第十七届全国大学生

智能汽车竞赛

技 术 报 告

完全模型组

学　　校:　同济大学

队伍名称:　智行·神话千里

参赛队员:　陈冠佑、郭子瞻、周宏韬、韩意

　　王相栋

指导老师:　张志明、余有灵

领队老师:　张志明

关于研究论文使用授权的说明

本人完全了解第十七届全国大学生智能汽车竞赛关于保留、使用研究论文的规定，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：_____

带队教师签名：_____

日　　期：2022.8.20

目　录

第一章 引言

1.1 全国大学生智能汽车竞赛介绍

全国大学生智能汽车竞赛是以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程实践活动，是教育部倡导的大学生科技竞赛之一。该竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，旨在促进高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神，为优秀人才的脱颖而出创造条件。

参赛选手须使用竞赛秘书处统一指定并负责采购竞赛车模，自行采用 32 位微控制器作为核心控制单元，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动、转向舵机控制等，完成智能汽车工程制作及调试，于指定日期与地点参加场地比赛。参赛队伍之名次（成绩）由赛车现场成功完成赛道比赛时间为主，技术方案及制作工程质量评分为辅来决定。

1.2 全国大学生智能车竞赛完全模型组介绍

全国大学生智能汽车竞赛是以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程的实践活动，是教育部倡导的大学生科技竞赛之一。竞赛以立足培养，重在参与，鼓励探索，追求卓越为指导思想，培养大学生的创意性科技竞赛能力。

近年来，人工智能这一科技浪潮正在深刻改变着世界，图像识别作为人工智能的一个主要方向已经开始渗透进日常生活的方方面面。完全模型组是第十七届全国大学智能汽车竞赛竞速比赛中依据百度 PaddlePaddle 人工智能平台，将人工智能技术向智能车竞赛竞速比赛自认延伸的一个组别。该组别采用由百度提供的 I 型车模，在百度的 EdgeBoard 与 Infineon 单片机控制下，在正常的赛道上，完全有人工智能图像识别模型支持车模在赛道上以及赛道下完成相应的运行动作。

1.2.1 赛事基本介绍

百度完全模型竞速赛分为线上资格赛、线下分区赛和全国总决赛三个阶段，组委会将综合考虑线上资格赛和线下分区赛的成绩来进行全国总决赛名额的选

拔，其中线上成绩占 10%，线下成绩占 90%。需要使用参赛选手必须使用开源深度学习平台飞桨完成模型的训练、推理和部署，不得使用其他深度学习平台或飞桨未包含的学习方法参赛。

完全模型组的基本比赛任务为：选手制作的车模完成从车库出发沿着赛道运行两周。车模需要分别通过道路设置的各种元素，识别道路中心的标志完成特殊路段通行。比赛时间从车模驶出车库到重新回到车库为止。如果车模没有能够停止在车库内停车区内，比赛时间加罚 5 秒钟。对于未完成的任务会通过相应的加罚时间叠加在比赛时间上。

1.2.2 竞赛规则解读

本次比赛完全模型组属于竞速组，行驶循环赛道，环形赛道是由赛道元素形成一个封闭赛道，如下图所示。

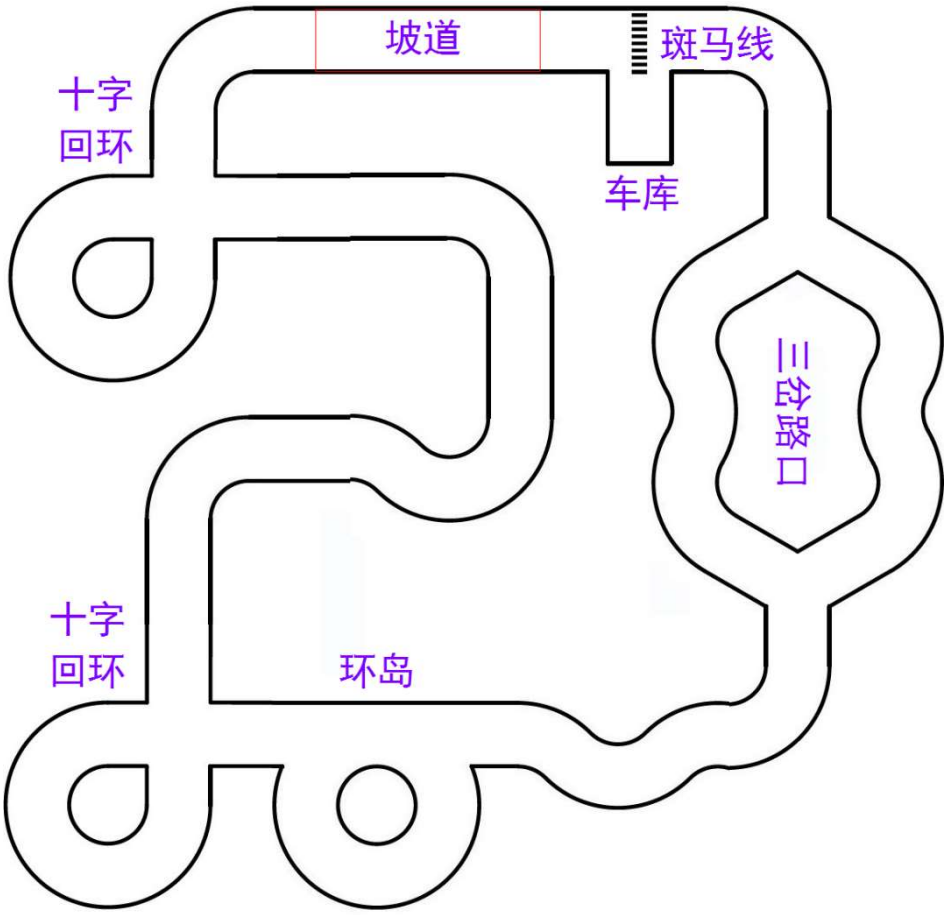


图 1.1 室内循环赛道示意图

比赛时，车模需要运行两周。车模需要在环形赛道上完成基础的赛道元素，具体如下。

（1） 直线赛道

这是赛道的基本形式。

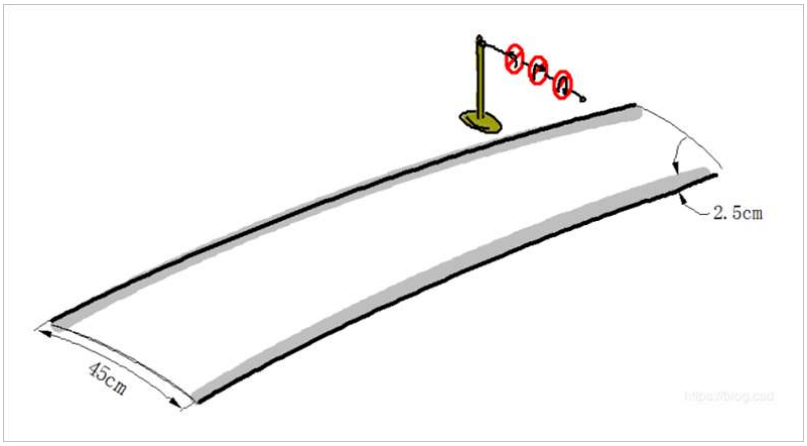


图 2.2 直线赛道示意图

（2） 曲线弯道

赛道中具有多段曲线弯道。这些弯道可以形成圆形环路，圆角拐弯，S 型赛道等。赛道中心线的曲率半径大于 50 厘米。

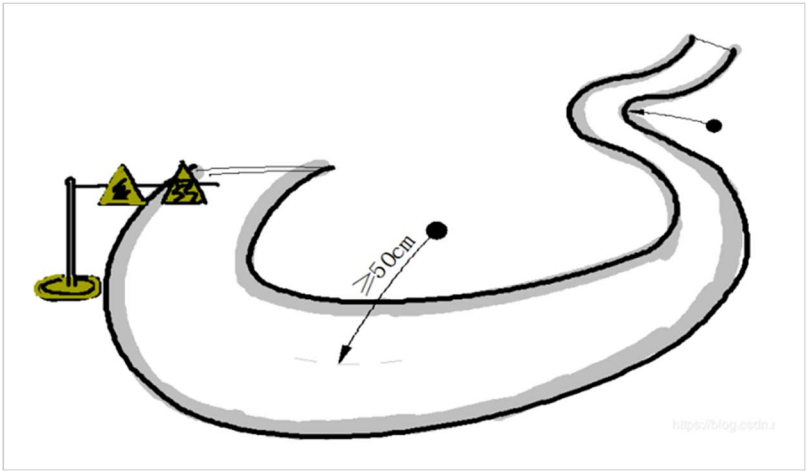


图 3.3 弯道赛道示意图

（3） 交叉路口

车辆通过十字交叉路口需要直行，不允许左转、右转。

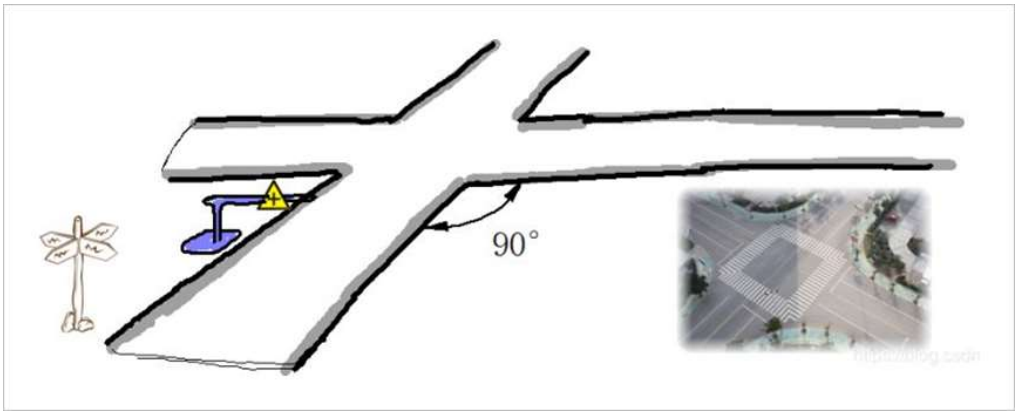


图 4.4 十字路口示意图

（4） 坡道

坡道的坡度不超过 20° 。坡道可以不是对称的。坡道的过渡弧长大于 10 厘米。坡道的长度、高度没有限制。一般情况下坡道的总长度会在 1.5 米左右。电磁组的导引线铺设在坡道的表面。

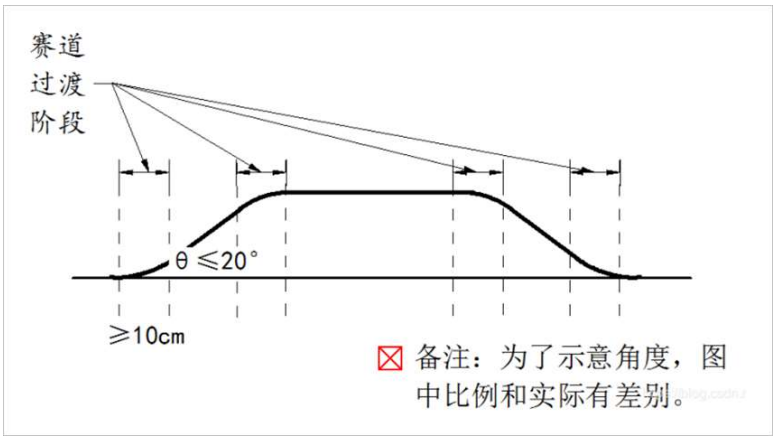


图 5.5 坡道示意图

（5） 环岛

赛车经过环岛时需进入环岛绕行一周后继续前行。环岛中心线半径不小于50厘米。电磁导线也是在环岛绕行一周。

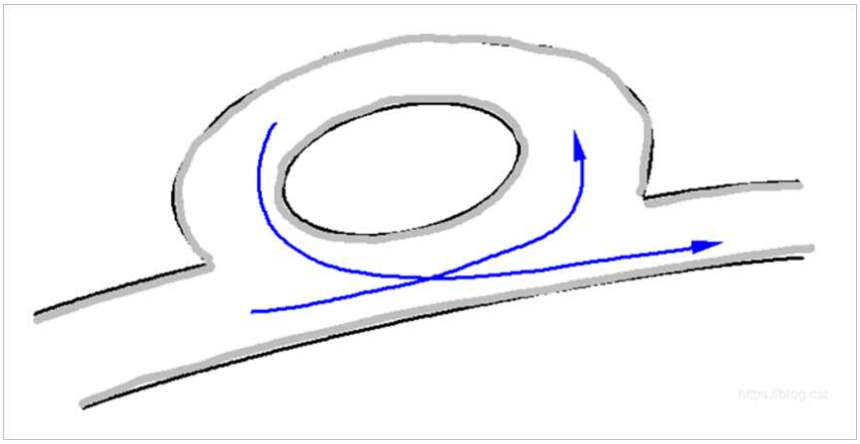


图 6.6 环岛示意图

（6） 三岔路口

在赛道上存在两个三岔路口，路口之间直线距离小于三米。三岔路口的三条进出口之间的夹角为 120° 。

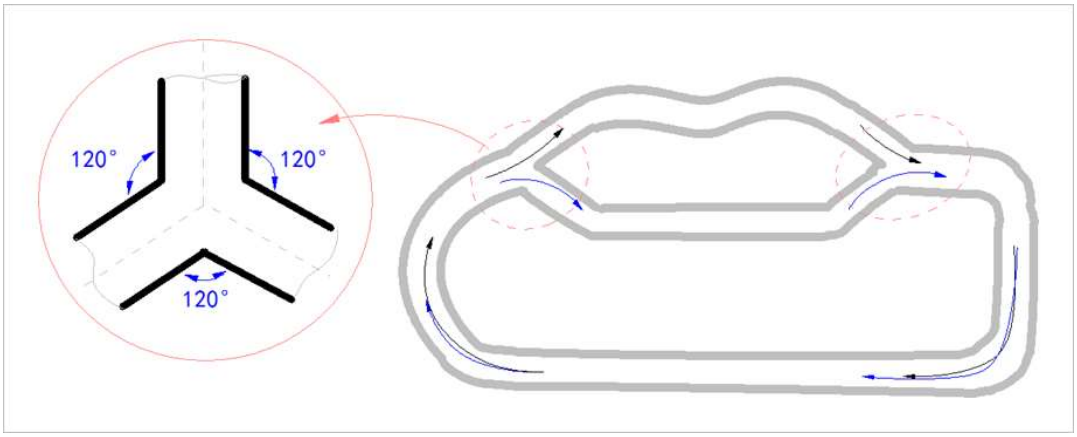


图 7.7 三岔路口示意图

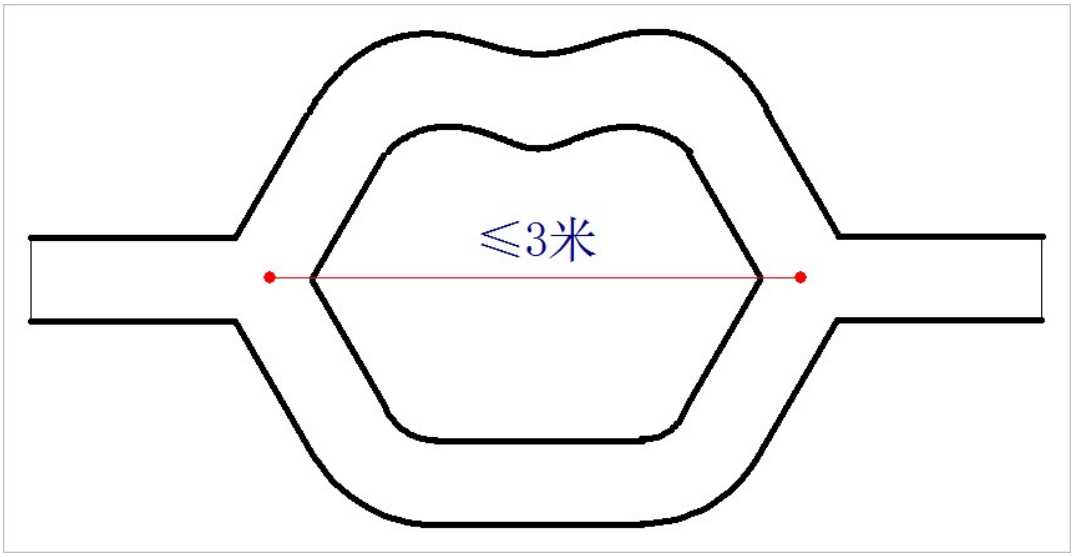
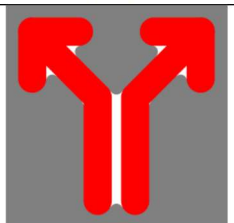
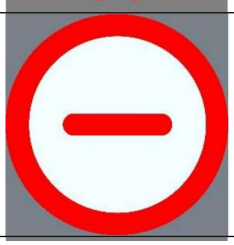


图 8.8 三岔路口示意图

在本次比赛当中，除了基础的赛道元素外，还要求完成特殊元素区域。特殊元素区域的前方指定的区域贴有固定的地面标志。标志的样式和含义如下表所示：

序号	名称	说明	图示
1	泛行区标志	表示前方三岔路口围成的泛行区域，内部区域包括蓝色底布均可行驶	
2	禁止通行标志	放置在泛行区域进出口连线上，车辆需要绕过此标志进行通行。（此标志底部为高密度海绵高出地面有 2cm，其他标志均紧贴赛道或地面）	

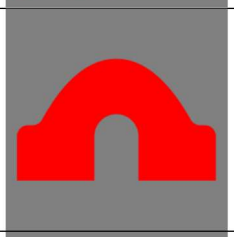
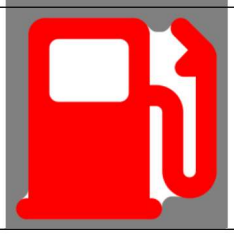
3	施工区标志	表示前方为施工区，需要绕行赛道外障碍桩围成的临时路段	
4	坡道标志	表示道路前方有坡道	
5	加油站标志	表示前方为加油站，车辆需要驶入加油站并按照指定的出口驶出加油站	
6	加油站出口数字标志	加油站设置有“1”和“2”两个出口，并在出口地面贴有对应的“1”和“2”数字标志。 比赛时加油站的入口处会随机放置车辆需要驶出时的出口数字。	

表 1.1 赛道标志

加油站两圈数字不相同！第一圈是数字 1，第二圈须换成数字 2。

标志的整体外框尺寸为 16cm*16cm 的正方形，标志颜色为红色，正方形内多余的灰色区域裁减掉，国赛采用可移除性不干胶制作，按照赛道任务和元素贴在赛道的指定位置（详见任务说明）。

比赛在基础赛道外设置有脱离基础赛道的任务区域，这些任务区域由可以移动的锥桶在基础赛道的边缘临时搭建而成。

锥桶由塑料材质制作而成，外表面为红色纯色，无任何标志。锥桶的底部直径 74mm，高度 74mm。在赛道搭建的过程中锥桶为可移动状态，车辆在运行过程中触碰到锥桶，导致锥桶偏离所在位置时，车模按冲出赛道处理。

图 9.9 锥桶示意图

第二章 智能车结构及硬件系统

2.1 硬件模块概述

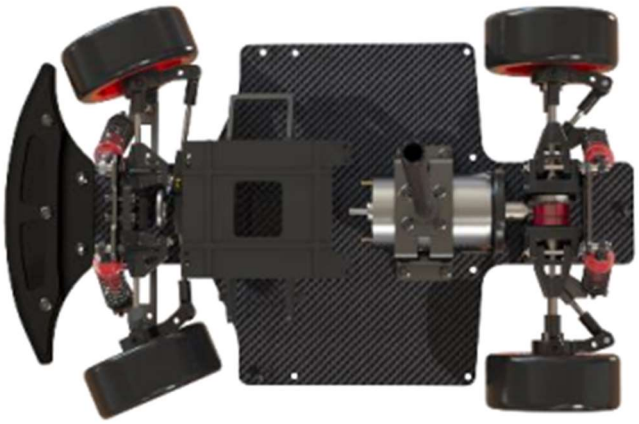


图 2.1 车模内部俯视图

图 2.2 I 模型车外观图

I 模型车是第十七届全国大学生智能汽车竞赛基础四轮比赛中完全模型组的指定车模，车体已安装电机、轮胎、轮毂、差速器、轴承、减震、拉杆、车壳、底盘等配件包。底盘采用高强度碳纤板，具有较高的强度。比赛时车模必须带有车壳，车壳必须完整的包裹车辆的本身，车身的底板外边缘、四个轮子和越过车身的传感器（摄像头）及其支撑件外，车辆的正视图、侧视图和俯视图看不到车辆的内部细节。选手需在车模的基础上增添其它各种模块，组装完成的车模要符合上述要求。

2.2 传感器

完全模型仅能使用 RGB 摄像头作为传感器，用于捕捉车道上的各种标识，

包括泛行区、泛行区内的禁止通行标志、施工区、坡道、加油站、加油站的数字标识。上位机基于其捕捉的图像分析小车前方中的各种路标，进而向下位机发送信息以做出不同的响应。

2.3 Edgeboard 计算上位机

在小车上承担“大脑”角色的是嵌入式计算平台 Edgeboard。百度面向嵌入式与边缘计算场景打造的嵌入式 AI 解决方案。携手芯片巨头，打造丰富的硬件选型，可适应多变的场景与边缘部署环境。无缝兼容百度大脑工具平台与算法模型，极大降低了项目部署门槛，轻松实现各种 AI 技能，适用于安防、工业、医疗、零售、教育、农业、交通等场景。

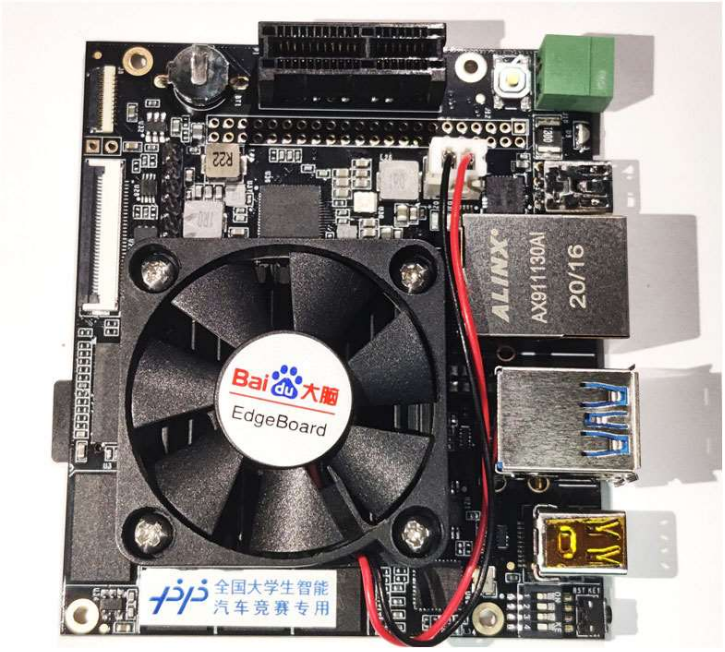


图 2.2 Edgeboard 计算平台

基于该平台，我们不仅可以完成小车寻迹赛道功能，还可以完成图像识别、特殊任务完成等功能来为小车智能驾驶提供更多方案。

2.4 Infineon 控制 MCU

完全模型下位机的主控板使用 Infineon 的 TC364 单片机，本组使用此单片机实现舵机的控制、与上位机的通信以及编码器信号的接收和与电机驱动板的通信。

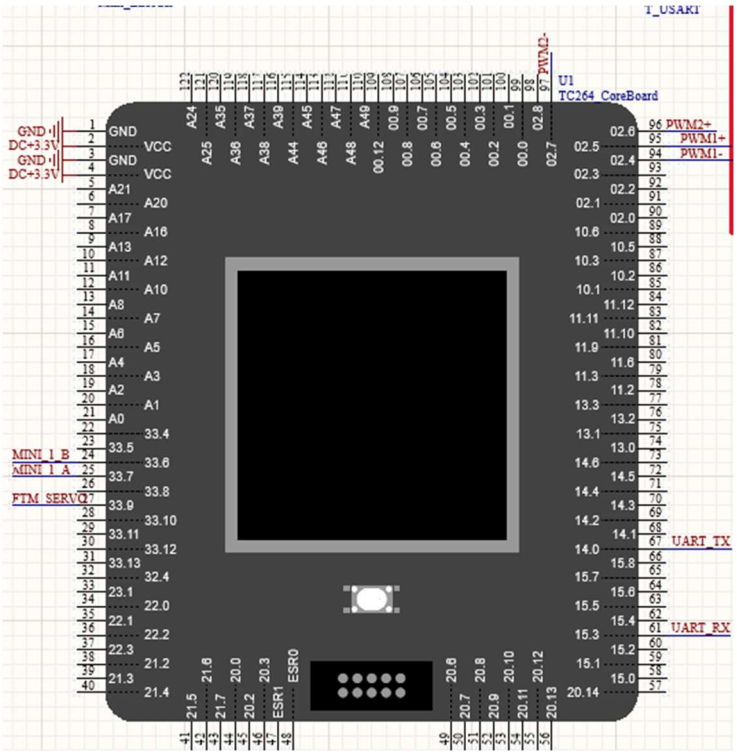


图 2.3 单片机使用的引脚

2.5 自制系统电路板

2.5.1 下位机

下位机电路主要包括电源输入、3.3V 稳压电路、5V 稳压电路以及电机驱动板，舵机的控制、和上位机之间的串口通信与编码器信号的接收。其中 3.3V 稳压电路通过一块 TPS76833 稳压元件来实现输出 3.3V 稳定电压，5V 稳压电路通过一块 LM1085 稳压元件来实现输出 5V 稳定电压。

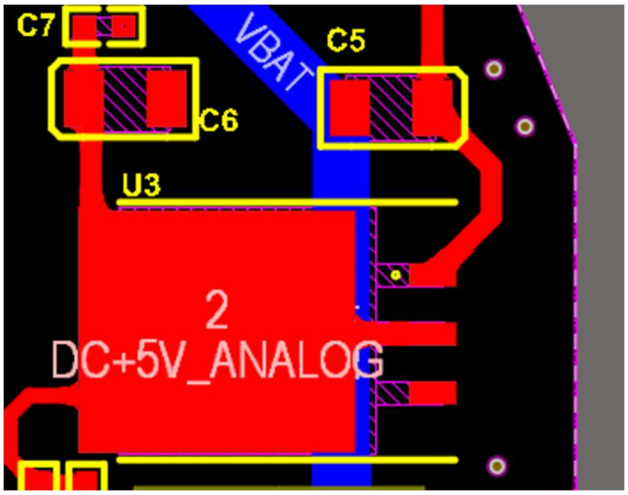


图 2.4 5V 稳压电路

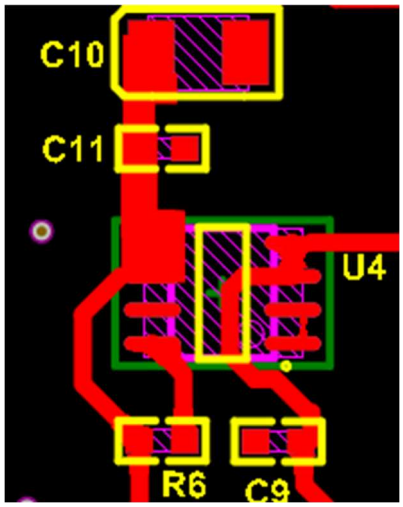


图 2.5 3.3V 稳压电路

整块 PCB 的 GND 分为模拟 GND 和数字 GND，从而将电源输入、5V 稳压等模拟电路部分与其余单片机 I/O 口等敏感器件分离开，两块地之间用两个并联的 0Ω 电阻互联。

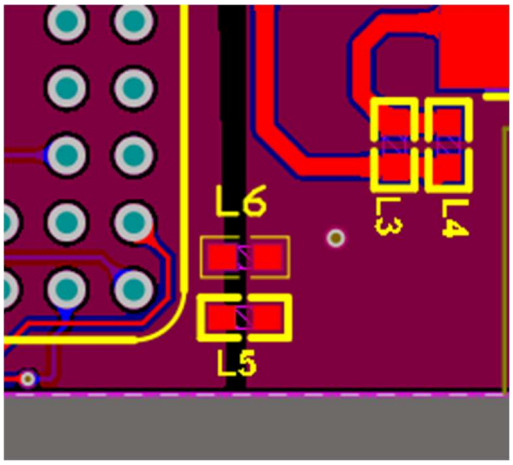


图 2.6 数字 GND 和模拟 GND

2.5.2 USB 转 TTL 模块

该模块使用 CH340N 芯片，与 USB 公头相接，辅以相应限流电阻与过流保护电路，输出接至一个 4P 排针，分别为 5V，GND，RX 与 TX，其中 5V 接口为输出电压，与下位机主板连接时，该接口不使用。

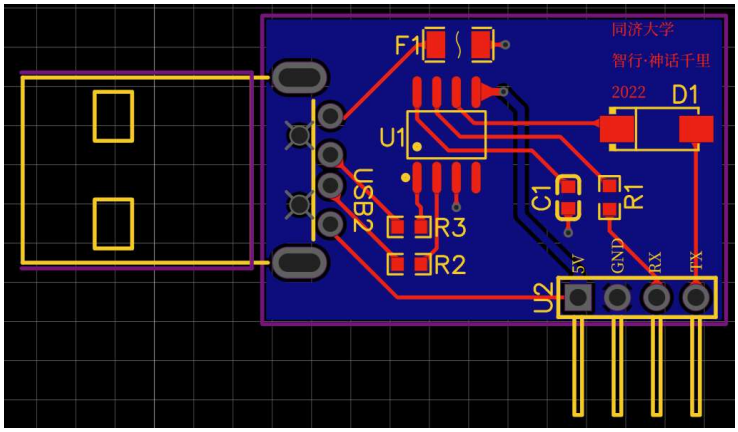


图 2.7 USB 转 TTL 模块

2.5.3 电机驱动板

核心主控电路由 DRV8701E 芯片构成，辅以缓冲电路，接至各个接口，并配上电源指示灯与使能指示灯（LED 灯）。

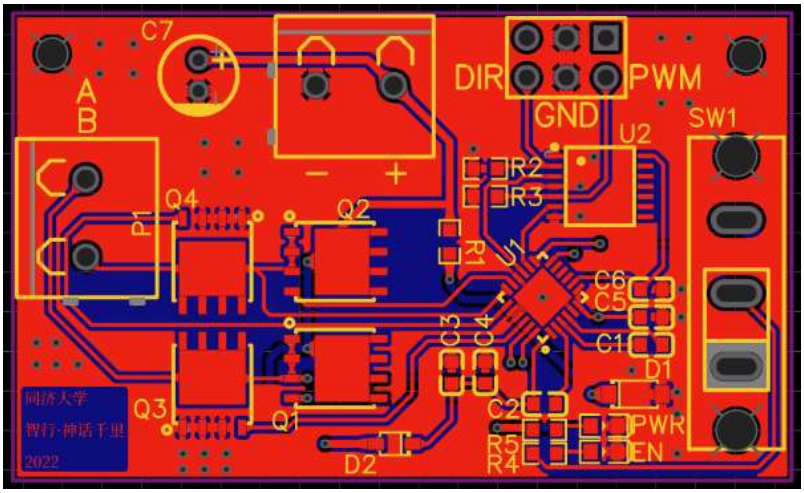


图 2.8 电机驱动板

2.6 电路系统设计

电路系统设计架构如下图所示：

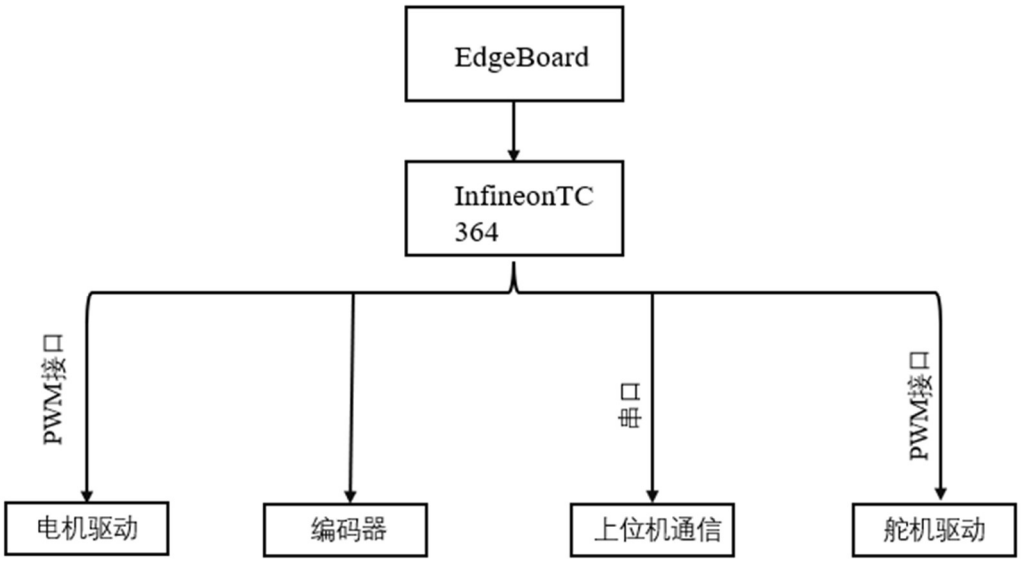


图 2.6 数字 GND 和模拟 GND

第三章 智能车微处理器控制系统

3.1 系统架构

微处理器使用英飞凌 TC364 芯片，开发中用到双核，其中，CPU0 负责通信，CPU1 负责输出电机与舵机的控制信号。系统架构图如下：

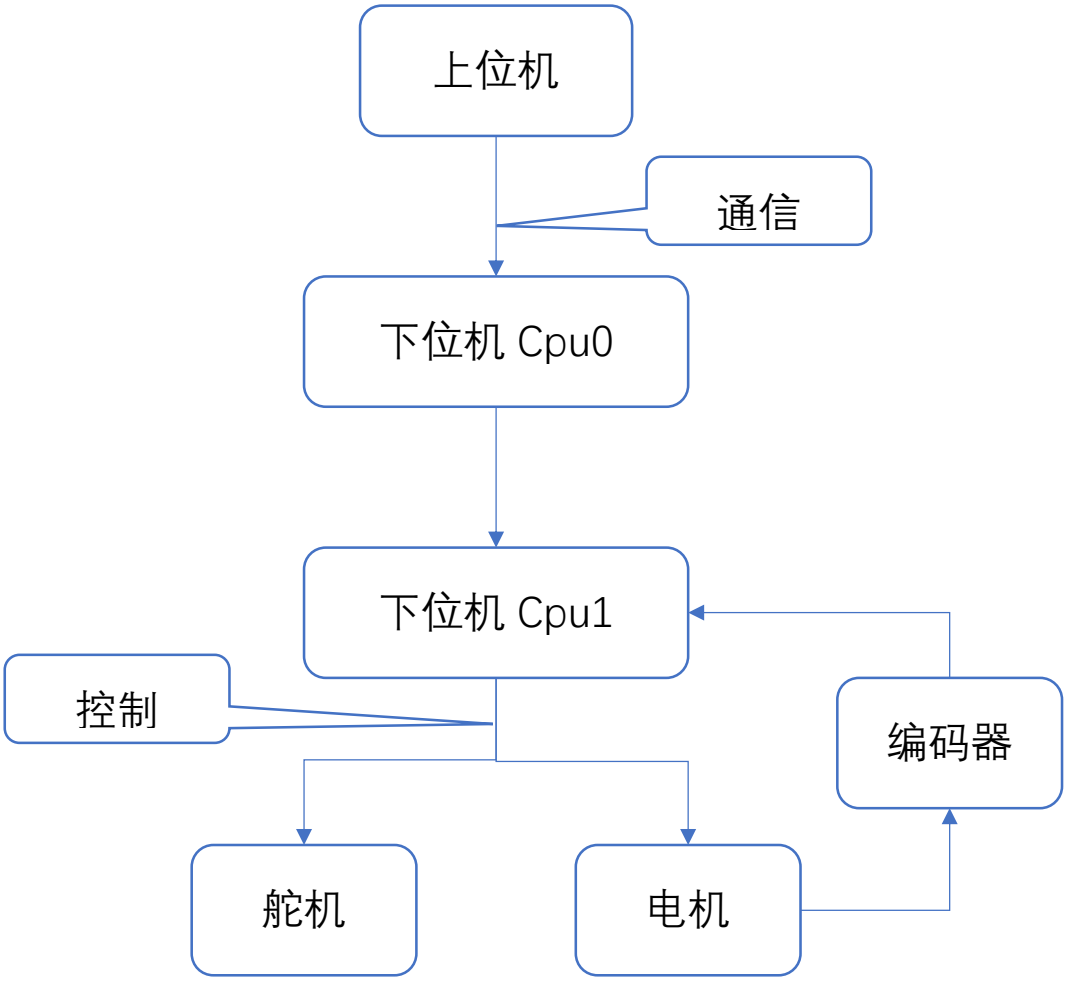


图 3.1 系统架构图

3.2 上下位机通信

上下位机通信采用 UART 通信协议，为变长帧通信，分为帧头，地址，帧长度，数据位和校验和，下面进行逐个叙述。

帧头规定为 0x42，接收到该数据表示通信开始。不同地址表示不同含义。0x01 表示接收上位机发来的电机速度与舵机方向信号，并进行后续控制。帧长度是为变长帧设计，以便计算校验和。校验和表示将先前接收的每一个字节相加

求和，以检验通信是否出错。

实际编程时，使用结构体来存储通信的状态。由于某些数据类型（例如 float 与 short 类型）不止一个字节，而通信时需要逐个字节收发，故采用 union 类型来存储这些数据。

部分代码如下：

```
typedef struct
{
    uint8 ReceiveStart;                          //数据接收开始
    uint8 ReceiveIndex;                          //接收序列
    uint8 ReceiveFinished;                       //数据队列接收并校验完成
    uint8 ReceiveBuff[USB_FRAME_LENGTH];         // USB 接收队列:临时接收
    uint8 ReceiveBuffFinished[USB_FRAME_LENGTH]; // USB 接收队列:校验成功
} USB_STRUCT;

typedef union
{
    unsigned char U8_Buff[4];
    float Float;
} UNION_BIT32;

typedef union
{
    unsigned char U16_Buff[2];
    short I16;
} UNION_BIT16;

USB_STRUCT usbStr;
void uart_receiveVerify(unsigned char data)
{
    if (data == USB_FRAME_HEAD && !usbStr.ReceiveStart) //监测帧头
    {
        usbStr.ReceiveStart = 1;
        usbStr.ReceiveBuff[0] = data;
        usbStr.ReceiveIndex = 1;
    }
    else if (usbStr.ReceiveStart && usbStr.ReceiveIndex < USB_FRAME_LENGTH) //通信开始|接收队列数据
    {
        usbStr.ReceiveBuff[usbStr.ReceiveIndex] = data;
        usbStr.ReceiveIndex++;
```

```
35      }
36
37      if (usbStr.ReceiveIndex >= USB_FRAME_LENGTH)
38      {
39          if (usbStr.ReceiveBuff[USB_FRAME_LENGTH - 1] == USB_FRAME
    _END) //帧尾
40          {
41              uint8 check = 0;
42              for (int i = 0; i < USB_FRAME_LENGTH - 2; i++)
43                  check += usbStr.ReceiveBuff[i]; //校验和
44
45              if (check == usbStr.ReceiveBuff[USB_FRAME_LENGTH - 2]
    ) //校验位
46              {
47                  memcpy(usbStr.ReceiveBuffFinished, usbStr.Receive
    Buff, USB_FRAME_LENGTH);
48                  usbStr.ReceiveFinished = 1;
49                  uint8 Addr = (uint8)(usbStr.ReceiveBuffFinished[1
    ] & 0xFF);
50                  uint8 BuffData[6] = {0};
51                  UNION_BIT32 UnionBit32;
52                  UNION_BIT16 UnionBit16;
53                  for (int i = 0; i < 6; i++)
54                  {
55                      BuffData[i] = usbStr.ReceiveBuffFinished[3 +
    i];
56                  }
57                  switch (Addr)
58                  {
59                  case 0x01: //智能车速度,智能车姿态（方向）
60                      UnionBit32.U8_Buff[0] = BuffData[0];
61                      UnionBit32.U8_Buff[1] = BuffData[1];
62                      UnionBit32.U8_Buff[2] = BuffData[2];
63                      UnionBit32.U8_Buff[3] = BuffData[3];
64                      speed = UnionBit32.Float; //电机速度
65
66                      UnionBit16.U16_Buff[0] = BuffData[4];
67                      UnionBit16.U16_Buff[1] = BuffData[5];
68                      angle = UnionBit16.I16; //舵机方向
69                      break;
70                  }
71              }
72          }
73          usbStr.ReceiveIndex = 0;
```

```
            usbStr.ReceiveStart = 0;
        }
    }

```

3.3 电机驱动

由于功率限制，单片机无法直接驱动电机，需要借助电机驱动板来进行控制。电机驱动使用 DRV8701E，功能为单电机双向可调速。驱动输入（信号接口）由单片机的两个引脚提供，其中一个表示速度，PWM 占空比越大，电机速度越快；另一个表示方向，低电平正转，高电平反转。

实际编程时，根据上位机发来的信号，速度为正表示电机正转，为负表示电机反转，将速度信号转换为 PWM 信号输出给电机驱动板。

3.4 舵机驱动

舵机驱动需要 3 个引脚，分别为 VCC, GND, 以及输入 PWM 信号的引脚，VCC 可以接入 5V 稳压电路的输出，PWM 引脚可以使用 TC364Demo 中的引脚 33.9。

舵机驱动需要精准，但也需要稳定。如果上位机发送了超出舵机角度限位范围之外的数据，舵机直接采用的话，由于无法转动至指定位置，长时间会导致舵机报废。因此在舵机驱动的相关单片机代码中，加入了保护部分，同时也舍弃了一部分过多的精准性，转而使用 short 类型的 angle 变量作为上下位机通信的桥梁。驱动部分的代码如下所示：

```
#define OFFSET 6 // negative as right, positive as left

/**
 * It takes an angle as an argument, and turns the servo motor to that angle
 *
 * @param angle the angle to turn the servo motor to, positive for left, negative for right.
 */
void turn_servo_motor(short angle)
{
    if (angle < -SERVO_MOTOR_MAX_ANGLE)
        angle = -SERVO_MOTOR_MAX_ANGLE;
    else if (angle > SERVO_MOTOR_MAX_ANGLE)
```

```
13.         angle = SERVO_MOTOR_MAX_ANGLE;
14.     uint32 pwm = OFFSET + 10000.0 * (1.5 + angle / (S_MOTOR_LIMIT
    / 2.0)) / 20.0;
15.     pwm_duty(S_MOTOR_PIN, pwm);
16. }
```

第四章 智能车上位机软件系统

4.1 系统架构

智能驾驶小车上的软件层面的实现集中在百度提供的 Edgeboard 板卡上，其实现了包括摄像头数据传输、构图、定位、导航、决策规划、目标检测、在内的诸多核心功能。其主要实现的功能如图 3.1 所示。为提高计算速度在实际应用中没有使用 IPM 俯视变换算法，使用原始图像处理就能够完成大部分内容。软件的系统架构请见 4.1 软件结构图，Edgeboard 中的文件目录结构请见 4.2 文件目录结构。

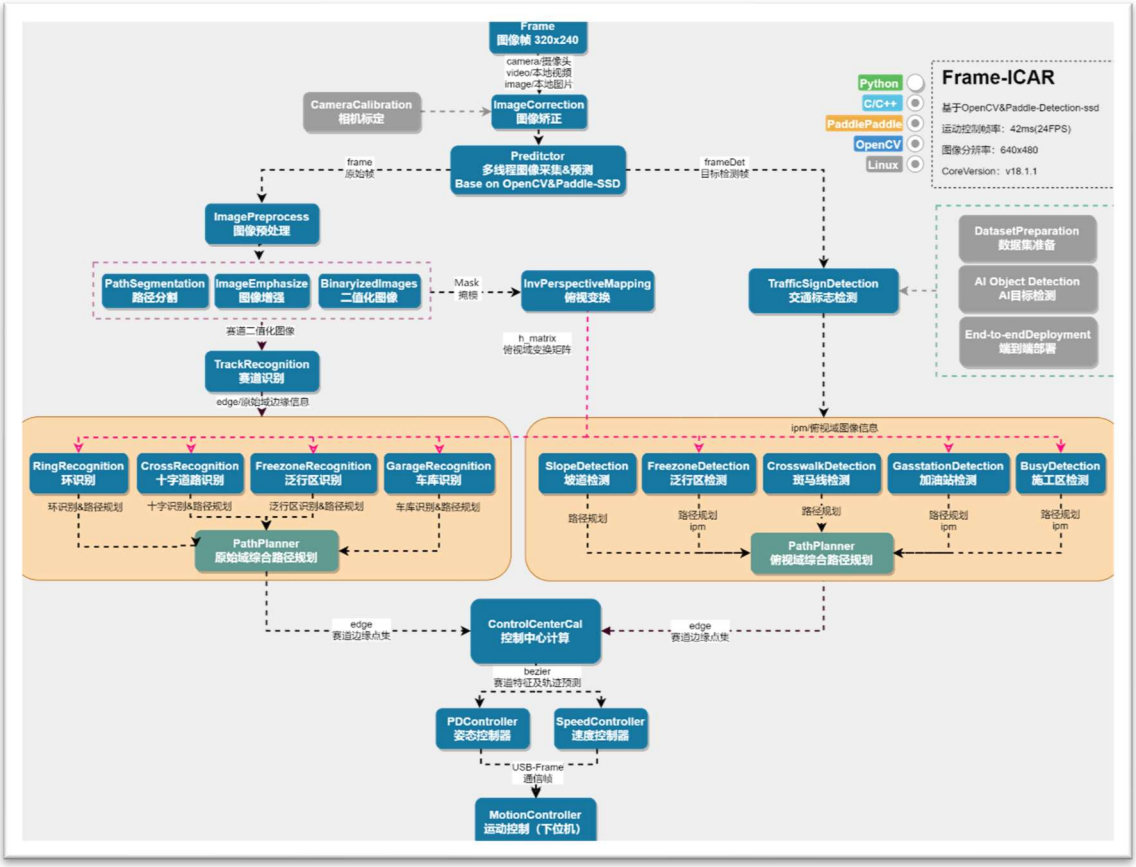


图 4.1 软件系统架构

摄像头的图像信息经过处理之后分别交给飞桨平台提供的模型预测接口库<paddle_api.h>以及 trackRecognition 边缘识别类进行进一步处理，然后 AI 模型预测结果和 trackRecognition 的边缘识别结果会一起送至各个特殊区域处理类，在各个处理类当中根据输入检测对应的特殊元素，如果检测到对应的特殊元素则会修改赛道边缘信息来完成重新更改路线的目的，最后 contolcenter 类根据边缘信息计算得到最后的偏差值。

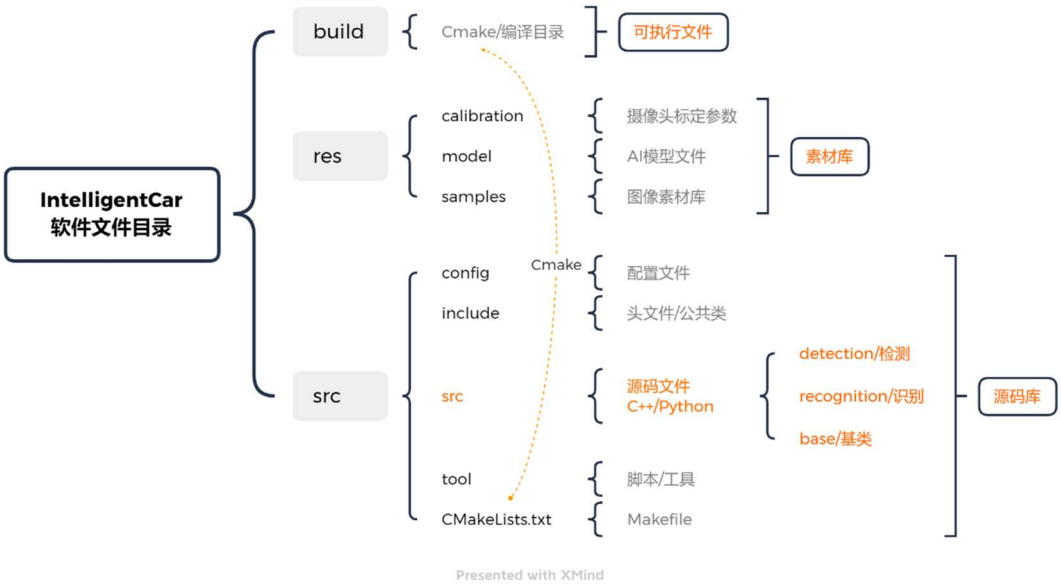


图 4.2 文件目录结构

Edgeboard 是一个“电脑”，其搭载的操作系统基于 Ubuntu 系统，百度面向嵌入式与边缘计算场景打造的嵌入式 AI 解决方案。携手芯片巨头，打造丰富的硬件选型，可适应多变的场景与边缘部署环境。无缝兼容百度大脑工具平台与算法模型，极大降低了项目部署门槛，轻松实现各种 AI 技能，适用于安防、工业、医疗、零售、教育、农业、交通等场景。熟悉 windows 系统的读者，可能对这个操作系统感到极度陌生。你可能会有疑问，为什么这个"电脑“不使用 windows 系统，原因首先是 linux 系统基于 UNIX 开发，稳定且效率高；其次，其系统源码全部开源，任何人都可以自由使用；最后，也是最重要的是 linux 系统可裁剪，最小只要不到 1M 的系统就可以完整驱动整个“电脑”，对于存储容量较小的嵌入式平台来说是刚需。

4. 2 图像处理与 OpenCV

送至 trackRecognition 类的图像信息应用的图像处理主要使用的是 opencv 基于 c++语言环境下的<opencv2>库进行计算和处理。OpenCV（Open Source Computer Vision Library：http://opencv.org）是一个开源的基于 BSD 许可的库，它包括数百种计算机视觉算法。文档 OpenCV 2.x API 描述的是 C++ API，相对还有一个基于 C 语言的 OpenCV 1.x API，后者的描述在文档 opencv1.x.pdf 中。OpenCV 具有模块化结构，这就意味着开发包里面包含多个共享库或者静态库。下面是可使用的模块：核心功能（Core functionality） - 一个紧凑的模块，定义了基本的数据结构，包括密集的多维 Mat 数组和被其他模块使用的基本功能。图

像处理（Image processing） - 一个图像处理模块，它包括线性和非线性图像滤波，几何图形转化（重置大小，放射和透视变形，通用基本表格重置映射），色彩空间转换，直方图等。影像分析（video） - 一个影像分析模块，它包括动作判断，背景弱化和目标跟踪算法。3D 校准（calib3d） - 基于多视图的几何算法，平面和立体摄像机校准，对象姿势判断，立体匹配算法，和 3D 元素的重建。平面特征（features2d）—突出的特征判断，特征描述和对特征描述的对比。对象侦查（objdetect）—目标和预定义类别实例化的侦查（例如：脸、眼睛、杯子、人、汽车等等）。highgui—一个容易使用的用户功能界面。视频输入输出（videoio）—一个容易使用的视频采集和视频解码器。在本次程序当中主要使用了图像裁剪和二值化功能。

4.3 赛道识别与特殊区域处理

4.3.1TrackRecognition 赛道识别类

赛道识别的基本类是 TrackRecognition 类，他的作用是从二值化图像下端（图像的顶端 10 行和底端十行会被舍弃）逐行扫描，保存黑白交接的边界点（图像边缘也是边界点），然后跟上一行的边界点进行匹配找到最接近续上一行的一组边界作为本行的边界，其中不符合宽度限制的边界会被舍弃。也就是说一张图像最多会有 240-20=220 组边界点，每组边界点有左右两个边界点坐标。

4.3.2RingRecognition 环岛识别类

左环岛和右环岛的逻辑和处理方式是对称的，下面我们按照右侧进行举例说明我们的策略是分成三个部分，识别环岛，进入环岛和出环岛。

识别环岛是在环岛的后半段进行识别，识别逻辑为左侧的线的整体斜率和非丢线部分的斜率方差进行阈值限制，右边缘部分由三部分组成，丢线段，凸起上侧和突起下侧，这三段中的点的个数必须满足一定要求。除此之外，右边缘的最左点的位置需要满足要求，并且突起部分的被切割成 4 段，斜率方差和整体斜率也需要满足我们设定的要求。（整体斜率：即第一个点至最后一个点的斜率，斜率方差：每一个点和临近的 8 个点进行斜率计算，最后求出所有的点的局部斜率进行方差计算）

补线策略，修改左边缘，连接右边的上端突变点和左侧的 1/4 处点。

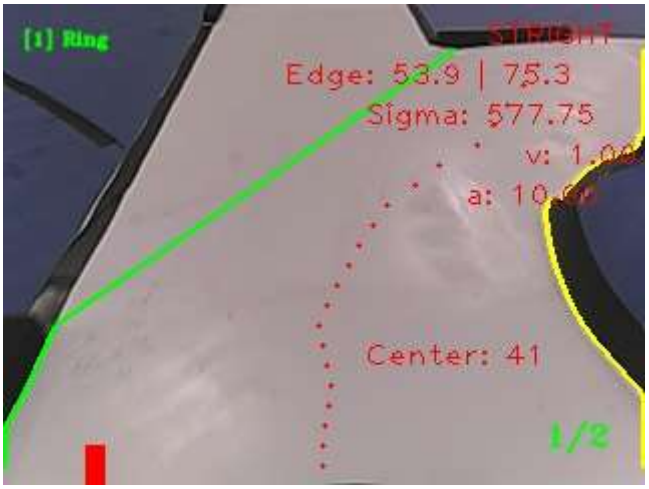


图 4.3 进入环岛

进入环岛中不进行特殊的补线处理，在环岛中的检测条件为，左端的线保持持续向右靠的趋势如下图所示

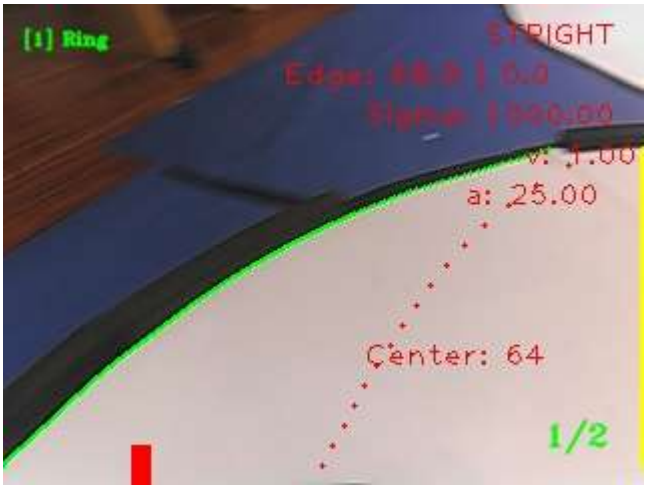


图 4.4 环岛中间

出环岛策略，一旦检测到了退出转弯，则将左侧的突变点和存在边缘的最后一行的中间位置进行连接，如下图所示，当左侧的线恢复直线（满足整体斜率和斜率方差的要求）退出环岛。

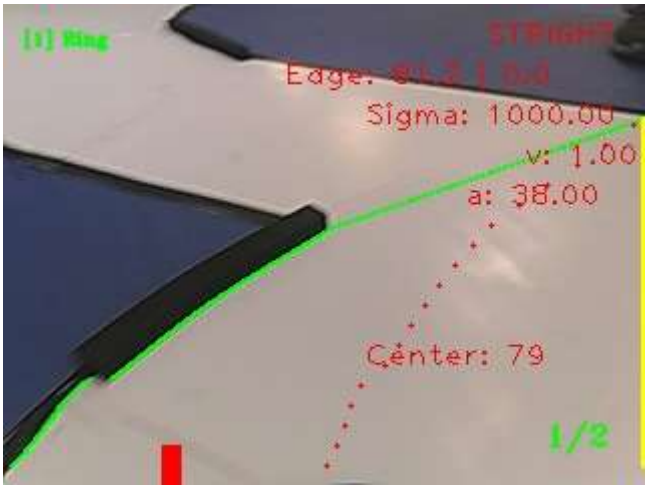


图 4.5 离开环岛

4.3.3FreezoneRecognition 泛行区识别类

泛行区整体分为两个工作模式，经过泛行区中间绕开标志和走两侧道路。如果检测到了分叉路口标志那么启用泛行区功能。

走两侧部分逻辑：识别到交叉路口标志进入泛行区，以从左侧进入泛型区为例，首先找出图片右下突变点和中间岔路点，路过没有叉路点则不补线，如果没有右下突变点则将图片右下角作为右下突变点。连接两点作为新的道路两侧，出岔路口和这个逻辑一致。

中间绕开禁行标志策略：在识别到交叉路口之后按照图片底端直线进行补线进入，如果底端没有直线那么直接直行，如果中间遇到进行标志，测算标志中心距离图像的距离，如果靠左调用右侧绕行子程序，执行绕行功能，如果重新回到赛道那么退出泛行区功能。

4.3.4CrossRecognition 交叉路口识别类

交叉路口的识别顺序位于所有识别类的最后面，当其他类完成识别之后才能启动交叉路口的补线功能，交叉路口的补线很简单，只需要检测图像的左上，左下，右上，右下四个突变点。左右两边分别补线，如果上下侧突变点都存在，那么连接这两个点，如果只存在上侧突变点那么延长上册的直线，只存在下侧突变点同理。

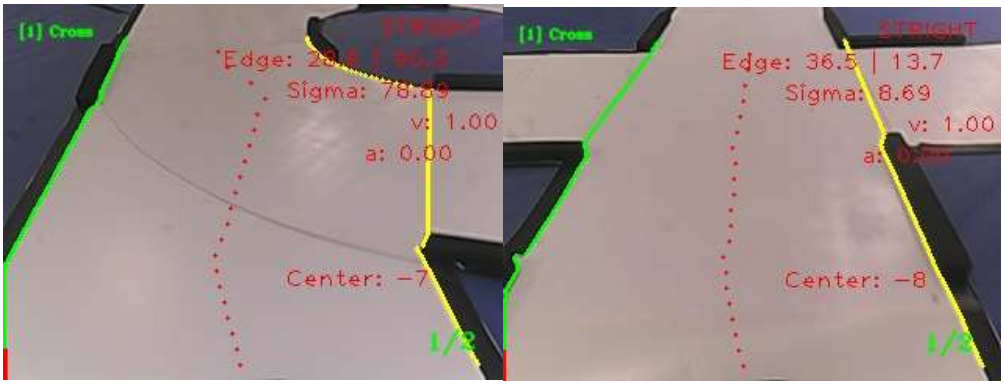


图 4.6 交叉路口补线

4.3.5GarageRecognition 车库与斑马线识别类

车库分为如车库和出车库两个部分。出车库部分首先让车子直接离开一小段距离然后开启补线功能，将右下角和最靠近图像中间的点进行连接即可，当右侧的直线方差小于阈值时即停止出库程序。

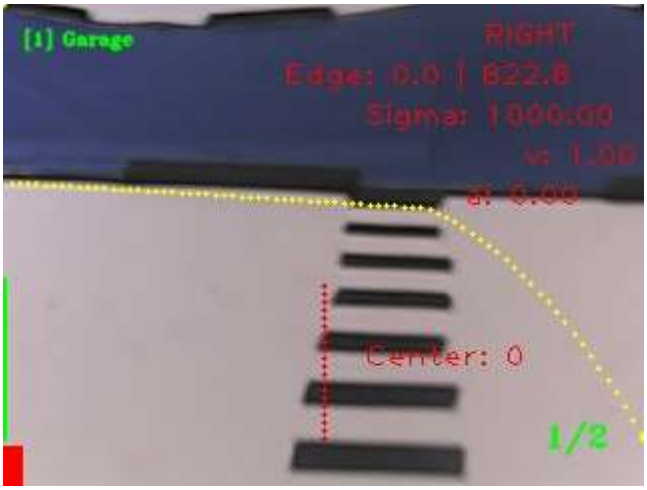


图 4.7 出库

当我们底二次看见斑马线（存在连续的几行由宽度差不多的白色色块）时调用入库程序，将左上突变点和右侧边缘进行连接即可，如果检测到边缘点个数较少那么停止入库

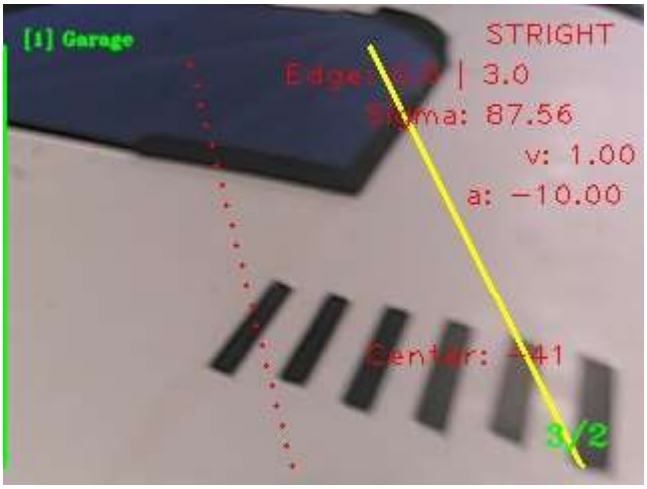


图 4.8 入库

4.3.6SlopeDetection 坡道检测类

坡道的处理比较简单，只需在需要坡道时设计图像上方的点保持直行即可。

4.4 模型预测与 PaddleLite

我们首先利用车载摄像头移动小车采集赛道信息，将赛道上的信息进行标注之后利用百度飞将平台结合实例代码进训练得到模型文件 model 和 param，将这两个文件存放在响应的文件夹中。预测使用了官方的示例代码，通过 Predictor 类运行图像识别程序，获取 predictResult 预测结果即可。

Paddle-Lite 框架是 PaddleMobile 新一代架构，重点支持移动端推理预测，特点高性能、多硬件、轻量级 。支持 PaddleFluid/TensorFlow/Caffe/ONNX 模型的推理部署，目前已经支持 ARM CPU, Mali GPU, Adreno GPU, Huawei NPU 等多种硬件，正在逐步增加 X86 CPU, Nvidia GPU 等多款硬件，相关硬件性能业内领先。

4.5 控制中心计算与 PID

控制中心主要用于计算最后的赛道中心偏差值 cotrolCenter 使用，我们对所有的所有的边缘点进行处理，如果存在锐角线段那么切割所有锐角以上的路段，如果存在大量丢线(边缘点在图像的两侧)，那么启用左右转弯模式计算偏差值。对于非转弯路段，我们按照等步长采用 30 组边缘点，计算边缘点的中心坐标进行加全平均，最后和图像中心水平坐标值差得到偏差，如果是转弯路段那么我们对丢线的另外一侧进行加减屏幕的一半得到这一组的中心点坐标，而不是上文中的两侧边缘取中间。

本次比赛我们采用经典控制算法 PID 控制，如图 4.9 中所示，根据系统输入与预定输出的偏差的大小运用比例、积分、微分计算出一个控制量，将这个控制量输入系统，获得输出量，通过反馈回路再次检测该输出量的偏差，循环上述过程，以使输出达到预定值。

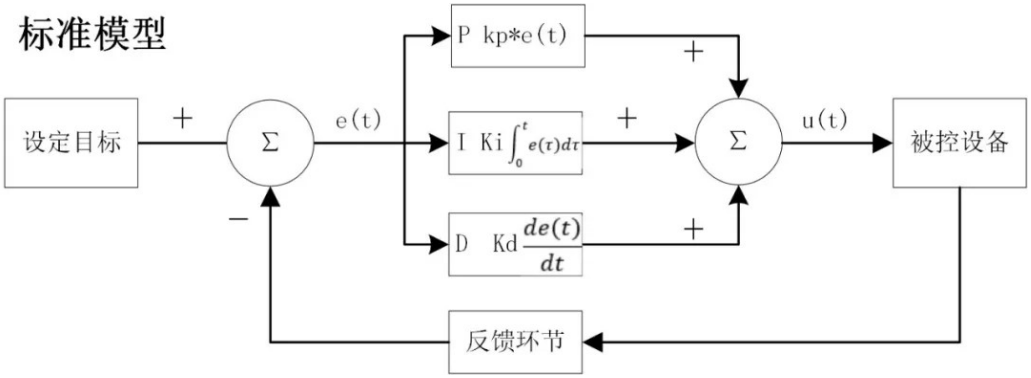


图 4.9 使用 PID 控制器的经典负反馈控制算法标准模型

位置式 PID 算法可由公式（2）表达：

$$\begin{aligned} \boldsymbol{u(k)} &= \boldsymbol{K_p}\left\{\boldsymbol{e(k)} + \frac{\boldsymbol{T}}{\boldsymbol{T_i}}\sum_{j=1}^{k}\boldsymbol{e(j)} + \frac{\boldsymbol{T_d}}{\boldsymbol{T}}[\boldsymbol{e(k)} - \boldsymbol{e(k-1)}]\right\} \\ &= \boldsymbol{K_p e(k)} + \boldsymbol{K_i}\sum_{j=1}^{k}\boldsymbol{e(j)} + \boldsymbol{K_d}[\boldsymbol{e(k)} - \boldsymbol{e(k-1)}] \end{aligned} \tag{2}$$

在 PID 算法中，比例环节 P 的作用是成比例地反映控制系统的偏差信号 e(t)，一旦产生偏差，比例控制环节立即产生控制作用以减小偏差。积分环节 I 的作用是消除静差，提高系统的无差度，在使用积分控制时，常常对积分项进行限幅以减少积分饱和的影响。微分环节 D 的作用是反映偏差信号的变化趋势，能够在偏差信号变化之前先引入一个有效的早期修正信号来提前修正偏差，加快系统的动作速度，减少调节时间。

位置式 PID 特点主要是一方面控制的输出与整个过去的状态有关，用到了误差的累加值（即用误差累加代替积分），另一方面公式输出直接对应对象的输出，对系统影响大。

第五章 开发调试过程说明

5.1 舵机机械调中

舵机调中的步骤大致如下：

1. 将舵机转角调整至中点，即调用函数：

```
pwm_duty(S_MOTOR_PIN, 10000.0 * 1.5 / 20.0);
```

2. 将舵机安装至车模前桥，固定相关螺丝，使车轮偏转大致为 0；
3. 烧写下面的程序，调节宏定义 OFFSET 的值，观察模型车远距离行驶的横向偏移，反复调节 OFFSET 的值使之最小。

如前所述，舵机的控制是通过调节 Infineon 单片机对应引脚的占空比实现的，为了保证程序的可读性和稳定性，我们将舵机控制函数单独封装，代码如下：

```
17.#define OFFSET 6 // negative as right, positive as left
18.
19./**
20. * It takes an angle as an argument, and turns the servo motor to
    that angle
21. *
22. * @param angle the angle to turn the servo motor to, positive fo
   r left, negative for right.
23. */
24.void turn_servo_motor(short angle)
25.{
26.    if (angle < -SERVO_MOTOR_MAX_ANGLE)
27.        angle = -SERVO_MOTOR_MAX_ANGLE;
28.    else if (angle > SERVO_MOTOR_MAX_ANGLE)
29.        angle = SERVO_MOTOR_MAX_ANGLE;
30.    uint32 pwm = OFFSET + 10000.0 * (1.5 + angle / (S_MOTOR_LIMIT
    / 2.0)) / 20.0;
31.    pwm_duty(S_MOTOR_PIN, pwm);
32.}
```

5.2 上位机图像显示

通过使用 VNC 软件，可视化访问 Edgeboard，将程序开启调试模式，得到各

赛道元素识别图像如下图所示。

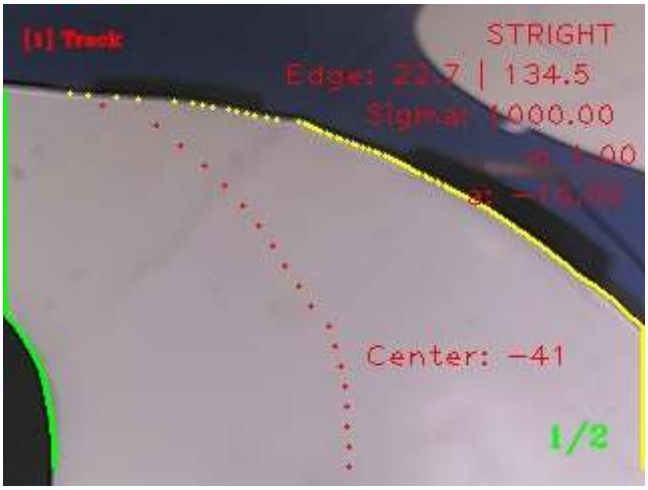
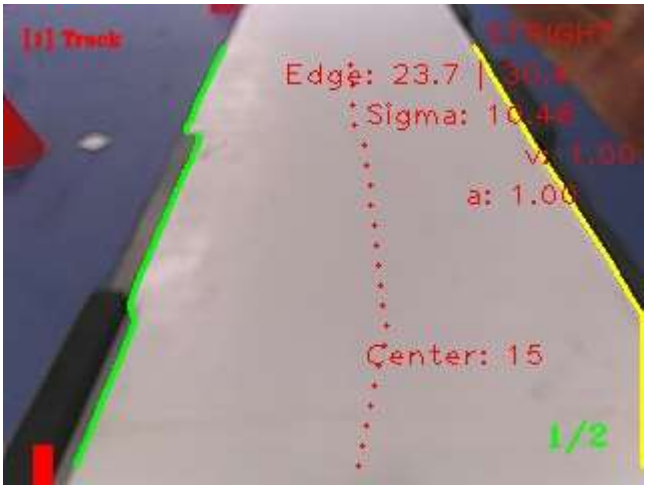
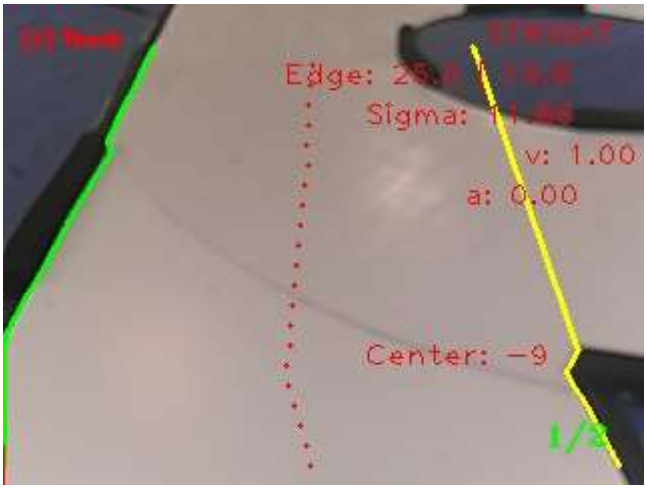


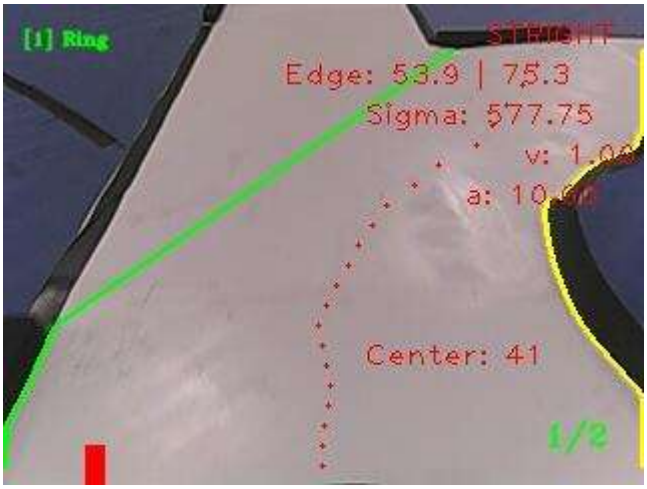
图 5.1 弯道识别图片



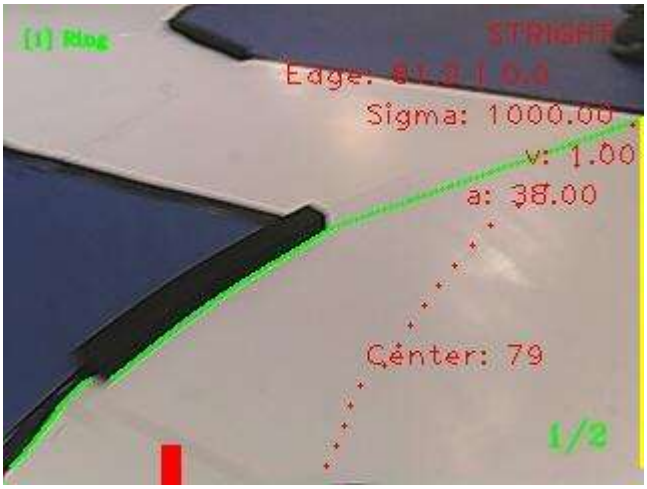
5.2 直道识别图片



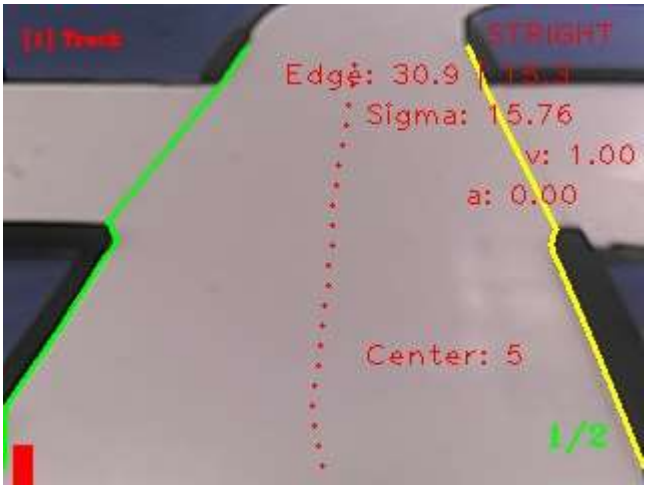
5.3 环岛识别图片一



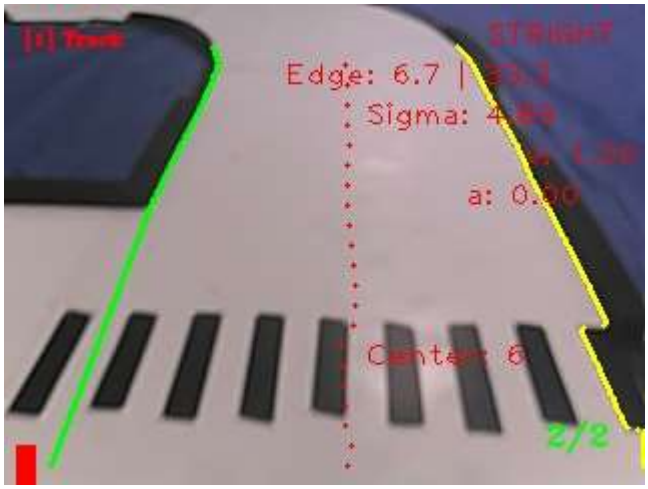
5.4 环岛识别图片二



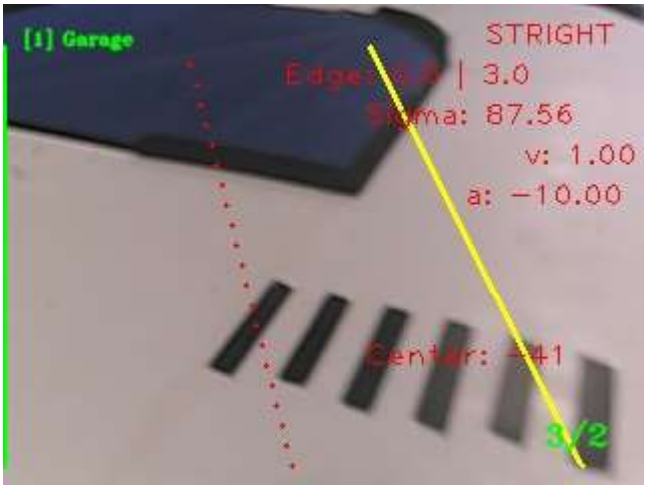
5.5 环岛识别图片三



5.6 十字路口识别图片



5.7 车库识别图片一



5.8 车库识别图片二

5.3 上下位机通信

如前所述，上下位机通信采用串口方式。为了检验串口方式是否成功，编写了代码如下：

```
#include "headfile.h"
#include "math.h"
#pragma section all "cpu1_dsram"

/* hardware parameters */
#define S_MOTOR_PIN ATOM1_CH1_P33_9
#define MOTOR_DIR P02_6
#define MOTOR_PWM ATOM0_CH7_P02_7
#define S_MOTOR_LIMIT 180        // degree
#define SERVO_MOTOR_MAX_ANGLE 40 // degree
#define OFFSET 6 // negative as right, positive as left
```

```
12.
13./* data sent by uart */
14.extern volatile float speed;
15.extern volatile short angle;
16.
17./**
18. * It takes an angle as an argument, and turns the servo motor to
    that angle
19. *
20. * @param angle the angle to turn the servo motor to, positive fo
   r left, negative for right.
21. */
22.void turn_servo_motor(short angle)
23.{
24.    if (angle < -SERVO_MOTOR_MAX_ANGLE)
25.        angle = -SERVO_MOTOR_MAX_ANGLE;
26.    else if (angle > SERVO_MOTOR_MAX_ANGLE)
27.        angle = SERVO_MOTOR_MAX_ANGLE;
28.    uint32 pwm = OFFSET + 10000.0 * (1.5 + angle / (S_MOTOR_LIMIT
    / 2.0)) / 20.0;
29.    pwm_duty(S_MOTOR_PIN, pwm);
30.}
31.
32./**
33. * It set the speed of the motor.
34. *
35. * @param speed the speed of the motor, in RPM
36. */
37.void set_motor_speed(float speed)
38.{
39.    int32 speed_power = SPEED2POWER(speed); // SPEED2POWER conver
   t formula
40.    if (0 <= speed_power)
41.    {
42.        pwm_duty(MOTOR_PWM, speed_power);
43.        gpio_set(MOTOR_DIR, 0);
44.    }
45.    else
46.    {
47.        pwm_duty(MOTOR_PWM, -speed_power);
48.        gpio_set(MOTOR_DIR, 1);
49.    }
50.}
51.void core1_main(void)
```

```
{
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword
());
    enableInterrupts();

    // init servo motor
    uint32 duty = 1.5 * 10000 / 20;
    gtm_pwm_init(S_MOTOR_PIN, 50, duty);

    // init motor
    gtm_pwm_init(MOTOR_PWM, 17000, 0);
    gpio_init(MOTOR_DIR, GPO, 0, PUSHPULL);

    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 0xFFFF);
    enableInterrupts();
    while (1)
    {
        /* A loop to turn the servo motor,
         * set the motor speed,
         * and delay for sample interval.
         */
        turn_servo_motor(-angle);
        set_motor_speed(speed);
        systick_delay_ms(STM0, SAMPLING_INTERVAL);
    }
}
#pragma section all restore
```

这个代码使用 extern 声明了 cpu0 中定义的变量 angle 和 speed，作为舵机电机驱动的数据。在 main 函数中，每隔 10ms 将 angle 和 speed 变量作为舵机和电机的控制信号直接输出，使得下位机通过串口获取角度和速度的值，并显示在车模的行为中。

具体的调试方法是：将自制的 USB 转 TTL 芯片插入电脑的接口，在电脑的串口通信助手应用中直接发送按照规定的协议编码好的二进制数据，观测车模上舵机的转动角度和电机的转速水平。

5.4 上位机参数调整

为了减少编译时间，我们调用了 json 库，将一些适时调整的参数（例如 PD 控制的系数，模型车运行的速度，赛道元素的使能等）放入 motion.json 文件中，

以供程序读入数据，每一次只需要重新运行二进制文件即可。

5.5 PID 控制调节

PID 的参数调节需要综合考虑模型车的机械参数（比如由于机械限位导致舵机存在最大转角），赛道的弯曲程度，以及摄像头的帧率等指标。由于在模型车的行进过程中，会存在不可消除的累计误差，因此我们取消了积分控制算法，而单纯使用 PD 控制（具体做法是在配置文件中将这一项的系数修改为零）。由于赛道同时存在较长的直道和较急的弯道，对于传统的线性比例控制而言，往往会出现：对于直道，KP 系数过大，导致震荡现象；对于弯道，KP 系数过小，导致车辆反应不及时，冲出赛道。因此我们在此基础上加以改进，使用了二阶 P 控制，数学公式如下：

$$u(n) = K_d[e(n) - e(n-1)] + K_{p2}e^2(n) + K_{p1}e(n)$$

模型车的舵机转角最大值为 40 度，控制中心计算结果的范围大致在-90 到+90，对原点和(90,40)两个点定标，绘制 P 控制的曲线如图所示：

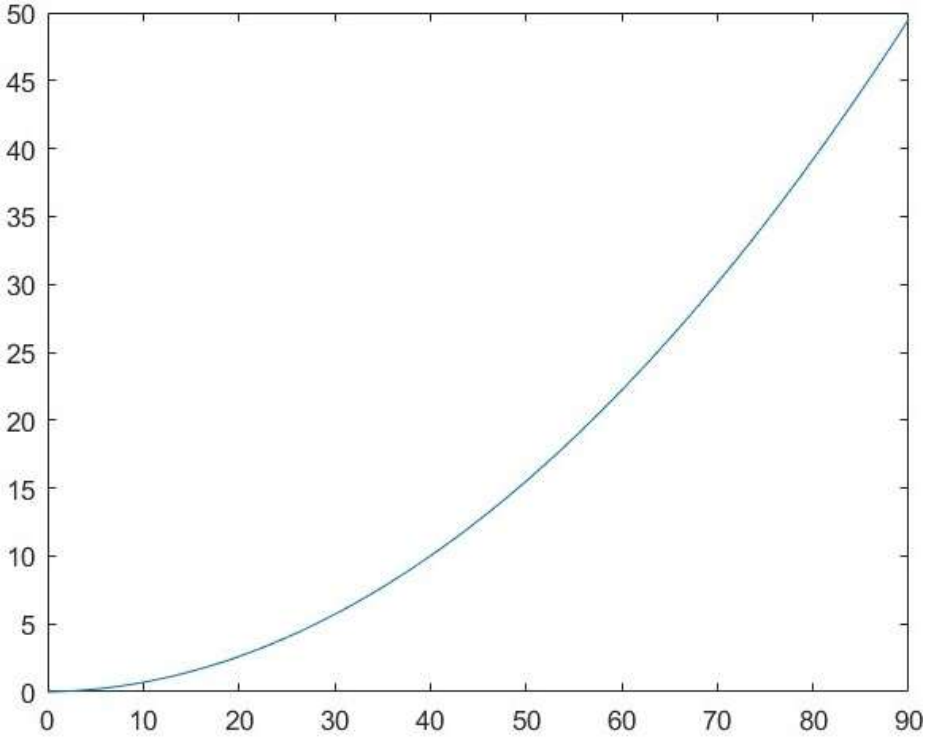


图 5.9 二阶 P 控制

当 controlCenter 的值小于 30（直道）时，舵机的转角不超过 5 度，当 controlCenter 的值大于 80（环岛、急弯等）后，舵机将达到最大转角，实现对于不同地形的分别控制。

在调节好 P 控制以后，通过微调K_d，消除震荡，完成 PID 的参数调节。

第六章 车模主要参数

车模类型：I 型车模

车模整体尺寸

4. 长：310mm
5. 宽：200mm
6. 高：370mm
7. 摄像头镜头中心距离地面高度：310mm
8. 传感器：RGB 摄像头 1 个
9. 微处理器：Edgeboard 1 个、英飞凌 TC364 1 个、编码器 1 个
10. 电池：CB_22003 全国大学生智能车竞赛专用高性能锂电池 1 个
11. 2200mAh,11.1V,3S1P,25C,24.42Wh
12. 电路板：Edgeboard、下位机电路板、电机驱动板、USB 转 TTL 电路板

第七章 项目创新点

7.1 采用 RGB 图像二值化识别赛道边缘

由于我们使用的是官方 RGB 摄像头，但是 RGB 摄像头拍摄的数据较大，使用 RGB 三通道识别赛道不满足竞速组速度较快的要求，因此我们将图像二值化，只保留黑白两个元素，赛道识别为白色，蓝底布识别为黑色，黑白边界即为赛道，使用绿色和黄色点标记在图像中。然后根据图像两个边界的赛道点方差、丢线情况识别左右转弯和转弯幅度，计算出 controlCenter。

7.2 采用 mobilenet-ssd 模型对标志进行识别

我们使用 AI studio 平台，通过小车的行驶拍摄数据集的 python 程序得到一系列各种赛道标志的图片，作为我们训练模型的数据集。然后通过多次调整参数和多次更换模型，得到识别准确率最高的模型和训练参数。我们使用的是 mobilenet-ssd 模型来训练图片，因为虽然 Yolo 模型训练识别的准确率更高，但是 Yolo 模型的训练和识别时间比较长，不适合在竞速组速度较快情况下的模式识别，且模型大小过大占用存储空间。而 mobilenet-ssd 模型虽然识别准确率相对较低，但是其中某一个准确率高于 0.5，显著高于其他标志的准确率，可以区分不同的标志，且 mobilenet-ssd 模型识别速度快、模型轻量化。mobilenet-ssd 模型在性能与质量方面得到了很好的权衡，使其能在精准完成任务的同时，不因运算单元算力的问题发生卡顿。

7.3 采用边线修正的方式执行特殊元素任务

当识别到特殊赛道元素后，转入对应元素的特定执行类，其中修改原来的边线。泛行区执行直线补线然后绕行，施工区和加油站执行对左或右侧前端第一个锥桶补线，进入后识别两侧锥桶并将其作为边线识别。最终车辆按照新的边线重新计算、执行。

第八章 项目总结

寻迹、识别和特殊任务是这项比赛中最为关键的三大功能展示，也与我们当今智能化产业的需求息息相关。“无人驾驶”是目前处在风口浪尖的一个话题，比赛的内容其实是“无人驾驶”技术的基础概念。

8.1 团队开展过程

我们团队为了能完成完全模型组的各项任务，学习 Ubuntu 和 PaddleLite，从蹒跚学步到一步步到熟练掌握。其中，AI studio 与 AURIX 是两个强大的工具，让我们团队能较为轻松和准确地进行模拟仿真和可视化处理。其中 AI studio 集开放数据、开源算法、免费算力三位一体,为开发者提供高效学习和开发环境、高价值高奖金竞赛项目,支撑高校老师轻松实现 AI 教学,并助力开发者学习交流,加速落地 AI 业务场景。PaddleLite 是飞桨基于 Paddle Mobile 全新升级推出的端侧推理引擎，在多硬件、多平台以及硬件混合调度的支持上更加完备，我们使用的是在 Edgeboard 上内置的 PaddleLite。

从今年 1 月份完全模型组确定被新增在智能车竞赛竞速组中，到 3 月份正式工具发布、拿到车模开始工作，到 4、5 月份上海疫情严重我们的工作许多被迫暂停、还有的不得不转到线上，到 5 月底 6 月初上海高校大学生大规模离校返乡，对在疫情中心态较差但是还在不断调试车模的我们在身体和心理上都还是有很大的影响，到 7 月份同学们坚持在学校、在实验室因疫情未准开放的情况下在宿舍楼大厅铺赛道调试，到 8 月份提前来到南京、为正式比赛前全体成员在线下调试比赛做好充足的准备。在 2022 年上半年上海疫情严重、全城封城、各项事项停摆或者转至线上三个多月的特殊经历中，我们成长了许多、也在不断进步。随着车模行驶不断稳定、一个个任务不断完成，我们提升了许多，最终得以在华东分赛中晋级，来到全国总决赛，我们的心情非常激动，也希望能在全国总决赛中能够取得较好的名次，加油！

8.2 结语

本报告主要介绍了我们的技术方案：使用 SASU 官方摄像头获取周围环境，从而得到车身姿态和赛道环境信息；使用 PaddleLite 训练 mobilenet-ssd 模型识别赛道特殊元素、准确率较高；执行特殊任务时从一般巡线类转到特殊任务执行类，完成特殊任务；通过最终补线计算控制中心送入 PID 得到速度和角度，最后上位机将数据传输至下位机，下位机完成对车轮和舵机的控制。

参考文献：

[1] 邵贝贝. 嵌入式实时操作系统[LC/OS-Ⅱ(第2版)[M].北京.清华大学出版社．2004
[2] 邵贝贝. 单片机嵌入式应用的在线开发方法[M].北京. 清华大学出版社．2004
[3] 王晓明. 电动机的单片机控制[M]．北京. 北京航空航天大学出版社．2002 [4] 张琦. 移动机器人的路径规划与定位技术研究[D]. 哈尔滨: 哈尔滨工业大学, 2015.
[4] 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
[5] 沈长生.常用电子元器件使用一读通[M].北京. 人民邮电出版社, 2004
[6] EdgeBoard嵌入式AI解决方案[EB/OL] https://ai.baidu.com/ai-doc/HWCE/Fkuqounlk

附录 部分程序源代码

```
#include <unistd.h>
#include <iostream>
#include <signal.h>
#include <opencv2/highgui.hpp>                    //OpenCV 终端部署
#include <opencv2/opencv.hpp>                     //OpenCV 终端部署
#include<opencv2/imgproc/imgproc_c.h>
#include "../include/uart.hpp"                    //串口通信驱动
#include "../include/detection.hpp"               //百度 Paddle 框架移动端部署
#include "../include/common.hpp"                  //公共类方法文件
#include "image_preprocess.cpp"                   //图像预处理类
#include "recognition/track_recognition.cpp"      //赛道识别基础类
#include "controlcenter_cal.cpp"                  //控制中心计算类
#include "motion_controller.cpp"                  //智能车运动控制类
#include "recognition/cross_recognition.cpp"      //十字道路识别与路径规划类
#include "recognition/garage_recognition.cpp"     //车库及斑马线识别类
#include "recognition/freezone_recognition.cpp" //泛行区识别类
#include "detection/busy_detectionnew.cpp"        //施工区 AI 检测与路径规划类
#include "detection/slope_detection.cpp"          //坡道 AI 检测与路径规划类
#include "recognition/ring_recognition.cpp"       /* 环岛识别类 */
using namespace std;
using namespace cv;

void callbackSignal(int signum);
void displayWindowDetailInit(void);
std::shared_ptr<Driver> driver = nullptr; //初始化串口驱动

enum RoadType
```

```
{
    BaseHandle = 0,    //基础赛道处理
    RingHandle,        //环岛赛道处理
    CrossHandle,       //十字道路处理
    FreezoneHandle,    //泛行区处理
    GarageHandle,      //车库处理
    GasstationHandle, //加油站处理
    BusyareaHandle,    //施工区处理
    SlopeHandle        //坡道处理
};

int main(int argc, char const *argv[])
{
    std::shared_ptr<Detection> detection = nullptr; //初始化AI 预测模型
    ImagePreprocess imagePreprocess;                //图像预处理类
    TrackRecognition trackRecognition;              //赛道识别
    ControlCenterCal controlCenterCal;              //车辆行驶路径拟合类
    MotionController motionController;              //运动控制
    CrossroadRecognition crossroadRecognition;      //十字道路处理
    GarageRecognition garageRecognition;            //车库识别
    FreezoneRecognition freezoneRecognition;        //泛型区识别类
    BusyareaDetection busyareaDetection;            //施工区检测
    SlopeDetection slopeDetection;                  //坡道（桥）检测类
    RingRecognition ringRecognition;                /* 环岛检测类 */
    uint16_t counterRunBegin = 1;                   //智能车启动计数器：等待摄像头图像帧稳定
    RoadType roadType = RoadType::BaseHandle;       //初始化赛道类型
    uint16_t counterOutTrackA = 0;                  //车辆冲出赛道计数器A
    uint16_t counterOutTrackB = 0;                  //车辆冲出赛道计数器B
```

```
56.     uint16_t circlesThis = 1;                                   //智能车
    当前运行的圈数
57.     uint16_t countercircles = 0;                                //圈数计
    数器
58.
59.
60.     // USB 转串口的设备名为 /dev/ttyUSB0
61.     driver = std::make_shared<Driver>("/dev/ttyUSB0", BaudR
    ate::BAUD_115200);
62.     if (driver == nullptr)
63.     {
64.         std::cout << "Create Uart-
    Driver Error!" << std::endl;
65.         return -1;
66.     }
67.     //串口初始化，打开串口设备及配置串口数据格式
68.     int ret = driver->open();
69.     if (ret != 0)
70.     {
71.         std::cout << "Uart Open failed!" << std::endl;
72.         return -1;
73.     }
74.
75.     ipm.init(Size(COLSIMAGE, ROWSIMAGE), Size(COLSIMAGEIPM,
     ROWSIMAGEIPM)); // IPM 逆透视变换初始化
76.
77.     signal(SIGINT, callbackSignal); //程序退出信号
78.
79.     motionController.loadParams(); //读取配置文件
80.     trackRecognition.rowCutUp = motionController.params.row
    CutUp;
81.     trackRecognition.rowCutBottom = motionController.params
    .rowCutBottom;
82.     garageRecognition.disGarageEntry = motionController.par
    ams.disGarageEntry;
83.     garageRecognition.GarageEnterLeft=motionController.para
    ms.GarageEnterLeft;
84.     zhtfreezoneRecognition.dirLeft=motionController.params.
    ZHTFreezoneLeft;
85.     if (motionController.params.GarageEnable) //出入库使能
86.         roadType = RoadType::GarageHandle;     //初始赛道元素
    为出库
87.
88.     if (motionController.params.debug) //调试模式
```

```
89.     {
90.         displayWindowDetailInit();                                  //显示窗口初始化
91.         detection = Detection::DetectionInstance("/dev/video0", "../res/model/mobilenet-ssd-v1"); // 视频输入源：本地视频 | AI 模型文件
92.         printAiEnable = true;                                       // 开启AI 检测结果图像绘制：耗时
93.     }
94.     else //比赛模式
95.     {
96.         cout << "等待发车!!!" << endl;
97.         detection = Detection::DetectionInstance("/dev/video0", "../res/model/mobilenet-ssd-v1"); // 视频输入源：本地视频 | AI 模型文件
98.         printAiEnable = false;                                      // 关闭AI 检测结果图像绘制：节省算力
99.
100.         // while (!driver->receiveStartSignal()) //串口接收下位机-比赛开始信号
101.         // {
102.         //     ;
103.         // }
104.         cout << "Garage Exit" << endl;
105.         for (int i = 0; i < 30; i++) // 3 秒后发车
106.         {
107.             driver->carControl(0, PWMSERVOMID); //智能车停止运动|建立下位机通信
108.             waitKey(100);
109.         }
110.         cout << "--------- System start!!! -------" << endl;
111.     }
112.
113.     while (1)
114.     {
115.         bool imshowRec = false; //特殊赛道图像显示标志
116.
117.         // 处理帧时长监测：显示单帧时长
118.         if (motionController.params.debug)
119.         {
```

```
120.             static auto preTime = chrono::duration_cast<c
   hrono::milliseconds>(chrono::system_clock::now().time_since
   _epoch()).count();
121.             auto startTime = chrono::duration_cast<chrono
   ::milliseconds>(chrono::system_clock::now().time_since_epoc
   h()).count();
122.             cout << "run frame time : " << startTime - pr
   eTime << "ms" << endl;
123.             preTime = startTime;
124.         }
125.
126.         //[01] 视频源选择
127.         cout << "[01]" << endl;
128.         std::shared_ptr<DetectionResult> resultAI = detec
   tion->getLastFrame(); //获取 Paddle 多线程模型预测数据
129.         Mat frame = resultAI->rgb_frame;
                          //获取原始摄像头图像
130.         if (motionController.params.debug)
131.         {
132.             //imshow("frame", resultAI->det_render_frame)
   ;
133.             //savePicture(resultAI->det_render_frame); //
   保存 AI 识别图像
134.         }
135.         /*
136.         else if (motionController.params.saveImage) //保
   存原始图像
137.                 savePicture(resultAI->det_render_frame);
138.         */
139.
140.         //[02] 图像预处理
141.         cout << "[02]" << endl;
142.         //Mat imgaeCorrect = imagePreprocess.imageCorrect
   ion(frame);         // 相机矫正
143.         Mat imgaeCorrect = frame;
144.         Mat imageBinary = imagePreprocess.imageBinaryzati
   on(imgaeCorrect); // Gray
145.
146.         //[03] 基础赛道识别：万物基于 Track！！！！！
147.         cout << "[03]" << endl;
148.         trackRecognition.trackRecognition(imageBinary); /
   /赛道线识别
149.         if (motionController.params.debug)
150.         {
```

```
151.                 Mat imageTrack = imgaeCorrect.clone();  // RG
     B
152.                 trackRecognition.drawImage(imageTrack); //图
     像显示赛道线识别结果
153.                 imshow("imageTrack", imageTrack);
154.                 savePicture(imageTrack);
155.             }
156.             //
157.             else if (motionController.params.saveImage)  //保
     存原始图像
158.             {
159.                 Mat imageTrack = imgaeCorrect.clone();  // RG
     B
160.                 trackRecognition.drawImage(imageTrack); //图
     像显示赛道线识别结果
161.                 savePicture(imageTrack);
162.             }
163.
164.             // [04] 出库和入库识别与路径规划
165.             cout << "[04]" << endl;
166.             if (motionController.params.GarageEnable) //赛道
     元素是否使能
167.             {
168.                 if (roadType == RoadType::GarageHandle || roa
     dType == RoadType::BaseHandle)
169.                 {
170.                     countercircles++; //圈数计数
171.                     if (countercircles > 500)   /* 200 */
172.                         countercircles = 500;
173.
174.                     if (garageRecognition.startingCheck(track
     Recognition/*resultAI->predictor_results*/)) /* 检测到起点
     （斑马线）: 修改为二值化识别 */
175.                     {
176.                         busyareaDetection.reset();   //施工区
     检测初始化
177.                         freezoneRecognition.reset(); //泛行区
     识别复位
178.
179.                         if (countercircles > 300)       //大
     于60 认为是有效走了一圈?
180.                         {
181.                             circlesThis++;
182.                             countercircles = 0;
```

```
183.                         }
184.
185.                         /* 保存认为是斑马线的图像 */
186.                         if (motionController.params.saveImage
    ) //保存原始图像
187.                         {
188.                             Mat imageGarage = Mat::zeros(Size
    (COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
189.                             putText(imageGarage, "crosswork",
     Point(COLSIMAGE / 2 - 5, 20), cv::FONT_HERSHEY_TRIPLEX, 0.
    3, cv::Scalar(0, 0, 255), 1, CV_AA);
190.                             savePicture(imageGarage);
191.                         }
192.                     }
193.
194.                     if (circlesThis == motionController.param
    s.circles + 1) //入库使能：跑完N圈
195.                         garageRecognition.entryEnable = true;
196.                     else if (circlesThis > motionController.p
    arams.circles)
197.                     {
198.                         cout << "circles > " << motionControl
    ler.params.circles << endl;
199.                         callbackSignal(0);
200.                     }
201.
202.                     if (garageRecognition.garageRecognition(t
    rackRecognition, motionController.params.GarageEnterDelayTi
    me))
203.                     {
204.                         roadType = RoadType::GarageHandle;
205.                         if (garageRecognition.garageStep == g
    arageRecognition.GarageEntryFinish) //入库完成
206.                         {
207.                             cout << ">>>>>>>   入库结
    束 !!!!!" << endl;
208.                             callbackSignal(0);
209.                         }
210.                         if (motionController.params.debug)
211.                         {
212.                             Mat imageGarage = Mat::zeros(Size
    (COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
213.                             garageRecognition.drawImage(track
    Recognition, imageGarage);
```

```
214.                            imshow("imageRecognition", imageG
    arage);
215.                            imshowRec = true;
216.                            savePicture(imageGarage);
217.                        }
218.                        else if (motionController.params.save
    Image)  //保存原始图像
219.                        {
220.                            Mat imageGarage = Mat::zeros(Size
    (COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
221.                            garageRecognition.drawImage(track
    Recognition, imageGarage);
222.                            savePicture(imageGarage);
223.                        }
224.                    }
225.                    else
226.                        roadType = RoadType::BaseHandle;
227.                }
228.            }
229.
230.            //[05] 施工区检测
231.            cout << "[05]" << endl;
232.            if (motionController.params.BusyAreaEnable) //赛
    道元素是否使能
233.            {
234.                if (roadType == RoadType::BusyareaHandle || r
    oadType == RoadType::BaseHandle)
235.                {
236.                    if (busyareaDetection.busyareaDetection(t
    rackRecognition, resultAI->predictor_results))
237.                    {
238.                        if (motionController.params.debug)
239.                        {
240.                            Mat imageBusyarea = Mat::zeros(Si
    ze(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
241.                            busyareaDetection.drawImage(track
    Recognition, imageBusyarea);
242.                            imshow("imageRecognition", imageB
    usyarea);
243.                            imshowRec = true;
244.                            savePicture(imageBusyarea);
245.
246.                            // 显示俯视域的赛道图像
247.                            // Mat imageIpm;
```

```
248.                             // ipm.homography(imageBusyarea,
    imageIpm); //图像的逆透视变换
249.                             // imshow("imageIpm", imageIpm);
250.                             // savePicture(imageIpm);
251.                         }
252.                         else if (motionController.params.save
    Image)  //保存原始图像
253.                         {
254.                             Mat imageBusyarea = Mat::zeros(Si
    ze(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
255.                             busyareaDetection.drawImage(track
    Recognition, imageBusyarea);
256.                             savePicture(imageBusyarea);
257.                         }
258.                         roadType = RoadType::BusyareaHandle;
259.                     }
260.                     else
261.                         roadType = RoadType::BaseHandle;
262.                 }
263.             }
264.
265.             // [06] 坡道（桥）检测与路径规划
266.             cout << "[06]" << endl;
267.             if (motionController.params.SlopEnable) //赛道元素
    是否使能
268.             {
269.                 if (roadType == RoadType::SlopeHandle || road
    Type == RoadType::BaseHandle)
270.                 {
271.                     if (slopeDetection.slopeDetection(trackRe
    cognition, resultAI->predictor_results))
272.                     {
273.                         roadType = RoadType::SlopeHandle;
274.                         if (motionController.params.debug)
275.                         {
276.                             Mat imageFreezone = Mat::zeros(Si
    ze(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
277.                             slopeDetection.drawImage(trackRec
    ognition, imageFreezone);
278.                             imshow("imageRecognition", imageF
    reezone);
279.                             imshowRec = true;
280.                             savePicture(imageFreezone);
281.                         }
```

```
282.                     else if (motionController.params.save
    Image)  //保存原始图像
283.                     {
284.                         Mat imageFreezone = Mat::zeros(Si
    ze(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
285.                         slopeDetection.drawImage(trackRec
    ognition, imageFreezone);
286.                         savePicture(imageFreezone);
287.                     }
288.                 }
289.                 else
290.                     roadType = RoadType::BaseHandle;
291.             }
292.         }
293.
294.         /* [07] 泛行区检测与识别：是AI 方式 */
295.         cout << "[07]" << endl;
296.         if(motionController.params.FreezoneEnable){
297.         if (roadType == RoadType::FreezoneHandle || roadT
    ype == RoadType::BaseHandle)
298.         {
299.             freezoneRecognition.dirLeft = motionControlle
    r.params.rateTurnFreezone;
300.             cout << "turn" << freezoneRecognition.dirLeft
     << endl;
301.             if (freezoneRecognition.freezoneRecognition(t
    rackRecognition, resultAI->predictor_results, motionControl
    ler.params.FreezoneEnable))
302.             {
303.                 cout << "freezone" << freezoneRecognition
    .freezoneStep << endl;
304.                 cout << "s:" << freezoneRecognition.stopc
    nt << endl;
305.                 if (freezoneRecognition.freezoneStep == f
    reezoneRecognition.FreezoneStop && freezoneRecognition.pass
    cnt == 0)
306.                 {
307.                     freezoneRecognition.passcnt++;
308.                 }
309.                 */
310.                 roadType = RoadType::FreezoneHandle;
311.                 if (motionController.params.debug)
312.                 {
```

```
313.                     Mat imageFreezone = Mat::zeros(Size(C
    OLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
314.                     freezoneRecognition.drawImage(trackRe
    cognition, imageFreezone, resultAI->predictor_results);
315.                     imshow("imageRecognition", imageFreez
    one);
316.                     imshowRec = true;
317.                     savePicture(imageFreezone);
318.                 }
319.                 else if (motionController.params.saveImag
    e)  //保存原始图像
320.                 {
321.                     Mat imageFreezone = Mat::zeros(Size(C
    OLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
322.                     freezoneRecognition.drawImage(trackRe
    cognition, imageFreezone, resultAI->predictor_results);
323.                     savePicture(imageFreezone);
324.                 }
325.             }
326.             else
327.                 roadType = RoadType::BaseHandle;
328.         }
329.         }
330.         else{
331.             if (roadType == RoadType::FreezoneHandle || r
    oadType == RoadType::BaseHandle)
332.             {
333.                 if (zhtfreezoneRecognition.freezoneRecogn
    ition(trackRecognition,resultAI->predictor_results))
334.                 {
335.                     roadType = RoadType::FreezoneHandle;
336.                     if (motionController.params.debug)
337.                     {
338.                         Mat imageCross = Mat::zeros(Size(
    COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
339.                         zhtfreezoneRecognition.drawImage(
    trackRecognition, imageCross);
340.                         imshow("imageRecognition", imageC
    ross);
341.                         imshowRec = true;
342.                         savePicture(imageCross);
343.                     }
344.                 }
345.                 else
```

```
346.                     roadType = RoadType::BaseHandle;
347.                 }
348.             }
349.
350.             /* [08] 环岛处理，暂时必须在第 8 个 */
351.             cout << "[081]" << endl;
352.             if (motionController.params.RingEnable) //赛道元素是否使能-？？？
353.             {
354.                 if (roadType == RoadType::RingHandle || roadType == RoadType::BaseHandle)
355.                 {
356.                     if (ringRecognition.ringRecognition(trackRecognition, motionController.params.RingEnterDelayTimeRight, motionController.params.RingEnterAngleRight,
357.                         motionController.params.RingEnterDelayTimeLeft, motionController.params.RingEnterAngleLeft, motionController.params.RingExitAngleRight, motionController.params.RingExitAngleLeft))
358.                     {
359.                         roadType = RoadType::RingHandle;
360.                         if (motionController.params.debug)
361.                         {
362.                             Mat imageCross = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
363.                             ringRecognition.drawImage(trackRecognition, imageCross);
364.                             imshow("imageRecognition", imageCross);
365.                             imshowRec = true;
366.                             savePicture(imageCross);
367.                         }
368.                     }
369.                     else
370.                         roadType = RoadType::BaseHandle;
371.                 }
372.             }
373.
374.             // [08] 十字道路处理
375.             cout << "[082]" << endl;
376.             if (motionController.params.CrossEnable) //赛道元素是否使能
377.             {
```

```
378.             if (roadType == RoadType::CrossHandle || road
   Type == RoadType::BaseHandle)
379.             {
380.                 if (crossroadRecognition.crossroadRecogni
   tion(trackRecognition))
381.                 {
382.                     //roadType = RoadType::CrossHandle;
383.                     if (motionController.params.debug)
384.                     {
385.                         Mat imageCross = Mat::zeros(Size(
   COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
386.                         crossroadRecognition.drawImage(tr
   ackRecognition, imageCross);
387.                         imshow("imageRecognition", imageC
   ross);
388.                         imshowRec = true;
389.                         savePicture(imageCross);
390.                     }
391.                     else if (motionController.params.save
   Image)  //保存原始图像
392.                     {
393.                         Mat imageCross = Mat::zeros(Size(
   COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
394.                         crossroadRecognition.drawImage(tr
   ackRecognition, imageCross);
395.                         savePicture(imageCross);
396.                     }
397.                 }
398.                 else
399.                     roadType = RoadType::BaseHandle;
400.             }
401.         }
402.
403.         // [09] 控制中心计算
404.         cout << "[09]" << endl;
405.         if (trackRecognition.pointsEdgeLeft.size() < 30 &
   & trackRecognition.pointsEdgeRight.size() < 30 && roadType
   != RoadType::FreezoneHandle && roadType != RoadType::SlopeH
   andle) //防止车辆冲出赛道
406.         {
407.             counterOutTrackA++;
408.             counterOutTrackB = 0;
409.             if (counterOutTrackA > 20)
410.                 callbackSignal(0);
```

```
411.          }
412.          else
413.          {
414.              counterOutTrackB++;
415.              if (counterOutTrackB > 50)
416.              {
417.                  counterOutTrackA = 0;
418.                  counterOutTrackB = 50;
419.              }
420.          }
421.          controlCenterCal.controlCenterCal(trackRecognitio
   n, motionController.params.AreaRevise, motionController.par
   ams.alpha); /* 新增AreaRevise、alpha */
422.          //controlCenterCal.controlCenterCal(trackRecognit
   ion); //根据赛道边缘信息拟合运动路径（控制中心）
423.
424.          // [10] 运动控制
425.          cout << "[10]" << endl;
426.          if (counterRunBegin > 20) //智能车启动延时：前几场
   图像+AI 推理不稳定
427.          {
428.              //智能汽车方向控制
429.              motionController.pdController(controlCenterCa
   l.controlCenter); // PD 控制器姿态控制
430.
431.              //智能汽车速度控制
432.              switch (roadType)
433.              {
434.              case RoadType::FreezoneHandle:
                                        //泛行区处理速度
435.                  if (motionController.params.FreezoneEnabl
   e)                                   // AI 区域的速度
436.                      motionController.motorSpeed = motionC
   ontroller.params.speedFreezone; //匀速控制
437.                  else
                                        //非AI 区域的速度
438.                      motionController.speedController(true
   , controlCenterCal);             //变加速控制
439.                  break;
440.              case RoadType::GasstationHandle:
                                    //加油站速度
441.                  motionController.motorSpeed = motionContr
   oller.params.speedGasBusy; //匀速控制
442.                  break;
```

```
443.                case RoadType::BusyareaHandle:
                                    //施工区速度
444.                    motionController.motorSpeed = motionContr
    oller.params.speedGasBusy; //匀速控制
445.                    break;
446.                case RoadType::SlopeHandle:
                                //坡道速度
447.                    motionController.motorSpeed = motionContr
    oller.params.speedSlop; //匀速控制
448.                    break;
449.                case RoadType::RingHandle:
                                //坡道速度
450.                    motionController.motorSpeed = motionContr
    oller.params.speedRing; //匀速控制
451.                    break;
452.                default:
                    //基础巡线 | 十字 |环岛速度
453.                    motionController.speedController(true, co
    ntrolCenterCal); //变加速控制
454.                    break;
455.                }
456.
457.                if (!motionController.params.debug) //调试模式
    下不控制车辆运动
458.                {
459.                    driver->carControl(motionController.motor
    Speed, motionController.servoPwm); //串口通信，姿态与速度控制
460.                }
461.            }
462.            else
463.                counterRunBegin++;
464.
465.            // [11]调试模式下图像显示和存图
466.            cout << "[11]" << endl;
467.            if (motionController.params.debug)
468.            {
469.                controlCenterCal.drawImage(trackRecognition,
    imgaeCorrect);
470.                switch (roadType)
471.                {
472.                case RoadType::BaseHandle:

                    //基础赛道处理
```

```
473.                 putText(imgaeCorrect, "[1] Track", Point(
    10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 0, 25
    5), 1, CV_AA); //显示赛道识别类型
474.                 break;
475.             case RoadType::RingHandle:

                //环岛赛道处理
476.                 putText(imgaeCorrect, "[1] Ring", Point(1
    0, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255, 0
    ), 1, CV_AA); //显示赛道识别类型
477.                 break;
478.             case RoadType::CrossHandle:

                //十字道路处理
479.                 putText(imgaeCorrect, "[1] Cross", Point(
    10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255,
    0), 1, CV_AA); //显示赛道识别类型
480.                 break;
481.             case RoadType::FreezoneHandle:

                //泛行区处理
482.                 putText(imgaeCorrect, "[1] Freezone", Poi
    nt(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 25
    5, 0), 1, CV_AA); //显示赛道识别类型
483.                 break;
484.             case RoadType::GarageHandle:

                //车库处理
485.                 putText(imgaeCorrect, "[1] Garage", Point
    (10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255,
     0), 1, CV_AA); //显示赛道识别类型
486.                 break;
487.             case RoadType::GasstationHandle:

                //加油站处理
488.                 putText(imgaeCorrect, "[1] Gasstation", P
    oint(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0,
    255, 0), 1, CV_AA); //显示赛道识别类型
489.                 break;
490.             case RoadType::BusyareaHandle:

                //施工区处理
```

```
491.                 putText(imgaeCorrect, "[1] Busyarea", Poi
    nt(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 25
    5, 0), 1, CV_AA); //显示赛道识别类型
492.                 break;
493.             case RoadType::SlopeHandle:

                //坡道处理
494.                 putText(imgaeCorrect, "[1] Slop", Point(1
    0, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255, 0
    ), 1, CV_AA); //显示赛道识别类型
495.                 break;
496.             }
497.
498.             putText(imgaeCorrect, "v: " + formatDoble2Str
    ing(motionController.motorSpeed, 2), Point(COLSIMAGE - 60,
    80), FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255), 1); //车速
499.             /* 显示角度 */
500.             putText(imgaeCorrect, "a: " + formatDoble2Str
    ing(motionController.servoPwm, 2), Point(COLSIMAGE - 90, 10
    0), FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255), 1);
501.             string str = to_string(circlesThis) + "/" + t
    o_string(motionController.params.circles);
502.             putText(imgaeCorrect, str, Point(COLSIMAGE -
    50, ROWSIMAGE - 20), cv::FONT_HERSHEY_TRIPLEX, 0.5, cv::Sca
    lar(0, 255, 0), 1, CV_AA); //显示圈数
503.             if (!imshowRec)

                                //保持调试图像存储顺序和显示一致性
504.             {
505.                 Mat imageNone = Mat::zeros(Size(COLSIMAGE
    , ROWSIMAGE), CV_8UC3); //初始化图像
506.                 imshow("imageRecognition", imageNone);
507.                 savePicture(imageNone);
508.             }
509.             imshow("imageControl", imgaeCorrect);
510.             savePicture(imgaeCorrect);
511.
512.             char c = waitKey(10);
513.         }
514.         else if (motionController.params.saveImage)  //保
    存原始图像
515.         {
516.             controlCenterCal.drawImage(trackRecognition,
    imgaeCorrect);
```

```
517.                switch (roadType)
518.                {
519.                    case RoadType::BaseHandle:

                        //基础赛道处理
520.                        putText(imgaeCorrect, "[1] Track", Po
    int(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 0
    , 255), 1, CV_AA); //显示赛道识别类型
521.                        break;
522.                    case RoadType::RingHandle:

                       //环岛赛道处理
523.                        putText(imgaeCorrect, "[1] Ring", Poi
    nt(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 25
    5, 0), 1, CV_AA); //显示赛道识别类型
524.                        break;
525.                    case RoadType::CrossHandle:

                        //十字道路处理
526.                        putText(imgaeCorrect, "[1] Cross", Po
    int(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 2
    55, 0), 1, CV_AA); //显示赛道识别类型
527.                        break;
528.                    case RoadType::FreezoneHandle:

                          //泛行区处理
529.                        putText(imgaeCorrect, "[1] Freezone",
     Point(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0
    , 255, 0), 1, CV_AA); //显示赛道识别类型
530.                        break;
531.                    case RoadType::GarageHandle:

                         //车库处理
532.                        putText(imgaeCorrect, "[1] Garage", P
    oint(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0,
    255, 0), 1, CV_AA); //显示赛道识别类型
533.                        break;
534.                    case RoadType::GasstationHandle:

                           //加油站处理
535.                        putText(imgaeCorrect, "[1] Gasstation
    ", Point(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar
    (0, 255, 0), 1, CV_AA); //显示赛道识别类型
536.                        break;
```

```
537.                 case RoadType::BusyareaHandle:
                         //施工区处理
538.                     putText(imgaeCorrect, "[1] Busyarea", Point(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255, 0), 1, CV_AA); //显示赛道识别类型
539.                     break;
540.                 case RoadType::SlopeHandle:
                        //坡道处理
541.                     putText(imgaeCorrect, "[1] Slop", Point(10, 20), cv::FONT_HERSHEY_TRIPLEX, 0.3, cv::Scalar(0, 255, 0), 1, CV_AA); //显示赛道识别类型
542.                     break;
543.             }
544.
545.             putText(imgaeCorrect, "v: " + formatDoble2String(motionController.motorSpeed, 2), Point(COLSIMAGE - 60, 80), FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255), 1); //车速
546.             /* 显示角度 */
547.             putText(imgaeCorrect, "a: " + formatDoble2String(motionController.servoPwm, 2), Point(COLSIMAGE - 90, 100), FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255), 1);
548.             string str = to_string(circlesThis) + "/" + to_string(motionController.params.circles);
549.             putText(imgaeCorrect, str, Point(COLSIMAGE - 50, ROWSIMAGE - 20), cv::FONT_HERSHEY_TRIPLEX, 0.5, cv::Scalar(0, 255, 0), 1, CV_AA); //显示圈数
550.             if (!imshowRec)
                        //保持调试图像存储顺序和显示一致性
551.             {
552.                 Mat imageNone = Mat::zeros(Size(COLSIMAGE, ROWSIMAGE), CV_8UC3); //初始化图像
553.                 savePicture(imageNone);
554.             }
555.             savePicture(imgaeCorrect);
556.         }
557.     }
558.
559.     return 0;
560. }
561.
562. /**
```

```
563.  * @brief OpenCV 图像显示窗口初始化（详细参数/Debug 模式）
564.  *
565.  */
566. void displayWindowDetailInit(void)
567. {
568.     //[1] 二值化图像：Gray
569.     string windowName = "imageTrack";
570.     cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
571.     cv::resizeWindow(windowName, 320, 240);     //分辨率
572.     cv::moveWindow(windowName, 10, 10);         //布局位置
573.
574.     //[2] 赛道边缘图像：RGB
575.     windowName = "imageRecognition";
576.     cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
577.     cv::resizeWindow(windowName, 320, 240);     //分辨率
578.     cv::moveWindow(windowName, 10, 320);        //布局位置
579.
580.     //[3] 原始图像/矫正后：RGB
581.     windowName = "imageControl";
582.     cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
583.     cv::resizeWindow(windowName, 640, 480);     //分辨率
584.     cv::moveWindow(windowName, 350, 20);        //布局位置
585.
586.     /*frame
587.     windowName = "frame";
588.     cv::namedWindow(windowName, WINDOW_NORMAL); //图像名称
589.     cv::resizeWindow(windowName, 320, 240);     //分辨率
590.     cv::moveWindow(windowName, 350, 320);        //布局位
    置
591.     */
592. }
593.
594. /**
595.  * @brief 系统信号回调函数：系统退出(按键 ctrl+c)
596.  *
597.  * @param signum 信号量
598.  */
599. void callbackSignal(int signum)
600. {
601.     driver->carControl(0, PWMSERVOMID); //智能车停止运动
602.     cout << "====System Exit!!!  -
    ->  CarStopping! " << signum << endl;
603.     exit(signum);
604. }
```