PNC 行为决策模块详细设计报告

一、Behavior Decider 模块输入

1．定位：车辆当前位置

2．示教轨迹线：用于生成 path boundary

3．EM 障碍物：主要考虑 3d bounding box 类型的障碍物（车辆，行人和锥筒）

4．EM lane：当前阶段暂时不用，后续可以对 path boundary 进行精细化处理

5．车位信息：泊入时在车位附近进行降速

6．交通元素信息：

（1）需要提前减速的元素：路口，坡道，减速带和道闸

（2）禁止绕行的元素：道闸，坡道

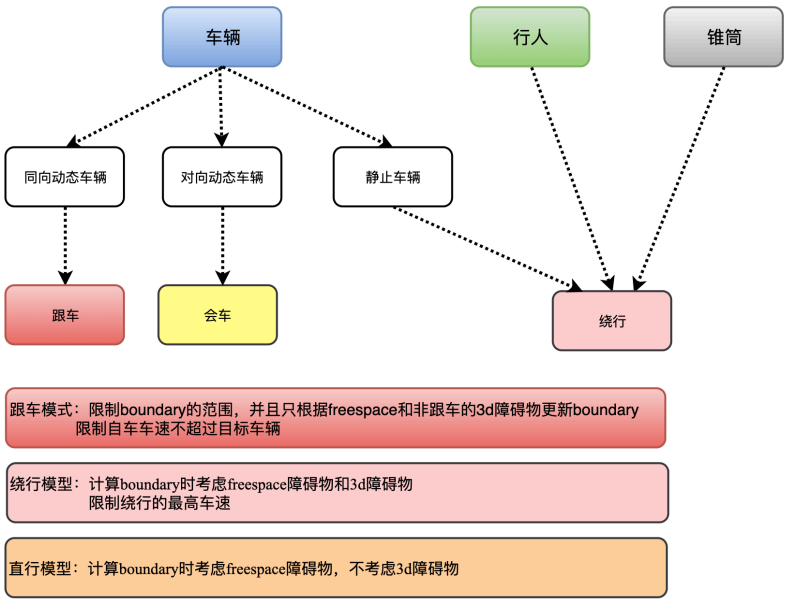
7．规划的状态：path reuse 等规划状态，在 path reuse 阶段保持上一时刻的行为决策结果不变

二、Behavior Decider 模块输出

Behavior：直行，跟车，绕行，靠边停车和会车

Path Decision：输出 SL Boundary（把现有的 path decider 模块拿上来）

Speed Decision：输出每个轨迹点的限速信息



三、行为决策具体流程

1．Behavior 默认是直行

2．遍历自车前方每个 3d 障碍物：

（1）如果障碍物在自车后方，则忽略处理

（2）如果障碍物在沿着示教轨迹线方向的前方 10 米（TBD，根据当前车速动态自适应）外，则忽略处理

（3）如果障碍物在示教轨迹侧向 3 米（TBD）外，则忽略处理

3．如果障碍物类型为车辆：

（1）如果车辆静止（车速小于 1m/s TBD），则 behavior 为绕行

（2）如果是同向动态车辆，且横向 *l* 小于 3 米（TBD）

如果车速大于最高巡航速度 80%（TBD），则 behavior 为跟车

如果车速小于最高巡航速度 80%（TBD），则 behavior 为绕行

否则 behavior 为直行

最高巡航速度需要区分直道和弯道

（3）如果是对向动态车辆，且横向 *l* 大于 3 米（TBD），则 behavior 为会车，否则 behavior 为直行

（4）如果是侧向/横穿的动态车辆，则 behavior 为直行

4．如果障碍物类型为行人，则 behavior 为绕行

5．如果障碍物类型为锥筒，则 behavior 为绕行

6．每遍历完一个 3d 障碍物，则进行 update()操作，将离车最近的一个障碍物对应的非直行状态下的行为决策更新为当前应该采取的 behavior。

7．如果前方 10 米（TBD，基于当前车速自适应）存在道闸，坡道，则将状态统一设置为直行状态？？？

四、横向 boundary 的生成

跟车模式：

①跟车模式下需要防止自车绕行，所以将直道场景下的 boundary 限制为[-0.5, 0.5]？？？，弯道场景下 boundary 限制为[-1,1]

②跟车模式下计算 boundary 时不考虑被跟踪的 3d 障碍物

③同时，需要根据 freespace 障碍物对 boundary 进行更新，如果上下界超过了第①条约束的上下界，则需要进行更新

④在横向规划时也不考虑 3d 障碍物，防止绕行

直行模式：

能够对 freespace 障碍物进行绕行，不对 3d 障碍物进行绕行

为此在这种模式下，横向 boundary 的生成仅仅依赖 freespace 类型障碍物。

横向规划时也不考虑 3d 障碍物，防止绕行

绕行模式：

能够对 freespace 障碍物和 3d 障碍物进行绕行

计算 boundary 时考虑 freespace 障碍物和 3d 障碍物

如果在绕行模式下，计算得到的 boundary 左边界小于右侧边界则认为没有足够的空间供车辆绕行，则将 behavior 更改为直行状态

会车模式：

暂时按照直行模式来做，暂不实现靠边停车或者避让的功能

五、轨迹点限速生成

跟车模式下：

需要限制自车车速不超过前车，直接将纵向规划速度上线设置为前车车速

绕行模式下：

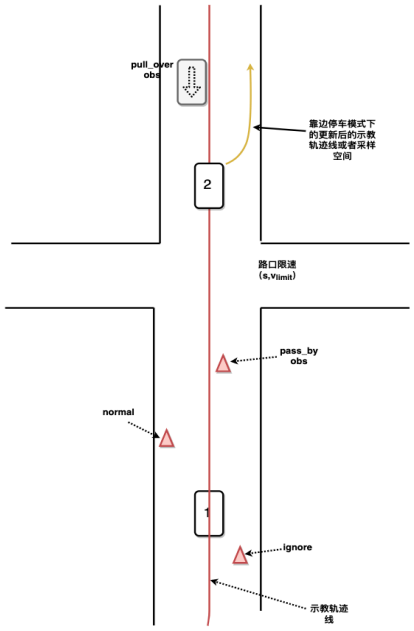
对绕行状态设置一个速度上线（10km/h，TBD）???

直行模式：

和当前纵向规划保持一致

交通元素限速：

如果前方 10 米（TBD，基于当前车速自适应）存在路口，坡道，减速带和道闸等道路元素，则对巡航最高速度进行限制



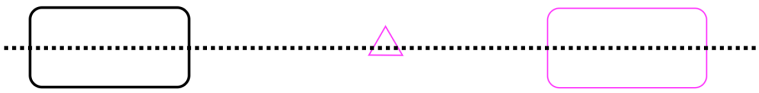
场景决策后需要增加能力校验（根据规划能力）

六、具体场景

Case1:

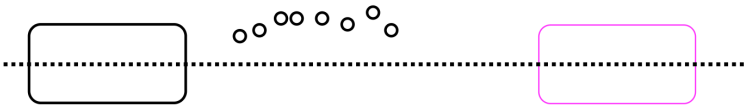
下面这种场景下需要能够先绕行再跟车，跟车时不能绕行

距离近的决策生效，锥筒前在绕行模式，绕过后再进入跟车模式



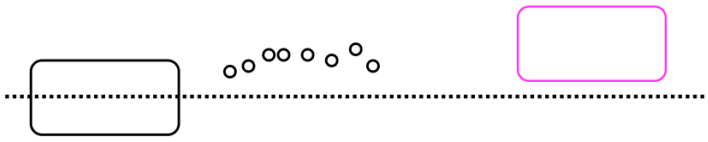
Case2:

这种场景下需要能够在跟车过程中绕行



Case3

这种场景下需要能够先绕过 freespace，然后在前方车辆后面继续进行跟车，不绕行

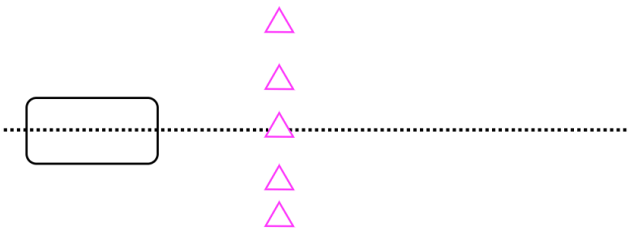


Case4

跟车模式不能绕行

Case5

绕行空间不够，则 behavior 为直行状态



七、数据结构定义

Enum Behavior {Straight, Follow, ByPass, Yield**}**

Struct DecisionObs {

int obs_id;

double frenet_s;

double frenet_l;

Behavior behavior;

FusionType fusion_type;

bool is_static;

double velocity;

vector3d position;

}

大概流程：

1.初始化：

DecisionObs decision_obs_;

decision_obs_. frenet_s = std::DBL_MAX;

2.遍历所有 3d 障碍物：

计算该障碍物的 frenet_s 和 frenet_l；

如果 frenet_s 或者 frenet_l 大于门限，continue;

如果 frenet_s <= **decision_obs_.** frenet_s，则进行 update()操作：

判断该障碍物处于直行，跟车，绕行和会车让行的哪个状态

然后对 decision_obs_中的 behavior 进行更新，原则如下：

依离自车最近的一个障碍物的 behavior 为准

3.根据 **decision_obs_中的** behavior 状态，开始更新计算横向采样 boundary。其中在绕行模式下，如果左边界小于右侧边界则认为没有足够的空间供车辆绕行，则将 behavior 更改为直行状态

4.根据 behavior 对每个示教轨迹线上点生成限速