Epslion论文阅读理解

绪论

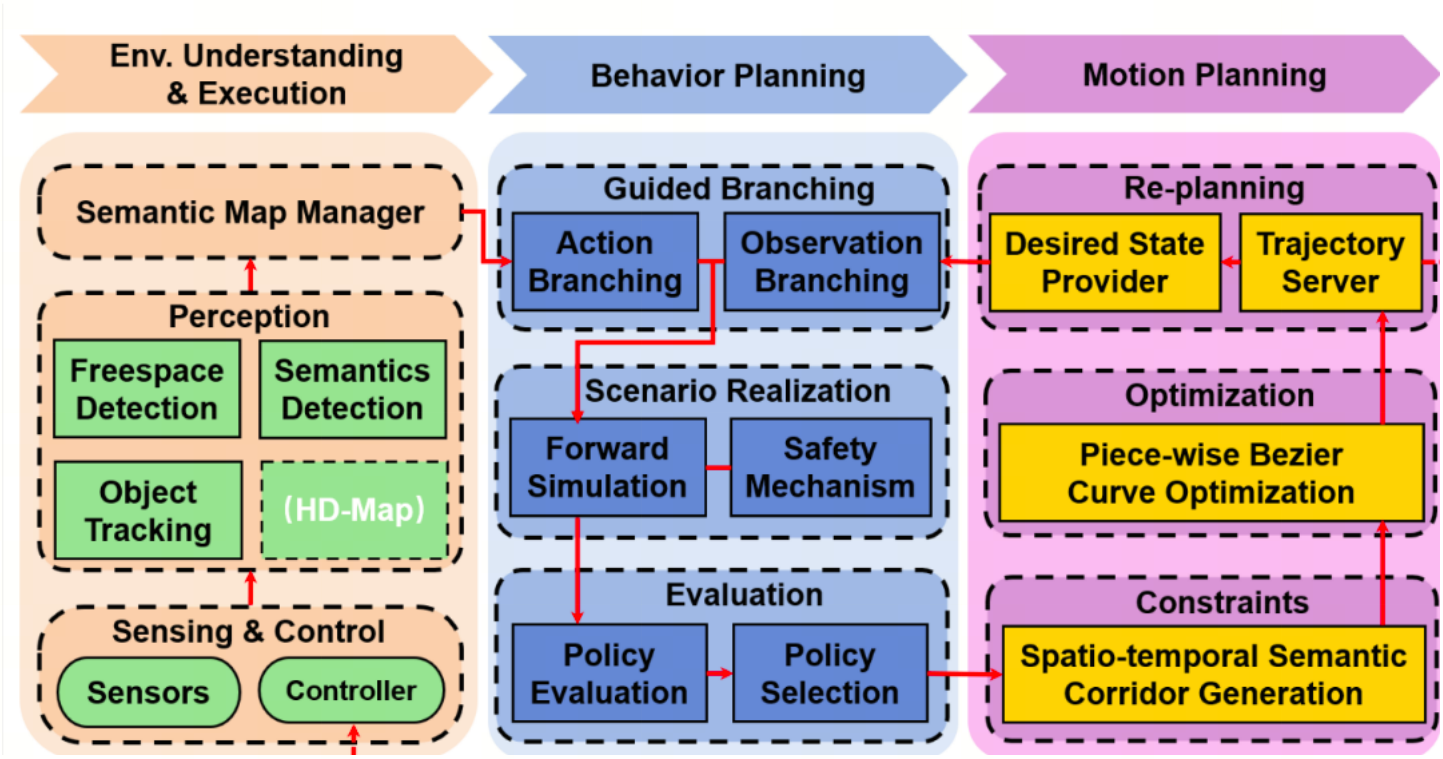
Epslion即**E**fficient **P**lanning **S**ystem for automated vehicles In high**L**y **i**nteractive envir**ON**ments的缩写，通俗点理解就是考虑环境周围障碍物下的轨迹规划系统。

Epslion系统主要包含两个模块，分别是behavior-planner层以及motion-planner层。

behavior-planner层的主要目的就是根据自车的状态，周围的环境(障碍物，交通灯)，驾驶员意图等信息，生成一条收益最高的初始轨迹；

motion-planner层的主要目的就是根据behavior-planner提供的初始轨迹，对其进行二次优化，使其满足可行性,安全性等约束，最终下发给执行机构。

架构



预准备知识

MDP(马尔可夫决策过程)

马尔可夫过程

马尔可夫过程（Markov process）指具有马尔可夫性质(当且仅当某时刻的状态只取决于上一时刻的状态时，一个随机过程被称为具有**马尔可夫性质**)的随机过程，也被称为**马尔可夫链**（Markov chain）。我们通常用元组(S,P) 描述一个马尔可夫过程，其中S是有限数量的状态集合，p是**状态转移矩阵**（state transition matrix）。假设一共有n个状态(s_{1},s_{2},...,s_{n})，此时。状态转移矩阵P定义了所有状态对之间的转移概率，即

$$p = \begin{bmatrix} p(s_1|s_1) & p(s_2|s_1) & \cdots & p(s_n|s_1) \\ p(s_1|s_2) & p(s_2|s_2) & \cdots & p(s_n|s_2) \\ \cdots & \cdots & \cdots & \cdots \\ p(s_1|s_1) & p(s_2|s_1) & \cdots & p(s_n|s_1) \end{bmatrix}$$

马尔可夫奖励过程(MRP)

在马尔可夫过程的基础上加入奖励函数R和折扣因子\gamma，就可以得到**马尔可夫奖励过程**（Markov reward process）。一个马尔可夫奖励过程由(S,P,R,\gamma)构成，各个组成元素的含义如下所示：

- $\mathcal{S}$是有限状态的集合。
- $\mathcal{P}$是状态转移矩阵。
- r是奖励函数，某个状态s的奖励 $r(s)$指转移到该状态时可以获得奖励的期望。
- γ是折扣因子（discount factor），γ的取值范围为$[0,1)$。引入折扣因子的理由为远期利益具有一定不确定性，有时我们更希望能够尽快获得一些奖励，所以我们需要对远期利益打一些折扣。接近 1 的γ更关注长期的累计奖励，接近 0 的γ更考虑短期奖励。

收益Return

在一个马尔可夫奖励过程中，从第t时刻状态开始，直到终止状态时，所有奖励的衰减之和称为**回报**G(t)（Return）

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$$

举例

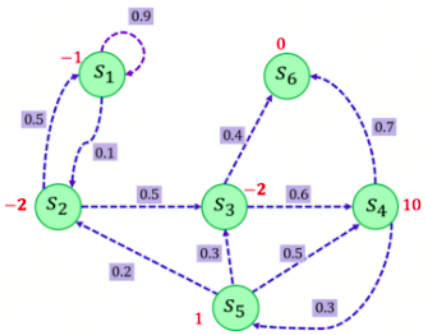


图3-2 马尔可夫奖励过程的一个简单例子

比如选取s_1为起始状态，设置$\gamma = 0.5$，采样到一条状态序列为$s_1 \to s_2 \to s_3 \to s_6$，就可以计算$s_1$的回报$G_1$，得到$G_1 = -1 + 0.5 \times (-2) + 0.5^2 \times (-2) = -2.5$。

马尔科夫决策过程

在马尔科夫奖励过程加入动作，就构成了马尔科夫决策过程

每个浅色小圆圈旁的数字代表在某个状态下采取某个动作能获得的奖励。虚线箭头代表采取动作后可能转移到的状态，箭头边上的数字代表转移概率，如果没有数字则表示转移概率为 1。例如，在s_2下， 如果采取动作"前往s_3"，就能得到奖励-2，并且转移到s_3；在s_4下，如果采取"概率前往"，就能得到奖励 1，并且会分别以概率 0.2, 0.4, 0.4 转移到s_2, s_3或s_4。

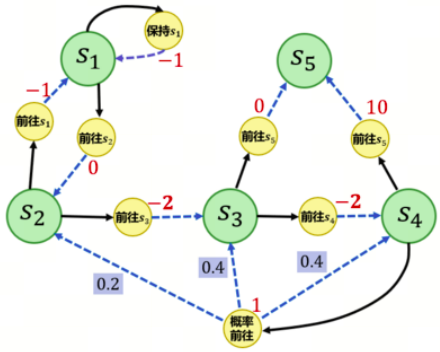


图3-4 马尔可夫决策过程的一个简单例子

behavior-planner

绪论

epslion中的behavior-planner就是利用POMDP(部分可观测马尔科夫决策过程)，在城市环境中大量与自车存在高交互度的障碍物的基础下，对这些障碍物的意图以及与自车的意图进行评价收益，最后选择收益最大的一个动作，作为本次规划过程的最优决策过程，然后根据预定义的控制器，离散化5s的时间，生成初步的轨迹。

算法框架

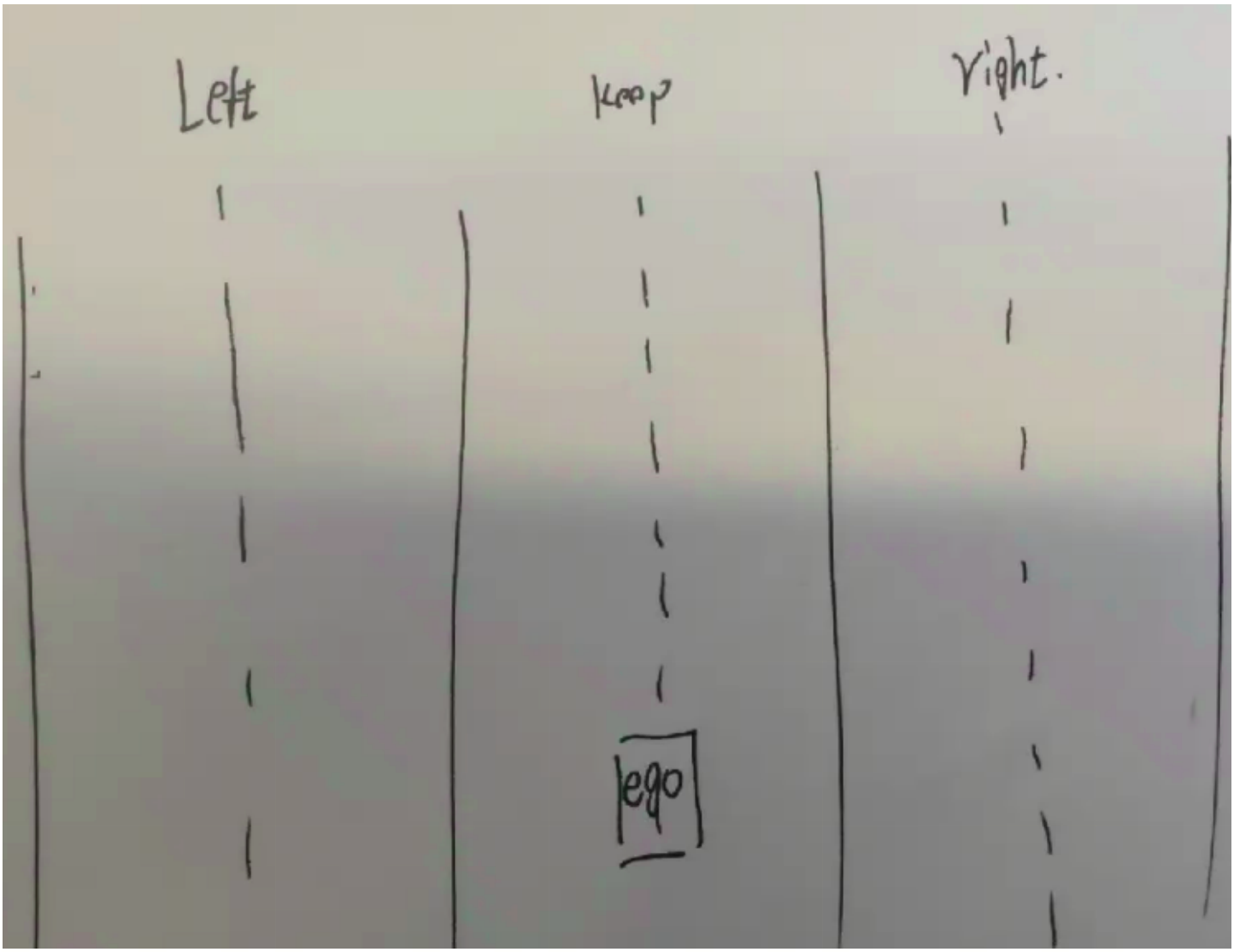
Algorithm 1: Process of behavior planning layer

Inputs: Current states of ego and other vehicles x; *Ongoing action* $\hat{\phi}$; Pre-defined semantic action set Φ; Planning horizon H;
$\mathfrak{R} \leftarrow \emptyset$; // set of rewards for each policy;
$\Psi \leftarrow \texttt{UpdateDCPTree}(\Phi, \hat{\phi})$; // DCP-Tree Ψ;
$\hat{\Pi} \leftarrow \texttt{ExtractPolicySequences}(\Psi)$;
foreach $\pi \in \hat{\Pi}$ **do**
    $\Gamma^{\pi} \leftarrow \emptyset$; // set of simulated trajectories;
    $\Omega \leftarrow \texttt{CFB}(x, \pi)$; // set of critical scenarios;
    foreach $\omega \in \Omega$ **do**
        $\Gamma^{\pi} \leftarrow \Gamma^{\pi} \cup \texttt{SimulateForward}(\omega, \pi, H)$;
    end
    $\mathfrak{R} \leftarrow \mathfrak{R} \cup \texttt{EvaluatePolicy}(\pi, \Gamma^{\pi})$;
end
$\pi^*, \hat{\phi} \leftarrow \texttt{SelectPolicy}(\mathfrak{R})$;

动作空间构建(action branch)

在创建动作空间的时候，论文提出了一种语义级动作。

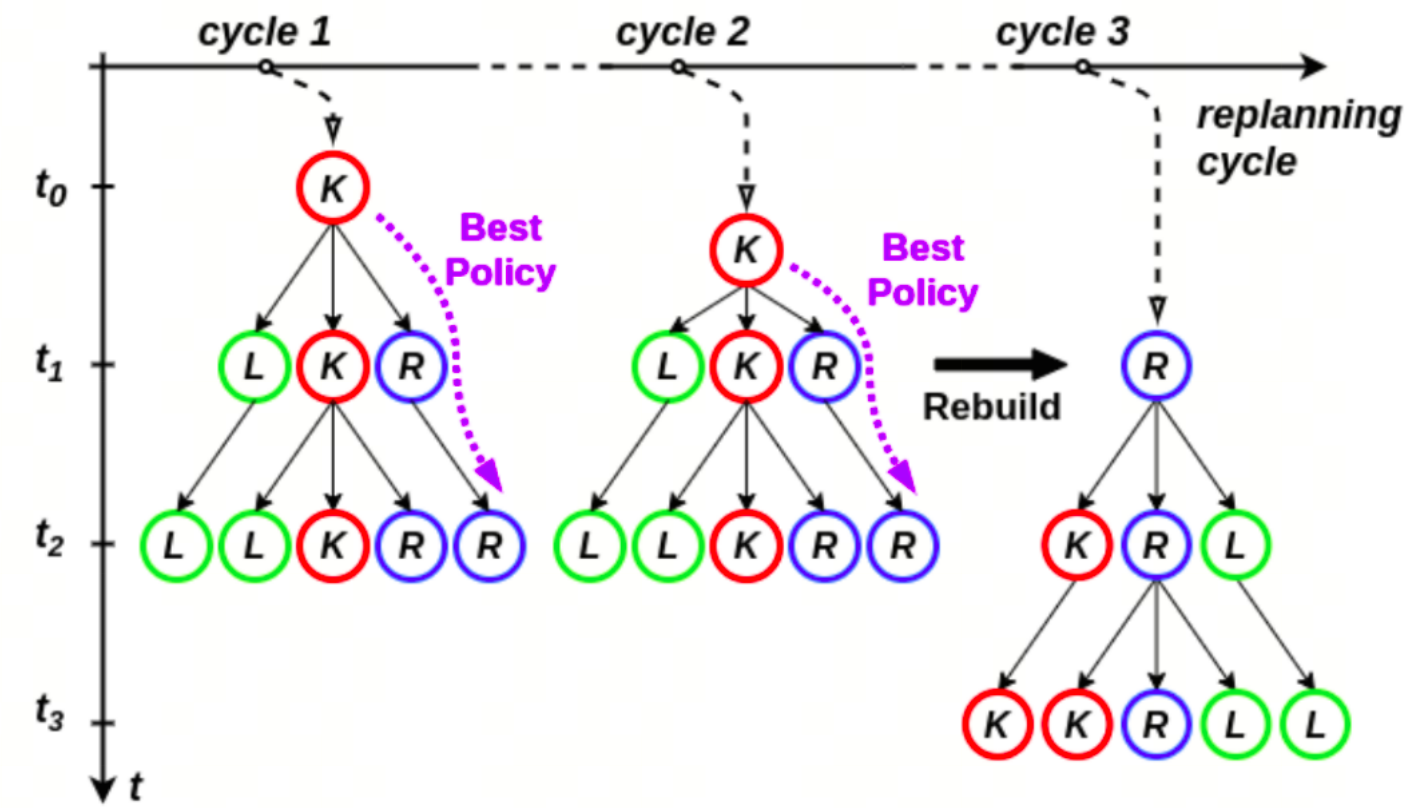
提出语义级动作的前提是，车基本都会遵循交通规则，行驶在某个车道内部，因此根据自车所处的车道位置以及附近车道信息，给出自车可能的车道级动作，比如本车道保持，向左车道行驶，向右车道行驶。如下图所示



如果是两车道，那么本车的动作空间就只会有两个动作：(keep，left|right)

DCP-tree构建

动作列表构建完成，下面就需要构建动作策略序列(通俗讲就是选择的状态序列)，为了构建动作策略序列，论文引入了一种动作结构DCP-tree(domain specific closed-loop policy tree),构建原理如下图



描述

每次DCP-tree构建的初始节点是上一帧规划流程中已经执行的节点，如上图中cycle 1 和cycle 2 中的K，都是上一帧正在执行的动作节点。

在论文中，设置DCP-tree深度为5，每层之间的间隔是1s，如果我们存在每次存在3个动作，那么5层就会存在81个动作序列，如果再将纵向的动作加入进来，那么动作序列会更多，会耗费非常多的计算资源。

因此为了简化动作序列树。论文中提出了一种剪枝的方法：

假设动作严格符合连续性(因为在行驶过程中车不可能一会走左边车道一会又选择走右边车道频繁的切换)，因此从初始动作开始，每个动作序列只能存在一个变化，这样就会大大减少无效动作序列。

DCP-tree构建之后，behavio-planner选择最佳的动作就简化成了从每个动作序列里面选择一个收益最佳的动作序列去执行。

观测模型(observation branch)

在可观测马尔科夫决策过程的定义中有这么一个设定，就是自车的状态以及其他障碍物的状态不一定是完全可观测的，可能有一部分存在很大的噪声，因此导致状态的不准确性，即存在概率性。

但是考虑这么多的状态概率，对MDP问题的求解造成很大的困扰，因为变量增多，求解困难；

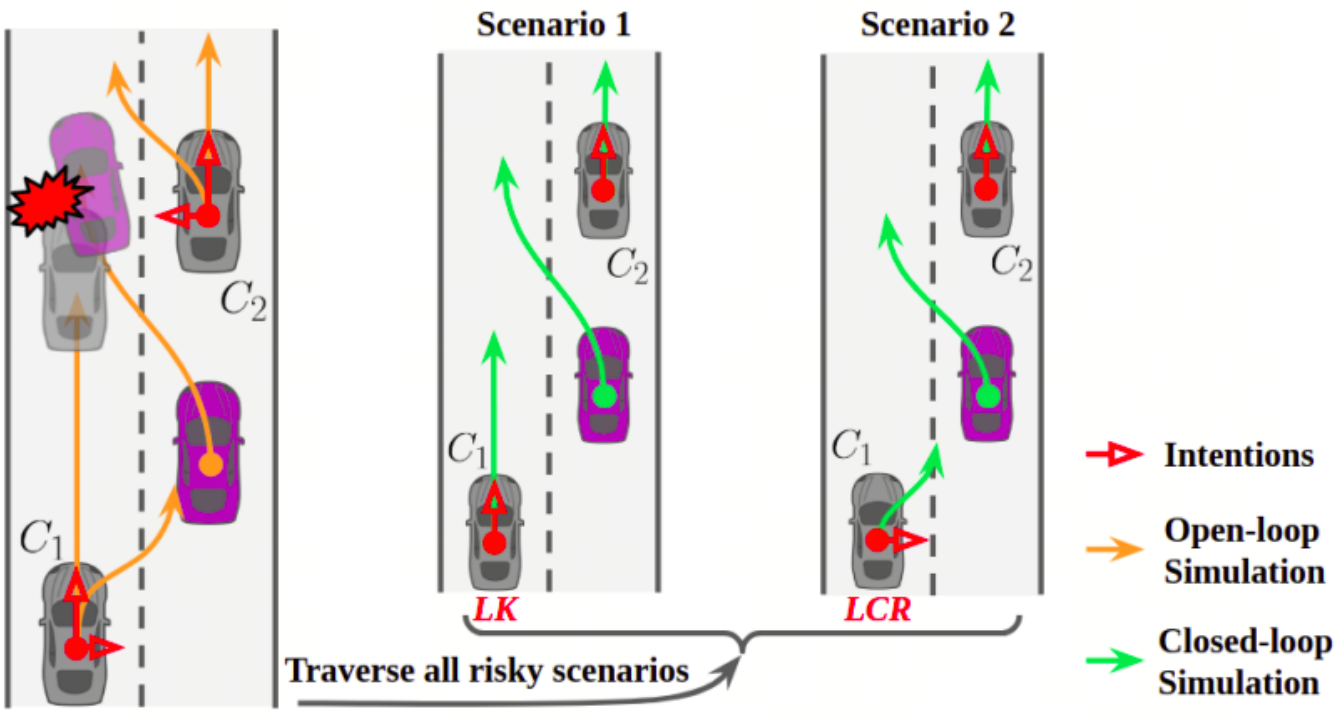
为了解决上述的问题，论文提出了一些简化的假设：

1. 假设**自车的状态是完全准确的**，且观测信息没有噪声(通俗讲就是自车的位置，速度，加速度，朝向角等信息是非常准确的)；

2. 假设**其他agent的意图与初始意图保持一致**(就是说如果在第一时刻认为障碍车辆要左变道，那么在剩余的时刻障碍物车辆的意图都是在左边车道行驶)

根据以上假设每个障碍物的每个意图组合起来，就可以生成多个场景。

但是环境中的障碍物有很多，但是不一定是都和自车交互存在危险，因此在构建场景之前需要先将高交互度障碍物筛选出来，然后对这几个障碍物的意图进行组合，构成多个场景(即算法中的**scenario**)，如下图所示：



现在动作序列有了，场景也有了，那么就需要对障碍物以及自车的状态进行采样离散(即**生成上图中的轨迹**)，为了使这些状态轨迹更加真实，因此论文中采用了提前定义好的控制方法去离散采样，即前向仿真模型(Multi-agent Forward Simulation)。

Multi-agent Forward Simulation

在论文中，针对自车提出了一种context-aware的控制算法，而针对其他的agent，论文采用的是之前一篇论文提出的简单模型控制方法。

context-aware控制算法：

横向：变形的pp控制算法

纵向：gap-informed的速度控制器

simple-model-based控制算法

横向：纯pp控制器

纵向：基于IDM(intelligent driver model)的速度控制器

解析：

如果当前的action是lane-keep,则直接采用简单模型控制方法,横向的控制(即方向盘的角度)直接采用pp控制算法，纵向采用idm的速度控制器；

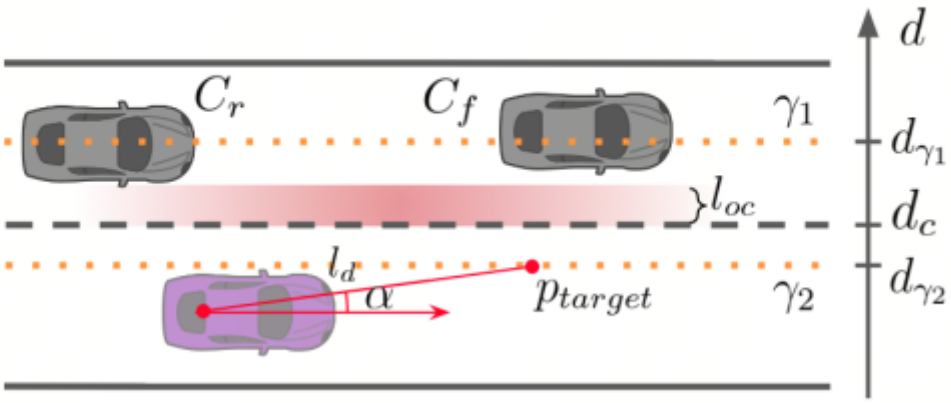
idm公式如下：

自车**加速度方程**：(ego vehicle)

$$\dot{v} = a[1 - \left(\frac{v}{v_0}\right)^{\delta} - (\frac{s^*(v,\ \Delta v)}{s})^2]$$

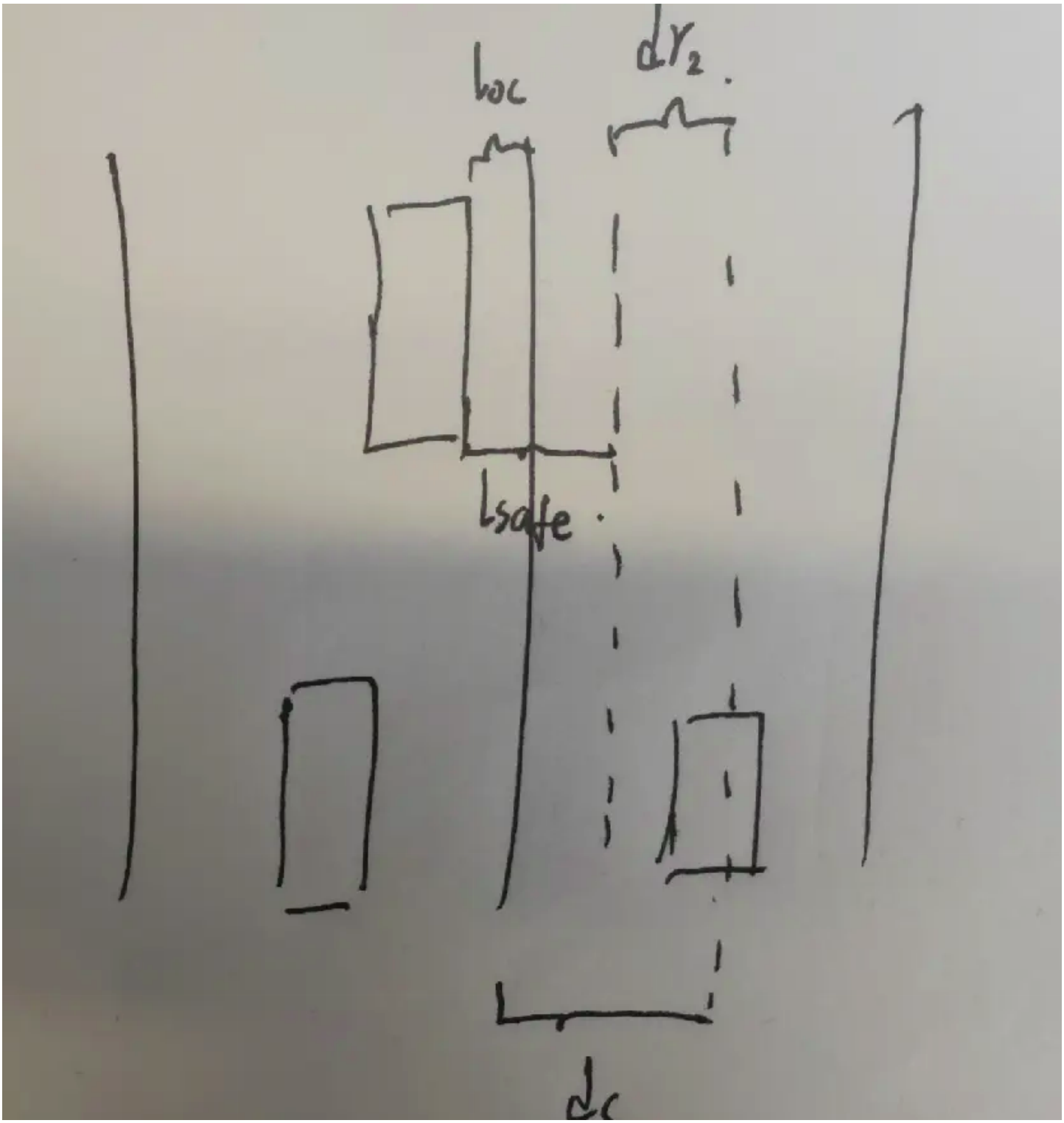
其中 a 为自车的最大加速度， v 为自车当前的车速， v0 为自车的期望车速， δ 为加速度指数， Δv 为自车与前车的速度差， s 为当前自车与前车的车距， s*(v,Δv) 为期望跟车距离。

如果当前的action属于lane-change,则采用context-aware的控制算法，横向采用变形的pp控制器，介绍如下



如果目标车道存在足够的变道空间，则直接采用目标车道中心线上的点作为pp的控制点(即图中 的r1),如果目标车道存在障碍物，且变道空间不是特别充足，则选择r2的位置作为pp的控制点，r2的计算方式如下所示:

图中loc表示障碍物与目标车道线的最短距离，lsafe表示提前规定的变道期间自车与障碍物的最短距离，dc表示自车与车道边线的距离



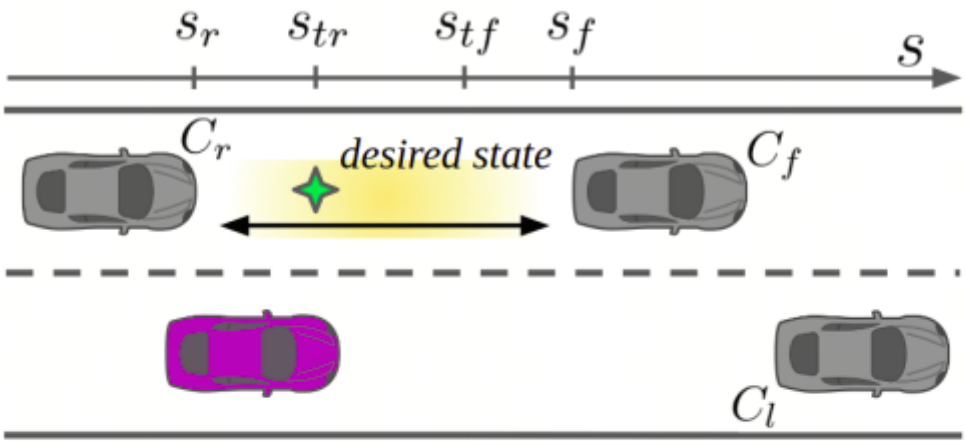
纵向采用选择gap的方式，计算加速度

$$a_{track} = K_v(\dot{s}_{des} + K_s(s_{des} - s_{ego}) - \dot{s}_{ego}),$$

$$s_{des} = \min(\max(s_{tr}, s_{ego}), s_{tf})$$
$$\dot{s}_{des} = \min(\max(\dot{s}_r, \dot{s}_{pref}), \dot{s}_f),$$

$$\begin{bmatrix} s_{tr} \\ s_{tf} \end{bmatrix} = \begin{bmatrix} s_r + l_{min} + T_{safe}\dot{s}_r \\ s_f - l_{min} + T_{safe}\dot{s}_{ego} \end{bmatrix},$$

obtained once the target point p_{target} is determined.



(b) The gap-informed longitudinal motion controller for lane-changing action. We consider three vehicles during lane-changing maneuvers, which are the current leading vehicle C_l and the new leader and follower C_f and C_r. The target gap formed between C_f and C_r is shown in yellow, while the desired longitudinal state is marked by the green star.

安全机制(safe-mechanism)

因为道路上的其他车辆并不一定会以我们预想的轨迹去执行，比如近距离加塞，多车道变道，急加速变道等情况，这样的情况会给我们提前计算好的轨迹带来安全风险，因此为了提高初始behavior的可执行性，需要引入安全机制。

论文中引入两个方法提高轨迹的安全性能，分别如下：

1. 检测自车轨迹的每一个点是否满足RSS模型(责任安全模型，根据障碍物位置给出一个位置范围，速度范围，加速度范围等信息)，如果不满足，则将超出安全范围的控制信号以RSS模型的方式覆盖输出);
2. 在选择出最佳behavior后还会选择一个backup的behavior(例如变道的时候会选择一个放弃变道的behavior)

筛选最佳动作序列

为了挑选最佳的动作序列，论文中提出了3个评价的cost，分别是效率(efficiency)cost，安全(safe)cost，以及导航一致性(navigation)cost

$$F_{total} = \lambda_1 F_e + \lambda_2 F_s + \lambda_3 F_n,$$

Efficiency-cost

效率cost,论文中是以自车速度与cipv速度还有期望速度来进行判断的，计算公式如下：

$$F_e = \sum_{i=0}^{N_a} F_e^i = \sum_{i=0}^{N_a} (\lambda_e^p \Delta v_p + \lambda_e^o \Delta v_o + \lambda_e^l \Delta v_l),$$

$$\Delta v_p = |v_{ego} - v_{pref}|.$$

$$\Delta v_o = \max(v_{ego} - v_{lead}, 0),$$

$$\Delta v_l = |v_{lead} - v_{pref}|.$$

自车的速度和期望速度之差的绝对值、自车车速和cipv车速之差的绝对值，以及cipv车速和自车期望车速绝对值之差

Safe-cost

安全cost。论文中由两部分组成，一部分是检测自车轨迹状态点与周围障碍物的轨迹点是否存在碰撞，如果存在碰撞，则给一个非常大的惩罚；另一部分是检测每一个动作最终状态的的速度是否满足RSS模型；

计算公式如下：

$$\begin{aligned} F_s &= \lambda_s^c b_c + \sum_{i=0}^{N_s} b_r F_s^i \\ &= \lambda_s^c b_c + \sum_{i=0}^{N_s} b_r v_{ego} \lambda_s^{r1} e^{\lambda_s^{r2} |v_{ego} - \min(\max(v_{ego}, v_{rss}^{lb}), v_{rss}^{ub})|}, \end{aligned}$$

下式中的b_c就是1，然后\lambda_sc表示碰撞的惩罚项

navigation-cost

导航cost表示的是决策和驾驶员意向的契合度,b是两个bool值，两个lambda表示的是惩罚项

$$F_n = \lambda_n^{user} b_{navi} + \lambda^{consist} b_{consist},$$

最终选择cost最小的动作树，作为最优的behavior，发送给motion-planner层

motion-planner

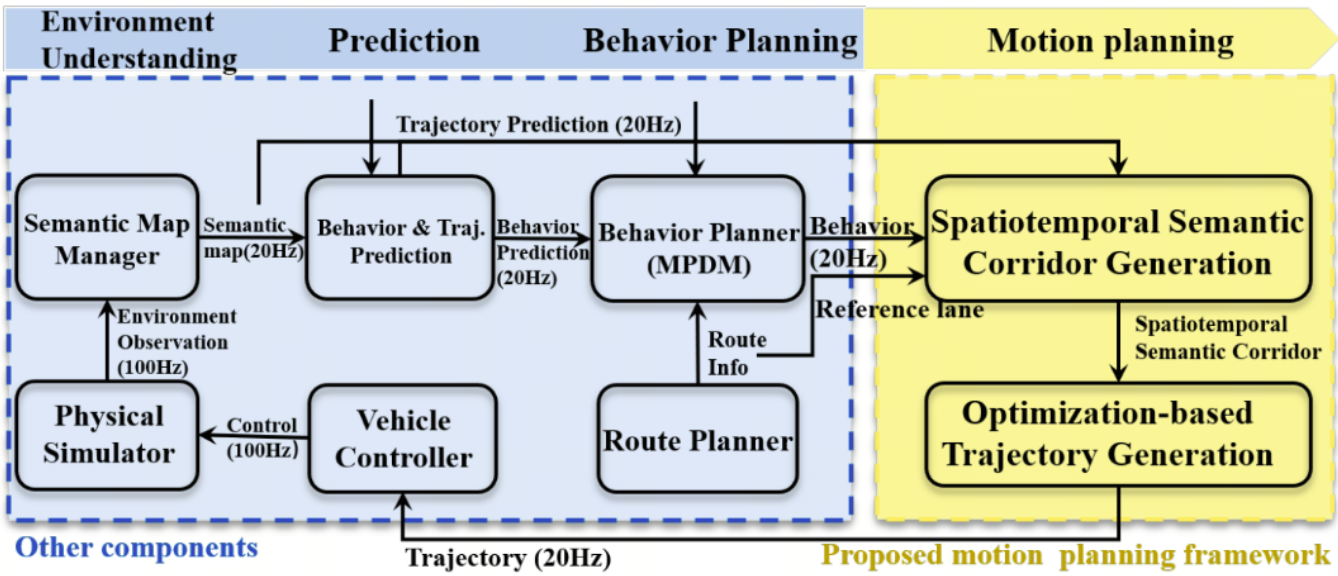
绪论

SSC即**spatio-temporal semantic corridor structure**-时空联合语义走廊，顾名思义，即根据道路上的一些语义元素(动态障碍物，lane_boundary,traffic-light以及限速信息等)， 将其在时空域的特定范围内，转化为时空障碍或约束，根据这些障碍物以及约束搜索safe-corridor，然后根据先验的轨迹状态点对corridor进行剪枝、膨胀、链接等操作，最后将其做为一个硬约束条件，进行二次优化，生成一个绝对安全的时空轨迹

论文核心：

给定统一的SSC表示，轨迹生成问题归结为在满足动力学约束的情况下在SSC内生成最优轨迹

架构



输入

构建SSC需要的四个输入

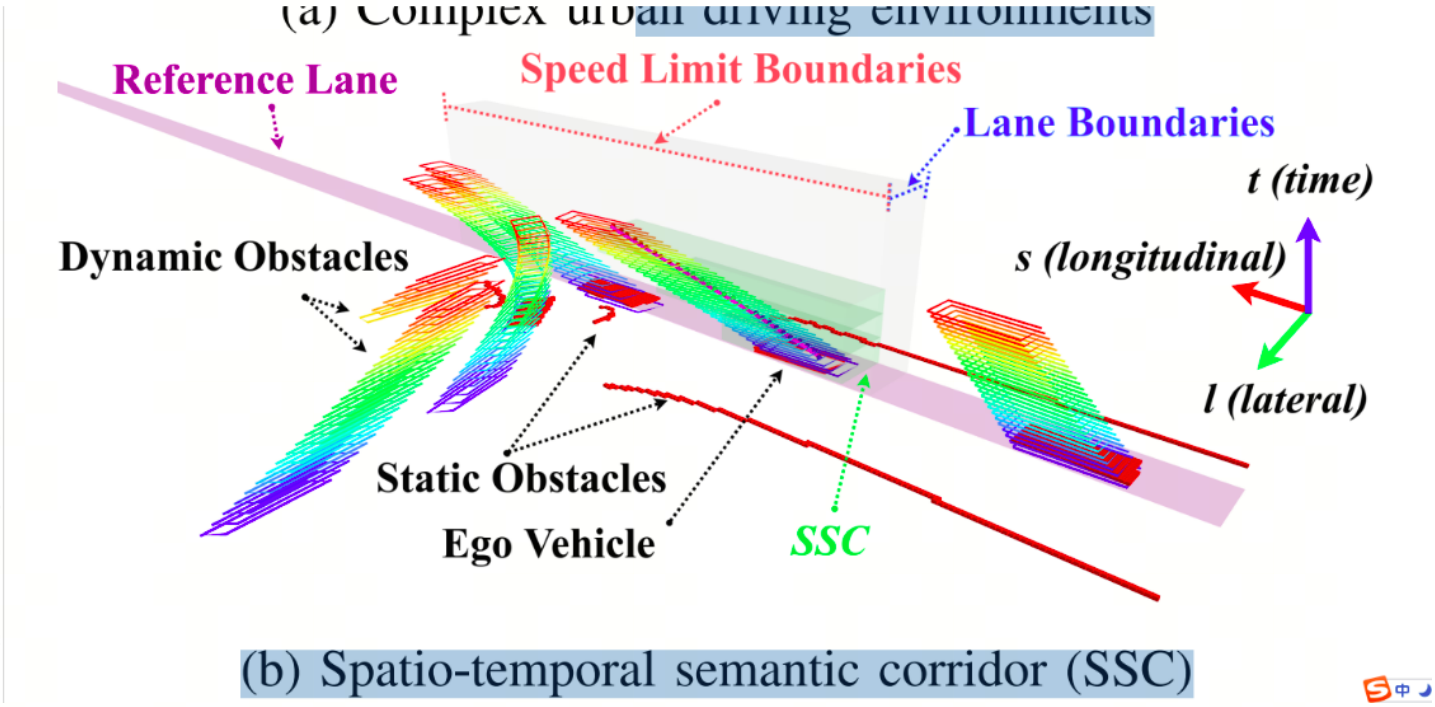
1 **语义栅格地图**

2 **动态障碍物**的预测轨迹，以及**静态障碍物**等信息

3 自车的**初始轨迹**(由behavior-planner输入)

4 **引导线**(用来生成frenet坐标系)

坐标体系



如上图所示，SSC构建的safe-corridor是一个三维立体的结构，且是布置在s,l坐标系下的。因为自动驾驶车辆基本都是行驶在结构化的道路上，结构化的道路拥有比较明确的几何形状

SSC地图构建

SSC将所有的道路参与者区分为两类：

障碍物类型以及约束类型

障碍物类型

顾名思义，障碍物类型的道路参与者就是表示无论在什么情况下，自车都不可以行驶到的区域，如果行驶过去则有碰撞的风险，比如道路上的障碍物，施工区域，交通灯区域(红灯)

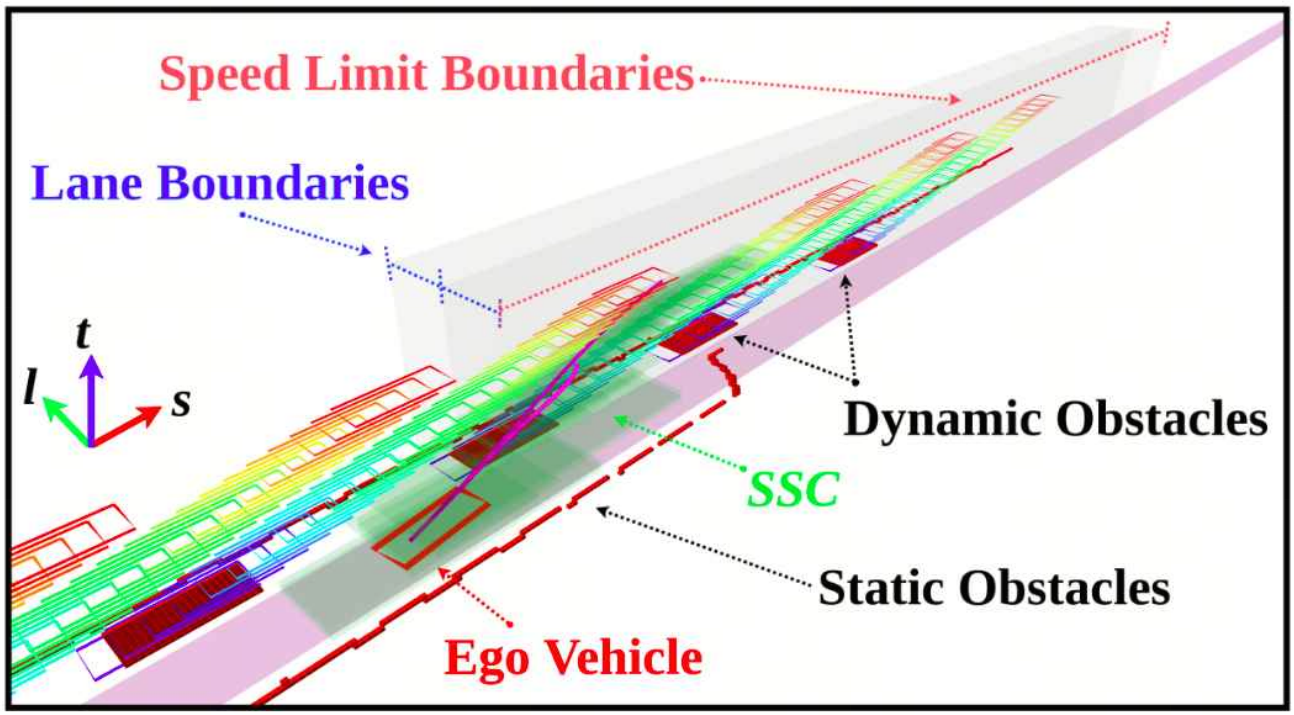
约束类型

约束类型指的就是在一般情况下尽量不要行驶到的区域，可以应用在速度限制等方面

举两个例子，如下图

静态障碍物，动态障碍物在地图中的表示

以及速度限制在地图中的表示



通过将所有的交通参与者以占据栅格的方式统一表示在局部地图中，建立SSC地图

SSC-corridor构建

地图建立之后就是开始构建safe-corridor

算法流程

Algorithm 1: Semantic Corridor Generation

Inputs: forward simulated states $\{x_0, x_1, \ldots, x_t\}$, initial state x_{des}, semantic boundaries $\mathcal{B}$, slt configuration space $\mathcal{E}$;

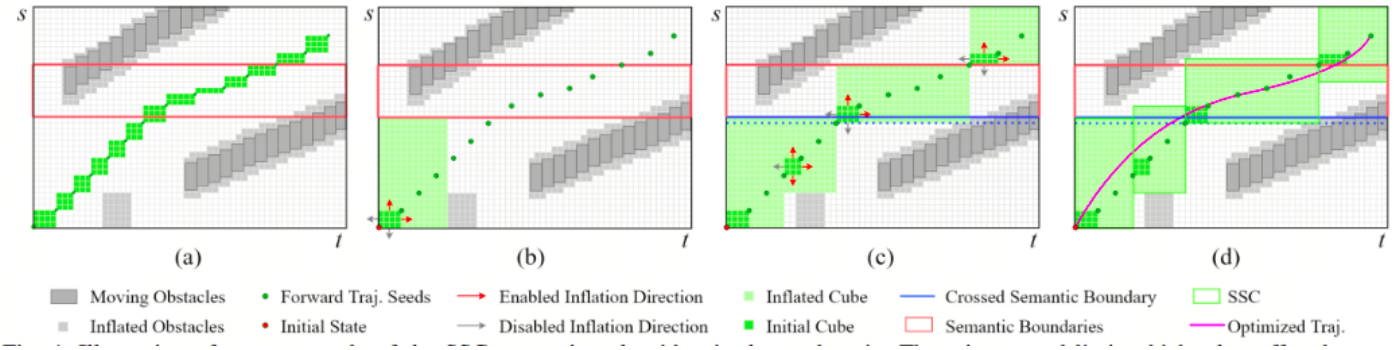
Initializes: seeds $\mathcal{S}^{\text{seed}} = \emptyset$;

$\mathcal{S}^{\text{seed}} \leftarrow \texttt{SeedGeneration}(\{x_0, x_1, \ldots, x_t\}, x_{\text{des}})$;

$\mathcal{C}^{\text{infl}} \leftarrow \texttt{CubeInflation}(\mathcal{S}^{\text{seed}}, \mathcal{B}, \mathcal{E})$;

$\mathcal{C}^{\text{infl}} \leftarrow \texttt{ConstraintAssociation}(\mathcal{C}^{\text{infl}}, \mathcal{B})$;

$\mathcal{C}^{\text{final}} \leftarrow \texttt{CubeRelaxation}(\mathcal{C}^{\text{infl}}, \mathcal{E})$;



后端二次优化

贝塞尔曲线介绍

贝塞尔曲线（Bézier）
1.公式

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i$$

其中 P_i 为贝塞尔曲线的**控制点**，由这些控制点确定了贝塞尔曲线的形态。 $B_{n,i}$ 为贝塞尔曲线的基函数，是一个多项式，公式为：

$$B_{n,i} = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}$$

其中 n 表示贝塞尔曲线的**阶**， i 表示**控制点**的下标 $i \in [0, n]$ ，**控制点个数=阶数+1**。

特性

点）、三阶贝塞尔曲线（4个控制点）等等。

特点一：曲线通过始点和终点，并与特征多边形首末两边相切于始点和终点，中间点将曲线拉向自己。

特点二：平面离散点控制曲线的形状，改变一个离散点的坐标，曲线的形状将随之改变（点对曲线具有整体控制性）。

特点三：曲线落在特征多边形的凸包之内，它比特征多边形更趋于光滑。

特性四：贝塞尔曲线的导数还是贝塞尔曲线，只不过是控制点是原来控制点的组合而已.

优化方式

因为我们可能生成多个corridor，因此我们采用分段贝塞尔曲线，公式如下

$$f_j^{\sigma}(t) = \begin{cases} \alpha_1 \cdot \sum_{i=0}^{m} p_i^1 \cdot b_m^i(\frac{t-t_0}{\alpha_1}), & t \in [t_0, t_1] \\ \alpha_2 \cdot \sum_{i=0}^{m} p_i^2 \cdot b_m^i(\frac{t-t_1}{\alpha_2}), & t \in [t_1, t_2] \\ \vdots & \vdots \\ \alpha_n \cdot \sum_{i=0}^{m} p_i^n \cdot b_m^i(\frac{t-t_{n-1}}{\alpha_n}), & t \in [t_{n-1}, t_n], \end{cases}$$

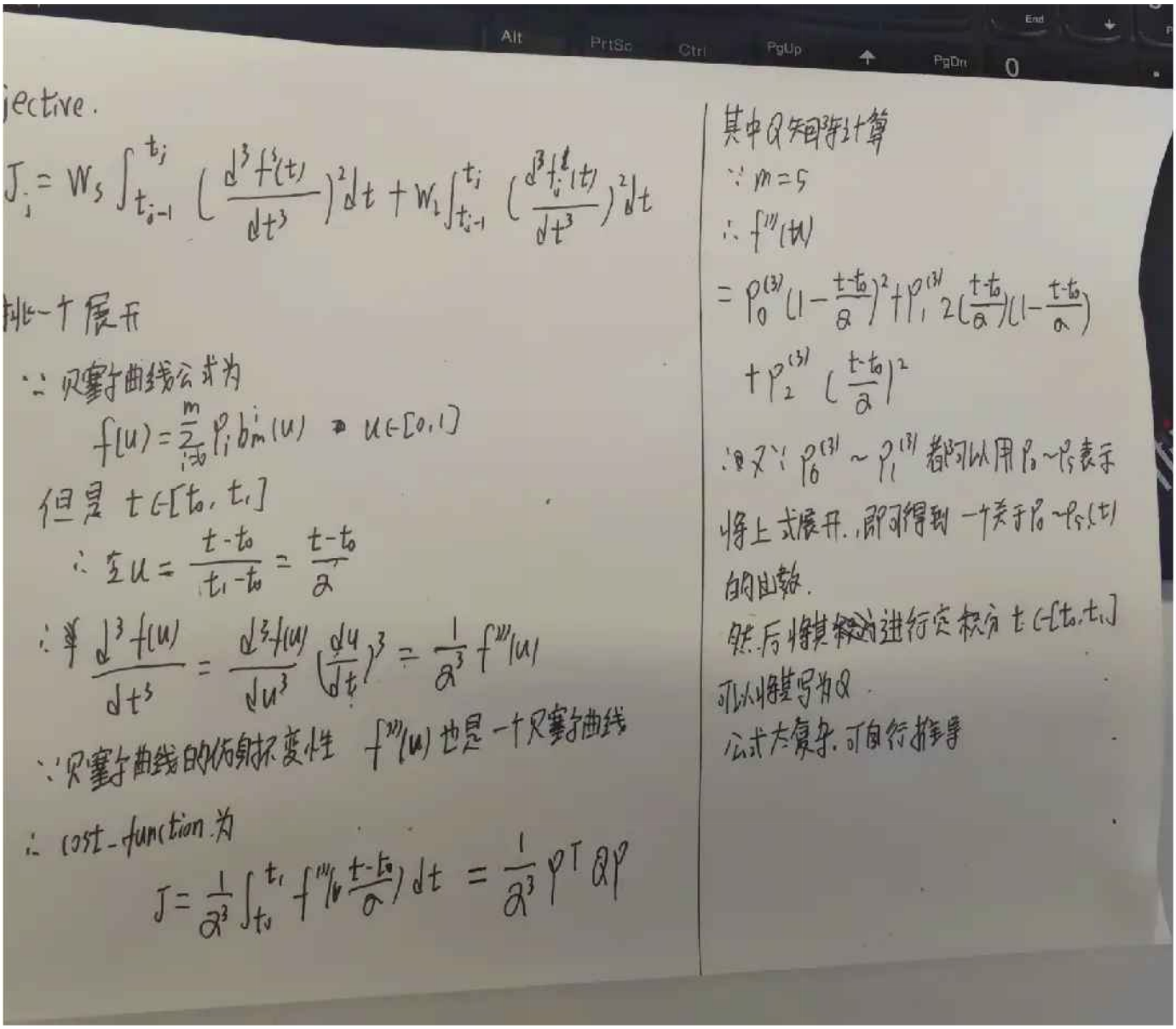
目标函数

minimum jerk

non-scaled $y_j^{\omega}(t)$ as follows,

$$J_j^{\sigma} = \int_0^1 \alpha_j \cdot \left(\frac{d^3(\alpha_j \cdot y_j^{\sigma}(t))}{d(u \cdot \alpha_j)^3} \right)^2 du = \frac{1}{\alpha_j^3} \cdot \mathbf{p}_j^{\mathrm{T}} \mathbf{Q} \mathbf{p}_j,$$

公式推导如下：



约束

起始点终止点约束

因为初始轨迹是带有时间属性的轨迹点，因此t0以及tn是已知，优化变量是控制点，因此起始点的p,v，a与第一段贝塞尔曲线的0-2阶导数值是一致的

因为初始轨迹是带有时间属性的轨迹点，因此t0以及tn是已知，优化变量是控制点，因此起始点的p,v，a与第一段贝塞尔曲线的0-2阶导数值是一致的

根据贝塞尔曲线特性，即0阶导数的第一个控制点等于起始点的p=$\alpha * p0$,以此类推，v = $m(p1-p0), a=m(m-1)\frac{1}{\alpha}(p2-2p1+p0)$

最后一个点类似以上过程

最后一个点类似以上过程

Based on this property, the k-th-order derivatives at the boundaries of $f_j^\sigma(t)$ can be expressed as

$$\frac{d^k f_j^\sigma(t_{j-1})}{dt^k} = \alpha_j^{1-k} \cdot q_{j,0}^{\sigma,(k)}, \frac{d^k f_j^\sigma(t_j)}{dt^k} = \alpha_j^{1-k} \cdot q_{j,m}^{\sigma,(k)}, \quad (5)$$

th segment can be written as

$$\frac{d^k f_j^\sigma(t_j)}{dt^k} = \frac{d^k f_{j+1}^\sigma(t_j)}{dt^k},$$

连续性约束

因为我们利用分段贝塞尔优化轨迹，因此每段贝塞尔曲线的连接点的P,v,a应该是相等的

而由于贝塞尔曲线的特性，曲线一定经过第一个点和最后一个点，所以利用控制点可以很轻易的写出等式关系

Based on this property, the k-th-order derivatives at the boundaries of $f_j^\sigma(t)$ can be expressed as

$$\frac{d^k f_j^\sigma(t_{j-1})}{dt^k} = \alpha_j^{1-k} \cdot q_{j,0}^{\sigma,(k)}, \frac{d^k f_j^\sigma(t_j)}{dt^k} = \alpha_j^{1-k} \cdot q_{j,m}^{\sigma,(k)}, \quad (5)$$

th segment can be written as

$$\frac{d^k f_j^\sigma(t_j)}{dt^k} = \frac{d^k f_{j+1}^\sigma(t_j)}{dt^k},$$

免碰撞约束

该约束就是我们上一步构建的corridor，硬约束关于p每个控制点在我们构建的corridor里面，即可保证优化轨迹是safe的

动力学约束

该约束就是我们设计的最大最小的速度和加速度，利用贝塞尔曲线的导数也是贝塞尔曲线的性质，将一阶导数的控制点都处于速度区间，二阶导数都处于加速度区间即可

将以上以及cost-function构建成等式约束以及不等式约束，做二次优化，优化轨迹

参考文献

马尔科夫决策

部分可观马尔科夫决策过程

部分可观测马尔科夫决策过程

IDM智能驾驶模型