**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

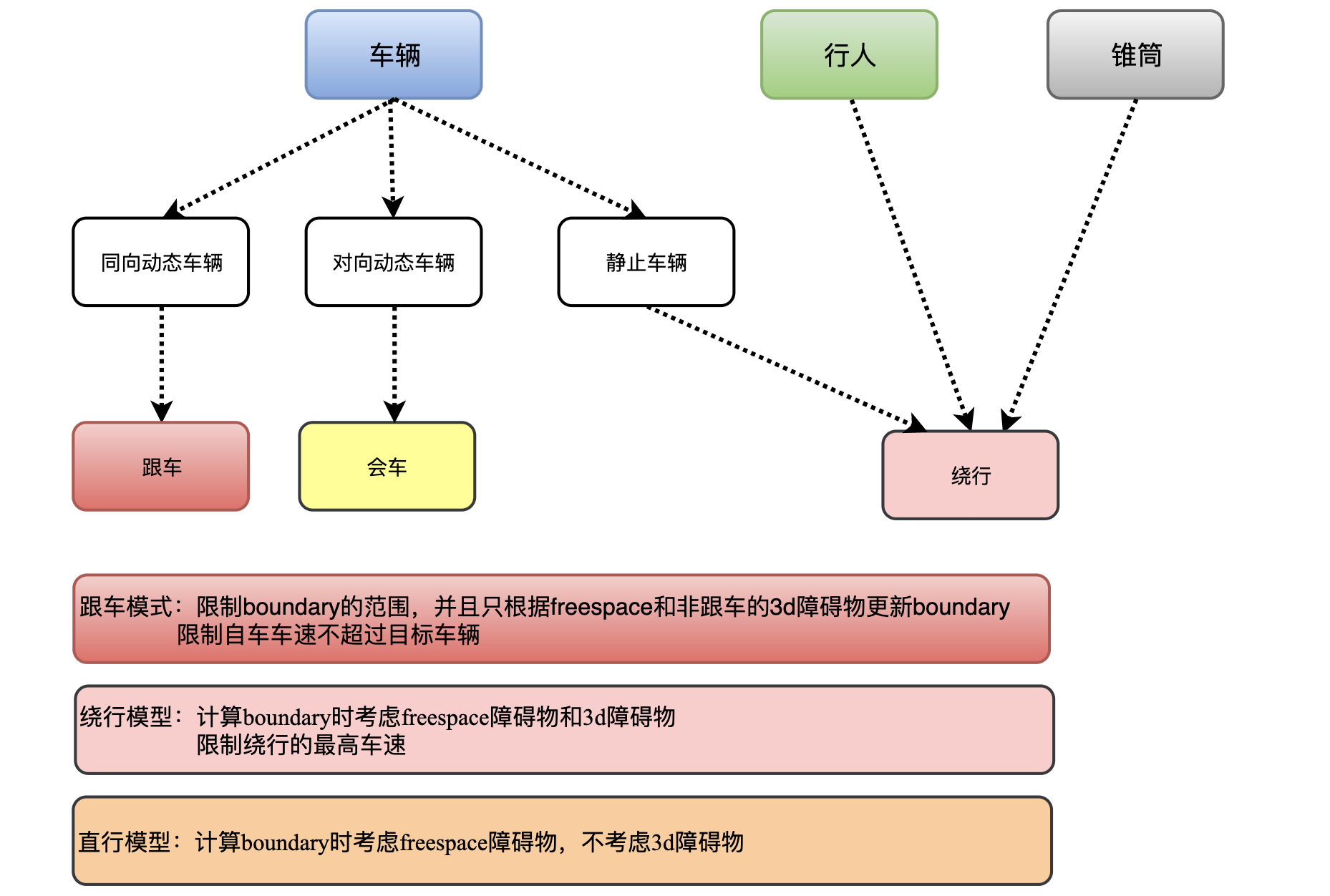
7．规划的状态：path reuse等规划状态，在path reuse阶段保持上一时刻的行为决策结果不变

**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：**

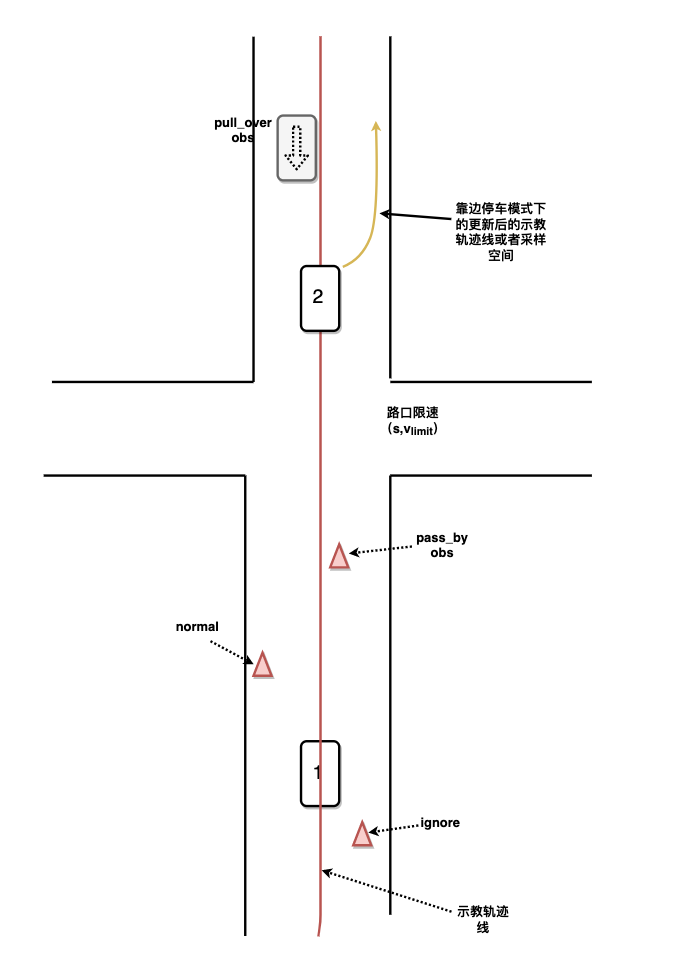
对绕行状态设置一个速度上线（10km/h，TBD）???

**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

****

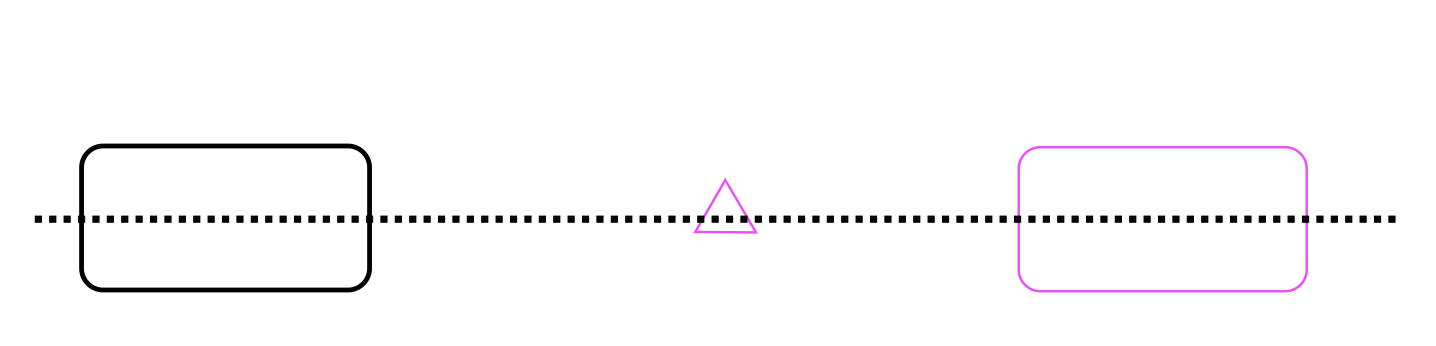
场景决策后需要增加能力校验（根据规划能力）

**六、具体场景**

Case1:

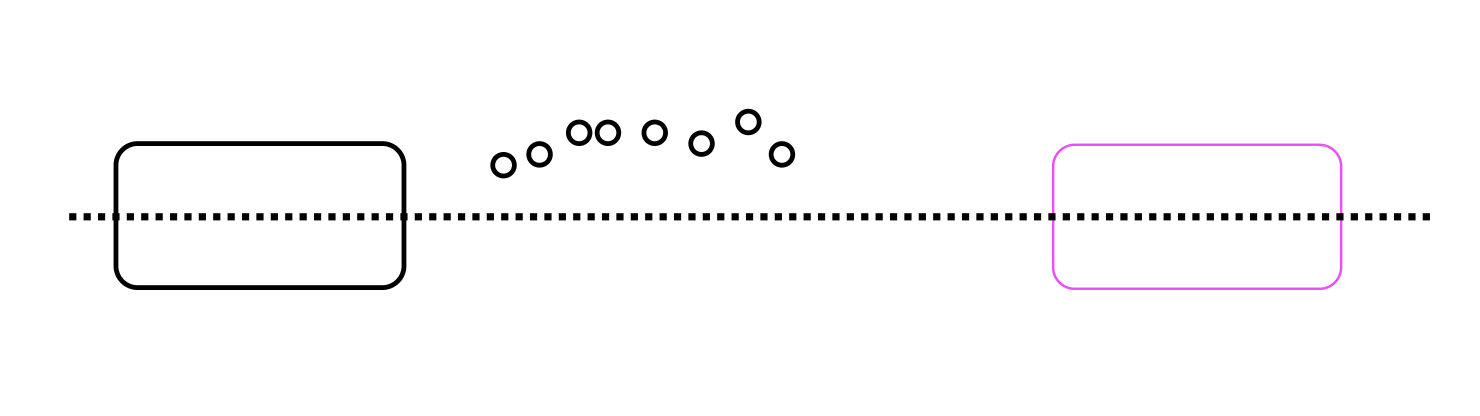
下面这种场景下需要能够先绕行再跟车，跟车时不能绕行

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



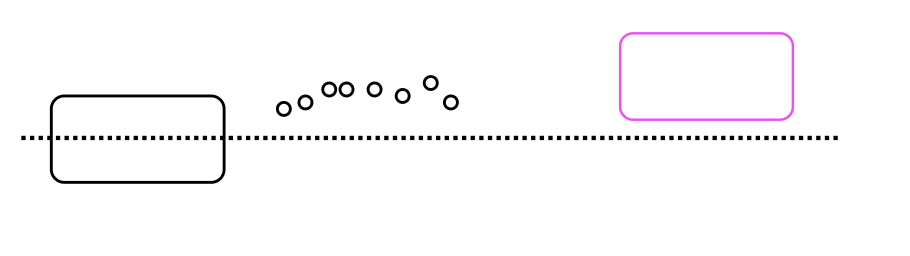
Case2:

这种场景下需要能够在跟车过程中绕行



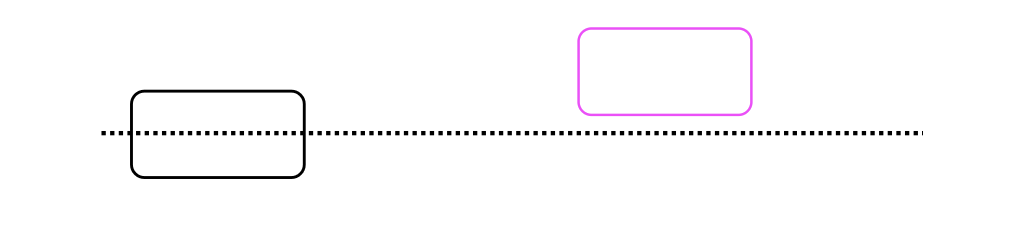
Case3

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



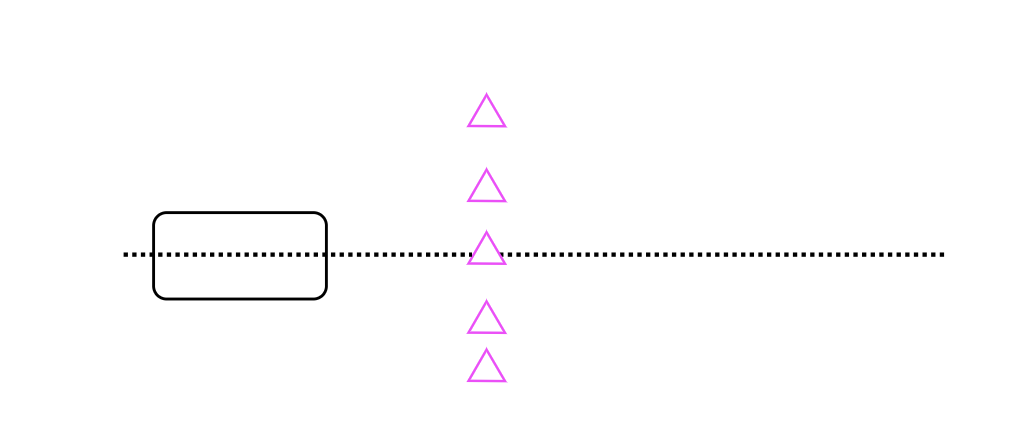
Case4

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