任务四 十字处理

背景介绍

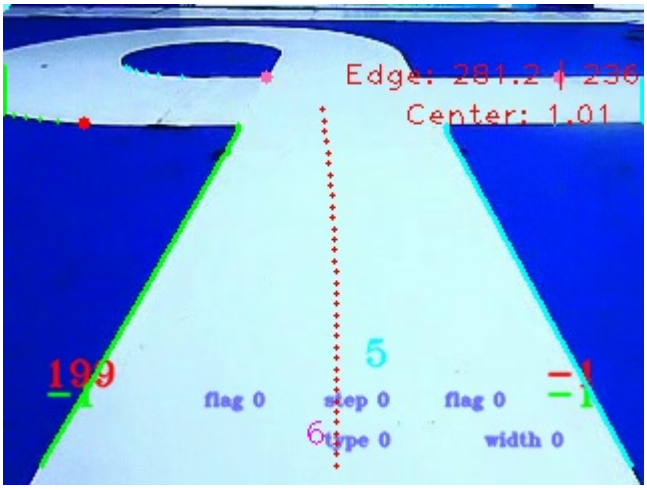
在智能车应用场景中，十字路口是赛道的典型元素。按照行比赛规则，车辆需直行通过十字路口。这一操作看似简单 —— 即使不做特殊处理，车辆理论上也能依据已拟合的中心线直行通过；但实际行驶中，若车身存在姿态偏差（如航向角偏移、位置偏移等），则需通过补线修正等手段调整轨迹，以确保车辆精准、稳定地通过路口。

环境配置

IDE：Pycharm
相关依赖：numpy、opencv

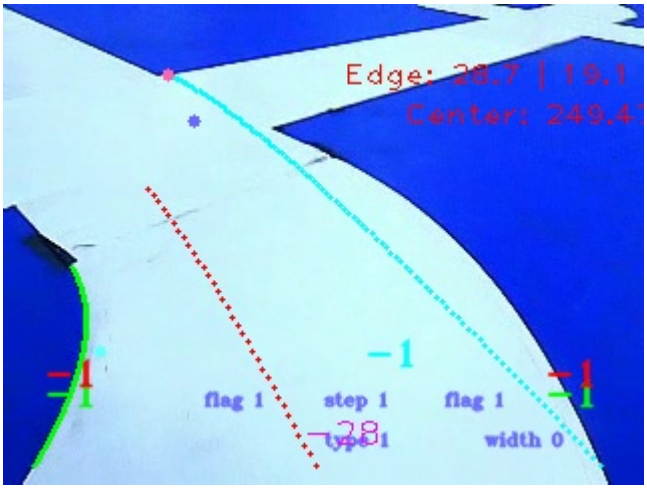
实践环节拆解

1. **入门：直入十字** 难度：⭐⭐⭐
   直入十字是最简单的情况。在这种模式下，需要完成3个算子的设计：
   1. 判入算子：判断车辆进入十字
   2. 补线算子：对左、右点集进行修正
   3. 判出算子：判断车辆驶出十字



2. **进阶：斜入十字** 难度：⭐⭐⭐
   车辆经过弯道进入十字，此时车辆姿态发生偏差。在这种模式下，依旧需要完成3个算子的设计：
   1. 判入算子：判断车辆进入十字
   2. 补线算子：对左、右点集进行修正

3. 判出算子：判断车辆驶出十字



3. **特征工程DIY —— 定制你的模式识别规则** 难度：⭐⭐⭐
   判断赛道模式（直道、弯道、十字），可以通过赛道的形态特征（赛道丢失、凹凸性等），也可以通过统计学特征（赛道点集的方差、均值等）；一个好的特征提取，可以大大简化判断算子的设计，提升模式识别的准确性与鲁棒性。
4. **补线神器：贝塞尔拟合** 难度：⭐⭐
   采用三阶贝塞尔曲线进行拟合时，仅需三个特征点即可拟合线。通过调整这三个点的选取方式，既能拟合出符合直道需求的直线，也能生成适应弯道的曲线，可灵活应对十字区域及过渡路段的补线需求，为轨迹修正提供高效解决方案。

📝任务要求

1. 利用res中的视频，至少完成一种十字模式（直入、斜入）的判入、补线与判出
2. 将修正后的赛道线、中心线可视化

📖参考资料

1. https://blog.csdn.net/qq_53239103/article/details/119959811
2. https://avoid.overfit.cn/post/f2e46bc964e24d1c9d8cb9a190b36413
3. 代码设计参考：

- 使用类
- 使用switch-case结构，实现状态机，便于维护、拓展

```
class Cross:
    class CrossStep(Enum):
        NONE = 0
        Fix = 1
    class CrossMode(Enum):
        NONE = 0
        Left = 1
        Right = 2
        Straight = 3
    def __init__(self):
```

```
        self.step = Cross.CrossStep.NONE
        self.mode = Cross.CrossMode.NONE
    def process(self,track):
        # python 无switch-case，暂时用if-else代替
        if(step == Cross.CrossStep.NONE)
            # 判断算子
            return False
        elif(step == Cross.CrossStep.Fix)
            # 判断算子
            # 补线算子
            return True
```