第十七届全国大学生

智能汽车竞赛

技 术 报 告

学　　校：西南交通大学

队伍名称：头文字 A 队

参赛队员：朱逸凡

刘云杰

王述杰

王翔

带队教师：蒋朝根

朱石磊

关于技术报告和研究论文使用授权的说明

本人完全了解全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：朱逸凡 刘云杰 王述杰 王翔

带队教师签名：蒋朝根 朱石磊

日　　期：2022.8.20

目录

摘 要

本文设计的智能车系统以第十七届全国大学生智能车竞赛专用I汽车模型作为研究平台。以百度Edgeboard FZ3B开发板为计算单元，选取英飞凌AURIX第二代双核单片机TC264为控制单元，上位机是通过vscode的ssh功能使用gcc和Makefile编译并管理c++工程，下位机使用 AURIX Development Studio集成开发平台进行单片机程序开发。通过车体上方的UVC免驱摄像头采集图像提取赛道信息；通过Edgeboard板载的Zynq fpga模块实现AI模型的部署与加速；通过光电编码器检测模型车的实时速度，使用PID控制算法控制电机转速,使用PD环节控制舵机打角，实现了对车运动速度和运动方向的闭环控制。通过mjpg-streamer的网页视频推流以及opencv的绘图功能实现了便捷有效的调试环节。实验结果表明，该系统设计方案稳定可行。

关键字：智能车，英飞凌TC264，UVC免驱摄像头，图像处理算法，AI模型预测，电机PID控制、舵机PD控制。

引 言

全国大学生智能汽车竞赛是以“立足培养、重在参与、鼓励探索、追求卓越”为指导思想，鼓励创新的一项创意性科技竞赛，是面向全国大学生的一种具有探索性工程实践活动。组委会提供汽车标准模型以及电机与舵机，使用指定单片机为核心控制模块，并通过使用道路传感器对黑白赛道或者电磁引导进行循迹。要求智能车能够在设计的跑道上自动识别并行驶，智能车竞速以时间来排名。其智能车设计涵盖了控制、模式识别、传感技术、汽车电子、电气、计算机、机械、能源等多学科知识。竞赛融科学性、趣味性为一体，充分发挥了参赛队伍的想象力，来实现完成赛道任务，制作自己的智能车。

在报告中，我们组通过对智能小车的整体设计方案，电路设计，算法以及调试过程进行介绍，阐述我们对于本次比赛任务的思考和创意。

第一章 系统总体设计

在本章中简要介绍智能车系统的整体设计内容，在后面的章节会分别对于机械结构与控制部分以及图像算法部分对于智能车系统进行分析。

1.1 系统总体方案的选定

根据十七届智能车竞赛的规则，仅能使用 I 车模，车模微控制器使用英飞凌系列的单片机。允许使用各类电磁、红外光电、摄像头激光、超声传感器件进行赛道和环境检测。

因此选择使用了英飞凌高性能双核单片机 TC264，其性能强大，芯片主频最高可以实现 200M 且拥有双核心，合理利用可以运行复杂算法，且效率极高。并且作为车载级芯片，其稳定，不会轻易出现跑飞的现象。并且选用了数字 UVC 免驱摄像头，参数可以通过 opencv 进行调节。编码器选用光电编码器，对于车的速度获取较为精确，使用其来对速度闭环反馈。

1.2 系统总体方案的设计

遵循本届竞赛规则规定，智能汽车系统采用英飞凌 TC264 高性能双核单片机作为核心控制单元用于智能车系统的控制。数字摄像头采集赛道信息，返回到 Edgeboard 作为舵机实时转向和速度控制的依据。编码器模块采集速度信息，用于速度的串级闭环控制。根据以上的系统方案设计，可以将智能车系统分为六个子系统：电源模块，Edgeboard 计算模块，单片机控制模块，摄像头图像采集模块，电机驱动模块，舵机驱动模块，人机交互调试模块。

1. 电源模块：电源模块采用不同的稳压芯片和其外围电路为 Edgeboard、单片机、舵机等硬件提供合适且稳定的电源。
2. Edgeboard 计算模块:主控是整个智能车的处理中心，其完成赛道数据的输入和处理，并且通过 pd 控制算法来实现对智能车舵机的实时控制。
3. 单片机控制模块：通过 pid 控制算法实现对智能车电机的实时控制。通过串口接收 Edgeboard 的指令，完成对舵机打角和电机转速的调整。通过 adc 模块实现对电池电压的检测与报警。
4. 赛道提取传感器模块：本部分即摄像头，可以通过其获取整个赛道的信息，使智能车能在赛道中稳定运行。
5. 电机驱动模块：包含电机驱动电路以及速度采样模块，使智能车能拥有稳定快速的速度输出。
6. 舵机驱动模块：包含舵机驱动电路，使用 CS-3120 舵机实现对智能车的方向控制。
7. 人机交互模块：通过 mjpg-streamer 的网页视频推流以及 opencv 的绘图功能实现了便捷有效的调试环节。

第二章　机械结构调整及优化

2.1 前轮

前轮作为舵机转向轮，其机械结构决定了智能车的转向上限。通过主销内倾的方式改变轮胎与地面的接触面积，提升转弯速度。

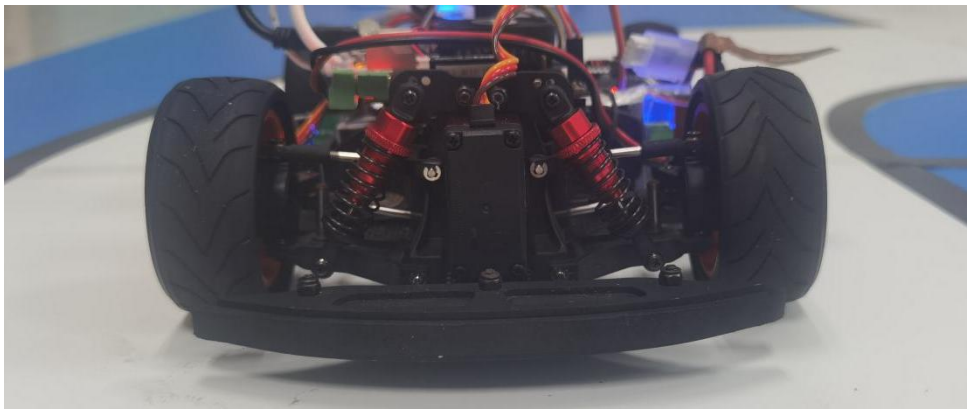


图 2.1 编码器安装

主销后倾和主销内倾都有使转向轮自动回正的作用。但主销后倾的回正作用与车速有关，而主销内倾的回正作用与车速无关。因此，高速时主要靠主销后倾的作用，而低速时主要靠主销内倾的作用。

2.2 编码器（速度传感器）的安装

智能车使用的编码器（encoder）是将转动路程信号转换为脉冲信号形式的设备。编码器把角位移转换成电信号。我们使用的是光电编码器，精度较高且不易烧坏。在安装过程中应注意码盘松紧，既不能让码盘松动，也不能让码盘卡住。

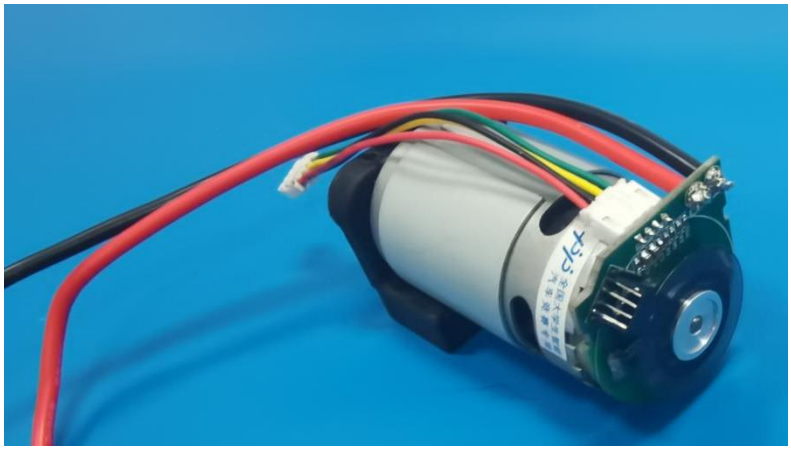


图 2.2 编码器安装

第三章 赛道标志识别

3.1 数据采集

使用动态和静态的两种不同的方法，对赛道标志数据的进行采集。采集的数据导入到“百度大脑 EasyData 智能数据服务平台——提供数据采集、标注、清洗、加工等一站式数据服务，助力开发者高效获取 AI 开发所需高质量数据”进行处理。原始数据分布如下图所示。

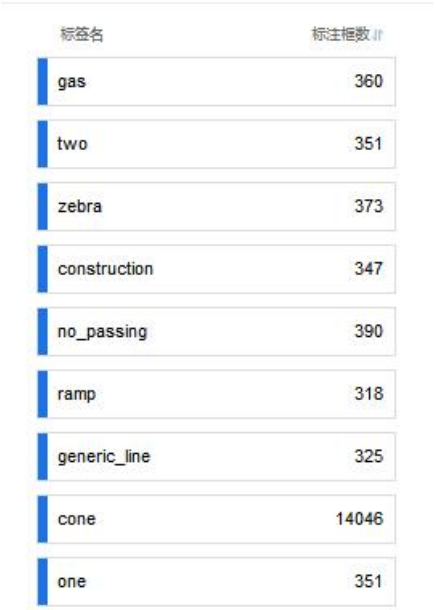


图 3.1 原始数据分布

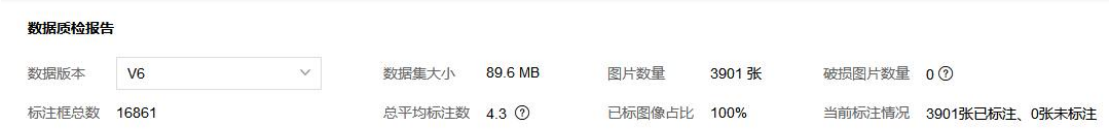


图 3.2 原始数据质检报告

3.2 数据处理

3.2.1 数据均衡

样本的不平衡会降低模型的训练效率与检测精度，所以需要调整每一类别的 bbox 数量相近。

3.2.2 数据标注

EasyData 提供了智能标注功能，为我们大大节省了数据标注的时间，只需要人工将每类标签打上 10 个以上的标注框后，即可开启智能标注。

3.2.3 数据增强

当在实践中无法收集到数目庞大的高质量数据时，可以通过数据增强策略，对数据本身进行一定程度的扰动和扩充，从而产生"新"数据。在训练时会通过学习大量的"新"数据，提高模型的泛化能力量。EasyData 的提供了多种数据增强算子。

如下图所示，我们采用了 EasyData 中提供多种算子，包括调整对比度、清晰度、反色、颜色平衡、减少每个颜色通道的 bits 至指定位数、旋转等，帮助模型更好地根据轮廓识别图像。

增强类型	增强方式	算子处理策略	开始时间	增强前数据集	增强后数据集	增强状态
物体检测	AutoContrast、Invert、Equalize、Solarize、Posterize、Color、Brightness、Sharpness、ShearX_BBox、ShearY_BBox、Solarize_add、Rotate_BBox	并行遍历	2022-08-16 00:28:56	tarindata-V1	输出数据-V6	● 已完成

图 3.3

另外，对于准确度要求比较高的 one，two，cone 三类标签，采用了过采样（OverSample）的增强策略，即将这三类别的图片在数据集中复制粘贴一次。

最后，共生成了 41716 张包含标注的图片。

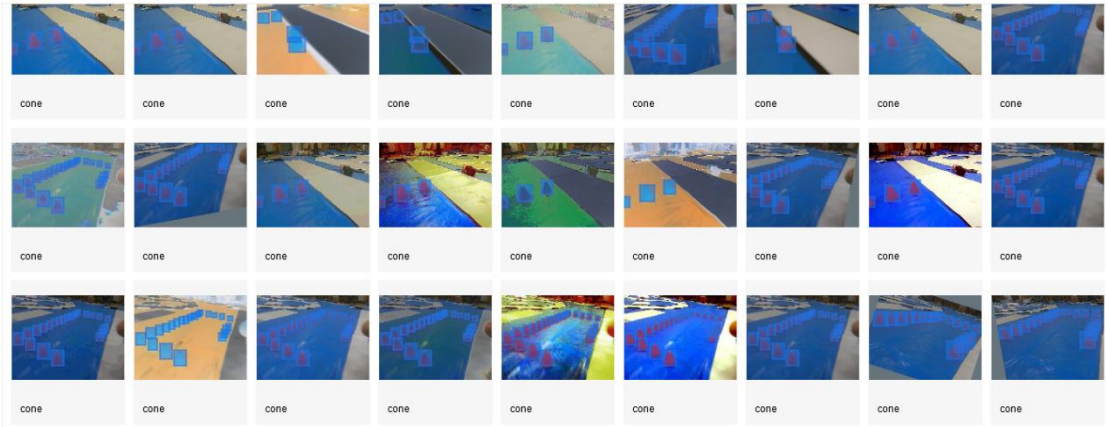


图 3.4

3.3 模型选择与训练

3.3.1 模型选择

虽然 FZ3B 支持的检测模型很多，但我们为了追求较高的速度，选择了轻量级网络 mobilenet_v1，并采用 320*320 的尺寸进行预测。

EdgeBoard FZ3 性能

模型名称	尺寸	FZ3A单帧耗时（ms）	FZ3B单帧耗时（ms）
mobilenet_v1	224x224	9.047096	8.618978
mobilenet_v2	224x224	10.554199	9.784529
InceptionV3	299x299	47.743299	46.719452
InceptionV4	299x299	54.055692	53.086103
ResNet18_vd	224x224	15.369884	14.931973
SE_ResNet18_vd	224x224	17.681101	16.98595
ResNet50	224x224	33.480389	31.120737
SE-ResNeXt50	224x224	52.403168	48.8562

图 3.5 EdgeBoard FZ3 性能（部分）

3.3.2 模型训练以及部署

在 EasyData 上将数据以 VOC 的格式导出后，完成数据集的划分（train：eval=8：2），导入到 Ai Studio 上，使用百度飞桨的目标检测开发套件 PaddleDetection(V0.5)中的 ssd_mobilenet_v1_voc 的配置文件完成训练，完成后按 320*320 尺寸导出后即可在 Edgeboard 上直接部署。

第四章 硬件电路方案设计

4.1 主控核心板

智能车主控选择的是英飞凌 TC264，为便于采购更换，选择逐飞科技核心板采用排针型式连接到车主板上。

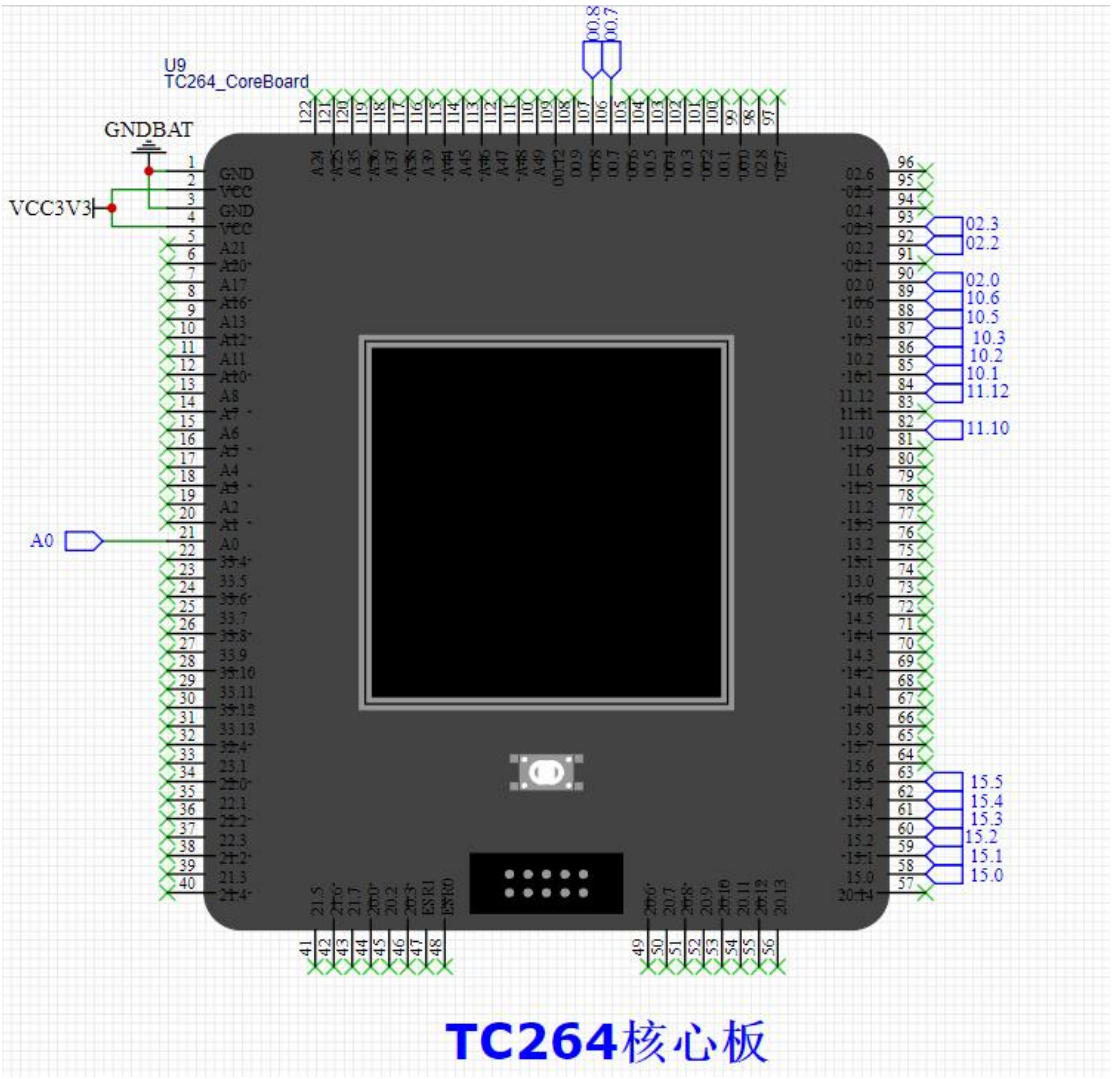
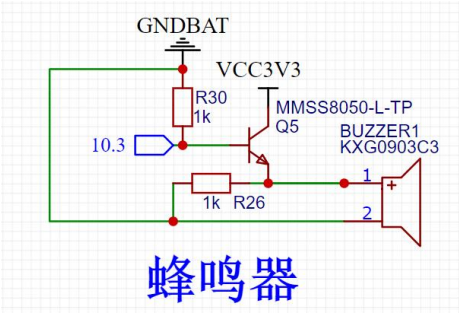


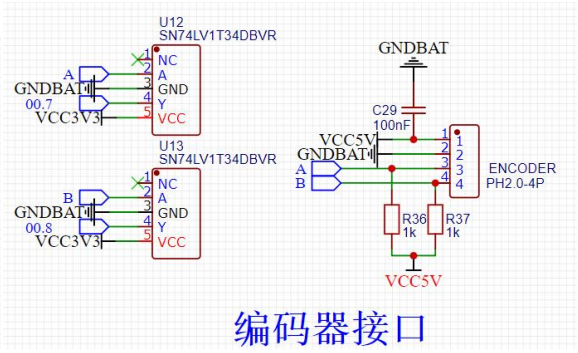
图 4.1 主控核心板

4.2 各类外设接口

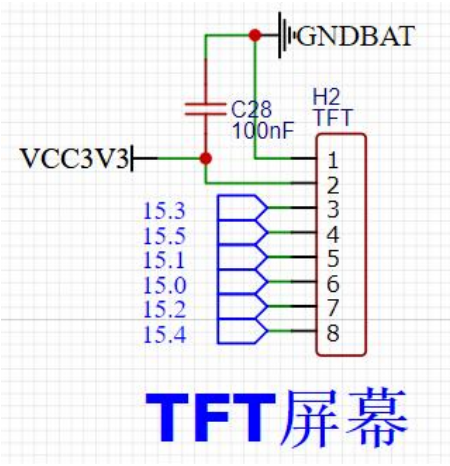
主板上安装了许多车辆平时运行、调试过程中可能会用到的基本外设。如 LCD 屏幕接口、蜂鸣器、陀螺仪、调试串口、电机接口、编码器接口、usb 转串口模块等元件。其中部分为排针排母连接，其余为 1.25mm 电子线接口。



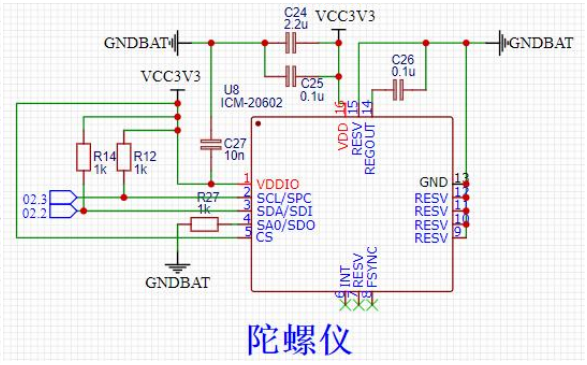
蜂鸣器



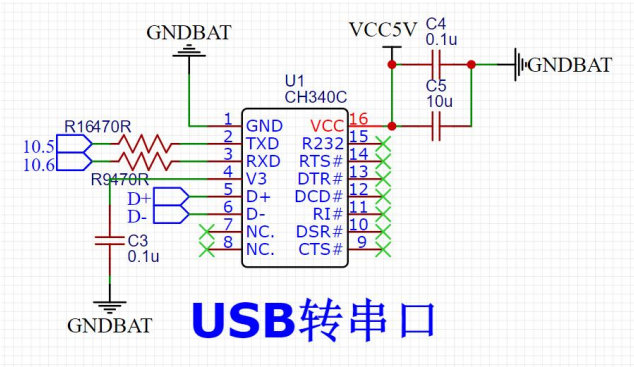
编码器接口



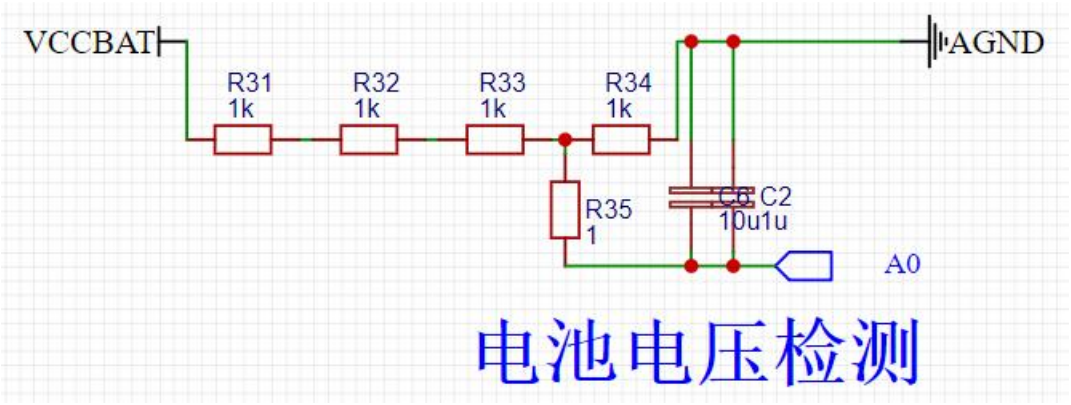
TFT屏幕



陀螺仪



USB转串口



电池电压检测

图 4.2 外设接口

4.3 电源电路

本系统主要设计了两套电源供电电路，分别用于给舵机供电（6.6V），给单片机供电（3.3V）。电源总开关 KERNEL 为总电源开关，开关 MOTOR 为电机电源开关。两路 BUCK 电路均从 VCCBAT 取电，同时电机驱动和 Edgeboard 供电也从 VCCBAT 取电。

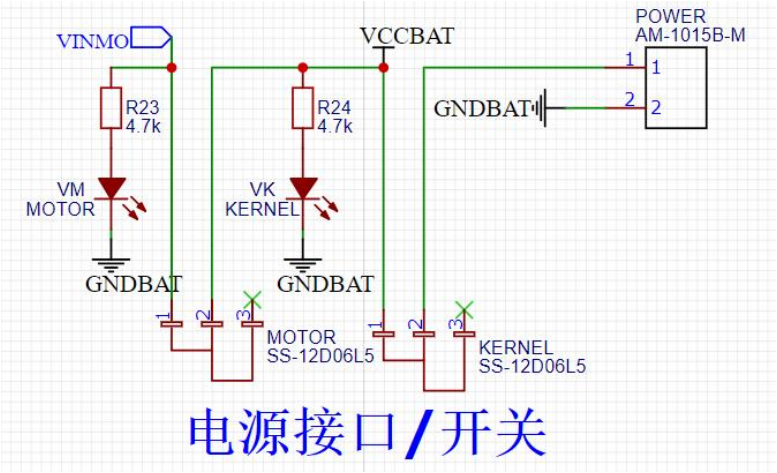
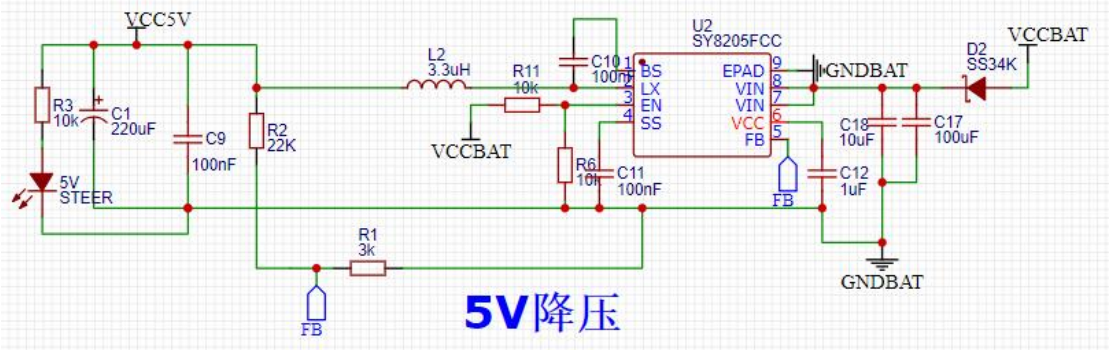


图 4.3 电源电路

电源树如下图 4.4 所示：

图 4.4 电源树

其中主控核心板供电采用的电源 IC 是 SY8205。降压至 5V 后再通过 RT9013 降压至 3.3V



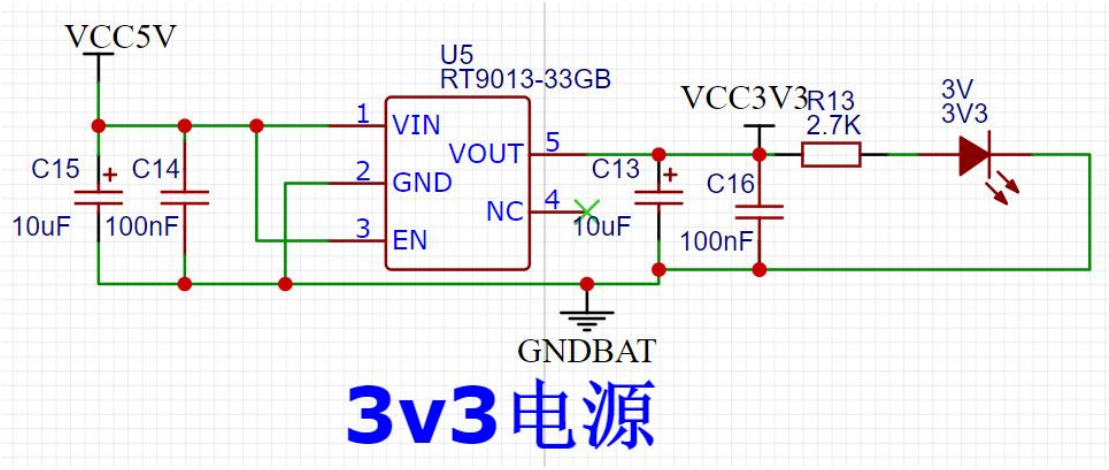


图 4.5 单片机供电

舵机供电同样使用 SY8205 降压至 6.6V

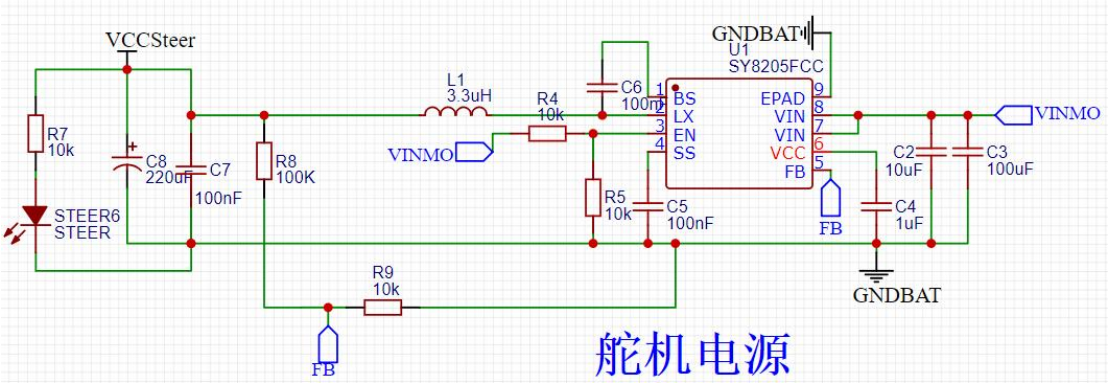


图 4.6 舵机供电

4.4 电机驱动电路

电机驱动采用英飞凌 IR2104 半桥驱动芯片以及英飞凌 IRFH7440 贴片 MOS，该 MOS 最大工作电压为 40V，最大导通电流为 85A，导通电阻 Rdson 约为 2.4mΩ，功率损耗小，散热效率高，适合较大功率的电机驱动场景。

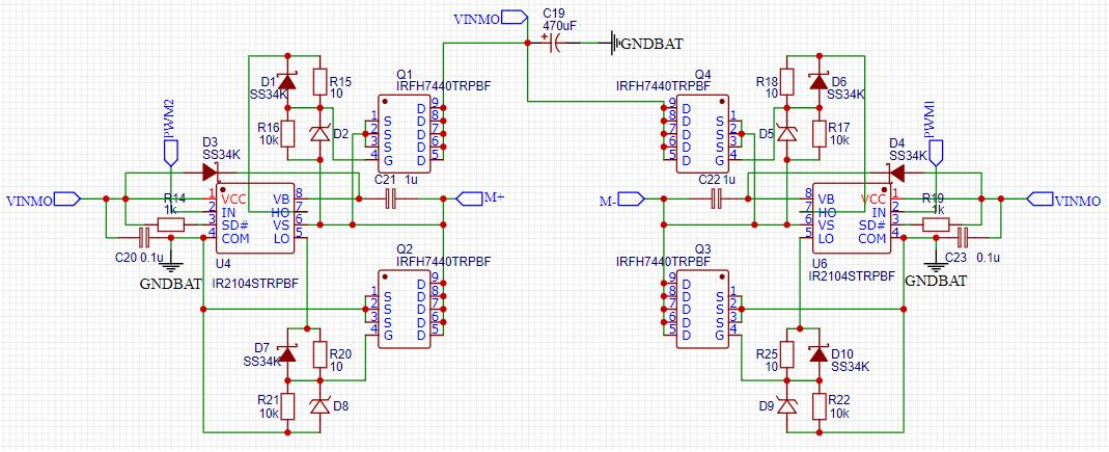
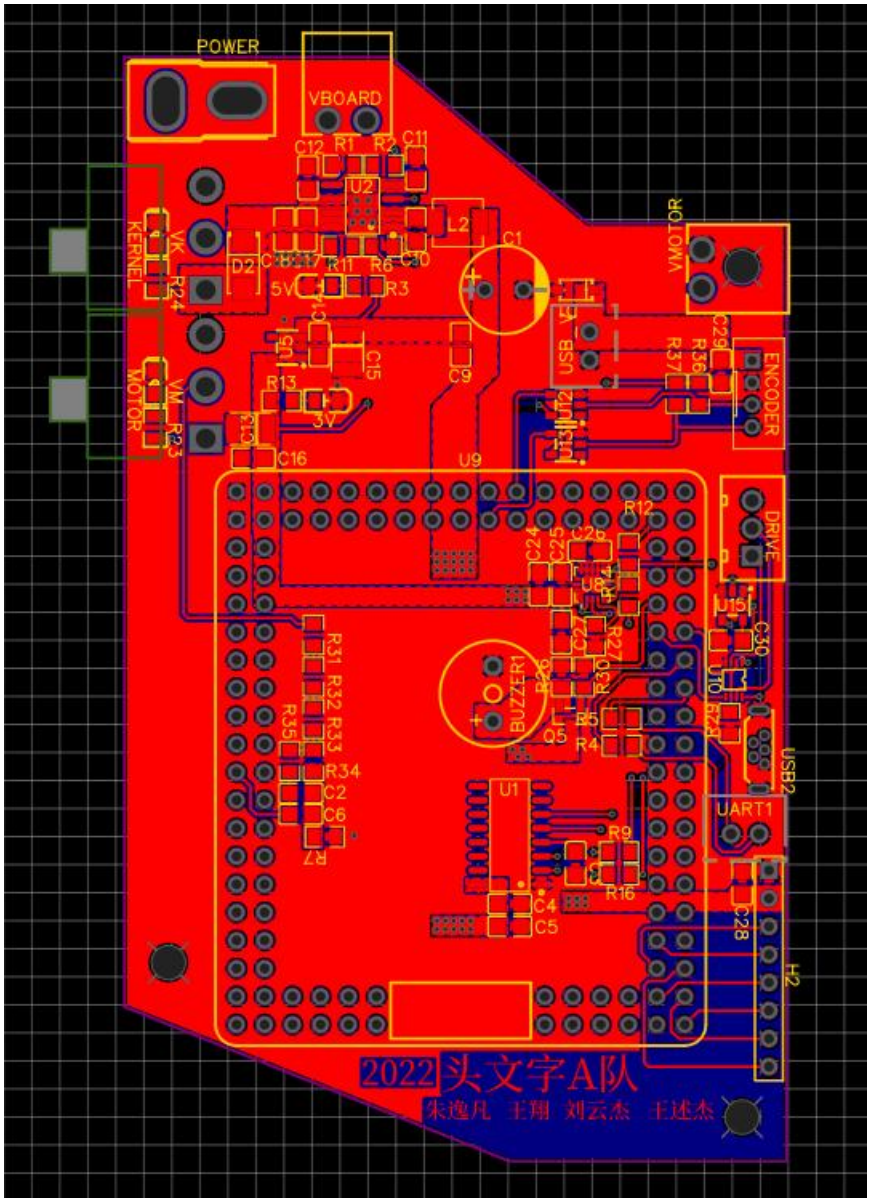


图 4.7 电机供电

主控、驱动 PCB 如图：


POWER
VBOARD
C12
R1
R2
C11
U2
L2
C1
D2
C14
C17
R11
R6
C30
5V
R3
VK
KERNEL
R24
VM
MOTOR
R23
U5
C15
C9
R13
3V
C13
C16
U9
VMOTOR
USB
R36
R37
C29
ENCODER
DRIVE
R12
C24
C25
C26
U8
R14
U15
C19
C27
R27
R30
U10
R28
BUZZER
R29
Q5
R5
R4
USB2
UART1
U1
R9
R16
C4
C5
C28
H2
R31
R32
R33
R34
R35
C2
C6
R7
2022头文字A队
朱逸凡 王翔 刘云杰 王述杰

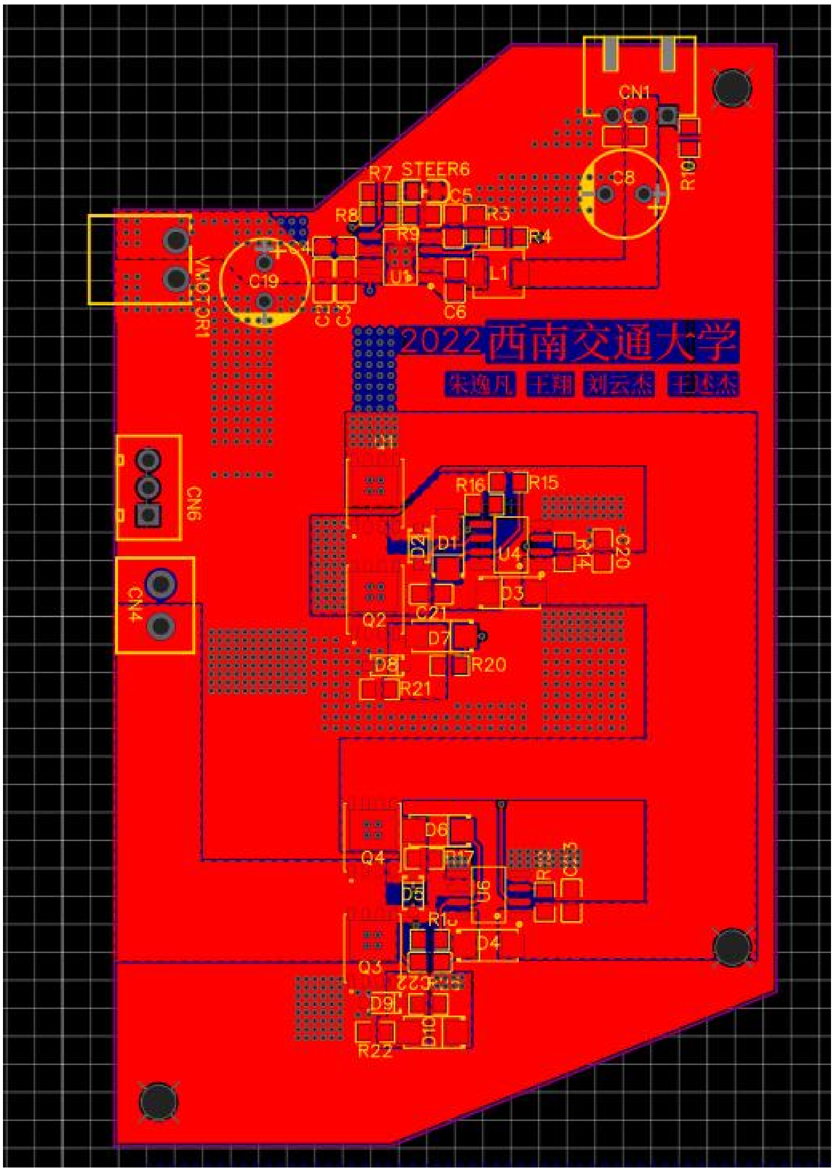

CN1
C8
STEER6
R7
R8
C5
R3
R4
R9
L1
C6
C19
2022西南交通大学
朱逸凡
王翔
刘云杰
CN6
CN4
R16
R15
D1
U4
D3
Q2
C21
D7
D8
R20
R21
D6
Q4
D5
D4
Q3
D9
R22

图 4.8 PCB

第五章 软件算法系统设计

软件系统要实现的功能是从采集到的赛道图像进行处理，并且根据赛道为舵机和电机施行不同的控制。并且为了算法高效稳定的执行，对于智能车的任务施行模块化分配，使车模快速稳定的完成比赛。

5.1 程序系统整体设计

软件编写的目的，是控制车模在既定规则前提下，目标是将硬件电路和机械性能发挥到最大，让车模用最快的速度完成比赛。使用 gcc 和 Makefile 编译并管理工程。程序使用 C++ pthread 多线程编程，主要分为三个线程：传统图像处理线程、AI 图像处理线程、串口发送线程。

5.1.1 程序整体运行流程图

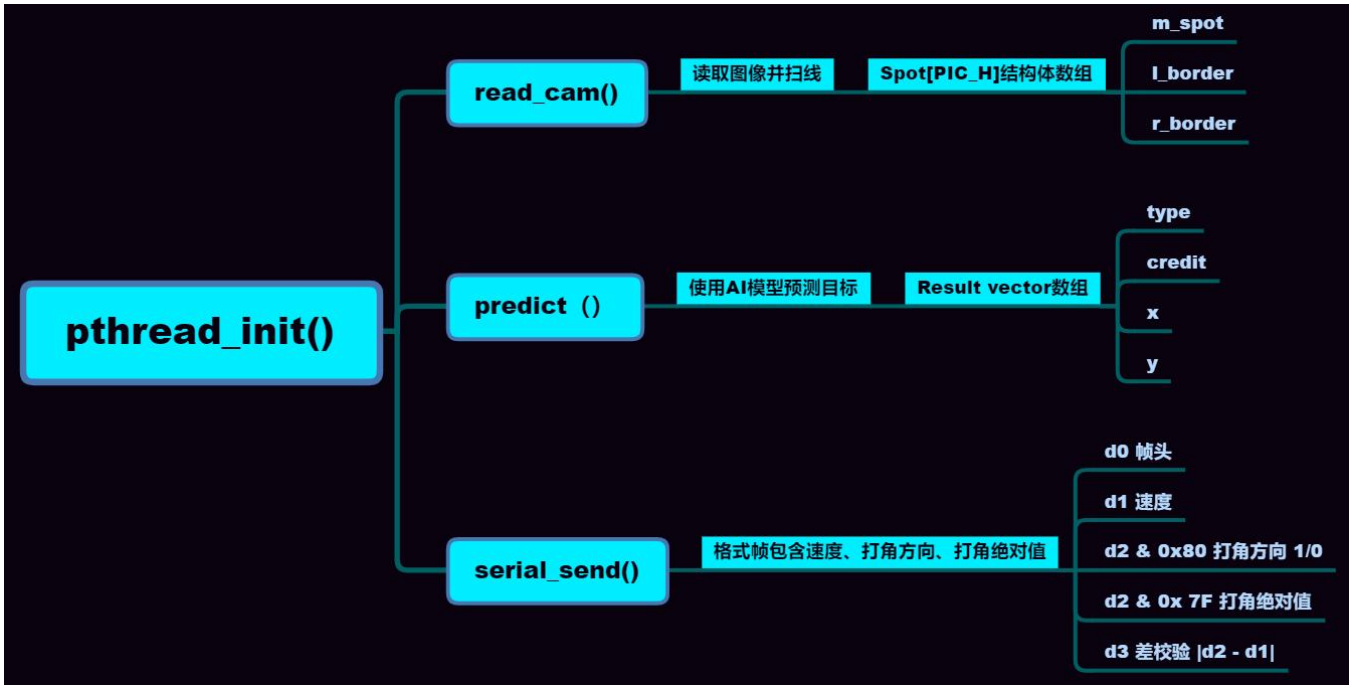


图 5.1 程序整体运行流程图

5.1.2 核心模块任务分配

单片机使用裸机，主函数完成电池电压检测，定时器中断完成编码器数据采集、电机 PID 控制,串口中断完成数据帧解析、舵机控制。

Edgeboard 完成摄像头图像采集、图像分析处理、舵机 pd 调节。调试时完成图像处理信息绘制、图像网页推流，实现可视化调参。

5.2 传统图像算法

5.2.1 灰度图像处理思路

将 UVC 免驱摄像头采集分辨率设置为 320*240，

1） 灰度图像处理

使用 opencv split()函数将传入的 RGB 图像拆分为 B、G、R 三个通道，传统巡线时使用红色通道以避免赛道元素对扫线的干扰。

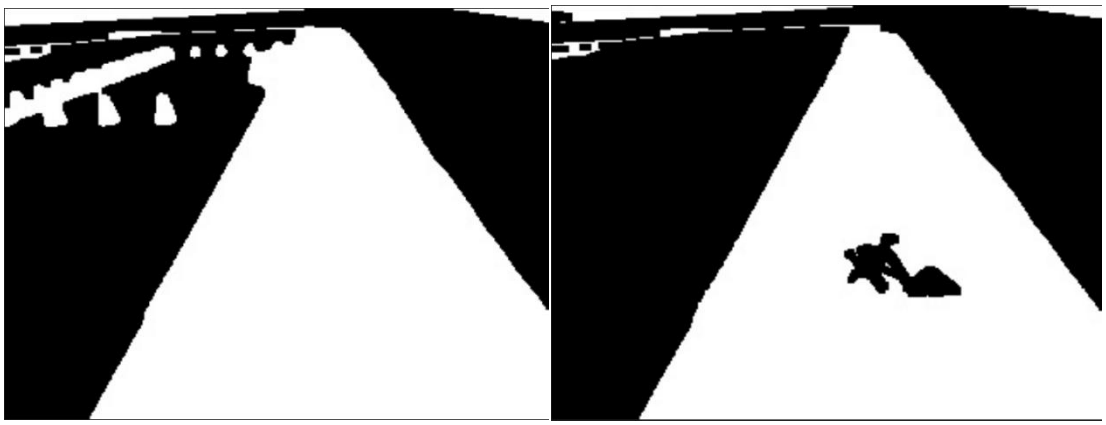


图 5.2 图像灰度处理（左为红色通道 右为原图转灰度图）

2） 阈值二值化处理

阈值化图像就是对灰度图像进行二值化操作，根本原理是利用设定的阈值判断图像像素为 0 还是 255，所以在图像二值化中阈值的设置很重要。图像的二值化分为全局二值化和局部二值化，其区别在于阈值是否在一张图像进行统一。

使用大津法 threshold(red, binary, 0, 255, THRESH_OTSU);对阈值进行动态调整避免环境光照不同对图像二值化的影响。

5.2.2 基本循迹扫线思路

1）扫线

扫线先扫描底边确定底边白色区域中点、边线位置，再由底边中点向上逐级生长中线以及左右边线。限制高度以及左右距离来保证中线的有效信息占比较为合理。通过判断上下两行左右边界是否错开来避免中线超出当前赛道。

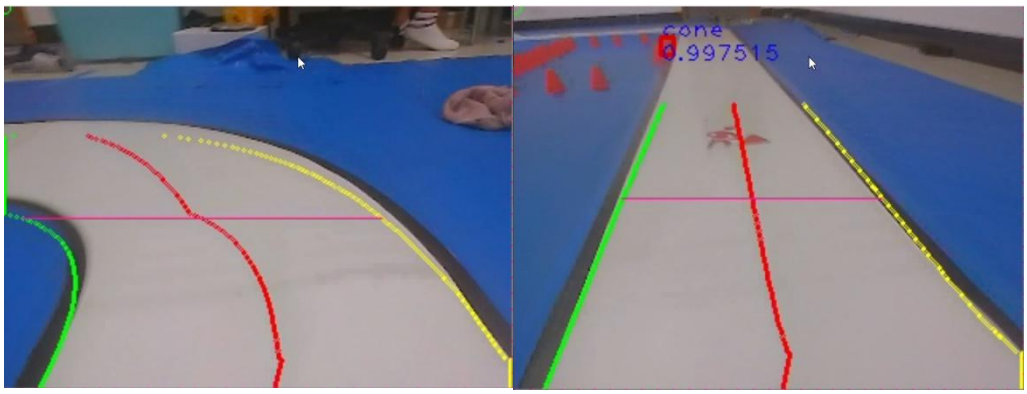


图 5.3 图像灰度处理（左为红色通道 右为原图转灰度图）

5.2.3 特殊元素处理思路

对于元素是否使用状态机进行判断主要依据其复杂度与图像特点。非状态机元素编写简单，但需要使用较短的代码适应整体元素，补线逻辑的适应性要求较高，适合较为简单的补线(如：取左右边线中值、单一边线的平移、单一边线拟合后的延长等)。状态机元素编写较为繁琐，但可以分段把元素拆分，单独编写判断与补线逻辑，较为精确。这里的处理方法仅为我的测试结果，并非绝对最优。

1）非状态机元素

直道，小弯，大弯，斜十字，正十字

十字：如图 5.4，在车的前瞻合适的情况下，十字的特征就是存在两边同时丢线在丢线的一定范围内，去找每一侧的拐点。根据所找到拐点的个数和位置，使用最小二乘来拟合补线。

对于左侧：

若同时找到两个拐点，则直接用这两个点拟合补线;

若只找到左上侧，则用左上拐点以及其上一定行数的点来拟合补线;

若只找到左下侧，则用左下侧以及其下一定行数的点来拟合补线。

右侧同理。

图 5.4 十字

2）状态机元素

圆环

圆环：圆环采用状态机实现，以左环岛为例，状态转换图如下图 5.5 所示：

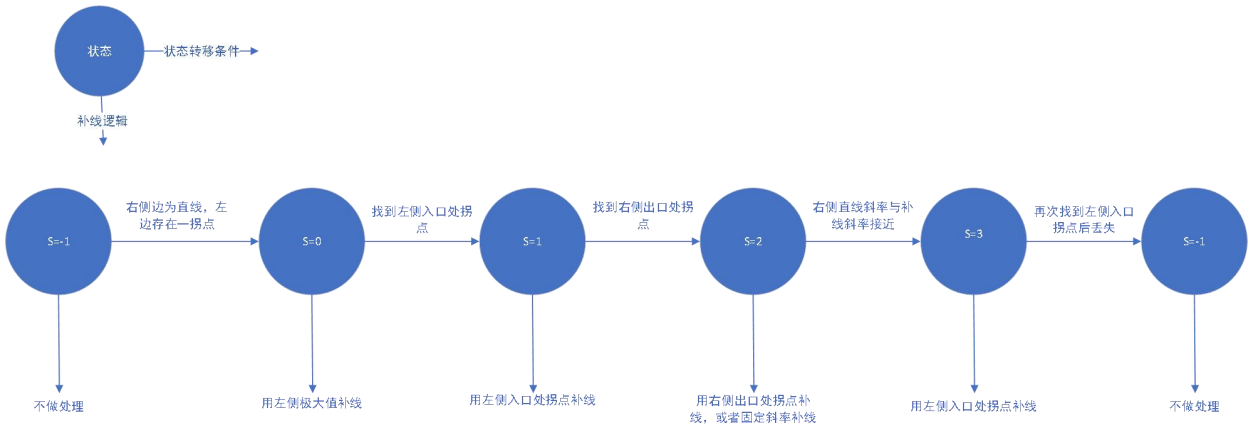


图 5.5 圆环状态转换图

补线效果如下图所示，依次为状态 0、1、2、3 的补线。

图 5.6.1

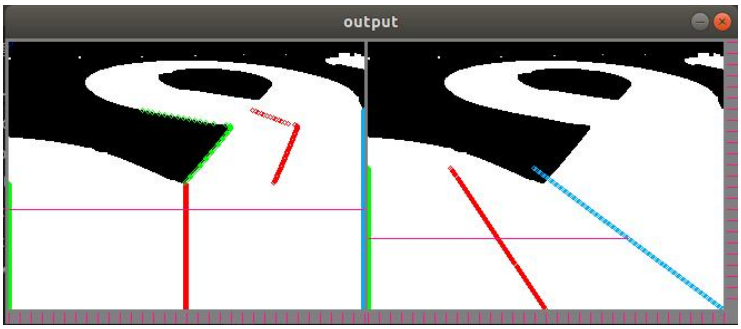


图 5.6.2

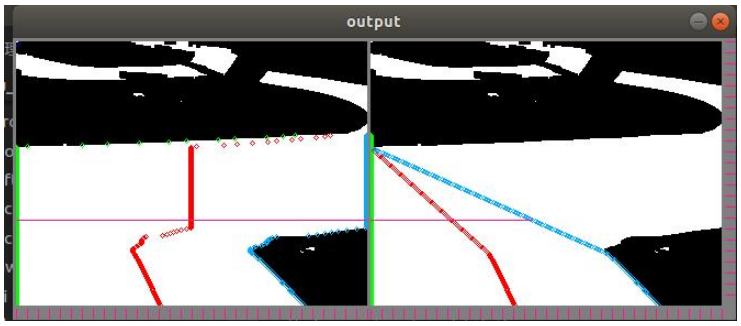


图 5.6.3

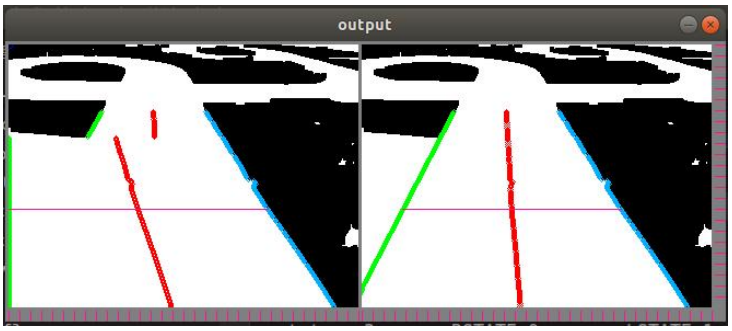


图 5.6.4

右侧环岛同理。

5.3 AI 图像算法

5.3.1 施工区处理

施工区分为 5 个状态来实现

-1 状态:AI 识别施工区标志，连续 3 帧正式进入施工区状态

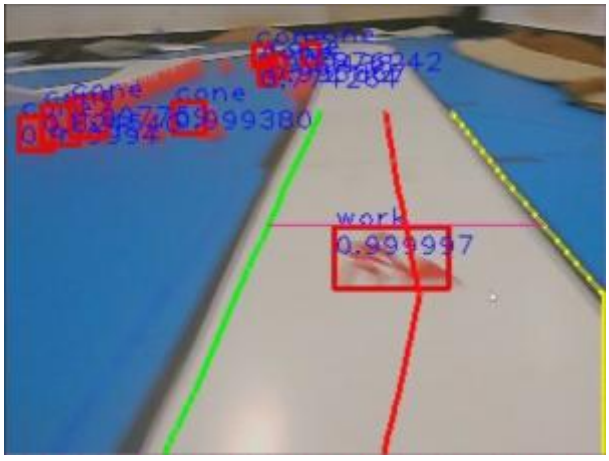


图 5.7.1

0 状态:准备进入锥桶区域状态，向传统补线左边线靠近

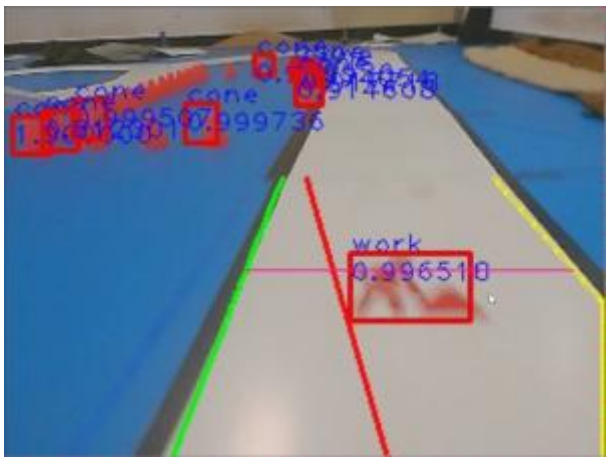


图 5.7.2

1 状态:正在进入锥桶区域状态，找到入口处的两个关键锥桶分别与左右下角的点连线形成左右边线。

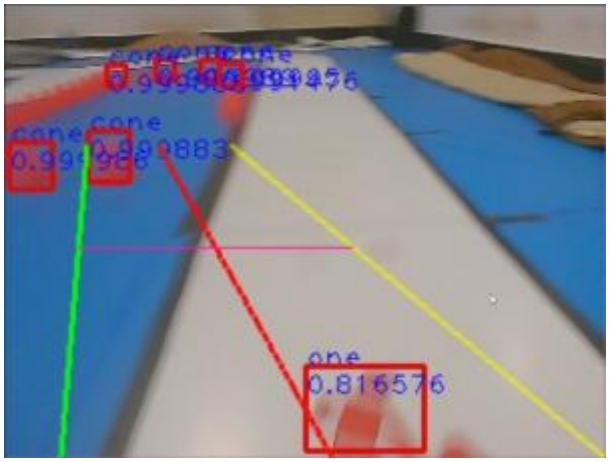


图 5.7.3

2 状态:深入锥桶区域状态，调整角度，以入口靠上的锥桶为关键锥桶，与右下角连线形成右边线，左边线为 x = 20 的一条线

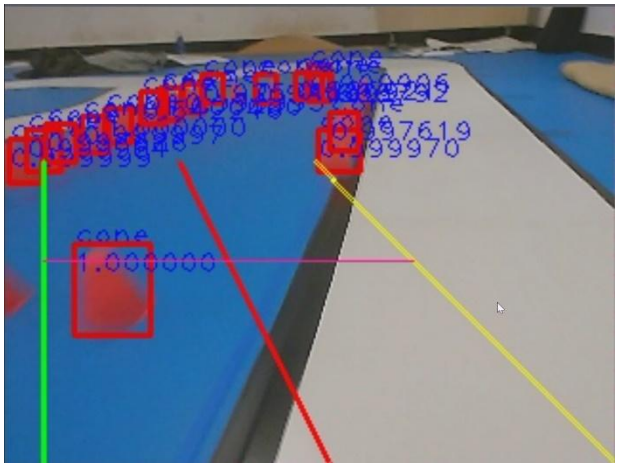


图 5.7.4

3 状态:出锥桶区域状态。以最上最右的锥桶为关键锥桶，与左下角连线形成左边线，以图像右边界作为右边线。

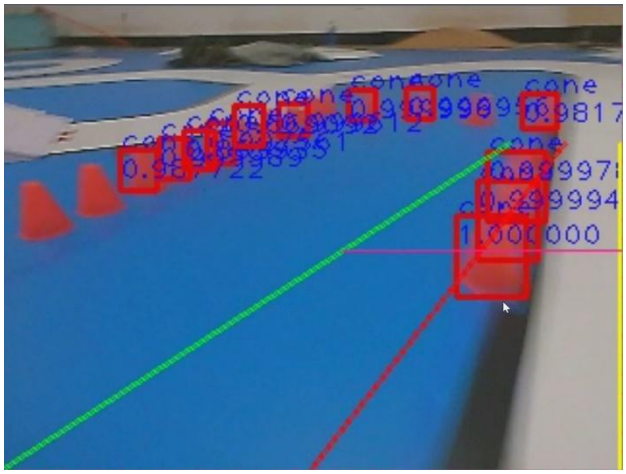


图 5.7.5

5.3.2 加油站处理

加油站分为 7 个状态实现

-1 **状态:**AI 识别加油站标志，连续 3 帧正式进入加油站状态。

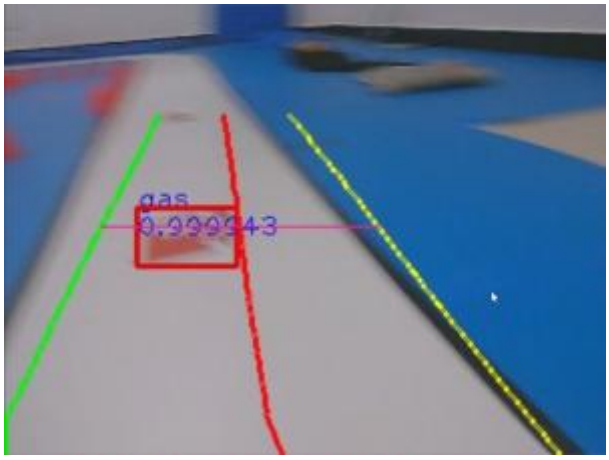


图 5.7.6

0 **状态:**进入锥桶区域准备状态。向传统补线左边线靠近。

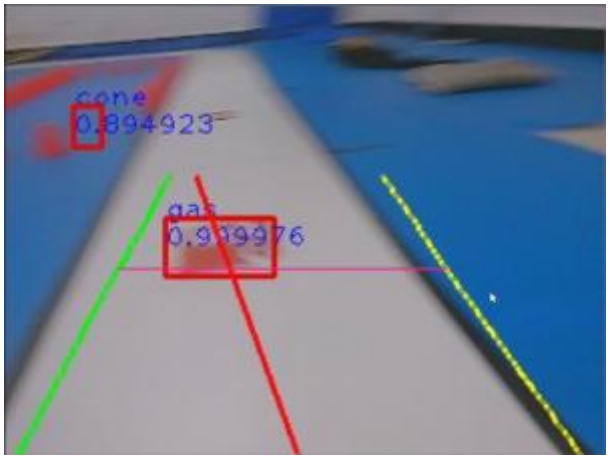


图 5.7.7

1 状态:正在进入锥桶区域状态。找到入口处的两个关键锥桶分别与左右下角的点连线形成左右边线。

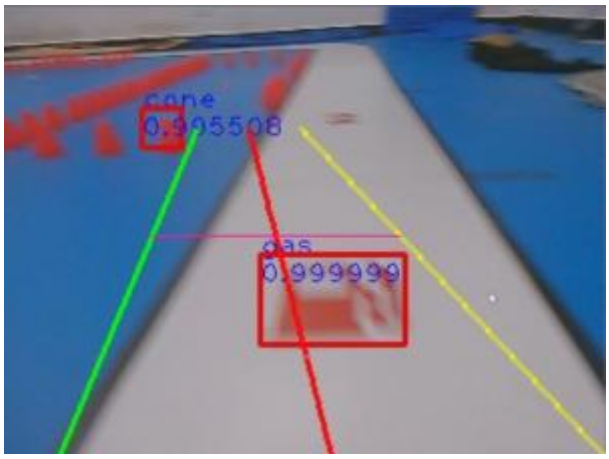


图 5.7.8

2 状态:深入锥桶区域状态，调整角度，以入口靠上的锥桶为关键锥桶，与右下角连线形成右边线，左边线为 x = 20 的一条线。

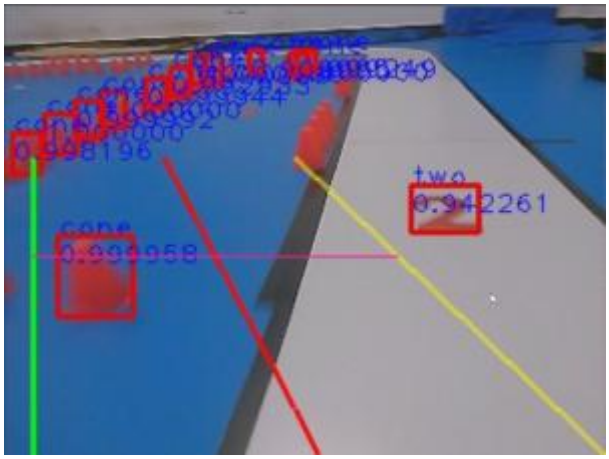


图 5.7.9

在 0，1，2 状态时，统计 AI 识别到的出口标志 1 和 2 的帧数，选择帧数更大者作为出口。若相同，默认选择 1 作为出口。

3_1 状态:从出口 1 驶出锥桶区域状态。找到最上最右的锥桶后，再先到此锥桶下一定范围内最右边的锥桶作为关键锥桶，与左下角连线形成左边线，以画面最右作为右边线。

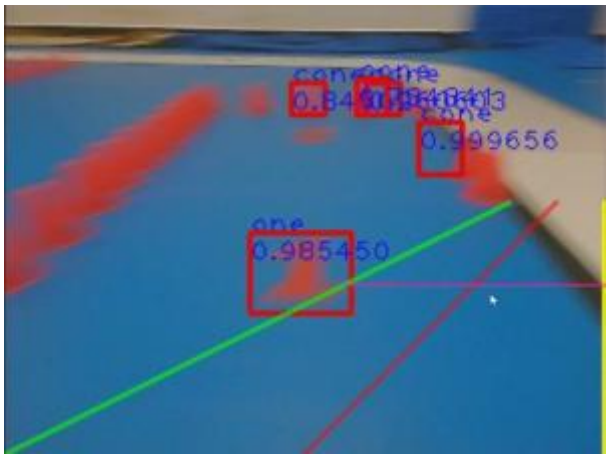


图 5.7.10

3_2 状态:继续向前深入锥桶区域状态。找到最上最左的锥桶作为关键锥桶，与左下角连线形成左边线，以画面最右作为右边线。

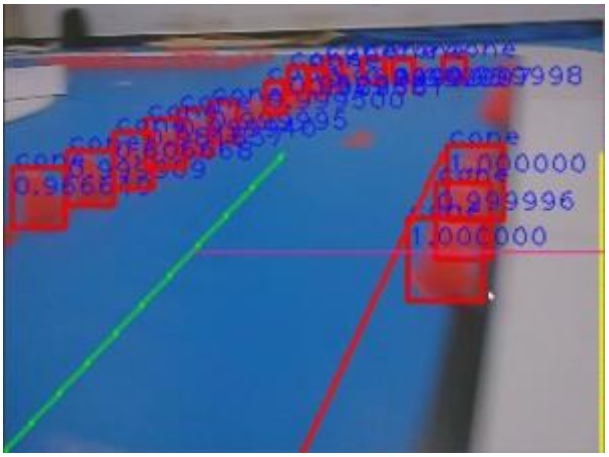


图 5.7.11

4_2 状态:从出口 2 驶出锥桶区域状态。以最上最右的锥桶为关键锥桶，与左下角连线形成左边线，以图像右边界作为右边线。

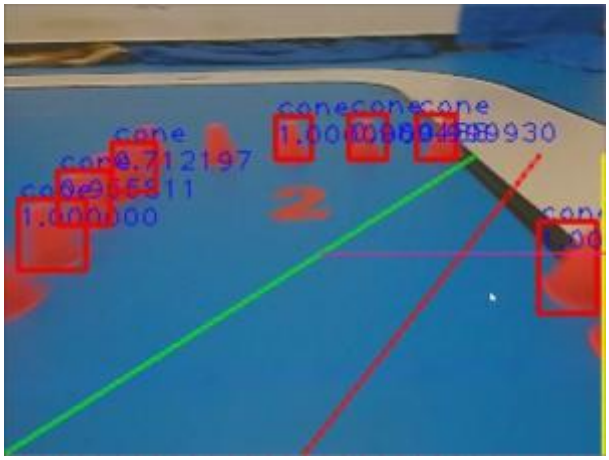


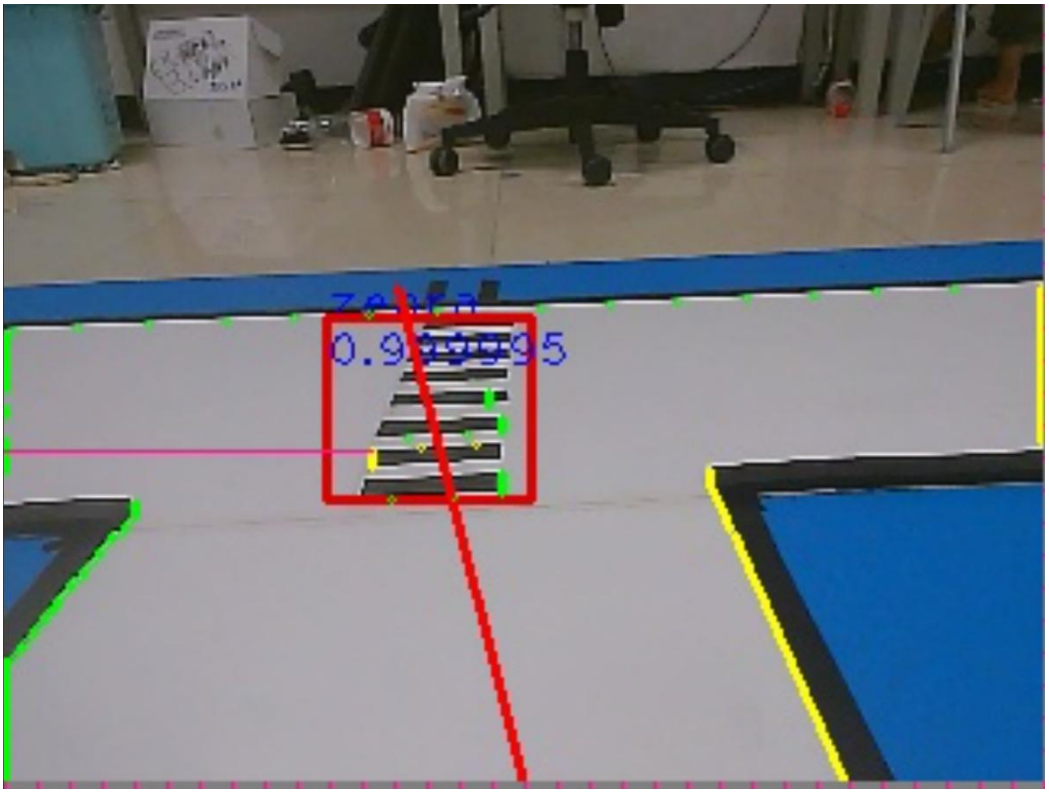
图 5.7.12

由于 AI 模型为小体量模型，需要保证实时性，对于小目标的检测具有一定的局限性，所以，寻找关键锥桶时采用了以 AI 确定大致搜索范围，opencv

进行准确定位的策略。

5.3.3 斑马线处理

斑马线分为 4 个状态来实现
AI 识别斑马线标志，连续 3 帧正式进入斑马线状态，并在斑马线内部再设置一个状态机，并赋予四个不同状态-1、0、1、2，四个状态含义及巡线策略如下：
-1 状态：未出库；AI 识别到的斑马线位置较远，即还未出库时，中线指向识别框的中心；
AI 识别到的斑马线位置较近，即即将出库时，需要根据斑马线左侧的巡线结果向左拐，



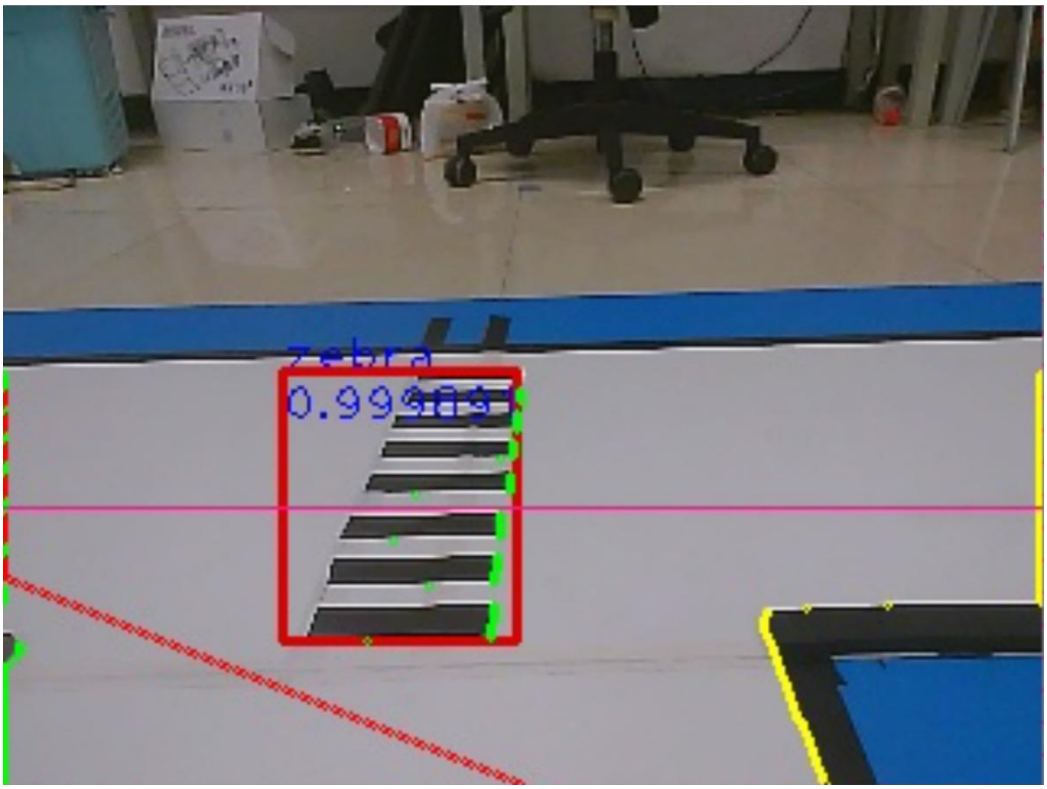


图 5.8.1

0 状态：已出库，未跑完第一圈；根据 AI 识别到的斑马线位置的远近来确定补线高度

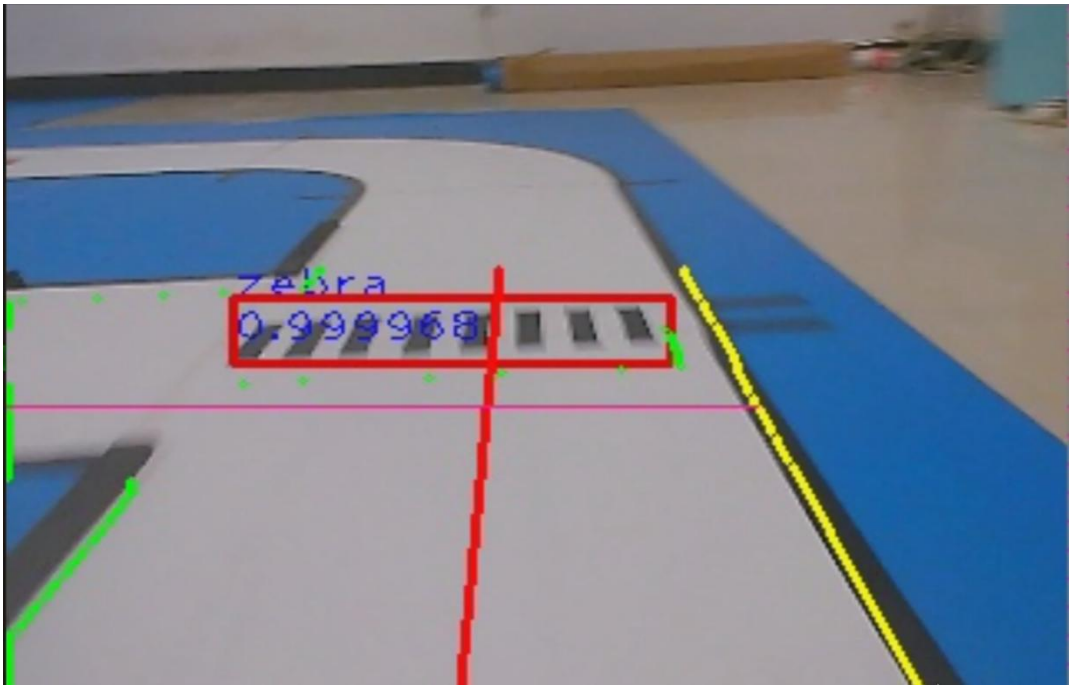
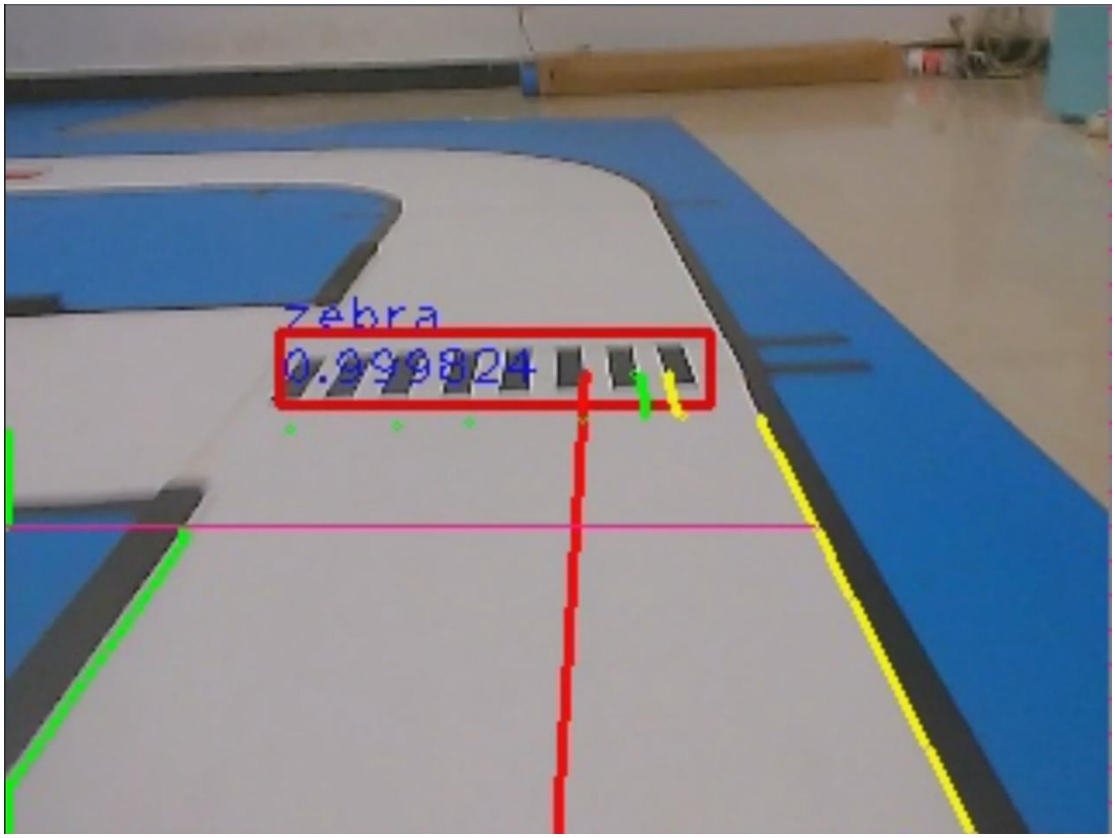


图 5.8.2

1 状态：已跑完第一圈，未跑完第二圈；根据 AI 识别到的斑马线位置的远近和车库的位置来确定补线，较远时，往右靠，避免压到路肩；较近时，根

据斑马线左侧的丢边判断车库位置，从而补出入库线路。



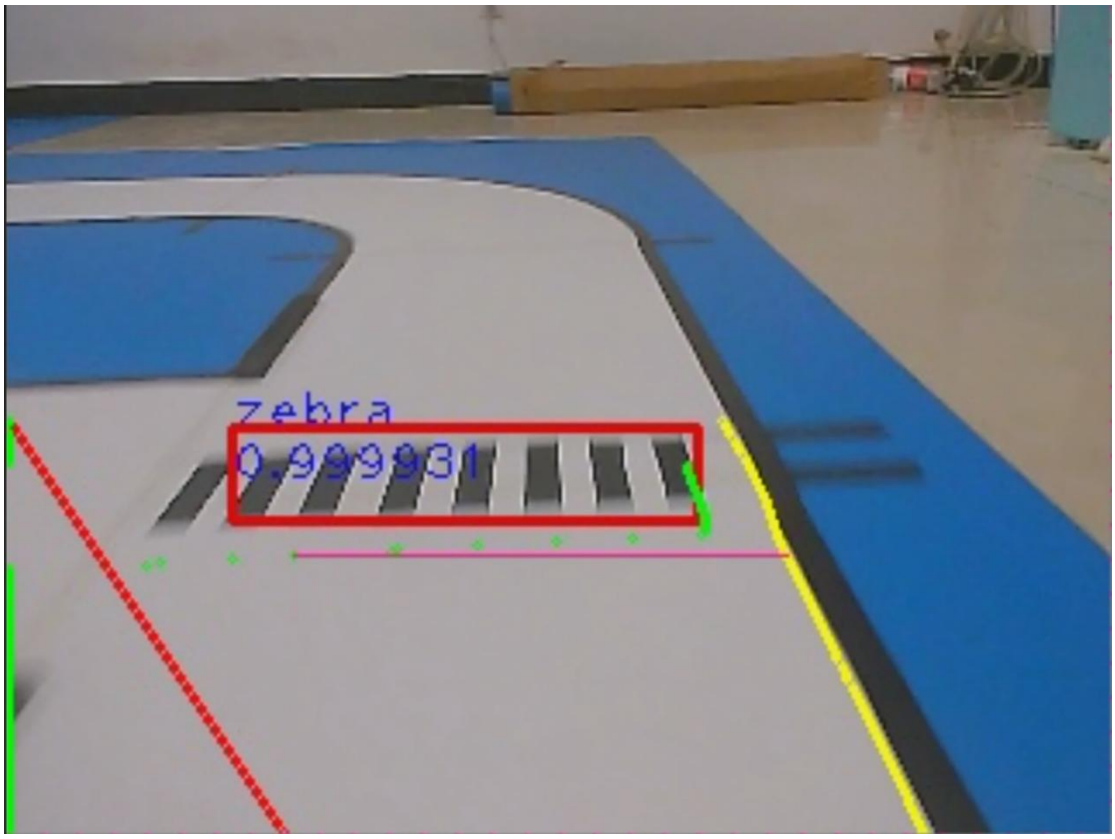


图 5.8.3

2 状态：已入库；停止

5.3.4 泛行区处理

泛行区分为 5 个状态来实现
AI 识别泛行区标志，连续 3 帧正式进入泛行区状态，并在泛行区内部设置一个状态机，并赋予四个不同状态-1、0、1、2、3，四个状态含义及巡线策略如下：

-1 状态：AI 识别加油站标志，连续 3 帧正式进入泛行区状态。

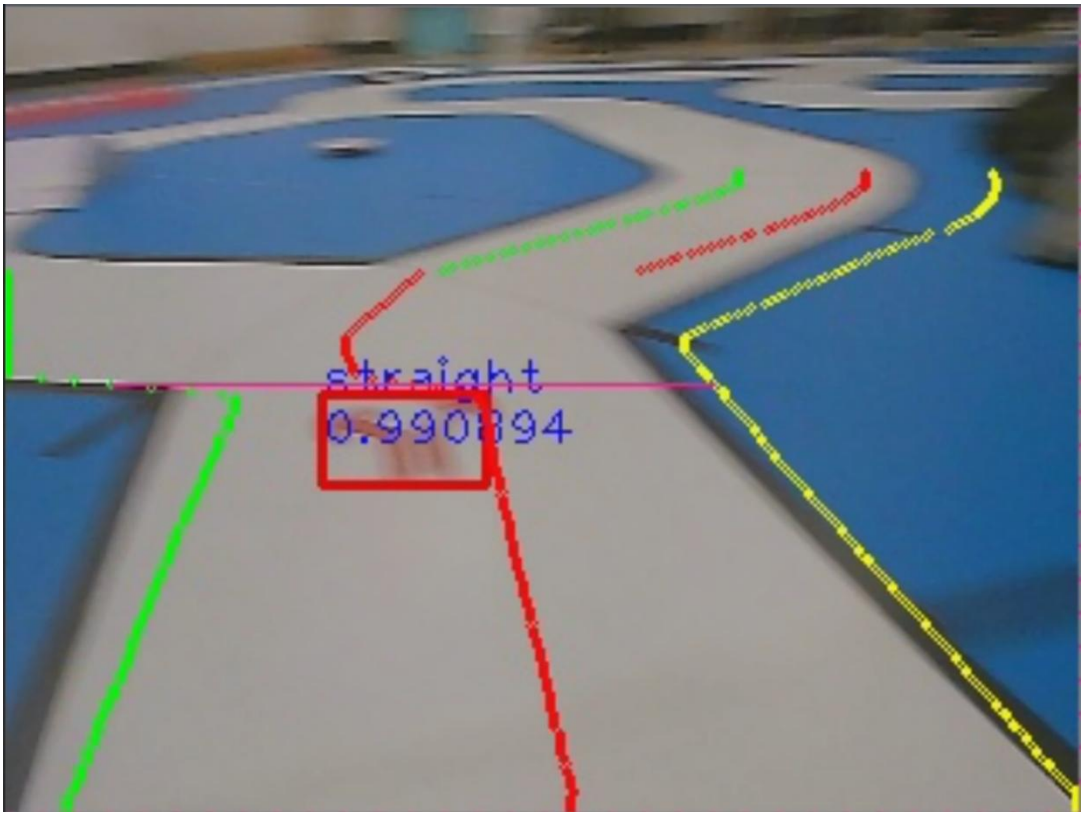


图 5.9.1

0 状态：AI 识别到泛行区标志，并且标志仍在视野内，寻找泛行区的最低点，连接图片最底部中点，形成中线

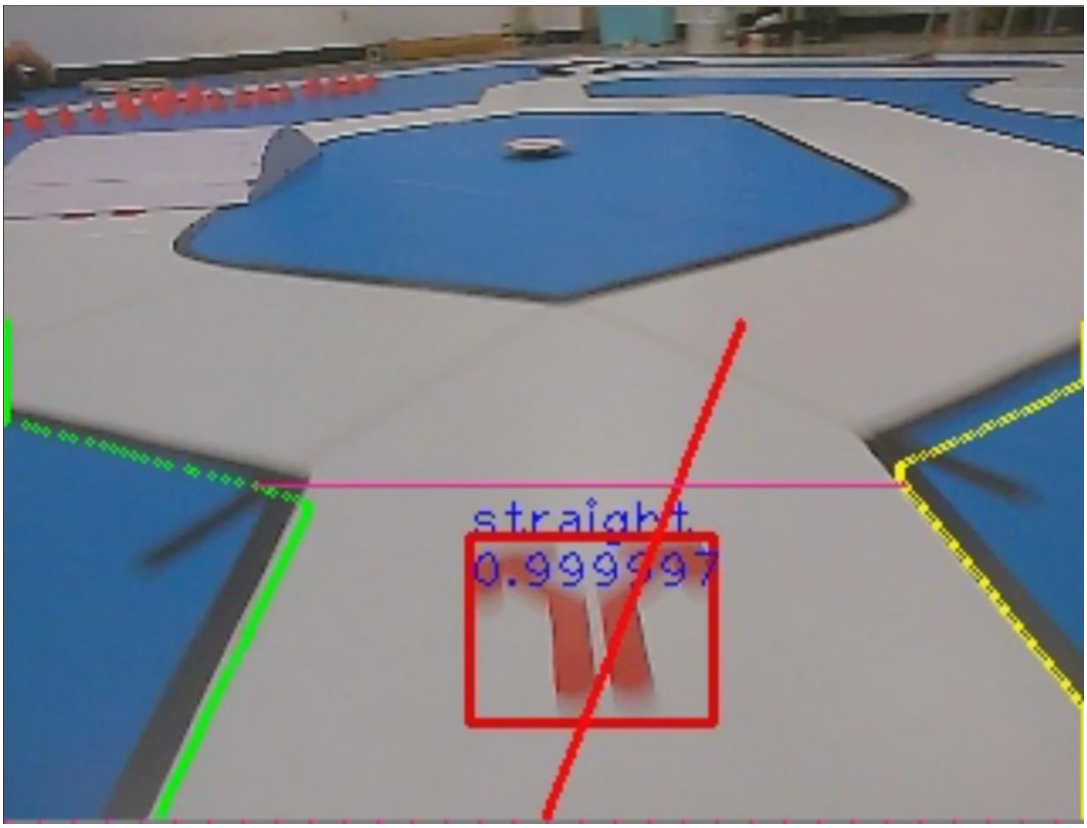


图 5.9.2

1 状态：识别到泛行区标志，并且标志不在视野内，且尚未进入泛行区，寻找泛行区的最低点，连接图片最底部中点，形成中线

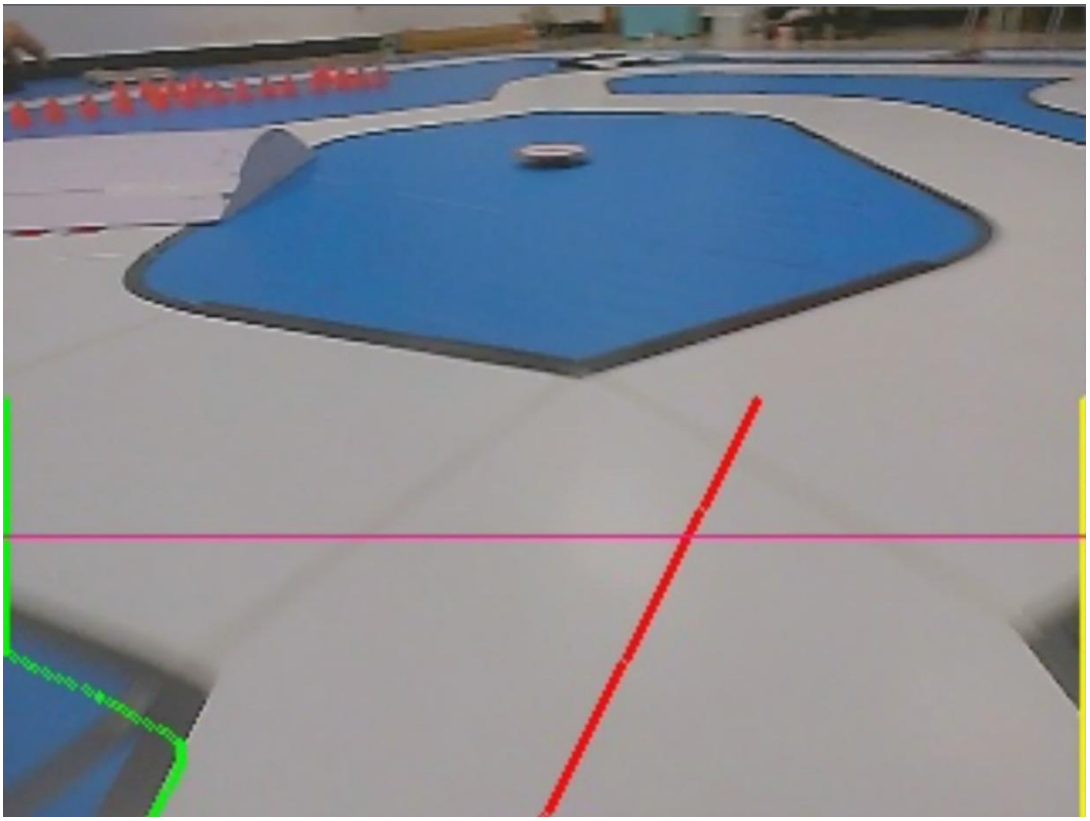


图 5.9.3

2 状态：即将进入泛行区，利用投影变换，判断此时的姿态下看 AI 识别出的禁行标志，应该左拐还是右拐

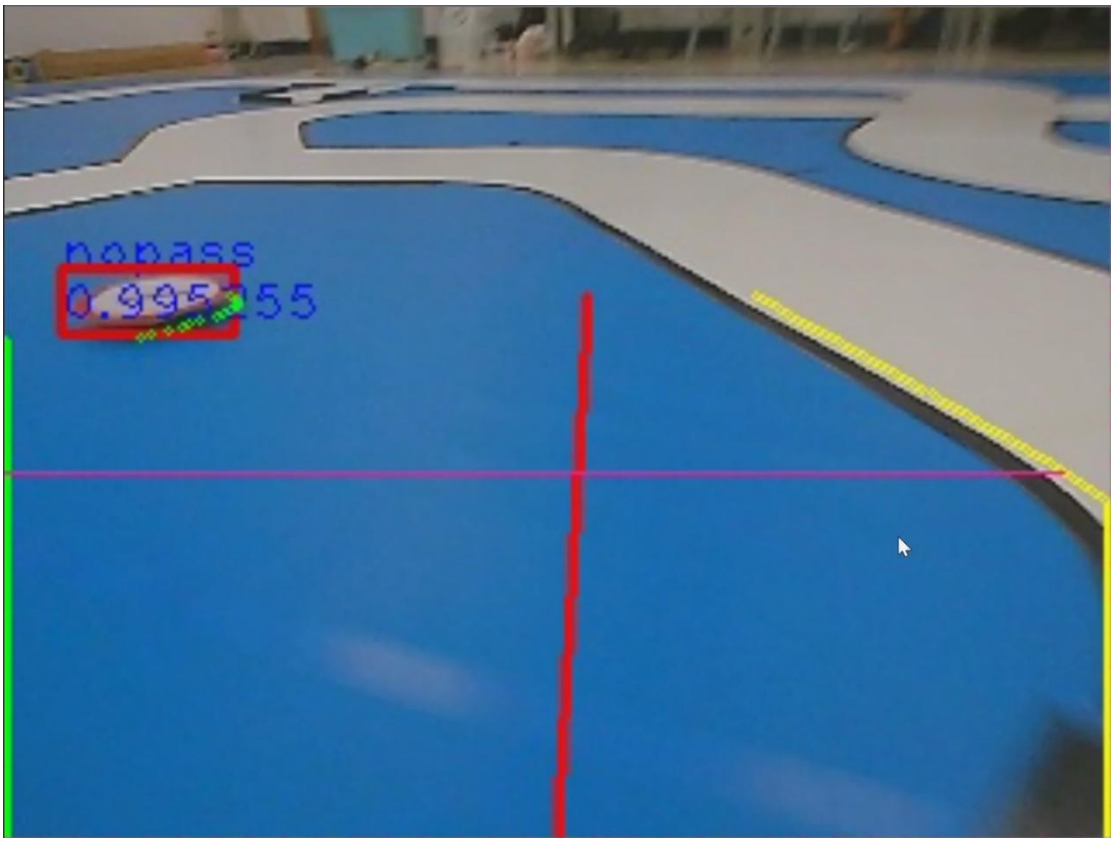


图 5.9.4

3 状态：已进入泛行区，在禁行标志位于直接到达出口的必经之路时，利用 2 状态的判断结果，结合 AI 识别到的禁行标志位置，左拐或右拐来绕开禁行标志；否则利用直接利用反色后的巡线离开泛行区，离开时将状态重置为 -1

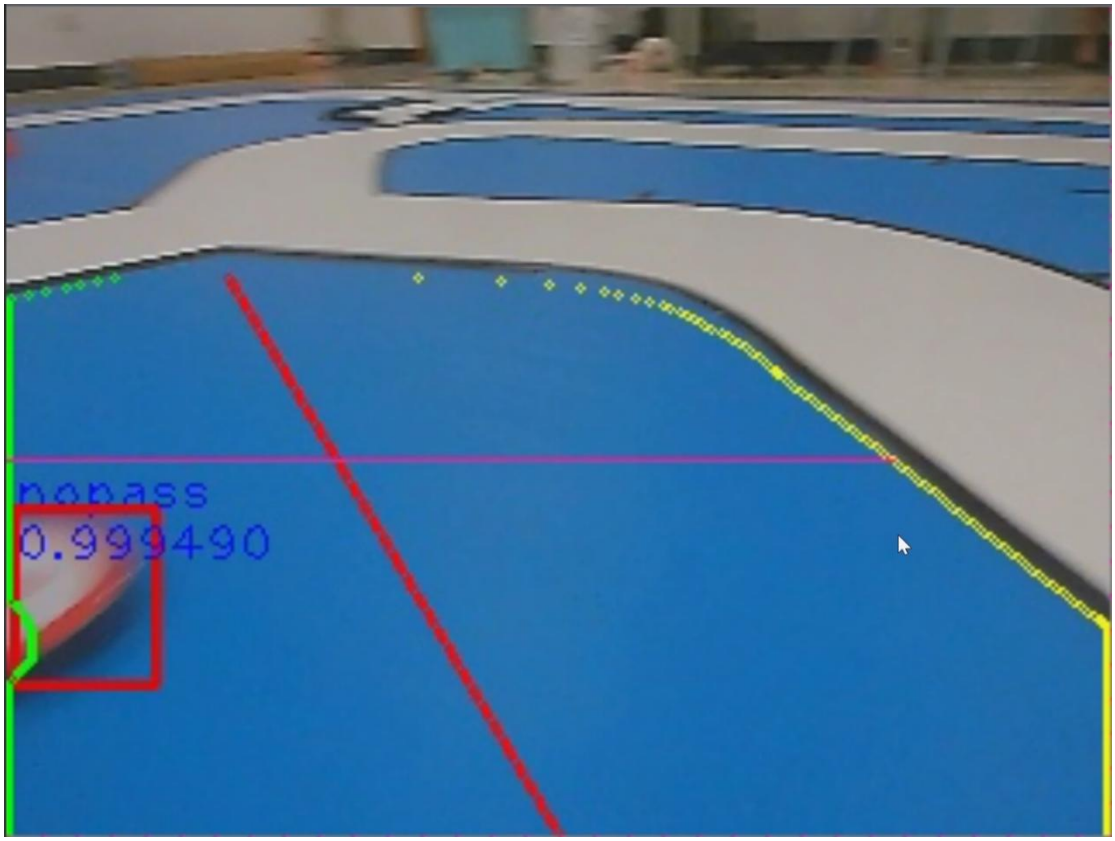


图 5.9.5

5.4 舵机控制转向打角控制

5.4.1 舵机整体控制思路

英飞凌芯片中 GTM 可以使用来产生固定频率的 PWM 波，考虑到摄像头采集帧率约为 30fps，为保证实时性，我们决定在串口中断中使用 50HZ 的 PWM 实现舵机控制。

在上位机串口发送线程中，我们根据计算后的图像偏移，将其代入舵机位置式 PD 中进行计算限幅后实现指定占空比的 PWM 输出。

5.4.2 舵机 PD 计算公式

由于舵机控制中 PID 中的 I 项（即积分项）不适用其中，因此使用 PD 控制，公式如下

```
engine_deviation = Engine_InitialDuty + Engine_P * Engine_Error_new +
    Engine_D * (Engine_Error_new - Engine_Error_prev)
```

5.4.3 动态 PD 参数的选取

在智能车行进在不同的赛道元素中，如果使用相同的 PD 会造成其在直道上运行抖动和弯道上转向不够或是不连续的情况出现。为此，我们测试了不同赛道上适合的 PD 参数，用摄像头赛道中值误差来对其不同参数拟合曲线。

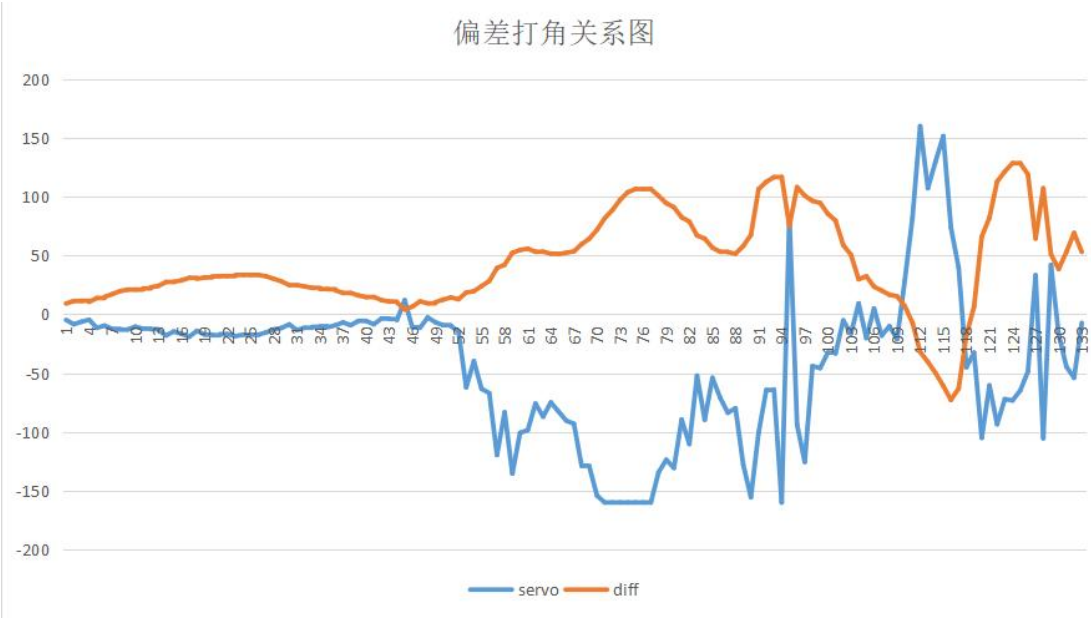


图 5.10

5.5 电机速度控制算法

5.5.1 增量式 PID 算法

增量式 PID 控制，数字 PID 控制算法的一种基本形式，是通过对控制量的增量（本次控制量和上次控制量的差值）进行 PID 控制的一种控制算法。

1）增量式 PID 的介绍

增量式 PID 控制，数字 PID 控制算法的一种基本形式，是通过对控制量的增量（本次控制量和上次控制量的差值）进行 PID 控制的一种控制算法。

增量式PID算法控制结构图

图 5.11

2）算法实现

Kp 为比例系数，KI 为积分系数，KD 为微分系数，e（k）为误差。 e（k）: 用户设定的值（目标值） - 控制对象的当前的状态值

以下为增量式 PID 的计算公式：

$$\Delta u(k) = u(k) - u(k-1)$$

$$= K_P[e(k) - e(k-1)] + K_I e(k) + K_D[e(k) - 2e(k-1) + e(k-2)]$$

图 5.12

5.6 车辆自启动

通过串口和定时器中断实现类似看门狗的逻辑，串口接收中断喂狗，定时器中断消耗狗粮，狗粮不够则 BARK，将车辆停止。实现了车辆随程序启停的功能。

第六章 开发工具、平台调试说明

6.1 开发工具

单片机裸机程序的开发是在 ADS 下进行的，通过交叉编译的方式编译可执行文件并烧录至单片机 flash 中。

AURIX™ Development Studio（以下简称 ADS）是英飞凌公司于 2019 年底推出的免费的集成开发环境，支持英飞凌 TriCore™内核 AURIX™ 系列 MCU；ADS 是一个完整的开发环境，包含了 Eclipse IDE、C 编译器、Multi-core 调试器、英飞凌底层驱动库（low-level driver,iLLD),同时对于编辑、编译及调试应用代码没有时间及代码大小的限制。

上位机运行 arm 架构的 linux 操作系统，由于 Edgeboard 不便于本地调试，则采用 ssh 连接的方式进行远程交互。

VScode 全拼是 VisualStudioCode,是由微软研发的一款免费、开源的跨平台代码编辑器,拥有许多方便的功能和插件。我们通过 VScode 的 remote-ssh 插件进行上位机代码的编写与调试。

使用 arm-gcc 配合自定义的 Makefile 对工程进行编译和管理。

6.2 调试软件

使用的调试软件包括 mjpg-streamer 和 serial_port。主要分为两部分的仿真调试 1）图像调试，2）控制调试。

6.2.1 图像调试

使用 mjpg-streamer 将 opencv 处理后的图像推流至 html 客户端，画面延迟可忽略。或者在采集赛道画面后，可以在电脑本地使用 opencv 进行逐帧调试。

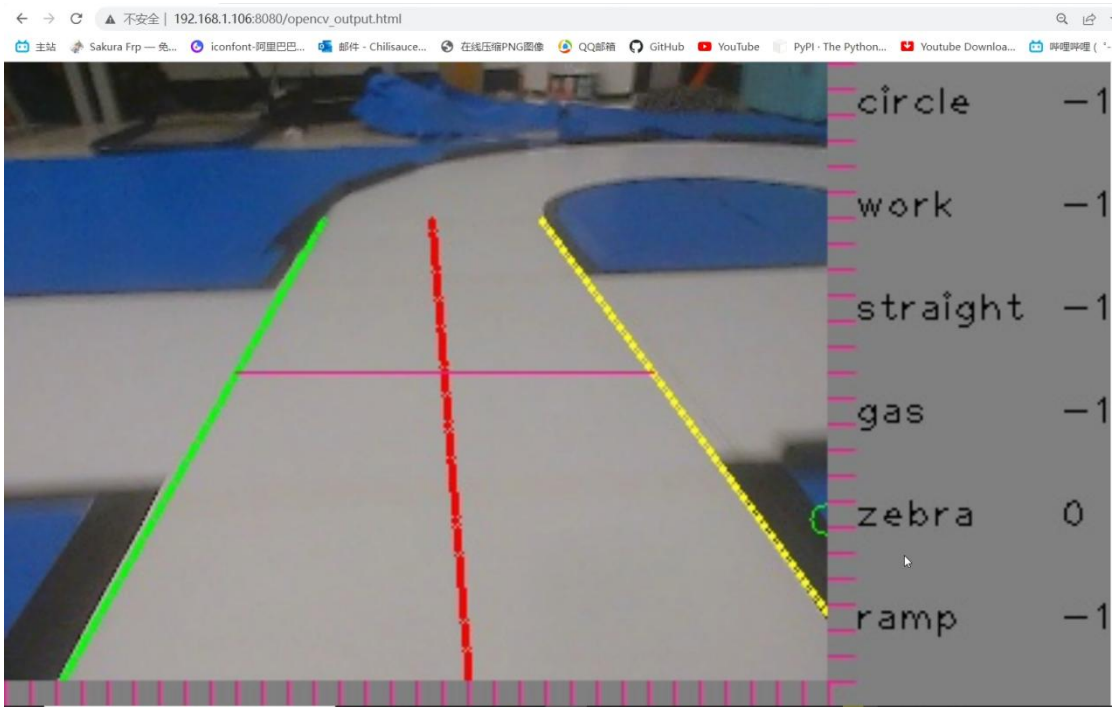


图 6.1

6.2.2 控制调试

电机 pid 参数使用串口传输至 serial_port 进行波形进行对比分析。

通过让电机转动固定时间，可设定多段阶跃目标值或步进目标值来分别测试电机响应速度与跟随程度，后将连续数据通过串口传输至上位机绘制波形。

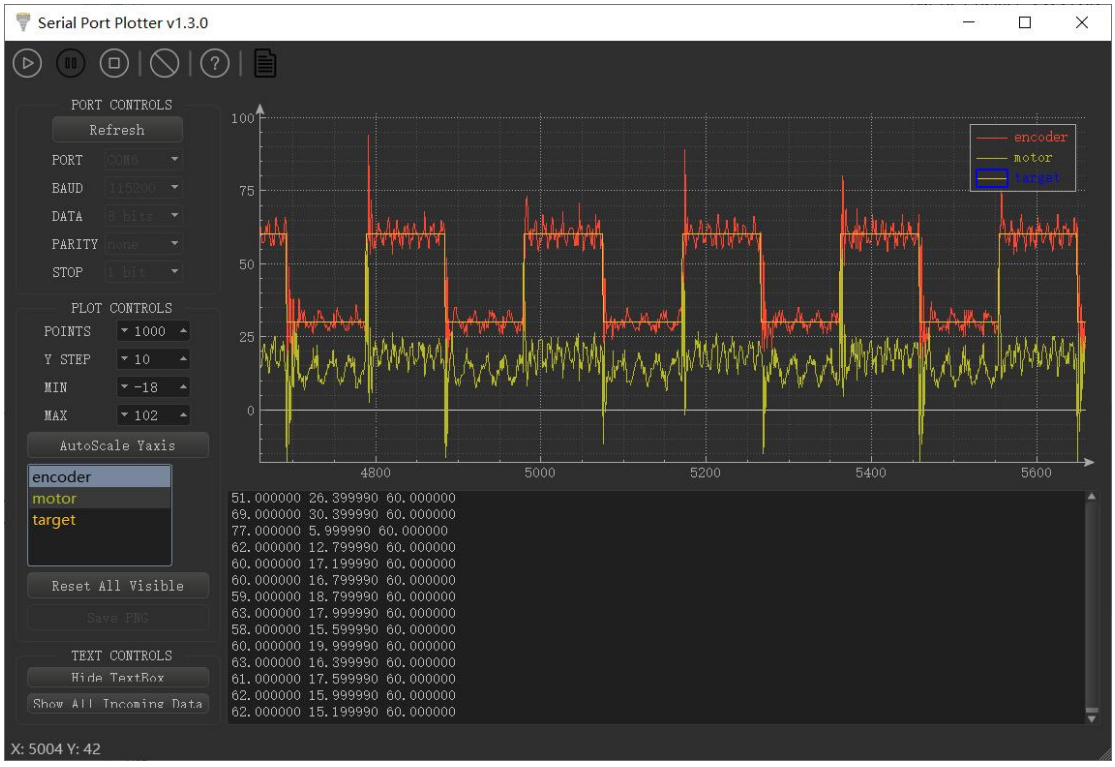


图 6.2 serial_port 调试过程

空载参数与负载参数相差较大，测试时应在空载时获得参数大致数量级，后直接进入负载落地整定参数，一般来说 pid 参数会有 3 到 10 倍乃至更多的变化。

舵机调试主要是通过 ssh 终端，在智能车运行过程中，传输舵机 pd 输出的打角值，与图像进行对比，不断调整相关参数，提高反应速度与稳定状态。

第七章 致谢

在备赛期间，我们小组遇到过很多问题，从传感器选型与方案确定，到后来的软硬件联合调试。在解决一个个问题之后，我们在技术上在不断成长，思想上不断成熟。而在这过程中，离不开学校，老师和学长的支持。

首先，我们要感谢学校对这次比赛的重视，场地和经费方面都得到了学校和学院的大力支持。没有他们的支持，我们的车模绝对上不了赛场。其次，我们要感谢指导老师的悉心教导，没有他们在方向上的指导和整体思路的规划安排，我们很难有今天的成果。最后，我们要感谢指导学长们对我们的经验传授与技术上的支持，感谢我们和谐的实验室氛围，感谢学弟们在赛道制作和其它方面的帮助。同时也感谢比赛组委会能组织这样一项有意义的比赛。

当然，也要谢谢各大智能车群与 csdn 相互分享经验的车友们，一场比赛使得大家拥有相同目标，是竞争，也是合作。

参考文献

[1] 童诗白，华成英. 模拟电子技术基础[M]. 北京: 高等教育出版社. 2000

[2] 卓晴，黄开胜，邵贝贝. 学做智能车 [M]. 北京: 北京航空航天大学出版社. 2007.

[3] 谢辉，徐辉.英飞凌多核单片机应用技术：AURIXTM 三天入门篇[M]. 天津: 天津大学出版社. 2017.

[4] 王成元，夏加宽，孙宜标. 现代电机控制技术[M]. 北京:机械工业出版社. 2014.

[5] Hughes, Austin. Electric Motors and Drives. Amsterdam: Elsevier. 2013.

[6] 尹怡欣, 陶永华. 新型 PID 控制及其应用[M]. 北京: 机械工业出版社. 1998.

[7] 尹勇. Protel DXP 电路设计入门与进阶[M]. 北京:科学出版社. 2004.

[8]Augmentation for small object detection, CVPR 2019.

附录

附录.1　Cpu0_Main.c 程序源代码：

```
#include "headfile.h"
#pragma section all "cpu0_dsram"
#include "battery.h"
#include "driver.h"
#include "beep.h"
#include "encoder.h"
#include "pid.h"
#include "imu.h"

//宏定义
#define RATIO          3                    //编码器速度:占空比
#define IMU_INTERVAL    5
#define ENC_INTERVAL    5                    //编码器定时器中断周期 ms
#define PID_INTERVAL    10                   //PID 定时器中断周期 ms
#define MAIN_DELAY      10                   //主循环延时 ms
#define TARGET_SPEED    0

//电机初始占空比
#define SQUARE_WAVE     0                    //方波调试
#define VISUAL_SCOPE    0                    //虚拟示波器调试

extern ICM icm_data;
extern EULER eulerAngle;
//全局变量定义
float battery_voltage[VOLT_DEPTH] = {0.0};  //电池电压 FIFO 容量在 battery.h 中
设置
int encoder_speed = 0;                      //读取的编码器速度
float motor_output = 0;                     //电机最终占空比输出
float servo_output = 0;                     //舵机最终占空比输出

int core0_main(void)
{
    motor_output = TARGET_SPEED / RATIO;
    get_clk();
    uart_init(UART_1, 115200, UART1_TX_P11_12, UART1_RX_P11_10);
    beep_init();
    lcd_init();
    battery_init();
```

```
    encoder_init(ENC_INTERVAL);             //设置定时器中断周期
    pid_init(TARGET_SPEED, PID_INTERVAL);   //设置目标期望速度初始值 和 定时器
中断周期
    driver_init(motor_output);              //设置电机起始速度

#if SQUARE_WAVE
    //1s 方波
    pit_enable_interrupt(CCU6_1, PIT_CH0);
    pit_init(CCU6_1, PIT_CH0, 1000*1000);
    pit_start(CCU6_1, PIT_CH0);
#endif

    //等待所有核心初始化完毕
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 0xFFFF);
    enableInterrupts();

    while (TRUE)
    {
        //获取电源电压
      update_battery_voltage();
      check_battery_voltage();
#if VISUAL_SCOPE
        printf("$%f %f %f;", (float)encoder_speed, motor_output,
(float)pid_get_target());
#else

#endif
        systick_delay_ms(STM0, MAIN_DELAY);
    }



    return 0;
}

#pragma section all restore
```

附录.2　isr.c 程序源代码：

```
#include "isr_config.h"
#include "isr.h"
#include "encoder.h"
#include "pid.h"
#include "driver.h"
#include "beep.h"
#include "SEEKFREE_ICM20602.h"

#define MSG_FIFO 4
#define LOAD_THRESHOLD 5
#define MSG_HEAD 0XFF
#define M_L 0x00
#define M_R 0x80
#define DOG_BARK 10

uint8 dog_feed = 0;
uint8 exam = 0x00;
uint8 data[MSG_FIFO];
uint8 msg_index = 0;
uint8 load_cnt = 0;
boolean load_start = FALSE;
boolean msg_start = FALSE;
extern float servo_output;


//PIT 中断函数  示例
IFX_INTERRUPT(cc60_pit_ch0_isr, 0, CCU6_0_CH0_ISR_PRIORITY)
{
    enableInterrupts();//开启中断嵌套
    PIT_CLEAR_FLAG(CCU6_0, PIT_CH0);
    encoder_speed = encoder_read();
    encoder_clear();
}


IFX_INTERRUPT(cc60_pit_ch1_isr, 0, CCU6_0_CH1_ISR_PRIORITY)
{
    enableInterrupts();//开启中断嵌套
    PIT_CLEAR_FLAG(CCU6_0, PIT_CH1);
    m_pid((float)encoder_speed);
```

```
    if(dog_feed > 0 && load_start)
        dog_feed--;
    if(dog_feed <= 0 && load_start)
    {
        load_start = FALSE;
        load_cnt = 0;
        pid_change_target(0);
        servo_control(0);
    }

    motor_control(motor_output * !low_voltage);
}

IFX_INTERRUPT(cc61_pit_ch0_isr, 0, CCU6_1_CH0_ISR_PRIORITY)
{
    enableInterrupts();//开启中断嵌套
    PIT_CLEAR_FLAG(CCU6_1, PIT_CH0);
    //方波调试
    if(pid_get_target() == 30)
        pid_change_target(60);
    else
        pid_change_target(30);
}

IFX_INTERRUPT(cc61_pit_ch1_isr, 0, CCU6_1_CH1_ISR_PRIORITY)
{
    enableInterrupts();//开启中断嵌套
    PIT_CLEAR_FLAG(CCU6_1, PIT_CH1);
    get_icm20602_accdata();
    get_icm20602_gyro();
    icmGetValues();
}




IFX_INTERRUPT(eru_ch0_ch4_isr, 0, ERU_CH0_CH4_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    if(GET_GPIO_FLAG(ERU_CH0_REQ4_P10_7))//通道0中断
    {
        CLEAR_GPIO_FLAG(ERU_CH0_REQ4_P10_7);
```

```
    }

    if(GET_GPIO_FLAG(ERU_CH4_REQ13_P15_5))//通道 4 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH4_REQ13_P15_5);
    }
}

IFX_INTERRUPT(eru_ch1_ch5_isr, 0, ERU_CH1_CH5_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    if(GET_GPIO_FLAG(ERU_CH1_REQ5_P10_8))//通道 1 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH1_REQ5_P10_8);
    }

    if(GET_GPIO_FLAG(ERU_CH5_REQ1_P15_8))//通道 5 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH5_REQ1_P15_8);
    }
}


IFX_INTERRUPT(eru_ch2_ch6_isr, 0, ERU_CH2_CH6_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    if(GET_GPIO_FLAG(ERU_CH2_REQ7_P00_4))//通道 2 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH2_REQ7_P00_4);

    }
    if(GET_GPIO_FLAG(ERU_CH6_REQ9_P20_0))//通道 6 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH6_REQ9_P20_0);

    }
}



IFX_INTERRUPT(eru_ch3_ch7_isr, 0, ERU_CH3_CH7_INT_PRIO)
{
```

```
    enableInterrupts();//开启中断嵌套
    if(GET_GPIO_FLAG(ERU_CH3_REQ6_P02_0))//通道 3 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH3_REQ6_P02_0);
    }
    if(GET_GPIO_FLAG(ERU_CH7_REQ16_P15_1))//通道 7 中断
    {
        CLEAR_GPIO_FLAG(ERU_CH7_REQ16_P15_1);

    }
}



IFX_INTERRUPT(dma_ch5_isr, 0, ERU_DMA_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
}


//串口中断函数  示例
IFX_INTERRUPT(uart0_tx_isr, 0, UART0_TX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrTransmit(&uart0_handle);
}
IFX_INTERRUPT(uart0_rx_isr, 0, UART0_RX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrReceive(&uart0_handle);
}
IFX_INTERRUPT(uart0_er_isr, 0, UART0_ER_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrError(&uart0_handle);
}

IFX_INTERRUPT(uart1_tx_isr, 0, UART1_TX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrTransmit(&uart1_handle);
}
```

```
IFX_INTERRUPT(uart1_rx_isr, 0, UART1_RX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrReceive(&uart1_handle);
    uint8 dat = 0x00;
    if(uart_query(UART_1,&dat))
    {
        if(dat == MSG_HEAD)
        {
            msg_start = TRUE;
            msg_index = 0;
            memset(data, 0 ,MSG_FIFO);
        }
        if(msg_start)
        {
            data[msg_index++] = dat;
        }
        if(msg_index > MSG_FIFO)
        {
            msg_start = FALSE;
            memset(data, 0 ,MSG_FIFO);
            msg_index = 0;
        }
        if(msg_start && msg_index == MSG_FIFO)
        {
            msg_start = FALSE;
            msg_index = 0;
            if(data[1] >= data[2])
                exam = data[1] - data[2];
            else
                exam = data[2] - data[1];
            if(data[3] == exam)
            {
                if(data[0] == MSG_HEAD)
                {
                    uint8 dir = data[2] & 0x80;
                    uint8 speed = data[2] & 0x7F;
                    uint8 servo = data[1];
                    switch(dir)
                    {
                        case M_L:
                            servo_output = (float)(-servo);
```

```
                        break;
                    case M_R:
                        servo_output = (float)(servo);
                        break;
                    default:
                        break;
                }
                if(!load_start)
                {
                    load_cnt++;
                     if(load_cnt == LOAD_THRESHOLD)
                     {
                         load_start = TRUE;
                         load_cnt = 0;
                     }
                }
                if(dog_feed < DOG_BARK)
                    dog_feed = DOG_BARK;
                if(!load_start)
                {
                    pid_change_target(0);
                    servo_control(0);
                }
                else
                {
                    if(pid_get_target() != speed)
                        pid_change_target(speed);
                    if(speed)
                        servo_control(servo_output);
                    else
                        servo_control(0);
                }
            }
        }
        memset(data, 0 ,MSG_FIFO);
    }
  }
}
IFX_INTERRUPT(uart1_er_isr, 0, UART1_ER_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrError(&uart1_handle);
```

```
}

IFX_INTERRUPT(uart2_tx_isr, 0, UART2_TX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrTransmit(&uart2_handle);
}
IFX_INTERRUPT(uart2_rx_isr, 0, UART2_RX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrReceive(&uart2_handle);
}
IFX_INTERRUPT(uart2_er_isr, 0, UART2_ER_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrError(&uart2_handle);
}

IFX_INTERRUPT(uart3_tx_isr, 0, UART3_TX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrTransmit(&uart3_handle);
}
IFX_INTERRUPT(uart3_rx_isr, 0, UART3_RX_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrReceive(&uart3_handle);
}
IFX_INTERRUPT(uart3_er_isr, 0, UART3_ER_INT_PRIO)
{
    enableInterrupts();//开启中断嵌套
    IfxAsclin_Asc_isrError(&uart3_handle);
}
```

附录.3　main.cpp 程序源代码：

```
#include "main.hpp"
#define THREAD_NUM 4
#define DEVICE 0
#define LAUNCH "launch.json"
#define M_L 0x00
#define M_R 0x80

uint8_t head = 0xFF;                //串口消息帧帧头
uint8_t set_speed = 0;              //电机目标转速
bool tids_1_done = false;           //线程 1 结束标志位
bool predict_update = false;        //预测图片更新标志位
bool out_range = false;             //出赛道标志位
int ret;                           //定义状态接收变量
int ai_state = 0;
```

```
Mat predict_img, predict_img_copy;  //预测后画面
Launch_Param param;                 //启动参数结构体
pthread_mutex_t mutex;              //定义互斥锁
pthread_t tids[THREAD_NUM];         //定义线程数组
ofstream servo_data;                //定义记录文件流
```

```
void* predict(void* img)
{
    while(1)
    {
        //互斥锁上锁
        pthread_mutex_lock(&mutex);
        Mat* img_copy = (Mat*)img;
        if(!img_copy->empty() && predict_update)
        {
            my_predict(*img_copy);
            predict_img_copy = predict_img.clone();
            predict_update = false;
        }
        //互斥锁解锁
        pthread_mutex_unlock(&mutex);
    }
}
```

```
void* read_cam(void* args)
```

```
{
    Mat raw, gray, binary, dilate_dst, rgb, copy, blur, raw_copy,invert, red, green, g_binary;
    vector<Mat> ch;
    //初始化摄像头
    VideoCapture cap;
    //文件名序号
    int index = 0;
    //文件名字符数组
    char file_name[40];
    //画布边沿宽度
    int canvas_border = 3;
    int canvas_thickness = 10 + canvas_border * 2;
    Mat canvas(PIC_H + canvas_thickness, PIC_W * 2 + canvas_border + canvas_thickness, CV_8UC3,
Scalar(128, 128, 128));
    Mat text_cover(10 + canvas_border, 10, CV_8UC3, Scalar(255, 0, 128));
    Mat mask = canvas(Rect(canvas_border * 2 +PIC_W, canvas_border, PIC_W, PIC_H));
    Mat text_mask = canvas(Rect(PIC_W * 2, canvas_border * 2, 10 + canvas_border, 10));
    Mat copy_mask = canvas(Rect(canvas_border, canvas_border, PIC_W, PIC_H));

    Mat ai_out(PIC_H + 10, PIC_W + 110, CV_8UC3, Scalar(128, 128, 128));
    Mat ai_text_mask = ai_out(Rect(PIC_W + 90, 0, 20, PIC_H + 10));
```

```
    cap.open("/dev/video0");
    cap.set(CAP_PROP_FRAME_WIDTH, PIC_W);
    cap.set(CAP_PROP_FRAME_HEIGHT, PIC_H);
    cout << "resolution: " << cap.get(CAP_PROP_FRAME_WIDTH) << "*" << cap.get(CAP_PROP_FRAME_HEIGHT)
<< endl;
```

```
    if(!cap.isOpened())
    {
        cerr << "ERROR! Unable to open camera\n";
        exit(-1);
    }
```

```
    if(param.predict)
    {
        ret = pthread_create(&tids[2], NULL  , predict, (void*)&predict_img);
        if(ret)
        {
            cout << "fail to create thread 2" << endl;
            exit(-1);
        }
    }
```

```
if(param.output == 2)
{
    char text[20];
    sprintf(text, "circle");
    Point origin(PIC_W + 10, 20);
    putText(ai_out, text, origin, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
    sprintf(text, "work");
    Point origin1(PIC_W + 10, 60);
    putText(ai_out, text, origin1, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
    sprintf(text, "straight");
    Point origin2(PIC_W + 10, 100);
    putText(ai_out, text, origin2, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
    sprintf(text, "gas");
    Point origin3(PIC_W + 10, 140);
    putText(ai_out, text, origin3, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
    sprintf(text, "zebra");
    Point origin4(PIC_W + 10, 180);
    putText(ai_out, text, origin4, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
    sprintf(text, "ramp");
    Point origin5(PIC_W + 10, 220);
    putText(ai_out, text, origin5, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
}
```

```
while(1)
{
    //读取单通道图片
    cap >> raw;
    flip(raw, raw, -1);
    split(raw, ch);
    red = ch[2];
    green = ch[0]/3 + ch[1]/3 + ch[2]/3;
    if(raw.empty())
    {
        cerr << "ERROR! blank frame grabbed\n";
        break;
    }

    //固定阈值二值化
    threshold(red, binary, param.threshold, 255, CV_THRESH_BINARY);
    threshold(green, g_binary, param.threshold, 255, CV_THRESH_BINARY);
    //滤除噪点
    Mat element = getStructuringElement(MORPH_RECT,Size(4,4), Point(-1,-1));
```

```
        dilate(binary, binary, element, Point(-1,-1));
        dilate(g_binary, g_binary, element, Point(-1,-1));
        element = getStructuringElement(MORPH_RECT,Size(5,5), Point(-1,-1));
        erode(binary, binary, element, Point(-1,-1));
        erode(g_binary, g_binary, element, Point(-1,-1));
        //互斥锁上锁
        pthread_mutex_lock(&mutex);
        if(param.predict)
        {
            predict_img = raw.clone();
            predict_update = true;
        }

        if(!ai_state)
        {
            //得到有效最高行
            top_line = find_mid(&binary,5,70);
            if(!out_range)
                out_range = check_range(spot);
            //绘制初始补线图片
            if(param.output == 1)
            {
                cvtColor(binary, copy, COLOR_GRAY2RGB);
                for(int i = PIC_H - 1; i > top_line; i--)
                {
                    Point p_mid(spot[i].m_spot, i);
                    Point p_l(spot[i].l_border, i);
                    Point p_r(spot[i].r_border, i);
                    circle(copy, p_mid, 1, Scalar(0, 0, 255));
                    circle(copy, p_l, 1, Scalar(0, 255, 0));
                    circle(copy, p_r, 1, Scalar(0, 255, 255));
                }
                line(copy, Point(spot[TARGET_LINE].l_border, TARGET_LINE),
Point(spot[TARGET_LINE].r_border, TARGET_LINE), Scalar(147, 20, 255));
                copy.copyTo(copy_mask);
            }
            //寻找十字
            find_cross(spot, top_line);
            //寻找环岛
            //find_circle(spot, top_line);
        }
```

```
        if(param.predict)
```

```
    {
        if(enter_state == STRAIGHT && straight_state>=0)
            if(straight_state>1&&straight_state<4)
            {
                bitwise_not(binary,invert);
                top_line = find_mid(&invert, 8, 60);
            }
            else
            {
                top_line = find_mid(&binary, 4, 0);
            }
        if(enter_state == ZEBRA && zebra_state <= 3)
        {
            if(zebra_state == 0)
            {
                top_line = find_mid(&binary, 8, 0);
            }
            else
            {
                top_line = find_mid(&binary, 4, 0);
            }
        }

        if(enter_state == RAMP && (ramp_state>=0)){
            top_line = find_mid(&binary, 4, 70);
        }

        if(enter_state == GASTATION)
        {
            if(gas_state <= 1)
                top_line = find_mid(&binary, 3, 0);
            else
                top_line = find_mid(&g_binary, 3, 0);
        }
```

```
        if(enter_state == CONSTRUCTION)
        {
            if(construction_state <= 1)
                top_line = find_mid(&binary, 3, 0);
            else
                top_line = find_mid(&g_binary, 3, 0);
        }
        if(enter_state && circle_state >= 0)
```

```
                circle_state = -1;
            ai_state = find_icon(spot, myresult, top_line, raw);
        }
        else
            ai_state = 0;
        //互斥锁解锁
        pthread_mutex_unlock(&mutex);
```

```
        //绘制最终补线图片
        if(param.output == 2)
        {
            if(enter_state == GASTATION && gas_state >= 0 && !img_binary.empty() &&
param.gas_invert)
                cvtColor(img_binary, rgb, COLOR_GRAY2RGB);
            if(enter_state == CONSTRUCTION && construction_state >= 0
&& !construction_img_binary.empty() && param.construction_invert)
                cvtColor(construction_img_binary, rgb, COLOR_GRAY2RGB);
```

```
            for(int i = PIC_H - 1; i > top_line; i--)
            {
                if((enter_state == GASTATION && gas_state >= 0 && !img_binary.empty() &&
param.gas_invert)
                || (enter_state == CONSTRUCTION && construction_state >= 0
&& !construction_img_binary.empty() && param.construction_invert))
                {
                    Point p_mid(spot[i].m_spot, i);
                    Point p_l(spot[i].l_border, i);
                    Point p_r(spot[i].r_border, i);
                    circle(rgb, p_mid, 1, Scalar(0, 0, 255));
                    circle(rgb, p_l, 1, Scalar(0, 255, 0));
                    circle(rgb, p_r, 1, Scalar(0, 255, 255));
                }
                else
                {
                    Point p_mid(spot[i].m_spot, i);
                    Point p_l(spot[i].l_border, i);
                    Point p_r(spot[i].r_border, i);
                    circle(predict_img_copy, p_mid, 1, Scalar(0, 0, 255));
                    circle(predict_img_copy, p_l, 1, Scalar(0, 255, 0));
                    circle(predict_img_copy, p_r, 1, Scalar(0, 255, 255));
                }
            }
```

```
            if((enter_state == GASTATION && gas_state >= 0 && !img_binary.empty() &&
param.gas_invert)
                || (enter_state == CONSTRUCTION && construction_state >= 0
&& !construction_img_binary.empty() && param.construction_invert))
                line(rgb, Point(spot[TARGET_LINE].l_border, TARGET_LINE),
Point(spot[TARGET_LINE].r_border, TARGET_LINE), Scalar(147, 20, 255));
            else
                line(predict_img_copy, Point(spot[TARGET_LINE].l_border, TARGET_LINE),
Point(spot[TARGET_LINE].r_border, TARGET_LINE), Scalar(147, 20, 255));

            if(enter_state == GASTATION && gas_state >= 0 && !img_binary.empty() &&
param.gas_invert)
            {
                Point p(gas_x_global, gas_y_global);
                circle(rgb, p, 6, Scalar(0, 255, 0));
            }
            else if(enter_state == CONSTRUCTION && construction_state >= 0
&& !construction_img_binary.empty() && param.construction_invert)
            {
                Point p1(construction_x_global, construction_y_global);
                circle(rgb, p1, 6, Scalar(127, 127, 0));
            }

            Mat ai_text_cover = Mat::zeros(Size(20, PIC_H + 10), CV_8UC3);
            ai_text_cover.setTo(Scalar(128, 128, 128));
            char text[20];
            sprintf(text, "%d", circle_state);
            Point origin(0, 20);
            putText(ai_text_cover, text, origin, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
            sprintf(text, "%d", construction_state);
            Point origin1(0, 60);
            putText(ai_text_cover, text, origin1, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
            sprintf(text, "%d", straight_state);
            Point origin2(0, 100);
            putText(ai_text_cover, text, origin2, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
            sprintf(text, "%d", gas_state);
            Point origin3(0, 140);
            putText(ai_text_cover, text, origin3, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
            sprintf(text, "%d", zebra_state);
            Point origin4(0, 180);
            putText(ai_text_cover, text, origin4, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);
            sprintf(text, "%d", ramp_state);
            Point origin5(0, 220);
```

```
            putText(ai_text_cover, text, origin5, FONT_HERSHEY_PLAIN, 1,Scalar(0, 0, 0), 1);

            ai_text_cover.copyTo(ai_text_mask);
        }

        if(param.output == 1)
        {
            cvtColor(binary, rgb, COLOR_GRAY2RGB);
            for(int i = PIC_H - 1; i > top_line; i--)
            {
                Point p_mid(spot[i].m_spot, i);
                Point p_l(spot[i].l_border, i);
                Point p_r(spot[i].r_border, i);
                circle(rgb, p_mid, 1, Scalar(0, 0, 255));
                circle(rgb, p_l, 1, Scalar(0, 255, 0));
                circle(rgb, p_r, 1, Scalar(0, 255, 255));
            }
            line(rgb, Point(spot[TARGET_LINE].l_border, TARGET_LINE),
Point(spot[TARGET_LINE].r_border, TARGET_LINE), Scalar(147, 20, 255));
            circle(rgb, Point(inflexpoint[0][1], inflexpoint[0][0]), 4, Scalar(255, 0, 0));
            circle(rgb, Point(inflexpoint[1][1], inflexpoint[1][0]), 4, Scalar(255, 0, 0));
            circle(rgb, Point(inflexpoint[2][1], inflexpoint[2][0]), 4, Scalar(255, 0, 0));
            circle(rgb, Point(inflexpoint[3][1], inflexpoint[3][0]), 4, Scalar(255, 0, 0));
            char text[10];
            int baseline;
            sprintf(text, "%d", circle_state);
            Point origin(spot[TARGET_LINE].m_spot - 20, TARGET_LINE);
            putText(rgb, text, origin, FONT_HERSHEY_PLAIN, 1,Scalar(0, 255, 0), 1);
            sprintf(text, "%d", construction_state);
            Point origin1(spot[TARGET_LINE].m_spot, TARGET_LINE);
            putText(rgb, text, origin1, FONT_HERSHEY_PLAIN, 1,Scalar(0, 255, 80), 1);
            sprintf(text, "%d", straight_state);
            Point origin2(spot[TARGET_LINE].m_spot + 20, TARGET_LINE);
            putText(rgb, text, origin2, FONT_HERSHEY_PLAIN, 1,Scalar(0, 255, 160), 1);
            sprintf(text, "%d", gas_state);
            Point origin3(spot[TARGET_LINE].m_spot + 40, TARGET_LINE);
            putText(rgb, text, origin3, FONT_HERSHEY_PLAIN, 1,Scalar(0, 255, 240), 1);
            //绘制刻度线
            for(int i = canvas_border; i <= PIC_W + canvas_border; i += 10)
                line(canvas, Point(i, PIC_H + canvas_border * 2), Point(i, PIC_H + canvas_thickness),
Scalar(147, 20, 255));
            for(int i = PIC_W + canvas_border * 2; i <= PIC_W * 2 + canvas_border * 2; i += 10)
```

```
                line(canvas, Point(i, PIC_H + canvas_border * 2), Point(i, PIC_H + canvas_thickness),
Scalar(147, 20, 255));
            for(int i = canvas_border; i <= PIC_H; i += 10)
                line(canvas, Point(PIC_W * 2 + canvas_border * 3, i), Point(PIC_W * 2 +
canvas_thickness + canvas_border, i), Scalar(147, 20, 255));
            rgb.copyTo(mask);
        }

        //保存绘制图片
        if(param.save)
        {
            imwrite(param.save_path + to_string(index) + ".jpg", raw);
            index++;
        }
        if(param.output == 1)
            imwrite(param.stream_path + ".jpg",canvas);
        if(param.output == 2)
        {
            for(int i = 0; i < PIC_W + 10; i += 10)
                line(ai_out, Point(i, PIC_H), Point(i, PIC_H + 10), Scalar(147, 20, 255));
            for(int i = 0; i < PIC_H + 10; i += 10)
                line(ai_out, Point(PIC_W, i), Point(PIC_W + 10, i), Scalar(147, 20, 255));
            Mat p_mask(ai_out, Rect(0, 0, PIC_W, PIC_H));
            if((enter_state == GASTATION && gas_state >= 0 && !img_binary.empty() &&
param.gas_invert)
            || (enter_state == CONSTRUCTION && construction_state >= 0
&& !construction_img_binary.empty() && param.construction_invert))
            {
                rgb.copyTo(p_mask);
            }
            else
                predict_img_copy.copyTo(p_mask);
            imwrite(param.stream_path + ".jpg",ai_out);

        }
        if(param.output == 3)
            imwrite(param.stream_path + ".jpg",invert);
    }

    cvDestroyAllWindows();
    //线程 1 运行结束
    tids_1_done = true;
```

```
    //退出线程
    pthread_exit(NULL);
}
```

```
void* read_pic(void* args)
{
    Mat raw, gray, binary, dilate_dst, rgb, copy, blur;
    char file_name[40];
    Mat dual(PIC_H, PIC_W * 2 + 10, CV_8UC3, Scalar(128, 128, 128));
```

```
    for(int index = 270;index <= 910; index = index + 1)
    {
        raw = imread(param.pic_path + to_string(index) + ".jpg", CV_LOAD_IMAGE_ANYCOLOR);
        if(raw.empty())
        {
            cerr << "ERROR! blank frame grabbed\n";
        }
        GaussianBlur(raw, blur, Size(5,5), 5, 5);
        cvtColor(blur, gray, COLOR_RGB2GRAY);
        //大津法二值化
        threshold(gray, binary, 0, 255, THRESH_OTSU);
        //滤除噪点
        Mat element = getStructuringElement(MORPH_RECT,Size(3,3), Point(-1,-1));
        dilate(binary, binary, element, Point(-1,-1));
        element = getStructuringElement(MORPH_RECT,Size(4,4), Point(-1,-1));
        erode(binary, binary, element, Point(-1,-1));
        //互斥锁上锁
        pthread_mutex_lock(&mutex);
        //得到有效最高行
        top_line = find_mid(&binary, 3, 70);
        find_cross(spot, top_line);
        //find_circle(spot,top_line);
        //互斥锁解锁
        pthread_mutex_unlock(&mutex);
        //绘制彩色图片
        cvtColor(binary, rgb, COLOR_GRAY2RGB);
        for(int i = PIC_H - 1; i > top_line; i--)
        {
            Point p_mid(spot[i].m_spot, i);
            Point p_l(spot[i].l_border, i);
            Point p_r(spot[i].r_border, i);
            circle(rgb, p_mid, 1, Scalar(0, 0, 255));
            circle(rgb, p_l, 1, Scalar(0, 255, 0));
```

```
            circle(rgb, p_r, 1, Scalar(0, 255, 255));
        }
        line(rgb, Point(0, top_line), Point(PIC_W - 1, top_line), Scalar(147, 20, 255));

        //合并两张图片
        rgb.copyTo(dual.colRange(0, rgb.cols));
        copy.copyTo(dual.colRange(copy.cols + 10, dual.cols));
        //保存绘制图片
        sprintf(file_name, "Final%d.jpg", index);
        if(param.save)
            imwrite(file_name, dual);
    }
    cout << "read_pic is done" << endl;
    cvDestroyAllWindows();
    //线程1运行结束
    tids_1_done = true;
    //退出线程
    pthread_exit(NULL);
}
```

```
void* serial_send(void* args)
{
    while(1)
    {
        uint8_t data1, data2, data3, ref_sum_h, ref_sum_l, exam;
        //互斥锁解锁
        int diff, servo;
        uint8_t s_out, dir;
```

```
        pthread_mutex_lock(&mutex);
        servo = s_pid(spot, top_line);
        pthread_mutex_unlock(&mutex);
        if(servo <= 0)
            dir = M_L;  //left
        else
            dir = M_R;  //right

        if(zebra_state == 2 || out_range)
        {
            data1 = (uint8_t)abs(servo);
            data2 = dir | 0x00;//设置速度
        }
        else
```

```
        {
            data1 = (uint8_t)abs(servo);
            data2 = dir | (uint8_t)set_speed;
        }
```

```
        if(data1 >= data2)
            exam = data1 - data2;
        else
            exam = data2 - data1;
        if(top_line != -1)
        {
            driver->senddata(head);
            driver->senddata(data1);
            driver->senddata(data2);
            driver->senddata(exam);
        }

        usleep(1000);
    }
}
```

```
int get_param(char* setting)
{
    ifstream json_file(setting);
    if(!json_file.is_open())
    {
        cout << "can't open json file" << endl;
        return -1;
    }
    json j;
    json_file >> j;
    param.stream = j["source"]["stream"];
    param.stream_path = j["source"]["stream_path"];
    param.pic = j["source"]["pic"];
    param.pic_path = j["source"]["pic_path"];
    param.save = j["save"];
    param.save_path = j["save_path"];
    param.predict = j["predict"];
    param.uart = j["uart"];
    param.speed = j["set"]["speed"];
    param.l_kp = j["pid"]["left"]["kp"];
    param.l_kd = j["pid"]["left"]["kd"];
    param.l_mkp = j["pid"]["left"]["mkp"];
```

```
    param.l_mkd = j["pid"]["left"]["mkd"];
    param.l_lkp = j["pid"]["left"]["lkp"];
    param.l_lkd = j["pid"]["left"]["lkd"];
    param.r_kp = j["pid"]["right"]["kp"];
    param.r_kd = j["pid"]["right"]["kd"];
    param.r_mkp = j["pid"]["right"]["mkp"];
    param.r_mkd = j["pid"]["right"]["mkd"];
    param.r_lkp = j["pid"]["right"]["lkp"];
    param.r_lkd = j["pid"]["right"]["lkd"];
    param.output = j["output"];
    param.threshold = j["threshold"];
    param.k_fix = j["k_fix"];
    param.circle_debug = j["debug"]["circle"];
    param.cross_debug = j["debug"]["cross"];
    param.mid_debug = j["debug"]["mid"];
    param.zebra_debug = j["debug"]["zebra"];
    param.construction_debug = j["debug"]["construction"];
    param.cone_debug = j["debug"]["cone"];
    param.straight_debug = j["debug"]["straight"];
    param.servo_debug = j["debug"]["servo"];
    param.data_record = j["record"]["data_record"];
    param.record_name = j["record"]["record_name"];
    param.gas_enable = j["gas"]["gas_enable"];
    param.gas_out = j["gas"]["gas_out"];
    param.gas_invert = j["gas"]["gas_invert"];
    param.construction_enable = j["construction"]["construction_enable"];
    param.construction_invert = j["construction"]["construction_invert"];
    param.loop = j["loop"];
    param.print();
    return 0;
}
```

```
int main(int argc, char *argv[])
{
    if(argc < 2)
    {
        cout << "enter json file xx.json" << endl;
        exit(-1);
    }
    char* setting = argv[1];
    ret = get_param(setting);
```

```
    icon_param_init();
```

```
    if(param.data_record)
    {
        servo_data.open(param.record_name);
        if(!servo_data)
        {
            cout << "data record error" << endl;
            exit(-1);
        }
        servo_data << "servo" << "," << "diff" << "," << "x1" << "," << "y1" << "," << "x3" << ","
<< "y3" << endl;
    }
```

```
    if(ret)
    {
        cerr << "get param failed" << endl;
        exit(-1);
    }
    //初始化串口
    ret = uart_init();
    if(ret)
    {
        cerr << "uart initialization failed" << endl;
        exit(-1);
    }
    //初始化预测
    ret = predict_init();
    if(ret)
    {
        cerr << "predict initialization failed" << endl;
        exit(-1);
    }
    //初始化互斥锁
    pthread_mutex_init(&mutex, NULL);
    //启动 read 线程
    if(param.stream)
    {
        ret = pthread_create(&tids[0], NULL, read_cam, NULL);
            if(ret)
            {
                cout << "fail to create thread 0" << endl;
                exit(-1);
            }
```

```
    }
    else if(param.pic)
    {
        ret = pthread_create(&tids[0], NULL, read_pic, NULL);
            if(ret)
            {
                cout << "fail to create thread 0" << endl;
                exit(-1);
            }
    }
```

```
    if(param.uart)
    {
        ret = pthread_create(&tids[1], NULL, serial_send, NULL);
            if(ret)
            {
                cout << "fail to create thread 1" << endl;
                exit(-1);
            }
    }
```

```
    //等待第一个线程结束
    ret = pthread_join(tids[1], NULL);
    if (ret)
    {
        cout << "Error:unable to join thread 1," << ret << endl;
        exit(-1);
    }
    else
        cout << "trip end!!!" << endl;
    //销毁互斥锁
    pthread_mutex_destroy(&mutex);
    return 0;
}
```