深蓝学院
shenlanxueyuan.com

第六章作业分享

主讲人 牛一勋

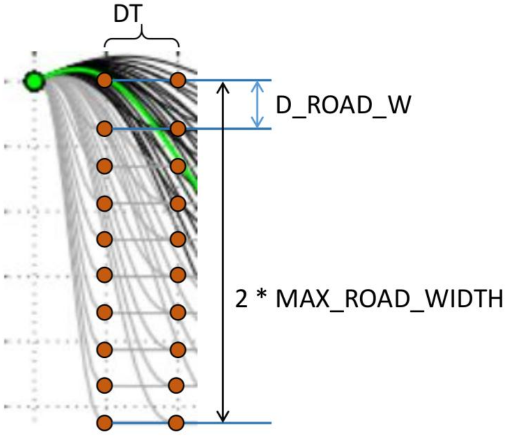
纲要

深蓝学院 shenlanxueyuan.com

- Lateral sampling
- Longitudinal sampling
- Trajectory cost function
- Path selection

Lateral sampling

1. 首先对横向位移（采样距离应限制在车道内）进行采样;
2. 时间进行采样，从而得到末端横向距离固定的不同曲线；
3. 首末端共**6个固定量**，用**五次多项式**拟合成一条用于描述横向采样性质的曲线。



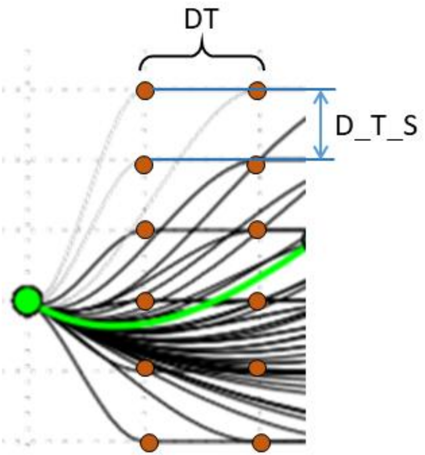

```
for (float di = 0; di <= 5; di += D_ROAD_W) {
    //different curves of end point d
    for (float ti = MINT; ti <= MAXT; ti += DT) {
    // Different driving time under the same d
        FrenetPath fp_without_s;

        QuinticPolynomial lateral_fp(c_d, c_d_d, c_d_dd, di, 0.0, 0.0, ti);

        for (float _t = 0.0; _t <= ti; _t += DT) {
            fp_without_s.t.push_back(_t);
            fp_without_s.d.push_back(lateral_fp.calc_point(_t));
            fp_without_s.d_d.push_back(lateral_fp.calc_first_derivative(_t));
            fp_without_s.d_dd.push_back(lateral_fp.calc_second_derivative(_t));
            fp_without_s.d_ddd.push_back(lateral_fp.calc_third_derivative(_t));
        }
```

Longitudinal sampling

1. 纵向对车速的采样，纵坐标为速度，横坐标为时间（此处的时间t值为d-t曲线中的离散时间值，保证s-d一一对应）；
2. 纵向速度确定然纵向位移不定，首末端共**5个固定量**，用**四次多项式**拟合成一条用于描述横向采样性质的曲线




```
// 当前遍历下，每一条s-t曲线复用上面刚刚生成的d-t曲线（只有一根），for循环每运行一次，生成一条完整的备选轨迹
for (float vi = TARGET_SPEED - D_T_S * N_S_SAMPLE; vi < TARGET_SPEED + D_T_S * N_S_SAMPLE; vi += D_T_S) {

    FrenetPath fp_with_s = fp_without_s;
    QuarticPolynomial longitudinal_qp(s0, c_speed, 0.0, vi, 0.0, ti);

    for (float _t : fp_without_s.t) {
        fp_with_s.s.push_back(longitudinal_qp.calc_point(_t));
        fp_with_s.s_d.push_back(longitudinal_qp.calc_first_derivative(_t));
        fp_with_s.s_dd.push_back(longitudinal_qp.calc_second_derivative(_t));
        fp_with_s.s_ddd.push_back(longitudinal_qp.calc_third_derivative(_t));
    }
```

Trajectory cost function

深蓝学院
shenlanxueyuan.com

- 参考论文 Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame 给出的评价函数，算出每条采样轨迹的代价；

```
// The lateral cost of each alternate trajectory.
fp_with_s.cd = KJ * Jp + KT * ti + KD * pow(fp_with_s.d.back(), 2);
// The longitudinal cost of each alternate track.
float ds = pow(TARGET_SPEED - fp_with_s.s_d.back(), 2);
fp_with_s.cv = KJ * Js + KT * ti + KD * ds;
// The weighted sum of the above costs -- the total cost
fp_with_s.cf = KLAT * fp_with_s.cd + KLON * fp_with_s.cv;

fp_list.push_back(fp_with_s);
```

◆横向代价

横向加速度；时间；末尾处横向位移

$$C_d = k_j J_p + k_t t + k_d d^2$$

◆纵向代价

纵向加速度；时间；末尾处速度偏差

$$C_v = k_j J_s\left(s\left(t\right)\right) + k_t t + k_{\dot{s}}\left[\dot{s}_1 - \dot{s}_2\right]^2$$

Path selection

- 碰撞检测
- 速度阈值检测
- 加速度阈值检测
- 曲线行驶曲率检测
- Jerk检测
- ……

```
Vec_Path FrenetOptimalTrajectory::check_paths(Vec_Path path_list, const Vec_Poi ob) {
    Vec_Path output_fp_list;

    for (FrenetPath _fpp : path_list)
    {
        bool collision_free = false;
        collision_free = check_collision(_fpp, ob);
        if(collision_free){
            if(fabs(_fpp.max_speed) < MAX_SPEED){
                if(fabs(_fpp.max_accel) < MAX_ACCEL){
                    if(fabs(_fpp.max_curvature) < MAX_CURVATURE){
                        output_fp_list.push_back(_fpp);
                    }
                }
            }
        }
    }
    return output_fp_list;
};
```

Path selection

- 可行路径中，根据轨迹的代价确定最优路径

```
FrenetPath FrenetOptimalTrajectory::frenet_optimal_planning(Spline2D csp, float s0, float c_speed, float c_d, float c_d_d, float c_d_dd, Vec_Poi ob) {
    Vec_Path fp_list = calc_frenet_paths(c_speed,c_d,c_d_d,c_d_dd,s0);
    calc_global_paths(fp_list, csp);
    Vec_Path save_paths = check_paths(fp_list, ob);
    float min_cost = std::numeric_limits<float>::max();
    FrenetPath final_path;
    for (auto path:save_paths){
        if(min_cost >= path.cf){
            min_cost = path.cf;
            final_path = path;
        }
    }
    return final_path;
};
```

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening