将计算最近添加的顶点，而不是最旧的顶点。结果是搜索在图形中迅速深入，然后最终返回到更早添加的顶点。

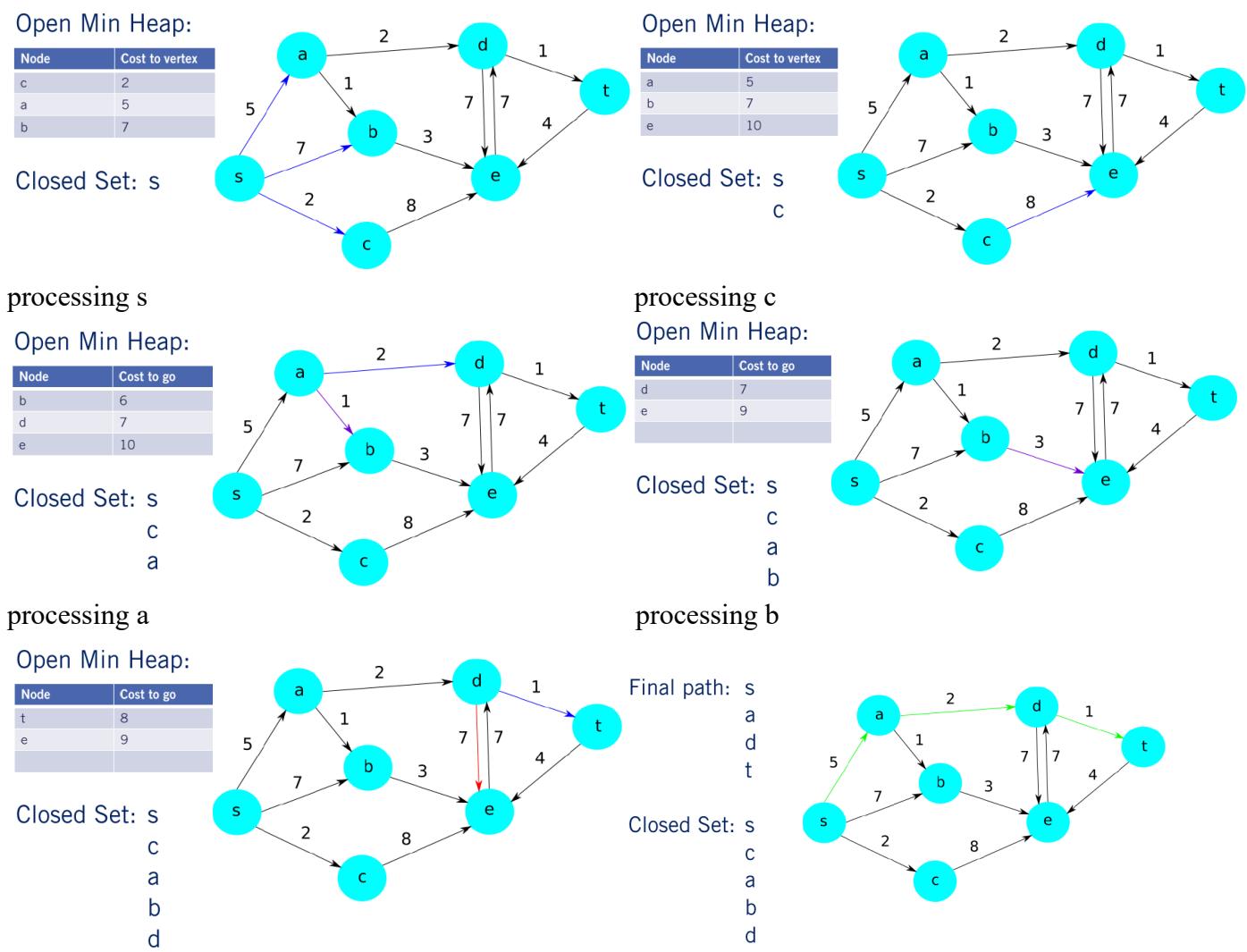
Lesson 2: Dijkstra's Shortest Path Search

在本课中，我们将修改上一课中的未加权图，以包含边权重。作为复习，回想一下对于我们的任务规划问题，目标是在我们的道路网络中找到从当前位置到所需最终目的地的最优路径。在上一个视频中，我们提出了广度优先搜索算法来解决这个问题。然而，在这个过程中，我们假设所有路段长度相等，这是一个过于简单的假设。根据道路长度、交通量、限速和天气等因素，不同路段的通行成本可能会有很大差异。为了简单起见，我们最初将只关注搜索图中的距离。为了反映这一点，我们将向图中的每条边添加边权重，这些边对应于相应路段的长度。在使用适当的边权重更新我们的未加权图之后，这里显示了这一点。权重的单位是任意的，只要它们对所有边都是公共的。

对于本例，假设边权重是穿过该路段所需的公里数。与以前一样，我们的目标是找到从当前位置 S 的

顶点到最终目标顶点 T 的最优路径。不幸的是，我们的 BFS 算法没有考虑边的权重。因此，在这种情况下不能保证找到最优路径。相反，我们需要使用另一种更强大的算法。这就是 Dijkstra 算法的用武之地。

Dijkstra 算法的整体流程与 BFS 非常相似。主要区别在于我们处理顶点的顺序。和以前一样，我们跟踪在封闭集中已经处理的顶点，以及在开放集中已经发现但尚未处理的顶点。关键的区别在于，我们将使用最小堆，而不是对开放集使用队列。最小堆是一种数据结构，它存储键和值，并根据它们的关联值从小到大对键进行排序。在我们的例子中，图中每个关键顶点的值将对应于到达该顶点所需的距离，沿着到目前为止我们找到的顶点的最短路径。从这个意义上说，**Dijkstra** 的算法处理顶点的累积成本比其他顶点低。因此，与 BFS 不同，在搜索中稍后添加的顶点可以在较早添加之前进行处理，只要其累积成本较低。除此之外，Dijkstra 的算法与 BFS 基本相同。在顶点中前进，同时将它们从最小堆中添加和弹出，直到处理目标顶点为止。然而，一个有趣的例子是，如果我们找到一个新路径，指向一个已经在开放堆中但尚未处理的顶点。在这种情况下，我们必须检查新找到的到这个顶点的路径是否比旧路径便宜。如果是的话，我们需要在最小堆中更新它的开销，否则，不需要任何操作。一旦我们处理了目标顶点，我们就必须处理目标节点的所有可能的前导顶点。因为前置对象的累积距离必须小于或等于目标顶点。由于所有前置顶点都已处理，我们将找到到目标顶点的最短路径。一旦目标顶点被处理，算法就可以终止。



虽然 Dijkstra 算法是一种有效的算法，但我们可以利用某些启发式算法使其在实践中更快。

Lesson 3: A* Shortest Path Search

在本课中，我们将在 Dijkstra 算法的基础上，引入一种新的可能更快的方法，用于任务规划问题。为了做到这一点，我们将在搜索中利用启发式 **heuristics**，这将帮助我们改进搜索过程，使其更有效。

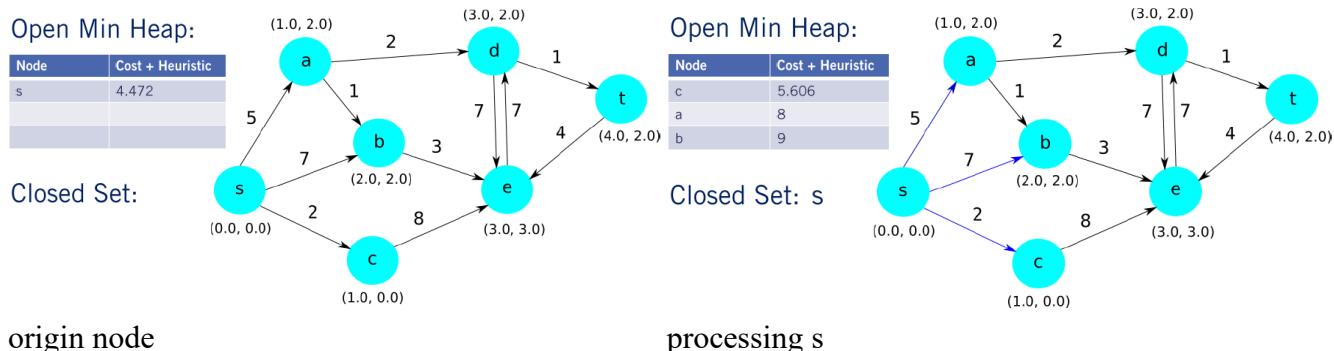
如果您还记得上一课，我们介绍了 Dijkstra 的算法来帮助我们解决加权图边的任务规划问题，这比我们以前的未加权图实例更现实，因为它允许我们考虑不同路段的可变距离。然而，Dijkstra 的算法要求我们搜索图中几乎所有的边，即使只有少数边对构造最优路径有用。对于我们的小示例图来说，这不是问题。当我们将问题扩展到更现实的比例时，它会引发问题，例如一个城市的完整道路网。

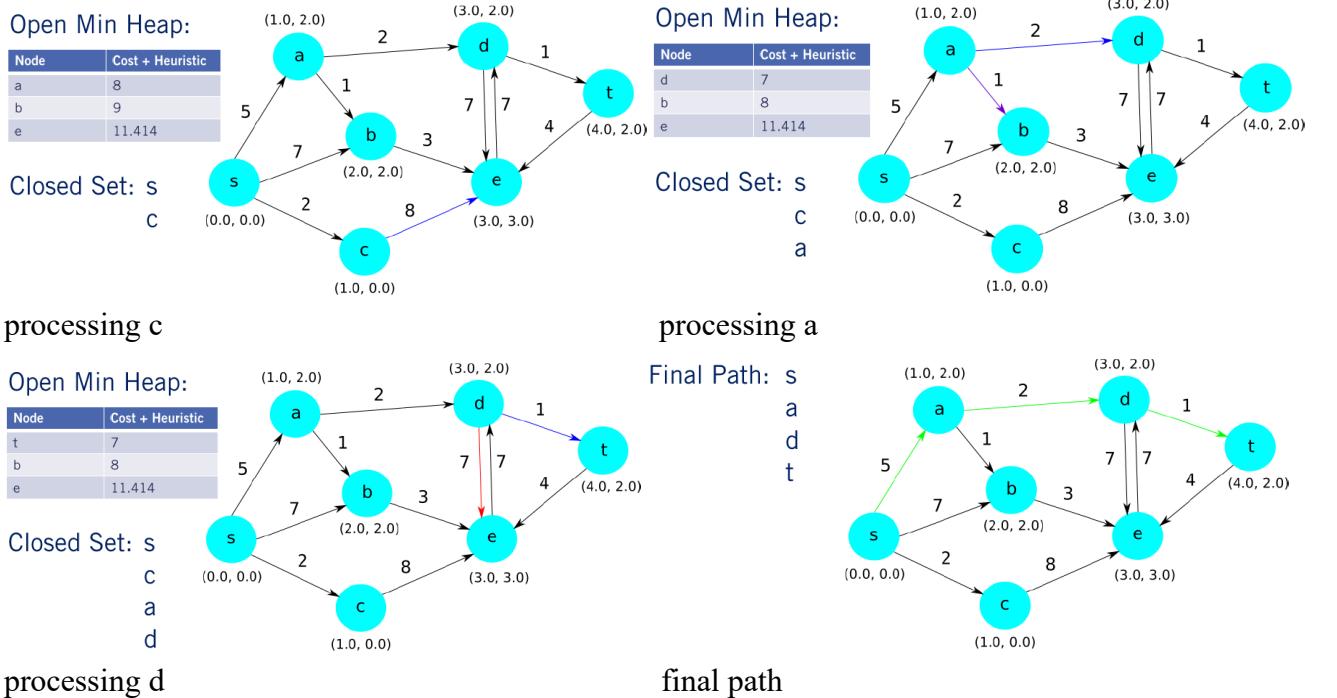
为了提高我们在实践中的效率，我们可以通过使用 A*算法而不是 Dijkstra 的搜索启发式来找到我们的目的地。什么是搜索启发式？在这种情况下，**搜索启发式 search heuristic** 是对从图中任何给定顶点到达目的顶点的剩余代价的估计。当然，我们使用的任何启发式方法都不是精确的，因为这需要我们已经知道搜索问题的答案。

对于给定的顶点 v 和目标 t ，对任意两个顶点之间的成本或长度的一个有用估计是它们之间的直线或欧几里德距离 $h(v) = ||t - v||$ (Euclidean heuristic)。因此，对于我们在搜索中遇到的任何顶点，我们对目标顶点的剩余成本的估计将只是该顶点和目标之间的欧几里德距离。注意，这个估计值总是低估了到达目标的真实距离，因为任意两点之间的最短路径是一条直线。这是 A*搜索的一个重要要求，满足此要求的启发式算法称为容许启发式算法 admissible heuristics。

A*算法与 Dijkstra 的算法基本相同，但有一些关键的区别。

Dijkstra 算法和 A*之间的最小区别是，我们不使用累计成本，而是使用累计成本加上 $h(v)$ ，即启发式估计的目标顶点的剩余成本，作为我们推送到最小堆 min heap 上的值。然后，最小堆根据目标的估计总成本对开放顶点进行排序。从这个意义上说，根据我们的搜索启发式，A*将搜索偏向可能是最优路径一部分的顶点。因为我们存储的是基于启发式的总成本和最小堆，所以我们还需要跟踪每个顶点的真实成本，这些成本存储在成本结构中。有趣的是，如果我们把所有顶点的启发式设为 0，这仍然是一个可接受的启发式，那么我们就得到 Dijkstra 算法。与 Dijkstra 算法之前一样，我们将把原点添加到最小堆，然后从堆中弹出每个顶点，并将所有相邻顶点添加到堆中，直到处理目标顶点为止。





当我们把问题扩展到更大的图形输入时，我们将看到 A* 算法中使用的启发式将导致它探索的顶点远远少于 Dijkstra 算法。渐进地说，A* 永远不会比 Dijkstra 做得更糟，实际上 A* 会导致更快的任务规划过程。

现在，我们已经简化了任务规划问题，例如在地图中边缘权重或沿路段的距离。然而，如果我们将交通、速度限制或天气等其他因素纳入我们的任务规划问题，那么路径上的道路距离将过于简单，无法捕捉问题的范围。为了解决这个问题，我们可以将估计的穿过路段的时间作为我们的边权重，这将考虑所有提到的因素。然而，这使得我们的欧几里德距离度量毫无用处，因为它不再捕捉到目标在时间方面的真正成本。为了解决这个问题，我们可以使用到目标点的时间的下限估计值，即欧几里德距离除以所有路段允许的最大速度。汽车不应超速行驶。因此，即使在理想的交通和天气条件下，以及在通往目标的直线路径上，这也是汽车能够到达该目标的绝对最短时间。这意味着它始终是目标的真实成本的下限，因此它是一个可接受的启发式。

与严格关注最小化距离的问题实例相比，较低的下界会降低我们的启发式算法引导我们搜索到更复杂问题的目标的能力。更先进的方法可用于预先计算附加值，并考虑修改的启发式定义，允许有效搜索基于时间的行程估计的大型网络。

Lesson 1: Motion Prediction

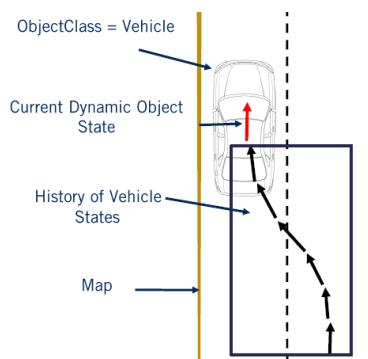
运动预测试图估计在一定的范围内的环境中的所有动态物体未来的位置，朝向，和速度。这对于运动规划问题至关重要，因为它允许我们根据其他物体的预期运动来规划自主车辆的未来动作和机动。预测的路径也允许我们确保自我车辆规划执行的路径，在未来时间不会与任何未来物体碰撞。为了能够预测运动物体的运动，我们必须获得一些关于我们周围环境的信息。尤其是当它涉及到动态对象时。对于所有动态对象，首先要知道对象的类 class。这些信息非常重要，因为大多数预测模型对车辆和行

人有不同的算法方法。接下来，我们需要了解动态对象的当前状态、位置以及在环境中的速度。

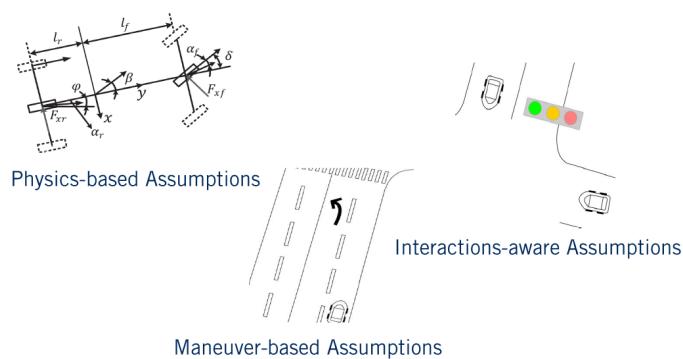
还有许多其他的信息，虽然不需要做出预测，但可以大大提高预测的准确性。一些用于改进预测的附加信息的主要来源，首先是**动态车辆状态的历史或车辆在环境中移动时的轨迹**。这是非常有用的。正如我们在示例中看到的，我们可以看到车辆状态历史显示为黑色箭头，位置航向和速度如前所示。

高清路线图也可以用作附加信息源，以确定动态对象的未来行为。正如本单元将进一步讨论的，车辆在行驶时往往会沿着各自的车道行驶。这可以提供强有力的线索以提高预测的准确性。

当前的动态对象状态也可以是有用的信息源，可以改进预测。这对车辆和行人都是如此。例如，对于车辆，图像可以提供有关当前指示灯或制动灯状态的信息。同样，行人的图像可以用来显示人的当前方向，这有助于预测未来的行驶方向，即使行人目前静止不动。



虽然运动预测任务的复杂性相当大，但是有几个假设可以用来简化问题。我们将从车辆的简化开始，然后讨论行人。我们所依赖的第一类假设是，**车辆必须遵循一组控制其运动的物理约束**。正如我们在课程一中讨论车辆运动学和动力学时所看到的。这些完全相同的车辆动力学可以应用于环境中的其他车辆，以预测其运动。我们把这种类型的预测称为基于物理的预测。可以使用的第二类假设是，**车辆在道路上的几乎所有运动都是由环境中受限域内的有限组策略组成的**。在这种情况下，我们假设道路上的车辆将留在道路上并遵守驾驶规则。例如，除非另有指示，否则他们很可能会留在车道上，并在需要停车的监管要素处停车。他们不太可能开车穿过人行道或草坪或障碍物。我们将这种假设称为基于机动的假设。最后，第三类的假设与基于机动的假设相同。然而，我们也可以**将动态物体相互反应和相互作用的假设结合起来，而不是仅仅独立地评估每辆车**。这类预测的一个例子是在车辆并入相邻车道的过程中。通常，目的车道上的车辆会减速，以便为进入的车辆留出更多空间，以保持安全的跟车距离。这些类型的假设称为交互感知假设。



对于行人，同样的三个类别可以用来总结我们可以做出的假设。根据基于物理的假设，行人的最高速度往往较低，但可以很快改变他们的运动方向和速度。这使得行人很难用纯粹基于物理的假设进行可靠的预测，但行人在短时间内能够到达的位置范围是有限的。对于基于机动的假设，行人往往不会与

车辆直接互动。因为他们在运动时主要使用人行道或其他行人专用区。当进入环境的可行驶区域时，他们可能会接触到车辆，他们主要使用人行横道。尽管将行人运动限制在这些区域是一个合理的假设，但这并不是一个硬约束，行人的不可预测性要求对其未来行为保持多种可能的假设。最后，行人最终拥有通行权，必要时自动驾驶汽车有义务停车。漫不经心的行人可能会在没有任何警告的情况下进入道路，但在受到迎面而来车辆的威胁时往往会展开双臂停下来。这些类型的交互假设也可用于行人的运动预测。

现在我们对运动预测有了更好的理解，让我们看看一个简单的计算效率高的算法，它可以同样适用于行人和车辆。该算法只对动态物体的运动做了一个极端的假设。**所有动态物体都将保持其当前速度的大小和方向 constant velocity prediction model。**

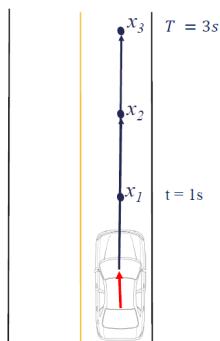
该算法采用三个基本输入：可预测的时间范围 time horizon 或预测对象未来位置的时间量 T ，更新率或路径模拟频率，即状态预测之间的时间量 dt ，当前对象的状态 x_{obj} ，包括动态对象的位置和速度。该算法以 dt 为增量，从当前时间零点迭代到视界 T 结束。

该算法的输出是预测视界中每个时间步的预测对象状态、位置和速度的列表 $x_{1:T}$

(tip: time horizon 是未来某个固定的时间点，在该时间点上，某些过程将被评估或假定为结束。)

- Input:
 - $T = 3$ seconds
 - $dt = 1$ second
 - x_{obj}
- Output:
 - Predictions

$x_1 \quad x_2 \quad x_3$



该算法无法捕获车辆动力学模型的全部复杂性，甚至无法捕获车辆加速或减速或应用非零转向指令的能力。等速假设的另一大缺陷是，它无法解释车辆跟随道路形状变化的趋势。同样，等速预测也没有考虑到路标对速度的调整。接近停车标志的车辆往往会减速，离开停车线的车辆往往会加速。该算法所作的假设是相当强的，不适用于大多数动态对象可以观察到的情况。

Lesson 2: Map-Aware Motion Prediction

在上一个视频中，我们探讨了等速运动预测，它仅在非常有限的场景中工作良好。

地图感知算法通过两大假设来改善特别是针对车的运动预测。**基于位置的假设 positional assumptions** 改善了车辆状态的位置分量，**基于速度的假设 velocity assumptions** 改善了速度分量。

为改进预测中的位置分量所做的第一个假设是，沿给定车道行驶的车辆通常会沿着该车道行驶。回到我们的简单示例场景，如果我们有一辆车在弯曲的道路上，车辆很可能会沿着道路转弯。可以做出的第二个假设是，可以根据车辆指示灯 **indicator signal** 的状态进行车道改变或方向预测。如果已经由车辆的感知堆栈进行了这样的检测，则可以切换预测以考虑该附加信息。基于速度的假设用于改进动态对象的速度预测。道路上的所有车辆都会受到道路几何结构的影响。因此，当车急转弯 **tight turn** 时，车辆可能会减速以避免超过其横向加速度限制。最后，如果同时考虑动态物体可能遇到的调控因素，

速度预测可以大大提高。例如，如果在一个动态物体前面的近距离处有一个停车标志，可以安全地假设车辆将执行减速停车操作，从而随着时间的推移速度降低。

添加到预测模型中的约束或假设越多，它对所有交通场景的概括性就越差。

让我们将这些基于地图的假设结合到我们的运动预测方法中。对于具有自然曲率的道路，我们假设位于车道上的车辆很可能沿着该车道通过曲线。我们的运动预测可以通过使用 **lanelet** 地图的中心线作为车辆的预测路径来更新，而不是由等速预测生成的直线路径。回想一下本课程模块 2 中地图定义的车道，该模块为道路上的每条车道提供了左侧和右侧车道边界，从中可以有一条中心线来构建车道。车道的中心线定义为组成一条多段线的一组点，该多段线与两条车道边界的间距相等。虽然可以预期与精确中心线的微小偏差，但中心线可以作为良好的运动预测近似。

但是，通过将路径预测限制在任何给定车道的中心线，将导致两个主要问题。首先是在没有额外信息时预测其他车辆的车道变换策略是困难的。在正常驾驶过程中，驾驶员经常变换车道。因此，如前所述，可以基于指示灯感知来预测这种机动策略。尽管并非所有由人类驾驶人操作之前都有指示。出现的第二个问题是，通常情况下，有多条 **lanelet** 中心线可供选择，比如在十字路口。例如在 T 形交叉路口，车辆可以左转或右转。这使得我们需要考虑基于场景中其他代理的可能行为的多个假设。

同样，我们有两个选择：我们可以使用对象的状态、外观和轨迹信息来识别最可能的行为，然后基于最可能的行为构建预测 **most likely prediction**，或者，我们可以构造一个最有可能的行为的预测，并根据状态出现和轨迹信息关联代理沿着特定路径的概率。

第二种方法是对第一种方法的略微概括，称之为多假设预测 **multi-hypothesis prediction**。在多假设预测方法的情况下，基于车辆在 HD 路线图中的当前位置处可用的全部可能性，考虑车辆的每个标称行为 **nominal behaviour**。

对于三向交叉口示例，我们可以包括三种可能性：左转、右转或保持静止。基于诸如指示灯、信号灯、中心线左侧或右侧的位置以及交叉口处车辆的状态等确凿证据。可以根据代理在预测范围内执行每一个假设的可能性来评估这三个假设中的每一个。这些概率很难精确地量化。因此，既可以从许多车辆通过类似交叉口的训练数据中学习，也可以从实际测试中设计和改进。传统上，这些方法为行为规划器提供了更多的模糊信息，需要在局部规划器中同时考虑多个场景，并处理信念的概率表示。然而，如果处理得当，这种方法也可以大大安全，启用防御驾驶策略，并在基于新信息的概率变化时快速重新规划。这种方法还有一个优点，就是能够适应基于人的驾驶错误，比如换车道时忘了打信号。

既然我们了解了如何使用路线图来改进预测轨迹的位置分量，现在让我们深入研究速度分量。这方面的第一个改进是基于已知的道路几何或曲率，以及预测其他车辆将如何对此做出反应。所有车辆无论其制造型号如何，在进入急弯或转弯时都会降低速度。我们可以使用预期的最大横向加速度，通常在 0.5 到 1 米/秒平方的范围内，来改进沿曲线的速度估计。

第二个也是更重要的改进是加入监管元素以改进速度估计。给定车辆的预期路径，任何道路要素（如停车标志、让行标志 yield sign、限速变化或红绿灯）都可以通知速度预测。如果是红绿灯，还需要指示灯状态。在每种情况下，可根据路线图中规定的监管要素线预测停车位置。然后将平滑减速应用于沿其路径的车速预测。事实上，给定一个 HD 路线图，就有可能对沿每条道路的标称轨迹的地图进行预处理 preprocess，并根据标称驾驶行为定义 lanelet 明确的多假设先验 lanelet specific multi-hypothesis

priors。这既可以作为自我载体在规划其行为和轨迹时的指导，也可以用于改进其他主体的运动预测。当然，车辆前方车道上的障碍物优先于交叉口处的到达信息和车道上的引导车辆，所有这些都可以进行集成，以改进对其他车辆的预测。

考虑到单个自动驾驶汽车决策的复杂性，可以使用多少变量和逻辑中的信息深度来改进运动预测是有限制的。我们可以依赖于预期动态对象行为的假设来预测未来行为的程度也是有限的。**动态对象的行为并不总是与驱动程序预期的标称行为一致**。他们不完全遵循中心线，不以限速行驶，例如以不同的速度加速和减速。此外，他们可能会对预测系统尚不可用的信息做出反应，例如前方道路上的坑洞或弹跳球。用这种方法**无法预测偏离路线图的车辆**。当车辆意外闯红灯时，他们可能根本不遵守法规。所有这些变化都必须加以考虑，这在某种程度上可以通过多假设方法来实现。因此，最好的方法是跟踪信念在一组假设中的演变，并在每个时间步根据来自感知堆栈的证据进行更新。这些方法非常复杂。

Lesson 3: Time to Collision

碰撞时间是衡量自动驾驶车辆行为安全性的重要指标，在基于当前驾驶环境的潜在机动评估中有着广泛的应用。通过了解碰撞是否迫在眉睫，何时可能发生，以及与哪个物体发生碰撞，自动驾驶系统可以更好地规划穿越环境的安全机动策略，或者在需要时准备紧急避让行动。

为了评估动态对象之间的碰撞时间，我们使用它们的预测路径来识别可能的碰撞点。如果存在碰撞点，则碰撞时间是对发生碰撞之前的时间量的度量。因此，计算碰撞时间可以分两步完成。首先，我们沿着动态对象之间的预测路径识别**和计算碰撞点的位置**，然后，如果存在这样的碰撞点，我们计算**动态对象到达该碰撞点所需的时间**。

计算碰撞时间的挑战在于，它在很大程度上依赖于对未来物体将占据的空间的准确理解，以获得准确的估计。显然，**准确预测物体的运动轨迹**对于确定碰撞发生时的碰撞点至关重要。精确预测物体的运动轨迹是我们在前两课中看到的一个难题。

此外，我们还需要**环境中所有动态对象的精确几何图形**。当计算碰撞时间时，需要考虑两辆车的几何形状来计算准确的碰撞点。由于环境中的对象只能从当前和以前的视野看到，因此在驾驶时精确定义其几何图形可能具有挑战性。由于这两个要求都有错误，碰撞时间应始终视为近似值，定期更新，并在做出驾驶决策时使用一些安全缓冲。

计算两个物体碰撞时间的方法主要有两种。**基于仿真的方法 simulation approach** 和**基于估计的方法 estimation approach**。

基于仿真的方法模拟了这个场景中每辆车在未来有限时间内的运动。在每个模拟时间步，计算每个动态对象的新位置、航向和占用范围预测。这是通过向前传播 propagate forward 预测轨迹来实现的。一旦在模拟未来的给定时间内知道所有动态对象的位置，将进行检查以确定是否有任何两个或多个动态对象相互碰撞。如果它们发生碰撞，则记录碰撞的位置和时间。请注意，这是一种不同于静态情况的碰撞检查类型，在静态情况下，构建整个路径的线束，并将其与静态对象位置进行比较。因为这些碰撞检查是在移动对象之间执行的，所以我们只需要在每个时刻检查每个对象的几何体之间的碰撞，因为对象将在下一个时间步占据新的位置。

仿真时间步长的大小可以不同于预测的路径时间步长，以提高碰撞检查近似精度。

基于估计的方法通过计算每辆车在其预测路径中移动时的几何演变来发挥作用。结果是每辆车的一个样本，然后可以比较潜在的碰撞。一旦计算了线束 swath，如果任意两个或多个动态对象将同时占用同一空间，则可以通过计算探索它们的交点以寻找潜在的碰撞点。如果这是真的，潜在的碰撞点被标记，并且每个车辆何时到达每个碰撞点的估计被用来估计碰撞时间。传统上，这种方法通过简化假设来加速计算。这些假设包括：基于物体路径交点确定碰撞点位置，基于简单几何图元（如边界框）估计物体空间占用率，以及基于等速剖面估计到达碰撞点的时间。

每种方法都有各自的优点和缺点。基于估计的方法依赖于简化的假设，这使得碰撞时间的计算在内存占用 **memory footprint** 和处理工作方面的计算成本更低。仿真方法的计算成本往往更高，因为它们涉及逐步评估和对象几何体相交计算。然而，由于其简化的假设，基于估计的方法通常会对碰撞点和碰撞时间产生不太准确的估计。另一方面，仿真方法不需要依赖于这些近似，并且步长小，可以对碰撞时间进行高保真评估。这些相对的差异使得每种算法适合不同的应用。对于实时应用，如车载 onboard 碰撞时间计算，基于估计的方法是首选。而离线情况下，在数据集创建和基于仿真的测试等精度至关重要的情况下，则首选基于仿真的方法。

Simulation Approach

- Computationally expensive
- Higher accuracy if simulated with high fidelity
- Offline Applications (Dataset evaluation or Simulations)

Estimation Approach

- Computationally inexpensive
 - Memory footprint
 - Computational time
- Less accurate due to approximations and estimations
- Real Time Applications (In Car Prediction)

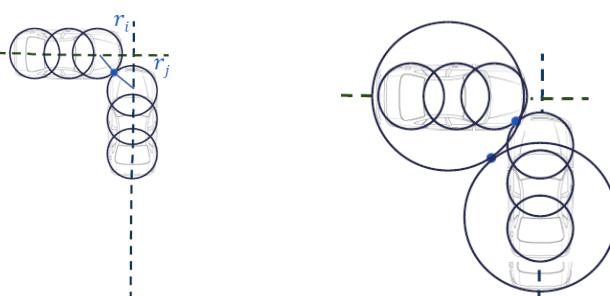
一个简单的基于仿真的方法来计算碰撞点和到达每个碰撞点的时间。

1. Represent each car as a set of circles
2. Check if a collision will occur between two circles

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \quad r_i + r_j \geq d_{i,j}$$

3. Calculate collision point

$$c_x = \frac{(x_i * r_j) + (x_j * r_i)}{(r_i + r_j)}, \quad c_y = \frac{(y_i * r_j) + (y_j * r_i)}{(r_i + r_j)}$$



通常情况下，在碰撞时间估计的准确性 **accuracy** 和总体计算时间 **number of computations** 之间存在权衡 **tradeoff**。当我们增加碰撞检查的保真度时，我们也增加了所需的计算数量。同样地，如果我们考虑的圆的数量减少到一个，计算的数量会减少，但是精确碰撞点的精度会显著降低。

- C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms, "[Autonomous Driving in Traffic: Boss and the Urban Challenge](#)," AI Magazine, vol. 30, no. 2, p. 17, 2009. This gives an overview of some of the methods used to handle dynamic obstacles in the DARPA Urban Challenge.

Lesson 1: Behaviour Planning

一个行为规划系统，规划一组高级驾驶动作 **action** 或机动策略 **maneuver**，以在各种驾驶情况下安全地完成驾驶任务。规划的一系列机动应该考虑到道路规则 **rules of the road**，以及与环境中所有静态和动态物体的相互作用 **static and dynamic objects around the vehicle**。规划器做出的一系列高层次决策必须确保车辆在环境中的安全和高效运行 **safe and efficient**。

作为行为规划器角色的一个例子，假设自动驾驶车辆到达繁忙的十字路口。行为规划器必须计划何时何地停车、停车多长时间以及何时通过交叉口。行为规划器必须以计算效率高的方式执行此类决策，以便能够对环境的变化做出快速反应，并部署在自主车辆硬件上。行为规划器也应该能够处理输入不准确，被测量噪声破坏，又不正确，受到感知错误（如假阳性检测和假阴性检测）的影响。

我们将考虑行为规划器的五种行为。首先是**轨迹速度 track speed**。这种行为相当于在开放道路上无约束驾驶，这意味着向前行驶的唯一限制是应遵守速度限制。接下来，是**跟车 follow leader**。车辆前方的车速应匹配，并保持安全的跟车距离。三是**减速停车 decelerate to stop**。在规划视界内，ego 车辆的车道上存在一个停车点，车辆应在该停车点减速至静止。每一个需要完全停止的监管元素都会触发这种行为。接下来，**停车 stop**。车辆应在固定时间内保持静止。例如，当车辆在停车标志处停车时，应保持停车至少 3 秒钟。最后，**合流 merge**。此时车辆应汇入左侧或右侧车道。

然而，在实践中需要考虑更多的行为，因此，行为规划者的整体复杂性将会增加。

行为规划器的主要输出是在当前环境中执行的驾驶操作。随着驾驶机动的进行，行为规划器还输出了一组约束，约束了局部规划问题。我们将在本模块中使用的约束包括，从车辆当前位置到目标目的地的**默认路径 ideal path**，大多数是车辆当前车道的中心线。沿默认路径的速度限制 **speed limit**。在正常行驶条件下应保持的当前车道的**车道边界 lane boundary**。车辆需要到达的任何未来停车位置 **stop location**，并且仅当选择了相关机动时才应用 populated 此约束。最后，本地规划器应该关注的一组**动态对象 set of interest vehicles**。

这些动态对象可能是重要的，因为接近或估计未来的路径。为了使行为规划器能够产生所需的输出，它需要来自自治堆栈 **autonomy stack** 中许多其他软件系统的大量信息。首先，行为规划器依赖于对车辆附近道路网络的充分了解。这些知识来自高清路线图 **high definition road map**。接下来，行为规划器必须知道要走哪条路才能到达目标位置。这以任务路径 **mission path** 在道路网络图上的形式出现。此外，车辆的位置对于能够在车辆周围的本地环境 **localization information** 中正确定位高清路线图元素也至关重要。最后，行为规划者需要所有相关的感知信息 **perception information**，以便充分理解需要采取的行动，从而安全地启动任务。此信息包括环境中所有观察到的动态对象 **all observed dynamic objects**，如汽车、行人或自行车。对于每个动态对象，其当前状态、预测路径、碰撞点和碰撞时间都是必需的。它还包括所有观察到的处于各自状态的静态物体 **all observed static objects**，如停放的车辆、施工锥和红绿灯，并显示其状态。最后，它包括一个本地占用网格图 **occupancy grid**，定义执行机动的安全区域。

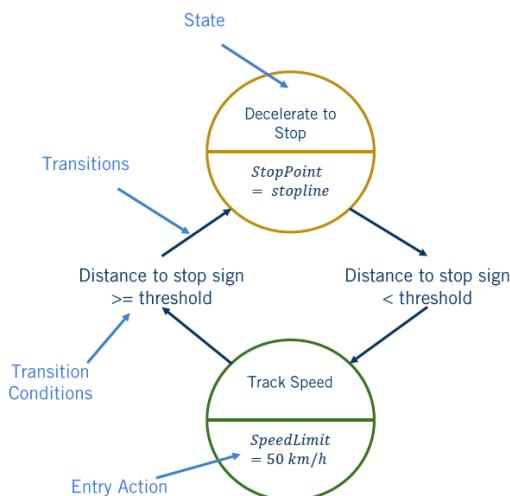
在所有必要的信息可用的情况下，行为规划器必须选择适当的行为，并定义必要的伴随约束，以保持

车辆的安全和高效行驶。为此，我们将显式 explicitly 或隐式 implicitly 地构造一组规则，考虑所有道路规则以及与其他动态对象的所有交互。

传统上用于表示解决行为选择所需的规则集的一种方法是有限状态机 finite state machine。在本模块中，我们将逐步构建一个基于有限状态机的行为规划器。我们将讨论这种方法的一些局限性。为了更好地理解有限状态机方法，让我们通过一个单一场景的有限状态机的简单示例，处理没有交通的停车标志交叉口。

有限状态机的第一组组件是状态集 the set of states。对于行为规划系统，状态将代表可能遇到的每一种驾驶行为。在我们的例子中，我们只需要两种可能的机动或状态，跟踪速度和减速停止。由行为规划器定义的机动决策由有限状态机的状态决定。每个状态都有一个与之相关联的进入动作，该动作是在第一次进入状态时要采取的动作。对于我们的行为规划器来说，这些输入操作包括设置必要的约束输出以伴随行为决策。例如，一旦我们进入减速停止状态，我们还必须定义沿路径的停止点。类似地，track speed 状态的输入条件将速度限制设置为 track。

有限状态机的第二组组件是转换，它定义了从一个状态到另一个状态的移动。在我们的两态例子中，我们可以从轨道速度到减速到停止，从减速到停止再到轨道速度。请注意，也可能有一些转换将我们返回到当前状态，从而触发 entry 操作以重复该状态。每个转换都伴随着一组转换条件，这些条件在转换到下一个状态之前需要满足。在状态中监视这些过渡条件，以确定何时应发生过渡。对于我们的简单示例，从轨道速度到减速到停车的过渡条件涉及检查停车点是否在当前车道的阈值距离内。同样地，如果我们在停止点达到零速度，我们可以从减速到停止再回到轨道速度。



这两个状态的例子突出了基于有限状态机的行为规划器最重要的方面。随着场景和行为数量的增加，所需的有限状态机变得更加复杂，具有更多的状态和转换条件。

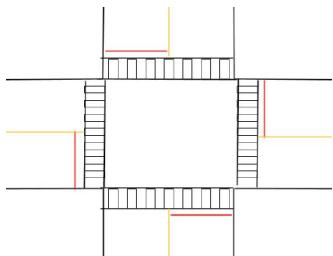
有限状态机是一种非常简单有效的行为规划工具。我们可以将其视为行为规划定义的直接实现，它要求我们定义机动或状态，以及满足道路规则的局部规划约束或进入动作，并检查在环境或过渡条件下与其他动态和静态对象的安全交互。通过跟踪当前机动和驾驶环境的状态，只需考虑当前状态以外的相关转换，大大减少了每次迭代时要检查的条件数量。由于将行为规划分解为一组状态，并在这些状态之间进行转换，因此所需的单个规则仍然相对简单。这将导致简单的实现，在不同的行为之间有明确的划分。

然而，随着状态数的增加，定义所有可能的转换及其条件的复杂性也随之增加。也没有明确的方法来处理输入数据中的不确定性和错误。这些挑战意味着，当我们接近完全的五级自治时，有限状态机方法往往会遇到困难。但对于操作设计领域受限的系统来说，这是一个很好的起点，允许有可管理的状

态数。在本模块的最后一段视频中，我们将探讨这些限制和行为规划的替代方法。

Lesson 2: Handling an Intersection Scenario Without Dynamic Objects

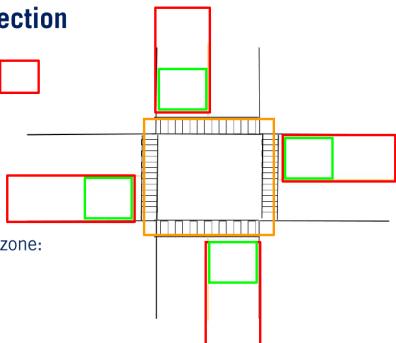
我们将尝试处理的场景是一个四向交叉口 four way intersection，每个方向有两条车道和停车标志。红线表示车辆需要停在后面的停车线。在视频结束时，这辆车应该能够左行、右行或直行通过这个十字路口。我们暂时不考虑动态对象。我们现在已经定义了一个非常有限的操作设计域，供我们的行为规划器处理。



让我们从研究如何离散化 discretize 交叉口开始，这样我们就可以更简单地在环境中做出决策。车辆应开始安全制动的交叉口区域被定义为交叉口的 **approaching zone**，以红色突出显示。交叉口中车辆必须停车并等待适当时间行驶的区域称为 **at zone**，以绿色突出显示。最后，车辆穿过实际交叉口的区域定义为 **on zone**，并以橙色突出显示。

Discretizing the Intersection

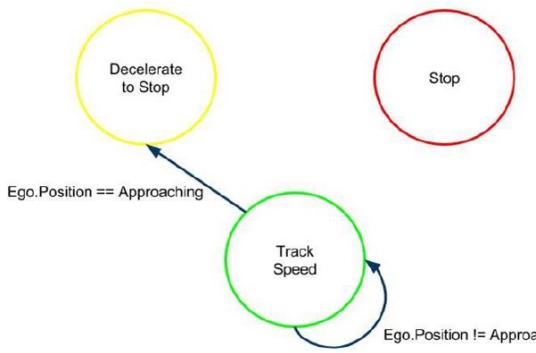
- Approaching an intersection
- At an intersection
- On an intersection
- Determining the size of each zone:
 - Ego vehicle velocity
 - Size of the intersection



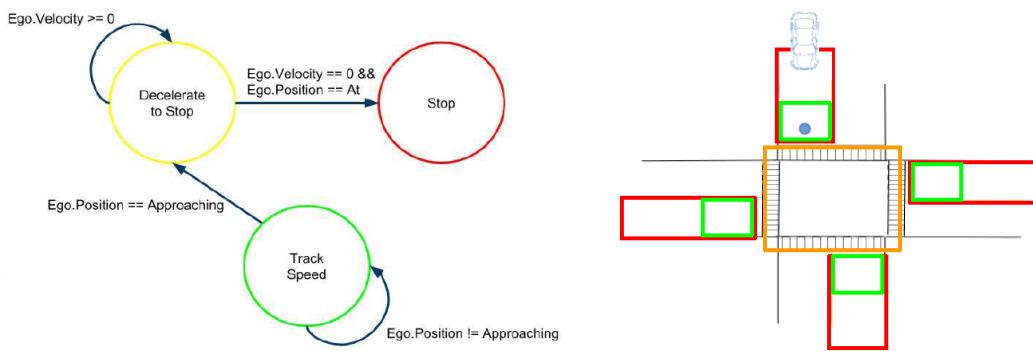
上述每个区域的大小是基于两个主要因素动态变化的：**车辆速度**，在较高的速度下，我们将需要更多的距离来安全舒适地停车，以及**交叉口的大小**。交叉口越大，每个区域就越大。

为了应对这种情况，我们需要三次高级驾驶机动。**轨道速度 track speed**，此机动状态仅受当前道路速度限制。传统上，这是在进入交叉口任何区域之前或安全进入交叉口区域后给出的机动。状态进入动作设置速度限制。**减速停止 decelerate to stop**，这种机动状态迫使物体的未来轨迹在到达停止点之前停止。进入动作定义了停止点的位置。**停止 stop**，这个动作告诉车辆在当前位置保持停止。

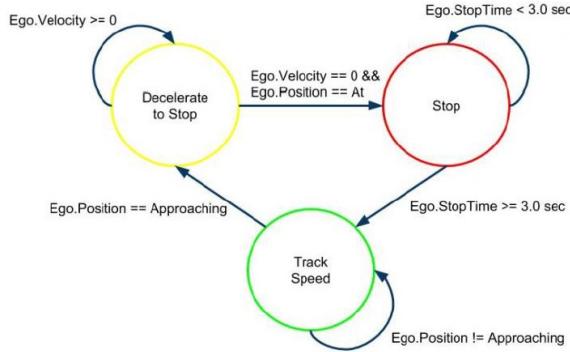
进入行动 entry action 是启动计时器 timer 等待一个固定的时间量，然后继续通过交叉口。我们现在将看看在这个场景中，ego 载体将遇到的不同情况，并找出编码正确的行为规划解决方案所需的有限状态机元素。让我们先看看 ego 车辆进入交叉口区域之前的情况，在交叉口区域中，它有一个单一的约束条件来遵循道路的速度限制。此约束是基于“轨迹速度”状态的输入动作设置的。当车辆进入接近区、红色区时，必须开始减速至停车标志，从而过渡到减速至停车状态。因此，从轨道速度到减速再到停车的过渡条件就是进入接近区。



然后，一旦减速，自动驾驶车辆必须执行的下一个操作是在停车线之前或在交叉口的 at zone 完全停车。为确保发生这种情况，车辆保持在减速至停止状态，直到其速度为零且位置在 at 区域内。减速至停止状态的进入动作 entry action 是建立安全停止位置。由于该场景的简单性，这是一个 single stop location，即高清路线图提供给规划器的停车线。然而，我们将在下一课中看到，一旦其他动态对象与自我载体交互，这将变得更难做到。



一旦完全停止，汽车进入停止状态。作为进入动作，计时器将启动，以确保车辆在按照典型驾驶规则继续行驶前保持停止状态 3 秒钟。一旦计时器完成，规划器自动转换到轨道速度状态，并沿着任务规划器提供的路线通过交叉口，在左侧、右侧或直行。



这就是用有限状态机处理简单的四向交集所需的全部计算。

对状态机性能有很大影响的一个特殊问题是输入中的**噪声**问题。上面定义的状态转换条件是精确的，并且依赖于车辆到达停止点并精确实现零速度。但是车辆状态的定位估计也可能包含噪声，不能完全满足这些条件。为了处理这种类型的输入噪声，引入**噪声阈值超参数 hyperparameter**。这是一个很小的阈值，允许在停止时接受接近零的速度(Ego.Velocity ≥ 0 改为 Ego.Velocity $\geq \text{StopThreshold}$)。在下一课中，当处理动态对象的更复杂场景时，我们将继续越来越多地看到这些超参数。

既然我们已经完成了有限状态机的创建，我们如何测试它是否工作？有四个阶段的测试需要确认一个行为规划系统的功能，这是我们在课程 1 中讨论的整车安全评估。

首先，我们执行**基于代码的测试 code based tests**，以确认代码的逻辑是正确的。例如，基于代码的测试可以告诉程序员路线图中设置的速度限制是否是有限状态机产生的速度限制。但是，这些检查无法确认状态转换是否正确，或者状态是否能够处理给定场景中的所有情况。为此，我们必须查看程序代码是否按预期正确处理所有情况。

接下来，我们进行**仿真测试 simulation tests**，它是在类似 Carla 的模拟环境中执行的，其中状态机执行它设计用来处理的场景。这种类型的测试能够确认状态机转换和状态覆盖是否正确。在模拟中执行的测试的数量应该代表所有可能的情况，当驱动场景捕捉程序员可能错过的任何边缘情况时，可以看到这些情况。

一旦确信状态机在模拟中的性能符合预期，我们就进入了**封闭轨道测试 private track tests**。这类测试测试在模拟中很难准确确定的特定场景，例如参数调整和噪声，以及真实环境中感知输出的错误。

最后，我们进行**道路验证测试 limited scoped close supervision road tests**。尽管之前的所有测试都是在高度受控的环境中进行的，但是道路测试可能非常不可预测，并且经常以工程师无法想象的方式破坏系统。然后可以将场景的新变化合并到测试过程的早期阶段。

Lesson 3: Handling an Intersection Scenario with Dynamic Objects

在上一个视频中，我们使用有限状态机开发了一个基本的行为规划器，用于无交通的四向交叉口。我们现在关注的是在同一个四向交叉口合并交通交互。

同样，我们在本视频中研究的场景是一个单车道四向停车交叉口。我们希望能够以任何可能的方向通过这个十字路口，同时处理与其他车辆的互动。在本课程中，我们将重点讨论与其他动态对象的交互，特别是与其他车辆的交互。

为了能够安全地处理这些车辆，一个重要的方面是测量与其他物体的各种相互作用点之间的距离。要考虑的第一个距离度量是**到动态对象的距离 distance to dynamic object**。这被定义为当前自我位置到环境中任何动态对象中心之间的欧氏距离。第二个是**到碰撞点的距离 distance to collision point**，即到另一个动态对象的潜在碰撞点的距离。最后是**到达给定碰撞点所需的时间 time to collision (TTC)**。



我们的计算方法在之前的视频中讨论过。现在让我们扩展一个有限状态机来适应动态对象的额外复杂性。我们需要将所需的机动次数增加到 4 次，以便在这种情况下正确处理与车辆的所有交互。我们需要添加的新策略是**跟随前导车 follow leader**。这种机动状态要求 ego 车辆跟随车辆的速度，并与前方

车道上的任何车辆保持安全距离。安全距离取决于速度，安全速度和安全距离在每次迭代的行为规划器中都会更新为进入动作。

现在我们已经为扩展的有限状态机定义了所有的状态，我们开始填充转换。在轨道速度状态下，ego 车辆位于交叉口外。正如我们在上一课中所看到的，ego 车辆只有在接近区域才会进入减速停止状态。但是，还有一个问题，即另一辆车出现或进入车辆前方车道的情况。在这种情况下，我们将通过执行跟车检查 follow check 来尝试跟车。以下检查可分解为两个元素：**距离检查 distance check** 和**同一车道检查 same lane check**。

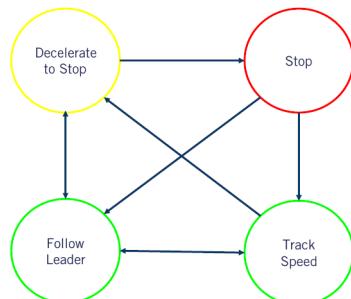
距离检查确认车辆距离足够近，因此应该跟随它。同一车道检查确认车辆实际上与 ego 车辆在同一车道上。在这种情况下，我们可以使用到动态对象的距离，即到对象的欧氏距离，并将其与阈值进行比较。有几种方法可用于检查车辆是否与 ego 车辆在同一车道上。然而，为了简单起见，我们将检查引导车辆是否在车道限制内，以及动态对象的前进方向是否在 ego 车辆的给定阈值内，以指示其在同一方向上行驶。

在 follow leaders 状态中，我们只定义了两个转换。首先，引导车辆可能在 ego 车辆接近或在交叉口前退出车道，导致切换到轨道速度状态。或者引导车辆驶出当前车道，通常在 ego 车辆已经处于 at 或 approaching 区域时驶入交叉口，导致直接过渡到减速至停止状态。我们添加了转换条件，即当 ego 车辆处于 approaching 或 at 区域时，碰撞距离大于到停止点的距离。在这种情况下，ego 车辆应过渡到减速停车，并将停车点设置到交叉口停车线的正确位置。与之前一样，在减速至停止状态下，ego 车辆继续减速，直到在停车线处停车。这将导致切换到停止状态，如上一课所示。

然而，我们必须再次考虑到，车辆能够在交叉口从车道或超车机动车道驶向自动驾驶汽车前面。和以前一样，当 ego 车辆处于接近或添加区域时，我们在减速至停止状态的整个过程中执行跟随检查，如果检查结果为真，则转换为跟随前车。然而，在这种情况下，距离检查会考虑到前方车辆到碰撞的距离是否小于停车点距离，因此应优先考虑前方跟随模式。

我们现在可以定义停止 stop 状态的转换。将有两组转换。跟车或跟踪速度，取决于通过交叉口的车道内是否有满足跟车检查的车辆。在每种情况下，过渡条件都会根据任务计划要求的通过交叉口的路径而变化。当 ego 车辆需要左转时，任何从左侧、右侧接近的车辆或任何迎面而来的车辆必须在 ego 车辆行驶前驶离交叉口区域。当车辆需要直行时，只有从左侧或右侧接近的车辆才需要驶离交叉口。最后，当 ego 车辆需要右转时，只有从左侧接近的车辆需要驶离交叉口。从停止状态转换的这个简单定义中，我们希望您能够直观地看到由于交叉口的到达顺序以及其他车辆也可能左转或右转而需要进行的额外检查，以处理优先级。

生成的完整状态机包括以下状态和转换。



虽然由于我们所依赖的过于简单化，这种状态机无法用于真正的自主车辆，但它很好地演示了将给定

的操作设计域转换为功能有限状态机的过程。

到目前为止，在创建状态机的过程中，我们对动态对象的行为做出了一个特别有力的假设。也就是说，**所有动态对象都遵守道路规则**。然而，情况并非总是如此，这种困难导致许多边缘情况也需要考虑。这并不是所有可能性的完整集合，行为安全评估和测试的主要目标之一是尽可能多地揭示意外行为的变化，以便它们也可以被检测、分类并纳入行为规划设计过程。然而，仅仅揭露极端例子 **edge case** 是不够的。相反，我们必须定义自动驾驶汽车应该如何应对每一种情况。为此，需要紧急机动，如急转弯或急刹车，需要更多的过渡和条件来确定。

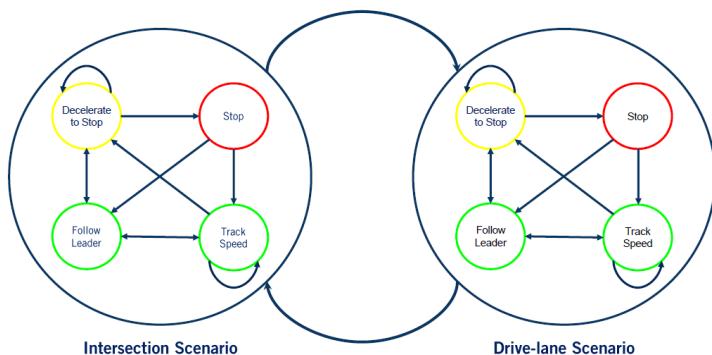
Lesson 4: Handling Multiple Scenarios

到目前为止，在这个模块中，我们已经开发了一个有限状态机行为规划器，它可以处理一个带有动态对象的四路交叉口场景。然而，在驾驶过程中，会遇到很多这样的情况。例如，有三向停车交叉口、红绿灯控制交叉口和直行道路场景，仅举几个例子。虽然这些场景中的许多都有一些相似之处，主要是因为处理它们所需的逻辑，但可以认为每个场景都需要根本不同的驾驶行为。

处理这些多个场景的一种方法是在不断扩展的行为网络中，向单个状态机 **single state machine** 添加额外的状态、转换和条件。然而，这种方法有几个问题。第一个问题是，我们需要开发额外的逻辑来区分当前场景，以及单独处理每个场景所需的规则。这很快就会遇到一个称为“**规则爆炸 rule explosion**”的问题，在这个问题中，添加其他场景所需的规则数量变得难以想象。单一状态机的第二个问题是，由于每一步需要评估的额外规则和转换的数量，所需的**计算时间显著增加 increase in computational time**，最后，由于必须担心的案例数量迅速增加，这样一个系统的**创建和维护非常复杂 complicated to create and maintain**。**复用性差**，几乎不可能在多个项目中使用相同的 FSM。

然而，我们可以使用另一种方法。我们可以将每个高级场景表示为单个状态。这将允许我们在高级场景状态之间创建更简单的转换。

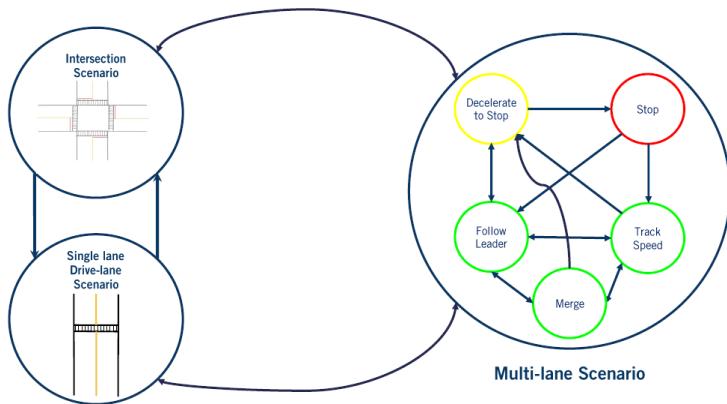
在本例中，我们可以看到这些状态表示一个直线道路场景和我们之前处理的四路交叉口场景。对于这些高级场景状态中的每一个，我们将存储一个低级状态机，我们可以用它独立地处理场景。例如，对于交叉点场景，低级状态机 **low-level state machine** 将是在上两节课中构建的状态机。这种表示方法称为**层次状态机 hierarchical state machine**，**超级状态 super-states** 表示每个场景，**子状态 sub-states** 表示每个场景中要处理的机动。高级场景状态机之间的转换将是一个规则，它根据 HD 路线图和动态车辆信息定义何时进入新场景。



假设我们有一个简单的分层状态机，有两个场景，一个用于有人行横道的直行道路，另一个用于基于停车标志的交叉口。要从直行道路行驶到停车标志交叉口，自动驾驶车辆必须靠近与该交叉口的交叉口。这是通过沿任务规划到下一个十字路口的距离检查来完成的。我们在这里讨论的所有运动都是在超级状态的层面上。

我们要回答的下一个问题是，在处理子状态机动时，我们如何在场景之间切换？一种方法是在当前场景超级状态之外引入关键机动子状态的转换。让我们继续以交叉口场景为例。首先，让我们确定哪个状态将是场景的关键退出状态。我们能够驶出交叉口的唯一方法是在通过交叉口后 track speed 或 follow leader 状态。所以让我们把出口转换 exit transition 放在那里。转换的条件是确认这个场景确实已经完成。在本例中，我们可以使用到下一个停车标志交叉口的距离。如果这大于给定的阈值，我们可以退出交叉口场景。有了这种转换方法，我们还能够在场景切换之间保持机动。

在这种情况下，交叉口超级状态下的轨道速度将连接到我们进入的下一个场景的轨道速度。通过创建一个新的超级状态场景节点，以及一个能够处理该场景的子状态机，可以向这个分层状态机添加一个额外的多通道场景。



层次状态机有一些固有的优点和缺点。这种方法的优势在于它的细分性质。通过在每个超状态中创建单独的状态机，这种方法能够通过重构子空间来限制任何一个时间步所需的计算时间。这种细分的特性也使得层次状态机的测试和编程变得非常容易。

然而，层次化状态机也并非没有问题，在限制规则数量的同时，这种方法仍然无法完全处理规则爆炸问题。产生大量规则的一个原因是所有场景都有完全分离的状态机，因此许多规则彼此重复，以便在不同场景中处理相似的行为。最终，分层方法允许系统设计者处理比平面单层有限状态机更大的复杂性。只要能在每一层定义关键状态，这个思想就可以扩展到多个层。实际上，由有限状态机强制执行的观点，即单个行为在任何给定时间都处于活动状态，并且可以显式定义到其他行为的所有转换，对它可以处理的场景集的大小有限制。然而，它为自动驾驶汽车的典型行为规划提供了一个有用的工具。

Lesson 5: Advanced Methods for Behaviour Planning

让我们从确定状态机模型的问题开始。在本模块的整个开发过程中，我们发现的第一个问题是规则爆炸问题。这意味着，随着我们开发一套更完整的场景和策略来处理所需的规则数量会迅速增长。这一限制意味着，虽然可以开发一个有限状态机行为规划器来处理有限的操作设计领域。用有限状态机几乎不可能开发出一个完整的 L4/L5 能力的车辆。

处理噪音环境是下一个问题。在第二和第三课中，我们看到了如何添加超参数来处理一些噪音。这种

类型的噪声处理只能处理一些有限的情况。为了处理所有类型的输入不确定性，需要采用不同的方法。在这方面，我想指出的下一个问题是超参数的实际调整 **hyperparameter tuning**。随着所需行为变得越来越复杂，用于离散化环境和处理一些低噪声的超参数数量迅速增加。所有这些超参数都必须非常仔细地调整，这可能是一个漫长的过程。

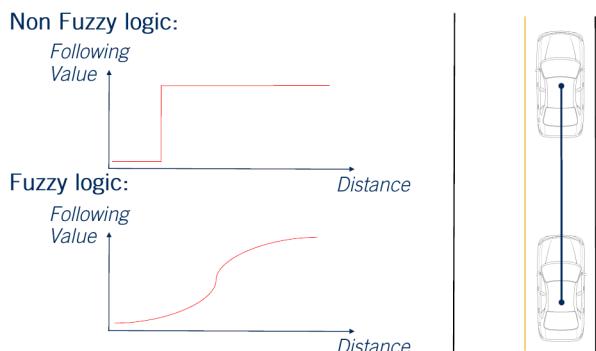
最后，我们将讨论的最后一个问题是**如何处理未出现的场景**。由于这种方法的程序性质，很可能会出现这样一种情况，即系统的程序逻辑将以不正确或意外的方式做出反应。基于状态机的方法被归类为专家系统 **experts system**（一种模拟人类专家解决领域问题的计算机程序系统）。

然而，状态机并不是唯一被设计出来的专家系统。还有一些方法依赖于在每个时间步应用于输入数据的规则数据库 **rule-based behaviour planner**。这些规则形成了一个层次结构，其中安全关键规则优先，优先于舒适性规则或防御性驾驶规则。**每一个规则只有在适当的条件出现时才生效，以影响输出机动的选择**。这样的规则数据库不会遇到需要复制我们在层次状态机中遇到的规则的问题。因为规则可以适用于整个 ODD (operational design domain) 或超过它的重要部分。

然而，基于规则的系统必须非常谨慎地制定规则，使它们不会相互产生负面影响。或在同时激活多个规则时导致意外结果。由于普遍依赖专家用户为所有可能的场景进行设计，这两种类型的专家系统都面临上述相同的问题。

对于上面提到的一些问题，一个解决方案是扩展人类专家类型规划或合并**模糊逻辑 fuzzy logic**。模糊逻辑是一个系统，通过一组清晰的 **crisp**、定义良好的 **well-defined** 值来创建一组更连续的模糊状态。

对于一个基于模糊的系统如何工作的简单例子，让我们看一看车辆跟随行为的例子。而之前我们设置了一个参数化的距离，将空间分为跟车和不跟车。有了模糊系统，我们就可以有一个连续的空间，在这个空间上可以应用不同的规则。例如，当一个模糊系统非常接近它时，它可能会对一辆领先的汽车做出强烈的反应。但在距离较远时，不要太在意速度匹配或距离跟踪要求。

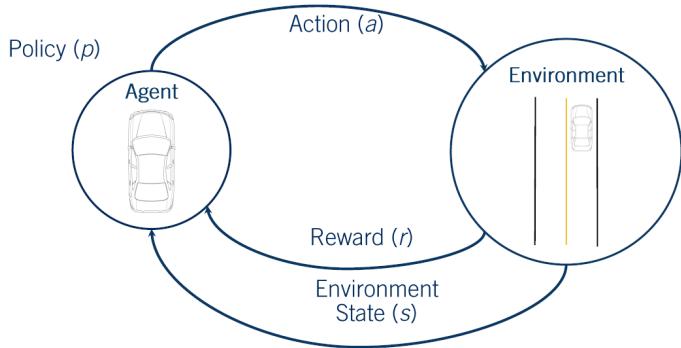


而基于模糊的规则系统比传统的离散系统更能处理系统的环境噪声。

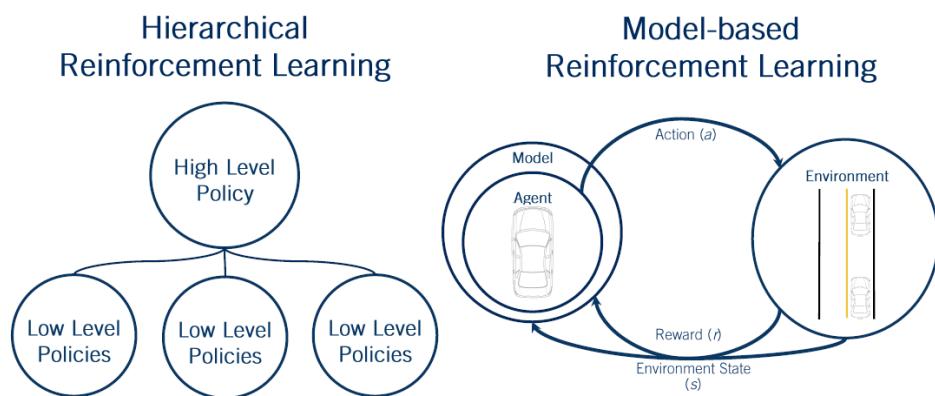
规则爆炸 rule explosion 和 **超参数调整 hyperparameter tuning** 仍然是模糊系统的问题。事实上，**模糊系统会导致更大程度的规则爆炸**，因为处理模糊输入集需要更多的逻辑。伴随着规则爆炸，超参数调整也面临着巨大的挑战。

我想讨论的下一个令人兴奋的研究方法是在行为规划中使用**强化学习**。强化学习是机器学习的一种形式，在这种学习中，智能体通过采取行动和获得持续的奖励来学习如何与给定的环境进行交互。代理首先参与模拟环境的游戏。这意味着在一开始，代理将不擅长于特定的任务，然而，随着时间的推移，

代理最大化其回报，他们最终学会掌握任务。通过在模拟环境中学习，在学习过程中经历的许多失败中，不会产生真实世界的影响。更具体地说，我们可以说，代理正在尝试学习一个策略，用 p 表示，它将给定的环境（用 s 表示）映射到给定的动作（用 a 表示）。要将强化学习的概念与行为规划联系起来，我们可以说，我们正在尝试教自主车辆如何跟随引导车辆。在这种情况下，我们的政策 p 将尝试进行车辆跟踪。我们的行为可以显式地定义为要执行的行为，也可以通过选择期望的加速度和转向率来隐式地定义行为。环境可以用一组连续的变量来表示，这些变量告诉我们相关的信息，比如到所有物体的距离和我们的任务路径。最后，奖赏函数可以基于最优跟车距离。在你喜欢的距离给予最高的奖励，惩罚离得太近比离得太远更重。



由于自主车辆可能会遇到各种各样的场景和输入，直接强化学习用于行为规划是不可能成功的。相反，通常会进行一些进一步的调整。一个这样的适应被称为分层强化学习 *hierarchical RL*。在这里，我们将问题分为机动空间中的低级策略和场景中的高级策略。这类似于层次有限状态机。然后，每个低层策略被独立地学习，只有成功地学习了一个高层策略，才能完成一个场景。第二种常用的技术叫做基于模型的强化学习 *model-based RL*。在基于模型的强化学习中，不仅是 agent 试图学习策略 p ，而且是 agent 周围当前环境的模型。这种方法如何应用于自动驾驶汽车的行为规划的一个例子是包含动态对象的运动模型。如果代理了解动态对象的运动模式，它可以通过环境创建更有效的计划。



虽然强化学习是一个非常令人兴奋和非常有前途的研究领域，但它也并非没有自身的局限性。许多用于学习自动驾驶所需策略的**模拟环境过于简化**。由于它们的简单性，所学的策略可能无法转移到现实世界的环境中。过于逼真的模拟器会导致严重的计算需求问题。尤其是在进行数千次重复的大范围变化的自驾学习场景时。第二个是**安全问题**。而强化学习中有一些技巧，试图确保强化学习者创建的轨迹上的安全约束。仍然没有办法对一个学习系统进行严格的安全评估，因为就决策的方式而言，它们大多是黑匣子。

许多其他有趣的想法也正在研究界进行探索，其中许多试图从人类驾驶行为中学习。例如，在逆强化学习中，方法是使用人类驾驶数据作为策略，而不是试图获得给定奖励函数的策略。试图学习人类使用的奖励功能。一旦学习了奖励函数，算法就可以执行类似于人类驾驶员的驾驶操作。最后，端到端方法将原始传感器数据作为输入，并尝试输出油门、制动和转向命令。通过学习，再次，从人类驾驶命令的模仿学习方法。Nvidia 的研究人员在他们的论文 End-to-End Learning for auto-Driving Cars 中率先提出了这种方法。虽然这没有明确地归类为行为规划，但是端到端方法仍然隐式地执行作为其输出选择过程的一部分的行为选择任务。这些简短的解释仅仅触及了一个巨大而迅速发展的研究领域的表面。行为规划一直是自动驾驶研究的热点之一。同时也是实现现实世界 L5 的最大瓶颈之一。

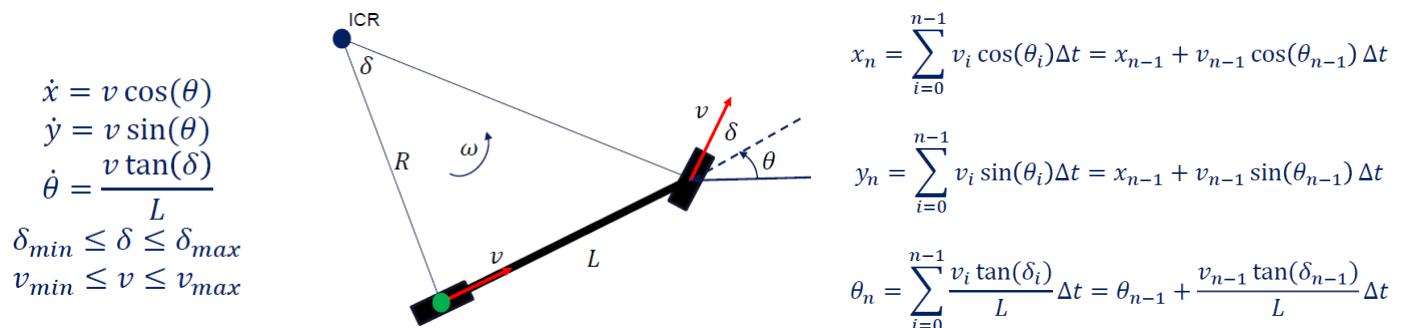
- J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi, "[A behavioral planning framework for autonomous driving](#)," 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014. This gives a nice overview of an example framework that can be used in behaviour planning.
- R. S. Sutton and A. G. Barto, [Reinforcement learning an introduction](#). Cambridge: A Bradford Book, 1998. Gives a great introduction to reinforcement learning concepts.

Lesson 1: Trajectory Propagation

反应式运动规划器 **reactive motion planner** 是一种从机器人周围环境中获取局部信息以生成无碰撞的轨迹并向某个目标位置前进的规划器。在本模块中，我们将坚持静态环境，作为规划自动驾驶汽车行为和路径的第一步。

首先，我们要做一个简短的回顾什么是运动学模型。运动学模型给出了我们的机器人的运动方程，同时忽略了质量和惯性对其运动的影响。我们可以将其与动态模型进行对比，动态模型考虑了质量和惯性，但代价是更加复杂。运动学模型侧重于线速度和角速度，偶尔，它们的导数作为输入，而动力学模型侧重于力和扭矩作为输入。为了说明这一点，我们有一个运动学粒子模型，与这里显示的带摩擦的动力学粒子模型进行了对比。在路径规划和轨迹优化中，我们经常关注运动学模型，以使运动规划问题更易于计算，并将动力学简化引起的问题留给控制器。

通过关注离散模型，它使我们能够轻松有效地传播给定控制输入序列的轨迹。这些方程离散化的结果是，我们可以递归 **recursive** 地实现序列中所有更新的和，这允许我们为给定的输入序列迭代地建立完整的轨迹。这节省了计算工作量，因为我们可以仅使用先前计算的点增量计算轨迹的下一点，而不是重新计算轨迹中每个点的所有先前更新的总和。



这不仅对轨迹规划很有用，也对运动预测也很有用，因为我们知道驾驶场景中不同代理的运动学模型，并且我们对它们将接受的控制输入有一个有根据的猜测。由此，我们可以估计它们未来的运动轨迹，从而帮助我们规划运动以避免碰撞。作为一个例子，假设我们采取一个恒定的速度输入以及恒定的转向角，这将导致机器人以恒定的速度穿过弧线，同时它的航向随着每个时间步缓慢变化。

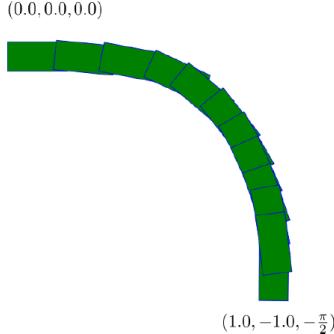
如果我们根据某些转向功能沿路径改变转向角，我们可以执行更复杂的机动，这对于避障等任务至关重要。这本质上是局部规划问题的关键，计算安全导航到给定目标点所需的控制输入。

Lesson 2: Collision Checking

在运动规划中，碰撞检查的核心是确保给定的轨迹或规划的路径在被物体穿过时不会与沿途的任何障碍物发生碰撞。碰撞检查的一个重要方面是，它是一个计算密集型 computationally intensive 问题，存在于许多领域，包括自动驾驶和其他机器人应用、游戏、三维动画和工程设计。保证一条安全无碰撞的最优路径不仅需要关于环境的完美信息，而且还需要大量的计算能力来精确计算，特别是对于复杂的形状和详细的环境。对于需要实时规划的自动驾驶，我们没有这两个功能。正如模块 2 所述，占用网格提供给我们的信息是一个不完美的估计，这意味着我们需要在碰撞检查算法中添加缓冲区 buffer，以使它们更具容错性 error tolerant。

确切地说，碰撞检查相当于沿着给定路径的每个点旋转和平移车辆的足迹。足迹中的每个点都由每个路径点的朝向旋转，并由同一路径点的位置平移。这个方法由集合 S 总结，其中 p 是由一组点 p 和 F 组成的路径，F 表示一个函数，该函数返回足迹中包含的设定点，该足迹由 θ 旋转，并由 x 和 y 平移。对路径上的每个点执行平移和旋转操作后，将通过每个足迹的并集来给出沿路径的汽车线束 swath。

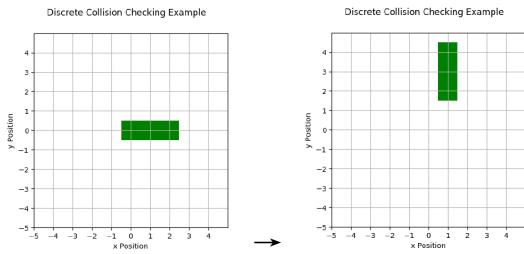
- Area occupied by car along path generated by rotating the car's footprint by each x, y, θ along the path
- Swath along path is the union of each rotated and translated footprint
- Swath can then be checked for collisions



$$S = \cup_{p \in P} F(x(p), y(p), \theta(p))$$

然后我们检查整个装置，看看里面是否有障碍物。如果有，则路径包含碰撞，否则无碰撞。

我们现在需要以抽象的形式进行 swath 计算，并将其转换为可由计算机计算的离散公式。假设我们有一个离散的代表性的汽车足迹和路径的占用网格，这是在模块二讨论。汽车足迹包含 K 个点，路径的端点很长。在算法上，计算 swath 需要我们旋转和平移汽车足迹中的所有 K 点，并为路径中的每个点乘以 1。让我们看一个使用占用网格的旋转和平移的具体示例。假设我们的占用网格的分辨率为一米，我们的足迹占用以下三个网格点：(0, 0), (0, 1), (0, 2)。假设我们有一个路径点，我们想旋转和平移到(1, 2, $\pi/2$)。应该是先旋转 $\pi/2$ ，再平移(1, 2)。



这个转换步骤的顺序很重要。如果我们先平移，然后围绕原点旋转，我们会得到一个不正确的位置变换点。为了得到实际的占用网格索引 index，我们用 X 和 Y 来抵消 x 和 y 点，X 和 Y 分别是占用网格的 x 和 y 维度的大小。然后我们可以将其除以网格分辨率 δ ，得到占用网格中的相关 index。然后我们将这些点添加到一个集合数据结构中。

$$x_i = \frac{x(p) + \frac{X}{2}}{\delta} \quad y_i = \frac{y(p) + \frac{Y}{2}}{\delta} \quad S = S \cup (x_i, y_i)$$

因为我们要维护一个集合，所以我们要确保在我们的线束中没有重复的点。可以想象，随着问题的扩展，这种计算变得相当昂贵，这使得在执行实时规划时很难使用。此外，由于我们的信息不完善，在计算 swath 时使用精确的足迹可能是危险的，因为在障碍物位置没有错误缓冲区。

由于这种基于 swath 的方法通常计算量大，因此当我们在运动规划算法中使用大量重复时，它通常很有用，如在晶格规划器中。由于我们在晶格规划器的每一步都被约束到一组小的控制动作，因此我们也被约束到一组小的线束，因此它们的并集可以离线预先计算。在线执行冲突检查时，问题归结为多个数组查找。

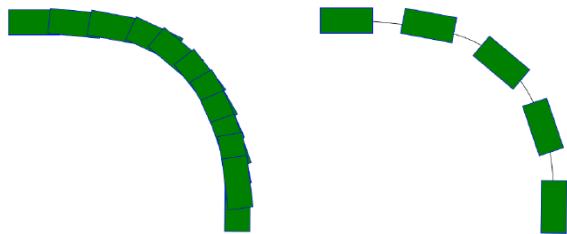
为了减轻不完全信息和大量计算需求的挑战，我们经常使用**保守近似 conservative approximation** 来进行碰撞检查，牺牲最优性来提高速度和鲁棒性。在这种情况下，最优性是通过选择一个合适的目标函数来定义的。由于我们切换到近似碰撞检查以获得计算性能，因此必须使用过于保守的近似。选择一个特定的近似值的目的是找到一个在不影响安全性的情况下提供极大算法加速的近似值，同时也最小化我们的轨迹变得次优的程度。

碰撞检查的保守近似方法是，即使路径上没有碰撞，也可能报告沿路径的碰撞，但如果确实发生碰撞，则永远不会遗漏碰撞报告。这意味着碰撞检查器可能包含一些假阳性 false positive 碰撞，但不会包含假阴性 false negative 碰撞。这为我们的汽车提供了一个很好的缓冲区，有助于缓解提供给我们的不完善信息带来的问题。这种圆近似对于某些算法很有用，因为检查点是否位于圆内的计算成本很低。我们所要做的就是检查障碍物的任何一点到圆心的距离是否小于圆的半径。

关于保守近似，有一点需要认识到，即使存在路径，它们也可以消除问题中所有可行的无碰撞路径，或者消除通过狭窄开口的安全通道。这可能会导致规划者在不应该的时候陷入困境，或者会导致规划者计算出比必要的更迂回的路线。根据您使用的运动规划算法和占用网格的结构，不同的碰撞检查算法将比其他算法更有效。它将由你作为一个自动驾驶工程师来决定哪种算法最适合你的应用。

在所有这些计算中需要注意的一点是，它们都是离散形式的，因为我们在计算机上执行这些计算。因

此，我们的碰撞检查精度 collision checking accuracy 也受到执行离散化时选择的分辨率的影响。为了说明这一点，假设我们有相同的路径和足迹，但假设一个线束的计算分辨率比另一个要粗得多。



我们可以看到，较粗的分辨率会导致我们的线束中存在较大的间隙，如果这些位置存在障碍物，则会导致错误。我们为碰撞检查选择的分辨率越精细，它就越精确。然而，更高的分辨率也会带来计算成本。所以，作为一个自动驾驶工程师，你需要在精确度和计算速度之间找到一个平衡点。有许多优秀的碰撞检查库可用。

Lesson 3: Trajectory Rollout Algorithm

我们将结合前面两节课中获得的一些知识，开发一种称为轨迹展开规划器 trajectory rollout planner 的反应式运动规划算法 reactive motion planning。

轨迹展开算法包括**轨迹传播 trajectory propagation** 步骤、**碰撞检查 collision checking** 步骤和**路径选择 path selection** 步骤，以实现期望的目标状态。在较高层次上，轨迹展开规划器使用第一课中讨论的轨迹传播方法来生成一组候选轨迹，机器人可以从其工作空间中的当前点开始跟踪这些候选轨迹。然后，我们获取机器人的局部障碍信息，确定哪些路径是无碰撞路径，哪些路径不是无碰撞路径。在这些无碰撞路径中，我们选择一个目标函数最大化的路径，其中包括一个奖励朝着目标进展的条件。通过反复执行此操作，我们最终得到了一个**滚动时域规划器 receding horizon planner**，它对环境做出反应，同时朝着目标稳步前进。

算法的第一步是在每个时间步生成一组轨迹。对于轨迹展开，每个轨迹将对应于在恒定时间范围 constant time horizon 内为机器人应用多个步骤的固定输入。我们在可用输入值范围内对这些固定输入进行统一采样，以生成各种潜在的候选轨迹。通过在输入频谱中获得各种各样的轨迹，我们可以提高轨迹搜索的质量和机动性，因为我们正在探索一组更广泛的机器人候选路径。

如果我们只使用**小范围的输入**，那么我们的计算时间就会提高。但是在我们的计算中可能会遗漏一些潜在的候选轨迹，这可能会降低规划器生成的轨迹的质量。

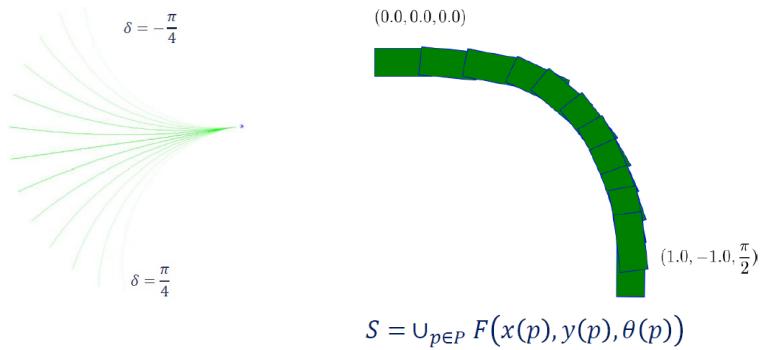
然而，**采样太多**的候选轨迹意味着我们在每一步都有额外的计算开销，因为每个额外的轨迹都需要生成、检查碰撞和评分。

一旦我们选择了一组输入，我们就需要通过使用我们在第一课中讨论的车辆运动学模型向前传播状态，沿着轨迹生成未来的状态。回想一下，对于自行车模型，两个输入是转向角的速度。如果我们保持速度不变，但是改变整个范围内的转向角，从 $-\pi/4$ 到 $\pi/4$ ，我们现在有一组弧作为候选轨迹。这些弧是通过递归计算运动学方程生成的，如我们在第一课中所讨论的。

$$x_n = \sum_{i=0}^{n-1} v_i \cos(\theta_i) \Delta t = x_{n-1} + v_{n-1} \cos(\theta_{n-1}) \Delta t$$

$$y_n = \sum_{i=0}^{n-1} v_i \sin(\theta_i) \Delta t = y_{n-1} + v_{n-1} \sin(\theta_{n-1}) \Delta t$$

$$\theta_n = \sum_{i=0}^{n-1} \frac{v_i \tan(\delta_i)}{L} \Delta t = \theta_{n-1} + \frac{v_i \tan(\delta_i)}{L} \Delta t$$



现在我们有了弧的集合，我们可以检查哪些弧是无碰撞的。对于碰撞检查算法，我们假设给定了一个表示车辆工作空间离散化的占用网格。这种离散化将以矩阵的形式存储，其中矩阵的每个值将表示工作空间中的相应位置是否被占用。然后，我们可以使用前面视频中讨论的基于 swath 的方法执行碰撞检查。回想一下，我们通过沿着路径扫过机器人的身体，并在给定轨迹的每个时间步取所有足迹的并集来生成线束。汽车的足迹将对应于占用率网格中的一组指标。因此，沿路径旋转和平移的每个点也将对应于占用网格中的不同索引。这些索引将存储在一组数据结构中，以消除重复。然后，我们可以检查线束的每个点，以查看线束的哪些点与占用网格的占用元素重叠。我们通过遍历 swath 集合中的每个点并检查占用网格中的相关索引来实现这一点。如果线束中的任何一点被占用，则该轨迹包含碰撞。

一旦我们遍历了上一步中生成的每个轨迹，我们将得到一组无碰撞的运动学上可行的轨迹，然后我们可以使用我们的目标函数进行评分。每一个目标函数所需要的基本元素是奖励朝着某个目标点或区域前进的某种方式，这是我们运动规划问题的最终目标。一个简单而有效的方法是在目标函数中有一个与候选轨迹末端到目标节点的距离成比例的项。然而，有时我们也会希望在我们的反应式计划中鼓励其他行为。正如我们在模块一中看到的，一些目标函数的例子包括最小化与车道中心线的距离，我们正在惩罚路径的曲率。有时，我们还希望奖励路径，使距离最近的障碍物的距离最大化，以使可行路径的灵活性最大化，以供规划器中的未来时间步使用。正如我们在第一单元中所讨论的，没有完美的目标函数，您需要构建自己的目标函数以适合您的应用程序。对于我们的反应式规划器，我们将使用到目标的距离作为最小化的目标函数。

$$J = \alpha_1 \|x_n - x_{goal}\| + \alpha_2 \sum_{i=1}^n \kappa_i^2 + \alpha_3 \sum_{i=1}^n \|x_i - P_{center}(x_i)\|$$

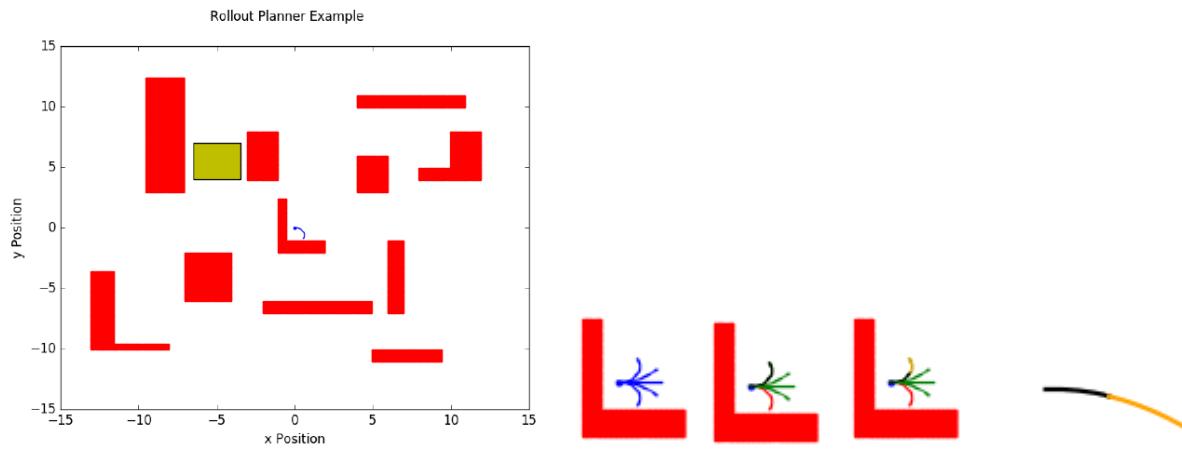
Distance to goal Curvature Deviation from centerline
 $J = \|x_n - x_{goal}\|$

一旦我们有了目标函数，我们就可以在无碰撞的轨迹上迭代，然后选择使目标函数最大化的路径，或者根据您如何制定目标使惩罚最小化的路径。

让我们看一个具体的例子，把整个算法放在一起。如图所示，占用率网格中的障碍物用红色表示，目标区域用黄色表示，汽车的初始点为(0, 0, 0)。假设我们的转向角范围在- π/4 到 π/4，步长为 π/8，假设我们使用的是 0.5 米每秒的恒定速度。另外，假设我们的规划器使用 0.1 秒的时间步长，每个规划轨迹总共持续 2 秒。

如果应用轨迹传播算法，我们概述了在第一课使用我们的自行车运动学模型，然后我们可以生成一组路径为每个选定的转向角在我们的转向范围。第一条轨道的转向角是负的，因此，我们可以看到轨道

向右弯曲。我们的下一个轨迹的转向角是负的，你可以看到，生成的轨迹的曲率比第一个小。下一步，我们有零转向角轨迹，这导致汽车在直线上向前行驶。正转向角轨迹与负转向角轨迹对称，如预期的那样，将汽车转向左侧。现在我们有了一组候选的轨迹，我们需要使用上一个视频中概述的冲突检查算法来检查每一个轨迹，看看它们是否没有冲突，在本视频的前面已经讨论过。在沿着每条轨迹平移和旋转脚印之后，我们检查结果线束中的每个占用栅格索引，以查看是否存在障碍物。如果任何一个索引包含一个障碍物，那么该路径被标记为有碰撞，我们用红色表示。所有无碰撞路径都是绿色的，然后我们可以使用我们的目标函数对其进行评估，以找到最佳路径。回想一下，我们的目标函数是到目标的距离。使到目标的距离最小化的路径现在被着色为黑色。这就完成了我们的第一个计划迭代。在这一点上，我们现在有了一个车辆执行的轨迹。



然而，在下一个规划周期之前，我们不会完全执行这个轨迹。相反，车辆将只执行循环的前几个点。具体数字取决于规划频率，我们的规划范围将根据我们的进展而向前移动。这正是您在本专业的第一门课程中应用于车辆控制的“滚动时域 receding horizon”方法。

把这一切结合起来，在每一个时间步，我们规划一个两秒钟的轨迹，但每次只执行一秒钟。通过这样做，我们的规划周期的结束时间为每个规划周期向前移动 1 秒。这被称为“滚动时域规划器”，因为在每个规划周期中，我们都具有一个固定的规划时间范围，该范围内的 n 次慢慢地向我们达到目标的时间点后退。这在这里说明，其中黑色是将在当前循环中执行的轨迹部分，橙色部分是剩余部分。一旦下一个计划周期开始，我们就再次重复整个过程。我们继续这个过程，直到我们计算出一条到达目标区域的轨迹，在每次迭代结束时进行检查。

现在您可以在我们的示例问题中看到规划器对目标区域采取的其余步骤。关于这个规划器，需要注意的一点是，它是短视的 **myopic**。也就是说，它没有规划一条直接通向目标的路径。取而代之的是，它贪婪地根据机器人离目标的距离来采样子路径。这会导致规划器目光短浅，陷入死胡同，通常会导致规划器找到次优路径。

然而，这种规划器大大降低了目标区域规划问题的复杂性，并且速度很快，可以作为在线规划器使用。

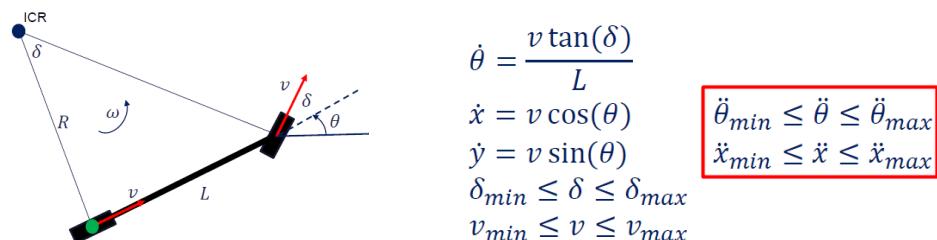
Lesson 4: Dynamic Windowing

在本视频中，我们将讨论如何使用称为**动态窗口 dynamic windowing** 的技术来增强上一课中开发的轨迹展开算法。动态窗口将允许我们在车辆轨迹上设置线性和角加速度约束，以便在车辆在规划周期

之间前进时提高舒适性。

首先，让我们重温第一课，我们讨论了自行车模型的运动学方程。基本上，自行车模型的两个输入是转向角的线速度，它随时间改变机器人的位置和方向。这整套运动学方程需要注意的一点是，**没有考虑高阶项，例如加速度或急动**。这些高阶项是导致车内乘客不适的原因，因此我们应该尝试在我们的运动学模型中解决这个问题。

我们可以限制所选输入以考虑快速变化对乘坐舒适性的影响。我们可以通过为自行车模型允许的线性和角加速度范围添加一个约束来实现这一点。这将限制在我们的车辆上的乘客在我们的车辆通过其计划的轨道时所受的力和扭矩。然而，这需要权衡。我们的运动规划器在每次规划迭代中都会失去一些**可操作性 maneuverability**。舒适性越高，可操作性越低。



具体地说，在添加这个角加速度约束之后，我们可能无法移动到转向角集的每个可能的转向角，因为它们可能会导致角加速度过高。此外，我们可能无法在规划迭代之间快速地提高或降低速度。

让我们关注角加速度约束，并导出由此产生的转向角 steering 约束。回想一下，自行车模型的角速度 $\theta dot = v \tan \delta / L$ 。因此，角加速度的大小大约是由开始和结束转向角的角速度之间的绝对差除以我们使用的时间步长得出的

$|\ddot{\theta}| = \left| \frac{\dot{\theta}_2 - \dot{\theta}_1}{\Delta t} \right|$ 。因为 v 和 L 总是正的，我们有这样一个要求：

$$|\tan(\delta_2) - \tan(\delta_1)| \leq \frac{\ddot{\theta}_{max} L \Delta t}{v}$$

为了说明这一点，我们将上节课中不允许的轨迹涂成红色。其余的绿色轨迹仍可用于我们在上一课中开发的反应式规划器的后续步骤。



这说明，一般来说，附加的约束会在一定程度上降低机器人的可操作性，同时促进更舒适的轨迹。约束集的限制性越强，机器人的可操作性就越差。我们还可以将类似的逻辑应用到线性加速度约束和一系列可用的线速度输入的情况，甚至可以将这两个约束同时应用到机器人。一般来说，动态窗口方法允许我们在规划过程中对轨迹如何演变进行更多限制，从而使运动更好地同时满足广泛的目标集。

- Fox, D.; Burgard, W.; Thrun, S. (1997). "The dynamic window approach to collision avoidance". Robotics & Automation Magazine, IEEE. 4 (1): 23–33. [doi:10.1109/100.580977](https://doi.org/10.1109/100.580977). This gives an overview of dynamic windowing and trajectory rollout.

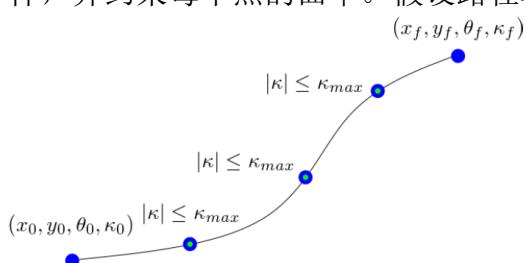
- M. Pivtoraiko, R. A. Knepper, and A. Kelly, “[Differentially constrained mobile robot motion planning in state lattices](#),” Journal of Field Robotics, vol. 26, no. 3, pp. 308–333, 2009. This paper is a great resource for generating state lattices under kinematic constraints.

Lesson 1: Parametric Curves

本地规划器是分层规划器的一部分，它以无冲突、高效和舒适的方式执行行为规划器请求的操作。这会产生一个轨迹，它是在给定时间内空间中的一系列点，每个点上具有所需速度剖面。作为开发的控制器的参考输入。

理解路径规划问题的第一步是首先理解其最基本的要求。对于路径规划问题，给定一个起始位置、朝向和曲率，找到一条满足运动学约束的终止位置、朝向和曲率的路径。在优化问题中，起始值和终止值可以表示为问题的**边界条件 boundary condition**，车辆的运动可以表示为对优化变量的连续时间约束。在这种情况下，**边界条件是必须在路径的任一端点上都成立**，给定的优化解才被认为是可行的。如果违反了这些边界条件，不管路径有多好，我们都没有达到我们真正想要达到的目标。这些边界条件将影响我们如何决定建立优化问题的基本结构。

对于我们的路径规划器，我们唯一的运动学约束是**限制沿路径的最大曲率**。一般来说，这是不容易满足的，因为在一条连续的路径上有无限多个点。相反，我们通常会在路径上的不同点处对曲率进行采样，并约束每个点的曲率。假设路径表现相对良好，这可能对应于满足约束的整个路径的曲率。



为了简化优化问题的表示，我们将路径定义为**参数曲线 parametric curve**。参数曲线是一种曲线，可以用一组具有特定参数的方程来描述。这些参数通常表示路径遍历，无论是通过弧长还是从 0 到 1 变化。例如，这里我们有一个三次样条参数方程组的 x 和 y 位置的路径。方程的参数 u 介于 0 到 1，当我们从路径的起点到终点行驶时。向量值 vector valued 函数 r 包含对应于给定 u 值的每个点的 x 和 y 位置。

$$\begin{aligned} r(u) = \langle x(u), y(u) \rangle \quad x(u) = \alpha_3 u^3 + \alpha_2 u^2 + \alpha_1 u + \alpha_0 \\ u \in [0,1] \quad y(u) = \beta_3 u^3 + \beta_2 u^2 + \beta_1 u + \beta_0 \end{aligned}$$

对于自动驾驶，我们通常（但并非总是）要求路径为参数曲线。因为规划方法根据 β_0 和 β_f 所示的边界条件、 α 所示的运动学约束和 f 所示的目标函数来优化给定路径。

$$\min f(r(u)) \text{ s.t. } \begin{cases} c(r(u)) \leq \alpha, & \forall u \in [0,1] \\ r(0) = \beta_0 \\ r(u_f) = \beta_f \end{cases} \quad (\text{s.t.} = \text{subject to, 使得...满足...})$$

路径的参数表示使得设置优化问题更简单，因为我们有一个函数，我们可以直接给我们的目标函数

objective functional f 。注意，术语 functional 指的是以函数为参数并返回实值的映射，因此它可以用来定义函数空间或参数化曲线上的代价 cost。我们可以将此参数曲线方法与模块 4 中的反应式规划器（使用非参数路径）进行对比。非参数路径指的是用空间中的一系列点而不是参数化的曲线来表示轨迹，因为我们遵循的曲线没有参数表示。

在自动驾驶领域，有两种常见的路径参数化。第一种是五次样条 **quintic splines**，是汽车 x 和 y 位置的五阶多项式函数。第二类是三次多项式螺旋 **cubic spirals**，由一个关于弧长的多项式曲率函数给出。这两个参数化曲线都给我们提供了满足我们刚才讨论的边界条件的方法，也给我们提供了在目标函数中使用的参数，以便根据我们的要求设计路径。选择其中任何一个都有相关的权衡。

首先，我们来讨论五次样条曲线。五次样条由两个方程给出，是关于 x, y 沿样条的级数。

$$x(u) = \alpha_5 u^5 + \alpha_4 u^4 + \alpha_3 u^3 + \alpha_2 u^2 + \alpha_1 u + \alpha_0$$

$$y(u) = \beta_5 u^5 + \beta_4 u^4 + \beta_3 u^3 + \beta_2 u^2 + \beta_1 u + \beta_0 \quad u \in [0, 1]$$

在这里我们可以看到五次样条有 12 个参数，6 个是关于 x 方程，6 个是关于 y 方程。这些参数对应于形成曲线形状的多项式系数。 $u=0$ 对应于路径的起点， $u=1$ 对应于路径的终点。五次样条的一个很好的性质是，对于给定的位置朝向和曲率边界条件，满足它们的样条系数有一个立即的闭式解。这个解决方案很长，所以我们不在这里列出，但是比起使用迭代优化方法生成路径，它仍然更便宜。还可以根据此应用程序进一步优化附加自由度。这是可取的。

五次样条曲线的缺点是，通常很难将曲率约束在一组特定的边界内，这是自动驾驶中经常需要的。如果我们研究参数曲线的曲率方程，我们可以看到，对于五次样条曲线，曲率作为弧长的函数一般不是多项式。这有可能在样条曲线中引入尖点，甚至曲率的不连续性，这使得很难在整个样条曲线域中近似满足曲率约束。我们将在下一课中更详细地讨论这些曲率约束。

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

作为另一种方法，我们也可以使用多项式螺旋来表示我们的路径。这些曲线为曲线沿其弧长的每个点的曲率提供了一个封闭形式的方程。对于自动驾驶，通常选择三次多项式作为弧长的曲率函数。然而，高阶函数也是可以接受的。使用多项式螺旋的主要优点是其结构有助于满足路径规划问题所需的近似曲率约束。由于螺旋是曲率的多项式函数，曲率值不会像五次样条曲线那样快速变化。这意味着我们只要约束螺旋曲线中几个点的曲率，而螺旋曲线很可能满足整个曲线的曲率约束。这在执行路径优化时非常有用，因为约束的数量大大增加了每个优化步骤的计算工作量。

$$\begin{aligned} \theta(s) &= \theta_0 + \int_0^s a_3 s'^3 + a_2 s'^2 + a_1 s' + a_0 ds' & x(s) &= x_0 + \int_0^s \cos(\theta(s')) ds' \\ \kappa(s) &= a_3 s^3 + a_2 s^2 + a_1 s + a_0 & & = \theta_0 + a_3 \frac{s^4}{4} + a_2 \frac{s^3}{3} + a_1 \frac{s^2}{2} + a_0 s & y(s) &= y_0 + \int_0^s \sin(\theta(s')) ds' \end{aligned}$$

曲率方程

朝向方程

位置方程

多项式螺旋曲线的缺点是，五次不同于样条曲线，螺旋曲线的位置和方向没有闭合形式的解。因此，我们必须进行迭代优化，以生成满足边界条件的曲线。位置方程产生菲涅耳积分 Fresnel integral，而菲涅耳积分没有封闭形式的解。因此，我们需要使用数值逼近技术来计算螺旋曲线的最终端点。在本模块中，我们将使用辛普森法则 Simpson's Rule 来近似这些菲涅耳积分。辛普森法则比其他近似方法更精确，点数更少，这将有助于我们设置优化问题。

Simpson's Rule:
$$\int_0^s f(s')ds' \approx \frac{s}{3n} \left(f(0) + 4f\left(\frac{s}{n}\right) + 2f\left(\frac{2s}{n}\right) + \dots + f(s) \right)$$

样条曲线仅基于起点和终点提供闭合形式的解决方案，而螺旋曲线则不提供。螺旋曲线可确保沿路径平滑曲率变化，而样条曲线则不能。因此，您需要根据具体的应用程序来确定哪种方法是合适的。简单地说，样条曲线可以提高计算效率，而螺旋曲线可以更容易地实现曲率约束。对于本模块，我们将在开发我们的路径规划器时重点关注多项式螺旋曲线，因为我们对确保本地规划器生成的路径能够被车辆顺利、安全地执行非常感兴趣。

Lesson 2: Path Planning Optimization

在本课程中，我们将讨论如何将模块 1 中讨论的一些目标和约束与上一课中介绍的边界条件中的三次螺旋结合起来，以创建路径规划优化问题。通过解决这个问题，我们将能够生成满足所有约束的平滑可行的路径。

上一课中，我们的边界条件描述了两点之间规划路径的绝对最小要求。基本上，它们要求对于给定的起始位置、航向和曲率，我们的规划路径也在特定的位置、航向和曲率处结束。这将为我们的优化问题提供第一组约束条件，称为边界条件。不幸的是，**三次螺旋并没有螺旋末端位置的闭式解**。要根据螺旋线的参数编写约束条件，我们需要使用数值积分技术，如辛普森法则。

让我们更仔细地看看辛普森法则。辛普森法则是一种常用的数值积分技术，通常比其他简单的数值方法**更精确**。这是因为它计算给定函数的二次插值的积分，而不是像某些方法（如中点和梯形规则）中那样计算线性插值的积分。

$$x_s(s) = x_0 + \frac{s}{24} \left[\cos(\theta(0)) + 4 \cos\left(\theta\left(\frac{s}{8}\right)\right) + 2 \cos\left(\theta\left(\frac{2s}{8}\right)\right) + 4 \cos\left(\theta\left(\frac{3s}{8}\right)\right) + 2 \cos\left(\theta\left(\frac{4s}{8}\right)\right) \right. \\ \left. + 4 \cos\left(\theta\left(\frac{5s}{8}\right)\right) + 2 \cos\left(\theta\left(\frac{6s}{8}\right)\right) + 4 \cos\left(\theta\left(\frac{7s}{8}\right)\right) + \cos(\theta(s)) \right]$$

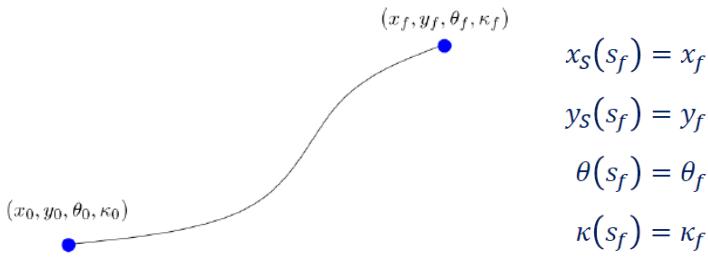
$$y_s(s) = y_0 + \frac{s}{24} \left[\sin(\theta(0)) + 4 \sin\left(\theta\left(\frac{s}{8}\right)\right) + 2 \sin\left(\theta\left(\frac{2s}{8}\right)\right) + 4 \sin\left(\theta\left(\frac{3s}{8}\right)\right) + 2 \sin\left(\theta\left(\frac{4s}{8}\right)\right) \right. \\ \left. + 4 \sin\left(\theta\left(\frac{5s}{8}\right)\right) + 2 \sin\left(\theta\left(\frac{6s}{8}\right)\right) + 4 \sin\left(\theta\left(\frac{7s}{8}\right)\right) + \sin(\theta(s)) \right]$$

$$\int_0^s f(s')ds' \approx \frac{s}{3n} \left(f(0) + 4f\left(\frac{s}{n}\right) + 2f\left(\frac{2s}{n}\right) + \dots + f(s) \right)$$

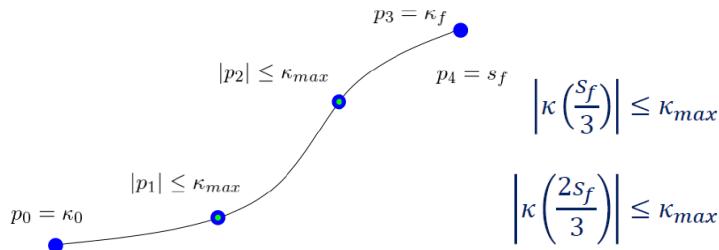
$n = 8$

辛普森法则通过定义由 n 定义的积分域的若干等距分段来进行，然后在每个分段和边界点处求和。例如，如果我们选择 n 等于 4，那么我们将积分域分成四个大小相等的段，因此我们的总和中包含五个点。总和中的每一项都是在除法点处计算的函数乘以适当的系数。在辛普森规则方程的内部，我们可以看到，除了端点项的系数为 1 外，每个项的交替系数为 4 和 2。随着 n 的增加，我们得到了更精确的积分近似值。让我们把这个应用到具体的规划问题上。如果我们在 Simpson 规则近似中取 n 等于 8，那么我们的近似将足够精确，不需要太多的计算开销。

现在，我们有了一个有用的近似值，即螺旋线在由弧长参数 s 定义的任何给定弧长点的 x 和 y 位置。我们将用辛普森法则计算的 x 和 y 的近似值表示为 x_s 和 y_s 。返回到我们的边界条件，现在我们有了一个路径结束位置的近似值，我们可以根据已知的螺旋参数写出边界条件。我们现在可以通过迭代优化螺旋线的参数以及它的总弧长 s_f 来生成从一个点到另一个点的螺旋线，该螺旋线满足给定的边界条件。



然而，在我们这样做之前，让我们回顾一下我们想要强制执行的运动学约束。特别是对于自动驾驶路径规划，我们将重点关注曲率约束。汽车有一个绝对最小转弯半径，需要保持在横向加速度限制内，以保持车轮牵引力和乘坐舒适性。我们将在以后的课程中更详细地讨论这些约束。现在，让我们假设我们的车可以达到两米的最小转弯半径。这对应于 0.5 弧米的最大曲率。现在，很难写出螺旋线上每个点的曲率约束。但是，由于螺旋的多项式性质，我们只需要约束几个均匀分布的点。因为曲率的多项式函数是连续的并且表现良好，所以在执行优化时，我们很可能会生成一个满足曲率要求的螺旋。为简单起见，让我们约束曲线的三分之一点和三分之二点处的曲率。起点和终点曲率已在边界条件中约束。一旦我们这样做了，我们现在有了曲率约束作为螺旋参数的函数，我们有了解决优化问题所需的所有约束。



最后一个难题是我们希望最小化的实际目标函数，保证路线顺畅和舒适。一种方法是沿路径均匀分布绝对曲率。这可以通过最小化我们规划的参数曲线的**弯曲能量 bending energy** 来实现。曲线的弯曲能量是沿路径的整个弧长的曲率平方的积分。

$$f_{be}(a_0, a_1, a_2, a_3, s_f) = \int_0^{s_f} (a_3 s^3 + a_2 s^2 + a_1 s + a_0)^2 ds$$

因为我们有一个多项式曲率函数来描述我们的三次螺旋，弯曲能量积分就螺旋的参数而言有一个封闭形式的解。另外，它的梯度也有一个闭式解。但是，这两个表达式都有许多项，因此最好由符号解算器 symbolic solver 来创建它们。

现在我们有了目标函数。出于我们的目的，我们将假设初始边界条件为零，这意味着我们在车辆框架中定义了局部规划问题，并使用我们在本视频中定义的辛普森规则，得到了航向和 x、y 近似的简化表达式。这意味着初始边界值约束可以被移除，因为它们已经在我们的积分计算中被考虑。现在我们可以把它作为路径生成优化问题。

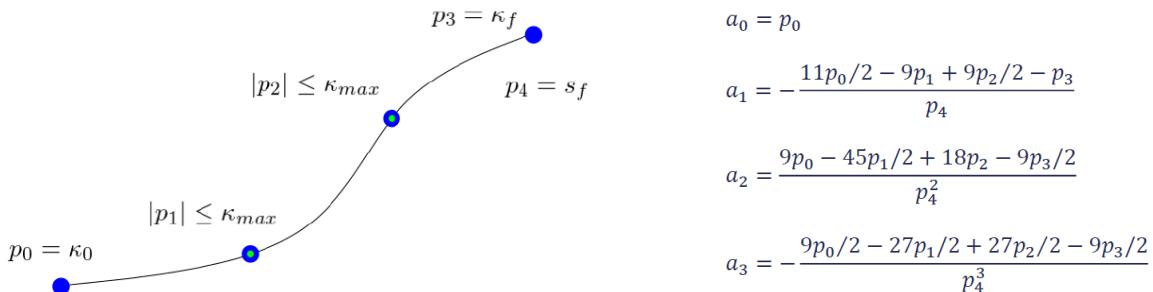
$$\min f_{be}(a_0, a_1, a_2, a_3, s_f) \text{ s.t. } \begin{cases} \left| \kappa\left(\frac{s_f}{3}\right) \right| \leq \kappa_{max}, & \left| \kappa\left(\frac{2s_f}{3}\right) \right| \leq \kappa_{max} \\ x_s(0) = x_0, & x_s(s_f) = x_f \\ y_s(0) = y_0, & y_s(s_f) = y_f \\ \theta(0) = \theta_0, & \theta(s_f) = \theta_f \\ \kappa(0) = \kappa_0, & \kappa(s_f) = \kappa_f \end{cases}$$

然而，在使用规范非线性规划解算器求解时，如何设置此优化问题可能会减慢其速度或导致其根本不收敛。我们现在就来解决这个问题。我们在这里看到的主要问题与最终位置和航向的相等约束有关。由于等式约束必须完全满足，因此数值优化器很难从一个不可行的起点生成一个可行解，而这个起点通常是给定给优化器的任意问题实例的起点。

为了缓解这一问题，在优化中常用**软不等式约束 soft inequality constraint** 来提高优化器的性能。软约束将严格约束 strict constraint 转化为目标函数中的**重惩罚项 heavily penalized term**。重惩罚是约束惩罚项系数应该至少比一般优化目标大一个数量级。尽管这允许优化器违反边界条件等式约束，但强烈鼓励优化器在弯曲能量惩罚项足够大以影响优化器之前收敛到尽可能接近边界条件的解。我们还将假设我们的初始曲率是已知的，并且通常设置为零，这对应于 $a_0 = 0$ 。这就减少了一个优化变量的数量。在软化这些约束之后，我们的新优化问题如下。

$$\min f_{be}(a_0, a_1, a_2, a_3, s_f) + \alpha(x_s(s_f) - x_f) + \beta(y_s(s_f) - y_f) + \gamma(\theta_s(s_f) - \theta_f) \quad \text{s. t.} \begin{cases} \left| \kappa \left(\frac{S_f}{3} \right) \right| \leq \kappa_{max} \\ \left| \kappa \left(\frac{2S_f}{3} \right) \right| \leq \kappa_{max} \\ \kappa(s_f) = \kappa_f \end{cases}$$

我们要解决的最后一个问题与优化参数有关。虽然在我们的目标函数中使用三次螺旋系数更直观，但我们实际上可以通过考虑最终曲率边界约束来减少我们正在搜索的参数的数量。让我们用一组不同的参数重新定义我们的三次螺旋，这些参数由向量 p 表示，其中 p 有五个元素。 p_0 到 p_3 ，表示起点，三分之一点，三分之二点和终点的曲率， p_4 是路径的弧长。方便的是，我们在曲率参数和螺旋参数之间有一个封闭的映射。因此，我们可以很容易地计算所有的约束条件和目标项，作为这些新 p 变量的函数，而不是螺旋的系数。



一旦优化问题得到解决，我们就可以使用这里的方程将结果映射回螺旋系数。因为我们已经知道了初始曲率和最终曲率，所以我们可以消除两个变量 p_0 和 p_3 。这使得我们在优化问题 p_1 、 p_2 和 p_4 中只有三个变量。通过使用边界条件，我们降低了优化问题的维数，从而大大提高了计算速度。最终的优化问题如下。

$$\min f_{be}(a_0, a_1, a_2, a_3, s_f) + \alpha(x_s(p_4) - x_f) + \beta(y_s(p_4) - y_f) + \gamma(\theta_s(p_4) - \theta_f) \quad \text{s. t.} \begin{cases} |p_1| \leq \kappa_{max} \\ |p_2| \leq \kappa_{max} \end{cases}$$

我们在重新映射到 p 参数后，用等价函数替换了螺旋参数的函数。注意，**初始和最终路径曲率 p_0 和 p_3 是常数**。所以优化变量现在只有 p_1 、 p_2 和 p_4 。这种简化是可能的，因为曲率的边界条件在路径的起点和终点是已知的。现在我们已经做了这些修改，我们可以有效地解决路径规划问题，并大大减少陷入局部极小的机会。

Lesson 3: Optimization in Python

在本课程中，我们将学习 Python 优化的基础知识，通过使用 SciPy 优化库来解决一般非线性优化问题所需的一些函数。

SciPy 优化库涵盖了一些最流行的优化算法，使它们易于访问，并确保其实现的合理效率。许多已实现的优化方法在所需参数类型方面具有相似的结构。因此，为了将其抽象为一个简单的接口，SciPy 优化库 optimized library 包含一个通用的**最小化函数 minimize function**。可用的优化方法的一些例子包括共轭梯度法 conjugate gradient、Nelder-Mead 法、dogleg 法和 BFGS 法。

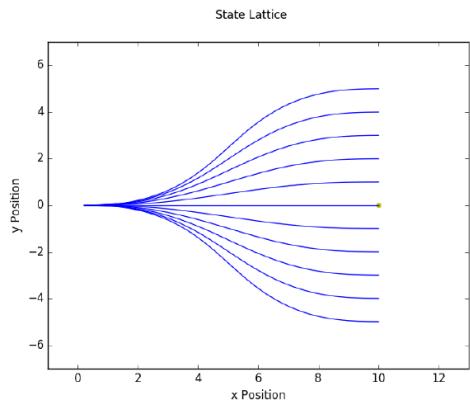
库运行的特定优化算法将取决于传递给此函数的方法参数 method parameter。方法参数还将决定优化算法需要哪些附加参数。例如，在我们将使用的 L-BFGS-B 算法中，我们不仅要求模型最小化，而且要求模型雅可比和变量界。当模型为单标量值函数时，雅可比矩阵降为梯度。这个 Jacobian 通过 jac 参数传递给 minimize 函数，如函数调用中所示。我们希望最小化的实际函数是最小化函数的第一个参数。约束作为约束字典或对象的列表传递给约束变量。此外，还有一个可选的 options 参数，高级用户可以使用它自定义优化器输出的内容。这些优化算法还需要对模型或目标函数的优化变量进行初始猜测，这在函数调用中由 x0 给出。

让我们看一下 BFGS 算法，了解如何使用 SciPy 实现优化的具体示例。基本上对于 BFGS 算法，我们需要传递一个函数指针，指向我们希望最小化的实际目标函数，以及一个函数指针，指向一个计算目标函数雅可比矩阵的函数。这些函数将包含所有优化变量的向量，以便在特定点评估目标函数和雅可比矩阵。一旦优化完成，最小化函数将返回一个结果变量。由 x 表示的结果的成员变量将返回优化变量的最终向量，其中已达到局部极小值。如前所述，我们还可以为优化问题指定约束。对于大多数算法，这些约束是以列表或字典的形式给出的。最简单的约束类型是关于目标变量的不等式约束，称为边界。边界由 L-BFGS-B 算法指定为列表的一个列表，其中每个子列表的长度为 2，并且包含每个优化变量的上限和下限。换句话说，第一个子列表对应于 x0 的边界，第二个子列表对应于 x1 等。这些边界随后被传递给最小化函数的约束可选参数。线性和非线性约束也可以传递给优化器，但现在我们将重点讨论如何使用优化约束的边界。有关更多详细信息，您可以在线查看 SciPy 优化文档。还可以通过传入希望在优化器函数中使用的每个约束对象的 Python 列表来组合多种类型的约束。总之，在本视频中，我们介绍了如何使用 SciPy 优化库设置优化问题。特别地，我们讨论了如何将用户定义的目标函数以及参数边界传递给优化器。

Lesson 4: Conformal Lattice Planning

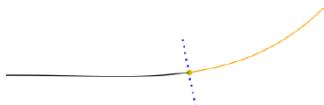
在本课程中，我们将使用在前面课程中开发的优化技术，导出一个成熟的路径规划器，称为共形晶格规划器 Conformal Lattice Planner。与所有的路径规划器一样，目标是规划一条从自主汽车当前位置到给定目标状态的可行无碰撞路径。共形格规划器利用道路的结构化特性，在避开障碍物的同时**加快规划过程**。通过只关注那些稍微转向目标路径左侧或右侧的平滑路径选项，共形晶格规划器生成与人类驾驶非常相似的计划。**在规划道路上的路径时，汽车通常不应考虑离开道路，除非出现紧急停车情况。**

正因为如此，共形网格规划器选择一个中心目标状态以及一系列交替目标状态，这些目标状态是相对于道路的方向从中心目标状态横向偏移 laterally offset 而形成的。



在这个共形晶格的例子中说明了这一点。其中每条路径的终点从中心路径横向偏移，**中心路径**对应于**道路上的目标点**。因为一般来说，汽车应该沿着自我车道前进。我们不太关心一条不会导致前进的路径，所以我们可以大大减少搜索空间，保持共形格规划器的计算可处理性。

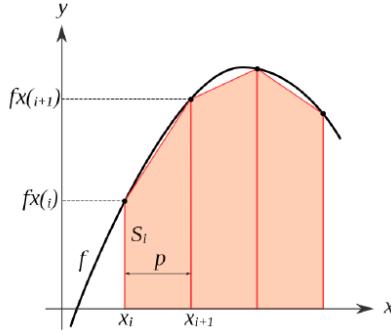
那么我们如何选择这个目标状态呢？一般来说，在为路径规划选择目标状态时有一个关键的权衡。如果您选择的目标状态接近当前 **ego** 车辆位置，则可以减少查找到目标点的路径所需的计算时间。但是，您也会降低规划器以平滑和舒适的方式避开路径上更远的障碍物的能力。这可能是问题，在较高的速度，汽车将涵盖更多的距离之间的规划周期。通常，在规划中使用的目标范围是动态计算的，基于诸如车速和天气条件等因素。不过为简单起见，在本模块中将使用固定的目标范围 **goal horizon**。



我们将目标点作为车道中心线上的点，即前方距离等于我们的目标范围。这里沿着这条路径用对应于所选目标位置的金色点来说明这一点。蓝色点对应于横向偏移的目标点，这些目标点将用作晶格中每个螺旋的备用端点约束。车道中心线的黑色部分的弧长等于我们选定的目标范围。在每一个规划步骤中，我们都会基于同一范围重新计算我们的目标点，并沿着车道前进。一旦找到了这些目标状态，我们就可以计算出达到每个目标状态所需的螺旋线。在这一点上，我们不必担心路径是否无碰撞，我们只需要到每个目标状态的运动可行路径。

因此，我们可以使用我们在第二课中开发的优化公式，来求解从当前位置到每个端点位置的三次螺旋。如果其中任何一个螺旋在运动上不可行或无法达到所需的目标状态，我们将丢弃这些螺旋，使它们不再被视为潜在路径。请注意，一旦一个优化问题得到解决，我们只有得到的参数向量 p 。然后我们必须撤消我们最初对螺旋系数执行的转换，以便从 p 向量中检索它们。一旦我们有了螺旋系数，我们就可以沿着螺旋对点进行采样，得到整个路径的离散表示。因为我们没有沿着螺旋线的位置的闭合解，所以我们再次需要进行数值积分。

然而，由于这一次，我们计算的积分和沿整个螺旋线的许多点，需要一个更有效的方法来解决这些积分。这里我们采用线性插值方法 - **梯形法则 trapezoidal rule**。在这种情况下，梯形规则比辛普森规则有效得多，因为曲线上的每个后续点都可以从上一个点构造出来。所以我们只需要在螺旋上扫一次就可以得到所有需要的点。另一方面，辛普森法则要求我们为每个点求解一个积分近似值，这样效率要低得多。

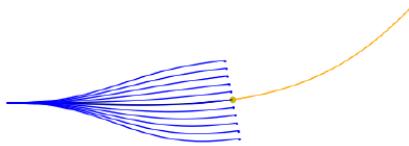


$$x(s) = x_0 + \int_0^s \cos(\theta(s')) ds'$$

$$y(s) = y_0 + \int_0^s \sin(\theta(s')) ds'$$

$$\int_0^s f(x) dx \approx \sum_{i=1}^{N-1} \frac{f(x_{i+1}) + f(x_i)}{2} (x_{i+1} - x_i)$$

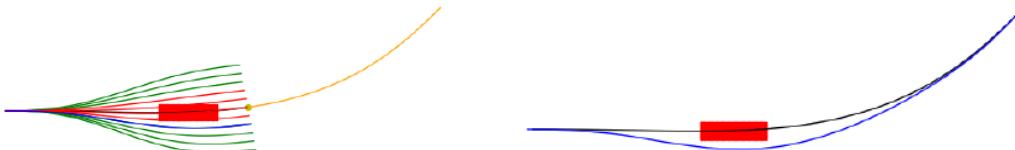
在 Python 中，我们可以使用累积梯形函数 `cumulative_trapezoid()` 来实现这一点。一旦应用了梯形规则，我们现在就有了每个目标点的每个螺旋的离散表示。跟踪每个点的曲率以及位置和航向是很重要的，因为这将有助于我们以后的速度剖面规划。我们有曲率和航向的闭式解，所以不需要数值积分。在使用梯形规则之后，我们现在为这里显示的每个目标状态生成了一组路径。



现在我们有了一整套路径，我们需要看看哪些路径是无碰撞的。要做到这一点，我们可以使用模块 4 中讨论的任何一种碰撞检查技术。一般来说，我们可以使用二进制占用网格，如果一个单元格被占用，则包含一个，否则为零。然后，我们可以根据占用率网格的单元来计算汽车的足迹，然后将足迹扫过螺旋中的每个点，生成路径的线束。如果 `swath` 中某个单元的占用网格包含障碍物，则所讨论的路径将与障碍物发生碰撞，并应标记为发生碰撞。如果沿整个路径的线束中的任何单元从未发生这种情况，则该路径被视为无碰撞。或者，如果 `ego` 车辆和 `ego` 车道上的每个障碍物都可以封闭在一个圆近似中，我们可以使用圆检查。然后我们在路径上的每个点上为 `ego` 车辆放置圆圈，并检查是否与 `ego` 车道内的每个障碍物发生碰撞。为了说明碰撞检查的结果，我们在路径中添加了一辆停着的车辆，现在需要对其进行规划。像我们在第四单元中所做的那样，将线束扫过每个螺旋后，我们将无碰撞路径标记为绿色，将与障碍物碰撞的路径标记为红色。



在这一点上，我们有一套可行的和无冲突的路径，但我们需要一个方法来选择最好的一个路径。选择过程在很大程度上是一种设计选择。因为对于给定的规划应用程序可能有多个有用的标准。例如，我们可能希望选择尽可能远离障碍物的路径。因此，我们可能会在占用网格中为过于接近障碍物的路径添加一个惩罚项。我们可能还想惩罚偏离最近车道中心线太远的条款，因为我们不想在执行车道变换时分割车道太长时间。现在，我们将使用一个简单的度量，使规划器偏向于从路径集中选择尽可能接近中心目标状态的路径。实际的惩罚函数并不重要，只要惩罚增加，你离中心目标越远。通过偏向中心路径，我们鼓励规划者遵循参考路径，并且仅在参考路径不可行或与障碍物碰撞时才允许其偏离参考路径。为简单起见，我们可以将此函数作为从中心目标状态到我们正在检查的路径的目标状态的位移。然后，我们可以迭代路径集中的每一条路径，找到一条使代价最小化的路径，并选择它作为最终路径发布。在我们的示例中，所选路径以蓝色高亮显示。



如果我们现在在汽车沿着路径移动时重复这个过程多个时间步，我们的规划器就能够规划一条收敛到目标状态的路径，同时也能避开障碍物。与我们在第四单元中开发的反应式规划器类似，这个规划器以后退的方式朝着车道尽头的目标前进。我们现在已经具备了在环境中形成平滑、无碰撞路径的所有必要条件，有利于在车道上向前推进。

Lesson 5: Velocity Profile Generation

在本课中，我们将介绍一种方法，为路径规划算法生成的给定输入路径生成速度剖面。如果你回想一下第一单元，有许多不同的因素会影响速度剖面的生成。特别是，我们将关注行为规划器提供的参考速度、前方动态障碍物的速度以及保持乘客舒适性和车辆稳定性所需的速度。

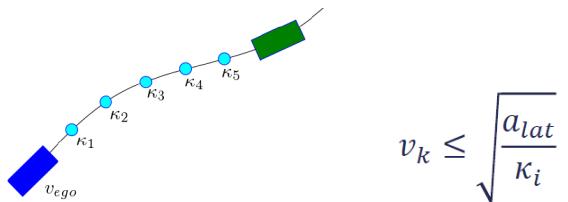
生成速度剖面的第一步是确定最终所需的速度。我们最终速度的一个好的初始值是行为规划器给我们的参考速度。该参考速度将在很大程度上受到基于当前驾驶场景选择的行为计划的机动的影响。例如，如果我们停在一个十字路口，而灯仍然是红色的，那么行为规划器将向本地规划器发出一个停止机动，包括输出一个零速度参考。如果我们现在正沿着一条给定的道路笔直行驶，那么参考速度可能就是当前道路的速度限制。我们将用 V_{ref} 表示这一点。

我们考虑的下一个输入是动态障碍物状态。特别是，我们关注的是前面的领头车。领头车的速度调节着我们的速度，因为如果我们超过领头车的速度，我们最终会与它相撞。回想一下，碰撞时间 time-to-collision (TTC) 是我们相对于领先车辆的相对速度以及 s 给出的领先车辆路径长度的函数。



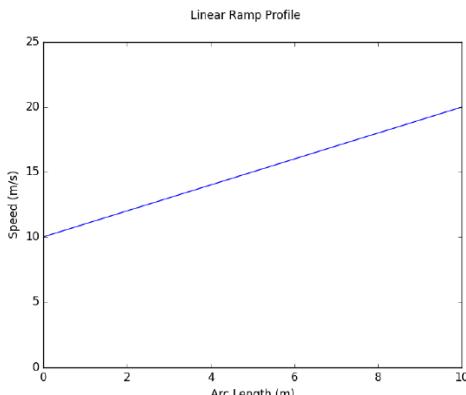
因此，为了保持安全碰撞时间，我们将参考速度取领先车辆速度和行为计划或参考速度的最小值。此外，我们需要确保在到达他们的当前位置之前，我们的车速低于领先车辆的速度，否则，我们仍然有与他们相撞的风险。一般来说，在处理动态障碍物时，为了安全起见，最好在计算中留出空间和时间缓冲区。因此，如果我们的路径终点位于领先车辆的当前位置之前，我们需要确保在该点之前达到最终速度，包括离开空间缓冲区，如沿路径的红色点所示。当我们到达速度曲线中的红色点时，如果车辆行驶速度比我们慢，我们就需要达到领先车辆的速度。当我们到达当前位置时，前面的车辆将向前移动。在那一点上，我们将达到它的速度，防止我们与它相撞。请注意，我们还可以进一步进行引导车辆跟踪，并在给定引导车辆速度的情况下直接确保碰撞或分离距离的安全时间，但这会将控制功能从相对速度跟踪更改为相对距离跟踪，因此在本视频中我们将坚持基于速度的方法。

我们考虑的最终输入是沿规划路径的最大曲率。回想上节课，当我们对优化路径进行采样时，我们记录了每个点的螺旋曲率 κ 。此外，我们在模块 1 中提到，保持在舒适矩形内需要最大横向加速度。横向加速度是瞬时曲率和沿曲线纵向速度的函数。因此，曲率限制了我们在穿越路径时可以采用的纵向速度。我们可以通过确保路径上每个点的速度保持在要求的极限以下来实现这一点。然而，如果曲率迅速变化，我们可能无法达到所需的速度，同时保持在我们的纵向加速度范围内。因此，我们还可以找到路径中所有点的最大曲率，然后找到该点的相关最大速度。我们的轮廓必须在路径的某个点达到所需的速度。因此，我们定义了在最小点减速到所需速度，然后加速。我们可以对曲率约束的下一个最大违反处重复这个过程，依此类推，直到速度剖面沿其长度满足曲率约束。



对于本课程中的评估，一个更简单的方法是确定最大曲率点，设置相关速度，然后简单地保持该速度，直到我们通过该点。由于我们不断地以 receding horizon 方式重新规划，一旦到达高曲率点，生成的新速度剖面将根据先前定义的其他目标自然提高速度。本质上，我们可以将速度剖面生成过程简化为将基于曲率的最大速度与行为规划器和领头车辆的前两个最大速度相结合的行为，将这三个速度中的最小值作为我们速度剖面的期望最终速度。 $v_f = \min(v_{ref}, v_{lead}, v_k)$

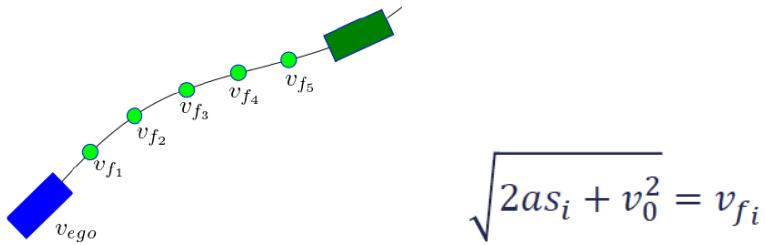
接下来，我们需要讨论速度剖面的形状。一个可能的简单选择是从我们的当前速度到我们之前计算的期望最终速度生成一个线性斜坡剖面 linear ramp profile。



在规划轮廓时，我们知道路径的总弧长，表示为 s ，以及初始和最终速度 v_0 和 v_f 。我们首先要计算的是所需的加速度，我们可以用给定的输入直接求解。我们必须小心，以确保计算出的加速度不会超过我们在本课程前面讨论的舒适矩形。如果它确实超过了舒适矩形，那么我们需要钳制 clamp 它。如果需要的加速或减速超过舒适矩形，但出于安全考虑（例如在紧急停车操作期间），则我们可以绕过该舒适矩形以防止碰撞。如果我们钳制我们的加速度，那么我们需要用我们的最大加速度 a_{max} 来更新我们的最终速度，而不是我们计算的加速度 a 。

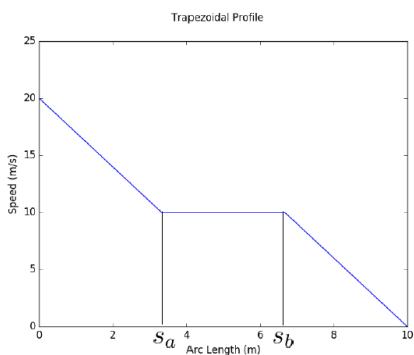
$$\frac{v_f^2 - v_0^2}{2s} = a \quad \sqrt{2as + v_0^2} = v_f$$

一旦我们有了加速度曲线，我们就可以通过观察到第 i 个点的弧长来计算路径上每个点的速度。通过遍历整个路径并计算每个点所需的速度，我们已经完全生成了一个沿路径的速度剖面，以获得我们所需的终点速度。



另一种速度剖面是梯形剖面 trapezoidal profile。当一辆车接近停车标志时，这是很有用的，我们想从我们的标称速度减速到较低的运输速度，然后再减速到停车标志处停车。

对于这个速度剖面，我们将初始和最终速度 v_0 和 0、期望的通过速度 transit velocity v_t 和期望的减速度 a_0 作为输入。这种减速通常选择在我们的舒适矩形内，以使轮廓尽可能平滑。使用这个规划器的第一步是计算在我们的初始减速到我们期望的运输速度期间我们将行驶的距离。这是在梯形轮廓的第一段中移动的弧长。从这一点，我们知道有多少弧长度沿我们的初始路径应专用于我们的初始减速。



$$v_{fi} = \begin{cases} \sqrt{2a_0s_i + v_i^2}, & s_i \leq s_a \\ v_t, & s_a \leq s_i \leq s_b \\ \sqrt{2a_0(s_i - s_b) + v_i^2}, & s_b \leq s_i \leq s_f \end{cases}$$

一旦我们有了这个弧长值，我们就可以迭代通过这些点，直到弧长，并使用第二个方程来计算第 i 个点所需的速度。然后，我们可以重复类似的过程，最终减速，从过境速度停在我们的停止点。我们将把路径的整个弧长表示为 S_f 。所以我们轮廓的第三段长度是 S_f 减去 S_b 。然后我们可以解出如下的问题。一旦我们有了 S_b ，我们就可以在这个弧长范围内迭代这些点，并为它们指定所需的速度，使它们缓慢减速到停止。剖面中间的其余点取我们的恒定通过速度 v_t 。

把所有的东西放在一起，我们的速度剖面有三个区域：一个初始下降到我们的慢速通过速度，一个在这个通过速度下的恒定穿过，最后一个下降到我们的停止点。

我们在这里向您展示了两种生成速度剖面的方法，但也有许多其他可用的选项。使用高阶方法，如双二次速度规划器，我们可以尽量减少沿轨道的急动。在我们这里展示的两种方法中，在我们的速度斜坡中应用高阶函数也是可能的，这两种方法可以生成更平滑、更舒适的速度剖面。最终，速度剖面可以优化，以同时满足多个目标，同时满足舒适性和安全性约束取决于执行的行为。

关于运动规划模块。让我们总结一下要点。首先学习使用两种曲线进行路径规划：样条曲线和螺旋曲线。然后定义制定路径规划问题所需的目标和约束。您开发了 psi pi 优化函数的经验，并将其应用于共形晶格规划器以识别无碰撞路径。最后，您学习了如何沿路径构造速度剖面以满足多个约束。您现在应该有足够的知识来集成路径规划器和速度剖面规划器，从而从头开始构建您自己的本地规划器。

- A. Kelly and B. Nagy, “[Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control](#),” The International Journal of Robotics Research, vol. 22, no. 7, pp. 583–601, 2003. This paper discusses the math behind generating spirals to desired terminal states.

