深蓝学院
shenlanxueyuan.com

自动驾驶规划与控制第六章作业分享
--LatticePlanner

主讲人 靳小鑫

纲要

- 第一部分：采样方案
- 第二部分：成本函数

采样方案

●横向采样 -- 合理设置道路宽度、采样步长

本作业最初未更改横向采样范围，而是在轨迹筛选时适当排除跑到道路外侧的轨迹。

后考虑可以在横向采样阶段就做出限制，由于本作业只有一条参考线，道路宽度3.75m,故暂时如右下设置横向采样范围。

Apollo的做法是根据可行车道设置多条参考线，对多条参考线均进行采样，即可直接按照道路宽度设置固定的横向采样范围。

```
// 若路径满足速度、加速度、曲率约束且无碰撞，则为可行解
for (FrenetPath path : path_list)
{
  if (path.max_speed < MAX_SPEED && path.max_accel < MAX_ACCEL &&
      path.max_curvature < MAX_CURVATURE && check_collision(path, ob)&&
      path.d.back() < 5.0 && path.d.back() > - 1.5)
  {
    output_fp_list.push_back(path);
  }
}
return output_fp_list;
};
```

```
// 对横向位移 d 进行采样
// for (float di = -1 * MAX_ROAD_WIDTH; di < MAX_ROAD_WIDTH; di += D_ROAD_W) {
for (float di = -1 * 3.75 * 1.5; di < 3.75 * 0.5; di += D_ROAD_W) {
```

```
multiple_best_path.clear();
//对每一条参考线都规划一条最优轨迹
for (int lane_index = 0; lane_index < sub_reference_line.lanes.size(); lane_index++) {
    update_path(sub_reference_line.lanes[lane_index]);
```

```
std::array<double, 7> end_d_candidates = {-1, -0.75, -0.5, 0, 0.5, 0.75, 1};
// std::array<double, 3> end_d_candidates = {-0.5, 0, 0.5};
```

采样方案

深蓝学院 shenlanxueyuan.com

- 纵向采样
  - 纵向时间序列采用 -- 合理设置时间序列长度、采样步长
  - 纵向车速采样 -- 合理设置车速范围、采样步长

本作业中暂时提取单个时间，取Ti = 3.0s,对速度进行纵向采样并调整采样范围以优化计算时间和轨迹长度。

```
// 对纵向时间序列采样
/*for (float Ti = MINT; Ti < MAXT; Ti += DT) */{
  // 当 (di,Ti) 确定后，可获得一条连接当前状态与 (di, Ti) 的五次多项式轨迹曲线
  float Ti = 3.0;
```

```
// 对纵向车速进行采样
for (float tv = TARGET_SPEED - D_T_S * N_S_SAMPLE;
     tv < TARGET_SPEED + D_T_S * 10.0; tv += D_T_S) {
  // 当 (vi, Ti) 确定后，可获得一条连接当前状态和 (vi, Ti) 的四次多项式轨迹曲线
```

Apollo中的横纵向采样序列（经验所得）。

```
std::vector<Condition> end_d_conditions;
// std::array<double, 13> end_d_candidates = {-1.5, -1.25 , -1, -0.75, -0.5, 0, 0.5, 0.75, 1 , 1.25 ,1.5};
std::array<double, 7> end_d_candidates = {-1, -0.75, -0.5, 0, 0.5, 0.75, 1};
// std::array<double, 3> end_d_candidates = {-0.5, 0, 0.5};
// std::array<double, 4> end_s_candidates = {10.0, 20.0, 40.0, 80.0};
std::array<double, 4> end_s_candidates = {5, 10, 15, 20}; //低速
```

纲要

- 第一部分：采样方案
- **第二部分：成本函数**

成本函数

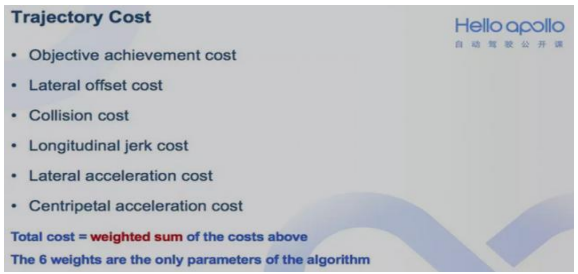
深蓝学院 shenlanxueyuan.com

- 成本函数
  - 横纵向jerk
  - 横纵向最大加速度
  - 横纵向偏移量
  - 时间序列常数
- Apollo的成本函数
  - 目的地cost
  - 横向偏移cost
  - 碰撞cost
  - 纵向jerk_cost
  - 横向加速度cost
  - 向心加速度cost等。

```
// 计算代价函数
float Jp = sum_of_power(fp.d_ddd)/fp.d_ddd.size();      // square of jerk
float Js = sum_of_power(fp_bot.s_ddd)/fp.d_ddd.size(); // square of jerk
float Jp_max = max_of_power(fp.d_dd);
float Js_max = max_of_power(fp_bot.d_dd);

// float Jd = sum_of_power(fp_bot.d);
// square of diff from target speed
float ds = (TARGET_SPEED - fp_bot.s_d.back());

fp_bot.cd = KJ * Jp + KT * Ti + KD * std::pow(fp_bot.d.back(), 2) + 2.0 * Jp_max;
fp_bot.cv = KJ * Js + KT * Ti + KD * ds + 2.0 * Js_max;
fp_bot.cf = KLAT * fp_bot.cd + KLON * fp_bot.cv;
```



https://blog.csdn.net/qq_24649627/article/details/121379155

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening