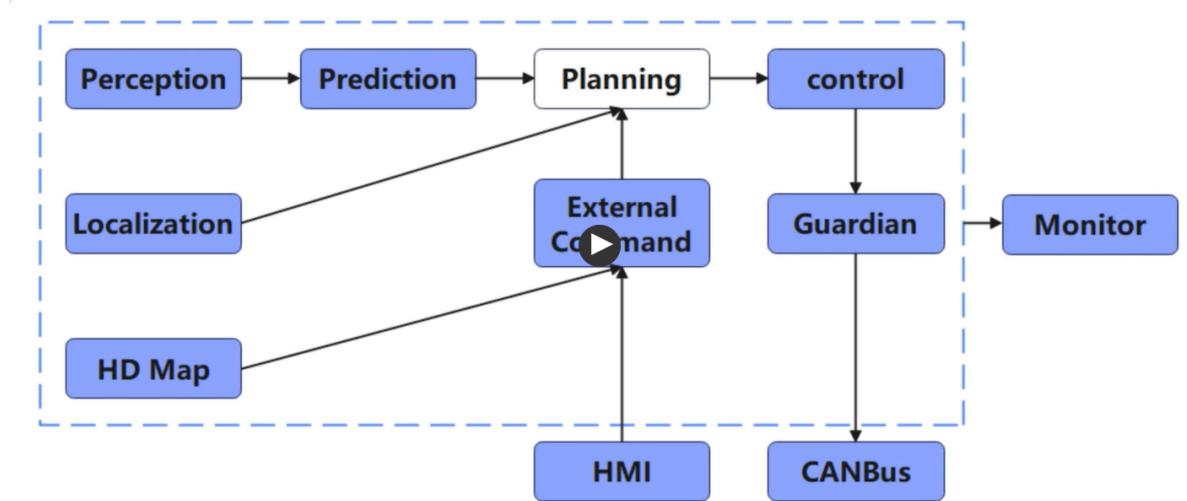
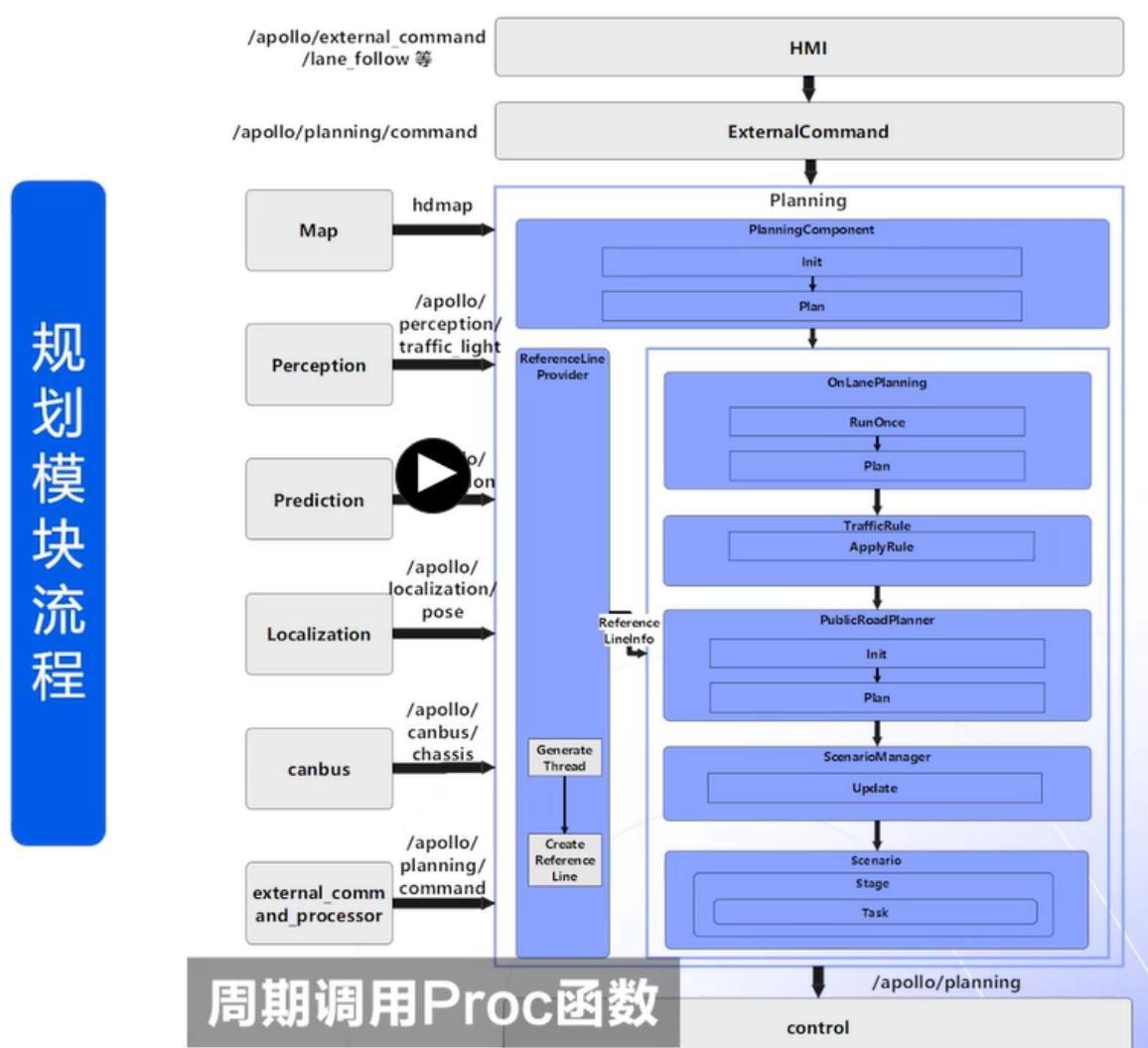


基础版

输入输出



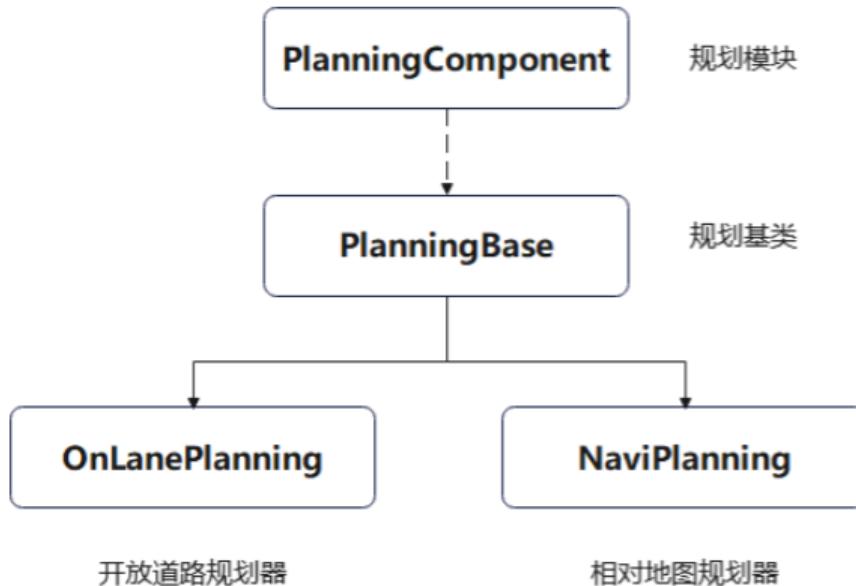
Planning流程



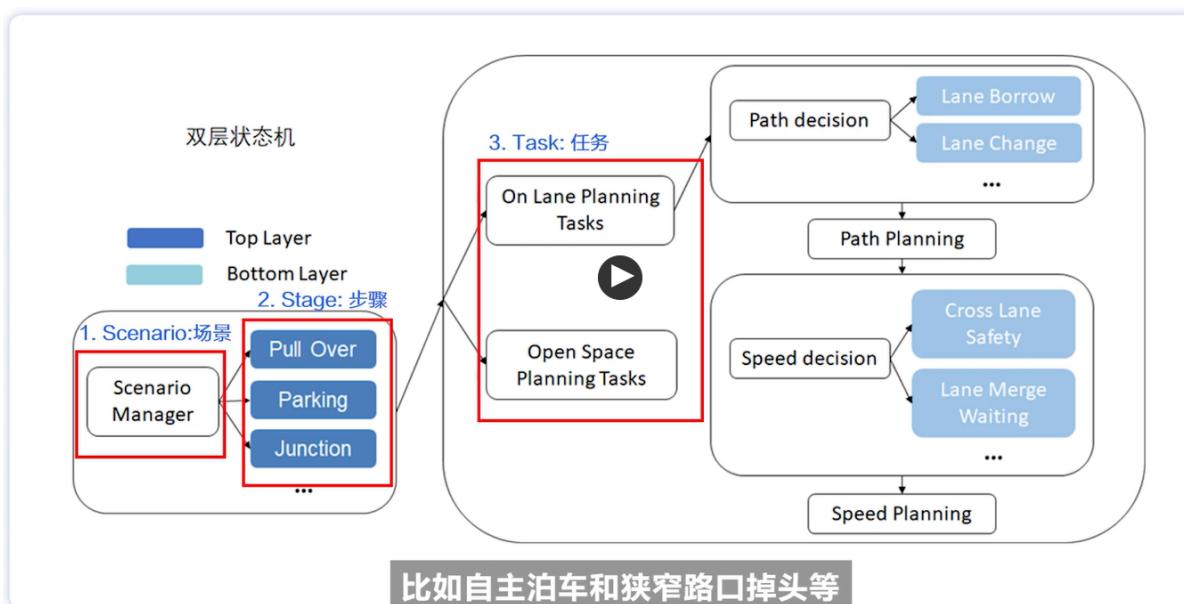
规划模式:

naviPlanner / Onlaneplanning(Lattice/rtk//publicRoadPlanner) / OpenSpacePlanning

- NaviPlanning - 主要用于高速公路的导航规划,
- OnLanePlanning - 主要的应用场景是开放道路的自动驾驶。



双层状态机:



Task



scenario:

```
.  
├── bare_intersection_unprotected          // 无保护交叉路口场景  
├── emergency_pull_over                  // 紧急靠边停车场景  
├── emergency_stop                      // 紧急停车场景  
├── lane_follow                         // 沿车道线行驶场景  
├── park_and_go                        // 靠边停车后沿车道线行驶场景  
├── pull_over                           // 终点靠边停车场景  
├── stop_sign_unprotected               // 无保护停止标志场景  
├── traffic_light_protected            // 有保护交通灯场景  
├── traffic_light_unprotected_left_turn // 无保护左转场景  
├── traffic_light_unprotected_right_turn// 无保护右转场景  
├── valet_parking                      // 泊车场景  
└── yield_sign                          // 让行标志场景
```

scenarioManage选择场景，stage将scenario分为几个阶段顺序执行，每个stage里有排序注册好的task_list，用于路径规划，速度规划等

进阶版：

第一课：交规决策

Q：如何识别红绿灯，停止线的

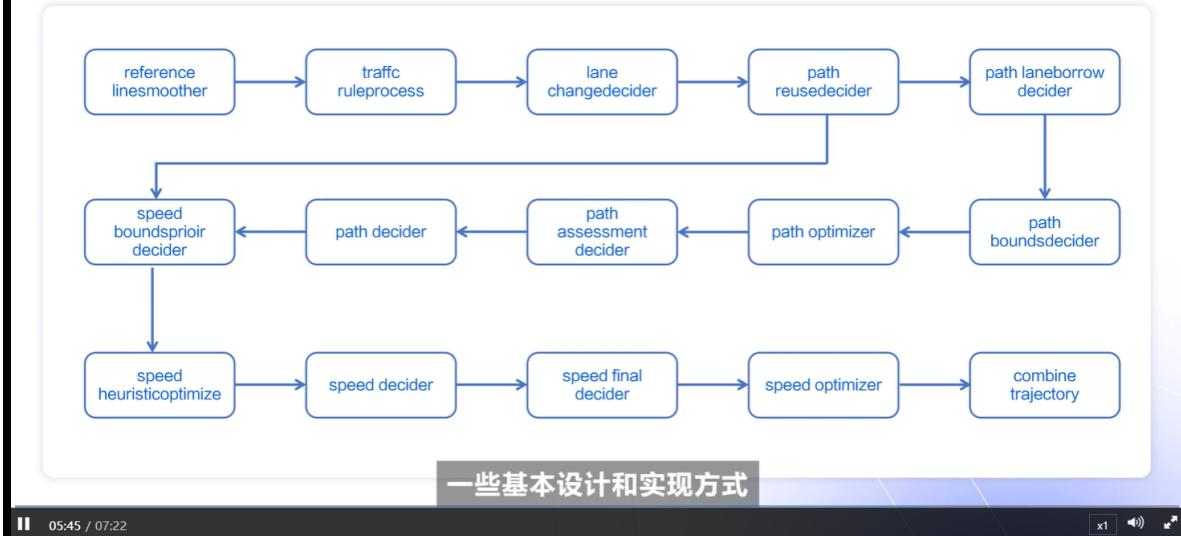
首先根据routing结果从HDMap获取道路中心线，平滑后得到参考线，之后遍历平滑后参考线上的所有交通标识

Q：如何切换运行场景

ScenarioManage

Q：如何进行路径规划和速度规划

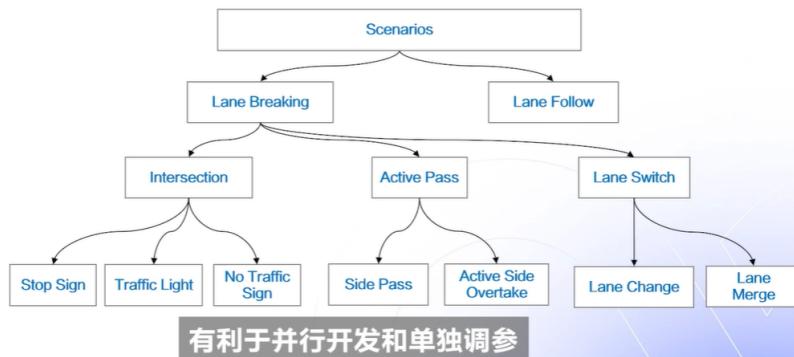
如果



场景 Scenarios

» 依据场景来做决策和规划有以下两个优点

- 1) 场景之间互不干扰，有利于并行开发和独立调参。



» Apollo对交通规则的处理是通过for循环来遍历配置文件

modules/planning/planning_base/conf/traffic_rule_config.pb.txt中设置的交通规则，处理后相关信息存入ReferenceLineInfo中。

The screenshot shows a video player interface with a dark theme. At the top, there is a table mapping rule codes to their meanings:

分别为：	
BACKSIDE_VEHICLE	主车后方来车
CROSSWALK	人行横道
DESTINATION	主车前往的目的地
KEEP_CLEAR	禁停区
REFERENCE_LINE_END	参考线结束
REROUTING	路由重启
STOP_SIGN	停车
TRAFFIC_LIGHT	交通灯
YIELD_SIGN	让行

Below the table is a code snippet from the `TrafficDecider::Execute` function:

```
    Status TrafficDecider::Execute(Frame *frame,
                                     ReferenceLineInfo *reference_line_info) {
        CHECK_NOTNULL(frame);
        CHECK_NOTNULL(reference_line_info);

        for (const auto &rule : rule_list_) {
            if (!rule) {
                AERROR << "Could not find rule ";
                continue;
            }
            rule->Reset();
            rule->ApplyRule(frame, reference_line_info);
            ADEBUG << "Applied rule " << rule->Getname();
        }

        BuildPlanningTarget(reference_line_info);
        return Status::OK();
    }
```

A yellow box highlights the word "通过" (Through) in the code, with the text "通过for循环来遍历配置文件" (Through a for loop to iterate over configuration files) overlaid.

第二课：参考线平滑

ReferenceLineProvider: 1、参考线生成 2、参考线平滑

参考线生成：

Q1、参考线从哪来的？

an: 由高精地图的routing结果搜索到的路径中心线组成

参考线平滑：

Q2、地图中使用的道路中心线能否直接用于规划，会带来什么问题？

AN: 不可以。因为地图中的道路中心线是一群不平滑的离散的点，所以需要根据routing的导航路径重新生成一条平滑的局部参考线

Q3、参考线平滑过程中需要考虑哪些因素

Q4、如何保证平滑后的参考线曲率能够满足车辆转弯半径的限制？

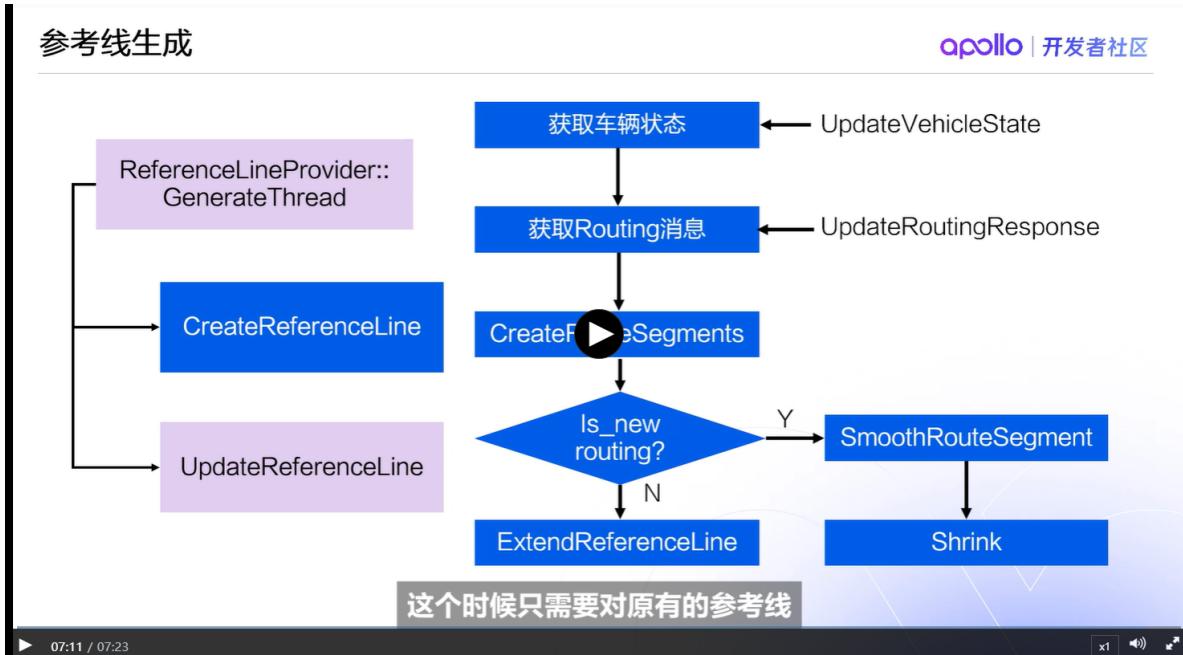
AN: 曲率约束

参考线的数据结构

AN: priority, speedLimit, refPoint, MapPoint

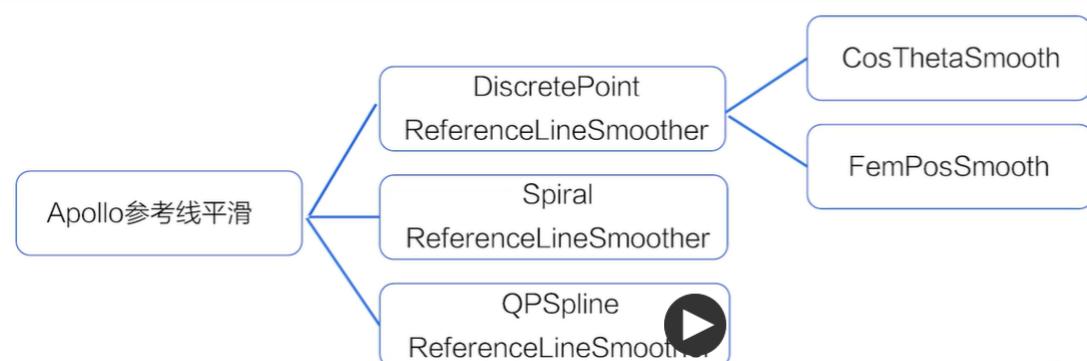


参考线生成

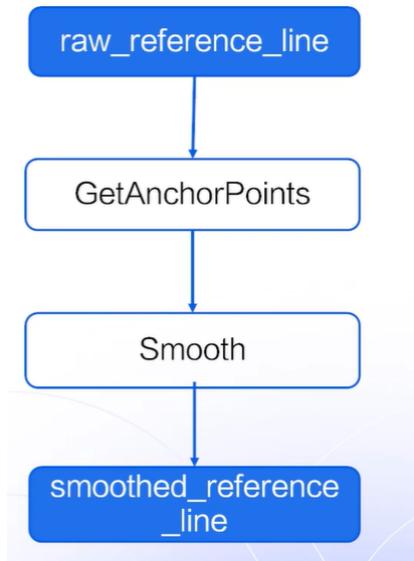


参考线平滑的三种方式：离散点平滑（default）、螺旋线平滑、样条曲线平滑

参考线平滑算法总览



Apollo离散点平滑算法：



AnchorPoint: $x, y, bound$ (横向纵向裕度) (纵向约束0.2, 横向约束0.1-0.5), enforcedPoint(强约束点, 起点和终点)

smooth: 对anchorPoint进行裕度内的有限偏移进行平滑

smooth算法: FEM_POS_DEVIATION_SMOOTHING (default), COS_THETA_SMOOTHING

FEM_POS_DEVIATION_SMOOTHING: 调用solve函数求解

solve函数: 考虑曲率约束 (OSQP线性求解, IPOPT非线性求解)、不考虑曲率约束 (OSQP线性求解 (二次优化))

APOLLO: 二次优化: (不考虑曲率约束)

这里应该是默认重新生成的参考线是贴近原始参考线的, 认为原始参考线的曲率很小, 所以这里不考虑曲率约束

而在泊车场景, 开放空间中没有参考线, 需要考虑曲率约束

优化问题构造和求解

apollo | 开发者社区

优化变量: 离散点 $P_k(x_k, y_k)$

优化目标:

- 平滑度
- 长度
- 相对原始点偏移量

优化函数:

$$cost = cost_{smooth} + cost_{length} + cost_{err}$$

$$cost_{length} = \sum_{i=0}^{n-1} (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2$$

$$cost_{err} = \sum_{i=0}^{n-1} (x_i - x_{i_ref})^2 + (y_i - y_{i_ref})^2$$

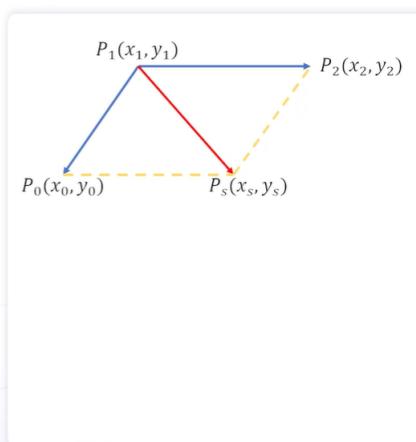
FemPosSmooth:

$$\text{cost}_{smooth} = \sum_{i=1}^{n-1} (x_{i-1} + x_{i+1} - 2 * x_i)^2 + (y_{i-1} + y_{i+1} - 2 * y_i)^2$$

CosThetaSmooth:

$$\cos\theta = \frac{\overrightarrow{P_0P_1} \cdot \overrightarrow{P_1P_2}}{\left| \overrightarrow{P_0P_1} \right| \cdot \left| \overrightarrow{P_1P_2} \right|} = \frac{(x_1 - x_0) \times (x_2 - x_1) + (y_1 - y_0) \times (y_2 - y_1)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} / \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

$$\text{cost}_{smooth} = - \sum_{i=1}^{n-2} \frac{\overrightarrow{P_{i-1}P_i} \cdot \overrightarrow{P_iP_{i+1}}}{\left| \overrightarrow{P_{i-1}P_i} \right| \cdot \left| \overrightarrow{P_iP_{i+1}} \right|}$$



矢量和的模的平方

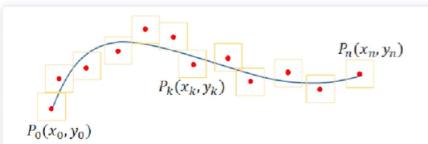
▶ 04:11 / 06:20

x1 ◀ ▶

这里的曲率约束只是每个点的最大曲率约束，不是kappa(s)

约束条件

1. 位置约束



约束方程:

$$x_{i_ref} - x_l \leq x_i \leq x_{i_ref} + x_u$$

$$y_{i_ref} - y_l \leq y_i \leq y_{i_ref} + y_u$$

2. 曲率约束

假设

- 离散点均匀采样: $\|p_i - p_{i-1}\| \cong \|p_{i+1} - p_i\|$
- 相邻线段间夹角很小: $\sin(\theta_i) \cong \theta_i$
- 弧长与弦长相近: $\|\overline{P_0P_1}\| \cong \overline{P_0P_1}$

约束方程:

$$\|2 * p_i - p_{i-1} - p_{i+1}\| \cong 2 * \|p_i - p_{i-1}\| * \sin\left(\frac{\theta_i}{2}\right)$$

$$\cong \|p_i - p_{i-1}\| * \theta_i$$

$$\cong \|p_i - p_{i-1}\| * \|p_i - p_{i-1}\| / R$$

$$\leq \Delta s^2 * cur_{cstr}$$

$$\text{即: } (x_{i-1} + x_{i+1} - 2 * x_i)^2 + (y_{i-1} + y_{i+1} - 2 * y_i)^2 \leq (\Delta s^2 * cur_{cstr})^2, \quad i = 1, 2, 3, \dots, n - 1$$

Δs 是采样平均长度, cur_{cstr} 是最大曲率约束

关于偏移量的硬约束

▶ 05:10 / 06:20

x1 ◀ ▶

第三课：路径规划算法

Q: 如何在平滑后的参考线上规划出行驶路径

Q: 规划路径时需要考虑哪些因素

包括路径决策和路径优化两个方面

一个task对应一条路径

路径规划步骤：

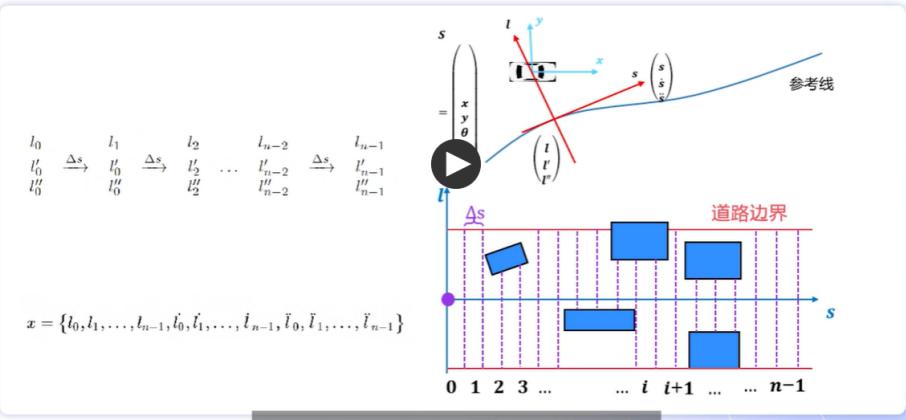
- 1、确定路径边界 (sl坐标系下的l的范围)
- 2、优化路径 (基于二次规划生成平滑，满足约束的sl路径点)
- 3、评估路径 (路径是否有效，选择最优路径)

二次规划算法优化路径

- 1、定义优化变量 (x向量)

基于二次规划的路径规划算法

» 定义优化变量



第二部分是每个点的l一阶导坐标

02:33 / 08:01

apollo | 开发者社区

- 2、设计目标函数 (Q矩阵, c向量)

基于二次规划的路径规划算法

» 目标函数设计

- 确保安全、礼貌的驾驶，尽可能贴近车道中心线行驶：
 $|l_i| \downarrow$
- 确保舒适的体感，尽可能降低横向速度/加速度/加加速度：
 $|l'_i| \downarrow, |l''_i| \downarrow, |l'''_{i \rightarrow i+1}| \downarrow$
- 确保终点接近参考终点：
 $l_{end} = l_{ref}$

$$\min f = \sum_{i=0}^{n-1} w_{l-ref}(l_i - l_{i-ref})^2 + w_l l_i^2 + w_{dl} \dot{l}_i^2 + w_{ddl} \ddot{l}_i^2 + w_{ddd} \ddot{\dot{l}}_i^2 + w_{end-l}(l_{n-1} - l_{end-l})^2 + w_{end-dl}(l_{n-1} - l_{end-dl})^2 + w_{end-ddl}(l_{n-1} - l_{end-ddl})^2$$
$$\ddot{\dot{l}}_i = \frac{\ddot{l}_{i+1} - \ddot{l}_i}{\Delta s}$$

则规划的终点参考点在靠边停车位

03:36 / 08:01

apollo | 开发者社区

» 目标函数二次项、一次项系数的矩阵表示：

$$\begin{aligned} \min f = & \sum_{i=0}^{n-1} w_{l-ref}(l_i - l_{i-ref})^2 + w_l l_i^2 + w_{dl} \dot{l}_i^2 + w_{ddl} \ddot{l}_i^2 + w_{end-l}(l_{n-1} - l_{end-l})^2 \\ & + w_{end-dl}(l_{n-1} - l_{end-dl})^2 + w_{end-ddl}(l_{n-1} - l_{end-ddl})^2 \end{aligned}$$

minimize $\frac{1}{2}x^T Px + q^T x$
subject to $l \leq Ax \leq u$

$$P = \begin{bmatrix} 2w_x & \dots & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 2w_x \\ -2w_{ref}w_l & & & 0 \\ -2w_{ref}w_l - 2w_{end-l}l_{end} & & & \vdots \\ 0 & & & 0 \\ \vdots & & & \vdots \\ -2w_{ref}w_l - 2w_{end-dl}l_{end}^u & & & 0 \\ 0 & & & 2w_{dx} \\ \vdots & & & \vdots \\ -2w_{ref}w_l - 2w_{end-dl}l_{end}^u & & & 2w_{dx} \\ 0 & & & 0 \\ \vdots & & & \vdots \\ -2w_{ref}w_l - 2w_{end-ddl}l_{end}^u & & & 0 \end{bmatrix}$$

$$q = \begin{cases} -2w_{ref}w_l \\ -2w_{ref}w_l - 2w_{end-l}l_{end} \\ 0 \\ \vdots \\ -2w_{ref}w_l - 2w_{end-dl}l_{end}^u \\ 0 \\ \vdots \\ -2w_{ref}w_l - 2w_{end-ddl}l_{end}^u \end{cases}$$

计算出P矩阵和q向量

▶ 03:44 / 08:01

x1 🔍 ↻

3、设计约束 (A, b)

» 要满足的约束：

必须在道路边界内，同时不能和障碍物有碰撞

$$l_i \in (l_{min}^i, l_{max}^i)$$

根据当前状态，主车的横向速度/加速度/加加速度有特定运动学限制：

$$l'_i \in (l_{min}'(S), l_{max}'(S)), \quad l''_i \in (l_{min}''(S), l_{max}''(S)), \quad l'''_i \in (l_{min}'''(S), l_{max}'''(S))$$

加速度 加加速度等车辆运动学特性的限制

▶ 06:20 / 08:01

x1 🔍 ↻

曲率约束

» 要满足的约束:

根据当前状态，主车的横向速度/加速度/加加速度有特定运动学限制：

$$\frac{d^2l}{ds^2} = -(\kappa'_{ref} l + \kappa_{ref} l') \tan(\theta - \theta_{ref}) + \frac{(1-\kappa_{ref} l)}{\cos^2(\theta-\theta_{ref})} \left[\frac{(\kappa(1-\kappa_{ref} l)}{\cos(\theta-\theta_{ref})} - \kappa_{ref} \right]$$



假设：
轨迹几乎沿着参考线规划
 $\Delta\theta \approx 0$

规划的曲率数值上很小

$$0 < \kappa_{ref} < \kappa \ll 1 \rightarrow \kappa_{ref} \kappa \approx 0$$

$$\frac{d^2l}{ds^2} = \kappa - \kappa_{ref}$$

根据车辆运动学关系
计算最大曲率：

$$\kappa_{max} = \frac{\tan(\alpha_{max})}{L}$$

$$-\kappa_{max} - \kappa_{ref} < \ddot{l}_i < \kappa_{max} - \kappa_{ref}$$

接下来我们就得到了二阶导的取值范围

Werling M, Ziegler J, Kammei S, et al. Optimal trajectory generation for dynamic street scenarios in a frenet frame[C]//2010 IEEE International Conference on Robot

曲率变化率约束

» 要满足的约束:

根据当前状态，主车的横向速度/加速度/加加速度有特定运动学限制：

$$\frac{d^3l}{ds^3} = \frac{d}{dt} \frac{d^2l}{ds^2} \cdot \frac{dt}{ds}$$



主路行驶中，实际车轮转角很小 $\alpha \rightarrow 0$ ，近似有 $\tan \alpha \approx \alpha$

$$\frac{d^2l}{ds^2} \approx \kappa - \kappa_{ref} = \frac{\tan(\alpha)}{L} - \kappa_{ref} \approx \frac{\alpha}{L} - \kappa_{ref}$$

$$v = \frac{ds}{dt}$$

$$\frac{d^3l}{ds^3} = \frac{\alpha'}{Lv} < \frac{\alpha'_{max}}{Lv}$$

带入得到3阶导计算公式

► 06:00 / 08:01

» 要满足的约束:

必须满足基本的物理原理：

$$\begin{aligned} l''_{i+1} &= l''_i + \int_0^{\Delta s} l'''_{i \rightarrow i+1} ds = l''_i + l'''_{i \rightarrow i+1} * \Delta s \\ l'_{i+1} &= l'_i + \int_0^{\Delta s} l''(s) ds = l'_i + l''_i * \Delta s + \frac{1}{2} * l'''_{i \rightarrow i+1} * \Delta s^2 \\ l_{i+1} &= l_i + \int_0^{\Delta s} l'(s) ds \\ &= l_i + l'_i * \Delta s + \frac{1}{2} * l''_i * \Delta s^2 + \frac{1}{6} * l'''_{i \rightarrow i+1} * \Delta s^3 \end{aligned}$$

是满足三阶导恒定的多项式关系

► 06:45 / 08:01

4、求解器求解

OSQP 二次规划算法求解器

第四课：速度规划算法

Q：速度规划算法需要考虑哪些因素

Q：不同规划算法的作用是什么？

速度规划算法：

st坐标系，s是路径规划出的路径

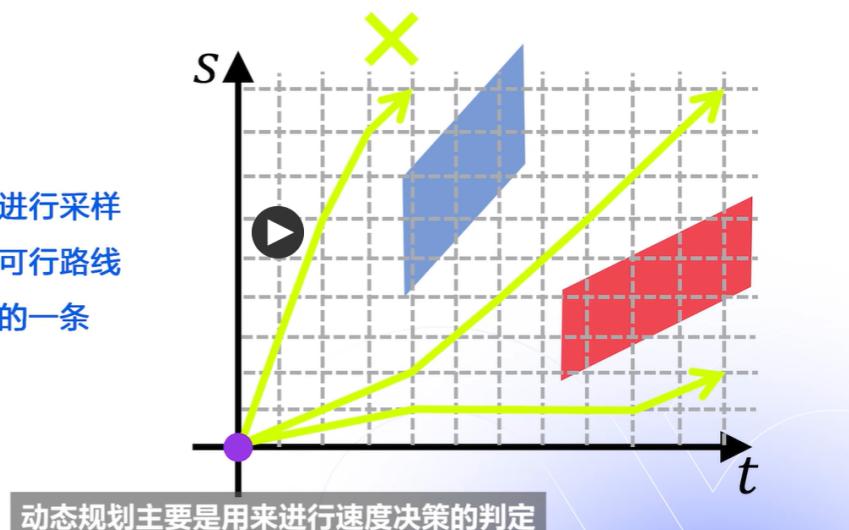
动态规划（粗规划）

主要用于速度决策,产生凸空间

基于动态规划的速度规划

apollo | 开发者社区

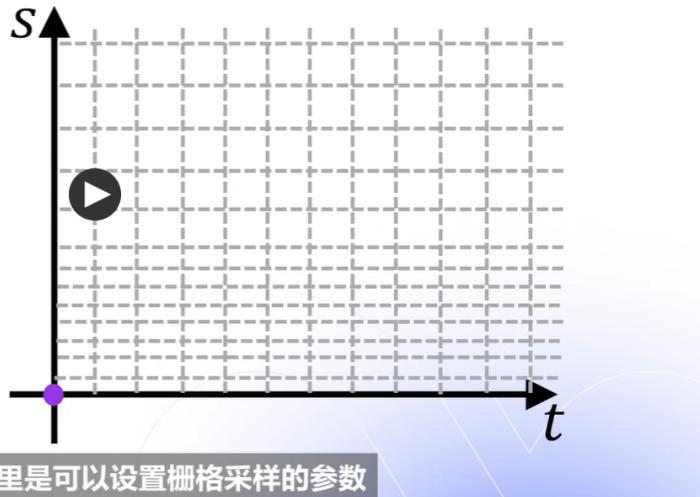
- » 对路程和时间进行采样
- » 搜索出粗略的可行路线
- » 选出代价最小的一条



» 对路程和时间进行采样

unit_t: 1.0
 dense_dimension_s: 101
 dense_unit_s: 0.1
 sparse_unit_s: 1.0

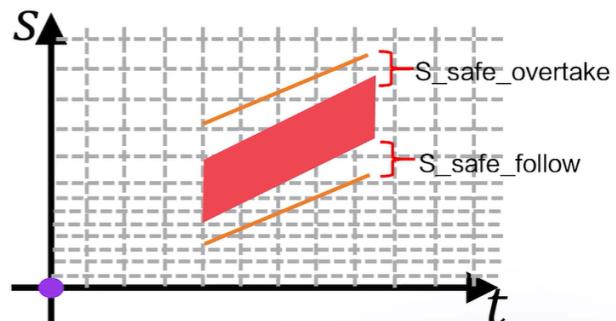
path_time_heuristic/conf/default_conf.pb.txt



基于动态规划的速度规划

» 设计状态转移方程

- 障碍物cost计算:
- 和障碍物满足安全距离
 - 无碰撞



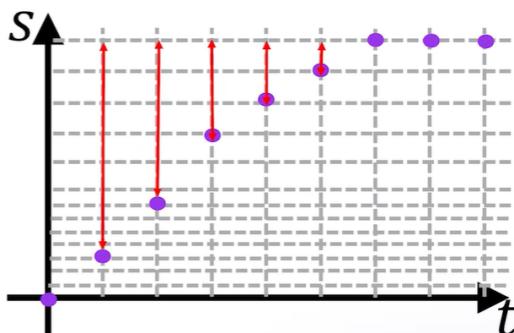
$$C_{obs}(i, j) = \begin{cases} \inf & s_{obs_lower}(i) \leq s(j) \leq s_{obs_upper}(i) \\ 0 & s(j) \geq s_{obs_upper}(i) + s_{safe.overtake} \\ 0 & s(j) \leq s_{obs_lower}(i) - s_{safe.follow} \\ w_{obs}[s(j) - (s_{obs_lower}(i) - s_{safe.follow})]^2 & s_{obs_lower}(i) - s_{safe.follow} \leq s(j) \leq s_{obs_upper}(i) \\ w_{obs}[(s_{obs_upper}(i) + s_{safe.upper}) - s(j)]^2 & s_{obs_upper}(i) \leq s(j) \leq s_{obs_lower}(i) + s_{safe.overtake} \end{cases}$$

其中i代表时间t方向的点索引

基于动态规划的速度规划

» 设计状态转移方程

- 距离cost计算:
- 越早到达终点s越好



$$C_{spatial}(j) = w_{spatial}(s_{total} - s(j))$$

接下来是距离cost的计算

节点之间转移的cost(edge):v, a, jerk

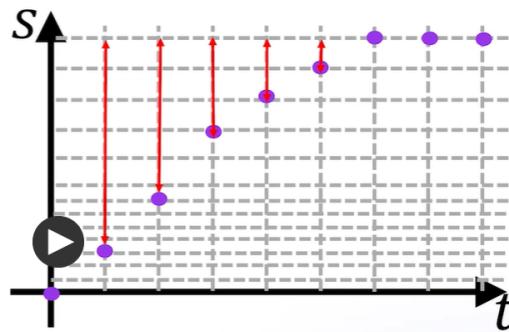
» 设计状态转移方程

状态转移cost计算:

$$C_{edge} = C_{speed} + C_{acc} + C_{jerk}$$

$$\text{节点间速度: } v = \frac{s(j+k) - s(j)}{\Delta t}$$

$$\text{限速比率: } v_{det} = \frac{v - v_{limit}}{v_{limit}}$$



$$C_{speed}(i, j, i+1, j+k) = \begin{cases} \inf & v < 0 \\ w_{exceed} v_{det} + w_{ref} |v - v_{ref}| & v > 0, v_{det} > 0 \\ -w_{lower.speed} v_{det} + w_{ref} |v - v_{ref}| & v > 0, v_{det} < 0 \end{cases}$$

还有节点之间转移的代价

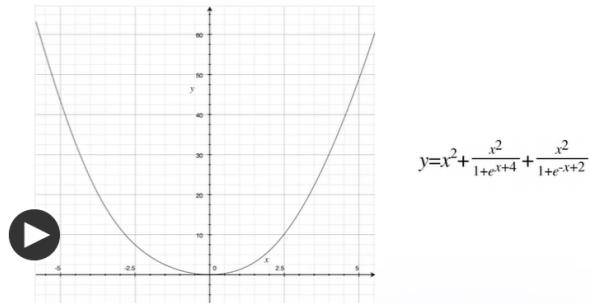
» 设计状态转移方程

状态转移cost计算:

$$C_{edge} = C_{speed} + C_{acc} + C_{jerk}$$

加速度:

$$a(i+1, j+k) = \frac{\frac{s(k+j) - s(j)}{\Delta t} - \frac{s(j) - s(l)}{\Delta t}}{\Delta t}$$



$$cost_a(i+1, j+k) = \begin{cases} \inf & a > a_{max} \mid a < a_{min} \\ w_{acc} a^2 + \frac{w_{deacc}^2 a^2}{1 + e^{a - a_{min}}} + \frac{w_{acc}^2 a^2}{1 + e^{-a + a_{max}}} & 0 < a < a_{max} \\ w_{deacc} a^2 + \frac{w_{deacc}^2 a^2}{1 + e^{a - a_{min}}} + \frac{w_{acc}^2 a^2}{1 + e^{-a + a_{max}}} & a_{min} < a < 0 \end{cases}$$

可以使加速度越接近于0代价越小

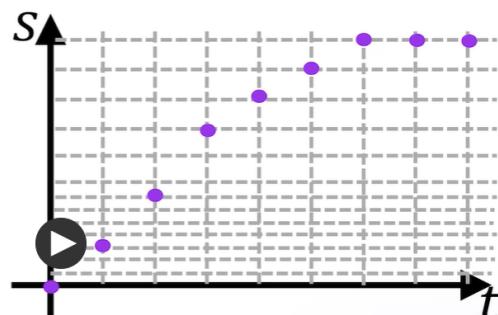
» 设计状态转移方程

状态转移cost计算:

$$C_{edge} = C_{speed} + C_{acc} + C_{jerk}$$

加加速度:

$$jerk = \frac{s_4 - 3s_3 + 3s_2 - s_1}{\Delta t^3}$$



$$cost_{jerk}(i+1, j+k) = \begin{cases} \inf & jerk > jerk_{max}, jerk < jerk_{min} \\ w_{positive.jerk} jerk^2 \Delta t & 0 < jerk < jerk_{max} \\ w_{negative.jerk} jerk^2 \Delta t & jerk_{min} < jerk < 0 \end{cases}$$

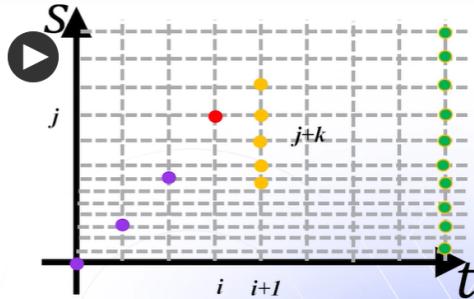
则cost为无穷大

» 设计状态转移方程

$$C(i+1, j+k) = \min \left\{ \begin{array}{l} C_{obs}(i+1, j+k) + C_{spatial}(i+1, j+k) + C(i, j) + C_{edge}(i, j, i+1, j+k) \\ C(i+1, j+k) \end{array} \right.$$

迭代范围：

$$s_j + v_i \Delta t + 0.5 a_{min} \Delta t^2 \leq s(j+k) \leq s_j + v_i \Delta t + 0.5 a_{max} \Delta t^2$$



然后回溯得到所有st点

二次规划 / 非线性规划 (细规划) 二者选其一

二次规划 (OSQP求解器)

基于二次规划的速度规划

- » 1. 确定优化变量
- » 2. 设计目标函数
- » 3. 设计约束

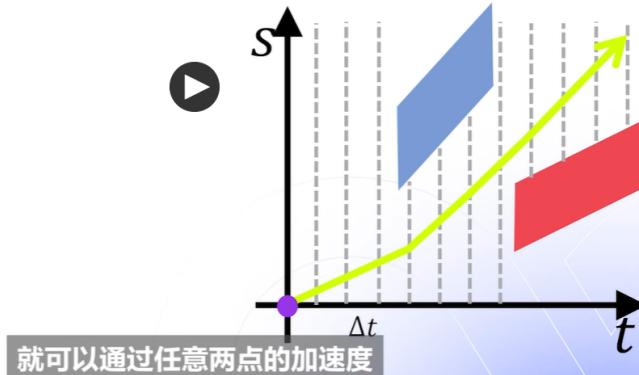
$$\begin{aligned} & \text{minimize} \frac{1}{2} \cdot x^T \cdot H \cdot x + f^T \cdot x \\ & \text{s. t. } LB \leq x \leq UB \\ & \quad A_{eq}x = b_{eq} \\ & \quad Ax \leq b \end{aligned}$$

基于二次规划的速度规划算法

» 1. 确定优化变量

$$x = \{s_0, s_1, \dots, s_{n-1}, \dot{s}_0, \dot{s}_1, \dots, \dot{s}_{n-1}, \ddot{s}_0, \ddot{s}_1, \dots, \ddot{s}_{n-1}\}$$

$$\ddot{s}_i = \frac{\ddot{s}_{i+1} - \ddot{s}_i}{\Delta t}$$



» 2. 设计目标函数

- 尽可能贴合决策时制定的st曲线：
 $|s_i - s_{i-ref}| \downarrow$
- 确保舒适的体感，尽可能降低加速度/加加速度：
 $|\ddot{s}_{i+1}| \downarrow, |\ddot{s}_{i \rightarrow i+1}| \downarrow$
- 尽可能按照巡航速度行驶：
 $|\dot{s}_i - v_{ref}| \downarrow$
- 在转弯时减速行驶，曲率越大，速度越小：
 $|p_i \dot{s}_i| \downarrow$

$$\min f = \sum_{i=0}^{n-1} w_s (s_i - s_{i-ref})^2 + w_v (\dot{s}_i - v_{ref})^2 + p_i \dot{s}_i + w_a \dot{s}_i^2 + w_j \ddot{s}_i^2$$

确定了i时刻的路径的曲率权重

» 3. 要满足的约束条件

- 主车不能和障碍物有碰撞：
 $s_i \in (s_{min}^i, s_{max}^i)$
- 根据当前状态，主车的加速度/加加速度有特定运动学限制：
 $\ddot{s}_i \in [\ddot{s}_{min}, \ddot{s}_{max}], \ddot{s}_{i \rightarrow i+1} \in [\ddot{s}_{min}, \ddot{s}_{max}]$
- 必须满足基本的物理原理：

$$s_{i+1} = s_i + \dot{s}_i \times \Delta t + \frac{1}{3} \ddot{s}_i \times \Delta t^2 + \frac{1}{6} \ddot{s}_{i \rightarrow i+1} \times \Delta t^3$$

$$\dot{s}_{i+1} = \dot{s}_i + \frac{1}{2} \ddot{s}_i \times \Delta t + \frac{1}{2} \ddot{s}_{i \rightarrow i+1} \times \Delta t^2$$

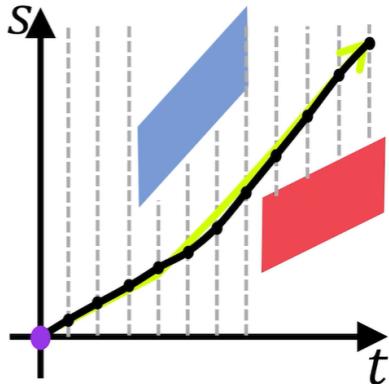
$$\ddot{s}_{i+1} = \ddot{s}_i + \ddot{s}_{i \rightarrow i+1} \times \Delta t$$

第三点是满足基本的物理原理

» 4. 转化为二次规划问题求解

- 输出一条平稳、舒适、能安全避开障碍物并且尽快到达目的地的速度分配曲线。

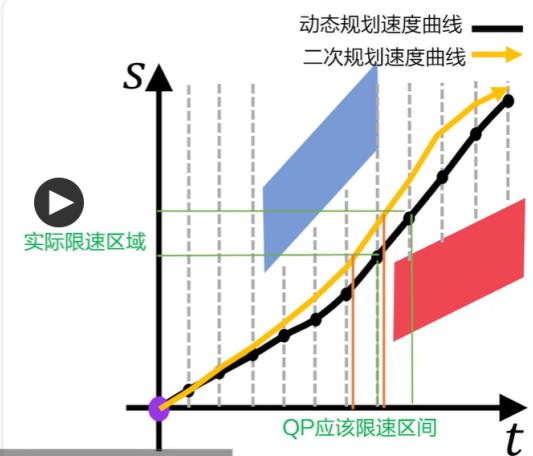
$$\begin{aligned} & \text{minimize} \frac{1}{2} \cdot x^T \cdot H \cdot x + f^T \cdot x \\ & \text{s. t. } LB \leq x \leq UB \\ & \quad A_{eq}x = b_{eq} \\ & \quad Ax \leq b \end{aligned}$$



并且尽快到达目的地的速度分配曲线

» 二次规划速度规划算法的问题

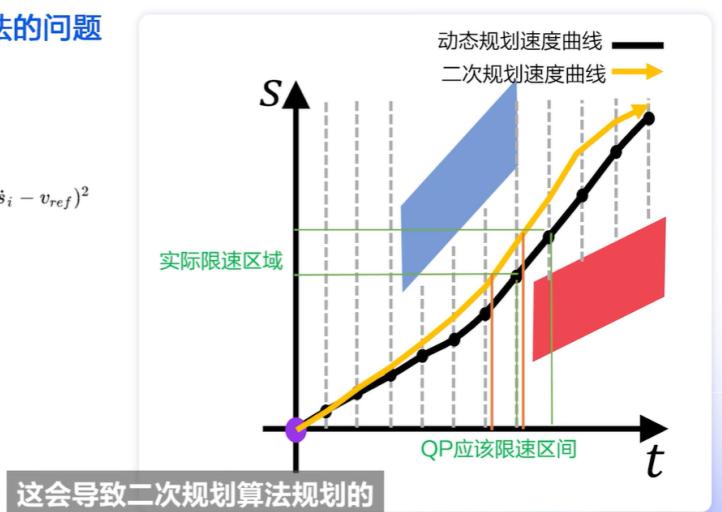
$$\begin{aligned} \min f = & \sum_{i=0}^{n-1} w_s (s_i - s_{i-ref})^2 + w_v (\dot{s}_i - v_{ref})^2 \\ & + p_i \ddot{s}_i + w_a \ddot{s}_i^2 + w_j \ddot{\dot{s}}_i^2 \end{aligned}$$



基于非线性规划的优化算法

» 二次规划速度规划算法的问题

$$\begin{aligned} \min f = & \sum_{i=0}^{n-1} w_s(s_i - s_{i-ref})^2 + w_v(\dot{s}_i - v_{ref})^2 \\ & + p_i \ddot{s}_i + w_a \ddot{s}_i^2 + w_j \ddot{s}_i^2 \end{aligned}$$



基于非线性规划的速度规划

» 1. 确定优化变量

$$x = \left(\begin{array}{l} s_0, \dots, s_{n-1}, \\ \dot{s}_0, \dot{s}_1, \dots, \dot{s}_{n-1}, \\ \ddot{s}_0, \ddot{s}_1, \dots, \ddot{s}_{n-1}, \\ s_{lower0}, s_{lower1}, \dots, s_{lower_{n-1}}, \\ s_{upper0}, s_{upper1}, \dots, s_{upper_{n-1}} \end{array} \right)$$

另外这里还有两组关于s的松弛变量

▶ 02:14 / 07:25

x1 🔍

基于非线性规划的速度规划

» 2. 定义目标函数

$$\begin{aligned} cost = & \sum_{i=0}^{n-1} w_{ref}(s_i - s_{ref_i})^2 + w_v(\dot{s}_i - v_{ref})^2 + w_a \ddot{s}_i^2 + w_j \left(\frac{\ddot{s}_{i+1} - \ddot{s}_i}{\Delta t} \right)^2 + \\ & w_{lat_acc} lat_acc_i^2 + \\ & w_{soft} s_{lower_i} + w_{soft} s_{upper_i} \end{aligned}$$

$$acc_lat_i = \dot{s}_i^2 * kappa(s)$$

以及松弛变量的两项

▶ 02:37 / 07:25

x1 🔍

基于非线性规划的速度规划

apollo | 开发者社区

» 3. 定义约束

- 规划的速度要一直往前走

$$s_i \leq s_{i+1}$$

- 加加速度不能超过定义的极限值

$$jerk_{min} \leq \frac{\ddot{s}_{i+1} - \ddot{s}_i}{\Delta t} \leq jerk_{max}$$

- 速度满足路径上的限速

$$\dot{s}_i \leq speed_limit(s_i)$$

还有速度要满足路径上的限速

04:21 / 07:25

x1 🔍 ↻

二次规划求解 v(s) (略)

第五课：开放空间规划算法（泊车或狭窄路口掉头）（没有参考线的非结构化道路）

Q: 开放空间中如何计算行车轨迹

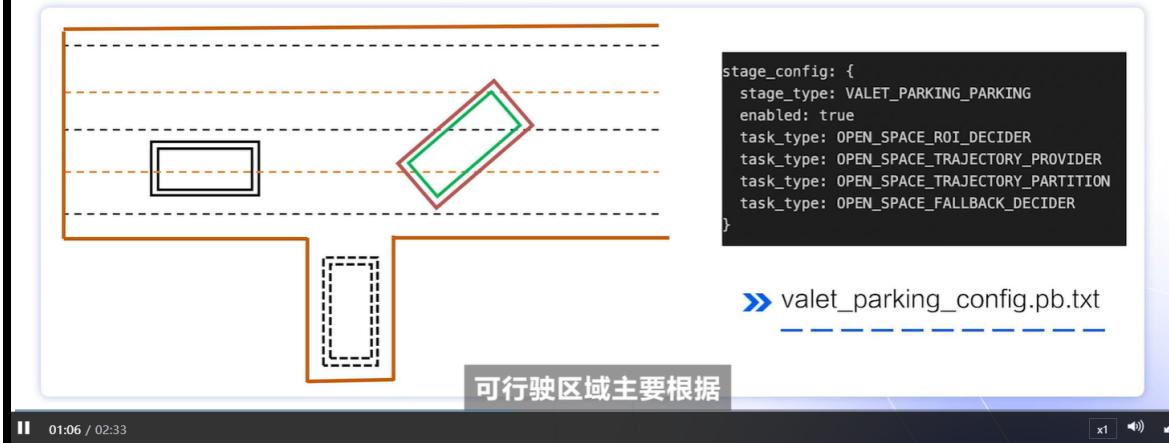
Q: Apollo如何建模并求解

总体流程

1、确定可行驶区域（道路边界，停车位边界，障碍物边界）

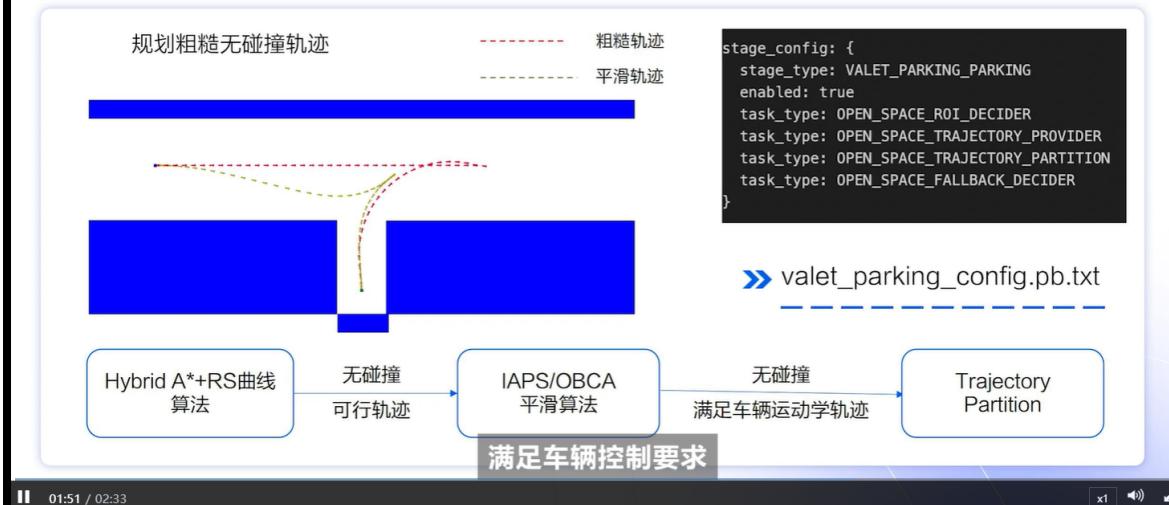
» Task: OPEN_SPACE_ROI_DECIDER

» 根据道路边界，车位边界生成可行驶区域



2.生成平滑轨迹

» Task: OPEN_SPACE_TRAJECTORY_PROVIDER



3、区分前进和后退（根据主车位置）

» Task: OPEN_SPACE_TRAJECTORY_PARTITION

对轨迹进行平滑

```
stage_config: {
  stage_type: VALET_PARKING_PARKING
  enabled: true
  task_type: OPEN_SPACE_ROI_DECIDER
  task_type: OPEN_SPACE_TRAJECTORY_PROVIDER
  task_type: OPEN_SPACE_TRAJECTORY_PARTITION
  task_type: OPEN_SPACE_FALLBACK_DECIDER
}
```

» valet_parking_config.pb.txt

4. 碰撞检测，遇障停车 (fallback)

路径规划算法介绍

1、hybird A*路径规划算法：处理开发空间复杂度较高的非凸问题

补充：A*的过程和问题

A*路径不满足运动学约束，其实是不满足车辆的转向约束

车辆运动学模型中：车辆航向等同于车辆后轴中心的速度方向，将车辆路径等同于后轴中心一段段不同转弯半径的弧线组成。由前轮转角可以计算得到后轴中心的转弯半径。对前轮转角采样，每次向前行驶等距离的弧

基于混合A*的路径规划算法

» A*算法的问题

$$R = \frac{L}{\tan(\alpha)}$$

(x, y, θ)
也就平滑了很多

混合A*算法搜索的路径是离散的（因为前轮转角是离散的，但搜索空间是连续的），所以可能搜不到同位置同方向的终点位置

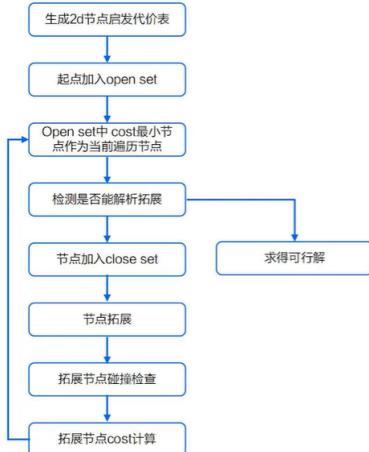
所以要结合Reedshepp算法判断从当前节点出发是否存在解析解

由动态规划得到HA*的启发函数?

基于混合A*的路径规划算法

apollo | 开发者社区

» Hybrid A*算法的求解过程



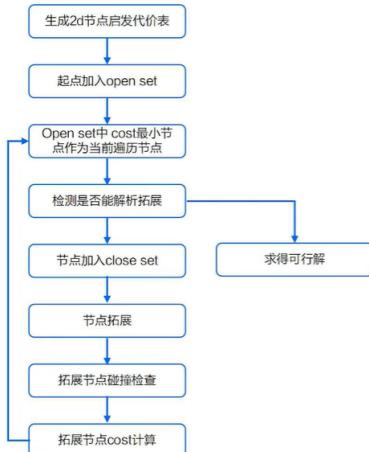
```
grid_a_star_heuristic_generator->GenerateDpMap(ex, ey, Xybounds_,  
obstacles_linesegments_vec_);  
open_set_.emplace(start_node->GetIndex(), start_node);  
open_pq_.emplace(start_node->GetIndex(), start_node->GetCost());  
size_t explored_node_num = 0;  
while (!open_pq_.empty()) {  
    open_rq_.pop();  
    std::shared_ptr<Node3d> current_node = open_set_[current_id];  
    if (AnalyticExpansion(current_node)) {  
        break;  
    }  
    close_set_.emplace(current_node->GetIndex(), current_node);  
    for (size_t i = 0; i < next_node_num; ++i) {  
        std::shared_ptr<Node3d> next_node = Next_node_generator(current_node, i);  
        if (close_set_.find(next_node->GetIndex()) != close_set_.end()) {  
            continue;  
        }  
        if (!ValidityCheck(next_node)) {  
            continue;  
        }  
        if (open_set_.find(next_node->GetIndex()) == open_set_.end()) {  
            explored_node_num++;  
            CalculateNodeCost(current_node, next_node);  
            open_set_.emplace(next_node->GetIndex(), next_node);  
            open_pq_.emplace(next_node->GetIndex(), next_node->GetCost());  
        }  
    }  
}
```

然后定义两个集合

基于混合A*的路径规划算法

apollo | 开发者社区

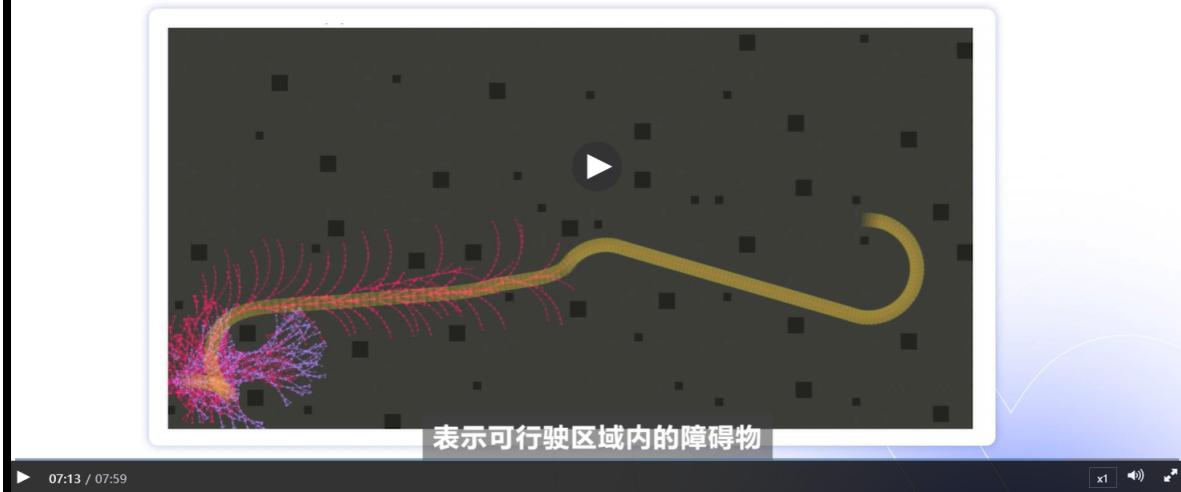
» Hybrid A*算法的求解过程



```
grid_a_star_heuristic_generator->GenerateDpMap(ex, ey, Xybounds_,  
obstacles_linesegments_vec_);  
open_set_.emplace(start_node->GetIndex(), start_node);  
open_pq_.emplace(start_node->GetIndex(), start_node->GetCost());  
size_t explored_node_num = 0;  
while (!open_pq_.empty()) {  
    open_rq_.pop();  
    std::shared_ptr<Node3d> current_node = open_set_[current_id];  
    if (AnalyticExpansion(current_node)) {  
        break;  
    }  
    close_set_.emplace(current_node->GetIndex(), current_node);  
    for (size_t i = 0; i < next_node_num; ++i) {  
        std::shared_ptr<Node3d> next_node = Next_node_generator(current_node, i);  
        if (close_set_.find(next_node->GetIndex()) != close_set_.end()) {  
            continue;  
        }  
        if (!ValidityCheck(next_node)) {  
            continue;  
        }  
        if (open_set_.find(next_node->GetIndex()) == open_set_.end()) {  
            explored_node_num++;  
            CalculateNodeCost(current_node, next_node);  
            open_set_.emplace(next_node->GetIndex(), next_node);  
            open_pq_.emplace(next_node->GetIndex(), next_node->GetCost());  
        }  
    }  
}
```

然后定义两个集合

» Hybrid A* + ReedShepp曲线规划效果

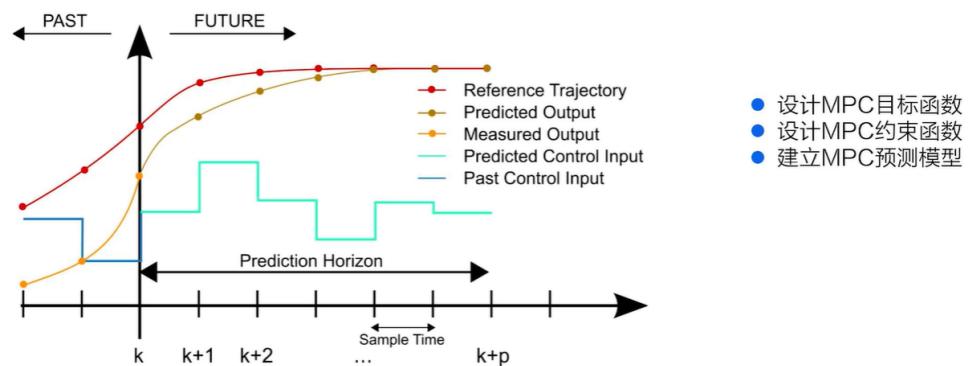


2、OBCA轨迹规划算法（轨迹平滑）

通过凸优化的强对偶性解决开放空间中和障碍物的无约束碰撞求解

MPC

» 模型预测控制MPC

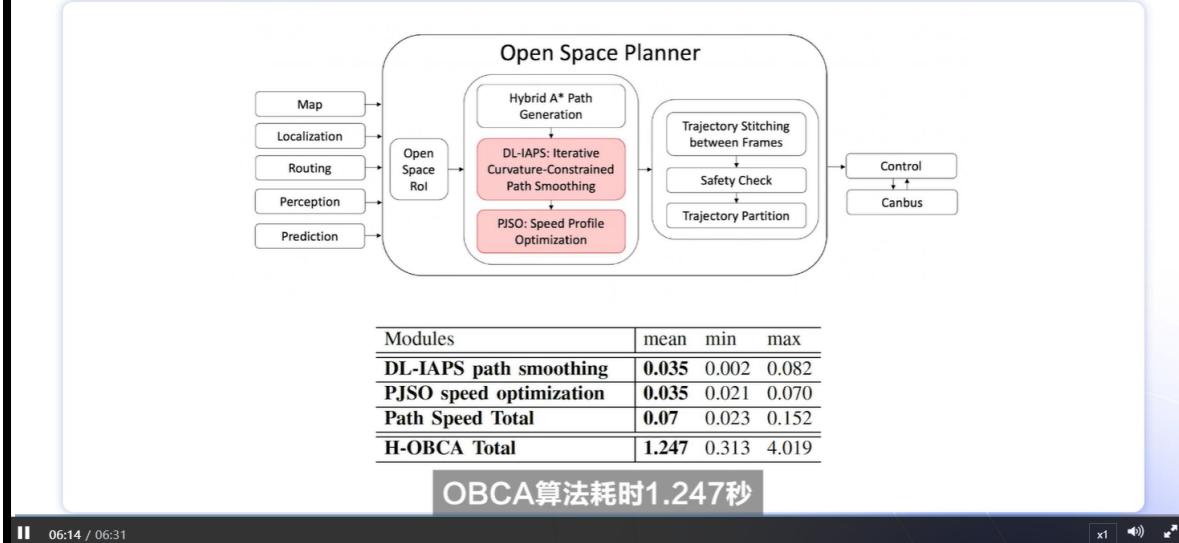


模型预测控制会通过一个采样时间将未来时域离散成多断，在给定控制模型和参考值曲线，计算使预测输出与参考输出最接近的输入序列，并将输入的第一个分量作用于系统

所以还需要设计MPC预测模型

拉格朗日函数，对偶问题，二次规划

3、DL_IAPS路径规划算法（轨迹平滑）



06:14 / 06:31

x1 ◀ ▶