第十七届全国大学生

智能汽车竞赛

**技 术 报 告**

基于摄像头图像识别的小车寻迹研究

学 校：大连理工大学

队伍名称：我们笑着说队

参赛队员：崔俊涛

魏桐

马文龙

带队教师：吴振宇 李胜铭

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：

带队教师签名：

日 期：

目录

[引 言 1](#_Toc81175553)

[第一章 方案设计 3](#_Toc81175554)

[1.1系统总体方案的选定 3](#_Toc81175555)

[1.2系统的总体方案设计 3](#_Toc81175556)

[第二章 机械结构调整与优化 5](#_Toc81175557)

[2.1 智能汽车车模的选择 5](#_Toc81175558)

[2.2 智能汽车传感器的安装 5](#_Toc81175559)

[2.2.1 速度传感器的安装 6](#_Toc81175560)

[2.2.2电磁传感器的安装 6](#_Toc81175561)

[2.2.3摄像头传感器的安装 6](#_Toc81175562)

[2.3.1 电路板的安装 7](#_Toc81175563)

[2.3.2 电池安放 7](#_Toc81175564)

[2.4 其他机械结构的调整 7](#_Toc81175565)

[2.4.1 机械差速的调整 7](#_Toc81175566)

[2.4.1 车轮表面的处理 7](#_Toc81175567)

[2.5 小结 8](#_Toc81175568)

[第三章 硬件电路方案设计 9](#_Toc81175569)

[3.1单片机最小系统模块 9](#_Toc81175570)

[3.2电源模块 9](#_Toc81175571)

[3.2.1电池的使用 9](#_Toc81175572)

[3.2.2 稳5.658V电源模块 10](#_Toc81175573)

[3.2.3 稳5V电源模块 10](#_Toc81175574)

[3.2.4稳3.3V电源模块 11](#_Toc81175575)

[3.3 电机驱动模块 12](#_Toc81175576)

[3.4 串口通信模块 12](#_Toc81175577)

[3.5 运算放大器 12](#_Toc81175578)

[3.6总结 13](#_Toc81175579)

[第四章 小车的控制方法 15](#_Toc81175580)

[4.1图像采集思路 15](#_Toc81175581)

[4.2图像处理思路 16](#_Toc81175582)

[4.2.1边界的获取 16](#_Toc81175583)

[4.2.2 特殊元素的处理 17](#_Toc81175584)

[4.3 电感采集、控制策略 18](#_Toc81175585)

[4.4 速度采集以及速度控制 19](#_Toc81175586)

[4.5 舵机打角控制策略 19](#_Toc81175587)

[第五章 开发工具、制作、安装、调试过程 20](#_Toc81175588)

[5.1 上位机 20](#_Toc81175589)

[5.1.1 串口功能 20](#_Toc81175590)

[5.1.2 画图功能 21](#_Toc81175591)

[5.1.3 CCD显示功能 23](#_Toc81175592)

[5.1.4 摄像头图像采集功能 24](#_Toc81175593)

[5.1.5 使用上位机调试过程 25](#_Toc81175594)

[5.2 AURIX Development Studio（ADS） 25](#_Toc81175595)

[5.2.1 单步调试与断点功能 25](#_Toc81175596)

[5.2.1 变量查看功能 26](#_Toc81175597)

[第六章 车模主要参数 27](#_Toc81175598)

[6.1智能车外形参数 27](#_Toc81175599)

[6.2智能车技术参数 27](#_Toc81175600)

[结论 29](#_Toc81175601)

[参 考 文 献 31](#_Toc81175602)

[附录 32](#_Toc81175603)

[附录A：程序源码 32](#_Toc81175604)

[附录B：原理图总图 69](#_Toc81175605)

**引 言**

随着近几年的电子科学技术的不断发展，电子科技的影响逐渐深入到各行各业之中，越来越多的自动化设备开始逐渐渗透到人们的生产生活之中，嵌入式的迅猛发展也为智能研究提供了更为广阔的平台。并且随着5G网络的到来，无人驾驶汽车的研究与发展步入了一个新的快车道。在此背景之下，通过联合各种传感器和控制器组成的控制方式就成为了新的研究热点，控制算法和控制策略也显得更加重要。

全国大学生杯智能汽车竞赛是国家教学质量与教学改革工程资助项目，通过使用德国英飞凌半导体公司生产的TC264单片机作为车载核心控制模块，增加道路传感器、摄像头、电机驱动电路以及编写相应的程序，制作出了一个可以自主识别道路的汽车模型。因而该竞赛是涵盖了智能控制、模式识别、传感技术、汽车电子、电气、计算机、机械等多个学科的比赛，对学生的知识融合和实践能力的提高，具有良好的推动作用。

本文采用第十七届全国大学生智能车竞赛的汽车模型——C型车模作为研究平台，以32位单片机TC264作为主控制单元，运用AURIX Development软件作为开发工具进行智能控制策略研究。道路信息检测模块为MT9V034总钻风摄像头来进行赛道信息的获取。本届车模是C型车模，采用后置两个电机驱动，需要两个编码器测速，通过PID算法，将速度进行闭环控制。

在电源模块的设计中，采用线性稳压电源LM2940和两个AMS-1117-3.3 分别产生5V和3.3V电压，其中一个AMS-1117-3.3的输出电压为编码器、摄像头、串口通信、按键、LCD、拨码等模块供电，另一个的输出只为为单片机供电，两个3.3V供电模块的分别使用可以保证单片机的供电电压稳定，是单片机运行更加可靠、高效。最后还有一片LM2596-adj为舵机供电，因为舵机的供电范围是不固定的，虽然舵机的型号是固定的，但是不同的工作电压会让电机有不同的响应速度和扭矩等，所以选用一个可调的稳压模块方便对车模的舵机模块进行调试优化。

单片机的算法设计部分为本论文论述的重点，主要体现了运用PID控制理论完成对车模速度控制，运用最长白列算法将获得的图像进行更精确的识别等。

本篇技术报告将从智能车的机械结构、硬件电路、算法设计等方面全面详细的介绍整个准备过程。

**第一章 方案设计**

本章主要介绍了智能车系统总体方案的选定和总体设计思路。

## 1.1系统总体方案的选定

本届智能汽车比赛，我队为四轮摄像头组别。在寻迹传感器方面，选择使用摄像头使用。摄像头选择MT9V034总钻风，使用编码器测速进行速度闭环控制。

## 1.2系统的总体方案设计

系统的硬件电路采用模块化设计方式。主要分为主控制板、电机驱动及测速、电磁信号采集和图像采集五个部分，主控制板上集成单片机最小系统，主控制板与其他功能板使用灰排线功率线连接。

主控板部分：

主控板上包括一片TC264作为核心控制器，一片LM2940作为5V供电的电源芯片，两片AMS1117-3.3分别为单片机和其他元件分开供电，还有一片LM2596为舵机供电，目的是让单片机在更稳定的电压环境中工作。以及一些辅助模块按键、LED、拨码、LCD等用于操作，通过预留串口进行蓝牙连接，通过方便调试和数据获取。

辅助模块通过互相配合和自己写的GUI界面进行辅助操作，方便观察车模运行状态和修改参数、调节档位等功能。

电源部分主要包括8V到5V降压模块，3.3V稳压模块，和舵机电源电路。其中由于舵机摆角时瞬时功率较大同时对电源纹波要求并不严格，所以选用的LM2596-adj供电，并且使用可调电压输出更方便对舵机的机械性能优化。外设和单片机选择3.3V线性电源供电。

信号采集：

信号采集部分分别为电磁和摄像头两部分。

电磁信号采集系统采用6.8nF电容和10mH电感配对采集电磁导引信号，经过LM324放大和滤波电路后得到波动较小的直流电压信号，利用TC264内部自带的ADC采集引脚进行信号的采集。

摄像头采集使用的是MT9V034总钻风，该摄像头可以通过写寄存器配置来改变对比度、曝光等设置，可以满足图像的采集功能。

电机驱动：

电机驱动板包含为MCU和各外设供电的电源电路，电机驱动板使用16P软排线排线与主控制板相连，包括一路电机驱动、一路测速以及液晶屏接口。电机驱动采用两个半桥驱动芯片BTN7971构成H桥电路，具有简单、高集成度的优点，可靠便捷。测速预留6pin的编码器接口。液晶屏选择SPI接口的ILI9341芯片控制的1.8寸TFT液晶屏。

**第二章 机械结构调整与优化**

智能汽车各系统的控制都是在机械结构的基础上实现的，因此在设计整个软件架构和算法之前一定要对整个模型车的机械结构有一个全面清晰的认识，然后建立相应的数学模型，从而再针对具体的设计方案来调整赛车的机械结构，并在实际的调试过程中不断的改进优化和提高结构的稳定性。本章将主要介绍C车模的机械结构和调整方案。

## 2.1 智能汽车车模的选择

本届比赛四轮摄像头组使用C车模.

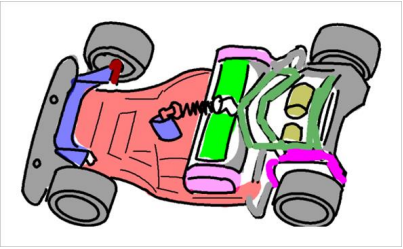


图2.1 智能汽车C车模

## 2.2 智能汽车传感器的安装

车模中的传感器包括有：速度传感器（编码器），摄像头传感器。下面分别介绍这些传感器的安装。

### **2.2.1 速度传感器的安装**

速度传感器我们使用了龙邱mini512线带方向编码器，安装方法如下图：

X线含义是编码器旋转一圈产生 X个脉冲，通过脉冲计数可以读取到数值A。这个数值A和小车的速度呈线性关系，一般情况下可以不去换算具体速度直接使用即可。

若想转换为具体速度，则需要则需要知道以下参数：车模轮子周长 X0、车模齿轮的齿数 X1、编码器齿轮的齿数 X2、编码器的线数 X3、ENC 读到数据 X4、ENC 读取时间间隔 X5，速度换算公式：V = ( (X4 / X3)  \* X2 / X1 ) \* X0 / X5。



图2.2 编码器安装示意图

### **2.2.2摄像头传感器的安装**

对于摄像头传感器，我们使用了MT9V034总钻风摄像头。虽然比赛没有限制了摄像头高度，但是由于为了获取图像信息完整，我们没有设置太高，并且通过螺丝螺母将摄像头固定在碳纤维杆上同时用热熔胶加固。对于摄像头的安装角度，由于摄像头是120°广角摄像头，角度太靠近地面会导致画面畸变较大，太远离地面又会导致画面无用部分所占比例较大，因此对于角度的调节要恰当适中。同时虽然有大津法计算图像阈值，但是大津法运算太慢，进而导致图像帧数不高，该问题也一直存在于我们整个调车过程中。

### **2.3.1 电路板的安装**

为了使智能车具有较好的稳定性及转向性能，我们在搭建智能车时将主控板放到了车模的底板上，并用热熔胶和尼龙柱牢固地固定到了车体上。电机驱动电路板则是直接安装到了电机上方，和主控板之间用灰排线和功率电源线连接。

### **2.3.2 电池安放**

由于我们使用的是体积和重量较大的镍镉电池，我们将电池放到了车底盘靠近电机的部分，通过粘带固定，方便的同时进一步降低车模的重心。

## 2.4 其他机械结构的调整

### **2.4.1 车轮表面的处理**

对于车胎，C车模采用了空心胎皮。但对于尤其是新买的车轮，因为车轮因为与赛道摩擦少，还是光亮面，会降低车轮的抓地力，所以需要对轮子的表面进行处理，以保持其粘滞性，提高抓地力。

## 2.5 小结

一个良好的机械结构是一辆“神车”基础，关乎车的整体的性能，包括车的加速性能、转弯性能等。在相同程序代码的控制环境下，机械结构直接影响着车模是否可以在较高速度下正常行驶。因此我们也花了较多的时间在改进车模的机械结构上，例如调节前轮悬挂的松紧，将前轮倾角调节至内倾内八，在可调节的范围内降低车模的重心等。经过大量的理论实验与研究调节机械结构之后，小车整体的稳定性和可靠性得到了较大的提高。

**第三章 硬件电路方案设计**

本系统采用模块化设计方式，主要包括单片机最小系统模块、电源模块、电磁信息采集模块、图像信息采集模块、串口通信模块、测速模块、显示操作模块等部分。

## 3.1单片机最小系统模块

本设计核心控制器为英飞凌公司生产的32位单片机TC264，该单片机具有144引脚，双核设计，最大频率位200MHz，具有2.5Mbyte的program flash和96Kbyte的Data Flash.具有硬件I2C接口和SPI接口，和独立的ADC模块等。

这些丰富的系统资源可以更方便的连接外部电路，多个独立的ADC通道也使得外围的某些ADC模块的设计得到了简化，多种通讯接口也方便了扩展更多的外设和电路功。最高200MHz的频率也让该单片机具有更快的计算速度，使得可以最大的减小数据处理时间。并且TC264双核的结构可以使两个和分别执行不同的命令，丰富功能，提高运行效率。

## 3.2电源模块

电源模块是系统稳定工作的基础不管是单片机的稳定运行还是ADC的采集数据等硬件电路的良好运行，都离不开一个稳定的电源系统。因此，一个具有稳定的电压是粗话模块和电流的稳定性在整个的智能车系统中起着十分重要的作用。

### **3.2.1电池的使用**

电池采用的是2000mAh的镍镉电池，该类型电池的特点就是“结实”、价格便宜，作为官方推荐的电池，镍镉电池完全可以胜任对整个智能车系统供电的要求。

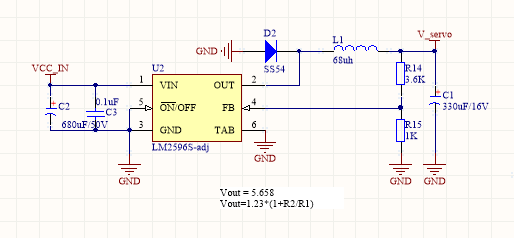
但是由于镍镉电池电池容量小，具有记忆性的缺点，每次使用不能对电池进行过度放电，并且每次充电前必须先放电才能在充电。

电池通常电流高，电池的爆发力会强些，但未必如电流低时饱满、持久。同时我们也必须注意充电电流不能过高，当电流过高时，不仅不能提高电池性能，反而会损坏电池，严重时会导致电池起火、爆炸。电池充满电时，电压大约为8V。在电池压小于8V时，应注意及时充电，电池过放会对其造成不可逆转的损害，电压低于7V会对电池造成毁灭性伤害，导致以后再使用时会出现放电量不足的现象，影响整个系统的运行，更严重的可能会导致单片机因为掉电而复位。

### **3.2.2 稳5.658V电源模块**

使用LM2596-adj将电压降到5.628V供舵机使用。电路原理图如图3.1所示。

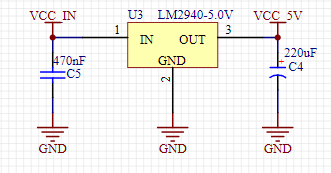
通过R14和R15构成的反馈回路调整输出电压的大小。因为舵机的型号种类不同，所以设计可调输出电压的稳压模块微电机供电就可以避免在更换电机的时候必须更换整个电路板的尴尬情况。



图片3. 1稳5.658电源模块

### **3.2.3 稳5V电源模块**

5V稳压电源模块如图3.2所示。

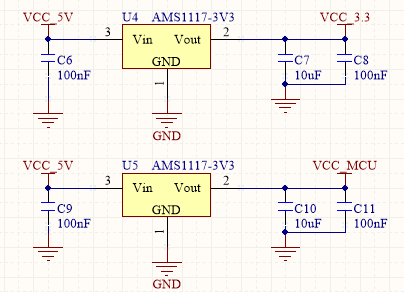


图片3. 2 5V稳压模块

### **3.2.4稳3.3V电源模块**

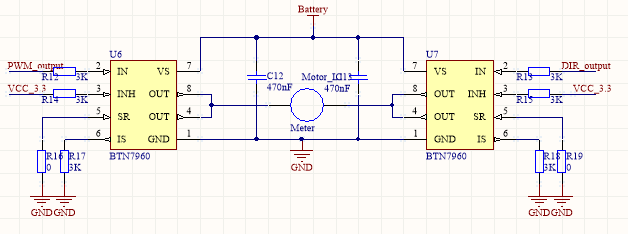
由于单片机的额定电压为3.3V，串口通信等模块也为3.3V，所以将电压稳定在3.3V为各个模块供电是必不可少的。

电路的设计原理如图3.3所示。



图片3. 3 3.3V稳压模块

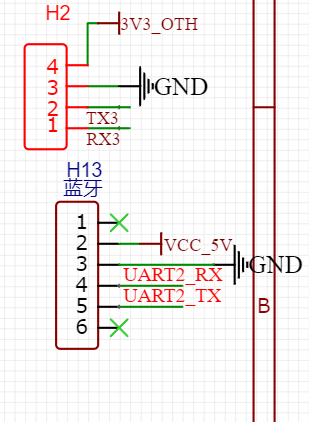
## 3.3 电机驱动模块

本智能车车模重量不大，并且电机型号较小，对电机驱动的输出电流要求并不苛刻，因此本设计的驱动电路由两片BTN7971芯片构成H桥电路。通过控制电流的流通方向来实现电机的正反转。并通过控制输入的PWM波的占空比来调节电机的平均输出功率，达到控制电机转速的目的。具体电路如图3.4所示。

图片3. 4 电机驱动电路

## 3.4 串口通信模块

调车的过程中，需要实时的观察变量变化情况，通过下载线在Live Watch 窗口中观察变量很受局限，所以采用蓝牙无线通信模块，波特率为115200，传输速度更快，配合上位机观察超调量，有助于调试PID参数，还可以用蓝牙来无线传输采到的数据。具体电路图如图3.5所示。



图片3. 5串口通信原理图

## 3.6总结

一个良好的、运行稳定的硬件电路是一个智能车系统的核心，电路就像是人体中的神经网络，将单片机发出的指令传给各个模块，使各个模块之间配合的更加高效。而电路的鲁棒性也需要更加的强大，才能保证智能小车可以更稳定、更持久的运行下去，尤其是今年国赛的要求是十分钟的最好成绩，这更加考验了智能小车硬件运行的稳定性和耐久性。

**第四章 小车的控制方法**

四轮摄像头组多采用摄像头采集数据，力求以最快的速度，跑完整个赛道。不论上哪种方案，软件的总体框架总是相似的，我们的想法是先调稳定，再提速度。软件上大类分为图像采集、图像处理，速度采集以及速度控制。其中以图像处理必要。

## 4.1图像采集思路

摄像头选择MT9V034，传输回的是160\*128的灰度图像

在我们日常练习的赛道上，另一个更加严峻的问题是自然强光的干扰，分为直射光和反射光。由于摄像头较低，想要获得较长的前瞻就不得不将摄像头俯角尽量减小，但是这样一来只有下半部分是赛道信息，上半部分不仅没有参考意义，还会有很强的直射自然光干扰，在对比之下赛道二值化成了一团黑色，调节对比度阈值也无济于事。也是由于摄像头较低，远处的自然光在地板、墙面上的反射也对摄像头二值化干扰较大，我们唯一能做的就是加装偏振片，但是效果仍不理想。 只得在程序中采取各种过滤、补齐方法减小影响。

## 4.2图像处理思路

图像处理的大思路是边扫描边处理，即每一行、每一列扫描的同时对赛道元素进行分析、操作。这样做是因为我们主要采用从上一行的中间点开始向两边扫描边界的方法，这样在十字等地方遇到向下的尖角时就面临选边问题（见图4.2.1），为了让中点有一个大致的取向，就采取了一定的预测手段，向前补线。

### **4.2.1边界的获取**

我们采用了横向扫描的方法。即从最下面扫到最上面，

补线方面有断点补线和宽度补线两种，都是在横向扫描每一行结束后的操作。断点补线就是在二阶差分足够大时认为其是断点，向下取一斜率，下一回开始以断点为依据，依斜率补线，此时关闭边界扫描及其他判定。宽度补线就是在一边丢线后依经验得到的宽度公式（这里只采用了y=ax+b的简易拟合），将另一边补出来。由于直道和弯道的宽度不同，由纵向贯通列的多少大致地区分，对应不同的预测公式。两个补线均具有预测性质，是基于人的习惯做出的模仿，断点补线的预测成分更大一些，且一开始就是为了应对十字而存在的，最后有了分辨十字元素的思路以后就显得鸡肋了，在弯道等地方会经常误触。加上路肩后斜率会求错，如果有机会继续调车的话多半会舍弃掉。

### **4.2.2 特殊元素的处理**

处理特殊元素的思路是抓住特征，不同阶段之间衔接灵活，最好可以在一定引导下正常循迹，同时存在冗余，保证未正常触发下一阶段时也可完成整个元素。

以圆环为例。入环使用的是最常规的补线，将纵向二阶差分大的点与左/右下角连接，为了衔接平滑，结束补线的标志是前瞻行搭到此断点以上。出环时先借助断点补线维持一定打角，当补线结束时再以环内打角的平均值延一段距离，解除死打角的标志是判定到直道。入环时为了保证不读到旁边的直道，横向中点读到纵向二阶差分大的地方会限幅到入环方向以内。

三叉的判别是根据俯视图“丁”字形的横向贯通、纵向截断的特点来的。入三叉方法与入圆环方法类似，将上边界最低点与左右下角连线，由于下凸不是很明显，难以由纵向二阶差分准确判断前瞻行是否搭在断点上面，于是模糊地认为最低点到边界一定范围内就进入三叉了。

## 4.3 速度采集以及速度控制

本次采用编码器测速，优点是用齿轮啮合，测速稳定，使用中出现的问题是测得的瞬时速度波动较大，单次测速作为参考不太可靠。应对措施是采取一定的过滤方法，并取十次速度的平均值作为参考。

电机开始采用增量式PID，可能是由于参数没调好，发现电机超调严重，例如目标速度是70时会一路加速到110以上，相比之下减速略显不足，弯道车模甚至会竖起来，电机也会容易发烫。于是改用了位置式PID，电机超调明显减缓，加减速响应及时了不少，甚至运行时的声音也低了不少。

速度的控制策略就是简单的直道高目标速度，弯道低目标速度，其中环岛采用固定目标速度，以求稳定通过。上坡采用比直道更高的目标速度，以减小机械上被动减速的影响。

## 4.5 舵机打角控制策略

在较高的电压下，普通的塑料舵机容易损坏，于是我们更换了金属舵机，之后再没有出现损坏。我们的舵机使用PWM波控制，由位置式PD给出占空比的值，频率采用50Ｈz，因为舵机的反应时间为0.1s左右，采用更高的频率意义不大（个人观点）。

输入舵机打角的值是参考行上读取的赛道中间值和屏幕中间值的差，参考行由纵向扫描的最长白列乘以一定的比例（宏定义）获得，随着调车进程推进，速度上升，比例也逐渐加大。

PD值的选取依然按直道和弯道加以区分，直道PD均小，弯道PD均大。考虑到一般的弯道有专门多余的打角以配合速度，环岛这种持续的打角额外采用比直道更小的PD来改善循迹。

**第五章 开发工具、制作、安装、调试过程**

## 5.1 上位机

在调试过程中，我们使用的是由智能车实验室成员自主设计开发并适合摄像头、光电组别的上位机。大体功能如下：

### **5.1.1 串口功能**

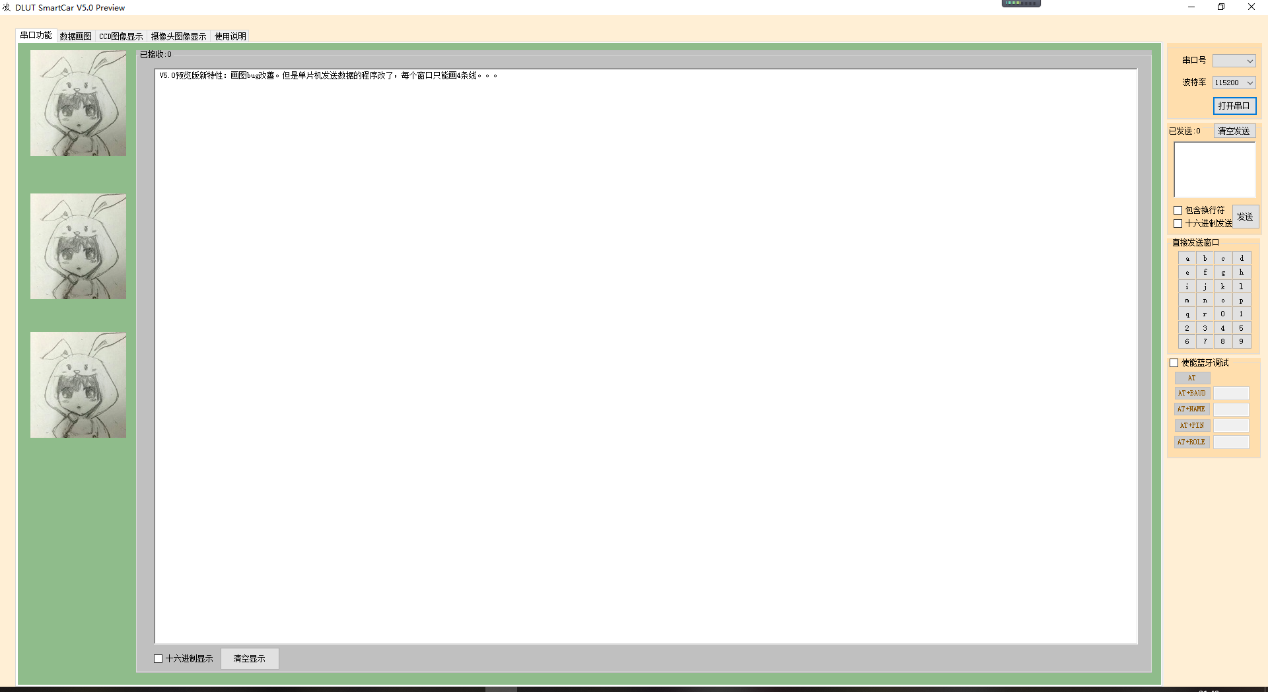


图5.1 串口功能界面

1、普通查看接收功能

选择串口号，和波特率。默认串口号位所有可以使用的串口中号码最小的，默认波特率位115200（考虑速度和误码率确定为115200），然后选择“打开串口”。

2、普通发送功能

必须打开串口此功能才能用。有“显示发送了多少字节”功能，“清空显示”功能，还可选择“包含换行符”，“16进制发送”功能。

3、直接发送字节功能

4、蓝牙调试功能

使用此功能必须打开串口，然后“使能蓝牙调试”框勾选。

注意：

AT+BAUD后边是1,2,3,4,5,6,7等。4代表9600bps，8代表115200bps

AT+NAME可以是英文或者数字组合。

AT+NAME可以是英文或者数字组合。

AT+ROLE后边是0，1。0是从机，连接单片机，1是主机，连接电脑。

### **5.1.2 画图功能**

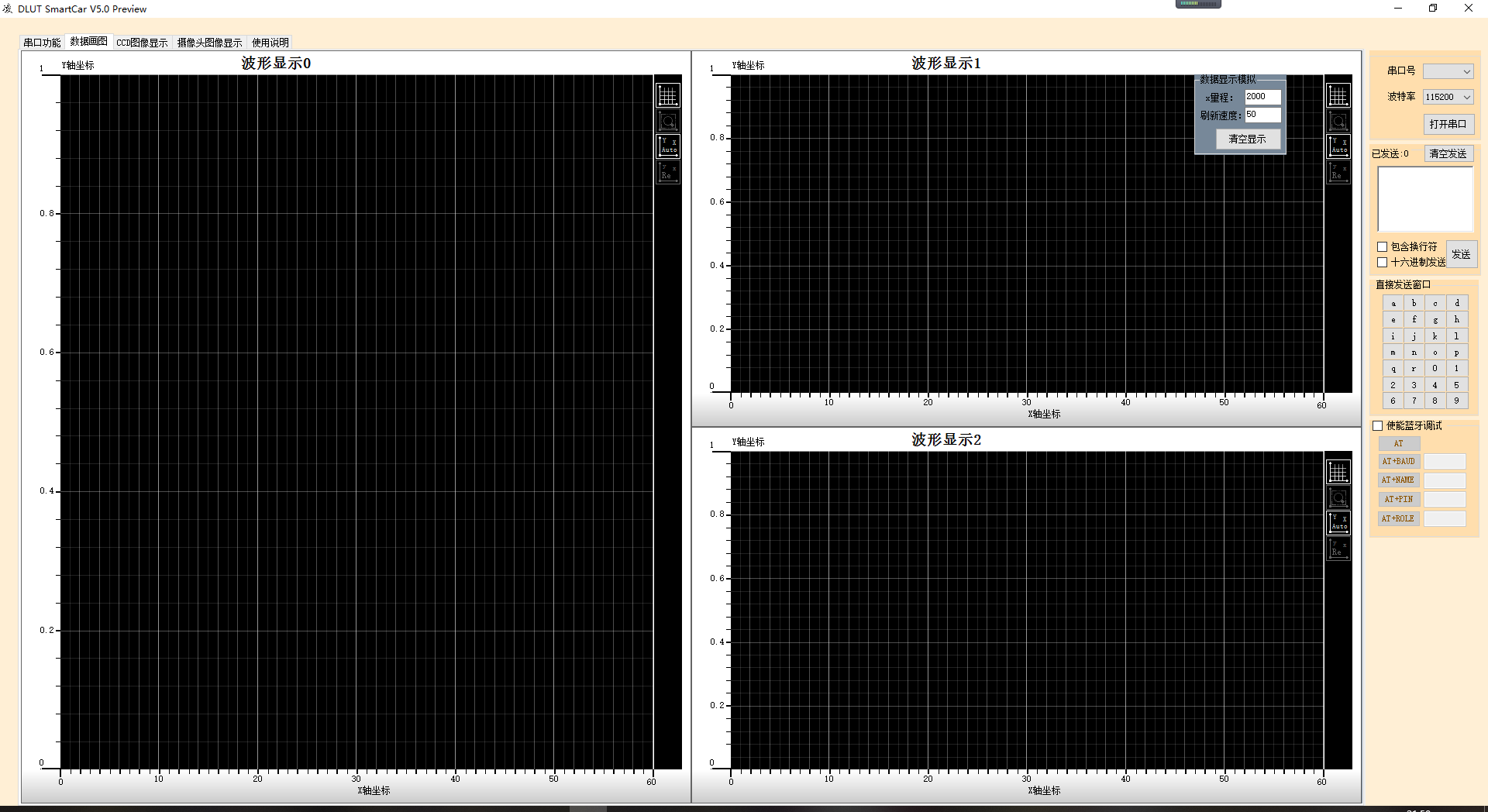


图5.2 画图功能界面

单片机程序：

//1.所用数据

static unsigned char i,j;//循环需要

static unsigned short int send\_data[3][4]={{0},{0},{0}};

//上位机可以接收3幅图像，每幅最多4条曲线，每条曲线代表一个16位变量

static unsigned char send\_num[3]={3,0,0};

//2.将想发的变量赋值给send1\_data，send2\_data，send3\_data,就是把代码中“10,20,30,40”//这些数字替换为全局变量变量即可。不发的数据默认为0。

send\_data[0][0] = (unsigned short int)(10);

send\_data[0][1] = (unsigned short int)(20);

send\_data[0][2] = (unsigned short int)(30);

send\_data[0][3] = (unsigned short int)(40);

send\_data[1][0] = (unsigned short int)(0);

send\_data[1][1] = (unsigned short int)(0);

send\_data[1][2] = (unsigned short int)(0);

send\_data[1][3] = (unsigned short int)(0);

send\_data[2][0] = (unsigned short int)(0);

send\_data[2][1] = (unsigned short int)(0);

send\_data[2][2] = (unsigned short int)(0);

send\_data[2][3] = (unsigned short int)(0);

//3.三幅图像的字头,start的前两个字母

uart\_putchar(uratn,'S');

uart\_putchar(uratn,'T');

//4.发送数据

for(i=0;i < 3;i++)

for(j=0;j < send\_num[i];j++)

{

uart\_putchar(uratn,send\_data[i][j]);

uart\_putchar(uratn,send\_data[i][j]>>8u);

}

### **5.1.3 CCD显示功能**

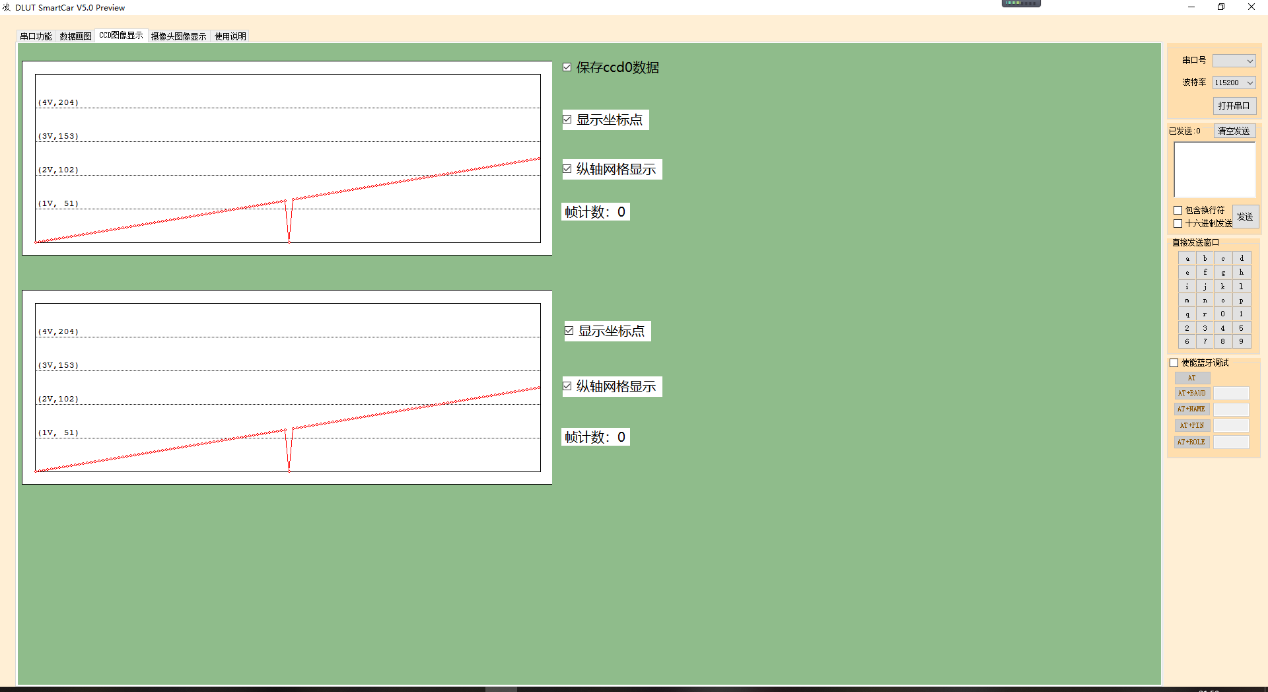


图5.3 CCD显示功能界面

单片机程序：

void Send\_ONEccddata\_to\_PC()

//发送一个ccd的数据，在上位机上也要进行选择，ccddata0为用户自己定义的ccd数据全局变量

{

//字头为0和255

uart\_putchar(0);

uart\_putchar(255);

for(int i=0;i<128;i++)

uart\_putchar(ccddata0[i]);

}

### **5.1.4 摄像头图像采集功能**

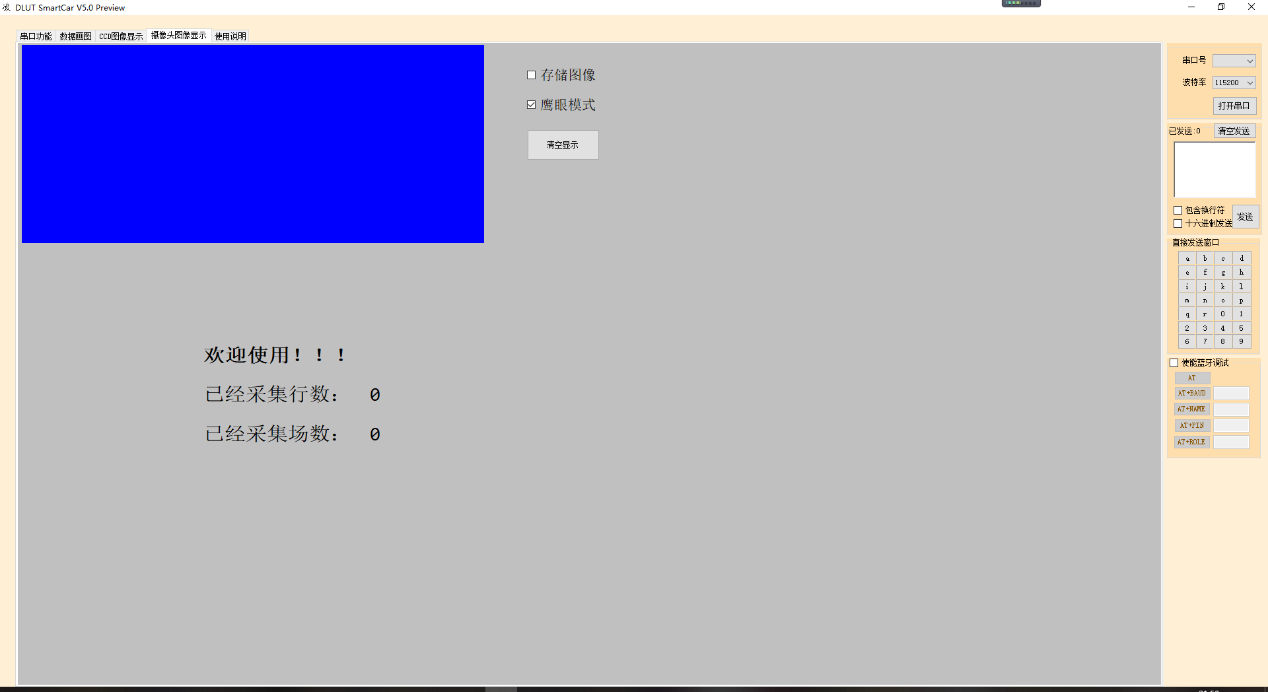


图5.4 摄像头图像采集功能界面

单片机程序：

//send fireImage camera Data Image fire为用户自己定义的摄像头数据全局变量

void SCI\_Send\_fireImage(UARTn uratn)

{

int H = 240, L = 320; //设置摄像头画面长宽

//send "CAMERA"

uart\_putchar(uratn, 'C');

uart\_putchar(uratn, 'A');

uart\_putchar(uratn, 'M');

uart\_putchar(uratn, 'E');

uart\_putchar(uratn, 'R');

uart\_putchar(uratn, 'A');

uart\_putchar(uratn, (unsigned char)(H >> 8));//send Hang

uart\_putchar(uratn, (unsigned char)(H & 0x00FF));

uart\_putchar(uratn, (unsigned char)(L >> 8));//send Lie

uart\_putchar(uratn, (unsigned char)(L & 0x00FF));

//send all data

for (int i = 0; i < H; i++)

for (int j = 0; j < L / 8; j++)

uart\_putchar(uratn, Image\_fire[i][j]);

}

### **5.1.5 使用上位机调试过程**

在智能车的调试过程中，便于人机交互的上位机系统，通过简单明了的可视化界面直观地显示智能汽车的状态，这对我们的调试有很大帮助。我们主要使用了上位机的串口功能和画图功能。在使用串口功能时，配合蓝牙通讯模块，通过printf语句，我们可以在上位机得到小车在行驶过程中实时的电磁传感器的数据、编码器的数据以及各项标志位的值，进而对数据进行分析，找到行驶过程出现问题的原因，避免了我们盲目修改程序，大大提高了我们寻找、发现和解决问题的效率。

## 5.2 AURIX Development Studio（ADS）

在程序开发过程中，我们使用了官方的AURIX Development Studio（ADS）作为开发环境。ADS提供的单步调试与断点功能、变量查看以及内存查看等功能，为我们的在线调试提供了很大的帮助，尤其是在设定某些阈值、查找程序bug等方面，很大程度上提高了调试效率。

### **5.2.1 单步调试与断点功能**

在需要设置断点的行数左侧双击设置断点。

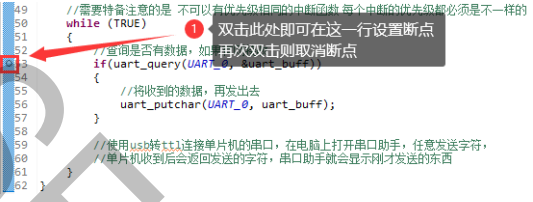


图5.5 设置断点示意图

进入Debug模式，点击工具栏Resume选项运行代码到断点处停止，此刻使用Step Into进行指令单步调试或使用Step Over进行代码单步调试。

### **5.2.1 变量查看功能**

在线调试时，在Variables选项卡可实时查看全局变量，通过Add Global Variables选择添加所需要查看的全局变量即可。

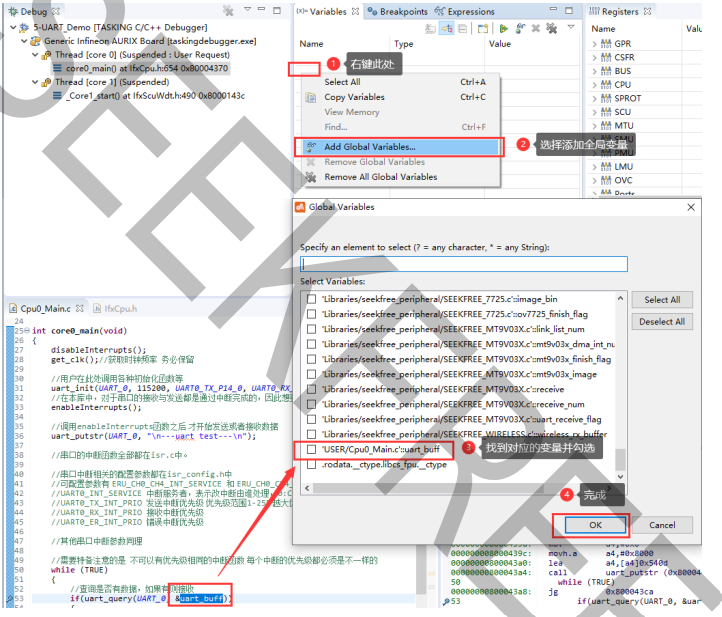


图5.6 查看全局变量示意图

**第六章 车模主要参数**

## 6.1智能车外形参数

经过改装后，智能汽车外形参数为：

车C

车长：312mm 车宽：180mm 车高：350mm

## 6.2智能车技术参数

智能汽车相关技术参数如表6.1所示：

|  |  |
| --- | --- |
| 项目 | 参数 |
| 车模轴距/轮距（mm） | 200 |
| 车模平均电流（匀速行驶）（mA） | 400 |
| 电容总量（uF） | 2400 |
| 传感器种类及个数 | 测速编码器 1个  MPU6050陀螺仪 1 个  MT9V032 1个 |
| 新增加伺服电机个数 | 0 |
| 赛道信息检测空间精度（mm） | 10 |
| 赛道信息检测频率（次/秒） | 100 |

表6. 1

**结论**

在这份报告中，我们主要从智能车的机械结构、硬件电路、控制算法、调试方法等介绍了我们准备比赛的整个过程，摄像头的寻迹程序大体上是继承了前辈们的精髓，但是也做出了我们自己的创新。创新之处大体为以下几点：

1. 我们选择单一的摄像头传感器寻迹。由于今年四轮摄像头组别的摄像头高度没有限制，这就导致了视野的无限可能，但是一辆车如果太多太复杂，会影响赛道的判断，所以为了获得更好的图像处理，我们不得不牺牲一部分的前瞻 来换取更清楚的视野，所以车辆远处的信息就无法通过摄像头来获取。于是我们选择了改变舵机PD和图像前瞻来进行快速判断。
2. 今年使用的车模是C车模，该车模后置两个个电机，依靠不同电机PWM差速来调整轮速差。为了提高车辆的拐弯性能，我们限制了差速只能减，不能增来帮忙控制小车，并且改变了前轮的倾角来提升转弯时车辆的抓地性能。

智能车是培养大学生综合动手能力的一个很好的、很成熟的一个竞赛，在比赛里我们不仅需要编程能力，还需要对车辆机械结构有着较深的理解，也需要熟悉各种传感器和电路的性能，你会发现不管是什么专业的学生都会在这个比赛中找到自己的定位。智能车比赛中最重要的是你需要足够多的创新能力和想法，追随习主席的指引，坚持创新驱动，和国家已经把创新放在了国家全局的核心位置，如今的竞赛一方面是在考验大学生的基本知识，更实在考验每一个参赛大学生的创新能力，所以只有创新才可以从全国大学生中脱颖而出。

在这个比赛中每个人都会学有所得，做有所获，通过这个比赛我也懂得了团队的重要性，学会分工合作，也找到了自己的擅长一面，同时也弥补了自己的不足之处。在长达几乎一年的准备过程中，我们团队三个人并没有一帆风顺，但我们不惧艰难，苦难与折磨只会让我们更加的强大，一年中的欢笑也会是我内心永远的花朵。

希望这篇技术文档会对未来参与智能车这项比赛的同学有所帮助，也祝愿他们可以在比赛中收获快乐

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] 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

[3] 阎石.数字电子技术基础[M].北京:高等教育出版社,1998.

[4] 雷霏霖,梁志毅.基于CMOS传感器 OV7620 采集系统设计[J].电子测量技术,2008,12(31):110-112.

[5] 谭浩强.C程序设计[M].北京:清华大学出版社,2005.

[6] 郭芳,曹桂琴.数据结构基础[M].大连：大连理工大学出版社，1994.

[7] 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.

[8] 胡寿松.自动控制原理(第六版)[M].科学出版社,2014.

[9] 张文春．汽车理论[M]．北京．机械工业出版社．2005

[10] YUAN Quan,ZHANG YunZhou,WU Hao,et al. Fuzzy Control Research In The Courses Of Smart Car[C]. Machine Vision and Human-Machine Interface (MVHI), Kaifeng,China, 2010: 764-767.

[11] 侯虹.采用模糊PID控制律的舵机系统设计[J].航空兵器，2006,2(1):7-9.

[12] 孙浩,程磊,黄卫华,等.基于HCS12的小车智能控制系统设计[J].单片机与嵌入式系统应用,2007,03(16):51-54.

[13] LU Zhenlin,LI Jingjiao,Zhang Minghui.Intelligent Control Research based on the Smart Car[C]. Advanced Computer Control (ICACC),Shenyang,2010:292-297.

**附录**

## **附录A：程序源码**

PID控制：

**void** **MotorPID\_right**(**void**)

{

**static** **int** error[3] = {0};

**long** P\_data = 0,I\_data = 0,D\_data = 0;

**long** delta\_output = 0;

error[0] = SpeedTarget\_right - Speed\_right;

P\_data = MotorP\_right \* (error[0] - error[1]);

I\_data = MotorI\_right \* error[0];

D\_data = MotorD\_right \* (error[0] - 2 \* error[1] + error[2]);

delta\_output = P\_data + I\_data + D\_data;

Motor\_output\_right += delta\_output;

**if**(Motor\_output\_right > 10000)

Motor\_output\_right = 10000;

**else** **if**(Motor\_output\_right < -10000)

Motor\_output\_right = -10000;

error[2] = error[1];

error[1] = error[0];

}

**void** **MotorPID\_left**(**void**)

{

**static** **int** error[3] = {0};

**long** P\_data = 0,I\_data = 0,D\_data = 0;

**long** delta\_output = 0;

error[0] = SpeedTarget\_left - Speed\_left;

P\_data = MotorP\_left \* (error[0] - error[1]);

I\_data = MotorI\_left \* error[0];

D\_data = MotorD\_left \* (error[0] - 2 \* error[1] + error[2]);

delta\_output = P\_data + I\_data + D\_data;

Motor\_output\_left += delta\_output;

**if**(Motor\_output\_left > 10000)

Motor\_output\_left = 10000;

**else** **if**(Motor\_output\_left < -10000)

Motor\_output\_left = -10000;

error[2] = error[1];

error[1] = error[0];

}

舵机PD：

**void** **Get\_angle**(**void**)

{

**static** **float** Bias,Last\_Bias;

Bias=80.0-(**float**)Middle\_last; //提取偏差：120改成160

Angle=Bias\*(kp\*1.0/10000)+(Bias-Last\_Bias)\*(kd\*1.0/10000); //PD控制

Last\_Bias=Bias; //保存上一次的偏差

}

子函数：

**void** **CrossRoad**()

{

**int** Right\_CrossRoad=158;

**for**(**int** i=158;i>110;i--){

**if**(mt9v03x\_image[LookForward-20][i]<otsu&&mt9v03x\_image[LookForward-20][i-1]>otsu){

Right\_CrossRoad=i;

**break**;

}

}

//lcd\_showfloat(0,4,Right\_CrossRoad,3,0);

**if**(CrossRoad\_flag==1&&CrossRoadFlag==0){

CrossRoadFlag=1;

}

**if**(CrossRoadFlag==1&&Right\_CrossRoad==158){

CrossRoadFlag=2;

}

**if**(CrossRoadFlag==2&&Right\_CrossRoad!=158){

CrossRoadFlag=3;

}

**if**(CrossRoadFlag==3&&Right\_CrossRoad==158){

CrossRoadFlag=4;

}

**if**(CrossRoadFlag==4){

**int** Max=Edge[0][126],MaxCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[0][column]>Max)

{

Max=Edge[0][column];

MaxCol=column;

}

}

**if**(MaxCol>50&&(Edge[0][MaxCol]>Edge[0][MaxCol+10])&&(Edge[0][MaxCol]>Edge[0][MaxCol-10])){//左边尖尖角

CrossRoadFlag=5;

}

}

**if**(CrossRoadFlag==5&&Right\_CrossRoad!=158){

CrossRoadFlag=6;

}

**if**(CrossRoadFlag==6&&Right\_CrossRoad==158){

CrossRoadFlag=0;

CrossRoad\_flag=0;

}

**switch**(CrossRoadFlag){

**case** 5:

Middle\_last=Right\_CrossRoad-30;

**break**;

**case** 6:

Middle\_last=Right\_CrossRoad-40;

**break**;

**default**:

**break**;

}

}

**int** **CirclePJudge**()

{

**int** countc=0, cross\_right=0, cross\_left=0;

**for**(**int** i=90;i>10;i--)

{

**if**((Edge[1][i+1]-Edge[1][i])>10/\*&& Edge[0][i-3]!=Edge[1][i-3]\*/&&

mt9v03x\_image[i+1][Edge[1][i+1]]>otsu && mt9v03x\_image[i][Edge[1][i+1]]>otsu && mt9v03x\_image[i-1][Edge[1][i+1]]>otsu)//右上进环点判断

{

countc=1;

**break**;

}

**if**((Edge[0][i]-Edge[0][i+1])>10/\*&& Edge[1][i-3]!=Edge[0][i-3]\*/&&

mt9v03x\_image[i+1][Edge[0][i+1]]>otsu && mt9v03x\_image[i][Edge[0][i+1]]>otsu && mt9v03x\_image[i-1][Edge[0][i+1]]>otsu)//左上进环点判断

{

countc=4;

**break**;

}

}

//防撞P

**if**(countc==1)

**for**(**int** i=120;i>2;i--)

**if**((Edge[1][i]-Edge[1][i+1])>10 && Edge[0][i-3]!=Edge[1][i-3])

{

countc=0;

**break**;

}

//防撞P

**if**(countc==4)

**for**(**int** i=120;i>2;i--)

**if**((Edge[0][i+1]-Edge[0][i])>10 && Edge[0][i-3]!=Edge[1][i-3])

{

countc=0;

**break**;

}

// if(abs(Edge[1][i+1]-Edge[1][i])>10&&(Edge[1][i+1]==158 || Edge[1][i]==158))//右线两个断点（十字判断）

// cross\_right++;

// if(abs(Edge[0][i+1]-Edge[0][i])>10&&(Edge[0][i+1]==2 || Edge[0][i]==2))//左线两个断点（十字判断）

// cross\_left++;

// if((Edge[1][i]-Edge[1][i+1])>10 && Edge[0][i-3]!=Edge[1][i-3])//右P判断

// countc=-2;

// if(countc==-2 && (Edge[1][i+1]-Edge[1][i])>10 && Edge[0][i-3]!=Edge[1][i-3])

// {

// countc=2;

// break;

// }

// lcd\_showfloat(0,3,cross\_right,2,0);

// if(cross\_left==2 && cross\_right==2)

// CrossRoad\_flag = 1;

//左线直道判断

**int** j;

**for**(j=110;j>10;j-=2)

{

**if**( (Edge[0][j+2]>Edge[0][j]) || (Edge[0][j] == 2) )

**break**;

}

**if**(**abs**((Edge[0][40]-Edge[0][70])-(Edge[0][90]-Edge[0][120]))>1)

j=120;//随便一个数，不等于10就行

**if**(j!=10 && countc==1)

countc=0;

//右线直道判断

**int** k;

**for**(k=110;k>10;k-=2)

{

**if**( (Edge[1][k+2]<Edge[1][k]) || (Edge[1][k] == 158) )

**break**;

}

**if**(**abs**((Edge[1][70]-Edge[1][40])-(Edge[1][120]-Edge[1][90]))>1)

k=120;//随便一个数，不等于10就行

**if**(k!=10 && countc==4)

countc=0;

// if(CircleFlag!=0)

// countc=0;

//lcd\_showfloat(0,0,countc,2,0);

**return** countc;

}

//圆环处理

**void** **CircleHandle\_2**(**int** countc)//右后

{

**int** i,j=0,k,BreakPoint;//j：尖尖点的纵坐标，k：补线底点

**float** Differ;

**if**(CircleFlag\_Right==2)

{

**for**(j=120;j>3;j--)//补线尝试:1.找2环尖尖点

{

**if**((Edge[1][j+1]-Edge[1][j])>10)

{

BreakPoint=Edge[1][j];

**break**;

}

**if**((Edge[0][j]-Edge[0][j+1])>10)

{

BreakPoint=Edge[0][j];

**break**;

}

}

//补线尝试:2.找底线起始点

**for**(k=2;k<20;k++)

{

**if**(mt9v03x\_image[127][k]>otsu)

**break**;

}

Differ = (**float**)(BreakPoint-k)/(**float**)(127-j);

**if**(gpio\_get(SW1)==1)

**for**(**int** f=127;f>j;f--)

{

lcd\_drawpoint((**int**)(k+Differ\*(127-f)),f,PURPLE);

}

}

**if**(countc==1&&CircleFlag==0){

CircleFlag=1;

CircleFlag\_Right=2;

}

**if**(CircleFlag==1&&j>=100/\*&&mt9v03x\_image[100][Edge[0][100]]>otsu\*/){

CircleFlag=3;

}

// if(CircleFlag==2&&mt9v03x\_image[120][Edge[0][120]]<otsu){

// CircleFlag=3;

// }

**if**(CircleFlag==3){

//记录一下环内角度

**static** **int** array[10]={0},A;

array[A] = (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2;;

**if**(A<10)

A++;

**else** **if**(A==10)

{

MiddleCircleRecord=0;

**for**(**int** a=0;a<10;a++)

MiddleCircleRecord += array[a];

MiddleCircleRecord = MiddleCircleRecord/10;

A=0;

}

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(0,3,MiddleCircleRecord,3,0);

**int** Max=Edge[0][126],MaxCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[0][column]>Max)

{

Max=Edge[0][column];

MaxCol=column;

}

}

**if**(MaxCol>30&&(Edge[0][MaxCol]>Edge[0][MaxCol+10])&&(Edge[0][MaxCol]>Edge[0][MaxCol-10])){//左边尖尖角

CircleFlag=4;

}

}

**if**(CircleFlag==4&&mt9v03x\_image[110][Edge[0][110]]>otsu){

CircleFlag=5;

}

**if**(CircleFlag==5&&mt9v03x\_image[110][Edge[0][110]]<otsu){

CircleFlag=-5;

}

**if**(CircleFlag==-5){

**int** PointR=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[1][column]-Edge[1][column-1])-(Edge[1][column-1]-Edge[1][column-2]);//二阶差分

**if**(Sod<-6)

{

PointR=column;

**break**;

}

}

**if**(PointR!=0)

CircleFlag=6;

}

**if**(CircleFlag==6&&mt9v03x\_image[110][Edge[1][110]]<otsu){

CircleFlag=0;

CircleFlag\_Right=0;

}

// int i;

// for(i=158;i>40;i--)

// {

// if(mt9v03x\_image[LookForward+30][i]>otsu)

// break;

// }

**switch**(CircleFlag){

**case** 1:

Middle\_last= ((**int**)(k+Differ\*(127-(LookForward)))+Edge[1][LookForward])/2;//i-40;

**if**(Velocity\_last==0){

Velocity\_last=Velocity;

Velocity=250;

}

//gpio\_set(BELL,1);

**break**;

// case 2:

// Middle\_last= i-45;

// break;

**case** 3:

Middle\_last= (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2; //在环内

gpio\_set(BELL,0);

**if**(Velocity\_last!=0){

Velocity=Velocity\_last;

Velocity\_last=0;

}

**break**;

**case** 4:

Middle\_last= MiddleCircleRecord;//Edge[1][LookForward+30]-50;

**break**;

**case** 5:

Middle\_last= MiddleCircleRecord;//Edge[1][LookForward+30]-50;

**break**;

**case** -5:

Middle\_last= Edge[0][LookForward+30]+50;

**break**;

**case** 6:

Middle\_last= Edge[0][LookForward+30]+50;

**break**;

**default**:

**break**;

}

}

**void** **CircleHandle\_4**(**int** countc)//左后

{

**int** i,j=0,k,BreakPoint;//j：尖尖点的纵坐标，k：补线底点

**float** Differ;

**if**(CircleFlag\_Left==2)

{

// for(i=2;i<120;i++)//极限救援

// {

// if(mt9v03x\_image[LookForward+30][i]>otsu)

// break;

// }

**for**(j=120;j>3;j--)//补线尝试:1.找2环尖尖点

{

**if**((Edge[0][j]-Edge[0][j+1])>10)

{

BreakPoint=Edge[0][j];

**break**;

}

**if**((Edge[1][j+1]-Edge[1][j])>10)

{

BreakPoint=Edge[1][j];

**break**;

}

}

//补线尝试:2.找底线起始点

**for**(k=157;k>100;k--)

{

**if**(mt9v03x\_image[127][k]>otsu)

**break**;

}

// for(int f=0;f<10;f++)

// if(j+f<126)

// lcd\_drawpoint(BreakPoint,j+f,PURPLE);

Differ = (**float**)(k-BreakPoint)/(**float**)(127-j);

**if**(gpio\_get(SW1)==1)

**for**(**int** f=127;f>j;f--)

{

lcd\_drawpoint((**int**)(k-Differ\*(127-f)),f,PURPLE);

}

}

**if**(countc==4&&CircleFlag==0){

CircleFlag=1;

CircleFlag\_Left=2;

}

**if**(CircleFlag==1&&j>=100/\*mt9v03x\_image[100][Edge[1][100]]>otsu&&Edge[1][100]==158\*/){

CircleFlag=3;

}

// if(CircleFlag==2&&mt9v03x\_image[120][Edge[1][120]]<otsu&&Edge[1][120]!=158){

// CircleFlag=3;

// }

**if**(CircleFlag==3){

//记录一下环内角度

**static** **int** array[10]={0},A;

array[A] = (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2;;

**if**(A<10)

A++;

**else** **if**(A==10)

{

MiddleCircleRecord=0;

**for**(**int** a=0;a<10;a++)

MiddleCircleRecord += array[a];

MiddleCircleRecord = MiddleCircleRecord/10;

A=0;

}

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(0,3,MiddleCircleRecord,3,0);

**int** Min=Edge[1][126],MinCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[1][column]<Min)

{

Min=Edge[1][column];

MinCol=column;

}

}

**if**(MinCol>30&&(Edge[1][MinCol]<Edge[1][MinCol+10])&&(Edge[1][MinCol]<Edge[1][MinCol-10])){//左边尖尖角

CircleFlag=4;

}

}

**if**(CircleFlag==4&&mt9v03x\_image[110][Edge[1][110]]>otsu){

CircleFlag=5;

}

**if**(CircleFlag==5&&mt9v03x\_image[110][Edge[1][110]]<otsu){

CircleFlag=-5;

}

**if**(CircleFlag==-5){

**int** PointL=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[0][column]-Edge[0][column-1])-(Edge[0][column-1]-Edge[0][column-2]);//二阶差分

**if**(Sod>6)

{

PointL=column;

**break**;

}

}

**if**(PointL!=0)

CircleFlag=6;

}

**if**(CircleFlag==6&&mt9v03x\_image[110][Edge[0][110]]<otsu){

CircleFlag=0;

CircleFlag\_Left=0;

}

**switch**(CircleFlag){

**case** 1:

Middle\_last= ((**int**)(k-Differ\*(127-(LookForward)))+Edge[0][LookForward])/2;//i+50;

**if**(Velocity\_last==0){

Velocity\_last=Velocity;

Velocity=250;

}

//gpio\_set(BELL,1);

**break**;

// case 2:

// Middle\_last= i+50;

// break;

**case** 3:

Middle\_last= (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2; //在环内

**if**(Velocity\_last!=0){

Velocity=Velocity\_last;

Velocity\_last=0;

}

**break**;

**case** 4:

Middle\_last= MiddleCircleRecord;//Edge[0][LookForward+30]+50;

**break**;

**case** 5:

Middle\_last= MiddleCircleRecord;//Edge[0][LookForward+30]+50;

**break**;

**case** -5:

Middle\_last= Edge[1][LookForward+30]-50;

**break**;

**case** 6:

Middle\_last= Edge[1][LookForward+30]-50;

**break**;

**default**:

**break**;

}

}

**void** **CircleHandle\_1**(**int** mod,**int** countc)//右先

{

**int** i,j=0,k,BreakPoint;//j：尖尖点的纵坐标，k：补线底点

**float** Differ;

**if**(CircleFlag\_Right==1 && CircleFlag>=3)

{

**for**(j=120;j>3;j--)//补线尝试:1.找2环尖尖点

{

**if**((Edge[1][j+1]-Edge[1][j])>10)

{

BreakPoint=Edge[1][j];

**break**;

}

**if**((Edge[0][j]-Edge[0][j+1])>10)

{

BreakPoint=Edge[0][j];

**break**;

}

}

//补线尝试:2.找底线起始点

**for**(k=3;k<40;k++)

{

**if**(mt9v03x\_image[127][k]>otsu)

**break**;

}

// for(int f=0;f<10;f++)

// if(j+f<126)

// lcd\_drawpoint(BreakPoint,j+f,PURPLE);

Differ = (**float**)(BreakPoint-k)/(**float**)(127-j);

**if**(gpio\_get(SW1)==1)

**for**(**int** f=127;f>j;f--)

{

lcd\_drawpoint((**int**)(k+Differ\*(127-f)),f,PURPLE);

}

}

**if**((CircleFlag==0)&&mod==2&&CircleFlag\_Right==0){ //环岛状态1 右缺边 沿右线

**int** PointR=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[1][column]-Edge[1][column-1])-(Edge[1][column-1]-Edge[1][column-2]);//二阶差分

**if**(Sod>6)

{

PointR=column;

**break**;

}

}

// for(int column=2;column<40;column++){

// if(mt9v03x\_image[column][3]>otsu){

// threeRoad\_Cir=1;

// }

// }

//左线直道判断

**int** j;

**for**(j=124;j>10;j-=2)

{

**if**( (Edge[0][j+2]>Edge[0][j]) || (Edge[0][j] == 2) )

**break**;

}

**if**(**abs**((Edge[0][40]-Edge[0][70])-(Edge[0][90]-Edge[0][120]))>1)

j=120;//随便一个数，不等于10就行

// &&mt9v03x\_image[PointR-2][Edge[1][PointR]+3]>=otsu

**if**(Edge[1][PointR]<155&&j==10&&PointR!=0&&Edge[1][PointR-10]==158&&mt9v03x\_image[PointR+15][Edge[1][PointR]]>otsu){ //右下角拐角为锐角尖尖

CircleFlag=1;

CircleFlag\_Right=1;

}

}

**if**(CircleFlag==1&&mt9v03x\_image[120][Edge[1][120]]>otsu){ //右下角全白 沿左线

CircleFlag=2;

}

**if**(CircleFlag==2&&mt9v03x\_image[120][Edge[1][120]]<otsu){ //右下角出现环岛中心黑圆 沿右线

CircleFlag=3;

}

**if**(CircleFlag==3){ //左下角全白 沿右线Edge[0][126]

**if**(countc==1){

CircleFlag=4;

}

**int** Max=Edge[0][126],MaxCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[0][column]>Max)

{

Max=Edge[0][column];

MaxCol=column;

}

}

//早发现 70

**if**(MaxCol>90&&(Edge[0][MaxCol]>Edge[0][MaxCol+5])&&(Edge[0][MaxCol]>Edge[0][MaxCol-5])&&CircleFlag==3){//左边尖尖角

CircleFlag=0; //左边尖尖，圆环清零

//P出环

PFlag\_Left=1;

}

}

**if**(PFlag\_Left==1){

**int** num\_L=0;

**int** num\_R=0;

**for**(**int** column=120;column>60;column-=2){

**if**(Edge[1][column]<158){

num\_L++;

}

**if**(Edge[0][column]>2){

num\_R++;

}

**if**(num\_R>23&&num\_L>23){

PFlag\_Left=2;

**if**(Velocity\_last!=0){

Velocity = Velocity\_last;

Velocity\_last=0;

}

**break**;

}

}

}

**if**(PFlag\_Left==2&&mt9v03x\_image[120][Edge[0][120]]<otsu){

PFlag\_Left=0;//P结束，恢复正常

CircleFlag\_Right=0;

}

**if**(CircleFlag==4&&j>100/\*&&mt9v03x\_image[120][Edge[0][120]]<otsu\*/){ //左下角出现黑（正式入环） 沿左线Edge[0][126]

CircleFlag=5;

}

**if**(CircleFlag==5&&mod!=1){ //不是左缺边（排除入环尖角误判）沿右线

//记录一下环内角度

**static** **int** array[10]={0},A;

array[A] = (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2;;

**if**(A<10)

A++;

**else** **if**(A==10)

{

MiddleCircleRecord=0;

**for**(**int** a=0;a<10;a++){

MiddleCircleRecord += array[a];

}

MiddleCircleRecord = MiddleCircleRecord/10;

A=0;

}

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(0,3,MiddleCircleRecord,3,0);

//记录结束

**int** Max=Edge[0][126],MaxCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[0][column]>Max)

{

Max=Edge[0][column];

MaxCol=column;

}

}

**if**(MaxCol>30&&(Edge[0][MaxCol]>Edge[0][MaxCol+10])&&(Edge[0][MaxCol]>Edge[0][MaxCol-10]))//左边尖尖角

CircleFlag=6; //出环，左边有钝角尖尖

}

**if**(CircleFlag==6&&mt9v03x\_image[110][Edge[0][110]]>otsu) //左下全白 沿右线

CircleFlag=7;

**if**(CircleFlag==7&&mt9v03x\_image[110][Edge[0][110]]<otsu) //左下出现黑色边界（左边赛道） 沿左线

CircleFlag=8;

**if**(CircleFlag==8) //左下出现黑色边界（左边赛道） 沿左线

{

**int** PointR=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[1][column]-Edge[1][column-1])-(Edge[1][column-1]-Edge[1][column-2]);//二阶差分

**if**(Sod<-6)

{

PointR=column;

**break**;

}

}

**if**(PointR!=0)

CircleFlag=-8;

}

**if**(CircleFlag==-8&&mt9v03x\_image[110][Edge[1][110]]<otsu){ //右下出现黑色赛道（环岛通过，准备P元素

CircleFlag=0;

CircleFlag\_Right=0;

}

**switch**(PFlag\_Left){

**case** 1:

//早发现 30

Middle\_last=30;

**break**;

**case** 2:

Middle\_last=Edge[1][LookForward+30]-40;

**break**;

**default**:

**break**;

}

**int** Side\_Offset=40;

**switch**(CircleFlag){

**case** 1:

Middle\_last= Edge[0][LookForward+30]+Side\_Offset;

**break**;

**case** 2:

Middle\_last= Edge[0][LookForward+30]+Side\_Offset;

**break**;

**case** 3:

Middle\_last= Edge[0][LookForward+30]+Side\_Offset;

**if**(Velocity\_last==0){

Velocity\_last = Velocity;

Velocity = 250;

}

**break**;

**case** 4:

Middle\_last= ((**int**)(k+Differ\*(127-(LookForward)))+Edge[1][LookForward])/2;//i-Side\_Offset;

**break**;

**case** 5:

Middle\_last= (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2; //在环内

**if**(Velocity\_last!=0){

Velocity = Velocity\_last;

Velocity\_last=0;

}

//gpio\_set(BELL,0);

**break**;

**case** 6:

Middle\_last= MiddleCircleRecord;//Edge[1][LookForward+30]-50;

**break**;

**case** 7:

Middle\_last= MiddleCircleRecord;//Edge[1][LookForward+30]-50;

**break**;

**case** 8:

Middle\_last= Edge[0][LookForward+30]+Side\_Offset;

**break**;

**case** -8:

Middle\_last= Edge[0][LookForward+30]+Side\_Offset;

**break**;

**default**:

**break**;

}

}

**void** **CircleHandle\_3**(**int** mod,**int** countc)//左先

{

**int** i,j=0,k,BreakPoint;//j：尖尖点的纵坐标，k：补线底点

**float** Differ;

**if**(CircleFlag\_Left==1 && CircleFlag>=3)

{

// for(i=2;i<120;i++)//极限救援

// {

// if(mt9v03x\_image[LookForward+30][i]>otsu)

// break;

// }

**for**(j=120;j>3;j--)//补线尝试:1.找2环尖尖点

{

**if**((Edge[0][j]-Edge[0][j+1])>10)

{

BreakPoint=Edge[0][j];

**break**;

}

**if**((Edge[1][j+1]-Edge[1][j])>10)

{

BreakPoint=Edge[1][j];

**break**;

}

}

//补线尝试:2.找底线起始点

**for**(k=157;k>100;k--)

{

**if**(mt9v03x\_image[127][k]>otsu)

**break**;

}

// for(int f=0;f<10;f++)

// if(j+f<126)

// lcd\_drawpoint(BreakPoint,j+f,PURPLE);

Differ = (**float**)(k-BreakPoint)/(**float**)(127-j);

**if**(gpio\_get(SW1)==1)

**for**(**int** f=127;f>j;f--)

{

lcd\_drawpoint((**int**)(k-Differ\*(127-f)),f,PURPLE);

}

}

**if**((CircleFlag==0)&&mod==1&&CircleFlag\_Left==0){

**int** PointR=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[0][column]-Edge[0][column-1])-(Edge[0][column-1]-Edge[0][column-2]);//二阶差分

**if**(Sod<-6)

{

PointR=column;

**break**;

}

}

// for(int column=2;column<40;column++){

// if(mt9v03x\_image[column][157]>otsu){

// threeRoad\_Cir=1;

// }

// }

//右线直道判断

**int** k;

**for**(k=124;k>10;k-=2)

{

**if**( (Edge[1][k+2]<Edge[1][k]) || (Edge[1][k] == 158) )

**break**;

}

**if**(**abs**((Edge[1][70]-Edge[1][40])-(Edge[1][120]-Edge[1][90]))>1)

k=120;//随便一个数，不等于10就行

// mt9v03x\_image[PointR-2][Edge[0][PointR-3]]>=otsu

**if**(Edge[0][PointR]>5&&k==10&&PointR!=0&&Edge[0][PointR-10]==2&&mt9v03x\_image[PointR+15][Edge[0][PointR]]>otsu){ //右下角拐角为锐角尖尖

CircleFlag=1;

CircleFlag\_Left=1;

}

}

**if**(CircleFlag==1&&mt9v03x\_image[120][Edge[0][120]]>otsu){ //右下角全白 沿左线

CircleFlag=2;

}

**if**(CircleFlag==2&&mt9v03x\_image[120][Edge[0][120]]<otsu){ //右下角出现环岛中心黑圆 沿右线

CircleFlag=3;

}

**if**(CircleFlag==3){ //左下角全白 沿右线Edge[0][126]

**if**(countc==4){

CircleFlag=4;

}

**int** Min=Edge[1][126],MinCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[1][column]<Min)

{

Min=Edge[1][column];

MinCol=column;

}

}

//早发现 70

**if**(MinCol>90&&(Edge[1][MinCol]<Edge[1][MinCol+5])&&(Edge[1][MinCol]<Edge[1][MinCol-5])&&CircleFlag==3){//you边尖尖角

CircleFlag=0; //左边尖尖，圆环清零

//P出环

PFlag\_Right=1;

}

}

**if**(PFlag\_Right==1){

**int** num\_L=0;

**int** num\_R=0;

**for**(**int** column=120;column>60;column-=2){

**if**(Edge[0][column]>2){

num\_L++;

}

**if**(Edge[1][column]<158){

num\_R++;

}

**if**(num\_L>23&&num\_R>23){

PFlag\_Right=2;

**if**(Velocity\_last!=0){

Velocity = Velocity\_last;

Velocity\_last=0;

}

**break**;

}

}

}

**if**(PFlag\_Right==2&&mt9v03x\_image[120][Edge[1][120]]<otsu){

PFlag\_Right=0;//P结束，恢复正常

CircleFlag\_Left=0;

}

**if**(CircleFlag==4&&j>100/\*mt9v03x\_image[120][Edge[1][120]]<otsu\*/){ //左下角出现黑（正式入环） 沿左线Edge[0][126]

CircleFlag=5;

}

**if**(CircleFlag==5){ //不是左缺边（排除入环尖角误判）沿右线

//记录一下环内角度

**static** **int** array[10]={0},A;

array[A] = (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2;

**if**(A<10)

A++;

**else** **if**(A==10)

{

MiddleCircleRecord=0;

**for**(**int** a=0;a<10;a++)

MiddleCircleRecord += array[a];

MiddleCircleRecord = MiddleCircleRecord/10;

A=0;

}

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(0,3,MiddleCircleRecord,3,0);

//记录结束

**int** Min=Edge[1][126],MinCol=126;

**for**(**int** column=126;column>40;column--)

{

**if**(Edge[1][column]<Min)

{

Min=Edge[1][column];

MinCol=column;

}

}

**if**(MinCol>50&&(Edge[1][MinCol]<Edge[1][MinCol+10])&&(Edge[1][MinCol]<Edge[1][MinCol-10]))//左边尖尖角

CircleFlag=6; //出环，左边有钝角尖尖

}

**if**(CircleFlag==6&&mt9v03x\_image[110][Edge[1][110]]>otsu) //左下全白 沿右线

CircleFlag=7;

**if**(CircleFlag==7&&mt9v03x\_image[110][Edge[1][110]]<otsu) //左下出现黑色边界（左边赛道） 沿左线

CircleFlag=8;

**if**(CircleFlag==8){

**int** PointL=0;

**for**(**int** column=126;column>10;column--)

{

**int** Sod=(Edge[0][column]-Edge[0][column-1])-(Edge[0][column-1]-Edge[0][column-2]);//二阶差分

**if**(Sod>6)

{

PointL=column;

**break**;

}

}

**if**(PointL!=0)

CircleFlag=-8;

}

**if**(CircleFlag==-8&&mt9v03x\_image[110][Edge[0][110]]<otsu){ //右下出现黑色赛道（环岛通过，准备P元素

CircleFlag=0;

CircleFlag\_Left=0;

}

**switch**(PFlag\_Right){

**case** 1:

//早发现 130

Middle\_last=130;

**break**;

**case** 2:

Middle\_last=Edge[0][LookForward+30]-50;

**break**;

**default**:

**break**;

}

**int** Side\_Offset=50;

**switch**(CircleFlag){

**case** 1:

Middle\_last= Edge[1][LookForward+30]-Side\_Offset;

//gpio\_set(BELL,1);

**break**;

**case** 2:

Middle\_last= Edge[1][LookForward+30]-Side\_Offset;

**break**;

**case** 3:

Middle\_last= Edge[1][LookForward+30]-Side\_Offset;

**if**(Velocity\_last==0){

Velocity\_last=Velocity;

Velocity=250;

}

**break**;

**case** 4:

Middle\_last= ((**int**)(k-Differ\*(127-(LookForward)))+Edge[0][LookForward])/2;

**break**;

**case** 5:

//gpio\_set(BELL,0);

Middle\_last= (Edge[1][LookForward+20]+Edge[0][LookForward+20])/2; //在环内

**if**(Velocity\_last!=0){

Velocity=Velocity\_last;

Velocity\_last=0;

}

**break**;

**case** 6:

Middle\_last= MiddleCircleRecord;//Edge[0][LookForward+30]+50;

**break**;

**case** 7:

Middle\_last= MiddleCircleRecord;//Edge[0][LookForward+30]+50;

**break**;

**case** 8:

Middle\_last= Edge[1][LookForward+30]-Side\_Offset;

**break**;

**case** -8:

Middle\_last= Edge[1][LookForward+30]-Side\_Offset;

**break**;

**default**:

**break**;

}

}

//二值化显示

**void** **BinShow**()

{

**int** i,j;

//otsu=otsuThreshold(mt9v03x\_image[0],MT9V03X\_W,MT9V03X\_H);//大津法求阈值

**for**(i=1;i<128;i++)//宽

{

**for**(j=1;j<160;j++)//长

{

**if**(mt9v03x\_image[i][j]<otsu)//阈值

lcd\_drawpoint(j,i,BLACK);

}

}

lcd\_clear(WHITE);

}

//显示三线

**void** **ShowEdge**(**int** mod)

{

**int** column;

**for**(column=126;column>2;column--)//扫描范围126 - 20，可调

{

lcd\_drawpoint(Edge[0][column],column,RED);

lcd\_drawpoint(Edge[1][column],column,GREEN);

lcd\_drawpoint((Edge[0][column]+Edge[1][column])/2,column,GREEN);

}

}

//斑马线

**void** **Zebra**(**int** column)

{

**int** numdown=0,numup=0;

**if**(Zebras==1)

{

**for**(**int** i=158;i>3;i--) //从最右往最左扫描黑白跳变点

{

**if**(mt9v03x\_image[column+20][i]>otsu&&mt9v03x\_image[column+20][i-1]<otsu)//白变黑跳变点

**continue**;

**if**(mt9v03x\_image[column+20][i]<otsu&&mt9v03x\_image[column+20][i-1]>otsu)//黑变白跳变点

numdown++;

}

}

**if**(Zebras==0)

{

**for**(**int** i=158;i>3;i--) //从最右往最左扫描黑白跳变点

{

**if**(mt9v03x\_image[column][i]>otsu&&mt9v03x\_image[column][i-1]<otsu)//白变黑跳变点

**continue**;

**if**(mt9v03x\_image[column][i]<otsu&&mt9v03x\_image[column][i-1]>otsu)//黑变白跳变点

numdown++;

}

}

**if**(numdown>5) //如果有至少6个，证明有斑马线，近处斑马线

Zebra\_flag=1;

**else**

numdown=0;

// if(numup>5) //如果有至少4个，证明有斑马线，远处斑马线

// Zebra\_flag=2;

// else

// numup=0;

//适用于左出库

**int** LeftI;

**for**(LeftI=1;LeftI<158;LeftI++)

{

**if**(mt9v03x\_image[LookForward+30][LeftI]>otsu)

**break**;

}

**int** RightI;

**for**(RightI=158;RightI>1;RightI--)

{

**if**(mt9v03x\_image[LookForward+30][RightI]>otsu)

**break**;

}

//第一次路过车库直接过

**if**(Zebra\_flag==1&&Zebras==0)

{

CircleFlag\_Right=0;

CircleFlag\_Left=0;

CircleFlag=0;

**if**(gpio\_get(SW2)==1)//左出库

Middle\_last=LeftI+40;

**else** **if**(gpio\_get(SW2)==0)//右出库

Middle\_last=RightI-40;

gpio\_set(BELL,1);

**int** i;

**if**(gpio\_get(SW2)==1)//适用于左出库，第一次路过斑马线完成操作

{

**for**(i=126;i>40;i--)

**if**(Edge[1][i]==158)

**break**;

}

**if**(gpio\_get(SW2)==0)//适用于右出库，第一次路过斑马线完成操作

{

**for**(i=126;i>40;i--)

**if**(Edge[0][i]==2)

**break**;

}

**if**(i==40){

Zebras=1;

Zebra\_flag=0;

gpio\_set(BELL,0);

}

}

//第二次路过车库进库

**if**(Zebra\_flag==1&&Zebras==1)

{

//Middle\_last=30;//左入库

CircleFlag\_Right=0;

CircleFlag\_Left=0;

CircleFlag=0;

Velocity = 150;

// if(gpio\_get(SW2)==1)

// Middle\_last=RightI-40;

// else if(gpio\_get(SW2)==0)

// Middle\_last=LeftI+40;

**if**(/\*(Edge[0][110]==2||Edge[1][110]==158) &&\*/ Time\_Zebra==-1){

systick\_start(*STM0*);

Time\_Zebra=systick\_getval\_ms(*STM0*);

gpio\_set(BELL,1);

}

**if**(Time\_Zebra!=-1){

**if**(gpio\_get(SW2)==1)

Middle\_last=30;

**else** **if**(gpio\_get(SW2)==0)

Middle\_last=130;

**if**(systick\_getval\_ms(*STM0*)>Time\_Zebra+200){

Zebra\_flag=0;

Zebras=2;

// Time\_Zebra=-1;

gpio\_set(BELL,0);

}

}

}

**if**(Zebras==2)

{

SpeedTarget\_right=1;

SpeedTarget\_left=1;

Velocity=1;

**if**(gpio\_get(SW2)==1)

Middle\_last=30;

**else** **if**(gpio\_get(SW2)==0)

Middle\_last=130;

// while(1){

// pwm\_duty(ATOM0\_CH0\_P21\_2,1);

// pwm\_duty(ATOM0\_CH6\_P23\_1,1);

// }

}

}

**void** **PLeft**()

{

//右边界最大值点

**int** Min=Edge[1][126],MinCol=126;

**for**(**int** column=126;column>10;column--)

{

**if**(Edge[1][column]<Min)

{

Min=Edge[1][column];

MinCol=column;

}

}

**if**(Pleft==0)

Pleft=1;

**if**(Pleft==1&&mt9v03x\_image[120][Edge[0][120]]>otsu){

Pleft=-1;

}

**if**(Pleft==-1&&mt9v03x\_image[120][Edge[0][120]]<otsu){

Pleft=-2;

}

**if**(Pleft==-2&&Zebra\_flag==0&&MinCol>80&&MinCol<130&&(Edge[1][MinCol]<Edge[1][MinCol+5])&&

(Edge[1][MinCol]<Edge[1][MinCol-5]))//右边尖尖角

Pleft=2;

**if**(Pleft==2&&Zebra\_flag==0)

{

**int** num=0;

**for**(**int** column=110;column>60;column-=2){

**if**(Edge[0][column]>2){

num++;

}

**if**(num>23){

Pleft=3;

**break**;

}

}

}

**switch**(Pleft){

**case** 1:

Middle\_last=Edge[1][LookForward-40]-35;

**break**;

**case** -1:

Middle\_last=Edge[1][LookForward-40]-35;

**break**;

**case** -2:

Middle\_last=Edge[1][LookForward-40]-35;

**break**;

**case** 2:

Middle\_last=130;//Edge[1][LookForward]-30

**break**;

**case** 3:

Middle\_last=Edge[0][LookForward]+50;

**break**;

}

}

//车库

**void** **Garage**()

{

**int** GarageDir=0;

**int** Garage\_num=0;

**int** Middle\_Garage=80;

**for**(**int** column=126; column>2;column--)

**if**(mt9v03x\_image[column][Middle\_Garage]>otsu&&mt9v03x\_image[column-1][Middle\_Garage]<otsu)

Garage\_num++;

**if**(Garage\_num>4)

Garage\_Flag=1;

**else**

Garage\_Flag=0;

**if**(Garage\_Flag==1){

**if**(gpio\_get(SW2)==0)

GarageDir=1; //右出库

**else**

GarageDir=2; //左出库

// if(Time\_Garage==-1){

systick\_start(*STM0*);

Time\_Garage=systick\_getval\_ms(*STM0*);

// }

}

**if**(systick\_getval\_ms(*STM0*)>Time\_Garage+400&&(GarageDir==2||GarageDir==1)){

Garage\_Flag=0;

GarageDir=0;

//Time\_Garage=-1;

}

// GarageDir=Garage\_Flag;

**switch**(GarageDir){

**case** 1:

Middle\_last=130;//右出库

**break**;

**case** 2:

Middle\_last=30;//左出库

**break**;

**default**:

**break**;

}

}

//三岔

**void** **ThreeRoad**(){

**int** max=Edge[0][126],maxcolumn=126,i,iup,MaxCord=126;

**int** MinUp=2,Mincolumn=158;

**for**(i=126;i>50;i--)//左边尖尖

{

**if**(**abs**(Edge[0][i-1]-Edge[0][i])>10)

**break**;

**if**(Edge[0][i]>max)

{

max=Edge[0][i];

maxcolumn=i;

MaxCord=i;

}

}

//(max,i)左边尖尖坐标

// for(int l=0;l<15;l++)

// lcd\_drawpoint(Edge[0][maxcolumn]+l,maxcolumn,RED);

**if**(max<80&&maxcolumn>40&&Edge[0][maxcolumn]>=Edge[0][maxcolumn-5]&&Edge[0][maxcolumn]>=Edge[0][maxcolumn+5]&&i==50)

maxcolumn=888;

**int** NoLeft;

**for**(NoLeft=126;NoLeft>60;NoLeft--)//左边看不到了

{

**if**(Edge[0][NoLeft]!=2)

**break**;

}

**int** min=Edge[1][126],mincolumn=126,j;

**for**(j=126;j>50;j--)//右边尖尖

{

**if**(**abs**(Edge[1][j-1]-Edge[1][j])>10)

**break**;

**if**(Edge[1][j]<min)

{

min=Edge[1][j];

mincolumn=j;

}

}

// for(int l=0;l<15;l++)

// lcd\_drawpoint(Edge[1][mincolumn]-l,mincolumn,RED);

**if**(min>80&&mincolumn>40&&Edge[1][mincolumn]<=Edge[1][mincolumn-5]&&Edge[1][mincolumn]<=Edge[1][mincolumn+5]&&j==50)

mincolumn=888;

**int** NoRight;

**for**(NoRight=126;NoRight>60;NoRight--)//右边看不到了

{

**if**(Edge[1][NoRight]!=158)

**break**;

}

// lcd\_showfloat(0,1,maxcolumn,4,0);

// lcd\_showfloat(0,2,mincolumn,4,0);

**if**((maxcolumn==888&&mincolumn==888)||(maxcolumn==888&&NoRight==60)||(mincolumn==888&&NoLeft==60))//第一次点判断，第二次点判断

{

**int** hash = 888;

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(0,5,hash,4,0);

**int** UpEdge[200];

**int** UpNum=2;

**for**(**int** column=158;column>2;column--)

{

UpNum=2;

**while**(mt9v03x\_image[UpNum][column]<=otsu&&UpNum<125)

UpNum++;

UpEdge[column]=UpNum;//上边线存储数组

}

// for(int i=158;i>2;i--)

// {

// lcd\_drawpoint(i,UpEdge[i],YELLOW);

// }

**for**(iup=158;iup>2;iup--)

{

**if**(UpEdge[iup]>MinUp&&UpEdge[iup]!=125)

{

MinUp=UpEdge[iup];

Mincolumn=iup;

}

}

//(i,MinUp)顶点坐标

**if**(gpio\_get(SW1)==1)

lcd\_showfloat(50,5,Mincolumn,4,0);

**if**(Mincolumn>30 && Mincolumn<130 && UpEdge[Mincolumn]<63/\* && UpEdge[Mincolumn]>30 \*/&&

UpEdge[Mincolumn]>UpEdge[Mincolumn-10] && UpEdge[Mincolumn-10]>UpEdge[Mincolumn-20] &&

UpEdge[Mincolumn]>UpEdge[Mincolumn+10] && UpEdge[Mincolumn+10]>UpEdge[Mincolumn+20]) //终极点判断

{

ThreeRoad\_flag = 33;

}

}

**if**(ThreeRoad\_flag==33)

{

//再算一遍上边的

**int** UpEdge[200];

**int** UpNum=2;

**for**(**int** column=158;column>2;column--)

{

UpNum=2;

**while**(mt9v03x\_image[UpNum][column]<=otsu&&UpNum<125)

UpNum++;

UpEdge[column]=UpNum;//上边线存储数组

}

// for(int i=158;i>2;i--)

// {

// lcd\_drawpoint(i,UpEdge[i],YELLOW);

// }

**for**(iup=158;iup>2;iup--)

{

**if**(UpEdge[iup]>MinUp&&UpEdge[iup]!=125)

{

MinUp=UpEdge[iup];

Mincolumn=iup;

}

}

**float** Differ;

Differ = (**float**)(Mincolumn-max)/(**float**)(MaxCord-MinUp);

//lcd\_showfloat(50,0,MinUp,3,0);

**if**(gpio\_get(SW1)==1)

**for**(**int** k=MaxCord;k>MinUp;k--)

lcd\_drawpoint((**int**)(max+Differ\*(MaxCord-k)),k,PURPLE);

**if**(MaxCord>90)

Middle\_last=((**int**)(max+Differ\*(MaxCord-90))+Edge[1][90])/2;

**if**(MinUp>100)

ThreeRoad\_flag=0;

}

}

//找边线

**void** **FindEdge**()

{

**int** column,i,j,Middle;

//左边底点i

**for**(i=80;i>1;i--)

**if**(mt9v03x\_image[127][i]<otsu&&mt9v03x\_image[127][i-1]<otsu&&mt9v03x\_image[127][i-2]<otsu)//白-黑跳变点,3个黑点确定

**break**;

//右边底点j

**for**(j=80;j<159;j++)

**if**(mt9v03x\_image[127][j]<otsu&&mt9v03x\_image[127][j+1]<otsu&&mt9v03x\_image[127][j+2]<otsu)//白-黑跳变点

**break**;

//赛道整体靠边处理

**if**(mt9v03x\_image[127][80]<otsu)

{

//靠右侧

**if**(mt9v03x\_image[127][158]>otsu&&mt9v03x\_image[127][157]>otsu&&mt9v03x\_image[127][156]>otsu)

{

j=158;

**for**(i=158;i>1;i--)

**if**(mt9v03x\_image[127][i]<otsu)

**break**;

}

//靠左侧

**if**(mt9v03x\_image[127][2]>otsu&&mt9v03x\_image[127][3]>otsu&&mt9v03x\_image[127][4]>otsu)

{

i=2;

**for**(j=2;j<159;j++)

**if**(mt9v03x\_image[127][j]<otsu)

**break**;

}

//有时候元素混杂，需要根据上一状态判断

**if**(mt9v03x\_image[127][2]>otsu&&mt9v03x\_image[127][3]>otsu&&mt9v03x\_image[127][4]>otsu&&

mt9v03x\_image[127][158]>otsu&&mt9v03x\_image[127][157]>otsu&&mt9v03x\_image[127][156]>otsu)

{

**if**(Middle\_last>80)

{

j=158;

**for**(i=158;i>1;i--)

**if**(mt9v03x\_image[127][i]<otsu)

**break**;

}

**if**(Middle\_last<80)

{

i=2;

**for**(j=2;j<159;j++)

**if**(mt9v03x\_image[127][j]<otsu)

**break**;

}

}

}

//中线底点

**int** MiddleBottom=(i+j)/2;//坐标位置(MiddleBottom,127)

**int** MiddleL=MiddleBottom,MiddleR=MiddleBottom;

**for**(column=126;column>1;column--)//扫描范围126 - 1，可调

{

**while**((mt9v03x\_image[column][MiddleL]>=otsu||mt9v03x\_image[column][MiddleL-1]>=otsu||mt9v03x\_image[column][MiddleL-2]>=otsu)&&MiddleL>2)

MiddleL--;

Edge[0][column]=MiddleL;//左边线数组存储

//lcd\_drawpoint(MiddleL,column,GREEN);

**while**((mt9v03x\_image[column][MiddleR]>=otsu||mt9v03x\_image[column][MiddleR+1]>=otsu||mt9v03x\_image[column][MiddleR+2]>=otsu)&&MiddleR<158)

MiddleR++;

Edge[1][column]=MiddleR;//右边线数组存储

//lcd\_drawpoint(MiddleR,column,GREEN);

Middle=(MiddleL+MiddleR)/2;//中点拟合(未存储)

MiddleL=Middle;MiddleR=Middle;

}

}

//缺边判断

uint8 **EdgeJudge**()

{

uint8 mod=0;//1：左边缺边，2：右边缺边，3：双边缺边，0：不缺边

**int** column,Sod,FlagL=0,FlagR=0,MisJudge=0;

//左边线缺边判断

**for**(column=126;column>30;column--)

{

Sod=(Edge[0][column]-Edge[0][column-1])-(Edge[0][column-1]-Edge[0][column-2]);//二阶差分

**if**(Sod<-5)

{

mod=1;

FlagL=1;

}

}

//右边线缺边判断

**for**(column=126;column>30;column--)

{

Sod=(Edge[1][column]-Edge[1][column-1])-(Edge[1][column-1]-Edge[1][column-2]);//二阶差分

**if**(Sod>5)

{

mod=2;

FlagR=1;

}

}

//双边缺边判断

**if**(FlagL==1&&FlagR==1)

mod=3;

**if**(mt9v03x\_image[127][80]<otsu)

mod=0;

//弯道时误判

**for**(column=126;column>40;column--)

**if**(Edge[0][column]==Edge[1][column])

MisJudge++;

**if**(MisJudge>5)

mod=0;

**return** mod;

}

//确定左边下拐点

uint8 **TurnPointL**()

{

uint8 Point=126;

**int** column,Fod;

//左边下拐点

**for**(column=126;column>38;column--)

{

Fod=**abs**(Edge[0][column]-Edge[0][column-1]);//一阶差分

**if**(Fod>3)

{

Point=column;

**break**;

}

}

**return** Point;

}

//确定右边下拐点

uint8 **TurnPointR**()

{

uint8 Point=126;

**int** column,Fod;

//右边下拐点

**for**(column=126;column>38;column--)

{

Fod=**abs**(Edge[1][column]-Edge[1][column-1]);//一阶差分

**if**(Fod>3)

{

Point=column;

**break**;

}

}

**return** Point;

}

//常规补线(单边缺边，直接延伸)

**void** **RepairLine**(**int** Point,**int** mod)//1:左边 , 2:右边

{

**float** ConAdd;

**if**(mod==1&&Point<=123)//左边缺边补线

{

**float** Sub=(**float**)Edge[0][Point+2]-(**float**)Edge[0][126];

**float** Slope=Sub/(126-(Point+2));//拟合斜率，每行的X变化值

ConAdd=Edge[0][Point+1];//累加起点

**for**(**int** column=Point+1;column>30;column--)

{

ConAdd+=Slope;

Edge[0][column]=(**int**)ConAdd;

**if**(Edge[0][column]<3)

Edge[0][column]=3;

**if**(Edge[0][column]>157)

Edge[0][column]=157;

}

}

**if**(mod==2&&Point<=123)//右边缺边补线

{

**float** Sub=(**float**)Edge[1][Point+2]-(**float**)Edge[1][126];

**float** Slope=Sub/(126-(Point+2));//拟合斜率，每行的X变化值

ConAdd=Edge[1][Point+1];//累加起点

**for**(**int** column=Point+1;column>30;column--)

{

ConAdd+=Slope;

Edge[1][column]=(**int**)ConAdd;

**if**(Edge[1][column]>157)

Edge[1][column]=157;

**if**(Edge[1][column]<3)

Edge[1][column]=3;

}

}

}

// 全局大津法

uint8 **otsuThreshold**(uint8 \*image, uint16 col, uint16 row)

{

**#define** GrayScale 256

uint16 width = col;

uint16 height = row;

**int** pixelCount[GrayScale];

**float** pixelPro[GrayScale];

**int** i, j, pixelSum = width \* height;

uint8 threshold = 0;

uint8\* data = image; //指向像素数据的指针

**for** (i = 0; i < GrayScale; i++)

{

pixelCount[i] = 0;

pixelPro[i] = 0;

}

//统计灰度级中每个像素在整幅图像中的个数

**for** (i = 0; i < height; i++)

{

**for** (j = 0; j < width; j++)

{

pixelCount[(**int**)data[i \* width + j]]++; //将像素值作为计数数组的下标

}

}

//计算每个像素在整幅图像中的比例

**float** maxPro = 0.0;

**for** (i = 0; i < GrayScale; i++)

{

pixelPro[i] = (**float**)pixelCount[i] / pixelSum;

**if** (pixelPro[i] > maxPro)

{

maxPro = pixelPro[i];

}

}

//遍历灰度级[0,255]

**float** w0, w1, u0tmp, u1tmp, u0, u1, u, deltaTmp, deltaMax = 0;

**for** (i = 0; i < GrayScale; i++) // i作为阈值

{

w0 = w1 = u0tmp = u1tmp = u0 = u1 = u = deltaTmp = 0;

**for** (j = 0; j < GrayScale; j++)

{

**if** (j <= i) //背景部分

{

w0 += pixelPro[j];

u0tmp += j \* pixelPro[j];

}

**else** //前景部分

{

w1 += pixelPro[j];

u1tmp += j \* pixelPro[j];

}

}

u0 = u0tmp / w0;

u1 = u1tmp / w1;

u = u0tmp + u1tmp;

deltaTmp = w0 \* **pow**((u0 - u), 2) + w1 \* **pow**((u1 - u), 2);

**if** (deltaTmp > deltaMax)

{

deltaMax = deltaTmp;

threshold = (uint8)i;

}

}

**return** threshold;

}

## **附录B：原理图总图**

