

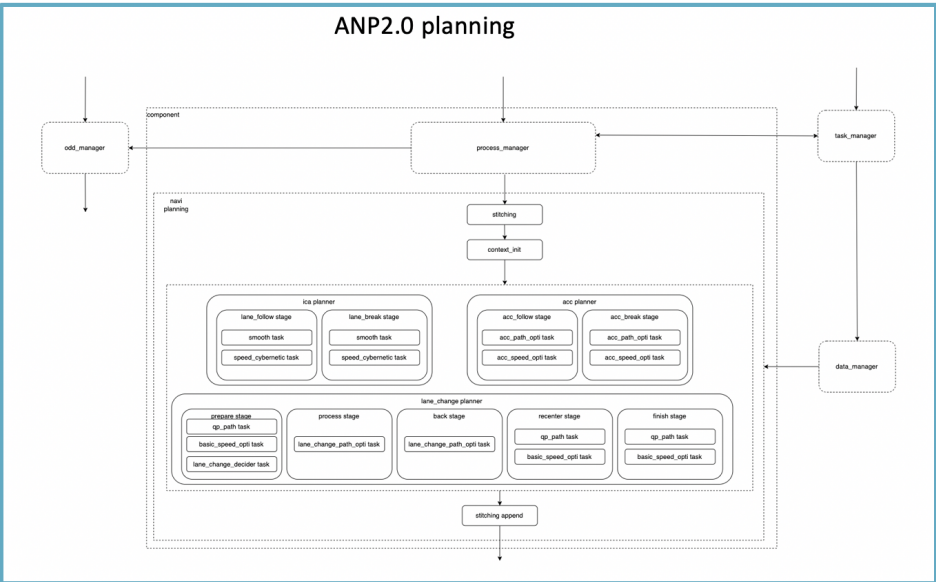
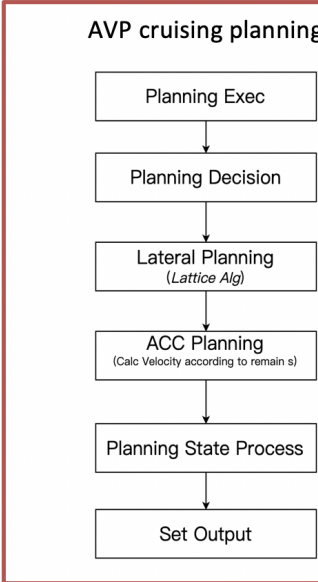
# HAVP巡航规划概述

## 目录

- 1.巡航新架构
- 2.决策模块
- 3.横向规划
- 4.纵向规划

## 1.巡航新架构

调整目的：(1) 算法与业务逻辑解耦 (2)方便后续开发新算法 (3)针对不同的场景可以配置不同的算法



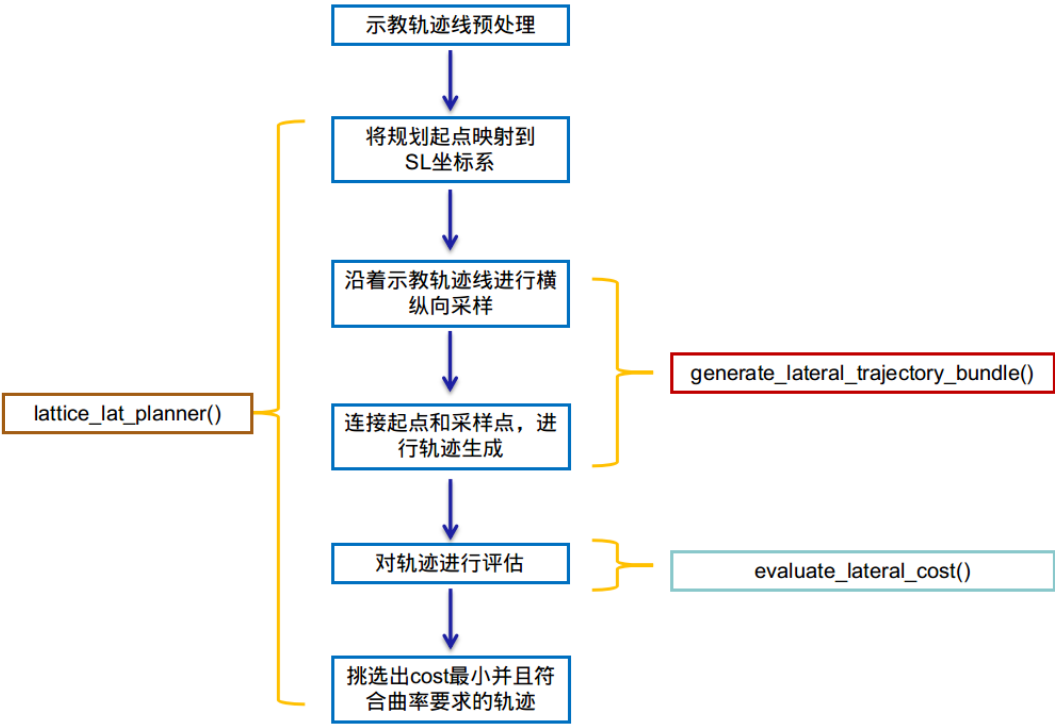
当前新增的决策模块，以及纵向高斯伪普算法已经完成了新架构改造

## 2.决策模块

AVP PNC行为决策模块详细设计报告\_V1.0.docx (455KB)

### 3.横向规划

横向轨迹规划流程图：



(1) 示教轨迹线预处理

该部分内容主要在cruising\_planning.cpp的process\_trace()函数中。

首先，调用\_decision\_core.trace\_filter()函数，对原始轨迹进行滤波操作。主要是使用了SG五次多项式滤波的方式对原始示教轨迹线轨迹进行滤波。

然后，调用\_decision\_core.trace\_refine()函数，该函数主要是对示教轨迹线在起点处进行插值。因为EM提供的示教轨迹线的第一个点可能在车辆当前位置前面，此时trace\_x[0]大于0。此时，为了方便后面将起点映射到Frenet Frame坐标系中，需要对轨迹线进行线性插值，让第一个点位于车辆当前位置的后面。

最后，调用`cal_features()`函数计算每个点的曲率，曲率的导数和航向角。

## (2) 将规划起点映射到SL坐标系下

调用`compute_init_frenet_state()`函数，该函数会进一步调用`cartesian_to_frenet()`函数，根据笛卡尔坐标系下的`_init_point`和映射得到`matched_point`，进行坐标转换，得到SL坐标系下的起点位置，分别保存在`init_s`和`init_d`中。

## (3) 沿着示教轨迹线进行横纵向采样

在`sample_latend_conditions()`函数中，进行横向和纵向采样。

横向采样间隔为`FLAGS_lat_sample_step`(默认0.1米)，横向采样个数为8个，上面3个，下面3个。再加上`d=0`的一个横向采样点，一共7个横向采样点。

纵向采样间隔为 $[(1.4 + i * s\_ratio) * ego\_speed]$ ，其中`s_ratio`为`FLAGS_sample_s_ratio`(默认是0.45)，`i`从1到8。再加上`s=1.4 * ego_speed`的纵向采样点，一共9个纵向采样点。

## (4) 横向轨迹生成

在`generate_trajectory_1dbundle<5>()`函数，调用`QuinticPolynomialCurve1d()`函数使用5次多项式，连接规划的起点和每个采样点。一共需要生成63条候选轨迹。

## (5) 横向轨迹评估

在`evaluate_lateral_cost()`函数中，对每条候选轨迹进行评估，具体操作如下：

首先，对候选轨迹进行采样，采样间隔为`FLAGS_lat_s_step_length`，默认值为0.3米。纵向采样点保存在`s_values`变量中。

调用`lat_offset_cost_new()`函数计算横向偏差的`cost`。横向偏差`cost`考虑两部分的因子：一部分是离散轨迹点中最大的横向偏移距离，另外一部分是轨迹终点的横向偏移距离。

调用`lat_comfort_cost_new()`函数计算舒适度的`cost`。舒适度主要是考虑轨迹的二阶导数，挑选离散轨迹点中二阶导数最大的值作为`cost`。

调用`frame_lat_diff_cost_new()`函数计算帧间跳动`cost`

调用`obs_diff_cost_new()`函数计算距离障碍物的横向距离以及每条轨迹最远能够行驶的距离，分别得到横向和纵向`cost`。

最后，利用`priority_queue`，根据上面计算得到的每个生产轨迹的`cost`值，维护一个小顶堆，结果保存在`_cost_queue`变量中。


## (6) 挑选cost最小并且符合曲率要求的轨迹

从小顶堆中挑选出`cost`最小的候选轨迹。然后将该轨迹转换到笛卡尔坐标系下，在该坐标系下判断轨迹的曲率是否满足要求。

如果满足要求，则认为该轨迹为最优轨迹。否则，从小顶堆中删除该轨迹，重新选择`cost`最小的轨迹。

如果最终没有找到合适的横向规划轨迹，则将示教轨迹线作为横向规划结果返回。

# 4.纵向规划

 AVP巡航纵向规划方案概要设计\_1226.docx (1MB)