第三章作业思路提示

Stanley

$$\delta(t) = \theta_e(t) + tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right), \delta(t) \in [\delta_{min}, \delta_{max}]$$

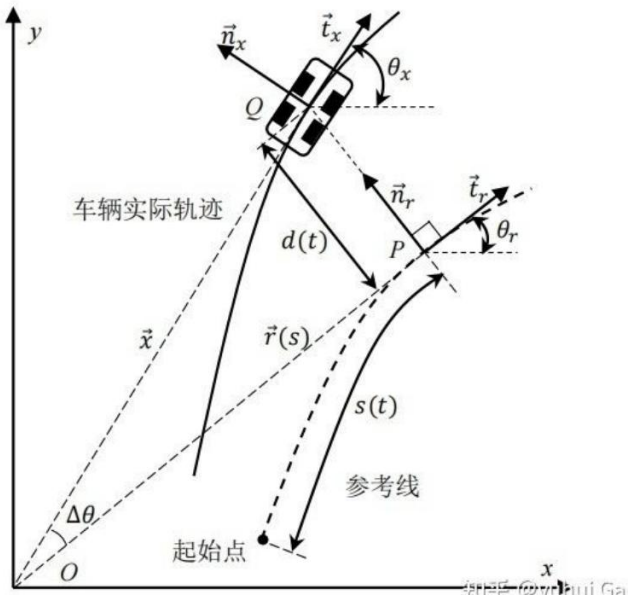
1. **首先是ComputerControlCmd函数，这一部分比较简单，就是根据Stanley算法的公式进行代码编写，需要调用接下来的误差计算函数，然后整个前轮转角控制命令分为两部分，分别是由航向误差和由横向误差引起的转角。需要注意的是：**
   1. 计算反正切函数值时，建议使用atan2函数，其返回值为点和原点连线与x轴正方向的夹角，值域对应为-pi到+pi；
   2. 实际的前轮转角有一个范围，即δ（t）∈[δmin，δmax]，所以需要对其进行限幅处理。
2. **为了提升横向控制器的性能，可以考虑以下改进措施：**
   1. 计算横向位置的时候，在分子上添加一个常数项，在低速条件下改善控制器性能；
   2. PID控制器中的微分环节相当于阻尼，加在航向误差引起的前轮转角上，抑制高速工况下的过大的前轮转角变化率；
   3. 在连续弯道中，引入一个前馈项来提高跟踪性能，前馈项和路径曲率相同就足够

Stanley

深蓝学院
shenlanxueyuan.com

```cpp
// /** to-do **/ 计算需要的控制命令， 实现对应的stanley模型,并将获得的控制命令传递给汽车
// 提示，在该函数中你需要调用计算误差
// 控制器中，前轮转角的命令是弧度单位，发送给carla的横向控制指令，范围是 -1~1
void StanleyController::ComputeControlCmd(const VehicleState &vehicle_state, const TrajectoryData &planning_published_trajectory, ControlCmd &cmd) {
    this->trajectory_points_ = planning_published_trajectory.trajectory_points;    // 参考路径点

    // 主车的状态信息
    double current_vehicle_x = vehicle_state.x;
    double current_vehicle_y = vehicle_state.y;
    double current_vehicle_heading = vehicle_state.heading;
    double current_vehicle_velocity = vehicle_state.velocity;

    double e_y = 0.0;
    double e_theta = 0.0;
    this->ComputeLateralErrors(current_vehicle_x, current_vehicle_y, current_vehicle_heading, e_y, e_theta);

    // e_y = std::atan2(this->k_y_ * e_y, current_vehicle_velocity); // atan2 返回的是弧度单位
    // TODO: atan2 返回的是弧度单位，在分子上添加一个常数项，在低速条件下改善控制器性能
    e_y = std::atan2(this->k_y_ * e_y, current_vehicle_velocity + 6.0);
    // 限制Cross track error 引起的前轮转角的取值区间
    if (e_y > M_PI) {
        e_y = e_y - M_PI * 2;
    }
    if (e_y < -M_PI) {
        e_y = e_y + M_PI * 2;
    }
    // TODO: PID控制器中的微分环节相当于阻尼，加在航向误差引起的前轮转角上，抑制高速工况下的过大的前轮转角变化率
    double e_theta_pd = e_theta_pid_controller.Control(e_theta, 0.01);    // 这个0.01和主程序的循环周期匹配上
    // 限制航向误差引起的前轮转角的取值区间
    if (e_theta_pd > M_PI) {
        e_theta_pd = e_theta_pd - M_PI * 2;
    }
    if (e_theta_pd < -M_PI) {
        e_theta_pd = e_theta_pd + M_PI * 2;
    }
    // TODO: 在连续弯道中，引入一个前馈项来提高跟踪性能，前馈项和路径曲率相同就足够
    TrajectoryPoint current_closest_point = this->QueryNearestPointByPosition(current_vehicle_x, current_vehicle_y);    // 得到距离最近的路径点的信息
    double kappa_factor_angle = 0;

    // std::cout << "Current kappa:" << current_closest_point.kappa << std::endl;

    if (isnan(current_closest_point.kappa)) {
        kappa_factor_angle = 0;
    } else {
        kappa_factor_angle = -current_closest_point.kappa;
    }

    double raw_steering_control = e_y + 0.5 * e_theta_pd + 2.6 * kappa_factor_angle;

    // 限制前轮转角的取值区间
    if (raw_steering_control > M_PI) {
        raw_steering_control = raw_steering_control - M_PI * 2;
    }
    if (raw_steering_control < -M_PI) {
        raw_steering_control = raw_steering_control + M_PI * 2;
    }

    // 限制前轮最大转角，这里定义前轮最大转角位于 [-20度~20度]
    if (raw_steering_control >= atan2_to_PI(20.0)) {
        raw_steering_control = atan2_to_PI(20.0);
    } else if (raw_steering_control <= -atan2_to_PI(20.0)) {
        raw_steering_control = -atan2_to_PI(20.0);
    }

    // Carla 里面的横向控制信号范围 -1~1 之间
    // raw_steering_control = raw_steering_control / atan2_to_PI(20.0) * 0.2;

    cmd.steer_target = raw_steering_control;    // 给出控制信号
}
```

Stanley



1. **误差计算：ComputeLateralErrors函数,也就是分别表示航向误差e_theta和横向误差e_y。通过QueryNearestPointByPosition可以得到距离当前自车位置最近的点。**
    1. 对于航向误差,即车身方向与参考轨迹最近点的切线方向的夹角,使用自车航向角减去参考点航向(e_theta=θx-θr),并转换到-pi到+pi之间即可。
    2. 对于横向误差,需要进行判断,笛卡尔坐标系下自车位置和参考点之间的距离可以表示为e_y=±sqrt((xr-x)^2, (yr-y)^2,),其中,若(yr-y)cosθr- (xr-x)sinθr>0, e_y为负,否则e_y为正。

Stanley

深蓝学院
shenlanxueyuan.com

```cpp
// /** to-do **/ 计算需要的误差，包括横向误差，纵向误差，误差计算函数没有传入主车速度，因此返回的位置误差就是误差，不是根据误差计算得到的前轮转角
void StanleyController::ComputeLateralErrors(const double x, const double y, const double theta, double &e_y, double &e_theta) {
    TrajectoryPoint current_closest_point = this->QueryNearestPointByPosition(x, y);    // 得到距离最近的路径点的信息

    e_y = sqrt((x - current_closest_point.x) * (x - current_closest_point.x) + (y - current_closest_point.y) * (y - current_closest_point.y));

    // 将位置误差转换为前轮转角的时候：需要将路径上距离车辆最近的点从世界坐标系变换到车辆坐标系下，根据路径点在车辆坐标系下的横坐标的正负决定前轮转角的方向
    // double closest_point_x_in_vehicle_coordinate =   (current_closest_point.x - x) * cos(theta) + (current_closest_point.y - y) * sin(theta);
    double closest_point_y_in_vehicle_coordinate = -(current_closest_point.x - x) * sin(theta) + (current_closest_point.y - y) * cos(theta);
    if (closest_point_y_in_vehicle_coordinate > 0)    // 车辆坐标系：X轴沿着车辆纵向，向前为正，Y沿着车辆横向，向左为正（从车头往前看的视角），在车辆坐标系下，距离车辆最近的路径点位于车辆左侧，车辆应该左转以跟踪参考路径，
    {
        e_y = -e_y;
    } else if (closest_point_y_in_vehicle_coordinate < 0) {
        e_y = e_y;
    }

    // std::cout << "*****************" << this->theta_ref_ << ", " << current_closest_point.heading << std::endl;
    e_theta = -(theta_ref_ - theta);    // 路径上距离车辆最近的点的参考航向角，大于车辆的当前航向角的话，车辆应左转以跟踪航向
}
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening