第五届“飞思卡尔”杯全国大学生智能汽车竞赛

技 术 报 告

附件 B 基于摇头摄像头的图像处理算法研究

学　　校：南京师范大学

队伍名称：先驱者

参赛队员：刘逸然

张程

李昊燃

带队教师：沈世斌

关于技术报告和研究论文使用授权的说明

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：刘逸然

张　程

李昊燃

带队教师签名：沈世斌

日　　期：2010-08-14

摘要

本文以第五届全国大学生智能车竞赛为背景，利用 Freescale 半导体公司生产的 16 位单片机 HCS12 和摄像头的配合来实现自动识别道路，让赛车可以在特定的跑道上行驶。通过摄像头的成像原理，把它采集的数据通过电子线路传送到单片机，在单片机内计算出赛车面对的道路，选择最优行进路线，并采用优化后的模糊控制策略使赛车能够快速安全的行驶。此系统是一个软硬件与机械相结合的复杂整体，其中硬件主要包括电源管理模块、电机驱动模块，速度测量模块、辅助调试模块、图像采集处理模块、舵机控制模块和单片机模块等；软件需要解决单片机初始化程序、速度测量程序、速度设定程序、速度控制程序、舵机控制程序、图像识别程序等方面的内容，另外，本届比赛对机械的改进和创新也成为提高速度必不可少的组成部分。

关键词：Freescale、单片机、摄像头、机械设计

Abstract

In the background of the 5th National Intelligent Car Contest for College Students, using the 16-bit MCU HCS12 produced by Freescale Semiconductor Company and the camera to identify the road automatically, so that cars can run fast safely on the specific runway . Through the camera's imaging principle, we have the collected data transmitted to the single-chip through the electronic circuit . In the single-chip ,the car can know the road in front of it through the program that we design .And then it can choose its own path, decide the speed and ensure safe driving. The system is a complicated combination of hardware ,software and the mechanism structure adjustment, The hardware circuit include the problem about the power management module, motor driver module, the speed of measurement module, auxiliary debugging module, image acquisition and processing module, steering control modules and single-chip module . And About the software side, we need to address the single-chip initialization programs, the speed of measurement programs, the speed set-up programs, speed control programs, steering control programs, recognition of image and so on, besides improving and innovating of the mechanism structure adjustment will be one of the most important thing to improve speed in this contest.

Key words: Freescale,Single-chip, Camera, Machine design

目　　录

第一章 引言

1.1 背景介绍

现在半导体在汽车中的应用原来越普及，汽车的电子化已成为行业发展的必然趋势。受教育部高等教育司委托(教高司函[2005]201 号文)，高等学校自动化专业教学指导分委员会主办“飞思卡尔”杯全国大学生智能汽车竞赛，他以迅猛发展的汽车电子为背景，涵盖了控制、模式识别、传感技术、电子、电气、计算机、机械等多个学科交叉的科技创意性比赛。

参赛选手须使用竞赛秘书处统一指定并负责采购竞赛车模，自行采用 16 位微控制器作为核心控制单元，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动、转向舵机控制等，完成智能汽车工程制作及调试，于指定日期与地点参加场地比赛。参赛队伍之名次（成绩）由赛车现场成功完成赛道比赛时间为主，技术方案及制作工程质量评分为辅来决定。

我们学校已经参加了两届“飞思卡尔”杯智能汽车比赛，也取得了一定的成绩。在总结了往届的经验和不足后，我们在此基础上重新备战第五届大赛。往界的经验是以‘稳’为主，我们在此基础上，大胆尝试，稳中求快，在电路设计上务求稳定，问题早发现早解决。在已经成熟的技术上做的更加完善和稳定，并对车模的机械结构做出相应的改进与创新，使赛车在稳定的前提下能够更快的完成比赛。

1.2 赛车总体介绍

1.2.1 智能车技术参数

此次比赛选用的 B 型赛车车模采用 1/16 的仿真越野车模。赛车本身是四轮驱动，而机械结构也比 A 型车模更加复杂，四驱车优点明显，对于跑到的适应性高，而且稳定易于控制，但前后差速的调整较为繁琐复杂，一旦调整不平衡，就会出现滑尺、打尺等现象，使车子性能下降，无法正常运动，因此，我们的赛车选用后轮驱动方案，前轮只用于转向控制，这样只需要将后轮差速调整好就可以了，具体车模数据如表 1.1：

表 1.1 赛车主要技术参数

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）	310mm/160mm/190mm
车模轴距/轮距（毫米）	179mm/159mm
车模平均电流（匀速行驶）(毫安)	2100mA
电路电容总量（微法）	1600μF
传感器种类及个数	摄像头/1　速度传感器 /1
新增加伺服电机个数	1
赛道信息检测空间精度（毫米）	3mm（近端）、120mm(远端)
赛道信息检测频率（次/秒）	30
主要集成电路种类/数量	LM1117/3　ov7620/1 LM2576　LM393 MAX232/1　tlc5510
车模重量（带有电池）（千克）	1.2

1.2.2 **智能车硬件电路**

我们团队采用摄像头进行道路识别，赛车的硬件电路主要有七个部分组成：MC9S12XS80 最小系统板，图像采样处理模块，速度检测电路，电机驱动电路，舵机驱动模块，电源管理模块，辅助调试模块。

(1)MC9S12XS80 最小系统板是系统的核心部分，负责接收赛道图像数据，赛车速度等反馈信息，并对这些信息进行恰当的处理，形成合适的控制量来对舵机与驱动电机进行控制。

(2)图像采样处理模块由高速 AD 及摄像头组成，是智能小车的“视觉系统”，用于获得前方道路情况以供单片机处理。

(3)速度检测电路旋转编码器，高速比较器以及 S12 的 ECT 脉冲捕捉功能构成，实现道路的闭环控制。

(4)电机驱动电路使用专用 MOS 管搭建的全桥驱动，可以实现电机的正反转。

(5)舵机驱动模块控制舵机的转向。

(6)电源管理模块给整个系统供电，保障系统安全稳定运行。

(7)辅助调试模块有 BDM、串行通信、SD card 等，主要用于赛车系统的程序烧写，功能调试和测试，赛车状态监控，赛车系统参数和运行策略设置等方面。

本赛车系统的结构示意图 1.2

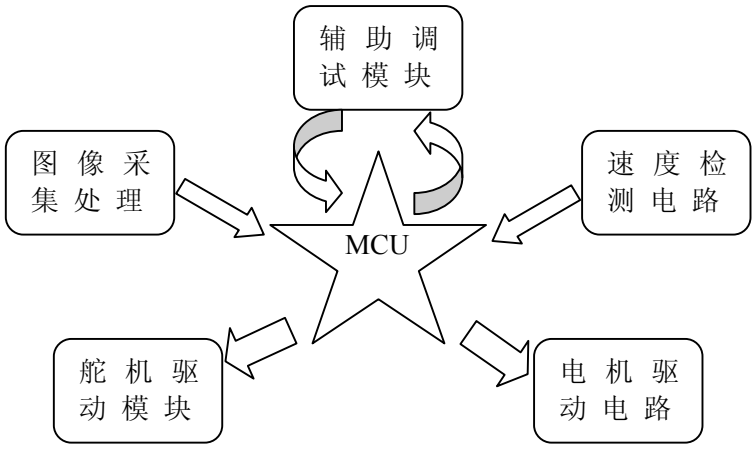


图 1.2　系统的结构示意图

1.2.1 智能车软件控制

系统硬件对于赛车来说是最基础的部分，软件算法则是赛车的核心部分。如果把一辆车和一个人做个类比的话，我们可以说，赛车的硬件结构相当于人的身体;赛车的软件算法相当于人的思想。只有“身体健康，思想进步”，才会取得好成绩。所以软件系统对于赛车来说至关重要。首先，赛车系统通过图像采

样处理模块获取前方赛道的图像数据，同时通过速度传感器模块实时获取赛车的速度。然后 S12 利用边缘检测方法从图像数据中提取赛道黑线，求得赛车于黑线位置的偏差，接着采用 PID 方法对舵机进行反馈控制，并在 PID 算法的基础上，整合加入模糊控制算法，有利于对小车系统的非线性特性因素的控制。最终赛车根据检测到的速度，结合我们的速度控制策略，对赛车速度不断进行恰当的控制调整，使赛车在符合比赛规则情况下沿赛道快速前进。设计赛车系统的软件结构如图 1.3 所示。

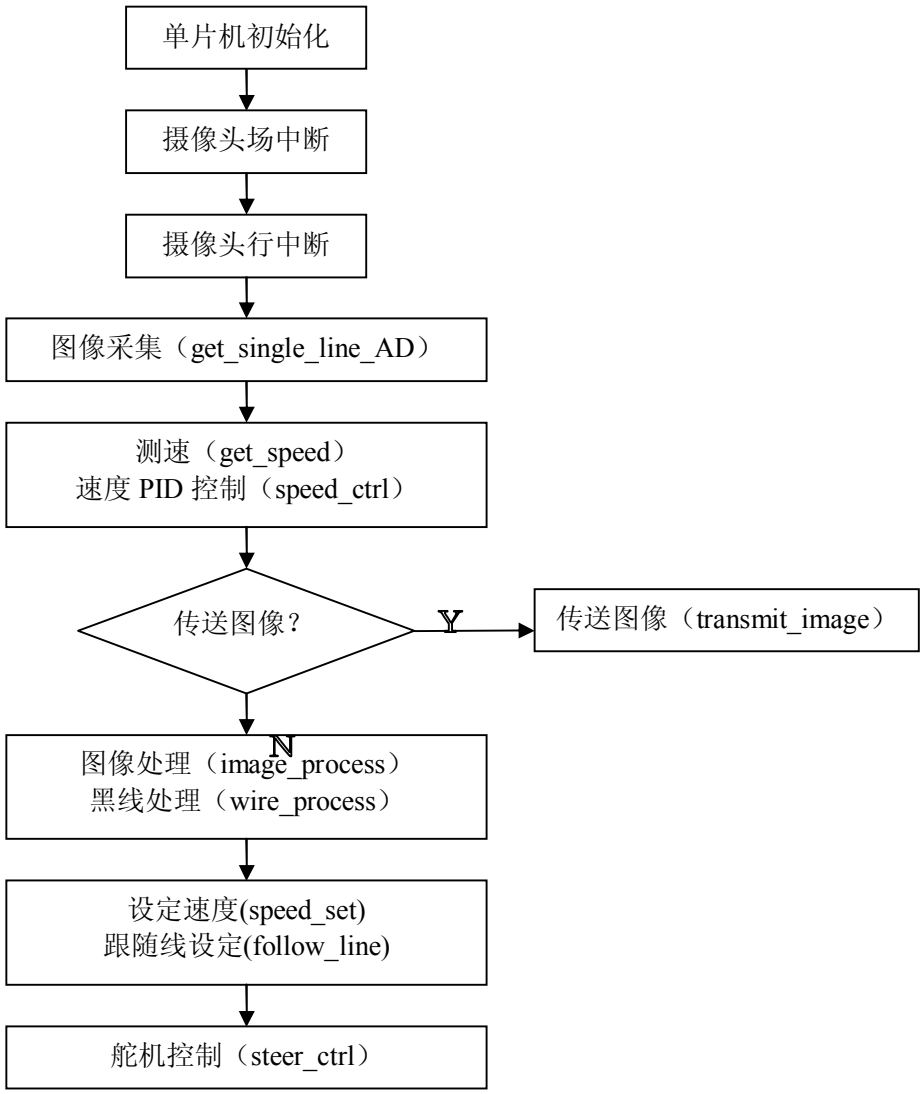


图 1.3　系统的软件方框图

1.3 本章小结

本章主要介绍了大赛的想换背景和赛车整体结构设计的概述，硬件，软件和机械部分的有效融合是赛车能否跑出好成绩的关键因素。赛车采用组委会统一提供的车模，由控制处理芯片 MC9S12XS80，图像采样模块，速度传感模块，舵机驱动模块，电机驱动模块和辅助调试模块组成，通过图像采集、黑线提取、速度控制等环节使赛车在规则下沿赛道快速前进。

第二章 赛车机械结构介绍和改进

本届比赛使用的 B 型车模是一款仿真四轮驱动车模，因此在原有车模基本调整(如前后轮调整，重心调整，传动调整等)的基础上，要想使车能够更快的行驶就必须对其本身的各种结构进行必要的改进和优化，尤其是各部分的差速的调整，更成为车模机械好坏的重要部分。

2.1 驱动选择

四驱车运行稳定，加减速明显，易于控制，而且适应性很强，是开始我们的首选。(如图 2.1)

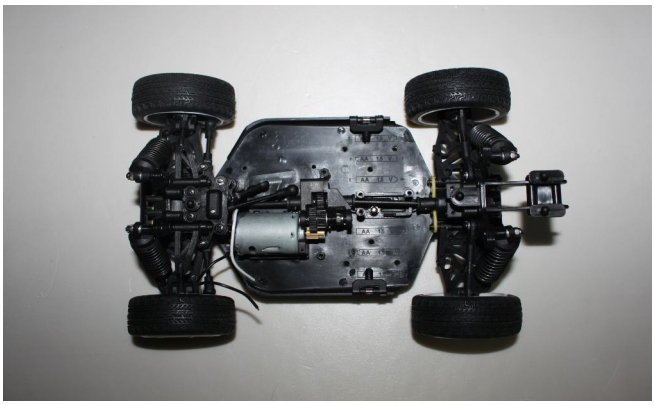


图 2.1 原装车模整体图

可是随着车子的正常调试，发现车模的前后差速阻力不同，一旦出现明显的加减速就会剧烈的噪音，如果再跑的话就会发现车模根本很难加减速了，把差速器打开后发现，齿轮已经磨秃了，根本无法传动。(如图 2.2)

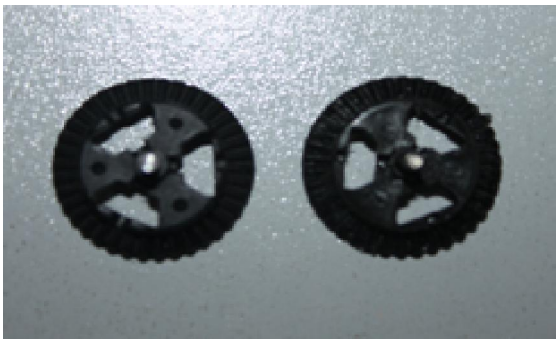


图 2.2 右边尺部分已经打光

在得知可以改成两轮驱动后，我们便尝试开始使用两轮驱动，因为此车模

的前轮保护很到位，因此，最开始我们选用的是前轮驱动，刚开始明显好了很多，转弯也更流畅了，可后来中间差速中与电机连接的部分又开始打尺，因前轮结构较为复杂，电机调整困难，故又改成后轮驱动，打尺现象有了明显的好转，故最终选择后轮驱动。

2.2 前轮调整

2.2.1 前轮调整

轮定位包括主销后倾角、主销内倾角、前轮外倾角和前轮前束四个内容。车轮定位的作用是使汽车保持稳定的直线行驶和转向轻便，并减少汽车在行驶中轮胎和转向机件的磨损。如图 2.3

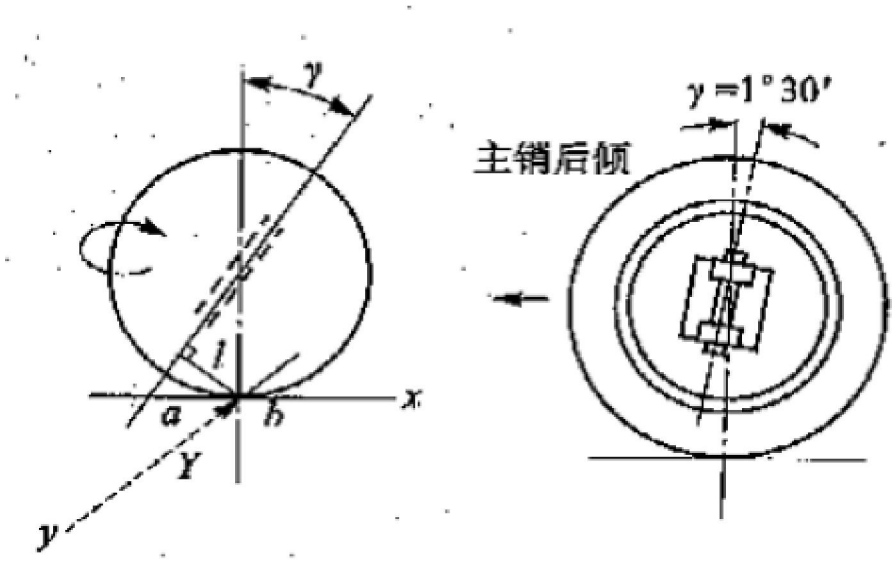


图 2.3 前轮定位原理图

（1）从车前后方向看轮胎时，主销轴向车身内侧倾斜，该角度称为主销内倾角。当车轮以主销为中心回转时，车轮的最低点将陷入路面以下，但实际上车轮下边缘不可能陷入路面以下，而是将转向车轮连同整个汽车前部向上抬起一个相应的高度，这样汽车本身的重力有使转向车轮回复到原来中间位置的因而舵机复位容易。

此外，主销内倾角还使得主销轴线与路面交点到车轮中心平面与地面交线的距离减小，从而减小转向时舵机的拉力，使转向操纵轻便，同时也可减少从转向轮传到舵机上的冲击力。但主销内倾角也不宜过大，否则加速了轮胎的磨

损。

（2）从侧面看车轮，转向主销(车轮转向时的旋转中心)向后倾倒，称为主销后倾角。设置主销后倾角后，主销中心线的接地点与车轮中心的地面投影点之间产生距离(称作主销纵倾移距，与自行车的前轮叉梁向后倾斜的原理相同)，使车轮的接地点位于转向主销延长线的后端，车轮就靠行驶中的滚动阻力被向后拉，使车轮的方向自然朝向行驶方向。设定很大的主销后倾角可提高直线行驶性能，同时主销纵倾移距也增大。主销纵倾移距过大，会使舵机沉重，而且由于路面干扰而加剧车轮的前后颠簸。

（3）从前后方向看车轮时，轮胎并非垂直安装，而是稍微倾倒呈现“八”字形张开，称为负外倾，而朝反方向张开时称正外倾。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。前轮外倾角俗称“外八字”，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。所以事先将车轮校偏一个外八字角度，这个角度约在 1°左右。

（4）脚尖向内，所谓“内八字脚”的意思，指的是左右前轮分别向内。采用这种结构目的是修正上述前轮外倾角引起的车轮向外侧转动。如前所述，由于有外倾，舵机转向变得容易。另一方面，由于车轮倾斜，左右前轮分别向外侧转动，为了修正这个问题，如果左右两轮带有向内的角度，则正负为零，左右两轮可保持直线行进，减少轮胎磨损。见图 2.3

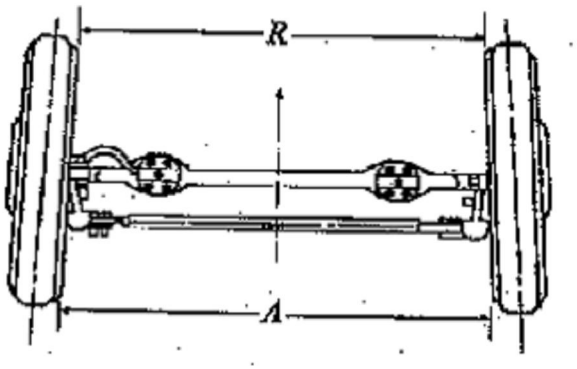


图 2.3 前轮约束示意图

2.2.2 舵机安装

舵机转向是整个控制系统中延迟较大的一个环节，为了减小此时间常数，通过改变舵机的安装位置，而并非改变舵机本身结构的方法可以提高舵机的响应速度。分析舵机控制转向轮转向的原理可以发现，在相同的舵机转向条件下，转向连杆在舵机一端的连接点离舵机轴心距离越远，转向轮转向变化越快。这相当于增大力臂长度，提高线速度。

针对上述特性，我们采用“长连杆”方式将舵机水平放在前面，大大的增加了舵机的力臂，如图 2.4

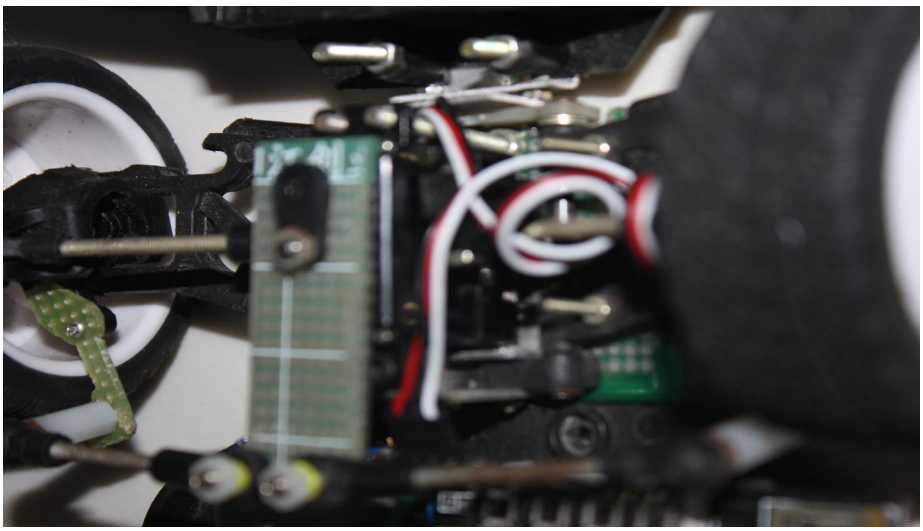


图 2.3 舵机安装方式

(1) 改变了舵机的力臂，使转向更灵敏；

(2) 舵机安装在了正中央，使左右转向基本一致；

(3) 增加前轮下压力，从而提高了前轮的抓地力，当然这样也加重了舵机负载易。

2.3 后轮调整

差速机构的作用是在车模转弯的时候，降低后轮与地面之间的滑动；并且还可以保证在轮胎抱死的情况下不会损害到电机。

当车辆在正常的过弯行进中 (假设：无转向不足亦无转向过度)，此时 4 个轮子的转速(轮速)皆不相同，依序为：外侧前轮＞外侧后轮＞内侧前轮＞内侧后轮。此次所使用车模配备的是后轮差速机构。差速器的特性是：阻力越大的一

侧，驱动齿轮的转速越低；而阻力越小的一侧，驱动齿轮的转速越高·以此次使用的后轮差速器为例，在过弯时，因外侧前轮轮胎所遇的阻力较小，轮速便较高；而内侧前轮轮胎所遇的阻力较大，轮速便较低。

后轮差速的调整主要是调整差速器中差速齿轮的咬合程度，差速的松紧与自己所要求的速度相匹配，已达到自己想要的状态。

2.4 其他调整

2.4.1 车体重心调整

车体重心位置对赛车加减速性能、转向性能和稳定性都有较大影响。重心调整主要包括重心高度和前后位置的调整。理论上，赛车重心越低稳定性越好。因此除了摄像头装得稍高以外，其他各个部件的安装高度都很低。除此之外，车辆重心前后方向的调整，对赛车行驶性能也有很大的影响。根据车辆运动学理论，车身重心前移，会增加转向，但降低转向的灵敏度（因为大部分重量压在前轮，转向负载增大），同时降低后轮的抓地力，影响加减速性能；重心后移，会减少转向，但增大转向灵敏度，后轮抓地力也会增加，提高加减速性能。因此，确定合适的车体重心，让车模更加适应比赛赛道是很关键的。今年我们赛车在车体中心位置上有了很大改革，将摄像头安装在车体靠后位置，这样使得赛车的重心后移，极大地增加了赛车的转向灵活度。

2.4.2 其他调整

除以上主要机械调整外，还包括摄像头的安装、电池的的安装、系统板的安装以及一些机械结构的微调。这主要跟所用的电路大小等有关，其中驱动电机的螺丝一定要上紧，并要经常检查，一旦在行驶中松动就会造成零件的损坏。尽量使车辆的重心放低，减少小车行驶过程中的震荡。

2.5 本章小结

本章主要介绍了赛车各个部分的机械结构。机械机构对于整车的性能至关重要，只有在较好的机械结构的前提下，控制算法才能发挥出其应有的效果。总而言之，调整舵机安装时以最大相应速度、较低负载为目标，至于放置位置则根据整车的重心调整决定，前轮和后轮差速则需在转弯性能和加减速性能之间权衡，齿轮传动机构和速度传感器安装则以较低的电机负载和适当的齿轮啮

合度为目标，除了以上部分的调整外，还可对主悬架弹簧松紧和底盘高度进行适当调整，通过增加避震弹簧的刚性、降低底盘高度、调整齿轮间隙，改善了赛车的行驶表现。

对各个部件的调整需从整车的角度考虑，不能各个击破，需要相互协调、取舍。

第三章 硬件设计

我们硬件电路总共由五大模块组成，分别为电源模块、摄像头与高速 AD 模块、电机驱动模块、通信与调参模块，速度检测模块等。其系统结构图见引言，各模块功能下面一一介绍。

3.1 电源模块

全部硬件电路的电源由 7.2V、2A/h 的可充电镍镉电池提供。由于电路中的不同电路模块所需要的工作电压和电流容量各不相同，因此电源模块应该包括多个稳压电路，将充电电池电压转换成各个模块所需要的电压。

主要包括如下不同的电压：

5V 电压。主要由电源芯片 LM1117 提供，总共用到了三块 LM1117，其在电路图中的名称为 U8、U9、U11。其中 U8 为单片机，数显模块，74LVC4245A 提供电源，U9 为摄像头级其信号处理电路 TLC5510 供电，U11 给加速度传感器和速度传感器供电。提供的电压要求稳定、噪声小，电流容量适合要求。

6.2V 电压。主要是为舵机提供工作电压，由 LM2576。实际工作时，舵机所需要的工作电流一般在几十毫安左右，电压无需十分稳定。

7.2V 电压。这部分直接取自电池两端电压，主要为后轮电机驱动模块提供电源。

18V 电压。由 34063 提供给自搭建驱动，其中的逻辑电路还需要 LM1117 提供 5V 电源。部分电源电路如图 3.1

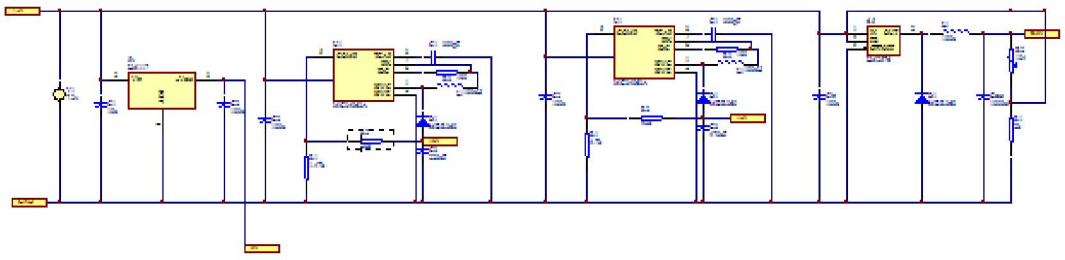


图 3.1 电源模块电路

整个电源模块的电路结构如图 3.2

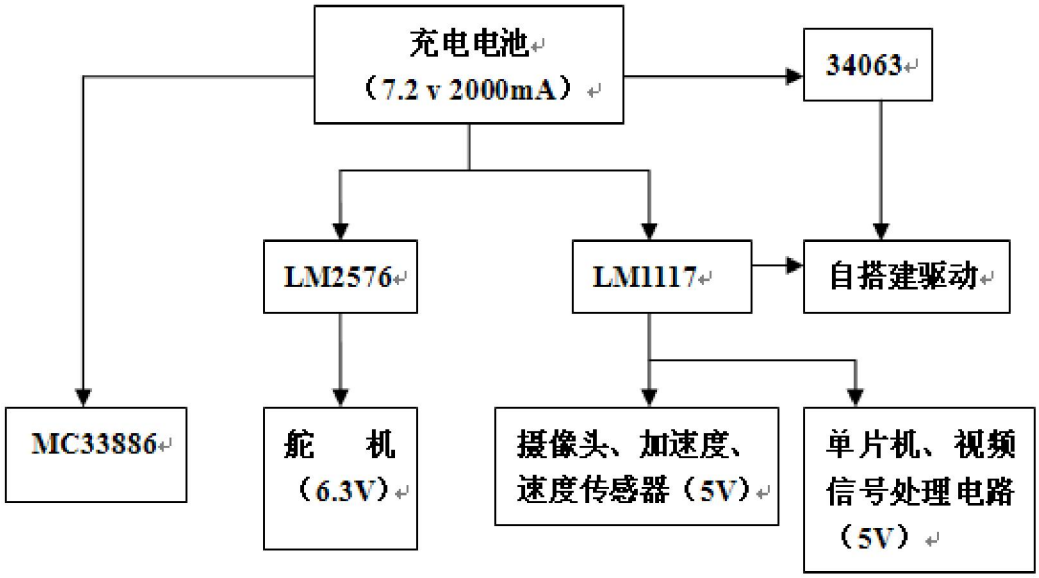


图 3.2 电源管理模块

3.2 摄像头与高速 AD 模块

3.2.1 摄像头模块

(1) 摄像头的工作原理

摄像头的工作原理是：按一定的分辨率，以隔行扫描的方式采集图像上的点，当扫描到某点时，就通过图像传感芯片将该点处图像的灰度转换成与灰度一一对应的电压值，然后将此电压值通过视频信号端输出。具体而言（参见图3.3），摄像头连续地扫描图像上的一行，则输出就是一段连续的电压信号，该电压信号的高低起伏反映了该行图像的灰度变化。当扫描完一行，视频信号端就输出一个低于最低视频信号电压的电平（如 0.3V），并保持一段时间。这样相当于，紧接着每行图像信号之后会有一个电压“凹槽”，此“凹槽”叫做行同步脉冲，它是扫描换行的标志。然后，跳过一行后（因为摄像头是隔行扫描的），开始扫描新的一行，如此下去，直到扫描完该场的视频信号，接着又会出现一段场消隐区。该区中有若干个复合消隐脉冲，其中有个远宽于（即持续时间长于）其它的消隐脉冲，称为场同步脉冲，它是扫描换场的标志。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。

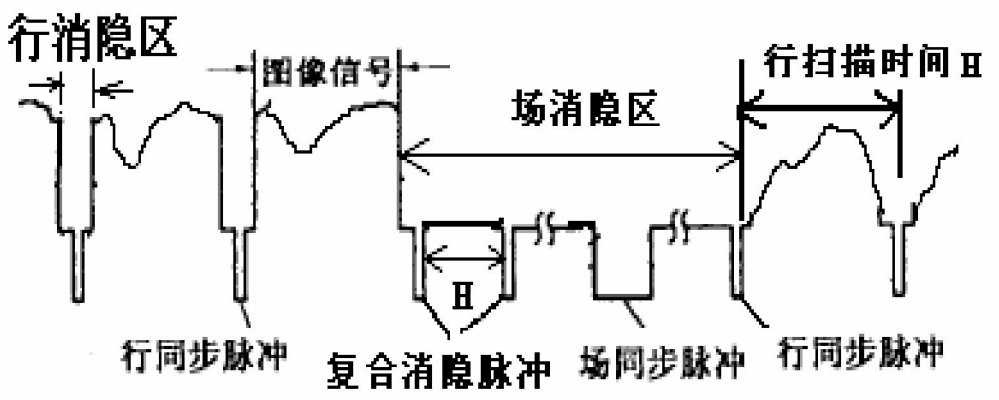


图 3.3　摄像头视频信号

摄像头有两个重要的指标：有效像素和分辨率。分辨率实际上就是每场行同步脉冲数，这是因为行同步脉冲数越多，则对每场图像扫描的行数也越多。事实上，分辨率反映的是摄像头的纵向分辨能力。有效像素常写成两数相乘的形式，如“320x240”，其中前一个数值表示单行视频信号的精细程度，即行分辨能力；后一个数值为分辨率，因而有效像素=行分辨能力×分辨率。

值得注意的是，通常产品说明上标注的分辨率不是等于实际分辨率（即每场行同步脉冲数），而是等于每场行同步脉冲数加上消隐脉冲数之和。因此，产品说明上标注的“分辨率”略大于实际分辨率。我们要知道实际的分辨率，就得实际测量一下。

通过 S12 单片机的定时器模块对单个脉冲的下降沿和上升沿间隔、两相邻脉冲上升沿间隔进行计时，可得每行信号和每个脉冲持续的时间。实际测得所用摄像头（1/3 Omni Vision CMOS）的时序参数见表 3.1。

从测得的结果可知，该摄像头扫描的每场中有 320 行信号，其中第 23 行到 310 行是视频信号，第 311 行到下一场的第 22 行是场消隐信号。在视频信号区，每行信号持续的时间相同，约为 62us；每行的行同步脉冲持续时间也相同，约为 4.7us。而在场消隐区，每行持续的时间会有所变化，每行对应的消隐

脉冲持续的时间，尽管其中大多数为 3.5us，但也有变化。在场消隐区中，第 320 行的消隐脉冲持续的时间远长于其他消隐脉冲的时间，此脉冲即为场同步脉冲。

表 3.1 摄像头时序参数

信号属性	行序数	行持续时间	行同步脉冲持续时间	消隐脉冲持续时间
场消隐区	1-4	23us		3.5us
	5	27.3us		8us
	6	37.3us		3.5us
	7-10	29.8us		3.5us
	11-22	62us		4.7us
视频信号区	23-310	62us	4.7us	
场消隐区（场同步脉冲）	311-314	62us		4.7us
	315	62us		3.5us
	316-319	29.8us		3.5us
	320	53.4us		28us

(2) 模拟摄像头视频分离电路设计

要能有效地对视频信号进行采样，首先要处理好的问题是如何提取出摄像头信号中的行同步脉冲、消隐脉冲和场同步脉冲。这里有两种可行的方法。第一，直接通过单片机 AD 进行提取。因为行同步脉冲、消隐脉冲或场同步脉冲信号的电平低于这些脉冲以外摄像头信号的电平，所以据此可设定一个信号电平阈值来判断 AD 采样到的信号是否为上述三类脉冲。第二，就是给单片机配以合适的外围芯片，此芯片要能够提取出摄像头信号的行同步脉冲、消隐脉冲和场同步脉冲以供单片机作控制之用。

考虑到单片机的速度有限，而一些脉冲的间隔时间又较短，同时为了减轻其处理负担，我们采用了第二种方法进行信号提取。LM1881 视频同步信号分离芯片（简称 LM1881）可从摄像头信号中提取信号的时序信息，如行同步脉冲、场同步脉冲和奇、偶场信息等，并将它们转换成 TTL 电平直接输给单片机的 I/O 口作控制信号之用。LM1881 的端口接线方式如图 3.4 所示。

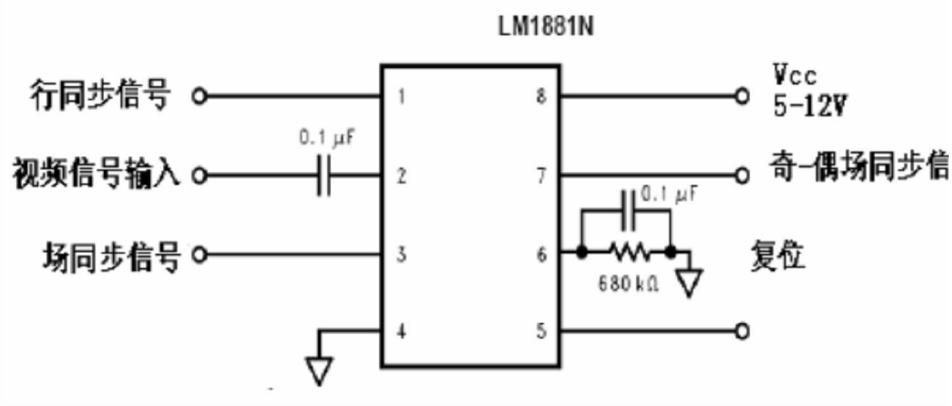


图 3.4 LM1881 连接图

其中，引脚 2 为视频信号输入端，引脚 1 为行同步信号输出端（如图 3.5 中的 b）。引脚 3 为场同步信号输出端，当摄像头信号的场同步脉冲到来时，该

端将变为低电平，一般维持 230us，然后重新变回高电平（如图 3.5 中的 c）。引脚 7 为奇.偶场同步信号输出端，当摄像头信号处于奇场时，该端为高电平，当处于偶场时，为低电平。事实上，不仅可以用场同步信号作为换场的标志，也可以用奇.偶场间的交替作为换场的标志。

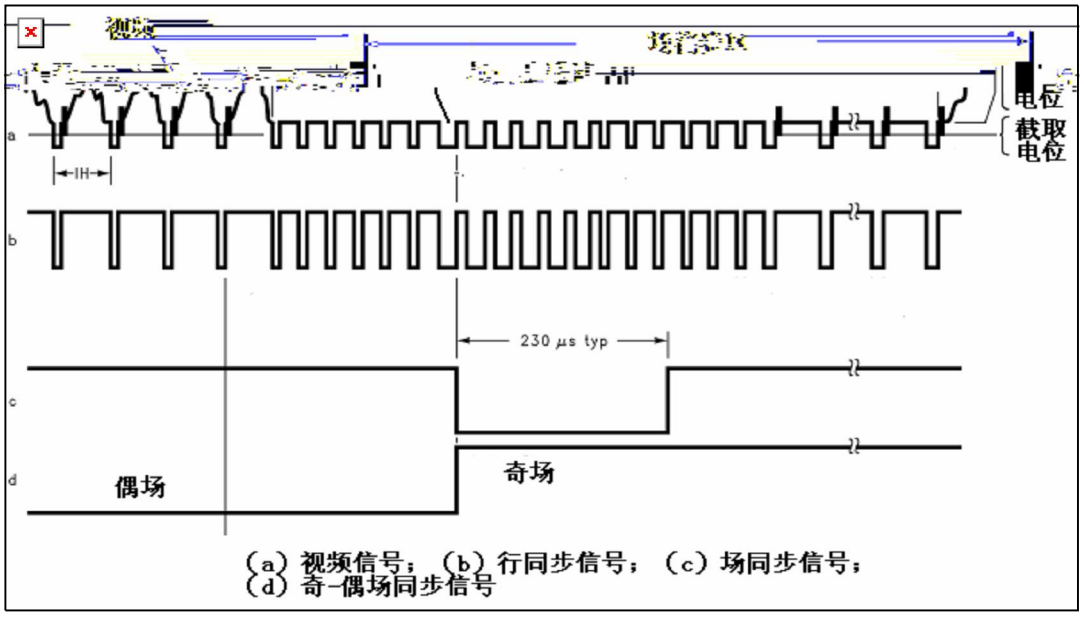


图 3.5 LM1881 信号时序图

由 LM1881 及其外围电路构成的摄像头采样电路如图 3.6 所示。摄像头视

频信号端接 LM1881 的视频信号输入端，同时也接入 S12 的一路 AD 转换端口（选用 AD0）。LM1881 的行同步信号端（引脚 1）通过跳线选择，分别接入 ECT 模块中的 PT0 和外部中断引脚（IRQ），通过跳线选择，既可以采用脉冲捕捉方式获取行同步信号，也可以采用硬件中断获取。同时将 LM1881 的场同步信号和奇.偶场同步信号也输入到 ECT 模块中（选用 PT1，PT2），这样，既可以采用查询方式获取奇偶场信号跳变，又可以采用脉冲捕捉方式获取电平变化。通过这样的接线，为软件开发提供了多种选择机会，使程序更加灵活。

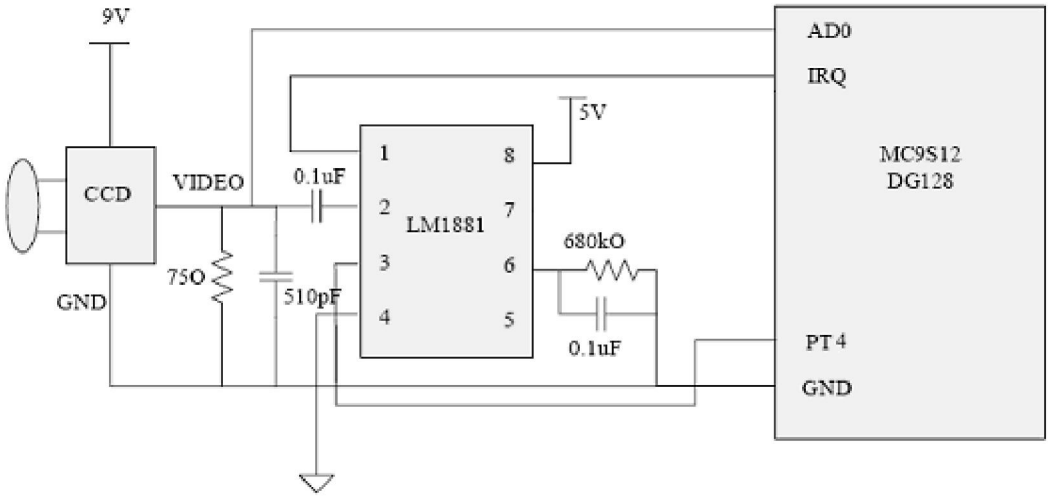


图 3.6 摄像头采样电路图

(3) 实际选择摄像头

针对模拟摄像头的视频分离电路过于繁琐，我们采用的是使用 OV7620 芯片的摄像头，直接将摄像头行场中断信号和单片机口连接，直接去掉了视频信号的视频分离电路，简化了电路的同时，减少了外部干扰的几率。

3.2.2 高速 AD 模块

摄像头获得的视频信号为模拟信号，而单片机处理的却是数字信号，所以由必要把视频模拟信号离散成数字信号，这就要用到模数转换技术。一般而言单片机都有 A/D 转换模块。我们所用单片机为组委会推荐使用的新一代单片机 mc9s12xs80。S12 内置了 2 个 10 位/8 位的 A/D 模块 ATD0 和 ATD1，通称为模数/转换器(ATD)。但是 S12 微控制器 ATD 的最高转换频率只有约为 2MHz，不能够满足我们的要求，所以我们自己设计了外部高速 AD 将摄像头的模拟信号先转换位数字信号后再通过 I/O 口送单片机内部进行数据处理。

我们采用 8 位高速 A/D 转换器 TLC5510 搭建。

TLC5510 引脚说明及工作原理

(1)TLC5510 的引脚说明

TLC5510 为 24 引脚、PSOP 表封装形式(NS)。其引脚排列如图 3.1 所示。

各引脚功能如下：

AGND：模拟信号地；

ANALOG IN：模拟信号输入端；

CLK：时钟输入端；

DGND：数字信号地；

D1～D8：数据输出端口。Ｄ１为数据最低位，Ｄ８为最高位；

OE：输出使能端。当 OE 为低时，D1～D8 数据有效，当 OE 为高时，D1～D8 为高阻抗；

VDDA：模拟电路工作电源；

VDDD：数字电路工作电源；

REFTS ：内部参考电压引出端之一，当使用内部电压分压器产生额定的 2V 基准电压时，此端短路至ＲＥＦＴ端；

REFT：参考电压引出端之二；

REFB：参考电压引出端之三；

REFBS：内部参考电压引出端之四，当使用内部电压基准器产生额定的 2V 基准电压时，此端短路至 REFB 端。

具体的芯片管脚图见图 3.7

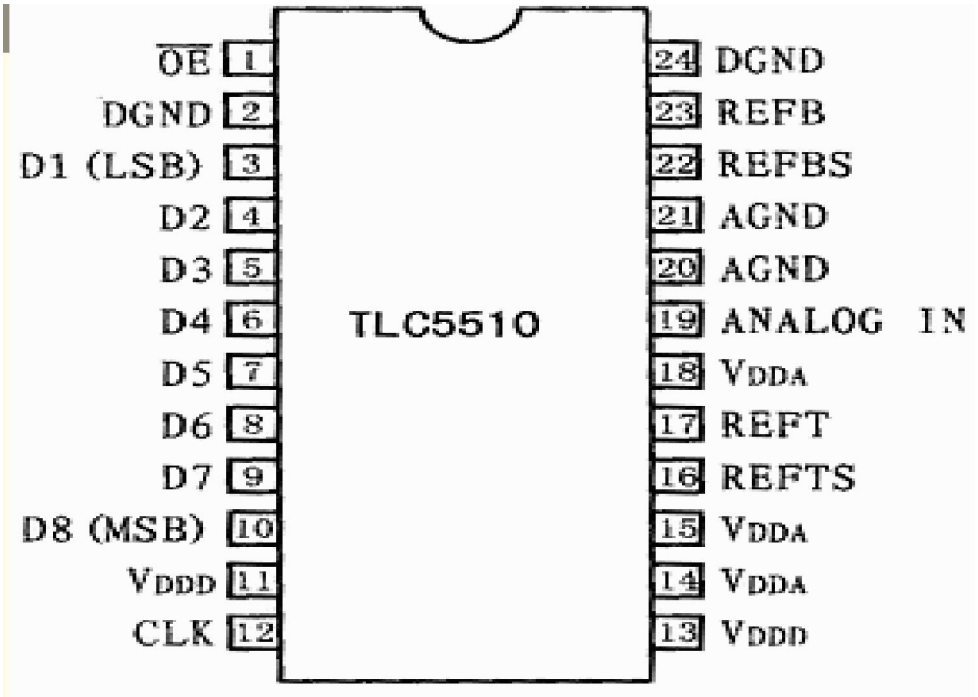


图 3.7 TLC5510 的引脚排列

(2) TLC5510 工作原理

TLC5510 的工作时序见图 3.8。时钟信号 CLK 在每一个下降沿采集模拟输入信号。第N次采集的数据经过 2.5 个时钟周期的延迟之后，将送到内部数据总线上。

在图 3.8 所示的工作时序的控制下，当第一个时钟周期的下降沿到来时，模拟输入电压将被采样到高比较器块和低比较器块，高比较器块在第二个时钟周期的上升沿最后确定高位数据，同时，低基准电压产生与高位数据相应的电压。低比较块在第三个时钟周期的上升沿的最后确定低位数据。高位数据和低位数据在第四个时钟周期的上升沿进行组合，这样，第N次采集的数据经过 2 2.5 个时钟周期的延迟之后，便可送到内部数据总线上。此时如果输出使能 OE 有效，则数据便可被送至8位数据总线上。由于CLK的最大周期为50ns，因此，TLC5510数模转换器的最小采样速率可以达到 20MSPS。

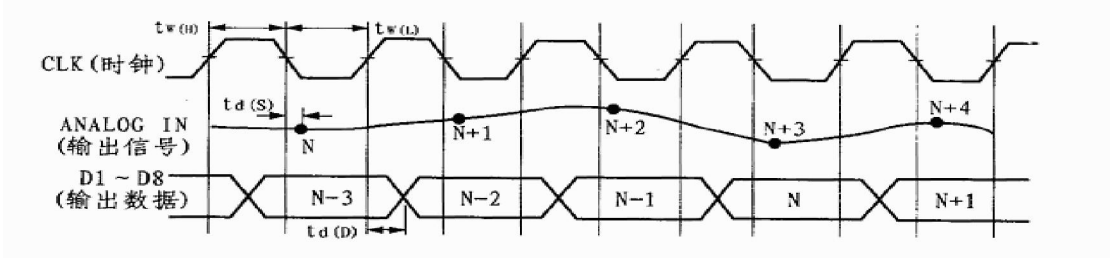


图 3.8 TLC5510 工作时序

(3)TLC5510 外围电路

a 为了减少系统噪声，外部模拟和数字电路应当分离，并应尽可能屏蔽。

b 因为 TLC5510 芯片的 AGND 和 DGND 在内部没有连接，所以，这些引脚需要在外部进行连接为了使拾取到的噪声最小，最好把隔开的双绞线电缆用于电源线。同时，在印制电路板布局上还应当使用模拟和数字地平面。

c VDDA 至 AGND 和 VDDD 至 DGND 之间应当分别用 1uF 电容去耦，推荐使用陶瓷电容器。对于模拟和数字地，为了保证无固态噪声的接地连接，试验时应当小心。

d VDDA、AGND 以及 ANALOG IN 引脚应当与高频引脚 CLK 和 D0～D7 隔离开。在印制电路板上，AGND 的走线应当尽可能地放在 ANALOG IN 走线的两侧以供屏蔽之用。

e 为了保证 TLC5510 的工作性能，系统电源最好不要采用开关电源。

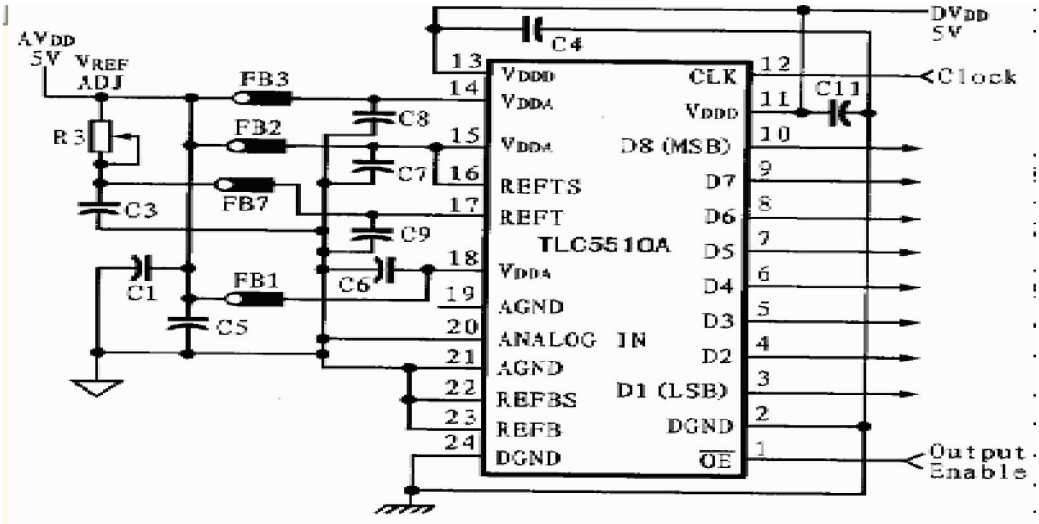


图 3.9 TLC5510 外围电路

3.3 电机驱动模块

在电机驱动上，我们用 MOS 管作为分立元件搭建了 H 桥驱动电路如图 3.8。通过逻辑设计，可以让电机处于多种模式下工作，经过在赛道上对赛车进行试验，电机的加减速效果很好，完全可以满足赛车对不同赛道加减速的要求。

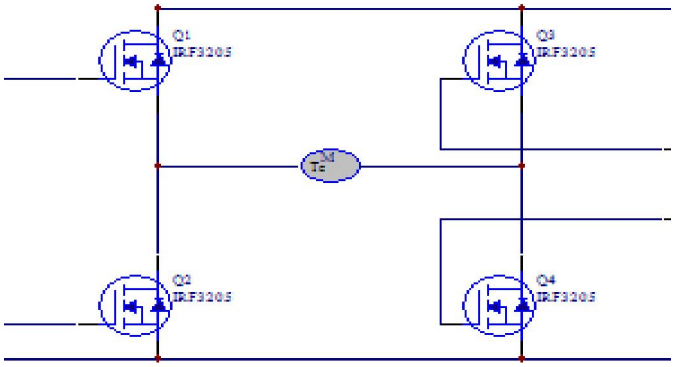


图 3.10 H 桥

在搭建的过程中我们试过了很多的前级驱动芯片，如 IR2110，33883，4427 等，MOS 管也用过了 IRF3205、IRF1404、IRF2804 等，通过上面各芯片的 datesheet，我们最后选择了 4427 和 IRF2804 的组合。图 3.11 为 4427 的引脚功能图，图 3.12 为其内部原理图。

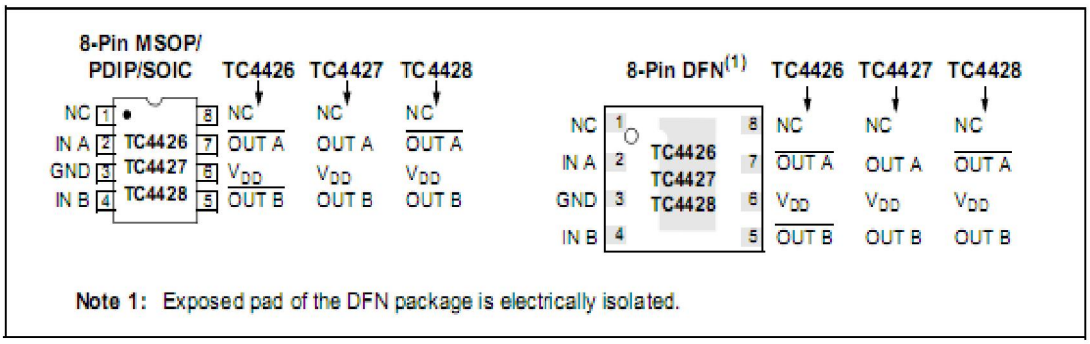
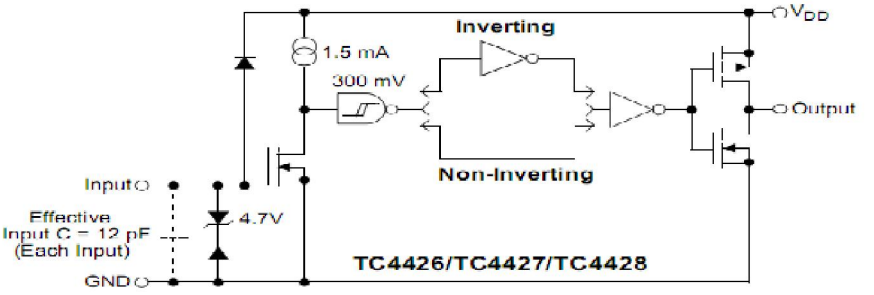


图 3.10　4427 的引脚功能图

Functional Block Diagram



Note 1: TC4426 has two inverting drivers, while the TC4427 has two non-inverting drivers. The TC4428 has one inverting and one non-inverting driver.

2: Ground any unused driver input.

3.11　4427 内部原理图

从图 3.9 可以很轻易知道 4427 有两路信号通道，A 通道和 B 通道，而且是完全对等的，而图 3.9 中的 MOS 管的关断与否要四各信号来控制，所以我们可以使用两片 4427 来控制，由于信号通道是完全对等的，两片 4427 共四路信号可以控制任意一个 MOS 管。对于 MOS 管的选择原则是尽可能的选择导通内阻低的，这样可以避免管子发热严重的现象。

在设计初期我们同时考虑到应避免图 3.10 中 Q1 和 Q2、Q3 和 Q4 同时导通的问题，本质上来说应该避免电源和地短路的问题，对于此，参考了大量资料后我们觉的可行的办法就是在前级驱动电路前面加上逻辑保护电路。实际中控制中我们用到了一路 PWM 口和一路 I/O 口来控制驱动电路。所以逻辑保护电路应该能够达成如下表 3.2 的控制效果。

表 3.11 驱动电路的控制逻辑

PWM	I/O	导通情况	效果
0	1	Q2、Q4 导通	电磁阻尼
1	1	Q2、Q3 导通	反转
0	0	Q1、Q3 导通	电磁阻尼
1	0	Q1、Q4 导通	正转

驱动电路从设计到制板在到焊接使用前后花了两个多月时间，值得高兴的是，制作出来的驱动电路驱动性能的确够好，也够强劲。图 3.12 为搭建驱动电路的电路原理图，以飨读者。

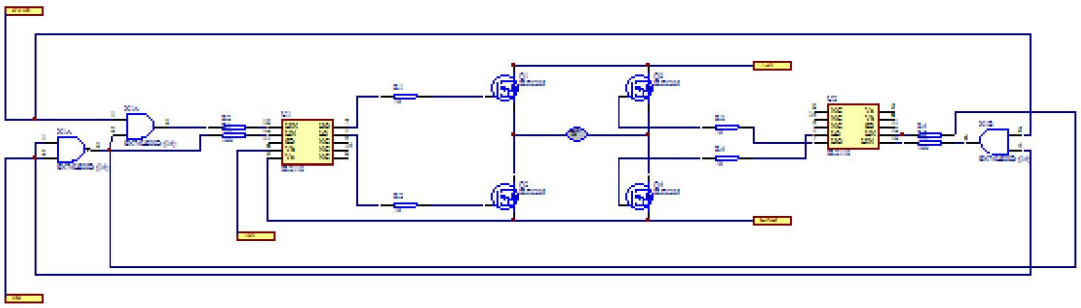


图 3.12　驱动电路图

3.4 通信与调参模块

3.4.1 通信模块

(1)串口通信模块

串行通信是微控制器与外围设备或其他计算机之间通信的一种重要渠道，计算机或微控制器与外界进行数据交换称为通信。我们所用 S12 串行通信模块包含多个独立的串行子系统，故我们可以进行 S12 和计算机之间的通信，进行在线调试。图 3.12 为串行通信模块电路图。其中 MAX232 的 9、10 两脚分别与 S12 的 PS0、PS1 连接。图 3.13 为 MAX232 引脚图。

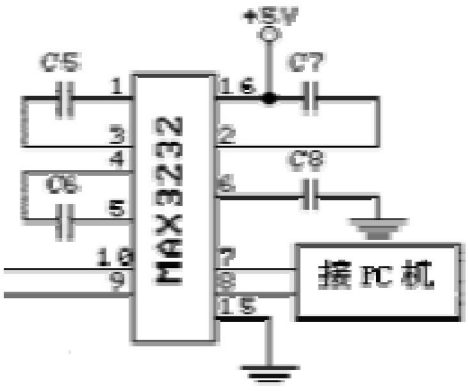


图 3.12　串口通讯电路

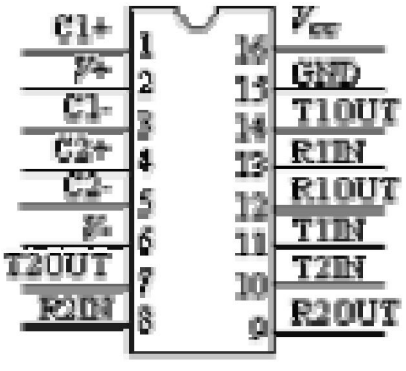


图 3.13　MAX232 引脚图

(2)SD 卡记录设备

在前几届的比赛中，大多数队伍均是通过串口将S12 的数据传到电脑中进行分析，便于识时进行调试。但这只能在静止条件下捕获赛车所采集的信息情况，难以辨知赛车在赛道中跑的时候所采集到的赛道信息是什么样的。

为了解决此难题，我们成功地研制出了SD 卡记录功能。SD 卡用过SPI 与S12 进行通信。通过SD 卡可以讲车模运动时的每一幅图片存储下来，以便事后分析，发现错误。

不过我们的 SD 卡记录系统还存在一些不足的地方，比如由于 S12 的限制，在使用 SD 卡的时候，会增加单片机的工作负担，从而无法跑出和没有 SD 卡时一样的速度。

3.4.2 调参模块

对于用来竞赛的车模必须要有很强的适应性，这就在要求你控制算法适应性强的同时，还可以适当的改变一些算法中的可调参数使车模能够很好的适应新的赛场环境。特别是在赛前的副管调试时，可以根据跑道所用材料的摩擦系数来适当的确定车模在接下来的比赛中所用参数。总之，一辆好的车模往往具有很强的可调性，这就需要确定好方便调参电路和必要的软件程序的支持。针对于此，设计初期，我们参考了上届其他大学的报告，有了几种方案。

(1) 使用带旋柄电位器对合适的电压进行分压，用 S12 的 ATD 模块对分得电压值采样。可以用 S12 的一个 A/D 口就可读入一个旋转信号，可以用它来等比例调节车速。

(2) 使用使用键盘阵列配合拨码盘和 LED。此方法占用单片机 I/O 口，而且外围电路过于大，模块化以后占车模面积也太大，所以没有采用。

(3) 旋转开关配合 LED。旋转开关在接好外围电路后，在正反转使产生不同脉冲信号，送入单片机进行处理，根据需要可以设置不同的可调参数。

在实际的使用中我们选择了第三种方法，它理论上可调的参数为九十九个，足够用来对车模的参数进行调节，LED 显示构成人机界面。如图 3.14 为旋转开关电路图，3.15 为人机界面电路图。

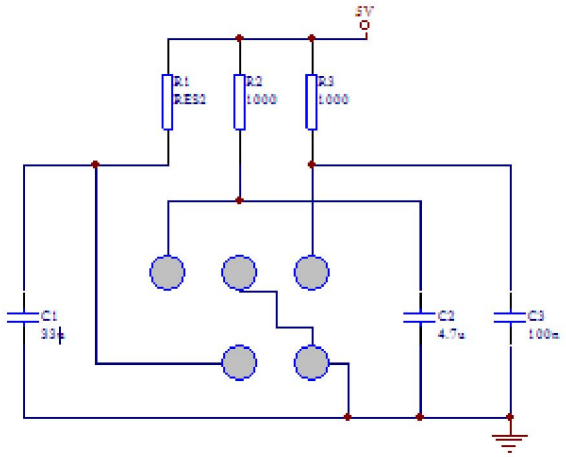


图 3.14 旋转开关电路图

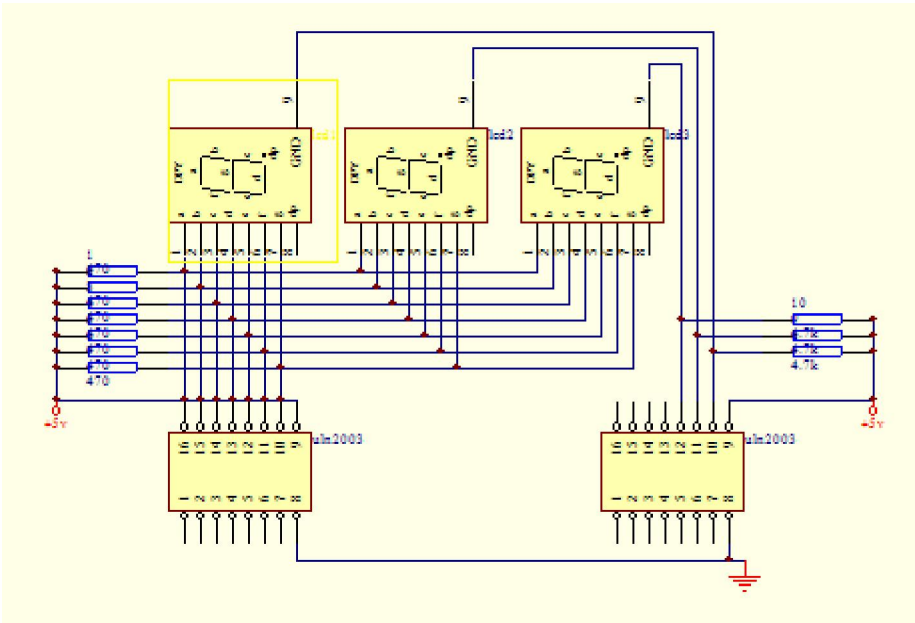


图 3.15 人机界面电路图

3.5 速度测量模块

为了使得赛车能够平稳地沿着赛道运行，需要控制车速，使赛车在急转弯时速度不至过快而冲出赛道。通过控制驱动电机上的平均电压可以控制车速，但是如果开环控制电机转速，会受很多因素影响，例如电池电压、电机传动摩擦力、道路摩擦力和前轮转向角度等。这些因素会造成赛车运行不稳定。通过速度检测，对车模速度进行闭环反馈控制，即可消除上述各种因素的影响，使

得车模运行得更稳定。

车速检测的方式有很多种，例如用测速发电机、转角编码盘、反射式光电检测、透射式光电检测和霍尔传感器检测。经过对去年测速方案和其它学校方案的比较，本次设计中速度传感器采用的是 OMRON 公司生产的 E6A2-CS100 型光电编码器。它由 5-12V 的直流供电，速度传感器用螺钉固定在塑料片上，塑料片固定在后轮支架上，这样固定好之后，就有了较高的稳定性。速度传感器通过后轮轴上的齿轮与电机相连，车轮每转一圈，速度传感器转过 2.75 圈。这样能够很好的稳定工作，且能很准确的得到电机的转速，比较可靠。

3.6 本章小结

本章主要介绍了整辆车的硬件电路部分，如果说机械结构是赛车的手足的话，那么硬件电路就是它的感官和大脑，所有的反馈都是由它获得，所有的图像处理和控制也经由它处理。硬件设计以稳定为主，特别是核心控制板，因为为了要提高指令执行速度，单片机是通过锁相环超频使用，所以时钟模块的电路设计应尽量避免电磁干扰。驱动电路板设计则应以小内阻、大电流为目标，以尽可能提高整车的加减速性能

第四章 软件设计

4.1 单片机简介

MC9S12XS80 微控制单元作为 MC9S12 系列的 16 位单片机，由标准片上外围设备组成，包括 16 位中央处理器、128KB 的 Flash 存储器、8KB 的 RAM、2KB 的 EEPROM、两个异步串行通信接口、两个串行外围接口、一组 8 通道的输入捕捉或输出捕捉的增强型捕捉定时器、两组 8 通道 10 路模数转换器、一组 8 通道脉宽调制模块、一个字节数据链路控制器、29 路独立的数字 I/O 接口、20 路带中断和唤醒功能的数字 I/O 接口、5 个增强型 CAN 总线接口，同时，单片机内的锁相环电路可使能耗和性能适应具体操作的需要。S12 的片内资源如图 4.1 所示。

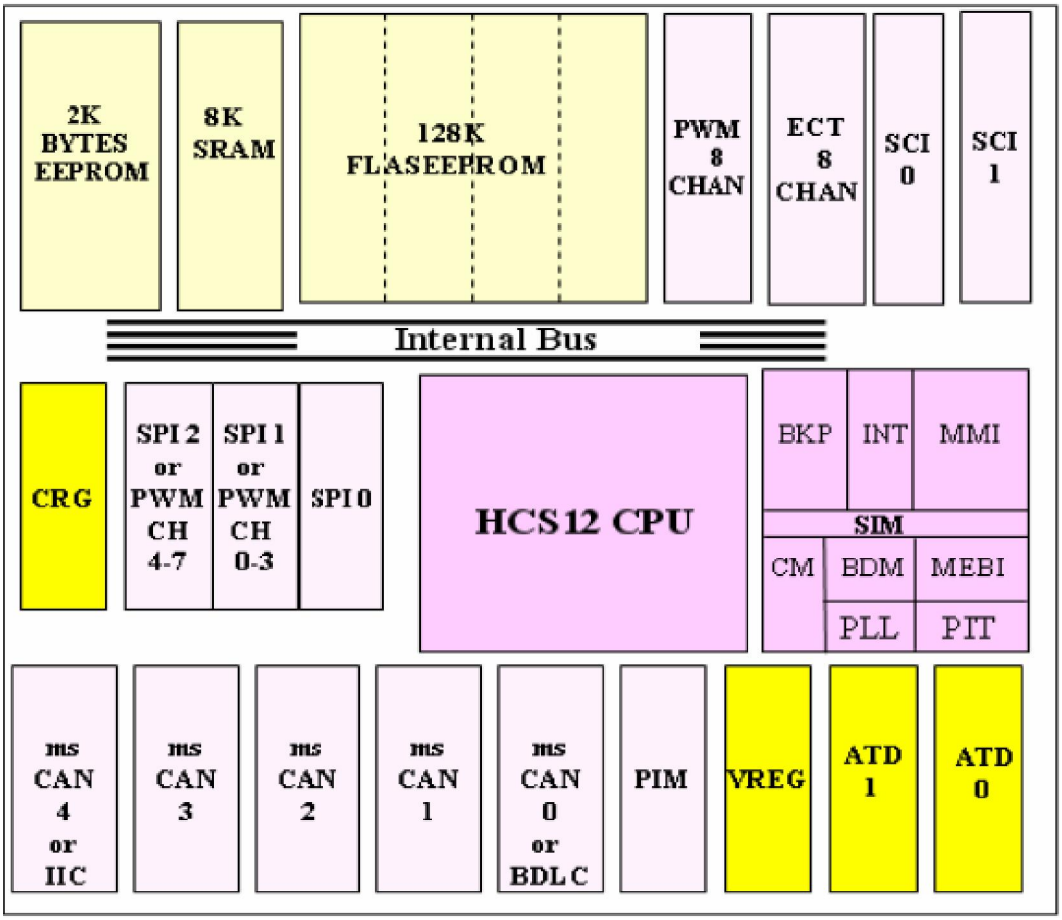


图 4.1 片内资源

具备了以上这些卓越的性能，MC9S12DG128 完全能够满足我们设计的需要，

并使得电路更加紧凑。以下将首先介绍系统设计中用到的各个软件功能模块的设置，然后再讨论黑线提取及车体控制算法。

4.2 系统的软件方框图

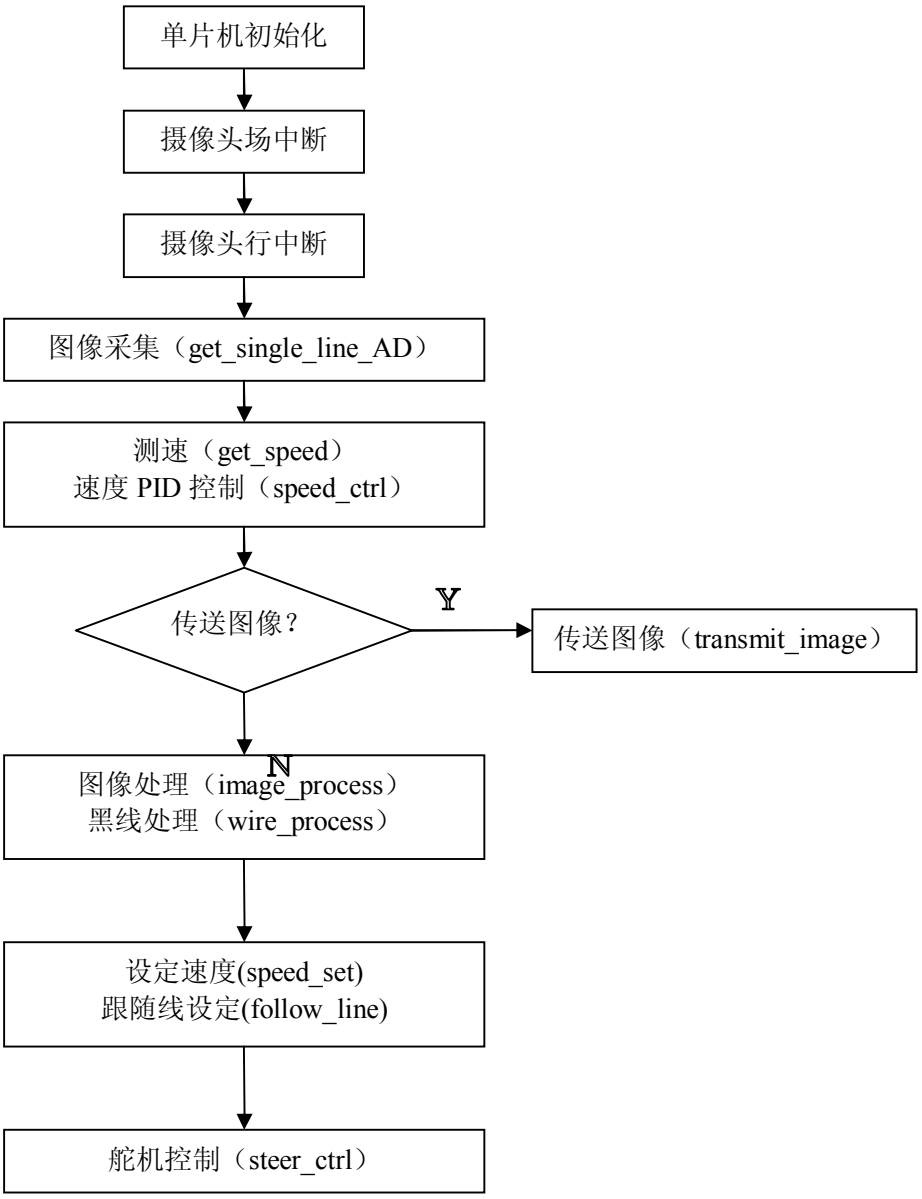


图 4.2 系统软件方框图

4.3 系统初始化

单片机要正常工作，必须对它进行初始化，对于 MCS12DG128B 单片机来说，

初始化的部分主要有一下几部分：总线时钟初始化,PWM 模块初始化，ECT 模块初始化，I/O 口初始化,SCI 模块初始化，A/D 转化初始化。下面我就分别介绍一下各个模块的初始化程序。

4.3.1 总线时钟初始化

和总线时钟初始化相关的寄存器主要有时钟合成寄存器 SYNR 和时钟分频寄存器 REFDV。锁相环产生的时钟频率 F=2*f*（SYNR+1）/(REFDV+1),式中 f 是振荡器的频率，SYNR 为示踪合成寄存器的值，REFDV 为时钟分频寄存器的值。下面就是总线时钟设定程序：

```
void PLL_init(void)                    //设定总线时钟 40MHz
{
SYNR=0X09;
REFDV=0X03;
while(CRGFLG_LOCK==0);
CLKSEL=0x80;
}
```

4.3.2 摄像头初始化

由于摄像头行同步信号接到了 S12 单片机的输入捕捉 PT1 引脚，所以当每行视频信号到来时，会产生一个中断,进入采集该行。

摄像头的初始化包括初始化中断，具体过程为：

```
TIE=0X02; //中断IC1使能
INTCR_IRQEN = 0; //清除外部中断标志
ATD0DIEN=0xFF; //设置 AN 为数字输入
```

4.3.3 PWM 模块初始化

脉宽调制（PWM）模块有 8 路独立的可设置周期和占空比的 8 位 PWM 通道，每个通道配有专门的计数器。该模块有 4 个时钟源，能分别控制 8 路信号。通

过配置寄存器可设置 PWM 的使能与否、每个通道的工作脉冲极性、每个通道输出的对齐方式、时钟源以及使用方式（单独输出还是合并输出）。

为了提高控制精度，将 PWM0、PWM1 两路 8 位通道合并为一个 16 位通道来控制舵机，这样可使舵机的控制精度从 1/255 提高到 1/65536。PWM 模块的初始化设置过程为：

（1）选择时钟　　PWMPRCLK 、 PWMSCLA 、 PWMSCLB 、 PWMCLK

（2）选择极性　　PWMPOL

（3）选择对齐模式　　PWMCAE

（4）对占空比和周期编程 PWMDTYx、PWMPERx

（5）使能 PWM 通道　　PWME

下面是 PWM 模块初始化程序：

```
void PWM_init(void){                    //PWM 初始化
PWMPRCLK=0x12;                  //A clock 10MHz ,  B clock 20MHz
PWMSCLA=0x01;                   //SA   5MHz=(10/(2*1))
PWMSCLB=0x01;                   //SB   10MHz=(20/(2*1))
PWMCLK=0xFF;                    //选择 ClockSA,ClockSB 时钟
PWMPOL_PPOL1=1;                 //周期开始，PWM 输出高电平
PWMPOL_PPOL7=1;
PWMPOL_PPOL3=1;
PWMCAE=0xff;                    //对齐方式为输出中心对齐
PWMCTL=0xb0;                    //通道 01，23，67 级联
PWMPER01=50000;                 //PWM 占空比初始化
PWMDTY01=steer_MIDDLE;
PWMPER23=1000;
```

```
PWMDTY23=0;
PWMPER67=1000;
PWMDTY67=0;
PWME=0xFF;                          //启动 PWM，下一个时钟输出 PWM
}
```

通过 PWM 初始化，我先对总线时钟进行了预分频，把 ClockA 的时钟设定为 10MHz=40MHz/4,ClockB 的时钟设定为 20MHz=40MHz/2。然后对 ClockA 和 ClockB 进行进一步分频，得到 ClockSA 的时钟 5MHz=10 MHz /(2*1)，ClockSB 的时钟 10 MHz=20 MHz /(2*1)。极性设定为 1，也就是周期开始时，PWM 输出为高电平 ；对齐方式为输出中心对齐方式，所以就得到占空比的计算公式：

占空比=【PWMDTYx/PWMPERx】*100%

4.3.4 I/O 模块初始化

S12MCU 的每个端口除可以作为通用 I/O 外，还分别具有专用 I/O 功能，可以根据需要选择其一。有的端口启动专用功能后，通用 I/O 功能自动关闭；有的端口专用功能只占用部分引脚，其余引脚的 I/O 功能仍然可以正常工作；此外，有的引脚仅具有单向输入或输出功能。具体情况取决于专用功能的特殊要求以及对 I/O 的设置。

下面是 I/O 初始化的程序：

```
void IO_init(void){                          //IO 口初始化
DDRB=0xff;                                   //设定 PORTB 为输出
PORTB=0xff;                                  //端口 PORTB 输出高电平
 DDRA=0x00;                                  //设定 PORTA 为输入
PUCR_PUPAE=0;                                //A 口的上拉电阻无效
DDRM=0x00;                                   //设定 PORTM 为输入
PERM=0x00;                                   //M 口的上拉电阻无效
DDRE=0x00;                                   //设定 PORTE 为输入
```

```
PUCR_PUPAE=0;                          //E 口的上拉电阻无效
DDRJ_DDRJ7=1;                          //设定 PORTJ_7 为输出
motor=1;                               //enable 33886
DDRH=0x00;                             //设定 PORTH 为输入
}
```

通过 I/O 模块初始化程序，完成 A 口,B 口，E 口,M 口，H 口以及 J 口的第七位的输入输出方式设定，同时也给 PORTB 设定了初值。

4.3.5 SCI 模块初始化

SCI 是一种采用标准的不归零数据 NRZ 格式的异步串行通信接口。S12MCU 的 SCI 的功能特点是：双线串行接口；标准 NRZ 格式；硬件自动生成奇偶标志；全双工操作；独立的波特率产生逻辑；独立的发送器和接收器允许控制位；通信过程可采用中断驱动机制；具有回送方式，方便了调试；可以监视发送器的输出，实现通信过程的自诊断。它的初始化很简单，使用起来非常方便。

下面是 SCI 的初始化程序：

```
void SCI_init(void)                    //SCI 模块初始化
{
SCI0BD=260;                            //  baud rate  9600
SCI0CR1=0x00;                          //正常 8 位模式，无奇偶校验
SCI0CR2=0x2C;                          //接受中断允许
}
```

SCI 的初始化完成后，S12MCU 就进入等待中断模式,等待上位机向其发送数据。

4.3.6 ECT 模块初始化

S12 的 ECT 具有 8 个输入捕捉（Input Capture，IC）/输出比较(Output Compare，OC)通道，可以通过设置 TIOS 寄存器的 IOSx 位选择输入捕捉/输出比较（IC/OC）

功能，8 个输入捕捉/输出比较通道各自具有自己的向量中断和控制寄存器。

下面是 ECT 模块的初始化程序：

```
void ECT_init(void)                          //ECT 模块初始化
{
 TIOS=0x00;                                 //通道设置为输入捕捉工作方式
 TSCR1=0x80;                                  //使能定时器
 TCTL3=0x02;                                  //通道 4 采用下降沿捕捉
 TIE=0x10;                                    //通道 4 定时器中断使能
 TSCR2=0x07;                                  //预分频系数为 128
 DLYCT=0x00;                                   //进制输入延时
 PACTL=0x40;                                  //输入脉冲累加使能
}
```

ECT 模块为我们系统测速功能提供了很大的方便，由于有了这个增强型输入捕捉定时器，测速模块的软件实现就显得比较简单。ECT 模块的初始化，正是为以后的测速做好了准备。

4.3.7 高速 AD 模块初始化

高速 AD 模块工作时，CPU 向该模块发出启动命令，然后进行采样，A/D 转换，最后将结果保存到相应的寄存器。高速 AD 模块 AD 转换芯片及相应的外围电路组成。

下面是 AD 模块的初始化程序：

```
  void ATD_get(void){
  for(z=0;z<135;z++){
     }
     for(x=0;x<X_limit;x++){
```

```
            image1[x]=PORTA;
            for(c=0;c<3;c++);
    }
       for(x=0;x<X_limit;x++){
       image[y][x]=image1[x];
    }
    y++;
       }
```

AD 模块是视频信息采集的基础，也是整个系统识别道路的基础。AD 初始化不是在主程序中完成的，而是在每次采集视频信息之前完成的。所以，AD 的初始化不同前面的初始化部分，它需要在每个控制周期都初始化一次。

4.3.8 串口初始化

S12有两个串行通信接口SCI0和SCI1，均为TTL电平输出。使用时，可以对波特率、数据格式（8 位或9 位）、发送输出极性、接收唤醒方式等进行选择。另外，发送和接收可分开使能，模块中还提供了多种避免传输错误的选项。在系统调试时，利用其中一个串口SCI0和上位PC通信，其初始化过程如下：

```
SCI0BD=130; // 波特率=40M/(16*SCI0BD)=19200
SCI0CR1=0x00; //正常工作, 8 个数据位, 1 个停止位
SCI0CR2=0x0c; //允许收发
```

4.3.9 参数设置模块初始化

系统参数设置模块由 1 个数码开关和 2 个数码管。通过数码开关选择要设置的参数类型和该参数的具体值，同时利用 LED 灯和数码管进行显示，可以方便地对各种参数进行设置。这部分的初始化过程为：

```
DDRA=0xFF; // LED#1
DDRB=0xFF; // LED#2
PORTA=0;
```

```
PORTB=0;

DDRT=0x00;

TIOS=0x00;

TSCR1_TEN=1;

TCTL3=0b01001000; // 捕捉 PT7 的上升沿和 PT5 的下降沿
```

4.4 视频采集与黑线提取

4.4.1 视频采集

通过摄像头捕获视频信号，采用查询方式或者中断方式对视频信号进行采样。结合前面的电路原理图，主程序采用 I/O 查询方式判别奇偶场信号跳变，并用中断方式处理行同步引发的中断。当奇偶场信号发生跳变（即新的一场到来）时，对行同步信号计数器清零。在行中断服务函数中，每来一个行同步则行计数器加 1，当行计数等于所需采样行时，结束采样进入控制算法，直到一个控制循环结束。行中断服务函数的流程为：

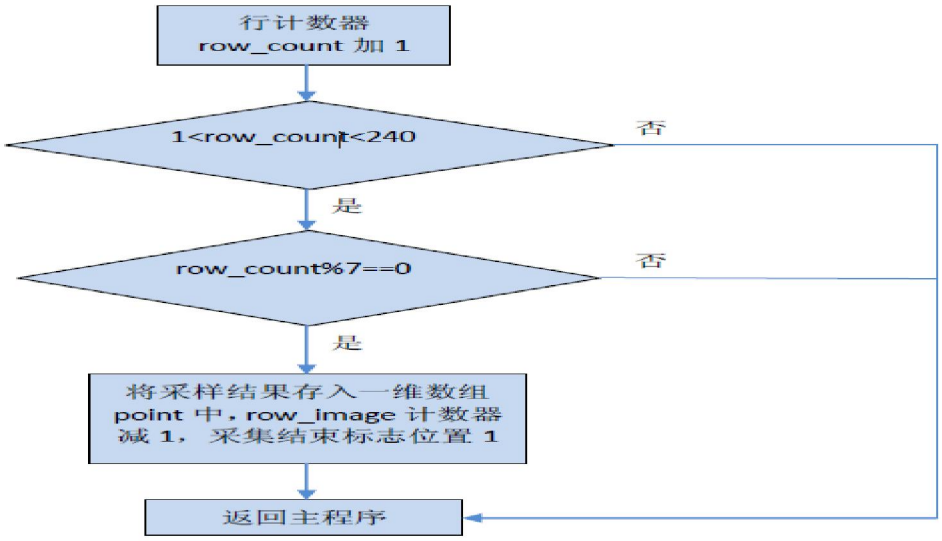


图 4.5 行中断服务函数流程

由于实际中没有必要对 240 行视频信号全部进行采集，所以选择第 1 行到 240 行间等间隔的 40 行视频信号进行采集。目前，单行视频采集的点数为 100，这一采样精度使得系统可以针对车体偏差，对舵机位置进行精确调整，以提高系统稳定性。将采集到的图像通过串口发送至 PC 机，然后用 Matlab 程序进行

显示。

4.4.2 黑线提取

摄像头采用的是隔行扫描的方式，为方便设计，我们忽略奇场和偶场在扫描位置上的细微差别，认为奇、偶场的扫描位置相同。由实测结果可知，所用摄像头每场信号的第 23 行至第 310 行为视频信号，即每场有 288 行视频信号，这已远远超出了系统所需的精度要求。

实际我们没必要对这 288 行中的每一行视频信号都进行采样，如此会增大 S12 存储和数据处理的负担，甚至会超出 S12 的处理能力。再者，这样做是没必要的，事实上，小车的定位系统在纵向上只要有 10～20 个像素的分辨能力就足够了。因此，我们只需对这 288 行视频信号中的某些行进行采样就行了。假设每场采样 40 行图像数据，我们可以均匀地对 288 行视频信号进行采样，例如采样其中的第 7 行、第 14 行、第 21 行、……、第 273 行、第 280 行，即采样该场信号的第 29 行、第 36 行、第 43 行、……、第 295 行、第 302 行（每场开始的前 22 行视频为场消隐信号）。

结合前面的电路图，采用 I/O 查询方式识别奇偶场信号跳变；采用中断处理方式处理行同步引发的外部中断。当奇偶场信号发生跳变时，对行同步信号重新计数。在行同步中断处理程序中，每个行同步信号，行计数加 1。当行计数到达所需采样行时，即初始化 AD 模块，开始对此行信号进行 AD 采样，直到下一个行同步信号到来。如此循环，直到采样完最后一行信号。

获取场跳变的方法有三种：1，脉冲捕捉场同步信号（捕获下降沿）2，捕捉奇偶场同步信号（上升/下降沿）3，查询方式识别奇偶场信号跳变。三种方式均可以稳定捕获信号，但是在程序优化上，采用捕获场同步信号的方法比较可靠，占用时间片最短。可以利用程序处理剩余的时间完成更多的算法操作。为了观察摄像头视频采样的效果，我们将 S12 采样到的每行图像数据通过串口发送到 PC 机上，然后利用 matlab 软件将图像数据组成的二维数组以灰度图的方式显示出来（如图 4.6 所示）。

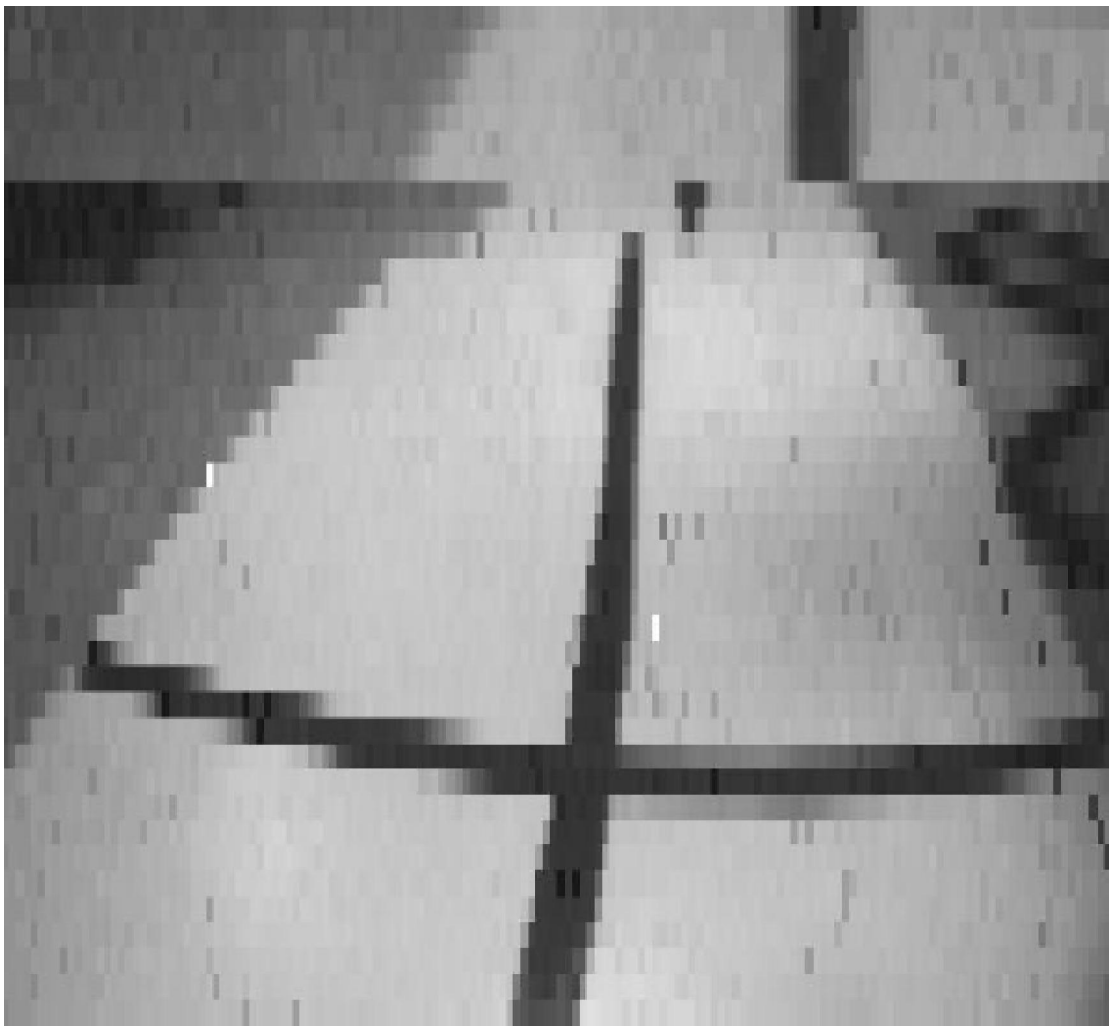


图 4.6 灰度图

有了可靠的图像数据，我们就可以对小车进行控制。应该说摄像头就是小汽车的眼睛，我们所做的工作就是确保，摄像头能看得尽量远，看得尽量清楚。为此，我们不断调整摄像头安装的方式，希望可以得到满意的结果。最后，我们确定了现在的方案，看似简单的想法，其实是在实践中不断探索的结果。

4.5 舵机控制

转向控制在系统的行进过程中非常重要，由于舵机的延迟，很可能使得系统在比较小的弯道来不及转向。因此，除了尽量让摄像头看得远之外，还应该在舵机的控制策略上下点功夫。下面是我们模拟的道路的几种基本情况，现实的情况要比这些情况复杂得多，现在就暂且做简化处理。

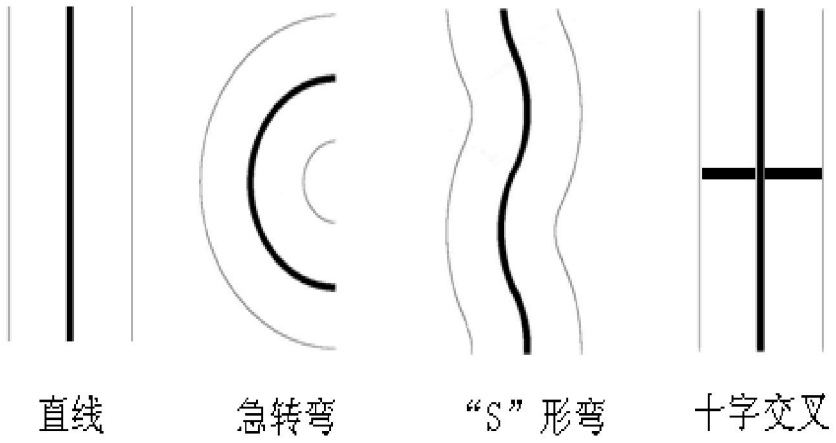


图 4.7 道路的基本形状

舵机的控制决定了小车运动的轨迹，为了尽可能提高车模运行平均速度，针对不同的路面需要对运动轨迹进行优化。对于急转弯路面，应尽量靠内道行驶，而“S”形弯应该取直穿过。理想运动轨迹如图 4.8 所示：

为了简化问题，可以将小车的运动规律进行简化：车体近似沿着道路中心线运行时，改变舵机输出转角即可改变小车前轮转向，并改变车体与道路中心线的相对位置，而位置的改变量近似等于舵机输出转角随着距离的积分，如公式 1 所示：

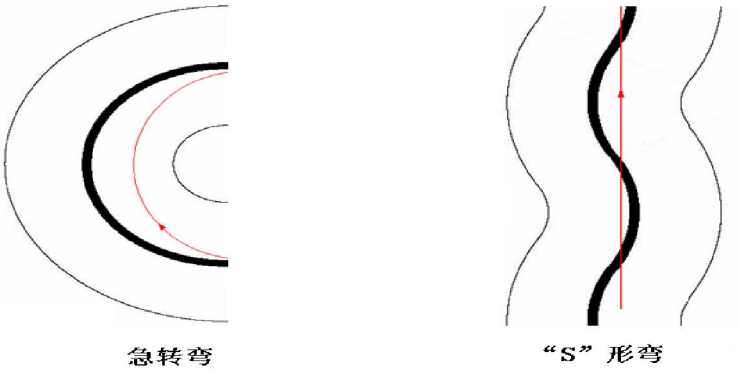


图 4.8 运动轨迹优化示意图

$$\Delta d(l) \approx k\int_0^l A(\tau)d\tau \qquad (公式1)$$

其中，$d(l)$ 为模型车与道路中心线的偏移量；$A(\tau)$ 为舵机输出转角；k 比例因子，与系统模型的尺寸、舵机之前轮转角的传动比等因素有关。

从这个简化系统运动模型可以看出，系统与道路中心线的相对位置是舵机

输出转角的积分量，舵机的控制规律可以采用比例负反馈控制，使得舵机输出转角大小正比于模型车与道路位置偏移量，且方向相反。因此本方案采用以下方法对小车的舵机进行控制：首先，在采集来的图像上建立一定的直角坐标系，通过对参考行黑点中心位置进行分析，获得前方路径的斜率 k，以及小车所在位置 x。然后根据路径斜率 k 及小车位置 x 的值，来设定小车应有的转弯角度，将道路斜率信息加入到控制率中等效于控制中增加了微分项。

舵机占空比的输出可按公式 2 给定：

$$\begin{cases} Rudder = Left\_Max + k_{left} \cdot x & (x <= 22) \\ Rudder = Right\_Max - k_{right} \cdot x & (x > 22) \end{cases} \qquad (公式\ 2)$$

式中 $Rudder$ 是指舵机输出量；$Left\_Max$、$Right\_Max$ 分别为舵机转角左右限是对应的输出量；x 指车体中心位置与视场中位置之差，与斜率有关；k_{left}、k_{right} 分别指舵机左右转时控制输出量的比例系数。实际中需要注意的是，由于机械连接等原因，舵机的输出转角与其控制信号 PWM 波的占空比并不是严格呈线性关系，且左右转角不严格对称，k_{left} 可能 k_{right} 并不相等，需要进行校正。

下面是舵机控制的 C 语言程序：

```
void steer_ctrl(void){                                    //转向控制
long int steer_temp;
float x=0.0;
x=(float)(9)/distance[follow];
steer_temp=(signed int)(steer_MIDDLE.wire_processed[follow]*(steer_P*x)
.(wire_processed[follow].last_wire_processed[follow])*(steer_D*x));
                                                          //舵机方向的 PD 控制
if(steer_temp<3200)                                       //很靠左时的控制
steer_temp=(unsigned int)((left_limit+400)                //很靠右时的控制
.((left_limit+400).steer_temp)*1.3);
```

```
else if(steer_temp>4000)
steer_temp=(unsigned int)((right_limit.400)
+(steer_temp.(right_limit.400))*1.3);
if(steer_temp<left_limit) steer_temp=left_limit;          //左极限控制
else if(steer_temp>right_limit) steer_temp=right_limit;   //右极限控制
//if(wire_confirm==1)
        PWMDTY01=(unsigned int)steer_temp;
        if(KEYSET1_1==0 && motor==0)
PWMDTY01=steer_MIDDLE;                    //舵机打到中间位置
        }
```

4.6 测速和速度设定

智能寻迹导航系统的测度功能主要是依靠定时器的 ECT 模块，通过记录脉冲的个数来得到测速的目的。下面就是测速部分的 C 语言程序：

```
void get_speed(void){                              //测  速
char s;
for(s=0;s<9;s++){
        speed_tmp[s]=speed_tmp[s+1];
}
speed_tmp[9]=PACN32;
PACN32=0;
speed=speed_tmp[9];
            if(R==5)   ts((unsigned char)speed);          //传送速度到上位机
}
```

速度的设定需要在后面的图像识别的基础上完成，速度的大小取决于道路

情况，直道上速度最快，小 S 弯道上速度其次，大弯道在次之,小弯道的上的速度最慢。通过图像识别就可以判断出道路的基本情况，然后就可以设定速度的大小。

```
void speed_set(void)            //设定速度
{
set_speed=130+min(8,max(0,(pre_lost_line+last_lost_line+lost_line)/3.16))*min(8,max(0,(pre_lost_line+last_lost_line+lost_line)/3.16));
if(line_or_S==1 && abs(wire_processed[27])<5)
set_speed=200;                  //直线时设定的速度
if(lost_line==27&&abs(X[0])<straight_line_value.5&&abs(X[1])<straight_line_value.5&&(abs(X[2])>=straight_line_value||abs(X[3])>=straight_line_value))
set_speed=140;                  //直线入弯时设定速度
//set_speed=135;                //巡航时的速度设定
if(R==4) ts((unsigned char)set_speed);   //传送速度到上位机
}
```

4.7 速度的 PID 控制

为了达到好的速度控制效果，对速度进行闭环控制是必须的。这里所说的速度控制策略是指设定速度的确定方法——设定速度主要由道路与直道的偏差来决定，道路越接近直道，设定速度越高，反之越低。

系统行进中的最低速度是这样确定的：令系统以较低的速度匀速行使，在保证安全的前提下，逐渐提高匀速行使的速度，直到系统出现不稳定行为，此速度再减去一个安全量，即为所需的最低速度。简单的说，变速行使的最低速度等于匀速行使的最高速度。

系统行进中的最高速度是这样确定的：在确定最低速度以后，加入变速策略，不断提高最高速度的设定值，直到系统出现不稳定行为，此速度再减去一个安全量，即为所需的最高速度。

系统行进过程中难免出现“失去道路”的情况，对此需要采取一定的安全策略，防止系统“盲跑”而导致危险。

速度控制的实现我们采用 PID 算法，下面就简单介绍一下：

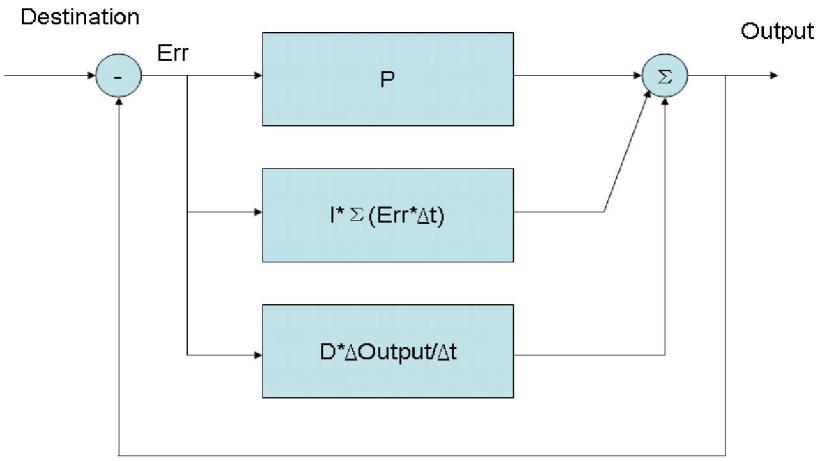


图 4.9　PID 算法框图

标准的直接计算法公式：

$$Pout(t) = Kp * e(t) + Ki * \Sigma e(t) + Kd * (e(t) - e(t-1));$$

上一次的计算值：

$$Pout(t-1) = Kp * e(t-1) + Ki * \Sigma e(t-1) + Kd * (e(t-1) - e(t-2));$$

两式相减得到增量法计算公式：

$$Pdlt = Kp * (e(t) - e(t-1) + Ki * e(t) + Kd * (e(t) - 2*e(t-1) + e(t-2));$$

三个基本参数 Kp,Ki,Kd 在实际控制中的作用：

比例调节作用（Kp）：是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节作用（Ki）是使系统消除稳态误差，提高误差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常数。积分作用的强弱取决于积分时间常数 Ti,Ti 越小，积分作用就越强。积分作用常与另外两种调节规律结合，组成 PI 调节器或 PID 调节器

微分调节作用（Pd）：微分作用反应系统的误差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，

已被微分调节作用消除。因此，可以改善系统的动态性能。

下面就是 PID 控制速度的 C 语言程序：

```
void speed_ctrl(void){                                    //速度 PID 控制
if(speed_RW==1){
pwm_add=15*(set_speed.speed)+10*(set_speed.speed.last_error);
pwm_temp=pwm_temp+pwm_add;                    //增量式速度调节
pwm_temp_min=.1000;
        if(pwm_temp<pwm_temp_min) pwm_temp=pwm_temp_min;
        else if(pwm_temp>1000) pwm_temp=1000;
        if(pwm_temp>=0){                                    //设定正转的极性
        PWMPOL_PPOL7=1;
        PWMPOL_PPOL3=1;
        PWMDTY23=pwm_temp;
        }else{                                              //设定反转的极性
        PWMPOL_PPOL7=0;
        PWMPOL_PPOL3=0;
            PWMDTY23=abs(pwm_temp);
            }
}
        if(R==8)  ts((unsigned  char)(pwm_temp+500)/100);   //传送数据到上位机
        pre_error=last_error;
        last_error=set_speed.speed;
}
```

速度的 PID 算法理论上实现起来简单，而在实际的应用过程中，它的三个系数需要不断调节才能达到很好的效果。在速度的控制上，我们采用了增量式 PID 控制的算法。在实验过程中，我们发现微分系数并不适应我们的系统，我们去掉了，实际上我们就用到了 PI 调节。关于 PI 控制的系数调节，我们采取的先后顺序是：先调节比例系数 P,再调节微分系数 I。

系统的总体软件设计是建立在图形识别有效的前提下的，我们做的这些软件不过是在为图像识别的做好铺垫，便于我们后面对图像识别的研究。看似简单的几行程序，其实是耗费了很多时间才调通并优化的。硬件是一个系统的骨架，软件就是一个系统的灵魂。在我们的这个系统中，图像识别就是我们软件的核心。没有了图像识别，我们前面所做的一切都是徒劳无益的。

4.8 本章小结

本章主要介绍了系统的初始化、车体控制相关的转向控制、速度控制和整个车辆导航系统的图像采集和处理的环节。

在硬件和机械结构相同的情况下，转向控制和速度控制对赛车的行驶表现起着至关重要的作用，转向控制根据前瞻不同P参数不同，总共有20个参数，这20个参数的整定较为麻烦，需要根据行驶表现做相应的调整，远端的P参数影响入弯的时机，但也影响直道和S弯的稳定性，近端的参数影响过弯的路线，这些参数需要经过反复调试才能得到较完美的参数。速度控制的最高速度和第二速度区顶速和底速需根据不同的赛道组合类型设定，没有明确的最优值。

车辆导航系统的图像采集和处理的环节包括了导引线的提取、干扰的消除和标志的识别，它们在介绍时虽然是分开介绍的，但体现在程序中却是紧密结合的一个整体，像标志的识别和干扰消除是环环相扣的，很多标志都是从干扰中提取出特征来识别的。这一部分是决定控制效果好坏的基础，如果提取的导引线错误的话，再好的控制也是无济于事，所以图像的采样和处理一定要力求稳定、准确。

第五章 总结和展望

本文详细介绍了为第四届全国智能车大赛而准备的智能车系统方案。该系统以Freescale 16位单片机MC9S12XS80作为系统控制处理器，采用基于的摄像头的图像采样获取赛道图像信息，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PID方式对舵机转向进行反馈控制。通过速度传感器对小车形成速度闭环控制。

文中介绍了赛车机械结构和调整方法，赛车各个主要模块的工作原理和设计思路，并叙述了系统开发过程中所用到的开发工具、软件以及各种调试、测试。

综合来看，智能赛车分硬件和软件两部分。硬件部分主要是赛车的安装和调整，以及电路板的设计。软件部分是本文的重点，除了基础工作外，主要分为两部分：

(1)图像处理

在图像处理部分，我们只是把黑线的位置提取出来了，并没有进一步判断出它的实际意义，也就是说，单片机下一步根据黑线决定方向和速度的过程和人的思考过程还是有很大差别的。这种模糊的判断可以保证系统正常工作，是由于我们的跑道比较简单，一旦面对复杂的道路信息就不行了。所以在软件方面可以尝试向智能化发展，模拟人的思维，让系统可以应付更加复杂的道路。

(2)车体控制

车体控制主要分为速度控制和舵角控制。速度控制相比去年有了很大进步，更短的控制周期使加减速更及时，也是速度更加平滑。更好的加减速性能使小车在不同赛道跑出响应的极限速度。舵机控制关系到方向的选择，好的舵机控制不仅要准确循线，还要能够沿最优路线前进。

总结整个设计过程，我们学到了很多，也克服了种种困难，不仅使我们得到了对已有知识进行实践的机会，更培养了一定的科研能力，拓宽了知识面，尤其是在硬件电路方案的确定过程中，经过不断的反复试验，最总确定了现在这套成熟稳定的方案。展望未来，智能车技术必将在更广阔的领域得到广泛应用。

参考文献

[1] 王锦标，方崇智.《过程计算机控制》.北京：清华大学出版社出版，2003.

[2] 邵贝贝编，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.

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

[4] National Semiconductor .LM1881 Video Sync Separator.National Semiconductor， Inc，2003.

[5] Freescale Semiconductor MC33886，Technical Data Freescale Semiconductor，Inc，2005.

[6] Motorola MC34063A DC-DC Converter Control Circuits National Semiconductor，Inc，2003.

[7] 俞斯乐等.电视原理（第五版）.北京：国防工业出版社，2000.

[8] 孙景琪等.视频技术与应用.北京：北京工业大学出版社，2004.

[9] 孙忠献.电机技术与应用.福建：福建科学技术出版社，2003.

[10] 王益全.电动机原理与实用技术.北京：科学出版社，2005.

[11] Motorola MC9S12DG128 Device User Guide Motorola，Inc，2001

[12] 陶永乐编著.新型PID控制及其应用（第二版）.北京：机械工业出版社，2002.

[13] 迈克•普瑞德科.机器人控制器与程序设计.北京：科学出版社，2004.

[14] 崔屹编著.图像处理与分析－数学形态学方法及应用.北京：科学出版社，2000.

[15] 阮秋琦.数字图像处理学.北京：电子工业出版社，2001.

[16] 章毓晋.图像处理和分析.北京：清华大学出版社，1999.

[17] 罗飞.运动控制系统.北京：化学工业出版社，2001.

[18] Todd D Morton. 嵌入式微控制器.严隽永译. 北京：机械工业出版社，2005.

[19] （日）船仓一郎.机器人控制电子学.北京：科学出版社，2004.

[20] 李世华，田玉平.移动小车的轨迹跟踪控制.控制与决策，2000,15(5):：626~628

附录 A：程序源代码

```
#include <hidef.h>
#include <mc9s12dg128.h>
#include <math.h>
#pragma LINK_INFO DERIVATIVE "mc9s12dg128b"
#define  VIDEO_MIDDLE          61                          //image_
parameter
#define  image_Y_limit         30   image_X_limit          108
#define  image_L_limit         106
#define  image_R_limit         16
#define  black_gate            36
#define  scratch_line_gate     35
#define  straight_line_value   12
#define  search_area           20
#define  motor           PTJ_PTJ7
#define  KEYSET1   PORTA                                   //keyboard_input
#define  KEYSET1_1    PORTA_BIT0
#define  KEYSET1_2    PORTA_BIT1
#define  KEYSET1_3    PORTA_BIT2
#define  KEYSET1_4    PORTA_BIT3
#define  KEYSET1_
5    PORTA_BIT4
#define  KEYSET1_6    PORTA_BIT5
#define  KEYSET1_7    PORTA_BIT6
```

```
#define    KEYSET1_8     PORTA_BIT7
#define    KEYSET2_1     PTM_PTM0
#define    KEYSET2_2     PTM_PTM1
#define    KEYSET2_3     PTM_PTM2
#define    KEYSET2_4     PTM_PTM3
#define    KEYSET2_5     PTM_PTM4
#define    KEYSET2_6     PTM_PTM5
#define    KEYSET2_7     PTS_PTS2
#define    KEYSET2_8     PTS_PTS3
unsigned int    steer_MIDDLE=3650;                          //stree_ctrl
unsigned int    left_limit=2950;
unsigned int    right_limit=4350;
unsigned int    follow=19;
unsigned int    steer_P=110;
unsigned int    steer_D=60;
char            speed_RW=1;
unsigned    int    image_X=0;                  //get_single_line_AD
unsigned int    image_Y=0;
unsigned int    Y_FLAG=1;
unsigned char image[image_Y_limit][image_X_limit]={0};
unsigned char line_temp[image_X_limit];

unsigned int    speed_tmp[10];                              //get_speed
unsigned int    speed=0;
```

```
unsigned int    stop=0;
int             pwm_temp=0;                          //speed_ctrl
int             last_error=0;
int             pre_error=0;
int             pwm_add=0;
int             pwm_temp_min=0;
unsigned int    flag=44;                             //image_process
int             ma=0;
int             mi=0;
int             expect;
int             min_dif=100;
int             black_num=0;
int             wire_confirm=1;
int             lost_line=27;                        //wire_process
int             last_lost_line=27;
int             pre_lost_line=27;
int             lost_limit=26;
int             lost=1;
int             last_lost=1;
int             temp[10]={3,3,3,3,3,3,3,3,3,3};
unsigned int    set_speed=170;                       //set_speed
unsigned int    set_speed_temp[4]={20,20,20,20};
unsigned int    length=0;                            //stop car
unsigned char   distance[30]={0,5,9,11,13,14,16,17,19,20,22,24,26,28,30,
```

```
33,36,40,44,49,54,60,66,73,81,90,99,109};
int    wire[30]={VIDEO_MIDDLE,VIDEO_MIDDLE,VIDEO_MIDDLE};
int                wire_processed[30];
int                last_wire_processed[30];
int                pre_wire_processed[30];
int                uniform_wire[15];
int                last_uniform_wire[15];
int                pre_uniform_wire[15];
unsigned char    R =0;                                    //SCI_receive
unsigned char    R1=0;
int                 time=0;
void boot(void);
void speed_ctrl(void);
void transmit_image(void);
void control(void);
void scratch_line_identify(void);
void speed_set(void);
void stop_car(void);
void get_speed(void);
void image_process(void);
void get_second_line(void);
void get_first_line(void);
void wire_process(void);
void steer_ctrl(void);
```

```
void get_single_line_AD(void);
void follow_line(void);
extern int abs(int x);
extern double atan2(double x,double y);
void PLL_init(void){                          //设定总线时钟 40MHz
        SYNR=0X09;
        REFDV=0X03;
        while(CRGFLG_LOCK==0);
        CLKSEL=0x80;
}
void    AD0_init(void){                       //AD 转换初始化
        ATD0CTL2=0xC0;
        ATD0CTL3=0x08;
        ATD0CTL4=0x81;
        ATD0CTL5=0xA0;
        ATD0DIEN=0x00;
}
void PWM_init(void){                          //PWM 初始化
PWMPOL_PPOL1=1;
PWMCTL=0xb0;
PWMCAE=0xff;
PWMPRCLK=0x12;                          //A clock 10MHz ,    B clock 20MHz
PWMCLK=0xFF;
PWMSCLA=0x01;                             //SA    5MHz=(10/(2*1))
```

```
PWMSCLB=0x01;                          //SB   10MHz=(20/(2*1))
PWMPER01=50000;
PWMDTY01=steer_MIDDLE;
PWMPOL_PPOL7=1;
PWMPOL_PPOL3=1;
PWMPER23=1000;
PWMDTY23=0;
PWMPER67=1000;
PWMDTY67=0;
PWME=0xFF;
}
void ECT_init(void){                          //ECT 模块初始化
  TIOS=0x00;
  TSCR1=0x80;
  TCTL3=0x02;
  TIE=0x10;
  TSCR2=0x07;
  DLYCT=0x00;
  PACTL=0x40;
}
void IO_init(void){                           //IO 口初始化
  DDRB=0xff;
  PORTB=0xff;
  DDRA=0x00;
```

```
  PUCR_PUPAE=0;
  DDRM=0x00;
  PERM=0x00;
  DDRS=0x00;
  PUCR_PUPAE=0;
  DDRJ_DDRJ7=1;
  motor=1; //enable 33886
  DDRH=0x00;
}
void SCI_init(void){                                    //SCI 模块初始化
SCI0BD=260; //    baud rate    9600
SCI0CR1=0x00;
SCI0CR2=0x2C;
}
int max(int x,int y){                                   //求最大值子函数
    if(x>=y) return x;
    else return y;
}
int min(int x,int y){                                   //求最小值子函数
    if(x<=y) return x;
    else return y;
}
void ts(unsigned char transmit_data){                   //传输图像子程序
      while(SCI0SR1_TC==0);
```

```
        SCI0DRL=transmit_data;
    }
    /********************                                    get_single_line_AD
**************************/
    void get_single_line_AD(void){                    //get_single_line_AD
    AD0_init();                                        //调用 AD 转换初始化
    for(image_X=0;image_X<=image_X_limit;image_X++) {
      while(ATD0STAT1_CCF0==0);
       line_temp[image_X]=ATD0DR0L;
      }
    ATD0CTL2=0x00;
     for(image_X=0;image_X<image_X_limit;image_X++) {
     image[image_Y][image_X]=line_temp[image_X];
     }
    image_Y++;
    }
    /************************                                  transmit_image
**************************/
    void     transmit_image(void){                          //传送图像
    TIE=0x00;
      for(image_Y=0;image_Y<image_Y_limit;image_Y++){
              for(image_X=0;image_X<image_X_limit;image_X++){
                  ts(image[image_Y][image_X]);
              }
      }
```

```
    TIE=0x10;
    R=0;
    }
    /*****************************main*******************************
*/
    void main(void) {                                   //主函数
    long int delay=0;
    DisableInterrupts;
    PLL_init();                                         //调用总线时钟初始化
    IO_init();                                          //调用 IO 口初始化
    SCI_init();                                         //调用 SCI 模块初始化
    PWM_init();                                         //调用 PWM 初始化
    ECT_init();                                         //调用 ECT 模块初始化
    for(delay=0;delay<1000000;delay++){
       asm nop;
          asm nop;
    }
    EnableInterrupts;
    for( ; ; );
    }
    /*************************
boot*************************************/
    void boot(void){                                    //选择执行的任务
      INTCR_IRQEN=0;
       switch(R){
```

```
    case 255:    transmit_image(); break;                    //传输图像命令
    default:control();                                       //进入控制周期
   }
  }
  /*************************                                  control
*********************************/
  void control(void){                                        //一个控制周期
  get_speed();
  speed_ctrl();
  image_process();
  wire_process();
  follow_line();
  steer_ctrl();
  speed_set();
  }
  /*************************                                  get_speed
*****************************/
  void get_speed(void){                                      //测    速
  char s;
  for(s=0;s<9;s++){
    speed_tmp[s]=speed_tmp[s+1];
  }
  speed_tmp[9]=PACN32;
  PACN32=0;
  speed=speed_tmp[9];
```

```
        if(R==5)    ts((unsigned char)speed);              //传送速度到上位机
    if(speed<10) stop++;
    else stop=0;
    if(stop>15) motor=0;
    if(speed>=35) motor=1;
    }
    /*************************get
wire*******************************/
    void get_first_line(void){                            //计算第一行的黑线位置
    expect=wire[1];
    min_dif=100;
    ma=max(image_R_limit,expect.search_area.10);
    mi=min(image_L_limit,expect+search_area+10);
    flag=expect;
      for(image_X=ma;image_X<=mi;image_X++){
        if(image[2][image_X]<=black_gate && abs(image_X.expect)<=min_dif){
          flag=image_X;
          min_dif=abs(image_X.expect);
        }
      }
      if(flag>=image_L_limit.3) wire[2]=image_L_limit;
      else if(flag<=image_R_limit+3) wire[2]=image_R_limit;
      else wire[2]=flag;
          if(R==6) ts((unsigned char)wire[2]);
```

```
    }

  void get_second_line(void){                    //计算第二行的黑线位置
    expect=wire[2];
    min_dif=100;
   ma=max(image_R_limit,expect.search_area);
   mi=min(image_L_limit,expect+search_area);
   flag=expect;
    for(image_X=ma;image_X<=mi;image_X++){
       if(image[3][image_X]<=black_gate && abs(image_X.expect)<=min_dif){
          flag=image_X;
          min_dif=abs(image_X.expect);
       }
  }
  if       (wire[2]==image_L_limit){
       wire[3]=image_L_limit;
   }
  else if(wire[2]==image_R_limit){
       wire[3]=image_R_limit;
   } else wire[3]=flag;
  }
  /*************************                                    image_process
**************************/
  void image_process(void){                        //图像处理程序
```

```
unsigned char a=0;
  wire[0]=wire[1];
  wire[1]=wire[2];
  get_first_line();
  get_second_line();
  for(image_Y=4;image_Y<=27;image_Y++){
  min_dif=100;
    expect=2*wire[image_Y.1].wire[image_Y.2];
    if          (expect>image_L_limit) expect=image_L_limit;
    else if(expect<image_R_limit) expect=image_R_limit;
    ma=max(image_R_limit,expect.search_area);
    mi=min(image_L_limit,expect+search_area);
    flag=expect;
    for(image_X=ma;image_X<=mi;image_X++){
       if(image[image_Y][image_X]<=black_gate){
          if(abs(image_X.expect)<min_dif){
          flag=image_X;
          min_dif=abs(image_X.expect);
          }
       }
    }
  if         (wire[image_Y.1]==image_L_limit && image_Y>=lost_limit){
     wire[image_Y]=image_L_limit;
  }
```

```
  else if(wire[image_Y.1]==image_R_limit && image_Y>=lost_limit){
     wire[image_Y]=image_R_limit;
  } else wire[image_Y]=flag;
   }
  black_num=0;
  wire_confirm=1;
  for(image_Y=20;image_Y<=27;image_Y++) {
 for(image_X=max(image_R_limit,wire[image_Y].15);
 image_X<=min(image_L_limit,wire[image_Y]+15);image_X++){
       if(image[image_Y][image_X]<black_gate)      black_num++;
   }
  }
  if(black_num>=35) wire_confirm=0;
  if(wire_confirm==0)      PORTB=0xff;                    //点亮 PORTB 口
  else PORTB=0x00;
       if(R==2){                                          //传送黑线信息到上位
机
       TIE=0;
          for(a=2;a<=27;a++){
          ts((unsigned char)wire[a]);
          }
       TIE=0x10;
       R=0;
       }
```

```
    }
    /*****************************wire_process*********************
****/
    void wire_process(void){                              //黑线处理程序
            unsigned char a;
            pre_lost_line=last_lost_line;
            last_lost_line=lost_line;
            for(a=2;a<=27;a++){
              if((wire[a]==image_L_limit||wire[a]==image_R_limit)
&&(wire[a.1]!=image_L_limit && wire[a.1]!=image_R_limit)){
                lost_line=a;
                wire[28]++;
                break;
              }
              }
              if(wire[28]==0){
                 if(wire[27]!=image_L_limit&&wire[27]!=image_R_limit)
lost_line=27;
                 else lost_line=17;
              }
              wire[28]=0;
              lost_limit=max(13,lost_line.3);
              lost_limit=16;
    for(a=0;a<=27;a++){
       pre_wire_processed[a] =last_wire_processed[a];
```

```
    last_wire_processed[a]=wire_processed[a];
  }
  for(a=2;a<=27;a++){
    wire_processed[a]=(int)((wire[a].VIDEO_MIDDLE)*(long
int)930/(2424.67*a));
  }
      if(R==1){                                    //传送数据到上位机
       TIE=0;
       for(a=2;a<=27;a++){
       ts((unsigned char)(wire_processed[a]+128));
       }
       TIE=0x10;
       R=0;
       }
  }
  /************************car_ctrl*********************************/
  void speed_ctrl(void){                           //速度 PID 控制
  if(speed_RW==1){
  pwm_add=15*(set_speed.speed)+10*(set_speed.speed.last_error);
  pwm_temp=pwm_temp+pwm_add;
  pwm_temp_min=.1000;
    if(pwm_temp<pwm_temp_min) pwm_temp=pwm_temp_min;
    else if(pwm_temp>1000) pwm_temp=1000;
    if(pwm_temp>=0){
```

```
    PWMPOL_PPOL7=1;
    PWMPOL_PPOL3=1;
    PWMDTY23=pwm_temp;
    }else{
    PWMPOL_PPOL7=0;
    PWMPOL_PPOL3=0;
     PWMDTY23=abs(pwm_temp);
    }
}
    if(R==8) ts((unsigned char)(pwm_temp+500)/100);       //传送数据到上位机
    pre_error=last_error;
    last_error=set_speed.speed;
}
int width=20;
int path[27];
int path23[27];
int path27[27];
int line_value_23=0;
int line_value_27=0;
int line_value_23_27=0;
int X[4];
int line_or_S=0;
int at_left=0;
int at_right=0;
```

```
void follow_line(void){                                    //跟随线控制
  int a=0;
  int b=0;
  int c=0;
  for(b=2;b<=27;b++){
     path[b]=wire_processed[b]*(b.2)/25;
path27[b]=wire_processed[2]+(wire_processed[27].wire_processed[2])*(b.2)/25
;
                }
for(b=2;b<=23;b++) {
path23[b]=wire_processed[2]+(wire_processed[23].wire_processed[2])*(b.2)/21
;
                }
                line_value_23=abs(wire_processed[ 9].path23[ 9])
                              +abs(wire_processed[14].path23[14])
                              +abs(wire_processed[17].path23[17])
                              +abs(wire_processed[19].path23[19])
                              +abs(wire_processed[21].path23[21]);
                line_value_27=abs(wire_processed[ 9].path27[ 9])
                              +abs(wire_processed[14].path27[14])
                              +abs(wire_processed[17].path27[17])
                              +abs(wire_processed[19].path27[19])
                              +abs(wire_processed[21].path27[21])
                              +abs(wire_processed[23].path27[23])
                              +abs(wire_processed[24].path27[24])
```

```
                                +abs(wire_processed[25].path27[25])
                                +abs(wire_processed[26].path27[26])
                                +abs(wire_processed[27].path27[27]);
    X[0]=wire_processed[17].wire_processed[ 2];
    X[1]=wire_processed[21].wire_processed[14];
    X[2]=wire_processed[24].wire_processed[19];
    X[3]=wire_processed[27].wire_processed[23];
    if(lost_line<17) X[0]=(X[0]/abs(X[0]))*15;
    if(lost_line<21) X[1]=(X[1]/abs(X[1]))*15;
    if(lost_line<24) X[2]=(X[2]/abs(X[2]))*15;
    if(lost_line<27) X[3]=(X[3]/abs(X[3]))*15;
    switch(lost_line){
    case 27:{                           //{0,5,9,11,13,14,16,17,19,20,22,24,26, 28, 30,
    if(abs(wire_processed[27])<30){         //33,36,40,44,49,54,60,66,73,81,90,99,10
9}
                c=0;
                at_left=0;
                at_right=0;
                if(path27[9]<wire_processed[9])          at_left++;
                else if(path27[9]>wire_processed[9])    at_right++;
                if(path27[14]<wire_processed[14])       at_left++;
                else if(path27[14]>wire_processed[14]) at_right++;
                if(path27[17]<wire_processed[17])       at_left++;
               else if(path27[17]>wire_processed[17]) at_right++;
```

```
                if(path27[19]<wire_processed[19])       at_left++;
                else if(path27[19]>wire_processed[19]) at_right++;
                if(path27[21]<wire_processed[21])       at_left++;
                else if(path27[21]>wire_processed[21]) at_right++;
                if(path27[23]<wire_processed[23])       at_left++;
                else if(path27[23]>wire_processed[23]) at_right++;
                if(path27[25]<wire_processed[25])       at_left++;
                else if(path27[25]>wire_processed[25]) at_right++;
                if(path27[26]<wire_processed[26])       at_left++;
                else if(path27[26]>wire_processed[26]) at_right++;
                if(path27[27]<wire_processed[27])       at_left++;
                else if(path27[27]>wire_processed[27]) at_right++;
                for(b=2;b<=27;b++){
                  if(path[b]+width<wire_processed[b]                                ||
path[b].width>wire_processed[b])    c++;
                }
               if(c==0 && at_left<=8 && at_right<=8) {
               line_or_S=1;
               follow=23;
                  steer_P=max(steer_P.90,45);
                  steer_D=90;
                }else line_or_S=0;
            }else line_or_S=0;
        }break;
```

```
        default: {
            if(lost_line>=14){
            follow=max(19,follow.1);
            steer_P=min(steer_P+20,90);
            steer_D=170;
            }else{
                follow=18;
                steer_P=min(steer_P+20,100);
                steer_D=170;
                    }
                }
        }
    //    follow=15;                                                    //巡航时用
    //    steer_P=90;
    }
    void steer_ctrl(void){                                             //转向控制
    long int steer_temp;
    float x=0.0;
    x=(float)(9)/distance[follow];
        steer_temp=(signed
int)(steer_MIDDLE.wire_processed[follow]*(steer_P*x).(wire_processed[follow].las
t_wire_processed[follow])*(steer_D*x));
    if(steer_temp<3200)
    steer_temp=(unsigned int)((left_limit+400).((left_limit+400).steer_temp)*1.3);
    else if(steer_temp>4000)
```

```
steer_temp=(unsigned int)((right_limit.400)+(steer_temp.(right_limit.400))*1.3);
        if(steer_temp<left_limit) steer_temp=left_limit;
else if(steer_temp>right_limit) steer_temp=right_limit;
//if(wire_confirm==1)
        PWMDTY01=(unsigned int)steer_temp;
        if(KEYSET1_1==0 && motor==0) PWMDTY01=steer_MIDDLE;
  }
//0,5,9,11,13,14,16,17,19,20,22,24,26,28,30,33,36,40,44,49,54,60,66,73,81,90,99,109

void speed_set(void){                                          //设定速度
//set_speed=min(180,max(50,180.abs(abs(wire_processed[2]+X[3]+X[2]+X[1]+X[0])/4+.abs(wire_processed[2]+X[1]+X[0])/4.abs(wire_processed[2]+X[2]+X[1]+X[0])/2)));
//ts(abs(abs(wire_processed[2]+X[1]+X[0].(wire_processed[1]+X[10]))));
set_speed=130+min(8,max(0,(pre_lost_line+last_lost_line+lost_line)/3.16))*min(8,max(0,(pre_lost_line+last_lost_line+lost_line)/3.16));
   if(line_or_S==1 && abs(wire_processed[27])<5) set_speed=200;
          if (lost_line==27 && abs(X[0])<straight_line_value.5 && abs(X[1])<straight_line_value.5 && (abs(X[2])>=straight_line_value || abs(X[3])>=straight_line_value))
           set_speed=140;
//set_speed=135;
   if(R==4) ts((unsigned char)set_speed);                  //传送速度到上位机
}
/**********************interrupts*********************************
```

```
****/
    #pragma CODE_SEG NON_BANKED
    void interrupt IRQ_ISR(void){                          //行中断服务程序
      INTCR_IRQE=1;
      Y_FLAG++;
      if(Y_FLAG%7==0 && Y_FLAG<=200) get_single_line_AD();
      else if(Y_FLAG>202) boot();
    }
    int full_length=0;

    void interrupt IC4_ISR(void){                          //场中断服务程序
    DisableInterrupts;
    TFLG1_C4F=1;
    Y_FLAG=0;
    image_Y=0;
    INTCR_IRQE=1;
    INTCR_IRQEN=1;
    full_length=full_length+speed;
    if(if_identify!=0) if_identify..;
    if(scratch_line_cnt>0) time++;
    if(scratch_line_cnt>=2){
      stop_car();

    }
```

```
EnableInterrupts;
}

void interrupt SCI_RECEIVE(void){                    //串口中断服务程序
SCI0SR1_RDRF=1;
R=SCI0DRL;
}
```

附录 B：基于摇头摄像头的图像处理算法研究

刘逸然，张程，李昊燃

（南京师范大学电气与自动化工程学院 南京，210042）

摘　要：本文介绍了智能车竞赛中摄像头组路径识别中的动态黑线处理的一种软件算法，重点介绍了此中算法的原理过程并提供了相应的源程序代码。

关键字：智能汽车，动态黑线处理，图形信息

Abstract ：This paper introduces the intelligent camera car race group to identify the path to deal with the moving black line of a software algorithm, there is a world focused on the principles of the process of algorithm and the corresponding source code.

Keyword：Smart car, moving black line processing, graphics information

智能车摄像头组竞赛中，要想跑出好成绩，就必须要有一套好的图像处理软件算法，图像处理中黑线提取和黑线的处理都是必不可少的。

由于上一届竞赛中的经验，我们发现存在这样一个问题：智能车在过弯道时由于视野不太宽阔，不能及时识别弯道，常常冲出赛道。为了解决这个问题，我们设计了摄像头摇头机构，由一个舵机带动固定摄像头的轻质材料玻璃纤维杆旋转，从而使摄像头能够在转弯时在控制算法的作用下，转动一定的角度，提前看到弯道，从而做相应的控制使智能车能够安全的通过弯道。本文重点对黑线的处理提出的一种软件算法。

黑线提取结束后，我们得到的黑线信息还不是我们需要最后结果，而是一个梯形的视野，而且其中还可能掺杂着干扰信息，如果直接用这个黑线信息做为系统的路线指引，很可能导致不明原因的误判，可控性 和可预见性不高。所以我们需要对得到的黑线信息进行处理，得到和现实情况下相近的黑线信息，并排除我们可以想到的所有干扰形式。这个黑线处理的部分主要还有两个方面，一个是梯形还原成矩形的过程，一个是超出视野的处理过程。其中梯形还原，我们采用的方式是按比例放大，它的基本算法很简单，见下图 1：

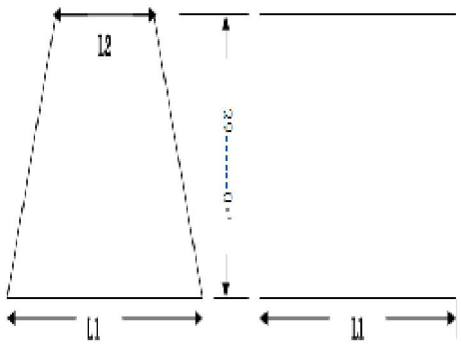


图 4.4 梯形还原示意图

如图所示，在梯形视野中，从 0 行到 29 行，每行是均匀变化的，我们可以得到每行的变化率为：

$$b=(L1-L2)/29$$

所以，我们就可以算出第 a 行的长度为：

$$La=L1-ab$$

这样就可以得到梯形还原后的矩形图形的信息，即：

W(矩形)=W（梯形）*L1/(L1-ab)

关于超出视野的处理，判断的依据是:第n行的黑线位置是不是超过左极限或是右极限，如果没超过就是有效的黑线信息，如果超过，再判断第n-1 行是不是也超过视野范围,如果没超过，则第 n 行的判断有效，如果超过了，则第 n 行的判断无效，这样做的目的是增强抗干扰能力，避免了误判断。

经过了梯形还原和超出视野处理后，得到的黑线的信息的可靠性大大加强，我们可以放心地让它作为引导系统行驶的参考依据，在实际的试验中，我们的图像处理部分经受住了严峻的考验，保证了系统的正常工作。

下面是黑线处理的软件方框图 2：

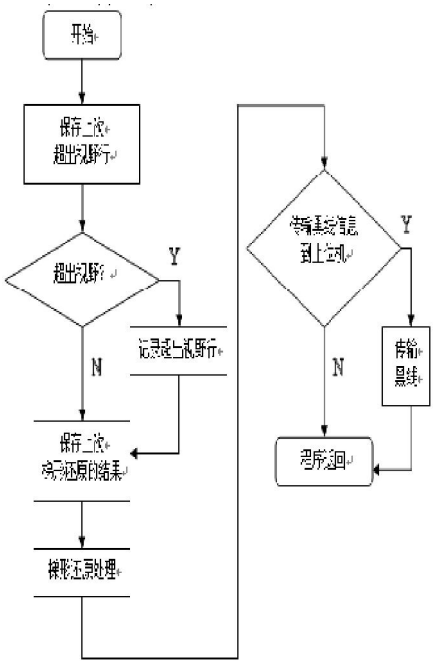


图 2 黑线处理软件方框图

通过黑线处理的软件方框图，我们可以清楚地了解处理过程，下面是黑线处理的 C 语言程序实现：

```
void
wire_process(void){
//黑线处理程序
    unsigned char a;
    pre_lost_line=last_lost_line;
    last_lost_line=lost_line;
    for(a=2;a<=27;a++){

if((wire[a]==image_L_limit||wire[a]==image_R_limit)
&&(wire[a.1]!=image_L_limit    &&
wire[a.1]!=image_R_limit)){
        lost_line=a;
        wire[28]++;
        break;
      }
      }
      if(wire[28]==0){

if(wire[27]!=image_L_limit&&wire[27]
!=image_R_limit) lost_line=27;
          else lost_line=17;
      }
      wire[28]=0;

lost_limit=max(13,lost_line.3);
      lost_limit=16;
    for(a=0;a<=27;a++){
```

```
        pre_wire_processed[a]
=last_wire_processed[a];


last_wire_processed[a]=wire_processed[
a];
    }
    for(a=2;a<=27;a++){


wire_processed[a]=(int)((wire[a].VIDE
O_MIDDLE)*(long
int)930/(2424.67*a));
    }


if(R==1){
  //传送数据到上位机
            TIE=0;
            for(a=2;a<=27;a++){
            ts((unsigned
char)(wire_processed[a]+128));
            }
            TIE=0x10;
            R=0;
            }
    }
```


参考文献：

［1］王锦标，方崇智.《过程计算机控制》.北京：清华大学出版社出版，2003.

［2］卓晴，黄开胜，邵贝贝.学做智能车----挑战“飞思卡尔”杯.北京:.北京航空航天大学出版社.2006

［3］谭浩强．C 语言程序设计(第二版)[M]．北京:清华大学出版社,2003