第四届“飞思卡尔”杯全国大学生
智能汽车竞赛

技 术 报 告

附件 B 起跑线识别算法的研究[注]

学　　校：西北师范大学

队伍名称：瞬之队

参赛队员：陈有生 孙越 汪国强

带队教师：摆玉龙　　严春满

关于技术报告和研究论文使用授权说明

本人完全了解第四届“飞思卡尔”杯全国大学生智能汽车邀请赛关保留、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，比赛组委会和飞思卡尔半导体公司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

竞速车模的电子控制系统设计

摘要：本文详细介绍了以第四届全国飞思卡尔智能车大赛为背景的竞速车模自动循线控制系统方案。文中主要介绍了竞速车模的整体框架、硬件设计、摄像头图像采集模块、光电编码器速度采集模块、执行模块电路设计、电源管理模块以及系统软件设计。在硬件设计方面，结合大赛选用芯片的要求，自行设计实现了系统的电路板，实践证明，该电路板较好的集成了智能车所需电路，使得整车的集成度提高，性能更加可靠。软件系统以Freescale16位单片MC9S12XS128作为系统控制处理器，采用CMOS数字摄像头OV6620获取实时赛道信息，通过边缘检测方法提取赛道黑线，求出小车与黑线间的位置偏差，采用PD控制算法对舵机转向进行控制。通过光电编码器YZ30D4S-2NA-200实时获取小车速度，采用P控制策略形成速度闭环控制。经过大赛实践证明，该套方案能够使智能汽车稳定并且快速运行。

关键字：飞思卡尔单片机 ；摄像头信息处理； PID控制

Racing Car Model Electronic Control System Design

Abstracts: Automatic line patrol control system prepared for the 4th national contest of Smart car was described in details in this paper. The paper introduces the overall framework, hardware design, camera image acquisition module, photo electrical encoder speed acquisition module, implementation module circuit design, power management module and system software design of the racing car. As for the hardware design, the racing car system, with single-chip MC9S12XS128 as its system control processor, uses CMOS digital camera to obtain the real-time track information, extracts the black line on the contest lane with the edge detection method, and calculates position deviation between the car and the black line, distinguishes the different shape of the lane, then PD control algorithm for steering engine steering control was given in this paper. Through the photo electrical encoder named YZ30D4S-2NA-200 to attain the real-time car speed, and uses incremental P strategy to achieve the speed closed-loop control. It is proved that the control system can achieve automatic line patrol control, so as to make the racing car driving stably and quickly.

Key words: Freescale Single-chip；Camera image processing；PID control

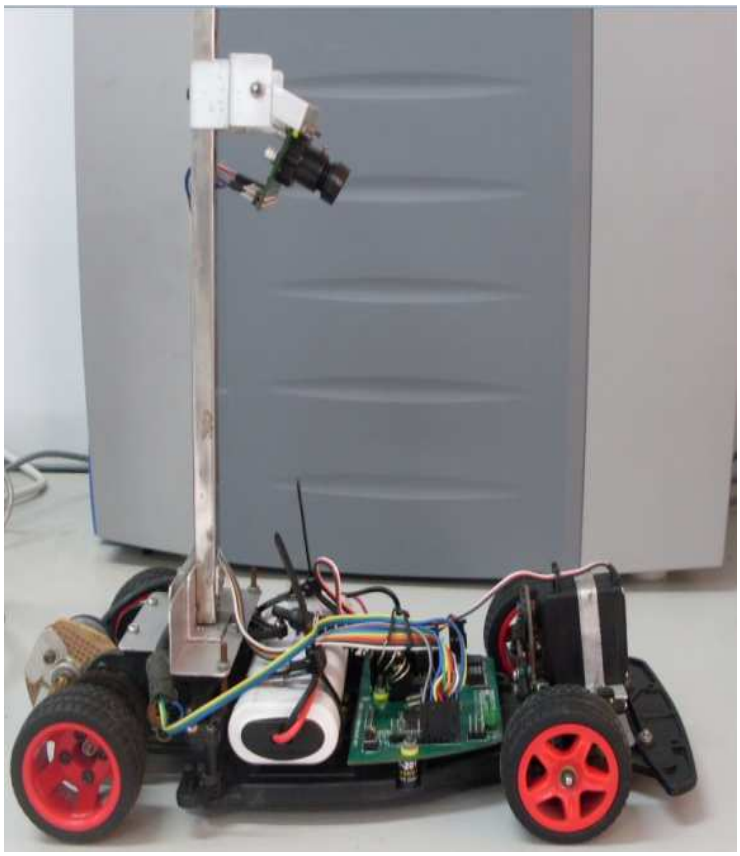
图目录

创 新 点

首先，在硬件设计方面，**结合大赛选用芯片的要求，自行设计实现了系统的电路板。**实践证明，该电路板较好的集成了智能车所需电路，使得整车的集成度提高，性能更加可靠。另外我们**自行设计的电机驱动电路**，使得小车的驱动能力更强，能够实现小车瞬间的加速和减速。

其次，在软件方面，我们对**路径做了具体的划分**，使得小车行驶的路线尽可能的取直线效果，最终实现了蛇形道直冲，大 s 内切，使车整体的行驶路径最大限度的减小。另外，针对**起跑线的识别**做了进一步的研究，使得小车在检测起跑线方面效率得到了极大的提高。

最后，在车的整体方面，**尽可能的降低了车的重心**，同时对摄像头的安装位置调节的较靠后使得小车的盲区很小，对差速，前轮，后轮都做了较大的调整，使得车整体效果得到了较大的提升。



第一章 引言

1.1 Freescale 车模比赛简介

教育部为了加强大学生实践、创新能力和团队精神的培养，在已举办全国大学生数学建模、电子设计、机械设计、结构设计等4项竞赛的基础上，委托教育部高等学校自动化专业教学指导分委会主办每年一度的全国大学生智能汽车竞赛（Freescale车模比赛）[1]。智能汽车竞赛所使用的车模是一款带有差速器的后轮驱动模型车，由组委会统一提供。赛道路面用专用白色基板制作，在初赛阶段时，跑道所占面积在5米*7米左右,决赛阶段时跑道面积可以增大。跑道包括普通赛道和窄道区两部分。普通赛道宽度不小于60厘米 ，窄道区的宽度不小于45厘米 ； 跑道表面为白色，中心有连续黑线作为引导线，黑线宽 25mm 。铺设赛道地板颜色不作要求，它和赛道之间可以但不一定有颜色差别 ；跑道最小曲率半径不小于 50 厘米 ； 跑道可以交叉，交叉角为 90 ° ；赛道直线部分可以有坡度在 15 度之内的坡面道路，包括上坡与下坡道路 ；在驶入窄道区和驶出窄道区时，赛道上有标志。该标志距离窄道区 25 厘米 在进入和驶离窄道区有两种标志(如图1.1所示)：

（1） 黑色正三角形，位于赛道中心。边长25厘米。

（2） 赛道突起，颜色白色，厚度0.5厘米，宽度3厘米。

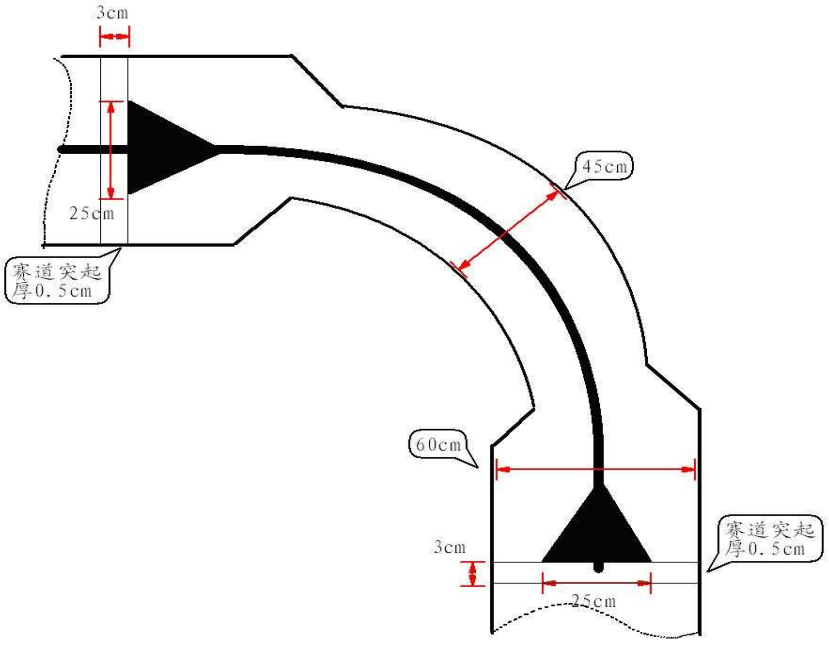


图 1.1 窄道区示意图

赛车可以根据上述标志判断是否进入或驶离窄道区。 赛道有一个长为1米的出发区，如图1.11所示，计时起始点两边分别有一个长度10厘米黑色计时起始线，赛车前端通过起始线作为比赛计时开始或者结束时刻。

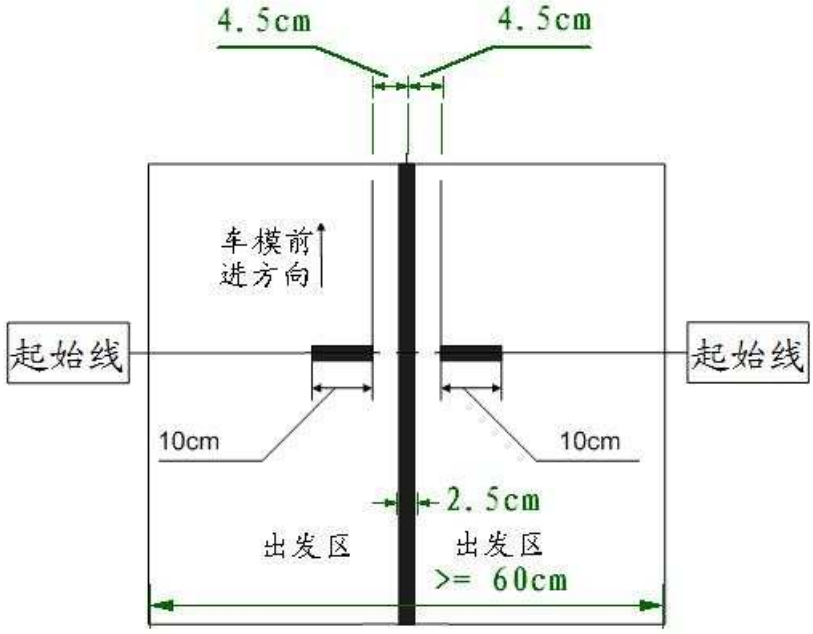


图 1.2 起始线示意图

1.2 发展现状

智能汽车，是一种集环境感知、规划决策、自动行驶等功能于一体的综合系统，集中地应用到自动控制、模式识别、传感器技术、汽车电子、电气、计算机、机械等多个学科，是典型的高新技术综合体，具有重要的军用及民用价值。

目前，智能车领域的研究已经能够在具有一定标记的道路上为司机提供辅助驾驶系统甚至实现无人驾驶。这些智能车的设计通常依靠特定道路标记完成识别，通过推理判断模仿人工驾驶进行操作。通常，智能车接受辅助定位系统提供的信息完成路径规划，如由GPS等提供的地图，交通拥堵状况，道路条件等信息。

1.3 章节安排

本文内容的安排如下所示：

第一章 引言 本章主要介绍了 Freescale 车模竞赛的基本情况，智能汽车的发展状

况。

第二章 系统整体框架 本章对系统硬件模块方案和软件控制方法进行了选择与论证。

第三章 机械结构的安装与调整 本章对机械结构的安装与改进，各个模块的安装技巧作了详细的介绍。

第四章 硬件电路的设计与实现 本章主要介绍了自行设计的基于飞思卡尔单片机的最小系统板的设计、电源模块、摄像头模块和速度传感器模块的设计与实现。

第五章 软件系统设计与实现 本章软件系统各模块的设计思路作了详细的介绍。特别介绍了图像处理中的各种技巧、PID 控制策略的应用和起跑线识别算法的设计等问题

第六章 开发工具及其调试 本章对开发工具与调试方法作了简单介绍。

结论 对整个参赛过程中的经验与教训作了总结。

第二章 系统整体框架

2.1 系统框架

硬件电路是整个系统的基础，下面是我们的硬件电路原框图：

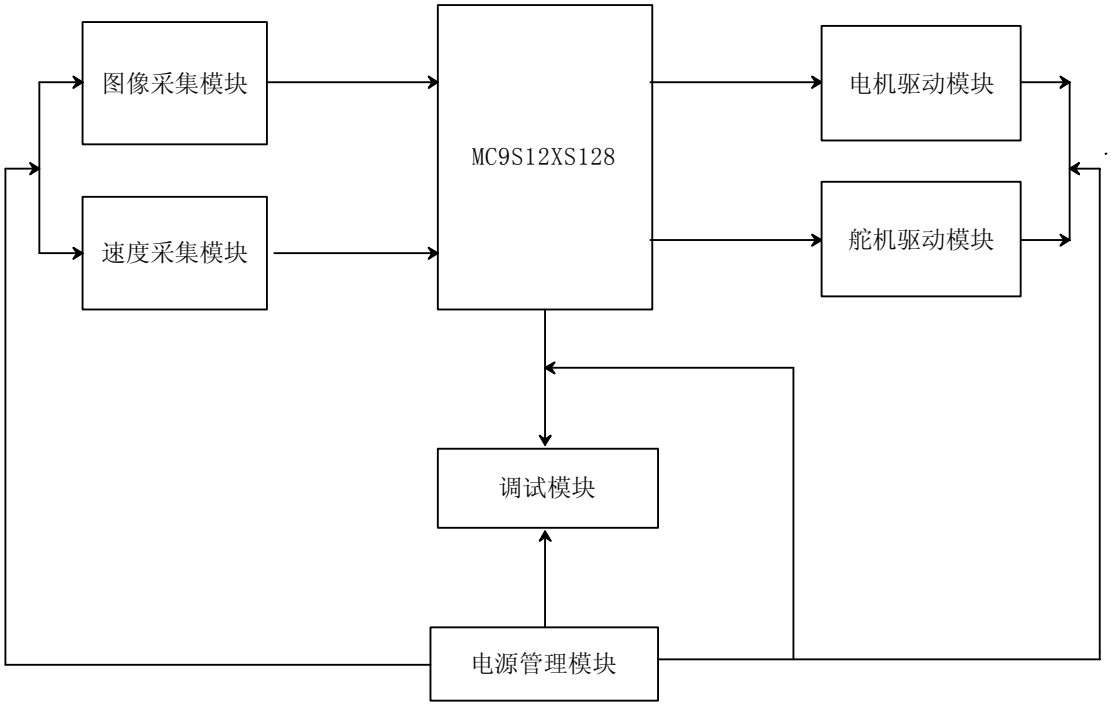


图2.1 整体框架

2.1.0 主要模块：

1) MC9S12XS128处理器：该模块是整个系统的核心和大脑部件，所有的信息都要由它处理，结合该处理器，我们最终自行设计了最小系统板。
2) 采集模块：采集模块主要有图像采集和速度采集，经过比较我们最终采用了CMOS的OV6620摄像头对图像进行采集。在速度采集方面，我们选用了YZ30D4S-2NA-200光电编码器。
3) 驱动模块：主要有电机驱动和舵机驱动。我们最初选用的电机驱动是MC33886并联的方式，经过测试，发现这种方案驱动能力不强，最终我们自行搭建了由场效应管组成的H桥，效果很好。为了保险起见，我们使用一个稳压芯片给舵机供电。
4) 电源模块：主要用到的有5V和6V，5V芯片选择的是LM2940,6V芯片选择的是

LM2941。摄像头和单片机，采用分别供电的方式供电。

5) 调试模块：调试模块主要用到BDM，并且为了简化电路，在最小系统板上没有设计串口的接口，只是把线引了出来，做了独立的串口调试模块。

2.1.1工作过程：

系统将图像采集模块，速度采集模块采集到的路况信息、速度信息和小车状态信息等送到整个系统的核心部件MC9S12XS128进行分析处理，然后发出相应控制命令输出到执行模块的电机和舵机执行适宜的速度和转向动作，配以BDM和串口调试，进而实现整个系统的闭环反馈控制。

2.2 系统软件

系统的软件流程图如图2.2

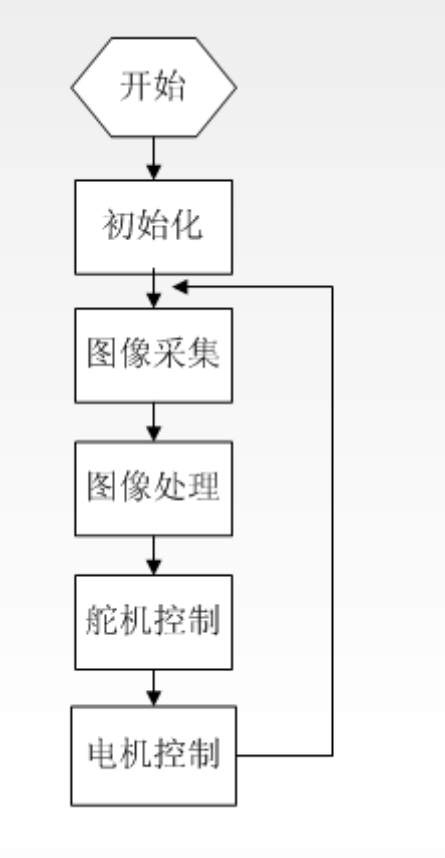


图2.2 系统流程图

首先对系统各部分进行初始化，初始化完毕后对图像进行采集，图像的采集和处理是实时进行的，采集后对图形处理，提取出黑线的位置，然后依照黑线的位置对舵机和电机进行控制。然后不断的重复，就可以让小车顺利的在赛道上行驶。

2.3 方案简介

在本次比赛中，组委会提供了3种单片机可供选择MC9S12XS128,MC9S12DG128和9S08AW60 。我们选择了总线频率较高的MC9S12XS128作为主控芯片，并且自己制作了最小系统板。图像采集我们经过对比最终选择了CMOS的OV6620作为图像采集传感器。将图像采集来后，为了减小干扰，首先我们对整幅进行了二值化，然后利用跟踪边缘算法对黑线进行提取，为了使提取的黑线更加准确我们对提取的黑线进行了中值滤波和限幅滤波。最终使黑线的变化更加平稳。提取出黑线后，我们采用了模糊控制与PD控制相结合的方式对舵机进行控制，让小S直接冲过去，大S尽量内切，最大难度的发卡弯沿线通过，对速度的控制我们参考了去年北科大的算法构建了一个二次函数。然后，小车的速度根据前方的路况自动调整。

在硬件方面，我们为了使电路更加简化，自己制作了最小系统板，使得单片机，电源，电机驱动等模块集中到了一块最小系统板上。

第三章　机械结构安装与调整

3.1 机械结构的安装

机械安装是整个系统中的基础，决定智能汽车在运行过程中的稳定性，因此对机械结构的合理安装就显得非常重要。

3.1.0 摄像头的安装

摄像头安装主要考虑的问题有： 固定摄像头的材料，摄像头的安装位置和摄像头的安装高度。固定材料我们选择了硬度较好而且质量轻的铝合金材料。安装位置刚开始我们把摄像头安装到舵机后面一点，但是发现那样车的盲区很大，为了减小盲区我们就把摄像头固定到了电机前方。同时摄像头的高度过高，重心就提高同时摄像头看的数据就不是太清晰，高度过低的话视野太小。我们进过多次反复的实验最终确定了摄像头镜头离地面的高度为24.2cm 。摄像头的固定如图3.1和3.2所示。

图3.1摄像头的固定

图3.2摄像头与车模的固定

3.1.1 舵机的安装

舵机起着转向的作用，其安装非常重要。如果安装不好就会出现转向不足或者转向左右不对称的问题。车模原始的固定方式是力臂一边长一边短，这样的安装使得车转向左右不对称。为此我们改变舵机的安装方式。将其立起来安装(如图3.3)，安装的高度也是需要恰当选择的。若安装的太高，灵敏度是提高了，但提供的总扭矩会减小。这个高度我们在实际的调试过程中不断调整，最终的高度如图3.3所示：需要注意的是，在舵机的转向连杆和前轮连接固定处，需给万用节留一定空隙，跟有利于转向。

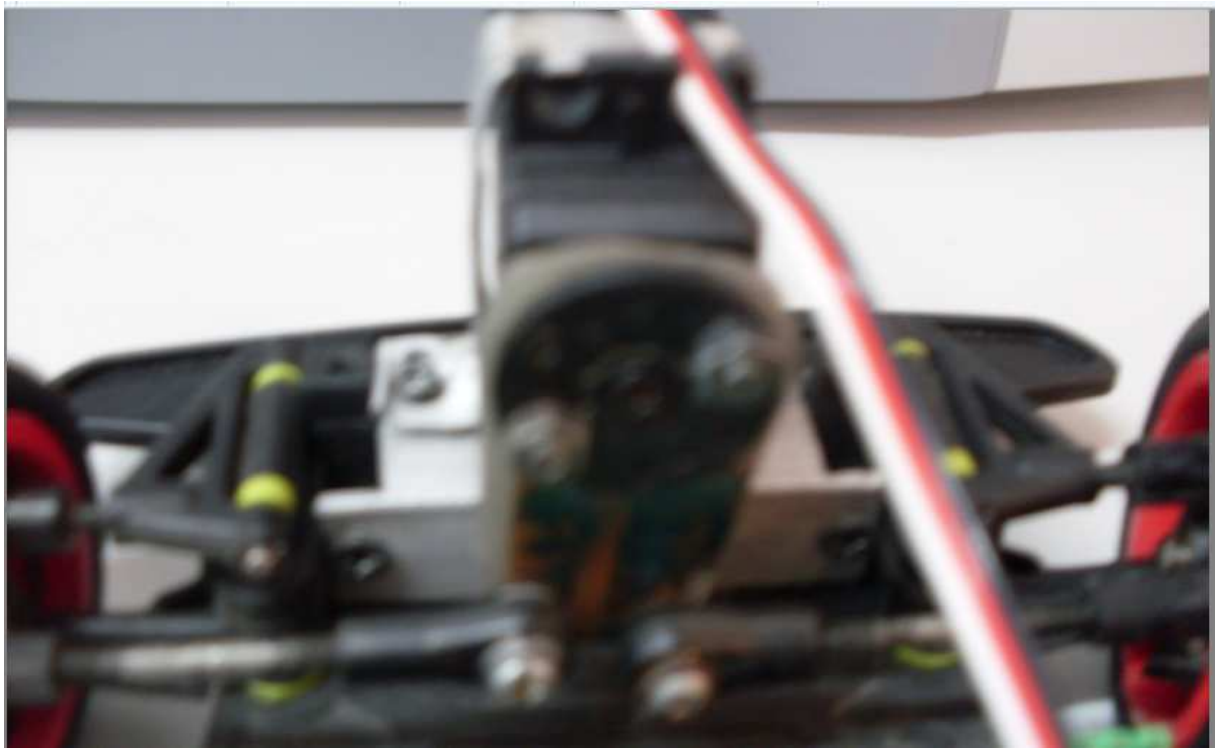


图3.3舵机安装

3.1.2 光电编码器的安装

光电编码器安装主要考虑的问题是与齿轮的咬合，太紧会使电机转动吃力并且会发出很大的噪声，太松有时候会丢齿。因此最好使得安装的编码器松紧程度能够调整最好。安装方式如图3.4

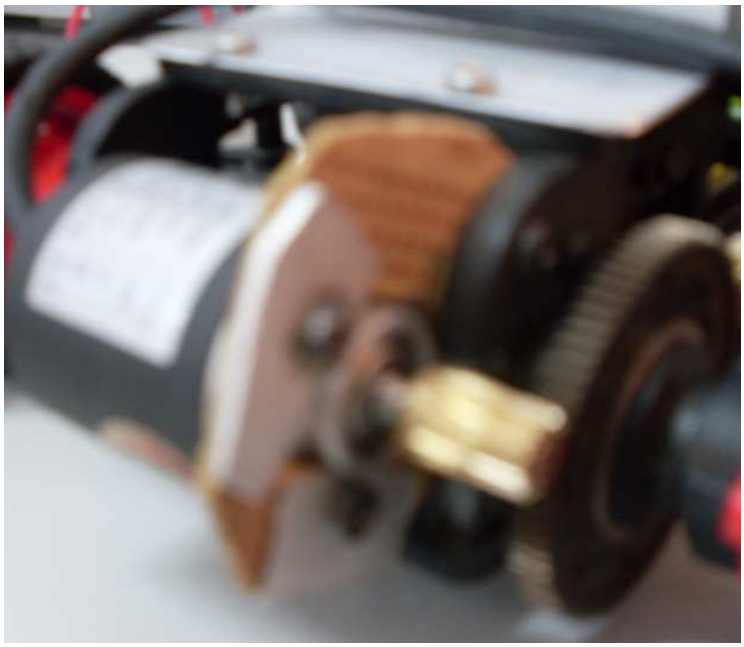


图3.4编码器安装

3.1.3 系统板的安装

最小系统板是我们依据车身情况定做的，所以安装容易。如图3.5

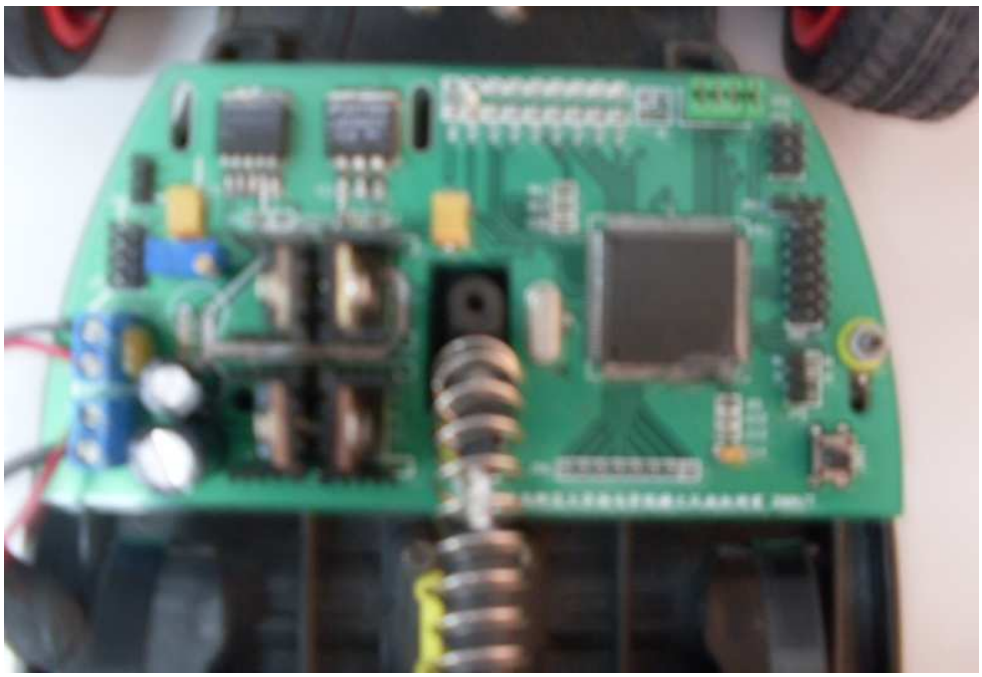


图3.5系统板的安装

3.1.3 总体布局

经过各个部件的安装车的总体布局如图3.6和3.7

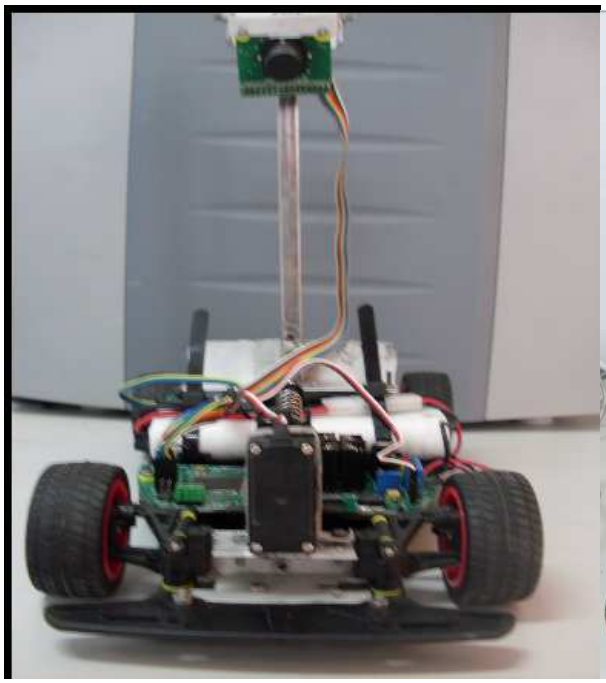


图3.6正视图

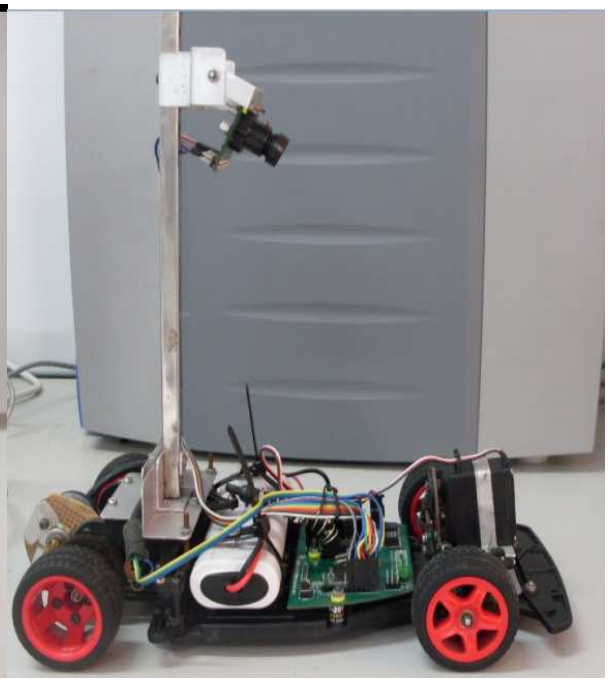


图3.7侧视图

3.2 机械调整

车模的前轮可调节的参数比较丰富，前轮定位的作用是保障汽车直线行驶的稳定性，转向轻便和减少轮胎的磨损。前轮是转向轮，由主销内倾、主销后倾、前轮外倾和前轮前束等4个项目。在刚开始调试的时候中，我们没有重视这几个参数的调节，后来发现这几个参数对赛车直线的稳定性和弯道灵巧性都有很重要的影响。

3.2.1 主销内倾

主销内倾是指主销装在前轴略向内倾斜的角度，它的作用是使前轮自动回正。角度越大前轮自动回正的作用就越强烈，但转向时也越费力，轮胎磨损增大；反之，角度越小前轮自动回正的作用就越弱。

3.2.2 主销后倾

主销后倾是指主销装在前轴，上端略向后倾斜的角度。它使车辆转弯时产生的离心力所形成的力矩方向与车轮偏转方向相反，迫使车轮偏转后自动恢复到原来的中间位置上。由此，主销后倾角越大，车速越高，前轮稳定性也愈好。我们将车模控制主销后倾的黄色垫片改为 3：1（前 3 后 1），使其倾角为负 2° ～3° 。这样则可以减小回正力矩的作用，使转向更为灵活，但也会使回正比原来稍慢。

3.2.3 前轮外倾角

前轮外倾角对汽车的转弯性能有直接影响，它的作用是提高前轮的转向安全性和转向操纵的轻便性。如果前面两个轮子呈现“V”字形则称正倾角，呈现“八”字则称负倾角。如果车轮垂直地面一旦满载就易产生变形，可能引起车轮上部向内倾侧，导致车轮联接件损坏。

3.3.4 前轮前束

前轮前束是前轮前端向内倾斜的程度，当两轮的前端距离小后端距离大时为内八字，前端距离大后端距离小为外八字。由于前轮外倾使轮子滚动时类似与圆锥滚动，从而导致两侧车轮向外滚开。但由于拉杆的作用使车轮不可能向外滚开，车轮会出现边滚变向内划的现象，从而增加了轮胎的磨损。前轮外八字与前轮外倾搭配，一方面可以抵消前轮外倾的负作用，另一方面由于赛车前进时车轮由于惯性自然的向内倾斜，外八字可以抵消其向内倾斜的趋势。外八字还可以使转向时靠近弯道内侧的轮胎比靠近弯道外侧的轮胎的转向程度更大，则使内轮胎比外轮胎的转弯半径小，有利与转向。

3.3.4 车体重心

由于车运行时，重心越低越好，而且重心集中在中后位置最利于转弯。于是我们利用调整底盘高度来调节车的重心。但是由于赛道存在凸起和坡度，重心太低车可能被卡住。为此我们把车底盘高度最终调整为离地面 1cm 高。

第四章　硬件电路的设计与实现

硬件电路主要包括：电源模块，驱动模块及调试模块。电源模块主要包括单片机电源，编码器电源，摄像头电源，舵机电源等等。驱动模块主要包括电机驱动和舵机驱动。调试模块主要包括 BDM 下载和串口的设计。

4.1　最小系统板的设计

我们采用了组委会提供的MC9S12XS128芯片作为主控芯片，参考了组委会提供的系统板的原理图，自行设计了最小系统板和外围器件的电路，同时为了尽可能的减小板子的质量和大小，我们没有将串口设计到最小系统板上，而是另外做了一块USB转串口的电路板。原理图如图4.1和4.2

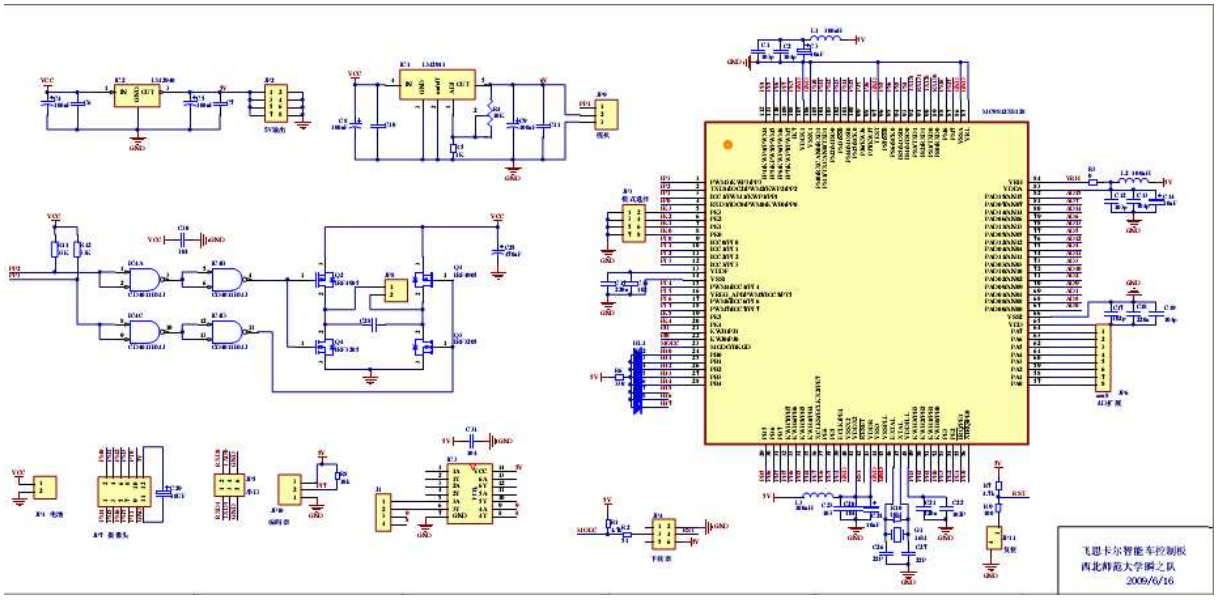


图4.1最小系统板原理图

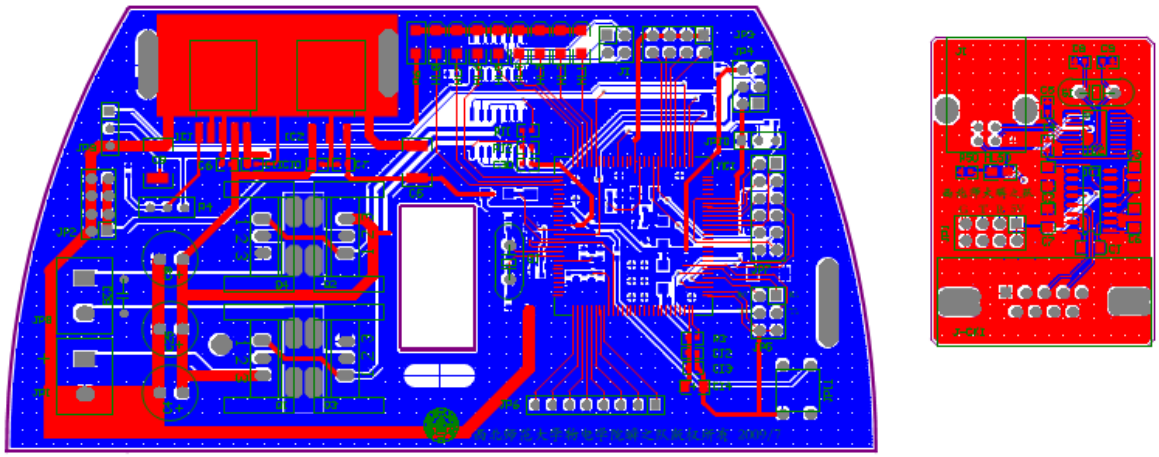


图4.2 PCB图

引脚功能介绍：

PORTB:LED状态指示

PORTM:摄像头数据采集口

PORTK:模式选择口

PP1:舵机控制

PP2:电机控制口

PP3：电机控制口

TXD0:串口通讯

RXD0: 串口通讯

4.2 电源模块

为了保证各个部件的正常工作，电源的供给是十分重要的，需要对配发的标准车模用蓄电池进行电压调节。单片机系统、摄像头、车速传感器电路等各个电路的工作电压不同，需要想办法来使得电压满足各自的要求，一种方法是利用升压或降压的芯片来达到它们的要求，另一种方法是利用双电源供电的方法，来实现各模块的不同需要，由于电路模块较多，该方案中仍需要升压或降压芯片。实际应用中，我们确定采用升压降压芯片等来实现对各个模块的供电要求。而且，在电路设计中，考虑到由于电机驱动所引起的电源不稳定，在电源输入端，各芯片电源引脚都加入滤波电路。

如图4.1所示电源模块的组成

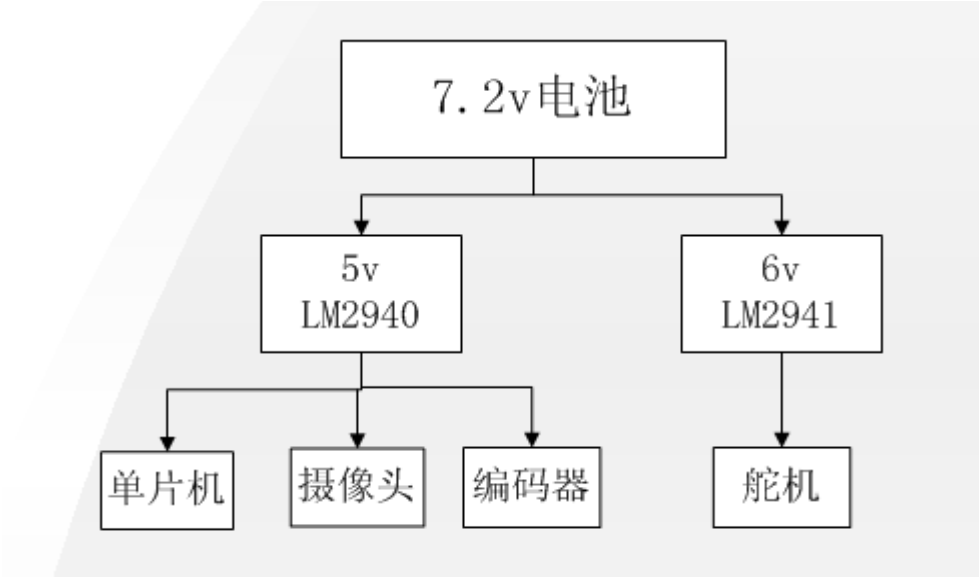


图4.3 电源模块组成框图

1）5v电源

市场上常用的5v芯片有LM2940，LM7805，LM2575，LM2596。其中LM2940和LM7805转换效率低（40%）输出波纹小,而且稳定，对于电源要求比较高的元件适合。LM2575和LM2596转换效率高（75%~80%）输出波纹大，可能会让单片机出现重启。所以我们选择前者而LM2940比LM7805压差小，而且更加稳定因此我们最终选择LM2940作为5v稳压芯片。原理图如图4.4所示

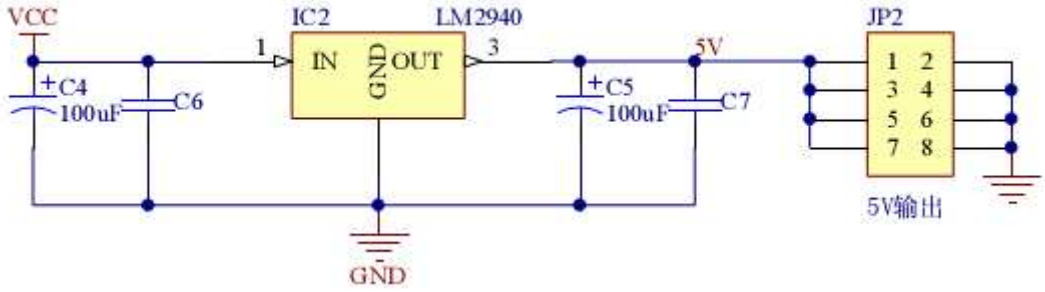


图4.4　5v稳压电路原理图

同时我们将摄像头编码器等5v电源直接做到了电路板上，这样连接方便，干扰小而且连线也少电路显得整洁。

2）6v电源

舵机的响应速度与其电源电压有关。因此，为了获得更快的响应速度，舵机的供电采用其工作上限电压+6V，舵机的工作电压为4-6v因此，为了稳定起见我们给舵机也做了稳压电路，器件选择的是LM2941.原理图如图4.5

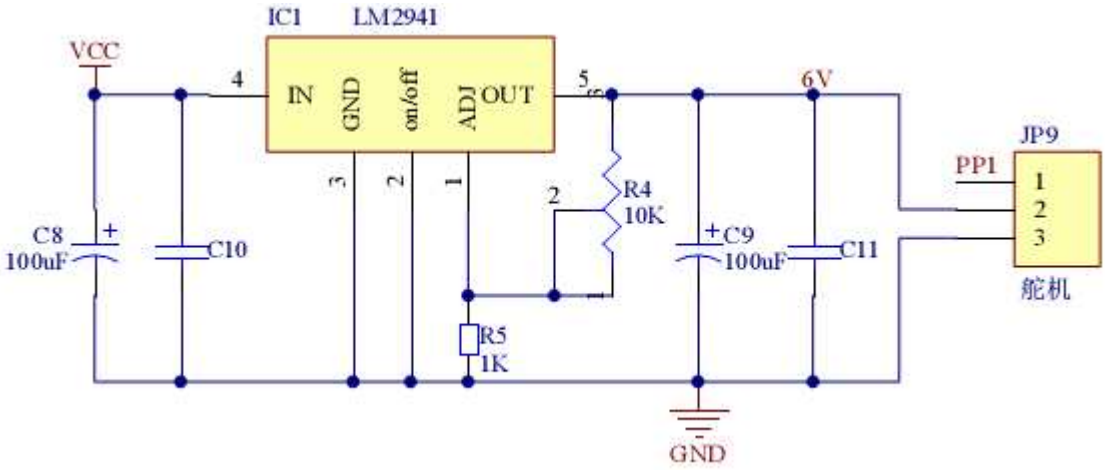


图4.5 舵机电源6v

4.2 电机驱动模块

最初我们的电机驱动元件选择的是MC33886,利用MC33886级联的方式来驱动电机，但是用了一段时间后我们发现，他的驱动能力不强，而且33886发热较严重，成本较高。于是我们自己用场效应管自行搭建了H桥电路，作为电机驱动。

图 4.6 和 4.7 就是一种简单的 H 桥电路，它由 2 个 P 型场效应管 Q1、Q2 与 2 个 N 型场效应管 Q3、Q3 组成，P 型管在栅极低电平时导通，高电平时关闭；N 型管在栅极高电平时导通，低电平时关闭，场效应管是电压控制型元件，栅极通过的电流几乎为“零”。

正因为这个特点，在连接好下图电路后，控制臂 1 置高电平（U=VCC）、控制臂 2 置低电平（U=0）时，Q1、Q4 关闭，Q2、Q3 导通，电机左端低电平，右端高电平，所以电流沿箭头方向流动。设为电机正转。

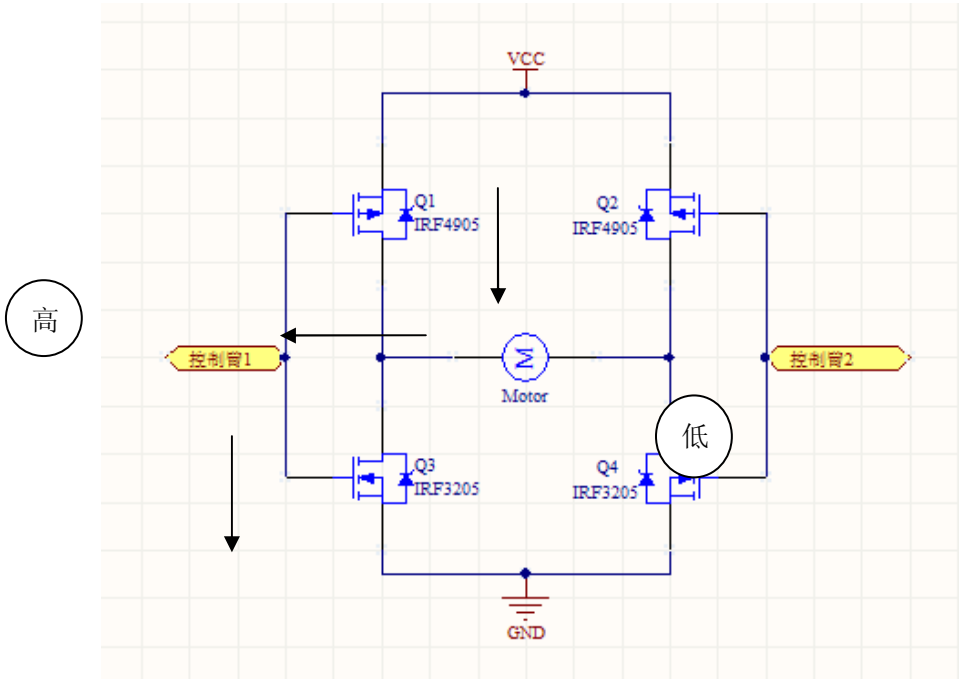


图 4.6 简单的 H 桥电路

控制臂 1 置低电平、控制臂 2 置高电平时，Q2、Q3 关闭，Q1、Q4 导通，电机左端高电平，右端低电平，所以电流沿箭头方向流动。设为电机反转。

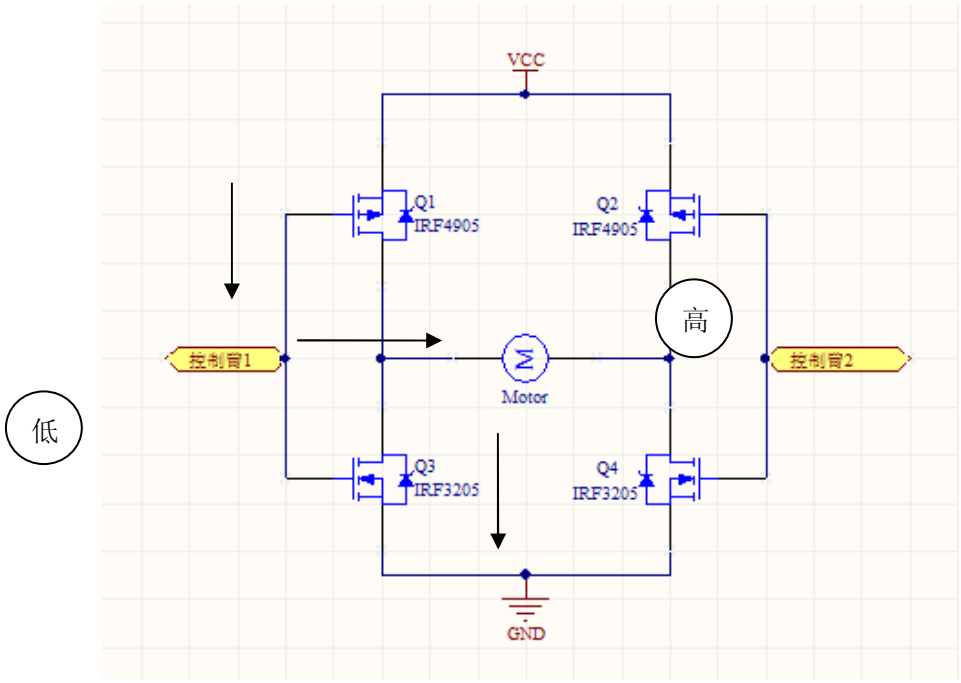


图 4.7 简单的 H 桥电路

当控制臂 1、2 均为低电平时，Q1、Q2 导通，Q3、Q4 关闭，电机两端均为高电平，电机不转；

当控制臂 1、2 均为高电平时，Q1、Q2 关闭，Q3、Q4 导通，电机两端均为低电平，电机也不转，

所以，此电路有一个优点就是无论控制臂状态如何，H 桥都不会出现“共态导通”（短路），很适合我们使用。

另外还有 4 个 N 型场效应管的 H 桥，原理基本一样，不再赘述。

图 4.8 是由与非门 CD4011 组成的栅极驱动电路，因为单片机输出电压为 0~5V，而 H 桥的控制臂需要 0~7.2V 电压才能使场效应管完全导通，设 PWM1 输入 0~5V 时，11 引脚输出电压为 0~7.2V，前提是 CD4011 电源电压为 7.2V。故 CD4011 仅做“电压放大”之用。

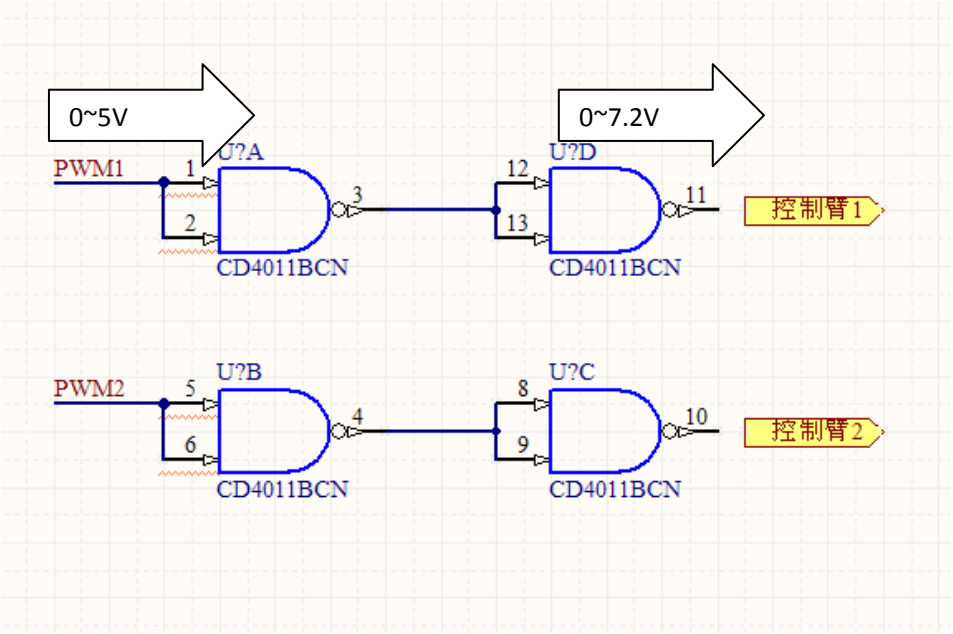


图 4.8 非门 CD4011 组成的栅极驱动电路

两者结合就是下面的电路图 4.9：

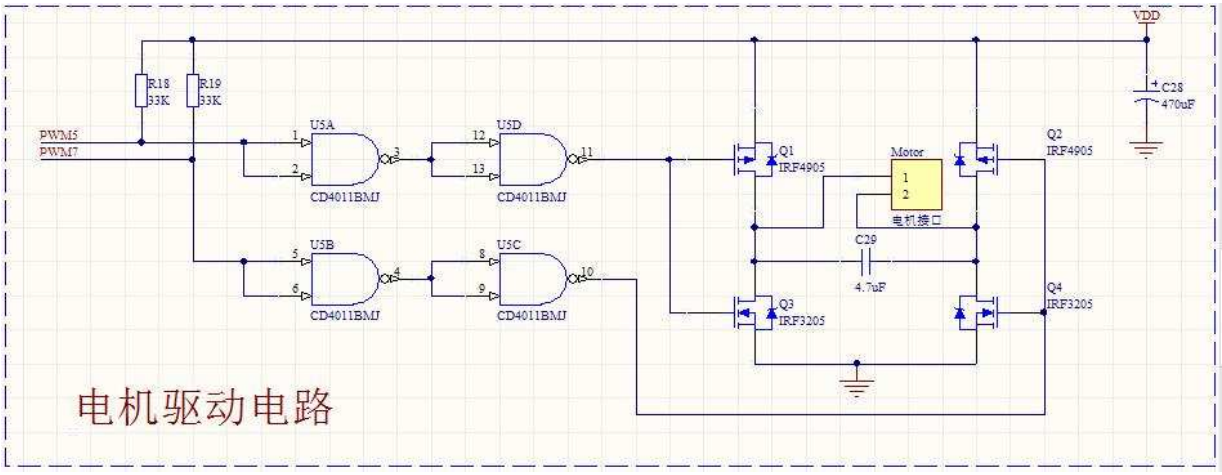


图 4.9 电机驱动电路

使用时单片机 PWM 输出 1 路为 PWM 方波信号，另一路为高电平（置 1）。

4.3 摄像头

在选择CCD还是CMOS摄像头的问题上，我们通过查阅前几届参赛的智能车的技术报告，发现用CCD的较多，目前CCD型摄像头的分辨率都比较高，图像质量较好，工作电压一般是9-12V ，但价格昂贵。COMS有分辨率低的，而且CMOS的也比较省电，工作电压有的5V就可以，考虑到系统实际总电压是7.2V，我们就选择了一款带有数字和

模拟同时输出的CMOS单板摄像头，其图像采集芯片用得是数字摄像头OV6620，而没有选择模拟摄像头。其有效像素和分辨率是356*292。

模拟的优势比较地明显：便宜，程序有现成的。缺点：消耗MCU资源，功耗大，取点个数少，需要做12V的供电模块（最近有队伍说把摄像头上的5V稳压芯片取下来飞飞线就可以直接用5V供电），外围处理电路多，还要LM1881。

数字的优点就是避免了模拟的缺点。行场同步中断信号有现成的，而且消隐区也十分有规律。可以用示波器对比一下模拟的和数字的，数字的信号非常漂亮，非常稳定。这对于图像采集来说是十分有利的。

最终我们选择了数字摄像头OV6620。

OV6620的管脚分布样子如图4.10和4.11所示：

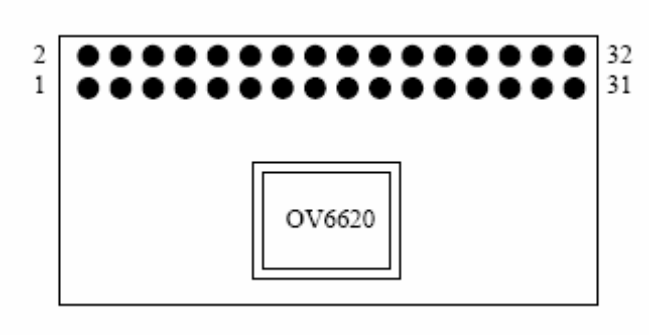


图4.10 OV6620管脚分布图　　图4.11 OV6620像头

PIN1-PIN8	灰度信号输出接口Y0-Y7
PIN11 SCCB	数据接口 SDA
PIN12	奇偶场同步信号 FODD
PIN13 SCCB	数据时钟 SCL
PIN14	行中断信号 HREF
PIN16	场中断信号 VSYN
PIN18	像素同步信号 PCLK(也叫TCLK)
Pin20	工作电压5V
Pin22	工作电压5V
Pin21	工作电压0V

PIN32　　模拟信号输出接口 VTO

4.4 速度传感器

要使车能够快速稳定的运行，并且很好的实现加速和减速，速度控制就是很重要的，所以我们就需要精确的测得车的速度，要选择较好的速度传感器。速度采集有多种方案。

方案一：采用霍尔传感器和磁钢。将霍尔传感器和磁钢分别安装在车架和车轴的适当位置，小车行驶时，每转动一圈，霍尔传感器产生开关信号，通过在单位时间对其计数可计算出车辆行驶的瞬时速度，累计开关信号可计算出小车行驶的距离。但是这种方法要求在轴上嵌入磁钢，实现复杂，并且不可能放太多磁钢所以精度不高。

方案二：采用红外对管和编码盘。将一个带有孔的编码盘固定在转轴上，然后由红外对管检测编码盘的孔对红外线的阻通。他的精度不高，而且制作较为困难，但是成本低。

方案三：采用光电编码器。光电编码器是一种通过光电转换将输出轴上的机械几何位移量转换成脉冲或数字量的传感器。这是目前应用最多的传感器，光电编码器是由光栅盘和光电检测装置组成。光栅盘是在一定直径的圆板上等分地开通若干个长方形孔。由于光电码盘与电动机同轴，电动机旋转时，光栅盘与电动机同速旋转，经发光二极管等电子元件组成的检测装置检测输出若干脉冲信号，其原理示意图如图 1 所示；通过计算每秒光电编码器输出脉冲的个数就能反映当前电动机的转速。这种测量方法方便简单可靠，是在高速车辆的很好选择。

最终我们选择了 200 线的光电编码器，型号为 YZ30D4S-2NA-200。

我们通过脉冲计数的方法来实现对小车速度的检测：在靠近小车右轮的轴上装光电编码器，这样当小车前进，车轮转动时，光电编码器跟随车轮同步转动，当一个黑色脉冲被摄像头检测到时，速度传感器的输出就变为高电平，产生脉冲，送给单片机的 ECT 模块，ECT 模块捕捉脉冲信号并对其进行计数，同样的，当白色被检测到时，也产生一脉冲，送以单片机计数，在一特定时间内（20ms，即摄像头采集一帧图像的时间）读出脉冲总数，将该总数除以车轮转动一圈移过的脉冲数目，便可以计算出车轮的转动圈数，再乘以车轮周长，得到行驶路程，再除以计数时间，最后得到小车的速度。假设N为一个采样周期T内ECT模块记录的脉冲个数，T为采样周期（单位为s），l为小车后轮周长，s为小车前进距离，v为小车的速度，n为光电编码器转过的圈数，此光电编码器上有200个齿，后轮上的装有的黑色减速器的齿轮为74。则：

$$n = \frac{N}{200} \times \frac{74}{13} \tag{1}$$

又由 $S = l \times n$ (2)

即有 $V = \frac{S}{T}$ (3)

经过测量，小车后轮周长l=157mm，而采样周期T=20ms，从而速度为：

v = 0.223423N (m/s)

4.5 串口和模式选择

因为我们将所有元件集中都一块最小系统板上，所以我们要尽量的减少元件数量，和元件的占用板子的面积。所以我们没有用通用的那种9针的串口，而是做了USB转串口的串口，如图4.12.同时模式选择我们直接用跳线来选择，如图4.13

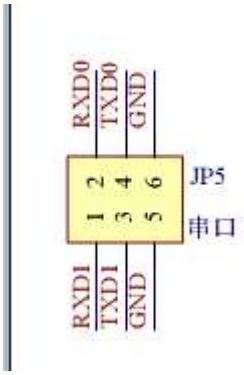


图4.12 串口

图4.13 模式选择

第五章　软件系统设计与实现

5.1 HCS12控制软件主要理论

智能车开发环境采用了飞思卡尔HCS12系列单片机开发软件CodeWarrior。该软件具有支持多种语言、开发环境界面统一、交叉平台开发以及支持插件工具等特点。在CodeWarrior界面完成编译后，通过BDM FOR S12工具，在CodeWarrior环境下向MC9S12XS128模块下载程序。BDM FOR S12工具使用简单，十分方便。本设计我们采用C语言进行编程。

5.2 软件各功能模块设计

5.2.1 时钟模块

时钟基本脉冲是CPU工作的基础。MC9S12XS128微控制器的系统时钟信号，由时钟振荡电路或专用时序脉冲信号提供。MCU内部的所有时钟信号都来源于EXTAL引脚，也为MCU与其他外接芯片之间的通信提供了可靠的同步时钟信号。

S12的总线时钟是整个MCU系统的定时基准和工作同步脉冲，其频率固定为晶体频率的1/2。对于S12，可以利用寄存器SYNR、REFDV来改变晶振频率f_{OSC}CLK，可以选用8MHz或16MHz外部晶体振荡器作外时钟。将SYNR设为3，REFDV设为1，可以得到32MHz的总线频率。

而锁相环产生的时钟频率f_{PLL}CLK=2*f_{OSC}CLK*(SYNR+1)/(REFDV+1)，设计中我们将SYNR设为3，REFDV设为1，因此，总线时钟为32MHz，CPU工作频率为64MHz。

5.2.2 PWM输出模块

MC9S12XS128集成了8路8位独立PWM通道，通过相应设置可变成4个16位PWM通道，每个通道都有专用的计数器，PWM输出极性和对齐方式可选择，8个通道分成两组，共有4个时钟源控制。PWM0、PWM1、PWM4、PWM5为一组，使用时钟源Clock A和Clock SA；PWM2、PWM3、PWM6、PWM7构成另一组，使用时钟源Clock B和Clock SB。Clock A和Clock B均是由总线时钟经过分频后得到，分频范围1~128，通过寄存器PWMPRCLK来设置，Clock SA和Clock SB是分别通过Clock A和Clock B进一步分频后得到的，分频范围为1~512，分别通过寄存器PWMSCLA和PWMSCLB来设置，计算公式为：

Clock SA=Clock A/（2*PWMSCLA）

Clock SB=Clock B/（2*PWMSCLB）

通过寄存器PWME来控制PWM0~PWM7的启动或关闭。为了提高精度，我们将PWM0和PWM1，构成16位的PWM通道，级联时，2个通道的常数寄存器和计数器均连接成16位的寄存器， 3个16位通道的输出分别使用通道7、3、1的输出引脚，时钟源分别由通道7、3、1的时钟选择控制位决定。级联时，通道7、3、1的引脚变成PWM输出引脚，通道6、2、0的时钟选择没有意义。

通过寄存器PWMPRCLK、PWMSCLA、PWMSCLB、PWMCLK对各通道的时钟源进行设置。

通过寄存器PWMPRCLK、PWMSCLA、PWMSCLB、PWMCLK对各通道的时钟源进行设置。

PWM模块的初始化设置过程如下所示：

```
    void extern PWM_Init(void)
  {
    PWME = 0x00;          //pwm禁止
    PWMCTL = 0x10;        //通道0，1级联，形成16位pwm通道
    PWMPRCLK = 0x00; //clock A,B分频值为总线时钟的0分频,40MHz
    PWMSCLA = 5;       //clock SA的频率为4MHz
    PWMSCLB = 10;      //clock SB的频率为2MHz
    PWMCLK = 0x0FF;      //时钟来源选择 clock SA  clock SB
    PWMPOL = 0x0FF;       //在周期开始时，PWM所有通道输出高电平
    PWMCAE = 0x00;        //所有PWM通道输出左对齐
    PWMPER01 = 40000;          //PTP1输出频率100Hz
    PWMDTY01 = 6000;      //通道1占空比0.075  右极限7200 左极限4700
    PWMPER2 = 200;        /*PTP1输出频率10000Hz*/
    PWMDTY2 = 88;     /*通道1占空比0.1*/
```

```
    PWME = 0x06;          //PWM1.2输出
  }
```

5.2.3 ECT模块

S12得ECT具有8个输入（IC）/输出(OC)比较通道，可以通过设置TIOS寄存器选择输入或输出比较功能。ECT既可以作为一个时基定时产生中断，也可以用来产生控制信号。

模数递减计数器（MDC）是S12微控制器ECT特有，它是一个16位计数器，其外围配备了常数寄存器MCCNT和控制寄存器MCCTL，分别为MDC提供定时常数和时钟信号。通过寄存器TCTL4设定各个引脚的各种动作，初始化设置过程如下所示：

```
//定时器初始化
void extern vECTInit(void)
{
 TIOS =0x00;//定时器通道0，1为输入捕捉
 TSCR1=0x80;//定时器使能
 TCTL4=0x09;//通道0捕捉上升沿通道1捕捉下降沿
 TIE=0x03;  //通道0，1中断使能
 TFLG1=0xFF;//清中断标志位
}
void extern IOC_Init(void)
{
 //TSCR2 = 0x04;     // 禁止定时器溢出中断,计数器自由运行,禁止复位,预分频系数为16
                     // busclock/16=48Mhz/16=3000000
 TIOS  = 0x00;     // 设置通道2工作在输出比较状态,其它通道工作在输入状态
 //TC2    = 0x2328;  // 0x2328*(1/3000000)=3ms
 //TCTL2 = 0x00;    // 切断OC2与输出引脚断开
 // TCTL3=0X80;
```

```
 TCTL4=0x09;
 //TSCR1_TFFCA=1;    // 通道自动清除
 // TIE= 0x83;         // 通道0,1,2,7中断使能
 //TIE= 0x84;
 TSCR1_TEN  = 1;  // 定时器使能
 PACTL = 0x40;     //脉冲累加器使能,事件计数,下降沿计数,16位A累加器
 PACNT = 0x0000;
 TFLG1=0xFF;       //清中断标志位
 PERT=0x80;//PT7口上拉电阻使能
 PPST=0x00;
}
```

通过ECT模块，我们实现了对行中断和场中断的PT0和PT1口的使能和捕捉方式，PT7口对脉冲进行计数，检测智能车的速度，对速度进行闭环控制。

5.3 图像采集

实际上，图像采集的这块最关键的也是时序的把握了。另外，中断的优先级一定要保证，要不然系统一运行起来,图像采到一半就丢了。

摄像头每扫描完一行，就输出一低于视频信号电压的的电平，相当于每行图像对应的电压信号之后会有一电压“凹槽”，此凹槽被称为行同步脉冲。扫描完该场的视频信号，接着会出现一段消隐区，此区中有若干个复合脉冲（简称消隐脉冲），在这些脉冲中，有一个脉冲，它远宽于其他的消隐脉冲，该消隐脉冲称为场同步脉冲。场同步脉冲标志着新的一场的到来，不过，场消隐区恰好跨在上一场的结尾部分和下一场的开始部分，得等场消隐区过去，下一场的视频信号才真正到来。摄像头每秒扫描25幅图像，每幅又分奇、偶两场，先奇场后偶场，故每秒扫描50场图像。奇场时只扫描图像中的奇数行，偶场时则只扫描偶数行。如图5.1所示：

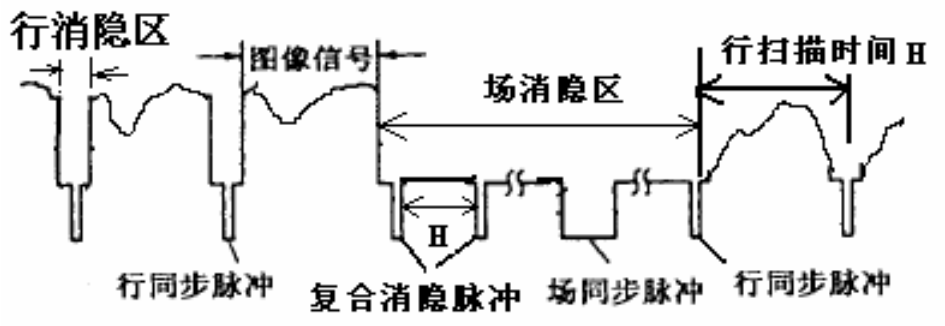


图5.1 摄像头视频信号

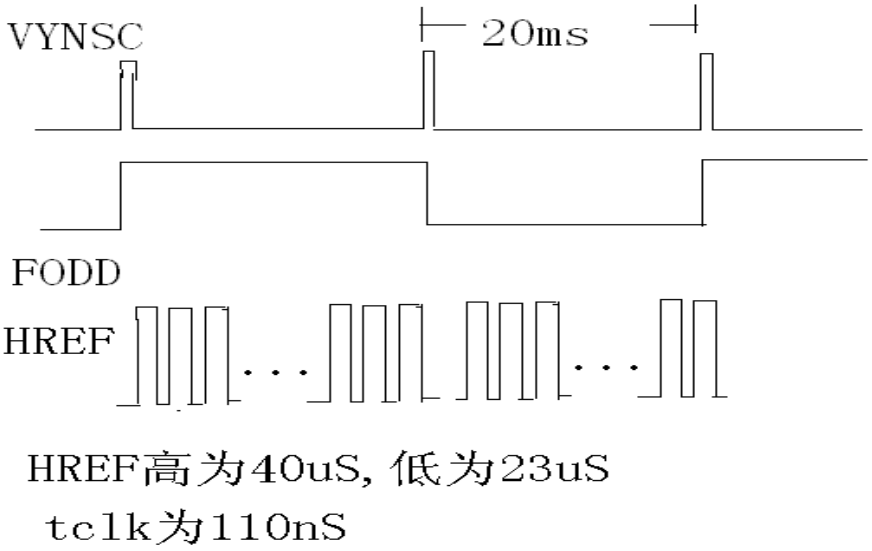


图5.2 摄像头信号采集时序图

如图5.2所示：

（1）在采集时乎略TCLK，首先是因为它太快了，捕捉不到，另外也没有必要捕捉到它。采集图像时尽快地一个点一个点的取就行了，和模拟摄像头一样。

（2）VYNSC是判断是否一幅图像开始，周期是20mS， 其中高电平持续时间很短，忽略； HREF是判断是否一行图像的开始，周期是63us左右，其中高电平持续时间为40us，低电平持续时间23us，那么可以算一下一场有多少行：20ms/63us=317，当然实际上没有这么多，消隐和无效信号去掉之后只有292行。

（3）必须明确：场中断要通过下降沿捕捉，行中断要通过上升沿捕捉。若用IRQ捕捉行中断必须加反相器。

（4）有效的灰度数据是在行中断之后的上升沿内，所以不要在行中断后的23US后

采集，那是废数据。计算一下一行OV6620有多少个点： 40us/110ns=363消隐和无效信号去掉之后只有356个点。

5.4 图像处理

5.4.1 二值化算法

图像的二值化处理就是讲图像上的点的灰度置为0或255，也就是将整个图像呈现出明显的黑白效果。即将256个亮度等级的灰度图像通过适当得阀值选取而获得仍然可以反映图像整体和局部特征得二值化图像。首先要把灰度图像二值化，得到二值化图像，这样子有利于图像进一步处理时，图像的集合性质只与像素值为0或255的点的位置有关，不再涉及像素的多级值，使处理变得简单，而且数据的处理和压缩量小。为了得到理想的二值图像，一般采用封闭、连通的边界定义不交叠的区域。所有灰度大于或等于阀值的像素被判定为属于特定物体，其灰度值为0表示，否则这些像素点被排除在物体区域以外，灰度值为1，表示背景或者例外的物体区域。因为赛道只有黑白两种颜色，其很容易分辨，采用二值化算法可以得到很好的效果，为了得到可靠的阀值，我们进行了大量的试验，最后得出了这个阀值为52，当小于这个阀值时，我们认定是检测到黑线，当大于这个阀值时，我们认为是检测到白线。图5.3为二值化后在串口上显示出来的图像。

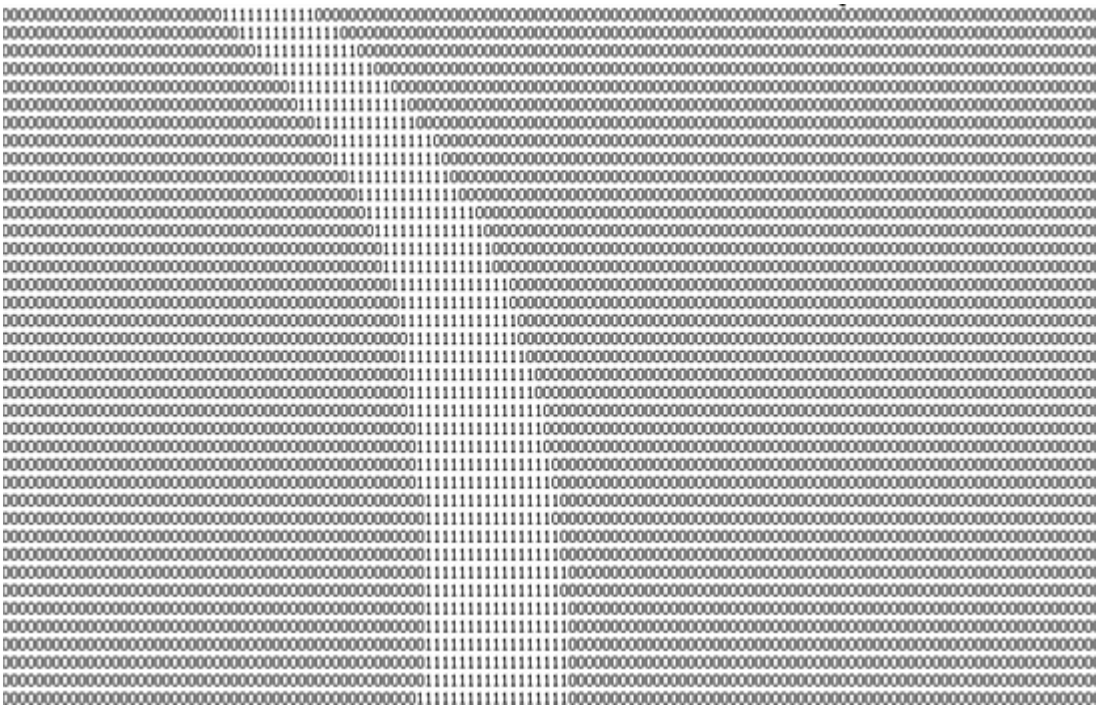


图5.3 二值化后的图像

5.4.2 黑线提取流程

黑线的提取我们参考了去年上海大学的黑线提取方法，在前十行采用边缘提取方法，十行以后的利用跟踪边缘提取方法。实验测得这种方法只要细节掌握好能够很好的提取出黑线。下面是具体的黑线提取方法。如图5.4

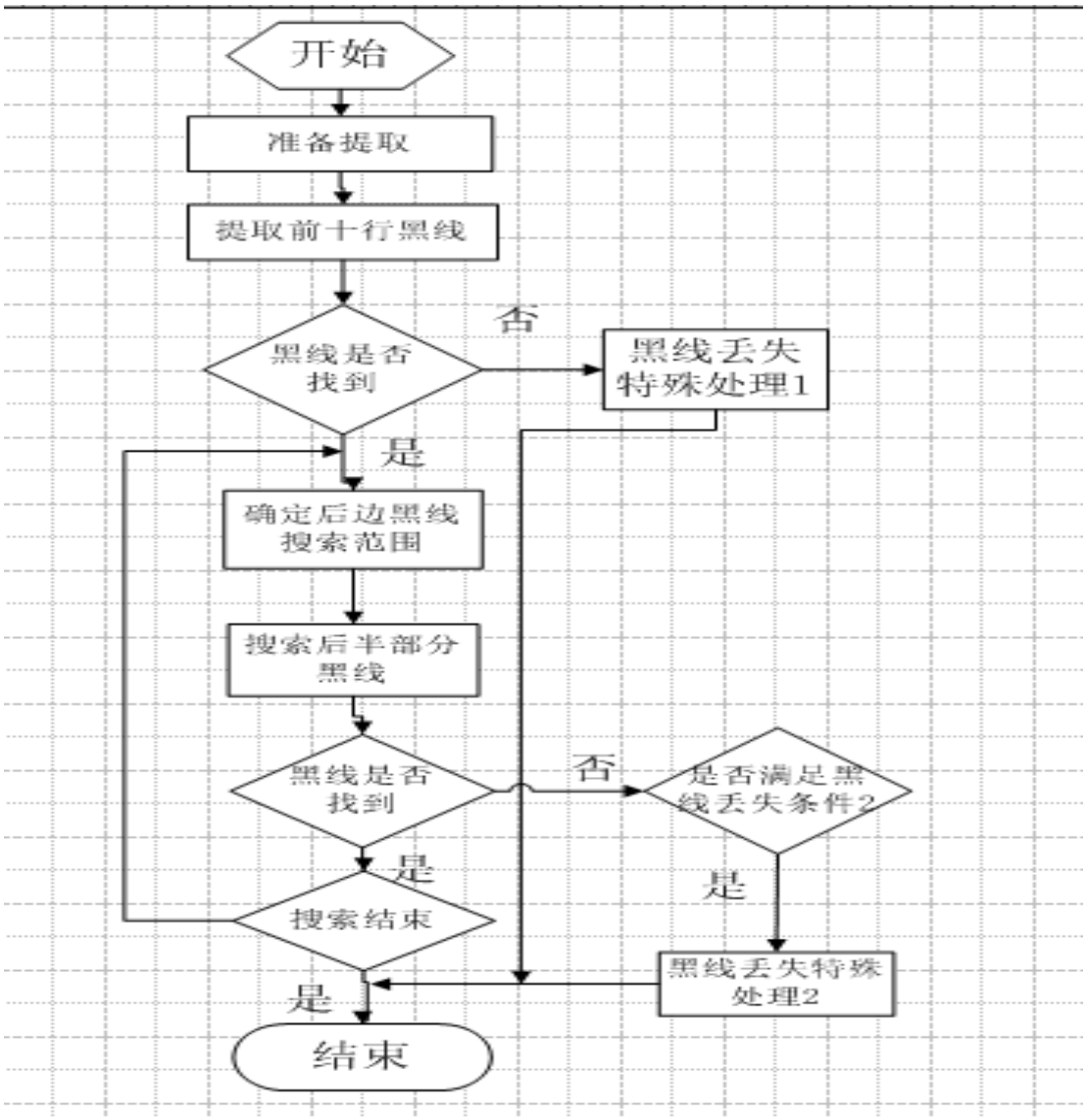


图5.4黑线提取流程图

1） 首先准备提取黑线

2） 用检测跳变的方法提取出前十行中每行的两个跳变点，然后求平均值就可以得到前十行的黑线位置。当搜索到多个跳变的时候，我们根据上一行跳变的位置确定出最优的那个跳变的位置作为本行的黑线跳变位置。当前十行都没有找到黑线的时候，我们就认为这幅图像的黑线丢失了，然后依据前一幅图像黑线的位置，给这幅图像的整体赋极值。当只有十行中的几行丢失时，我们就继续搜索黑线直道找完前十行位置。

3） 当前十行黑线存在时，我们利用前十行黑线的位置确定第十一行黑线的位置，然后在这个区间搜索黑线，依次类推用前一行黑线的位置确定后一行黑线的位置，当本行黑线没有找到时，此行黑线位置保持上行的值，下行搜索的位置相应的扩大。有

连续3行黑线搜索不到十我们就认为黑线丢失，退出搜索。这样既可以去除干扰，还可以大大的提高算法的效率。

值得注意的是：第十行和前一行第九行这个接口位置边缘确定非常重要，要考虑的非常全面，不然有可能就只能搜索到前十行的黑线，后面的黑线因为边缘的问题所有不到。

4）搜索完成后推出搜索。对搜索到的黑线进行中值滤波和限幅滤波。

只要搜索范围合理，这种算法有很强的抗干扰能力，并且可以滤除十字交叉和三角黑区的干扰。

5.5 控制算法

5.5.1 PID 简介

PID控制以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，控制理论的其它技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用PID控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用PID控制技术。PID控制就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。目前PID控制在工业控制系统中无处不见，随着控制效果的要求不断提高，PID逐渐向智能化发展，但形形色色“时髦”的现代控制理论中的PID最终还是源自经典PID理论。为什么PID应用如此广泛、又长久不衰？是因为PID解决了自动控制理论所要解决的最基本问题，既系统的稳定性、快速性和准确性。调节PID的参数，可实现在系统稳定的前提下，兼顾系统的带载能力和抗扰能力，同时，在PID调节器中引入积分项，系统增加了一个零积点，使之成为一阶或一阶以上的系统，这样系统阶跃响应的稳态误差就为零。由于自动控制系统被控对象的千差万别，PID的参数也必须随之变化，以满足系统的性能要求。

5.5.2 模拟PID与数字PID

PID控制算法包括模拟PID和数字PID两种控制算法。模拟PID 调节具有原理简单、易于实现、鲁棒性强和适用面广等优点，在实际应用中，根据实际工作经验在线整定PID各参数，往往可以取得较为满意的控制效果。数字PID控制则以此为基础，与计算

机的计算与逻辑功能结合起来，不但继承了模拟PID调节的优点，而且由于软件系统的灵活性，PID算法可以得到修正而更加完善，使变得更加灵活多样，更能满足生产过程中提出的各种控制要求。

5.5.3 PID控制原理

设系统的误差为$e(t)$，则模拟PID控制规律为：

$$u(t)=K_p\left[e(t)+\frac{1}{T_i}\int_0^t e(t)dt+T_d\frac{de(t)}{dt}\right] \quad (4)$$

式中，u（t）--控制量

K_P--比例系数

T_i--积分时间常数

T_d—微分时间常数

它所对应的连续时间系统传递函数为：

$$\frac{U(s)}{E(s)}=K_p\left[1+\frac{1}{T_i s}+T_d s\right] \quad (5)$$

其结构框图如下图5.5：

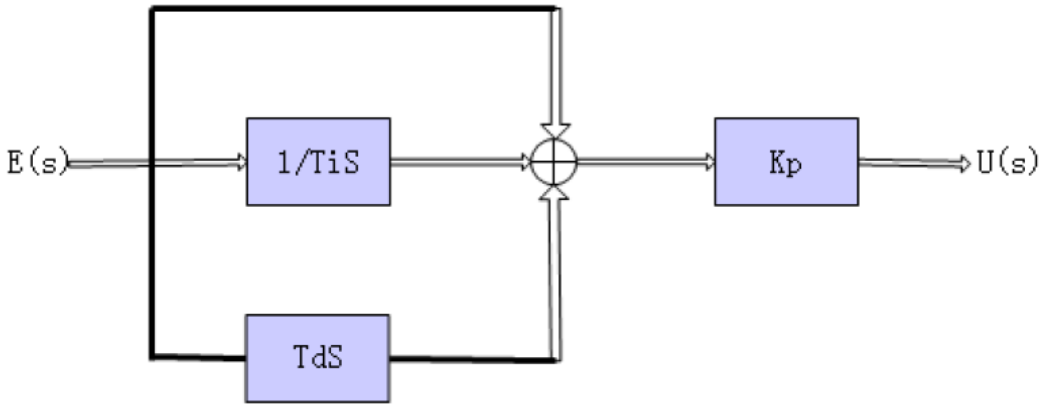


图5.5 结构框图

将上式离散化，即将描述连续系统的微分方程代之以等效的描述离散系统的差分方程，就可以得到相应的数字PID调节器。利用矩形法进行数值积分，即以求和代替积分，以差分代替微分，可得到数字形式的PID控制规律：

$$u(k)=K_p\left[e\left(k+\frac{1}{T_i}\sum_{i=0}^{k}e_iT+T_d\frac{e(k-e(k-1))}{T}\right)\right] \quad (6)$$

若T足够小，则上式可以相当准确的逼近模拟PID控制规律。

5.5.4 位置式PID与增量式PID

若以$u(k)$作为控制信号，$u(k)$即直接决定了执行机构的位置，因此上述公式6又称为位置式PID 算法。这种算法中有一累加项，随着时间K的增加，累加的向此也依次增加，不利于计算机计算。另外，如果由于某种干扰因素导致$u(k)$为某一极限值时，被控对象的输出也将作大幅度的剧烈变化，由此可能导致严重的事故。究其原因，位置式算式存在以上缺陷的主要原因是它所给出的只是当前控制量的大小，与此前时刻控制量的大小却完全不相关。为此，提出以下增量式PID 控制算法。根据以上公式6写出前一时刻的输出有：

$$u(k-1)=K_p[e(k-1)]+\frac{1}{T_i}\sum_{i=0}^{k-1}e_iT+T_d\frac{e(k-1)-e(k-2)}{T} \quad (7)$$

将两式相减，得到:

$$u(k)=u(k)-u(k-1)=K_p\left[e(k)-e(k-1)+\frac{T}{T_i}\sum_{i=0}^{k-1}e_iT+T_d\frac{e(k-1)-e(k-2)}{T}\right] \quad (8)$$

公式8每次给出的是控制量u(k)的增量，因此它被称为增量式PID算法。增量式较之位置式的优点是：计算机值输出增量，误动作影响小，必要时可增设逻辑保护；手动/自动切换时，冲击小；算式不需要累加，只需记住4个历史数据，占用内存小，计算方便。为了便于编程，同类项合并得到：增量式PID 控制算法的公式：

$$u(k)=K_p[A_1e(k)+A_2e(k-1)+A_3e(k-2)] \quad (9)$$

其中，$A_1=1+\frac{T}{T_i}+\frac{T}{T_d}$　　$A_2=-\left(1+\frac{2T_d}{T}\right)$　　$A_3=\frac{T_d}{T}$

控制输入量e（k）为小车偏离黑线的位置。

5.5.5 PID各环节作用

在实际应用中，可以根据被控对象的特性和控制要求，灵活地改变其结构，取其中一部分环节构成调节器。如比例（P）调节器、比例积分（PI）调节器、比例微分（PD）调节器等。

（1）比例调节

调节作用快，系统一出现偏差，调节器立即将偏差放大1/P倍输出。系统存在余差比例带越大，过渡过程越平稳，但余差越大，比例带越小，过渡过程易振荡，比例带太小时，就可能出现发散振荡。比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。当仅有比例控制时系统输出存在稳态误差（Steady-state error）。

(2)积分调节

积分调节作用的输出变化与输入偏差的积分成正比，积分调节作用的输出不仅取决与偏差信号的大小，还取决于偏差存在的时间，只要有偏差存在,尽管偏差可能很小，但它存在的时间越长，输出信号就越大，只有消除偏差，输出才停止变化。在积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误的或简称有差系统（System with Steady-state Error）。为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。因此，比例+积分(PI)控制器，可以使系统在进入稳态后无稳态误差。

(3)微分调节

微分调节的输出是与被调量的变化率成正比。在比例微分调节作用下，有时尽管偏差很小，但其变化速度很快，则微分调节器就有一个较大的输出。在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后(delay)组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而目前需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有比例+微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例+微分(PD)控制器能改善系统在调节过程中的动态特性。

5.6 路径识别

路径识别我们采取了东北大学的三点确定曲率的方法，在一幅图像的前中后各取一点，然后利用这三点计算出赛道的曲率。经过我们实验发现，直道曲率为一个恒定的值42，蛇形道的曲率值小于65，普通弯道小于75，大s和发卡弯都大于75.通过这种方法能够很好的识别出赛道，但是我们在写这个算法的时候由于自己水平有限出现过许多的问题，比如单片机的溢出问题，计算结果和理想结果不一致等，但是这都是算法的问题，经过我们一步一步的改进，最终用这种算法实现了对赛道的划分。

值得注意的是：前面我们在一副图像黑线丢失的时候给整幅图像都给的极致，判断曲率在这个时候也把他判断成了直道，后面我们根据整幅图像的偏差通曲率相结合的方法判断，发现这样避免了以前出现的误判问题。

对赛道识别出来后我们具体讲三道划分了3类，直道和蛇形道合并为一类。普通弯为一类。大s和发卡弯为一类。最终我们不同的赛道采用不同的控制策略。

5.7 舵机控制

舵机控制我们用了很多的方法，最终确定为PD控制和模糊控制相结合的方法对舵机实现控制。

我们首先确定出舵机的中心位置和舵机的左右极限位置，然后判断路况多舵机实行不同的控制策略。

下面是我们对舵机的控制计算式。当判断出赛道为第一类情况时及直道或蛇形道时舵机给很小的转量，并且只用p分量控制。当赛道为第二类时及普通弯道时，增大相应的p分量，并且增加d分量。当赛道为第三类的时候两个p和d分量都调节到最大。这些分量的值都是经过无数次的实验得出来的。

```
if((r>=42)&&(r<=65)&&(abs1(sum)<10))
  err=steer_centre+8*sum;
else if((r>65)&&(r<=80))
   err=steer_centre+15*sum+10*err0;
```

```
else
 err=steer_centre+20*sum+15*err0;

if  (err>left_limit)PWMDTY01=left_limit;
else if(err<rihgt_limit)PWMDTY01=rihgt_limit;
else  PWMDTY01=err;
```

其中 r 为曲率的值，sum 为赛道的偏差 steer_centre 为舵机中心位置，err0 为 d 控制量其值为上次图像的偏差和这次偏差的差值 left_limit 和 rihgt_limit 分别为舵机的左右极限值。

5.8 电机控制

电机的控制我们参考了去年北京科技大学的技术报告，将偏差构建成二次函数来调节速度。这样做的好处是，偏差越小说明前面的路平坦，车速很快，而且变化很小。当偏差较大时，车速变化就很快。并且我们在速度控制中加了反馈控制。当速度超过规定速度时就依据速度情况减速。具体的实现算法是：

```
 if(abs1(sum)<10||abs1(sum)>30)k_s=6;
 else k_s=4;
 speed=hight_speed-sum*sum*k_s/100;
if((!((R0>=42)&&(R0<=65)))) &&(abs1(DIS_ERR)>15)
 {
 if (Get_pulse>=400){PWMDTY3=200; speed=0;PORTB=0Xff;PORTB=0X00;}
 else if(Get_pulse>=65){PWMDTY3=200;speed=0;}
 else if(Get_pulse>=55) {speed=0;PWMDTY3=0; }
 else    PWMDTY3=0;
 }
else  { PWMDTY3=0;  }

if(speed<low_speed)PWMDTY2=low_speed;
```

else PWMDTY2= speed;

其中hight_speed为最高速度，low_speed为限定的最低速度。RO 和DIS_ERR为路段判断情况。

这种控制方案的最大优势是，出弯加速快，入弯减速也快，同时对每种弯道速度有了具体的控制。

值得注意的是：每种弯道车能通过的最大速度要经过试验测出来。弯道速度给的太高会冲出去，太低会使车速降低。

第六章　开发工具及其调试

6.1 上位机开发工具

本次比赛使用的是飞思卡尔公司提供的 16 位单片机 MC9S12XS128B,软件开发工具采用 Metrowerks 公司开发的软件集成开发环境 Codewarrior for HCS12，其包括集成环境 IDE、处理器专家库、全芯片仿真、可视化参数显示工具、项目工程管理器、C 交叉编译器、汇编器、链接器以及调试器，可以完成从源代码编辑、编译到调试的全部工作。

开发语言采用 HCS12C 语言，语法与标准 C 语言基本相同，支持多种数据类型，中断服务程序用中断函数形式来实现，并提供了内嵌汇编的功能。

另外,CodeWarrior 编译器提供了几种从 C 源代码产生实际汇编代码的优化方法，这些代码被编程到微控制器中。CodeWarrior 提供了大量的优化方法，选择不同的优化选项，生成的代码是不同的。在本程序设计过程中用到了很多分支程序，但由于 CodeWarrior 的分支优化功能使得一些算法不能实现，所以在编译时我们重新选择了编译优化选项下的优化功能。软件界面如图 6.1 和 6.2

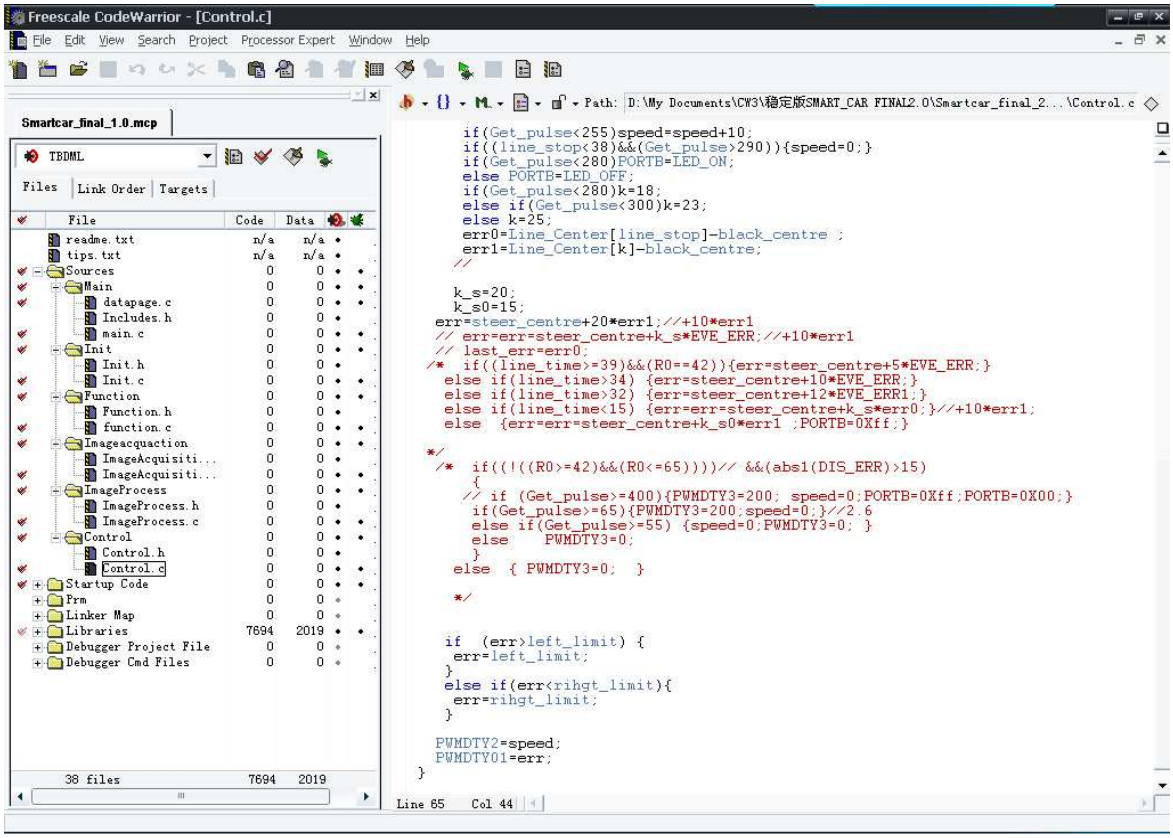


图6.1 软件开发界面

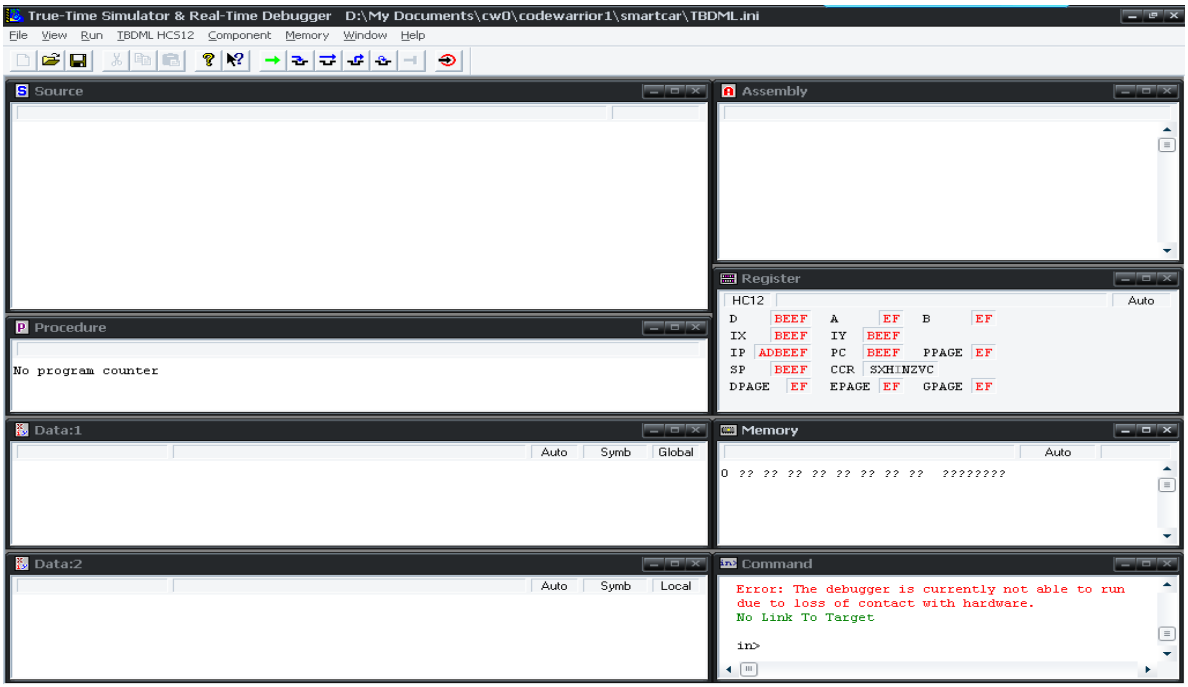
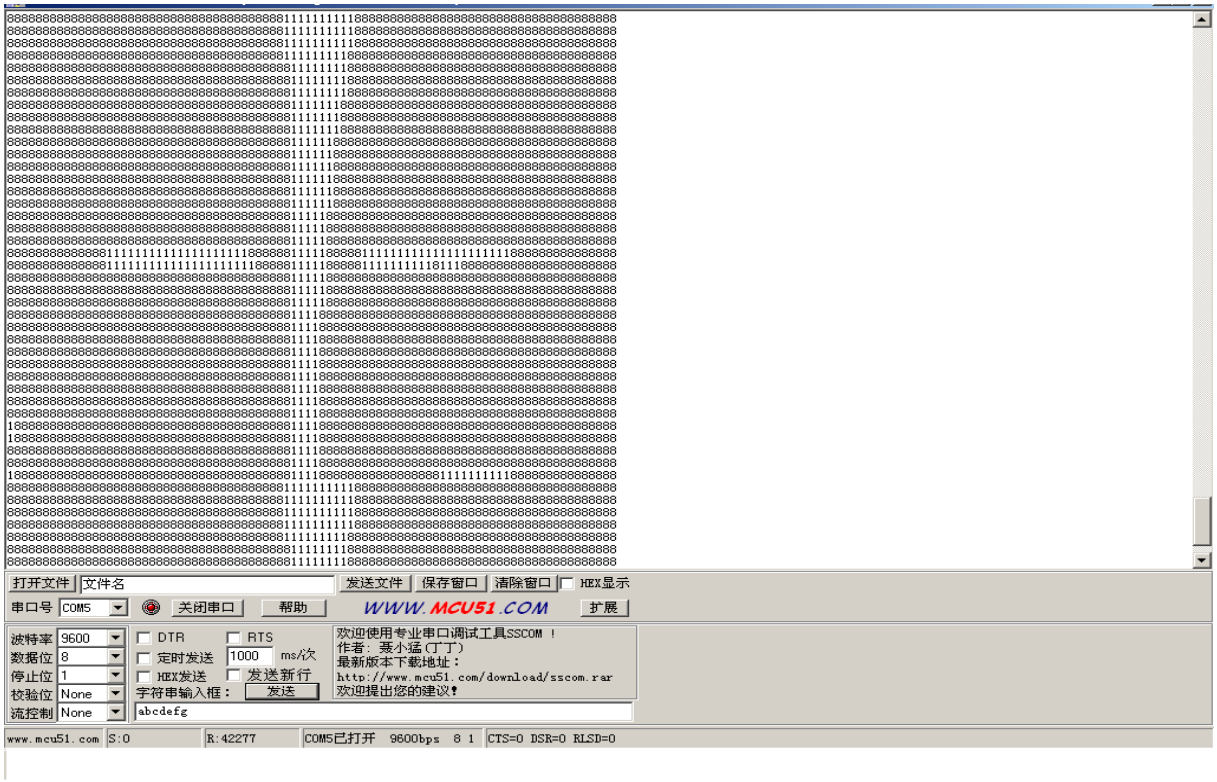


图6.2软件下载界面

6.2 串口调试工具

通过单片机的串口和PC机的串口我们将采集到的图像在串口上显示出来。串口调试软件我们没有自己编写，是从网站上下载得到的。图6.2是给软件的界面。



该软件可以接收和发送数据同时能设置相关的数据。

6.3 车模技术参数表格

2、模型车技术参数统计：

项目	参数
路径检测方法（赛题组）	摄像头组
车模几何尺寸（长、宽、高）（毫米）	280*170*290
车模轴距/轮距（毫米）	200/170
车模平均电流（匀速行驶）（毫安）	800mA
电路电容总量（微法）	1880
传感器种类及个数	摄像头 1 个光电编码器 1 个
新增加伺服电机个数	无
赛道信息检测空间精度（毫米）	1000*500
赛道信息检测频率（次/秒）	50
主要集成电路种类/数量	集成电路
车模重量（带有电池）（千克）	1.22

结 论

经过将近三个月的努力，到现在已经完成的部分，结果还是令人欣喜的。在整个过程中，全部的线路板的搭建布局都是亲力亲为，无论是摄像头的安装位置，还是硬件电路的设计，都是经过反复试验，反复修整而得出的。对于在摄像头采集信号，花费了很多精力，查阅了很多摄像头方面的期刊杂志，不断试验，最后终于获得有用的参数。

在这段紧张的时间内，让自己无论在电子知识方面还是动手能力方面有了很大的提高，更重要的是通过做完小车，无论在自信心方面还是知识钻研方面有了较大的提高。此次智能小车，前阶段经历了很多失败，也有过一筹莫展的时候，但到现在目前的状况，小车大S切弯做的还是不好，车也不是很稳定，所以要想车跑的更快，更稳，还需做更多的努力。这次控制算法上我们用了PID控制，PID 控制算法能够使小车加速减速迅速，容易实现“进弯减速，出弯加速”， 该控制算法具有很强的灵活性，可以根据实验和经验在线调整参数，可以更好的控制性能。具有控制精度高、超调小的优点，使静态、动态性能指标较为理想，同时又达到了准确、快速测定的目的。

致谢

首先感谢摆玉龙和严春满老师的指导和大力支持，感谢杨鸿武老师在 C 语言方面的支持，感谢马胜前和马永杰老师在单片机和控制方面的帮助，感谢王全洲老师和侏罗纪工作室所有同学的大力支持，感谢大四学长的支持！

参考文献：

[1] 卓晴，黄开胜，邵贝贝.学做智能车.北京：北京航空航天大学出版社.2007，3-1

[2] 刘进，齐晓辉，李永科.基于摄像头的智能小车设计与实现.维普资讯.

[3] OV6620的使用说明 ， http://shop36076594.taobao.com/

[4] 使用S12PWM控制电机 ，ppt

[5] 邵贝贝.单片机嵌入式应用的在线开发方法.北京：清华大学出版社. 2004

[6] 孙同景，陈桂友.Freescale 9S12 十六位单片机原理及嵌入式开发技术.2008，7-1

[7] 曾星星.基于摄像头的路径识别智能车控制系统设计. 湖北汽车工业学院学报,22-2

[8] 卓晴，王珽，王磊. 基于面阵CCD 的赛道参数检测方法 [R] . 清华大学自动化系

[9] 贾秀江,李颢. 摄像头黑线识别算法和赛车行驶控制策略 [R]. 上海交通大学机械设计及自动化研究所，10- 1

[10] 电动机的单片机控制－增量式PID控制算法

附录 A

程序源代码

```
#include "Includes.h"


void main(void)
{
    DisableInterrupts;
    xianshi_init();
    vPLLInit();
    PWM_Init();
    Timerch2Init() ;
    delay(3500);
    vIOPortInit();
    EnableInterrupts;
   //delay(1500);
    asm(nop);
    asm(nop);
    for(;;)
    {
     Image_binaryzation();
     black_extract();
     control_actuator();
    }
}
//*****************************************************************************
*
//*
                                        *
//*              ************************头文件及宏定义*******************
```

```
            *
//*
                                        *
//*************************************************************************
*
#ifndef __INCLUDES_H__
#define __INCLUDES_H__
#include <hidef.h>          /* common defines and macros */
#include <MC9S12XS128.h>         /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12xs128"

//*************************************************************************
*
//*              ************************初始化*****************************
*
//*************************************************************************
*
#include "Init.h"

//*************************************************************************
*
//*              ************************数据采集*****************************
      *
//*************************************************************************
*
#include "ImageAcquisition.h"

//*************************************************************************
*
```

```
//*            **********图像处理函数**********
      *
//*********************************************************************
*
#include "ImageProcess.h"

//*********************************************************************
*
//*            **********调试函数**********
      *
//*********************************************************************
*
#include "Function.h"

//*********************************************************************
*
//*            **********自动控制函数**********
      *
//*********************************************************************
*
#include "Control.h"

#endif
#ifndef __INIT_H__
#define __INIT_H__

extern void vPLLInit(void);        //锁相环初始化
//extern void vECTInit(void);       //定时器初始化
extern void vIOPortInit(void);   //I/O 口初始化
```

```
extern void PWM_Init(void);          //PWM 初始化
extern void Timerch2Init(void);
#endif
#include "Includes.h"


//锁相环初始化
void vPLLInit(void)
{
    CLKSEL=0x00;   //48mhz
    PLLCTL_PLLON=1;
    SYNR=0XC0 | 0X05;
    REFDV=0XC0 | 0X01;
    POSTDIV=0X00;
    _asm(nop);
    _asm(nop);
    while(0==CRGFLG_LOCK);//锁相环锁定
    CLKSEL_PLLSEL=1;//选定外部时钟
}

//定时器初始化
/*void   vECTInit(void)
{
  TIOS =0x00;//定时器通道 0，1 为输入捕捉
  TSCR1=0x80;//定时器使能
  TCTL4=0x09;//通道 0 捕捉上升沿通道 1 捕捉下降沿
  TIE=0x03;   //通道 0，1 中断使能
  TFLG1=0xFF;//清中断标志位
} */
```

```
void Timerch2Init(void)
{

    //TSCR2  =  0x04;       //  禁止定时器溢出中断,计数器自由运行,禁止复位,预分频系数为 16
                            // busclock/16=48Mhz/16=3000000
    TIOS    = 0x00;       //  设置通道 2 工作在输出比较状态,其它通道工作在输入状态
    //TC2     = 0x2328;    // 0x2328*(1/3000000)=3ms
    //TCTL2 = 0x00;     //  切断 OC2 与输出引脚断开
    // TCTL3=0X80;
    TCTL4=0x09;

    //TSCR1_TFFCA=1;      //  通道自动清除
    TIE= 0x83;            //  通道 0,1,2,7 中断使能
      //TIE= 0x84;
    TSCR1_TEN    = 1;    //  定时器使能
    PACTL = 0x40;       //脉冲累加器使能,事件计数,下降沿计数,16 位 A 累加器
    PACNT = 0x0000;
    TFLG1=0xFF;           //清中断标志位
}

 //端口初始化
void vIOPortInit(void)
{
   DDRM=0x00;//M 口为输入口
   DDRB=0x0FF;//B 口为输出口在指示中心位置
   DDRK_DDRK0 = 1;    //K0,K1 为输出口
```

```
    DDRK_DDRK1 = 1;
    asm(nop);
    asm(nop);
    PORTB=0x0FF;          //B 口为高电平
    PORTK_PK0 = 1;      //电机使能 K0=1;K1=0;
    PORTK_PK1 = 0;
}
//pwm 初始化
void PWM_Init(void)
{
    PWME = 0x00;                //pwm 禁止
      PWMCTL = 0x10;          //通道 0，1 级联，形成 16 位 pwm 通道
      PWMPRCLK = 0x00;      //clockA,B 分频值为总线时钟的 0 分频,40MHz
      PWMSCLA = 6;         //clockSA 的频率为 4MHz
      PWMSCLB = 12;             //clockSB 的频率为 2MHz
      PWMCLK = 0x0FF;       //时钟来源选择  clockSA    clockSB
      PWMPOL = 0x0FF;       //在周期开始时，PWM 所有通道输出高电平
      PWMCAE = 0x00;          //所有 PWM 通道输出左对齐
    PWMPER01 = 40000;       //PTP1 输出频率 100Hz
      PWMDTY01 = steer_centre;        //通道 1 占空比 0.075    右极限 7200  左极限 4700
      PWMPER2 = 200;          /*PTP1 输出频率 10000Hz*/
    PWMDTY2 = 88;        /*通道 1 占空比 0.1*/
    PWME = 0x06;                //PWM1.2 输出
}
#ifndef __IMAGEPROCESS_H__
#define __IMAGEPROCESS_H__

extern unsigned int    Line_Center[ROW_VALUE];//黑线中心数组
```

```
extern void    Image_binaryzation(void); //图像的二值化和缓存交换

extern void    black_extract(void); //黑线提取程序

#endif
#include "Includes.h"
//*************图像二值化和数据交换*****************//
#define stop_time 300

#define stop_sep_min    15
#define stop_sep_max    30

#define THRESHOLD 40    //黑白阈值  白天 45  晚上 35

#define WHITE         1      //白点
#define BLACK         0         //黑点


#define BLACK_LINE_MAX0    11 //最大黑线宽度 0
#define BLACK_LINE_MIN0    4    //最小黑线宽度 0
#define BLACK_LINE_MAX1    8    //最大黑线宽度 1
#define BLACK_LINE_MIN1    2    //最小黑线宽度 1
#define BLACK_LINE_MAX2    6    //最大黑线宽度 2
#define BLACK_LINE_MIN2    1    //最小黑线宽度 2

#define LEFT_LIMIT             1    //左极限黑线位置
#define RIHGT_LIMIT           84 //右极限黑线位置

#define BLACK_LOSTED          3    //黑线丢失限度
```

```
#define EDGE_SEP_MAX        8   //最大边缘间距
#define EDGE_SEP_MIN        5   //最大边缘间距

#define ROW_MIN             1   //黑线搜索起始
#define ROW_MIDDLE          6 //黑线搜索中间
#define ROW_MAX             25 //黑线搜索末尾

unsigned int    Line_Center[ROW_VALUE];//定义黑线中心数组

void Image_binaryzation(void)    //二值化程序
{ unsigned char *p_Image;
   unsigned char *q_Image;
     q_Image=&uca_Buffer1[0][0];

for(p_Image=&uca_Buffer[0][0];p_Image<=&uca_Buffer[ROW_VALUE-1][COLUMN_
VALUE-1];p_Image++)

{if((*p_Image<THRESHOLD)&&(*(p_Image+1)<THRESHOLD))*(q_Image++)=BLAC
K;
          else
              *(q_Image++)=WHITE;
           }

   /*unsigned thread=40;
   unsigned char ucRow,ucColumn,min,max;
   unsigned char *pucTemp;
   q_Image=&uca_Buffer1[0][0];
   p_Image=&uca_Buffer[0][0];
```

```
  for(ucColumn=0;ucColumn<ROW_VALUE ;ucColumn++)
  { min=50,max=80;
    if(ucColumn>=ROW_MAX)
     {
      for(ucRow=0;ucRow<COLUMN_VALUE;ucRow++)
     {
       pucTemp=p_Image+ucColumn*COLUMN_VALUE+ucRow;
       if(*pucTemp<min)
       min= *pucTemp;
       else if(*pucTemp>max)
       max=*pucTemp;
     }
      thread=(max+min)/2;
     }

    for(ucRow=0;ucRow<COLUMN_VALUE;ucRow++)
     {
       pucTemp=p_Image+ucColumn*COLUMN_VALUE+ucRow;
       if((*pucTemp<thread)&&(*(pucTemp+1)<thread))
          *(q_Image++)=BLACK;
       else
          *(q_Image++)=WHITE;

    }

  }*/
}

//********************黑线的提取**********************//
```

```
 void black_extract(void)
{
    unsigned char black_lost=0;//黑线丢失计数器
    unsigned char    stop=0;        //停车圈数计数器
    unsigned char i=0,j=0,left=0,right=0,start_flag1=0,start_flag2=0;
    unsigned char left_line=0,right_line=0,left_edge=0,right_edge=0;
    int temp=0;

 //DisableInterrupts;

  Line_Center[0]=Line_Center[ROW_VALUE-1];//每场的末尾给新的一场的开始，让图像连续，这句很重要！

//////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
//////////////////////图 像 前 端 黑 线 搜 索\\\\\\\\\\\\\\\\\\\\\\
//////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
  for(i=ROW_MIN;i<ROW_MIDDLE;i++)
  {   start_flag1=0;
      left=0; //左右跳变清零，这是必须的！！
      right=0;

      for(j=1;j<=COLUMN_VALUE-1;j++) //两边搜索

      { if(uca_Buffer1[i][j]-uca_Buffer1[i][j+1]>0)
          {left=j;left_line=left;start_flag1++; }

if(uca_Buffer1[i][COLUMN_VALUE-j]-uca_Buffer1[i][COLUMN_VALUE-j-1]>0)
          {right=COLUMN_VALUE-j;right_line=right;}
```

```
    }

  //过滤过宽的黑线和过细的黑线
   if(start_flag1==1)
    {
       if((right-left>=BLACK_LINE_MIN0)&&(right-left<=BLACK_LINE_MAX0))
        {    Line_Center[i]=(right+left+1)/2;
           _asm(nop);
           black_lost=0;//黑线丢失清零
        }
    }
     //如果不满足黑线的要求则利用插值求出本行位置 4 月 22 5 月 1 日修改
   else
    { //当黑线丢失三行以上认为黑线丢失，退出搜索！
       if(black_lost>BLACK_LOSTED)
            {if(Line_Center[i-1]==Line_Center[i-3])//判断是否相邻黑线位置相等
                {if(Line_Center[i-1]<=black_centre)  //若相等判断是小于黑线中心
还是大于
                  {                          //若小于则每行黑线位置减一
                   while(i<ROW_VALUE)//插值计算并且返回
                   {temp=Line_Center[i-1]-1;
                    if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                    Line_Center[i]=temp;
                    i++;
                    }
                    return;
                  }
                else //若大于黑线中心位置则每行加一
```

```
                {
                  while(i<ROW_VALUE)
                  {temp=Line_Center[i-1]+1;
                   if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
                   Line_Center[i]=temp;
                   i++;
                   }
                   return;
                  }
            }
          else //弱相邻不相等则利用二次插值计算相应的值
            {while(i<ROW_VALUE)
              {temp=2*Line_Center[i-1]-Line_Center[i-2];
              if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
              else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
              Line_Center[i]=temp;
              i++;}
             return;
            }
       }
//如果黑线丢失寄存器值小于丢失最大限度则插值计算当前的黑线位置
else
    {
      black_lost++;//黑线丢失加 1

      if(i<3) Line_Center[i]=Line_Center[i-1];
      else
        {temp=2*Line_Center[i-1]-Line_Center[i-2];
         if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
```

```
                    else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                    Line_Center[i]=temp;}
                }
            }

    }


        /*    if((start_flag1==3)&&(start_flag2==3))//首先判断跳变次数

            { if(Line_Center[i]>39&&Line_Center[i]<46)//判断是否为直道

              {if((left-right>=stop_sep_min)&&(left-right<=stop_sep_max))//判断相应的宽
度
                { if(system_time>stop_time)//判断时间
                    { stop++;system_time=0;}
                }
              }
              if(stop==1){ PWMDTY2=0;delay(50) ;PORTK_PK1 = 1;}//条件满足停车
            }   */

        ////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
//////////////////////图 像 中 间 黑 线 搜 索\\\\\\\\\\\\\\\\\\\\
////////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
for(i=ROW_MIDDLE;i<ROW_MAX;i++)
    {   start_flag1=0; //起跑线识别寄存器

        left=0;
        _asm(nop);
```

```
  right=0;
  left_edge=0;     //跳变缓存
  right_edge=0;
 for(j=5;j<=80;j++) //两边搜索

  { if(uca_Buffer1[i][j]-uca_Buffer1[i][j+1]>0)
          { start_flag1++;
            left_edge=j;
          if(abs1(left_line-left_edge)<EDGE_SEP_MIN)
            {left=left_edge;left_line=left;}
          }

if(uca_Buffer1[i][COLUMN_VALUE-j]-uca_Buffer1[i][COLUMN_VALUE-j-1]>0)
        { start_flag2++;
          right_edge=COLUMN_VALUE-j;
        if(abs1(right_line-right_edge)<EDGE_SEP_MIN)
          {right=right_edge;right_line=right;}

        }
    }

 //过滤过宽的黑线和过细的黑线
  if((right-left>=BLACK_LINE_MIN1)&&(right-left<=BLACK_LINE_MAX1))
    {   Line_Center[i]=(right+left+1)/2;
        _asm(nop);
        black_lost=0;//黑线丢失清零
    }

    //如果不满足黑线的要求则利用插值求出本行位置 4 月 22
```

```
        else
        {       if(black_lost>BLACK_LOSTED)
                 {while(i<ROW_VALUE)
                     { temp=2*Line_Center[i-1]-Line_Center[i-2];
                       if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
                       else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                       Line_Center[i]=temp;
                       i++;
                     }
                   return;
                   }

                else
                {
                   black_lost++;//黑线丢失加 1
                   temp=2*Line_Center[i-1]-Line_Center[i-2];
                   if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
                   else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                   Line_Center[i]=temp;
                }
        }

  }

////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
//////////////////////图 像 末 尾 黑 线 搜 索\\\\\\\\\\\\\\\\\\\\\\
////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

  for(i=ROW_MAX;i<ROW_VALUE;i++)
```

```
  {
      left=0;
      right=0;
      left_edge=0;
      right_edge=0;

      for(j=10;j<=74;j++) //两边搜索图像减小 10 列，因为两边的 10 列干扰较大！

      { if(uca_Buffer1[i][j]-uca_Buffer1[i][j+1]>0)
            { left_edge=j;

if(abs1(left_line-left_edge)<EDGE_SEP_MAX){left=left_edge;left_line=left;}
            }

if(uca_Buffer1[i][COLUMN_VALUE-j]-uca_Buffer1[i][COLUMN_VALUE-j-1]>0)
          { right_edge=COLUMN_VALUE-j;

if(abs1(right_line-right_edge)<EDGE_SEP_MAX){right=right_edge;right_line=right;}
          }


        }

     //过滤过宽的黑线和过细的黑线
      if((right-left>=BLACK_LINE_MIN2)&&(right-left<=BLACK_LINE_MAX2))
        {   Line_Center[i]=(right+left+1)/2;
            _asm(nop);
            black_lost=0;//黑线丢失清零
        }
```

```
        //如果不满足黑线的要求则利用插值求出本行位置 4 月 22
        else
        {       if(black_lost>BLACK_LOSTED)
                  {while(i<ROW_VALUE)
                     { temp=2*Line_Center[i-1]-Line_Center[i-2];
                       if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
                       else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                       Line_Center[i]=temp;
                       i++;
                     }
                   return;
                   }
                 else
                 {
                   black_lost++;//黑线丢失加 1
                   temp=2*Line_Center[i-1]-Line_Center[i-2];
                   if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
                   else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
                   Line_Center[i]=temp;
                 }

        }


  }
//二次插值与限幅滤波相结合

for(i=2;i<ROW_VALUE;i++)
  { if(abs1(Line_Center[i-1]-Line_Center[i])>4)
```

```
        {temp=2*Line_Center[i-1]-Line_Center[i-2];
         if(temp>RIHGT_LIMIT)temp=RIHGT_LIMIT;
           else if(temp<LEFT_LIMIT)temp=LEFT_LIMIT;
           Line_Center[i]=temp;}

 }
  //黑线的中值滤波程序!
 for(i=1;i<ROW_VALUE-1;i++)
     {temp=get_mid(Line_Center[i-1],Line_Center[i],Line_Center[i+1]);
       Line_Center[i]=temp;}

//EnableInterrupts;
} #ifndef __IMAGEACQUISITION_H__
#define __IMAGEACQUISITION_H__          /* derivative information */

#define SAMP_ROW_START 11 //起始行
#define SAMP_ROW_MAX      288//最终行
//#define SAMP_ROW_SEP      jiange    //行间隔
#define                                                                 ROW_VALUE
38//(SAMP_ROW_MAX-SAMP_ROW_START)/SAMP_ROW_SEP//行数
#define COLUMN_VALUE 86//列数(为一个定值)

extern unsigned char uca_Buffer[ROW_VALUE][COLUMN_VALUE] ;//图像采集数组
extern unsigned char uca_Buffer1[ROW_VALUE][COLUMN_VALUE] ;//图像处理数组
extern unsigned int    system_time ;//停车标志位时间计数器

#endif #include "Includes.h"
#pragma CODE_SEG NON_BANKED    //行同步 ,用于数据采集
unsigned char uca_Buffer[ROW_VALUE][COLUMN_VALUE] ;//@ BUFF_ADDR ;
```

```
unsigned char uca_Buffer1[ROW_VALUE][COLUMN_VALUE] ;//@ BUFF_ADDR ;
unsigned char SampleFlag=0    ;
unsigned char jiange;
unsigned int line=0,m=0,n=0,system_time=0;
void interrupt 8 Port0_interrupt()
{
   TFLG1_C0F=1;//清行中断
   line++;         //行计数器加 1
   if ( SampleFlag == 0 || line<SAMP_ROW_START ||    line>SAMP_ROW_MAX )
   {
      return;//判断是否从新的一场开始
    }

   if(line<=132)jiange=12;
   else if(line>132&&line<=200) jiange=8;
   else if(line>200&&line<=248) jiange=6;
   else jiange=4;
   if(line%jiange==0)
       {

                          uca_Buffer[m][0]=PTIM;    uca_Buffer[m][n+0]=PTIM;
                          uca_Buffer[m][1]=PTIM;    uca_Buffer[m][n+1]=PTIM;
                          uca_Buffer[m][2]=PTIM;    uca_Buffer[m][n+2]=PTIM;
                          uca_Buffer[m][3]=PTIM;    //uca_Buffer[m][n+3]=PTIM;
                          uca_Buffer[m][4]=PTIM;    uca_Buffer[m][n+4]=PTIM;
                          uca_Buffer[m][5]=PTIM;    uca_Buffer[m][n+5]=PTIM;
                          uca_Buffer[m][6]=PTIM;    uca_Buffer[m][n+6]=PTIM;
                          uca_Buffer[m][7]=PTIM;    //uca_Buffer[m][n+7]=PTIM;
                          uca_Buffer[m][8]=PTIM;    uca_Buffer[m][n+8]=PTIM;
```

```
uca_Buffer[m][9]=PTIM;    uca_Buffer[m][n+9]=PTIM;
uca_Buffer[m][10]=PTIM; uca_Buffer[m][n+10]=PTIM;
uca_Buffer[m][11]=PTIM; //uca_Buffer[m][n+11]=PTIM;
uca_Buffer[m][12]=PTIM; uca_Buffer[m][n+12]=PTIM;
uca_Buffer[m][13]=PTIM; uca_Buffer[m][n+13]=PTIM;
uca_Buffer[m][14]=PTIM; uca_Buffer[m][n+14]=PTIM;
uca_Buffer[m][15]=PTIM; //uca_Buffer[m][n+15]=PTIM;
uca_Buffer[m][16]=PTIM; uca_Buffer[m][n+16]=PTIM;
uca_Buffer[m][17]=PTIM; uca_Buffer[m][n+17]=PTIM;
uca_Buffer[m][18]=PTIM; uca_Buffer[m][n+18]=PTIM;
uca_Buffer[m][19]=PTIM; //uca_Buffer[m][n+19]=PTIM;
uca_Buffer[m][20]=PTIM; uca_Buffer[m][n+20]=PTIM;
uca_Buffer[m][21]=PTIM; uca_Buffer[m][n+21]=PTIM;
uca_Buffer[m][22]=PTIM; uca_Buffer[m][n+22]=PTIM;
uca_Buffer[m][23]=PTIM; //uca_Buffer[m][n+23]=PTIM;
uca_Buffer[m][24]=PTIM; uca_Buffer[m][n+24]=PTIM;
uca_Buffer[m][25]=PTIM; uca_Buffer[m][n+25]=PTIM;
uca_Buffer[m][26]=PTIM; uca_Buffer[m][n+26]=PTIM;
uca_Buffer[m][27]=PTIM; //uca_Buffer[m][n+27]=PTIM;
uca_Buffer[m][28]=PTIM; uca_Buffer[m][n+28]=PTIM;
uca_Buffer[m][29]=PTIM; uca_Buffer[m][n+29]=PTIM;
uca_Buffer[m][30]=PTIM; uca_Buffer[m][n+30]=PTIM;
uca_Buffer[m][31]=PTIM; //uca_Buffer[m][n+31]=PTIM;
uca_Buffer[m][32]=PTIM; uca_Buffer[m][n+32]=PTIM;
uca_Buffer[m][33]=PTIM; uca_Buffer[m][n+33]=PTIM;
uca_Buffer[m][34]=PTIM; uca_Buffer[m][n+34]=PTIM;
uca_Buffer[m][35]=PTIM; //uca_Buffer[m][n+35]=PTIM;
uca_Buffer[m][36]=PTIM; uca_Buffer[m][n+36]=PTIM;
uca_Buffer[m][37]=PTIM; uca_Buffer[m][n+37]=PTIM;
```

```
uca_Buffer[m][38]=PTIM; uca_Buffer[m][n+38]=PTIM;
uca_Buffer[m][39]=PTIM; //uca_Buffer[m][n+39]=PTIM;
uca_Buffer[m][40]=PTIM; uca_Buffer[m][n+40]=PTIM;
uca_Buffer[m][41]=PTIM; uca_Buffer[m][n+41]=PTIM;
uca_Buffer[m][42]=PTIM; uca_Buffer[m][n+42]=PTIM;
uca_Buffer[m][43]=PTIM; //uca_Buffer[m][n+43]=PTIM;
uca_Buffer[m][44]=PTIM; uca_Buffer[m][n+44]=PTIM;
uca_Buffer[m][45]=PTIM; uca_Buffer[m][n+45]=PTIM;
uca_Buffer[m][46]=PTIM; uca_Buffer[m][n+46]=PTIM;
uca_Buffer[m][47]=PTIM; //uca_Buffer[m][n+47]=PTIM;
uca_Buffer[m][48]=PTIM; uca_Buffer[m][n+48]=PTIM;
uca_Buffer[m][49]=PTIM; uca_Buffer[m][n+49]=PTIM;
uca_Buffer[m][50]=PTIM; uca_Buffer[m][n+50]=PTIM;
uca_Buffer[m][51]=PTIM; //uca_Buffer[m][n+51]=PTIM;
uca_Buffer[m][52]=PTIM; uca_Buffer[m][n+52]=PTIM;
uca_Buffer[m][53]=PTIM; uca_Buffer[m][n+53]=PTIM;
uca_Buffer[m][54]=PTIM; uca_Buffer[m][n+54]=PTIM;
uca_Buffer[m][55]=PTIM; //uca_Buffer[m][n+55]=PTIM;
uca_Buffer[m][56]=PTIM; uca_Buffer[m][n+56]=PTIM;
uca_Buffer[m][57]=PTIM; uca_Buffer[m][n+57]=PTIM;
uca_Buffer[m][58]=PTIM; uca_Buffer[m][n+58]=PTIM;
uca_Buffer[m][59]=PTIM; //uca_Buffer[m][n+59]=PTIM;
uca_Buffer[m][60]=PTIM; uca_Buffer[m][n+60]=PTIM;
uca_Buffer[m][61]=PTIM; uca_Buffer[m][n+61]=PTIM;
uca_Buffer[m][62]=PTIM; uca_Buffer[m][n+62]=PTIM;
uca_Buffer[m][63]=PTIM; //uca_Buffer[m][n+63]=PTIM;
uca_Buffer[m][64]=PTIM; uca_Buffer[m][n+64]=PTIM;
uca_Buffer[m][65]=PTIM; uca_Buffer[m][n+65]=PTIM;
uca_Buffer[m][66]=PTIM; uca_Buffer[m][n+66]=PTIM;
```

```
uca_Buffer[m][67]=PTIM; //uca_Buffer[m][n+67]=PTIM;
uca_Buffer[m][68]=PTIM; uca_Buffer[m][n+68]=PTIM;
uca_Buffer[m][69]=PTIM; uca_Buffer[m][n+69]=PTIM;
uca_Buffer[m][70]=PTIM; uca_Buffer[m][n+70]=PTIM;
uca_Buffer[m][71]=PTIM; //uca_Buffer[m][n+71]=PTIM;
uca_Buffer[m][72]=PTIM; uca_Buffer[m][n+72]=PTIM;
uca_Buffer[m][73]=PTIM; uca_Buffer[m][n+73]=PTIM;
uca_Buffer[m][74]=PTIM; uca_Buffer[m][n+74]=PTIM;
uca_Buffer[m][75]=PTIM; //uca_Buffer[m][n+75]=PTIM;
uca_Buffer[m][76]=PTIM;    uca_Buffer[m][n+76]=PTIM;
uca_Buffer[m][77]=PTIM;    uca_Buffer[m][n+77]=PTIM;
uca_Buffer[m][78]=PTIM;    uca_Buffer[m][n+78]=PTIM;
uca_Buffer[m][79]=PTIM;    //uca_Buffer[m][n+79]=PTIM;
uca_Buffer[m][80]=PTIM;    uca_Buffer[m][n+80]=PTIM;
uca_Buffer[m][81]=PTIM;    uca_Buffer[m][n+81]=PTIM;
uca_Buffer[m][82]=PTIM;    uca_Buffer[m][n+82]=PTIM;
uca_Buffer[m][83]=PTIM;    //uca_Buffer[m][n+83]=PTIM;
uca_Buffer[m][84]=PTIM;    uca_Buffer[m][n+84]=PTIM;
uca_Buffer[m][85]=PTIM;    uca_Buffer[m][n+85]=PTIM;
 /*uca_Buffer[m][86]=PTIM;    uca_Buffer[m][n+86]=PTIM;
uca_Buffer[m][87]=PTIM;    //uca_Buffer[m][n+87]=PTIM;
uca_Buffer[m][88]=PTIM;    uca_Buffer[m][n+88]=PTIM;
uca_Buffer[m][89]=PTIM;    uca_Buffer[m][n+89]=PTIM;
uca_Buffer[m][90]=PTIM;    uca_Buffer[m][n+90]=PTIM;
uca_Buffer[m][91]=PTIM;    //uca_Buffer[m][n+91]=PTIM;
uca_Buffer[m][92]=PTIM;    uca_Buffer[m][n+92]=PTIM;
uca_Buffer[m][93]=PTIM;    uca_Buffer[m][n+93]=PTIM;
uca_Buffer[m][94]=PTIM;    uca_Buffer[m][n+94]=PTIM;
uca_Buffer[m][95]=PTIM; //uca_Buffer[m][n+95]=PTIM;
```

```
                uca_Buffer[m][96]=PTIM;    //uca_Buffer[m][n+96]=PTIM;
                uca_Buffer[m][97]=PTIM; uca_Buffer[m][n+97]=PTIM;
                uca_Buffer[m][98]=PTIM;    uca_Buffer[m][n+98]=PTIM;
                uca_Buffer[m][99]=PTIM;    //uca_Buffer[m][n+99]=PTIM;
                uca_Buffer[m][100]=PTIM; // uca_Buffer[m][n+100]=PTIM;
                uca_Buffer[m][101]=PTIM; // uca_Buffer[m][n+101]=PTIM;
                uca_Buffer[m][102]=PTIM; // uca_Buffer[m][n+102]=PTIM;
                uca_Buffer[m][103]=PTIM;    //uca_Buffer[m][n+103]=PTIM;
                uca_Buffer[m][104]=PTIM;    //uca_Buffer[m][n+104]=PTIM;
                uca_Buffer[m][105]=PTIM;    //uca_Buffer[m][n+105]=PTIM;
                uca_Buffer[m][106]=PTIM;    //uca_Buffer[m][n+106]=PTIM;
                uca_Buffer[m][107]=PTIM;    //uca_Buffer[m][n+107]=PTIM;
                uca_Buffer[m][108]=PTIM;    //uca_Buffer[m][n+108]=PTIM;
                uca_Buffer[m][109]=PTIM;    //uca_Buffer[m][n+109]=PTIM;
                uca_Buffer[m][110]=PTIM; // uca_Buffer[m][n+100]=PTIM;
                uca_Buffer[m][111]=PTIM; // uca_Buffer[m][n+101]=PTIM;
                uca_Buffer[m][112]=PTIM; // uca_Buffer[m][n+102]=PTIM;
                uca_Buffer[m][113]=PTIM;    //uca_Buffer[m][n+103]=PTIM;
                uca_Buffer[m][114]=PTIM;    //uca_Buffer[m][n+104]=PTIM;
                uca_Buffer[m][115]=PTIM;    //uca_Buffer[m][n+105]=PTIM;
                uca_Buffer[m][116]=PTIM;    //uca_Buffer[m][n+106]=PTIM;
                uca_Buffer[m][117]=PTIM;    //uca_Buffer[m][n+107]=PTIM;
                uca_Buffer[m][118]=PTIM;    //uca_Buffer[m][n+108]=PTIM;
                uca_Buffer[m][119]=PTIM;    //uca_Buffer[m][n+109]=PTIM;

                 */
                m++ ;
}
```

```
 /*if(line<=195)jiange=9;
   else    jiange=6;*/

}


void interrupt 9 Port1_interrupt(void)
{

   TFLG1_C1F=1; //清场中断
   TFLG1_C0F=1; //清行中断
   m=0;                //列清零
   line=0;          //行计数器
   SampleFlag=1;//场标志位置 1
   system_time++; //系统时间加 1

   get_speed();

} #ifndef __CONTROL_H__
#define __CONTROL_H__
#define black_centre 41
#define steer_centre 5998
extern void    control_actuator(void);//控制程序
#endif #include "Includes.h"
#define steer_centre 5998
#define rihgt_limit 5490
#define left_limit    6500
#define black_centre 41
```

```
#define hight_speed    120
#define low_speed       90

void control_actuator(void)
{     unsigned int i=0,r=0,start=0,middle=0,end=0;
      int err=0,sum=0,sum_c=0;
      unsigned char speed=0,k_s;
// DisableInterrupts;
         asm(nop);
         start=Line_Center[1];
         middle=Line_Center[20];
         end=Line_Center[ROW_VALUE-1];

         r=curvature(1,20,38,start,middle,end);

        //if((r>=44)&&(r<=65))
      //    {
            for(i=1;i<ROW_VALUE;i++)
               {sum_c+=Line_Center[i];}
                sum=sum_c/(ROW_VALUE-1)-black_centre;
         // }
    /* else
         {
           for(i=0;i<25;i++)
               {sum_c+=Line_Center[i];}
                sum=sum_c/25-black_centre;
          } */

        if((r>=40)&&(r<=55)&&(abs1(sum)<10))
```

```
            err=steer_centre+8*sum;
        else if((r>55)&&(r<=65))
             err=steer_centre+10*sum;
        else if((r>65)&&(r<=75))
            err=steer_centre+12*sum;
        else if((r>75)&&(r<=85))
             err=steer_centre+15*sum;
        else
          err=steer_centre+18*sum;

        if    (err>left_limit)PWMDTY01=left_limit;
        else if(err<rihgt_limit)PWMDTY01=rihgt_limit;
        else    PWMDTY01=err;

   if(abs1(sum)<10||abs1(sum)>30)k_s=6;
   else k_s=4;
   speed=hight_speed-sum*sum*k_s/100;

    if(speed<low_speed)PWMDTY2=low_speed;
    else PWMDTY2= speed;




//EnableInterrupts;

}
```

附录B：基于起跑线识别算法的研究

摘要：按照本届智能车比赛的竞赛规则，比赛车辆在跑道上完成一圈比赛，以起始线为计时点，跑完一圈之后，赛车需要在通过起始线后三米范围内自动停止。如果没有停止在规定的范围内，比赛成绩时间增加 1 秒。在比赛中，一秒的时间对比赛成绩的高低至关重要，所以一个准确的起跑线识别算法就显得必不可少。本文主要论述了三种起跑线识别的思路，在实验比较的基础上，给出了一种起跑线识别的算法。实践证明，该算法对起跑线的识别是行之有效的。

关键词：飞思卡尔智能车；起跑线；模式识别

Abstract: According to the rules for the 4th national contest of Smart car, starting from the black line, the smart car should stop within three meters automatically after one routines racing. If it fails, the racing marks have to add one second. During the contest, one second is of utmost importance for the final marks. Therefore, an accurate starting line identification algorism is necessary. Three starting line identification algorisms were discussed in this paper. Based on the experiments results, one optimal algorism was given, which has been proved with the racing for identifying the starting line.

1 概述

按照第四届全国大学生“飞思卡尔”智能车大赛比赛规则要求：每辆赛车在赛道上跑一圈，以计时起始线为计时点，跑完一圈后赛车需要自动在起始线之后三米的赛道内，如果没有停止在规定的范围内，比赛成绩时间增加 1 秒 。本文主要给出了几种起跑线识别的算法，并且做了相应的对比。

2 起跑线特征的分析

要能够准确的识别起跑线那么我们首先要能够分析出是跑线的特征，并且抓住最主要的特征，也就是他不同于赛道其它的地方。图1是标准的起跑线

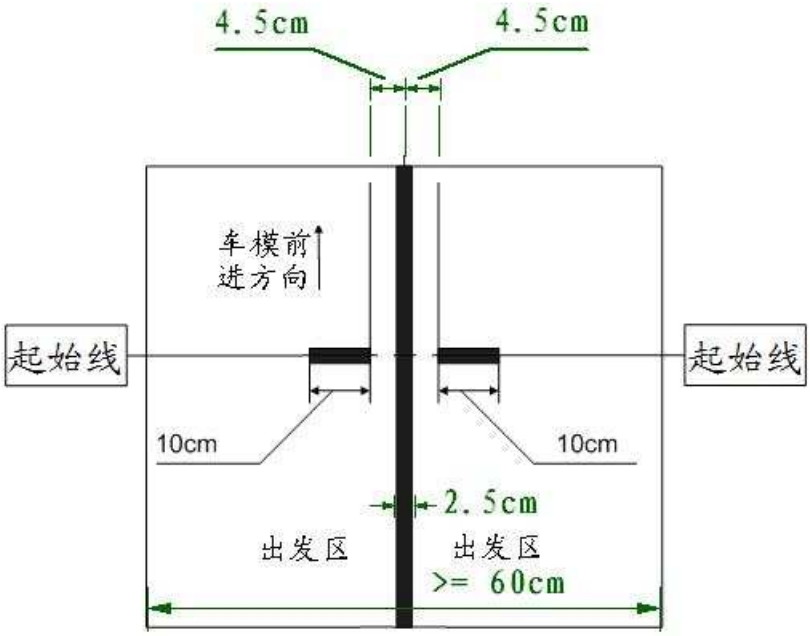


图 1 起跑线示意图

赛道有一个长为1米的出发区，如图1所示，计时起始点两边分别有一个长度10厘米黑色计时起始线，赛车前端通过起始线作为比赛计时开始或者结束时刻。

由上可知，起跑线前后最少有1m 的直道，有两条10cm 的黑道，并且他与黑色引导线的距离为4.5cm，最后是起跑线不会出现在窄道区域。

有了上面对起跑线的特征的分析我们就可以对起跑线进行识别。

3 起跑线识别算法

由于今年的比赛新增了赛车必须在起始线后三米内停下的规则，这就要求赛车必须具有稳定可靠提取起始线作为停车标志的功能。 提取起始线的难点在于高速冲过起跑线的时候，起跑线的漏检问题；车没有正对起跑线时候，起跑线的无识别问题，十字交叉的干扰问题，黑色三角区域的干扰问题，以及其他的一些不确定因素。

起跑线的识别第三届的参赛队伍提供了许多不同的方案，比如添加光电管识别起跑线，数黑点法，三段黑线法，两段黑线加一段白线法等等。我们主要参考上届的方案，并且对他们做了对比和综合，最终提出了自己的起跑线识别方法。

方案一：光电识别法

添加光电管识别起跑线，首先要依据起跑线的特征对光电管的布局做分析，我们可以安装5个光电管，黑线中间一个，中间4.5cm 处各一个，长10cm 黑线处两边各一个。然后就读取光电管的状态。首先我们考虑车直接正着冲过起跑线的时候，那么光电管的状态为最两侧和中间为低电平，中间为高电平。这种情况相对而言较简单。当车斜着通过起跑线的时候，按照上述思想就很难识别出起跑线。那么我们可以对判定起跑线的条件相应的放宽一些，当车斜着通过起跑线的时候，光电管中只要有间隔的两个为高电平，剩下的三个位低电平，这时候就判断是起跑线。因为当车通过十字交叉的时候，不可能出现 低 高 低这样的状态。

方案总结：这种方法要求额外的增加光电管，这样就对整个电路的负担加重而且识别难度较大，优点是算法较简单。

方案二：三段黑线

这种方法的主要思想是利用采集到数据提取跳变，因为起跑线从左到右有

白 黑 白 黑 白 黑 白 那么他相应的就有从左到右三段黑线，从右到左也为三段黑线，只要检测到三次这样的跳变就判断他是起跑线。但是这只是最理想的情况下的状态，当车在告诉运行的时候不可能这么完美的检测出这样的跳变次数，而且在光线不好的时候十字交叉也容易出现这样的跳变。所以就要对这种理想的情况做一些改变。首先考虑，车未能正对着起跑线的时候，那么他一行内就没有三次跳变，可能三段黑线错行出现，或者少了一段黑线等等。其次考虑图像采集出现干扰的时候，比如在十字交叉时出现三段黑线。

我们首先讨论第一个问题，车未能正对着起跑线的情况。当出现黑线错行的时候，我们就判断他下一行是否上一行有未检测到的这段黑线，如果有就判断是起跑线，如果没有就不是，这个问题相对而言容易解决。最主要的是车在运动的时候出现干扰十字交叉时候也可能出现这样的情况，再十字交叉的时候我们可以考虑添加相对应的判断条件，比如相应的黑点个数，在十字交叉的时候黑点一般都很大，接近于采集的列数。

方案总结：这种方法相对于方案一来说容易实现些，而且识别正确率有了很大的提高，但是对算法的要求高，无识别率中等。

方案三：特征提取法

前面的方案都没有最准确的体现出起跑线的最主要特征，起跑线区别于十字交叉和别的干扰的最主要特征应该是由左侧段黑线到中间黑线的跳变，和中间黑线到右侧黑线的跳变，如图2

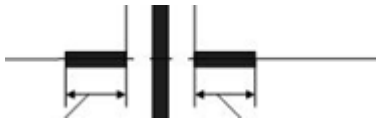


图 2

应为在正常的情况下，黑线的跳变次数为左右各一次。那么我们就在检测到跳变次数大于一的时候检测起跑线，而且在车出发5s 后在检测起跑线。

步骤如下：1 系统时间是否达到起跑线检测时间
2 检测是否为直道
3 判断跳变的次数是否达到检测要求
4 从中心向两侧确定中间那段白线
5 对白线的宽度做进一步的要求

具体如下：

因为前面已经说过起跑线的在直道上，弯道不可能出现起跑线，所以我们就检测赛道是否为直道，在弯道上不检测。在车开始5s 内不可能出现起跑线，我们在5s 内也不检测起跑线，然后当他的跳变次数超过1的时候，我们就从黑线的中心位置向两侧搜索另外一条黑线，并且记录这两次搜索的距离，这个距离应该左右大致相等，并且距离有个上限和下限。当达到这些条件的时候我们就认为他确定是起跑线。源代码如下

```
/判断停车，
if(r==42) {//判断是否为直道
  if(system_time>stop_time)//判断时间
    { if((start_flag1>=2)&&(start_flag2>=2))//首先判断跳变次数
      {
       for(ii=Line_Center[i-1];ii>0;ii--) //确定边缘
           {if (uca_Buffer1[i][ii]-uca_Buffer1[i][ii-1]>0)
```

```
                {l0=ii;break;}
                  }
          for(ii=Line_Center[i-1];ii<85;ii++) /确定边缘

                 {if (uca_Buffer1[i][ii]-uca_Buffer1[i][ii-1]<0)
                   {l1=ii;break;}
               }
           for(ii=Line_Center[i-1];ii>0;ii--) /确定边缘

               {if (uca_Buffer1[i][ii]-uca_Buffer1[i][ii-1]<0)
                  {l2=ii;break;}
                }
           for(ii=Line_Center[i-1];ii<85;ii++) /确定边缘

               if (uca_Buffer1[i][ii]-uca_Buffer1[i][ii-1]>0)
                  {l3=ii;break;}
               }

if((Line_Center[i-1]-l0<25)&&(l1-Line_Center[i-1]<25)&&(2*Line_Center[i-1]-l0-l1)<5)
   对边缘宽度的限定//
{

if((Line_Center[i-1]-l0>5)&&(l1-Line_Center[i-1]>5))//&&(2*Line_Center[i-1]-l0-l1)>=
0)
            {if((l3-l2>3)&&(l3-l2<12))
                       stop++;
                     }
                    if(stop>=1)
    {delay(150);PWMDTY2=OFF;PWMDTY3=OFF;STOPED=ON;}//条件满足停车
                   }
                 }
```

方案总结：本方案识别起跑线相对于前面两种来说，方案简单，特征明确，并且准确率高。因此我们最终选择了这种方案。

4 总结

起跑线识别的准确性体现着这个系统的准确性和稳定性。因此起跑线的识别要在整个系统都比较稳定的情况下识别。识别起跑线要在图像采集可靠的情况下进行，并且考虑误识别和漏识别的问题。但是只要图像采集合理，算法准确一定能够实现对起跑线的识别。

参考文献

[1] 卓晴，黄开胜，邵贝贝.学做智能车.北京：北京航空航天大学出版社.2007，3-1

[2] OV6620 的使用说明 ， http://shop36076594.taobao.com/ 设计及自动化研究所，10- 1

[3] 贾秀江，李颢. 摄像头黑线识别算法和赛车行驶控制策略 [R]. 上海交通大学机械

[4] 卓晴，王琎，王磊. 基于面阵 CCD 的赛道参数检测方法 [R] . 清华大学自动化系

[5]李铭泽,刘锋 ,宋晓喆. 第三届天津大学技术报告 [R] . 天津大学