第十一届“恩智浦”杯全国大学生智能汽车竞赛

技 术 报 告

学　　校：惠州学院

队伍名称：神迹队

参赛队员：黄浚濠

方嘉旭

谢向云

带队教师：迟正刚

黄近秋

技术报告和研究论文使用授权的说明

本人完全了解第十一届“恩智浦”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：黄浚豪，谢向云，方嘉旭

带队教师签名：迟正刚

日　　期：2018年8月15日

I

目录

第一章 引言

1.1 背景与意义

全国大学生“恩智浦”杯智能汽车竞赛由教育部高等教育司委托教育部高等学校自动化类专业教学指导委员会于2005年创办，已经连续举办十届，累计参与学生总规模超过20万人次。智能汽车竞赛旨在加强大学生实践、创新能力和团队精神的培养，竞赛内容以智能汽车在特定赛道上能自主行驶并完成复杂工程任务为主，涵盖了控制、模式识别、传感技术、电子、电气、计算机、机械等多个学科的知识。

第十一届智能汽车竞赛在沿用原来的组别基础上，增加了新的比赛组别，除了四个基础组外还设有两个提升组（双车追逐组、信标越野组），丰富了比赛内容。同时，创意比赛部分包括了节能指标组和物联网应用展示组，与当前社会发展的热点应用更加紧密结合。

2005年，在教育部的支持下，由吴澄院士牵头，创立了全国大学生智能车竞赛。十一年来，这项竞赛经历了教指委换届，也经历了从‘飞思卡尔杯’到‘恩智浦杯’的转换，但是始终不改‘立足培养，重在参与，鼓励探索，追求卓越’的初心，始终保持趣味性、创新性、综合性、挑战性和公正性的特色，始终致力于提升大学生基本素养、创新意识、工程实践能力、严谨学风和团队协作精神。竞赛得益于恩智浦公司的长期支持，赢得了全国高校自动化相关专业师生广泛支持和热情参与，也从中成长出许多具有工程创新、创业能力的优秀人才。

“恩智浦”杯智能汽车竞赛创办之初由飞思卡尔公司协办，2015年恩智浦与飞思卡尔合并后，新恩智浦继续协办这项赛事并给予了高度关注和支持。恩智浦在华发展已经超过三十年，一直积极参与和支持中国相关产业的人才培养与科研创新，目前已经与中国150所高校建立了200余个联合实验室，服务于高校电子类课程和学生实践创新活动。2015年底，恩智浦与教育部续签了“高等学校人才培养战略合作协议”。2016年7月，恩智浦与工业和信息化部人才交流

中心签订战略合作协议，围绕“中国制造2025”，在智能制造和集成电路等领域进行技术交流和人才培养合作。

1.2 整车设计思路

智能汽车比赛关键就是能以较快的速度在符合组委会规定下完成比赛，由此我们把小车重要模块分为三类：环境感知系统、自主决策系统、操作执行系统。这三个组成部分，相互联系、相互制约，共同完成控制任务。

环境感知系统，我们的该部分主要包括感知路面信息的传感器和感知车体状态的传感器。我们选用CMOS 摄像头：获取赛道信息。

光电码盘：得到车体当前速度。

自主决策系统，通过对单片机的编程来实现决策控制。

操作执行系统，也就是硬件系统，主要就是相应的驱动电路。

三系统相互合作，帮助小车跑的更好更快。

比赛要求在组委会提供统一智能车竞赛车模下、我们选用飞思卡尔公司的32 位微处理器 K60 为控制核心，应用 BDM 在线调试，采用 IAR 软件、无线串口、红外遥控等作为调试工具进行调试制作出一个能够自主识别路线的智能车，它将在专门设计的跑道上自动识别道路行驶。比赛要求在不违反大赛规则的情况下以最短时间完成单圈赛道。

第二章 智能车机械结构调整与优化

2.1 智能车参数要求

摄像头组使用 B 型车模。摄像头车模如图 2.1.1 所示

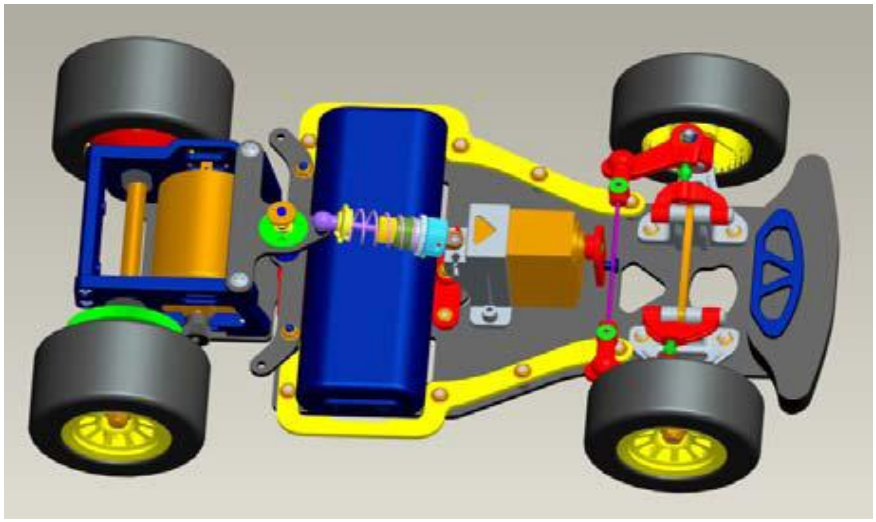

图 2.1.1 车模示意图

1. 车模尺寸要求：

摄像头组车模改装完毕后，车模尺寸不能超过：250mm 宽和 400mm 长。

2.伺服电机型号：S-D5 数字伺服，伺服电机数量不超过 3 个。

3.电机型号：RS-540

4.全部电容容量和不得超过 2000 微法；电容最高充电电压不得超过 25 伏。

2.2 智能车整体参数调校

智能车的整体参数，包括车体重心、舵机电机放置位置、高度，传感器排布方式等，都对整个智能车系统的稳定运行起着至关重要的作用。因此，对智能车机械系统的调节，有助于小车更快更稳定的运行。

小车的布局以精简、可靠、稳定为前提,通过对小车的布局，尽量保证小车左右平衡，以及寻找一个合适的重心，保证小车既能够可靠地抓牢地面，又能够对前轮舵机，后轮电机有较快的响应。

2.3 舵机安装

舵机安装直接关系到是否能快速灵敏地转向的问题。如果舵机调整不到位，将很大程度上限制转向角度和转向响应速度。

舵机安装有两种方式，一种是卧式安装，另外一种为立式安装。

卧式安装为车模默认安装方式，但这样安装会使左右两边轮子连杆不等长，根据杠杆原理可知舵机对长连杆轮子用的力要大些，因此造成了舵机对左右两边转向响应时间不一样。另外由于卧式安装会使连杆与水平面呈现一定角度，从力学知识可以知道在轮子转向获得的力只是舵机施加在连杆上力的一个水平方向上的分力。

立式安装把舵机架高，增长了力臂，使得小车反应更加灵活，但增大了阻力，力的作用减小。因此，根据舵机性能和实际情况确定高度，将舵机立式正放，不仅提高了其响应速度，还增加了小车底盘空间，易于安放电路板，降低小车重心。根据舵机形状制作了一个小巧坚固的舵机支架，支架边缘尽量少，以减少整车的重量，避免影响赛车提速。然后将支架以合适的高度固定在底盘上。如图 2.3.1 所示。

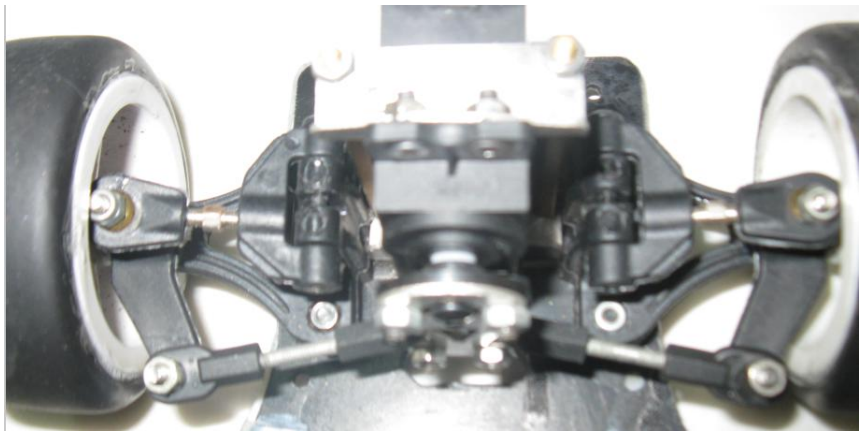

图2.3.1　舵机安装示意图

2.4 编码器安装

选用 5V 工作电压的光电编码器进行速度的测量。速度传感器用螺钉通过支架固定在后轮支架上，这样固定好之后，就有了较高的稳定性。然后调节编码器齿轮，使其与差速齿轮紧密咬合，增大测速的精确性，但是咬合过紧也增大了摩擦，减小了对电机做功的利用率，影响小车的快速行驶，因此减小摩擦同时增强齿轮间的咬合是我们主要考虑的因素。编码器安装示意图如图 2.4.1 所示：

图 2.4.1 编码器安装示意图

2.5 智能车机械参数调节

为保证智能小车直线行驶稳定，转向轻便灵活并尽可能的减少轮胎磨损，需要对小车的四轮定位参数进行调整。四轮定位内容主要有：主销后倾角，主销内倾角，前轮外倾角，前轮前束，外侧车轮二十度时，内外转向轮转角差，后轮外倾角，后轮前束。其中，前轮定位的参数对小车性能有着至关重要的影响，这四个参数反映了前轮、主销和前轴三者之间在车架上的位置关系。本文将对这四个参数做详细阐述。

2.5.1 主销后倾角

主销：转向轮围绕主销进行旋转，前轴的轴荷通过主销传给转向车轮,具备这两点的就叫做主销。

主销后倾角：主销的轴线相对于车轮的中心线向后倾斜的角度。
前轮重心在主销的轴线上由于主销向后倾斜使前轮的重心不在车轮与地面的接触点上，于是产生了离心力，主销后倾而形成的离心力，可以保证汽车直线行驶的稳定性还可以帮助车轮自动回正。主销后倾角延长线离地面实际接触越远，车速越高，离心力就越大。在高速行驶中保持汽车直线行驶的稳定性，适当的加大主销后倾角可以帮助转向轮自动回正，可有效扼制转向器的摆振，可使转向便轻，单独适量调一侧的主销后倾角可修理行驶跑偏。主销后倾角靠离心力保证汽车直线行驶和车轮自动回正。高速行驶时跑偏可通过主销后倾角调节。

但主销后倾角过大会造成高速时转向发飘。调整主销后倾角为 1°～3°，可通过增减黄色垫片数量来改变。如图 2.6.1 所示：

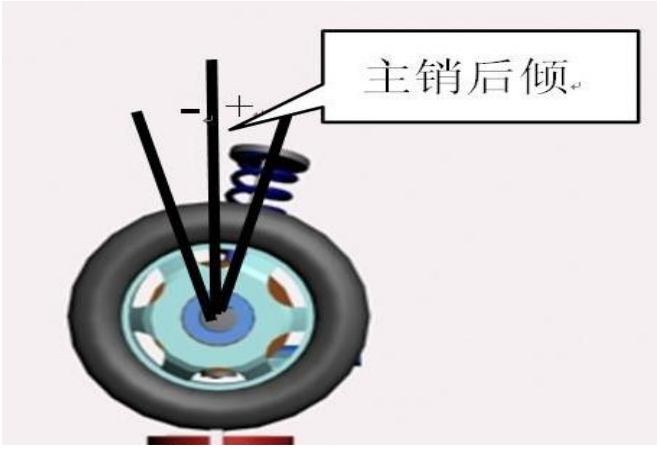

图2.6.1.1主销后倾角

2.5.2 主销内倾角

主销内倾角：在小车前后方向上，主销向内倾斜一个角度，主销轴线与垂线间的夹角称为主销内倾角。

由于主销轴线向内倾斜，所以使前轴荷更接近前轮中心线（前轴重心越接近前轮中心线转向越轻）麦弗逊式悬架分为零主销偏移和负主销偏移两种。当转向轮在外力作用下发生偏转时，由于主销内倾的原因，车轮连同整个汽车的前部将被抬起一定高度；当外力消失后，车轮就会在重力作用下恢复到原来的中间位置。故主销内倾角可保证汽车直线行驶的稳定性，还可帮助车轮自动回正，主销内倾轴线延长线在没超过前轮中心线的前提下，离前轮中心线越近，转向角越大，转向轮抬起的越高，车轮的回正力矩就越大。从而使转向操纵轻便，同时也减小了由于路面不平而从转向轮输出的力反馈。主销内倾角靠前轴轴荷保证汽车直线行驶和车轮自动回正。但主销内倾角不宜过大，否则在转向时车轮主销偏转的过程中，轮胎与路面将产生较大的滑动，从而增加轮胎与路面间的摩擦阻力，不仅会使转向变得沉重，还将加速轮胎的磨损。通常调节主销内倾角不大于 8 度。如图 2.6.2.2 所示：

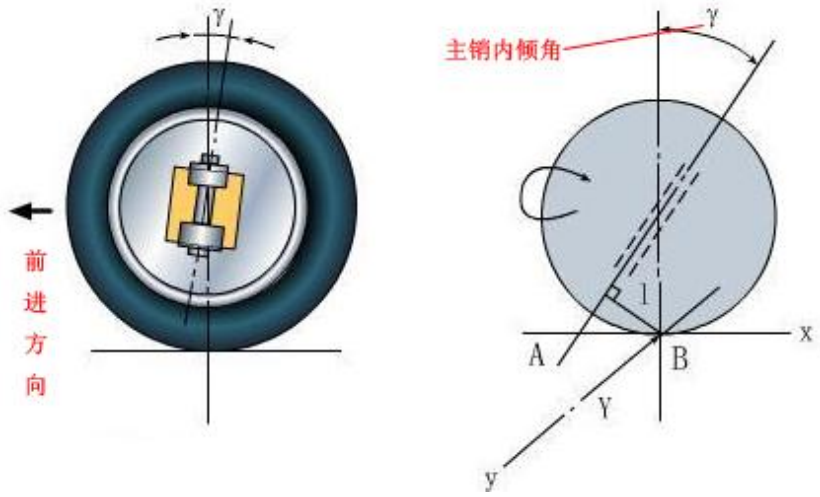

图2.6.2.2 主销内倾角

2.5.3 前轮外倾角

前轮外倾角：在汽车的横向平面内，前轮中心平面向外倾斜一个角度，称为前轮外倾角。

前轮外倾角一方面可以使车轮接近垂直路面滚动而滑动减小转向阻力，使小车转向轻便；另一方面减少了轴承及其锁紧螺母的载荷，增加了使用寿命，提高了安全性。一般前轮外倾角为 1° 左右，但对于有高速、急转向要求的车辆，前轮外倾角可减小甚至为负值。如图 2.6.3.1 所示:

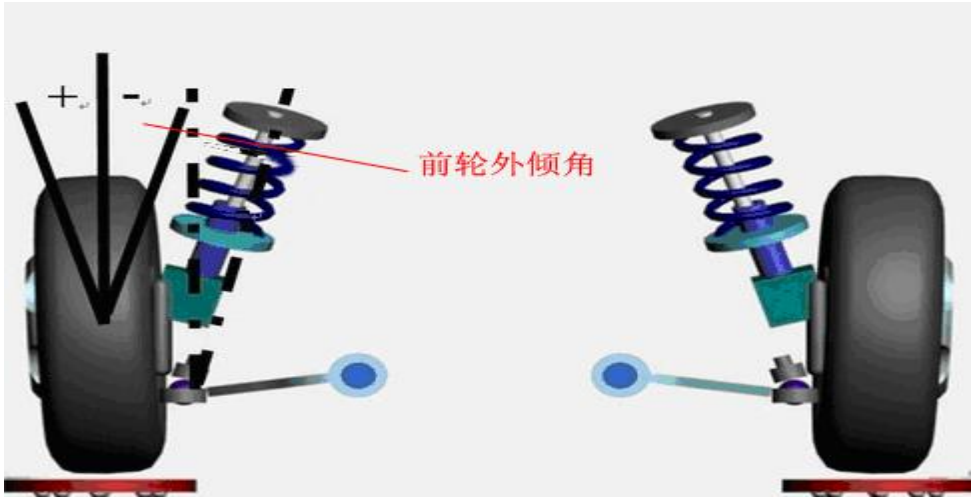

图2.6.3.1 前轮外倾角

2.5.4 前轮前束

前轮前束：俯视车轮，汽车的两个前轮的旋转平面并不完全平行，而是稍微带一些角度，这种现象称为前轮前束。车轮前束的作用是减轻或消除因前轮外倾角所造成的不良后果，二者相互协调，使前轮在汽车行驶中滚动而无滑动。

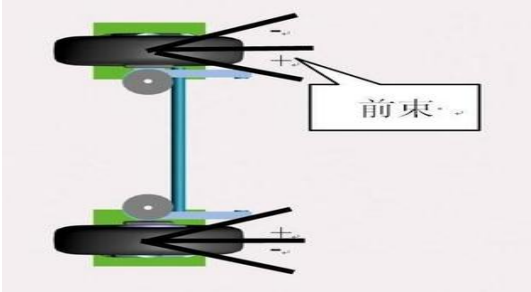

图2.6.4.1 前轮前束

2.6 其他部分调整

其他部分调整主要涉及到小车底盘高度、小车重心位置、后轮距、减震器、齿轮咬合、差速器等。具体调整如下：

1、底盘高度调整：底盘高度可以影响重心。适当降低底盘高度可以使小车重心降低，有利于过弯稳定。实际调整可以通过调整前轮高度、后轮轴高度调节块等方式来调节。

2、重心位置：重心位置同样影响小车性能。重心过前，增加转向阻力，引起转向迟滞。另外，如果小车速度很快的情况下，上下坡道的时候会造成前轮首先着地，很可能造成小车意外事故。重心过后，则会使小车前轮抓地不足，造成过弯非常不稳定。我们将电池放在靠近舵机的位置，保持各部分重量均衡。

3、后轮距调整：后轮距可以通过换装后轮宽度调整块来调整。合适的后轮宽度会是小车直线性能和弯道性能更优，利于小车稳定。

4、减震器弹簧强度调整：坡道时候，减震器影响显得尤为突出。小车平面行驶时，无垂直方向速度，遇到坡道时候，小车会瞬间在垂直方向上出现一个速度，此时减震器在耦合小车前部和小车后部时候会出现一个缓冲区，增加坡道稳定性。下坡时候则是垂直方向上的速度瞬间减小为 0，减震器的作用亦如上坡。实际调整中，我们通过加装弹簧调节块来调整。

5、齿轮咬合调整：调整齿轮咬合，以不松动，无卡滞，松紧合适为准。另外还要保证齿轮间咬合有足够的接触面积。

6、差速器调整：小车采用的差速器为滚珠式差速器。合适的差速器调整能够提高小车入弯速度，提高弯道性能。差速器调整可以通过右后轮轮轴上面的螺丝。注意调整过松，会严重影响直道加速性能；调整过紧则会使差速器处于无效状态。差速器滚珠处可以适当添加润滑剂，保证差速器平滑。

7、摄像头的固定：

摄像头安于电池的前方，有利于重心分布和盲区与前瞻的匹配。

8、电路板的安装电路板是模型车上的核心部分，在安装时，考虑到电路板的稳定性和模型车的重心等问题，我们把电路板做的比较小，安装在电池和摄像头支架中间的底板上。我们在底板上打 2 个螺孔，分别将电路板的 2 个固定孔固定在底板上，保证电路板的牢固性，同时还可以压低重心并使重心前移，提高模型车的过弯性能和稳定性。

第三章 硬件设计

3.1 电路总设计

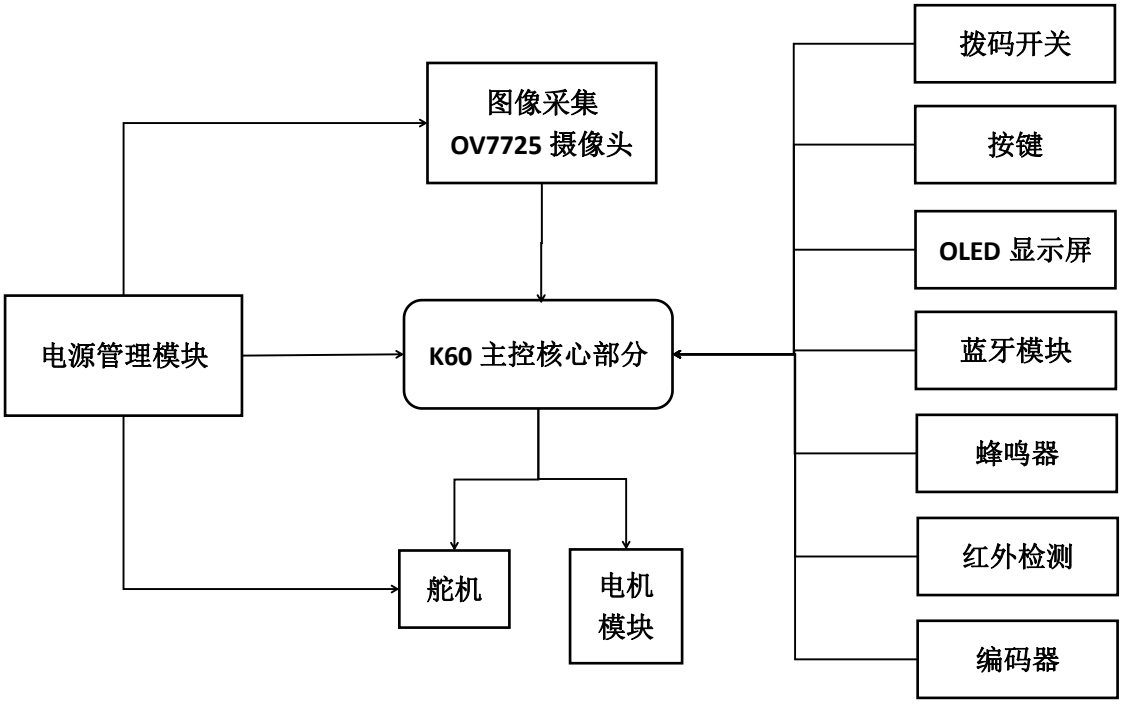

图 3.1　总体设计框图

以飞思卡尔公司的32位芯片K60作为主控模块的核心处理器；采用数据二值化摄像头OV7725作为传感器件；对主控核心、舵机、摄像头等进行单独稳压供电；扩展拨码开关、编码器、OLED显示屏等作为小车的辅助调试外围模块。总体设计框图如上图所示。

3.2 单片机系统设计

单片机最小系统是智能车系统的核心部件。我们采用了32位的K60芯片。选用山外K60最小系统核心板，将所需用到的引脚引出使用原理图如图3.2所示：

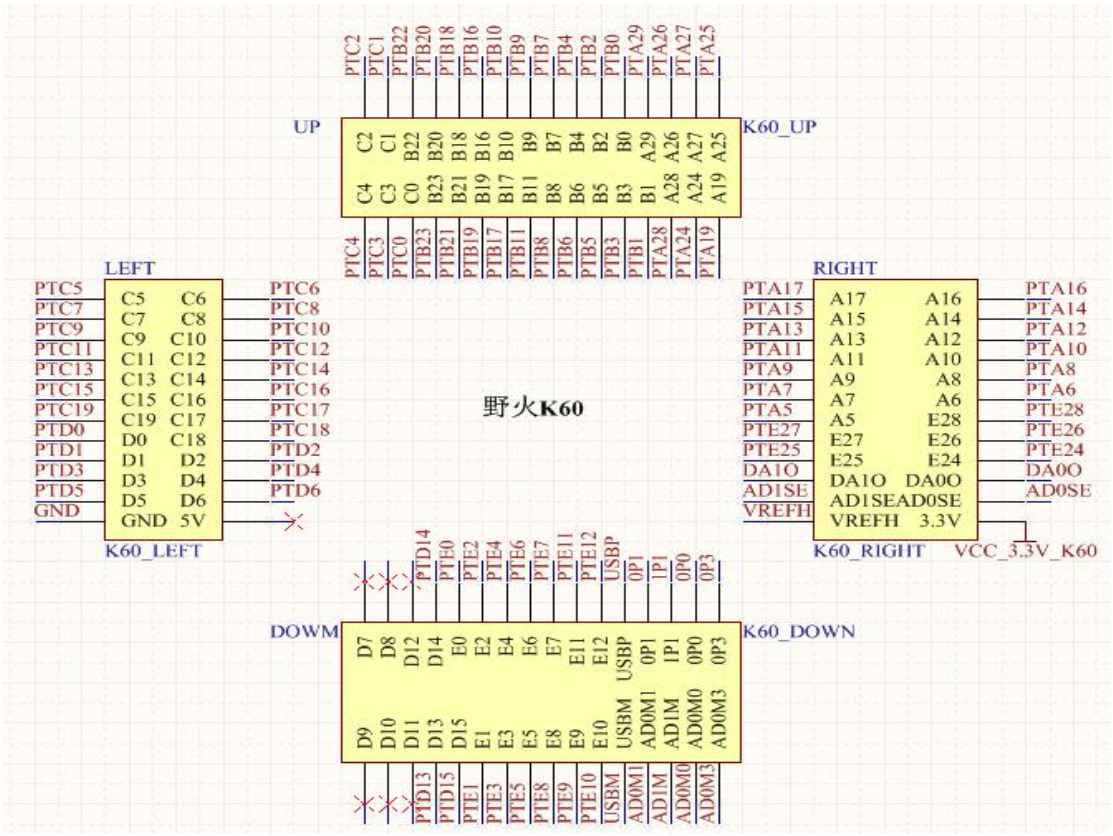

图 3.2　单片机系统原理图

3.3 电源模块设计

电源是智能车的原动力，电源供电的稳定性直接关系到整个系统的稳定性。所以我们要考虑电源的转换效率，尽量降低噪声，防止干扰。比赛规定智能车供电电源只能使用指定型号的7.2V 2000mAh Ni-Cd 电池供电。而系统单片机，摄像头以及其他外围设备如蓝牙模块、OLED模块需要3.3V 电源；电机驱动电压为7.2V；5V电压主要为编码器、数字芯片等功能模块供电。舵机需要6V电源。

由于单片机对电源的电流和电压的稳定性要求相对较高，所以我们决定用TPS7333芯片对其单独供电，以防止受到其他不必要的干扰。其他外围模块单独用一块TPS7333芯片供电。功能模块等用TPS7350供电。而由于总量大所以需要比较大的电流，所以我们采用LM2596为舵机供电，基本满足要求。

3.3V电压，主要供给主控芯片。所以该部分供电电压要求十分稳定，否则将

影响到主控芯片的正常工作。因此我们选用具有稳定输出电压，较大输出电流的TPS7333。基本达到要求。其电路图如3. 3图所示：

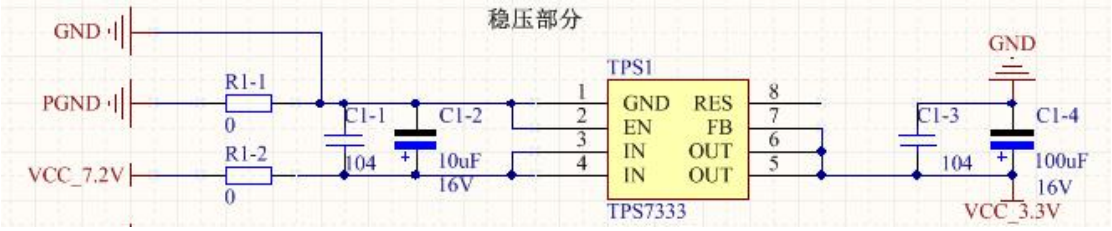

图 3.3 芯片供电电路原理图

舵机的工作电压为4.5V至5.5V。在这个范围之内，电压越高舵机响应速度越快。故我们采用LM2596芯片，该芯片配合外围电路后可调输出电压，我们将电压调到5.5V，使得舵机工作在5.5V。如电路图如3. 4图所示：

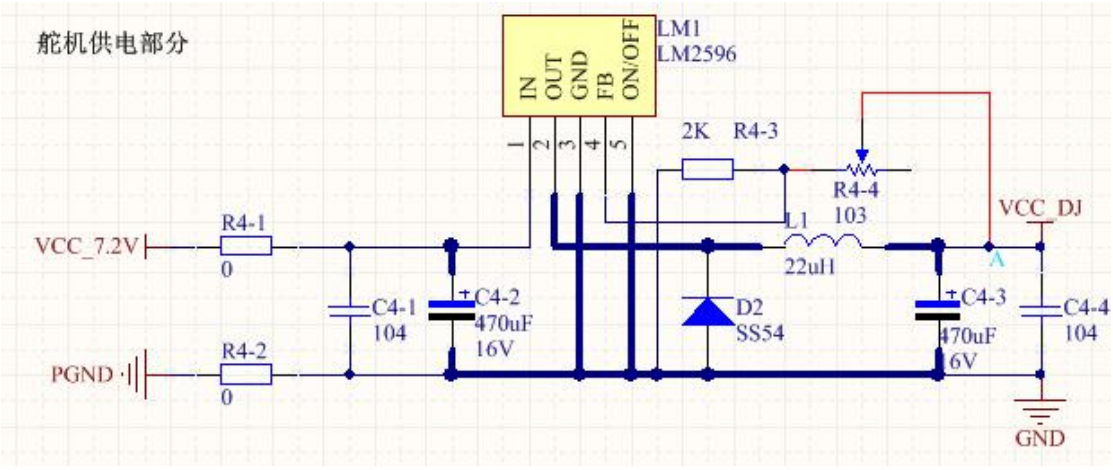

图 3.4 舵机供电电路原理图

3. 4 传感器方案

传感器是小车的眼睛，所以传感器模块对于小车能否快速准确的捕捉路况

信息十分重要。本系统采用山外鹰眼摄像头，鹰眼，基于 OV7725 CMOS 摄像头搭建硬件二值化摄像头，直接输出二值化图像，一次传输 8 个像素，相比与黑白摄像头一次传输一个像素快 8 倍、彩色摄像头两次传输一个像素快 16 倍。这大大提高了我们开发的进度。

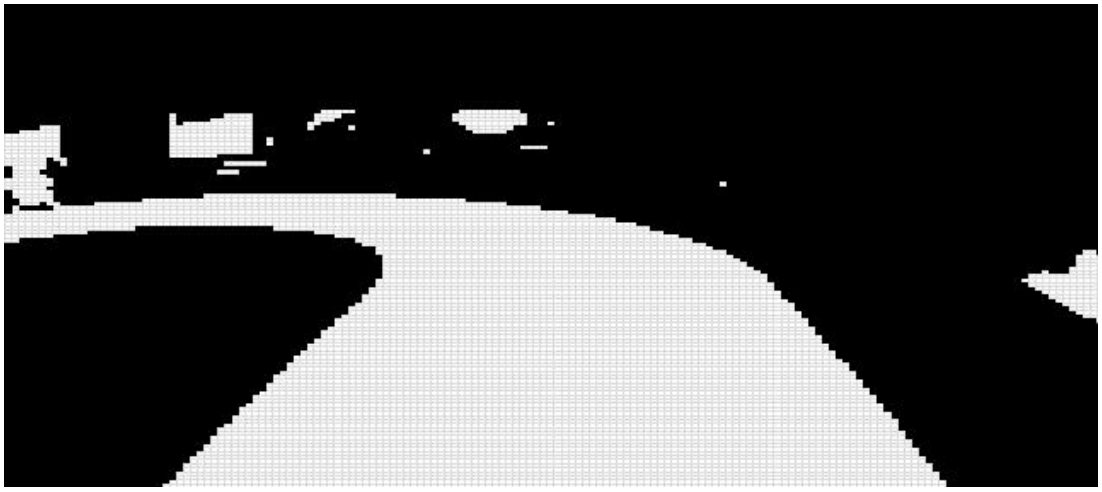

图 3.5 检测到的图像

3.5 驱动电路设计

电机驱动的设计，采用HIP4082+4NMOS构成H桥进行驱动。HIP4082是H桥式N通道MOS管驱动器集成电路，特别针对于脉宽调制电动机控制。它使基于桥式电路的设计更为简单和灵活。MOS管选用IRLR7843，具有大电流内阻极小的特点。其它外围元件还有B0512S，5V升12V集成电源模块，74HC02D，数字控制部分，也当隔离部分。如图3.6所示：

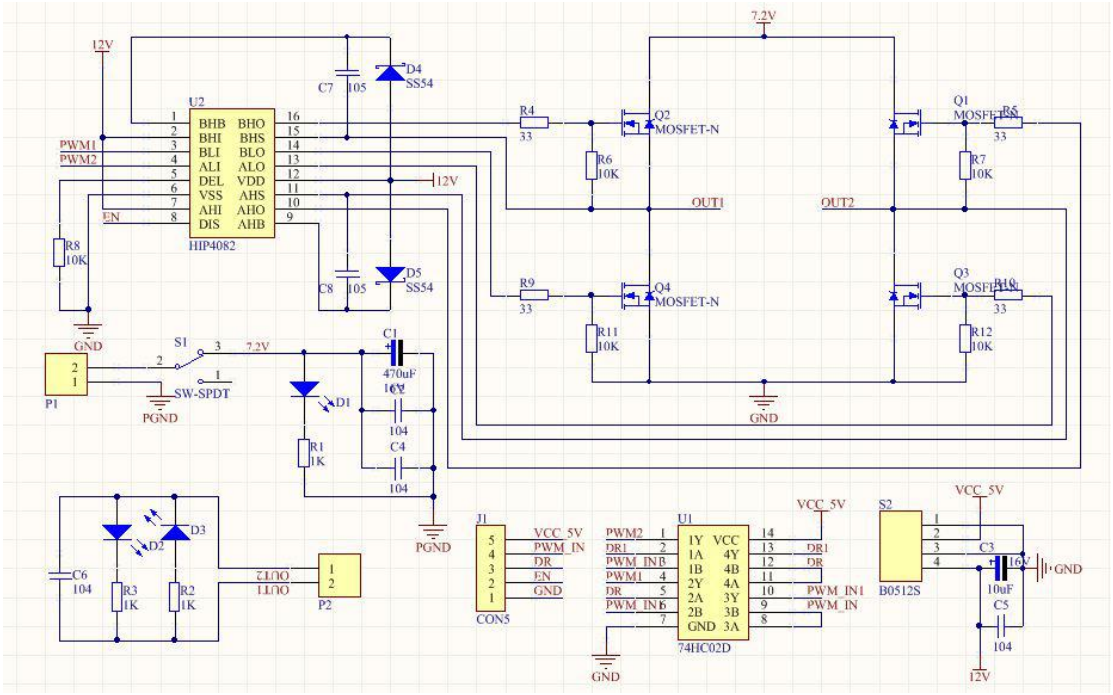

图 3.6 电机驱动模块原理图

原理图及PCB绘制使用Altium designer软件进行设计，这套软件通过把原理图设计、电路仿真、PCB绘制编辑、拓扑逻辑自动布线、信号完整性分析和设计输出等技术的完美融合，提供了全新的设计解决方案，使我们可以轻松进行设计。

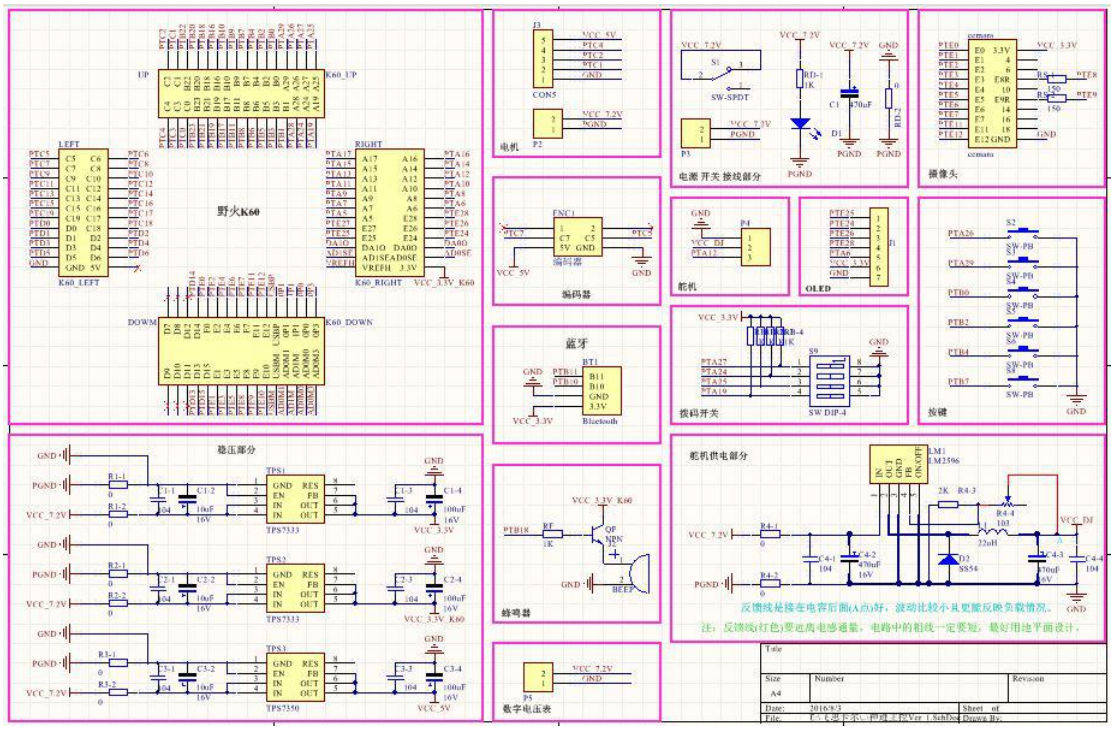

图 3.7 通过 Altium designer 绘制的核心板原理图

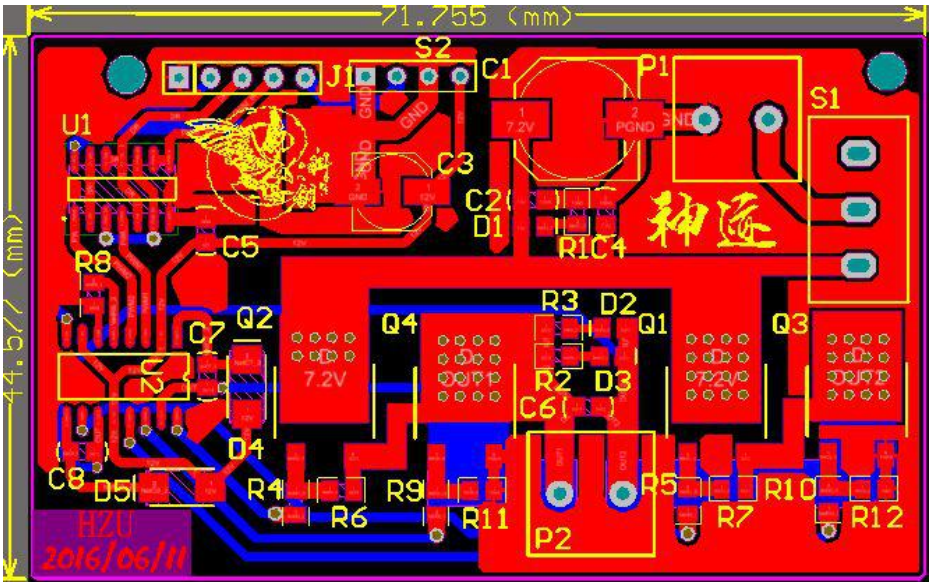

图 3.8 电机驱动 PCB 图

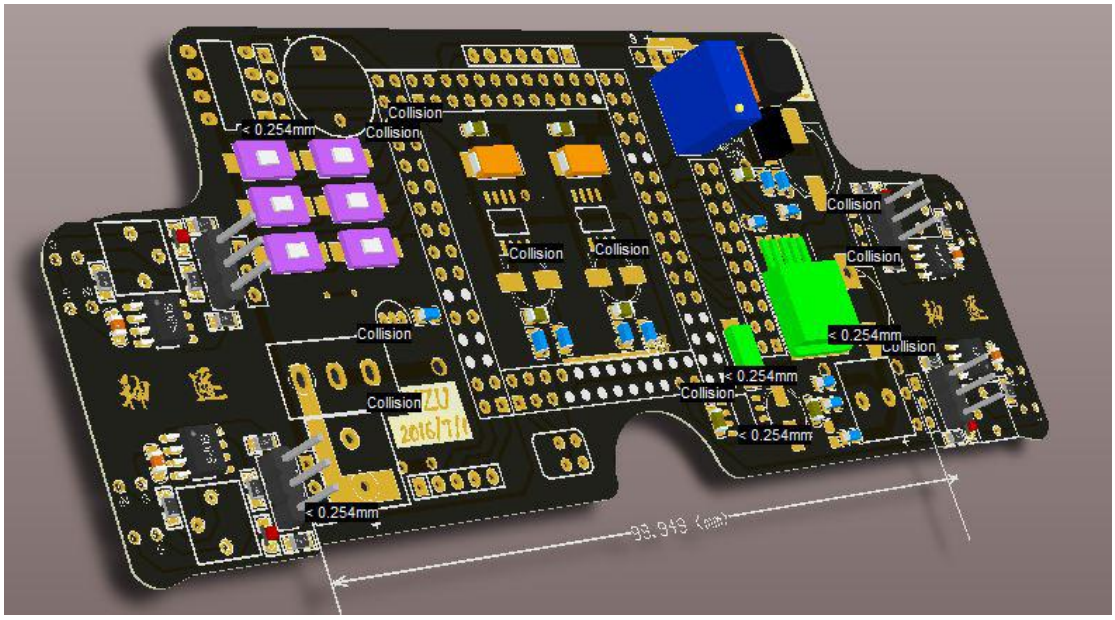

图 3.9 主控板 3D 预览图

第四章 软件设计

系统硬件位于底层，是整个系统的基础，系统软件结构则根据硬件和控制需求来制定。系统的基本软件流程为：首先，对各功能模块和控制参数进行初始化。然后，通过图像采集模块获取前方赛道的图像数据，同时通过速度传感器模块获取赛车的速度。根据采集到的赛道信息，采用 PD 对舵机进行反馈控制。另外根据检测到的速度，结合速度控制策略，对赛车速度不断进行适当调整，使赛车在符合比赛规则的前提下，沿赛道快速行驶。

4.1 黑线的提取和图像中心的计算

4.1.1原始图像的特点及校正

在单片机采集图像信号后需要对其进行处理以提取主要的赛道信息，同时，由于起点线的存在，光线、杂点、赛道连接处以及赛道外杂物的干扰，图像效果会大

打折扣。因此，在软件上必须排除干扰因素，对赛道进行有效识别，并提供尽可能多的赛道信息供决策使用。

因为 CMOS 图像传感器所扫描到的图像是一幅发射式的图像，产生了严重的畸变，如果不进行校正，那么扫描到的数据的正确性将无法保证，那么对于道路形状的判定将会不准确，造成赛车判断失误，走的不是最优路径，严重时甚至冲出轨道。因此，在进行赛车方向控制时，进行坐标的校正显得至关重要。下面对坐标校正的方法进行具体的介绍。

由于 CMOS 图像传感器所扫描到的图像总是向外扩张，扫描到的图像并非为一个标准的长方形，在不考虑纵向畸变的情况下可以近似认为是一个梯形[7]，严重影响到了数据采集的准确性，所以我们必须对扫描到的数据进行校正。我们根据实验发现，越是靠近 CMOS 图像传感器的地方，扫描到的范围就越窄，离 CMOS 图像传感器距离较远的地方，扫描到的范围就越宽，因此，在相同的情况下，如果 CMOS 图像传感器在最近处和最远处采集到

黑线的数字信号距离中心位置均是 30 的话，事实上最远处偏离中心位置的距离可能比最近处偏离中心位置的距离大很多。按照赛车的安装方法，在扫描到的数字量相同的情况下，在第 40 行处偏离中心位置的距离超过了第 0 行偏离中心位置的 4 倍还要多，故必须进行校正，校正的方法就是在不同的行乘以一个不同的系数，系数大小的确定是先进行理论粗略的计算，然后再在事物上进行校正。

4.1.2黑线的提取和中心的计算

黑线提取算法的基本思想如下：

(1) 直接逐行扫描原始图像，根据设定的阈值提取赛道左右两边的黑白跳变点；

(2) 图像数据量大，全部扫描一遍会浪费很多时间，利用前面已经求出的黑线中心位置判断出黑线的趋势，从而推断出下一行的黑线大概位置，确定出扫描范围， 避免整行逐点扫描，节省时间。求黑线中心时，因为近处的黑线稳定，远处黑线不稳定，所以采用由近及远的办法；根据上一行的左右两边的位置，确定本行的黑线扫描范围。在确定的黑线宽度范围内提取有效黑块，这样可以滤除不在宽度范围内的黑点干扰；

(3) 根据采集到的黑线宽度，判断是否采集为真正赛道，滤除噪点造成的影响；

(4) 图像是远处小近处大，所以黑线宽度范围和前后行黑线中心的位置差别都要动态调整；

(5) 比较左右黑线坐标的大小，判断赛道的有效性；

赛道中心线的提取：

由于智能车大赛在第七届对规则进行了改变，其中最为明显的就是将原本在赛道中心的黑线调成了赛道两边，而智能车想要在赛道上平稳的行驶，不冲出赛道，必须知道赛道的中心，为此，我们利用提取的左右两边的黑线坐标对赛道的中心进行了计算，基本思想如下：

（1）如果左右两边找的行数都大于10行，则图形中心为左右黑线坐标的平均值，长度和左右两边中较短的一边相等。

（2）如果两条边有一条边长度小于10行，则根据较长边的坐标，进行那个平移半个赛道的坐标，左边赛道往右移，右边赛道往左移，得到的新坐标就是赛道中心，长度和左右两边中较长的一边相等。

4.2 方向控制方案

4.2.1 方向控制数据的选取

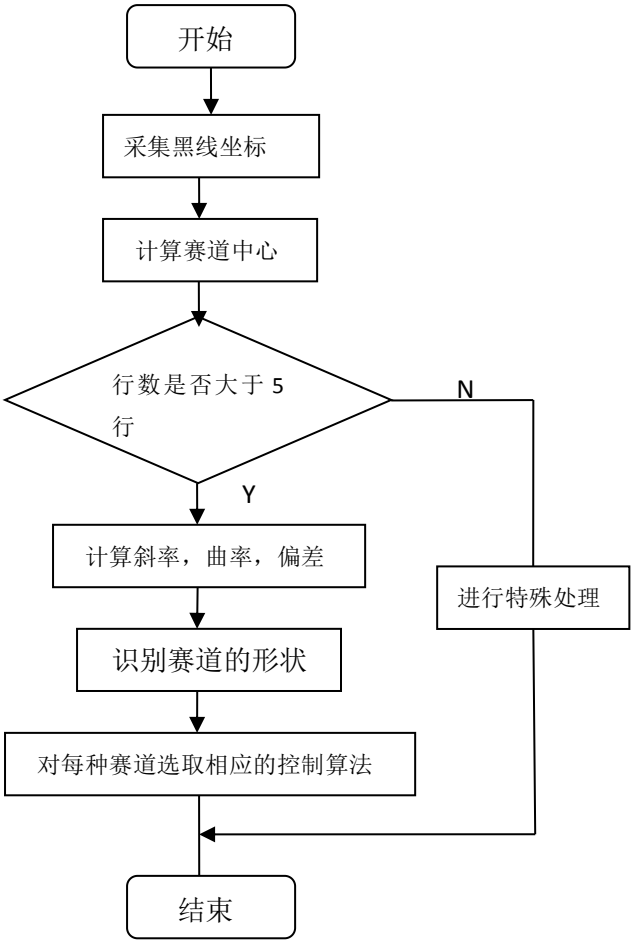

图 4.1 方向控制流程图

根据前面介绍的 CMOS 图像传感器采集处理得到的只是 40 行中每一行的黑线坐标。并没有得出赛道的具体形状，到底是直道还是弯道，是大弯还是急弯，是偏向一个方向的弯道还是“S 型”弯道，是“大 S 型”弯道还是“小 S 型”弯道。这就可以通过曲率的计算来得出。下面给出曲率的具体计算方法。

首先，将扫描到的 40 行中每 9 行分为一组，而且彼此交叉，也就是说，在能扫描到的情况下，划分为从第 0 行到第 8 行，从第 1 行到第 9 行，从第 2 行到第 10 行，依此类推，最后从 21 行到第 29 行，一共可以划分为 22 段，分别计算每一段的曲率，以第一段为例，计算的方法为：将第 i 行的坐标定义为 coordinate[i]，将 CMOS 图像传感器的中心坐标定义为 middle，那么每一行相对于中心坐标的偏差就为

$$ek[i]=coordinate[i]-middle \tag{4.1}$$

再乘以每一行各自的校正值 emendation[i]，计算公式为

$$ek[i]=ek[i]*emendation[i] \tag{4.2}$$

就得到了各行实际偏离中心位置的距离。然后用第 0 行的偏差加上第 8 行的偏差之和除以 2 再减去第 4 行的偏差，得出一个相对偏差 ek_comparatively。ek_comparatively 的计算公式为

$$ek\_comparatively = (ek[8]+ek[0])/2-ek[4] \quad (4.3)$$

然后计算出该段曲线对应的直道，即将第 0 行的黑点与第 8 行的黑点直接相连的直线的长度计算出来，计算公式为

$$distance\_beeline=sqrt((ek[8]-ek[0])*(ek[8]-ek[0])+48*48) \quad (4.4)$$

最后将公式(4.3)除以公式(4.4)，得到最后曲线的弯曲度为 curve_degree。计算公式为

$$curve\_degree=ek\_comparatively/\ distance\_beeline \quad (4.5)$$

就得出了曲线的弯曲程度，然后根据每一段的弯曲程度，最终得出所扫描到的全部道路的形状。

4.2.2 道路形状精确识别算法

利用 4.1.3 中对道路弯曲程度的计算，根据扫描到的道路每段的弯曲程度，可以对道路形状进行较为精确的识别，在进行具体的道路识别之前，先给出几个比较模糊的概念，这是描述弯道的弯曲程度的，弯曲程度非常小，比较小，比较大，非常大。这几个概念在下面对于赛道具体识别时有提及。

（1） 直道的识别

通过 4.2.1 节中对道路弯曲程度的计算，我们可以非常精准的识别出直道，因为如果扫描到的道路为直道，在上面的计算中，在计算相对偏差时，ek_comparatively = (ek[i+8]+ek[i])/2-ek[i+4]肯定会等于 0，因为如果道路为直道，而且采样是根据等间隔采样，第 i+4 点正好是第 i+8 点和第 i 点的中心点，根据直线的特性，不管是横纵坐标，直线的中心点都应该等于起点和终点的一半。就算扫描到的图像存在误差，那么计算出的弯曲程度 curve_degree 也应该是一个很小的数，也就是说，如果赛道的每一段的弯曲程度 curve_degree 都非常小，那么该道路就一定是直道，反过来，只要其中有一段的弯曲程度 curve_degree 比较大，那么就可以判断这段道路不是直道，当然，如果有很多连续的段的弯曲程度 curve_degree 都非常小，那么也可以将连续很小的那一段判断为直道，赛道中直道的形状如图 4.2 所示。

（2） 大弯的识别

通过 4.2.1 节中对道路弯曲程度的计算，不但可以得出道路的弯曲程度，而且还可以得出道路弯曲的方向，因为在计算 ek_comparatively = (ek[i+8]+ek[i])/2-ek[i+4] 时，如果道路弯曲的方向不一样，那么得到的 ek_comparatively 的就有正负号的区分，而 curve_degree 是在 ek_comparatively 的基础上除以了一个距离，距离必定为正，因此 curve_degree 也有正负之分，而 curve_degree 的正负就正好代表了道路弯曲的方向，如果道路在靠近车的部分为

直道，而在远离车的部分向着一个方向弯曲，且弯曲程度 curve_degree 也比较小，那么就可以将这种道路判断为大弯。

（3） 急弯的识别

与大弯基本相同，仍然是经过 4.2.1 节中的计算，得出道路中每段的弯曲程度 curve_degree，如果靠近车的部分的道路的那么得到的弯曲程度 curve_degree 都非常小，而在远离车的部分道路得到的弯曲程度 curve_degree 都非常大，且正负号完全相同，也就是说赛道向同一个方向偏，那么就可以将这种道路判断为急弯，如果在长直道后面出现一个急弯，那么一定要精确识别，如果不能精确识别，及时偏转并减速，那么赛车很可能冲出赛道，因此急弯的识别一定要非常精准。赛道中急弯的形状如图 4.3 所示

（4）“大 S 型”道的识别

“S 型”道是赛车最难走好的部分，但也是最能拉开差距的部分，因此也必须精准识别，“S 型”道又分为“大 S 型”道和“小 S 型”道，这里先讨论“大 S 型”道，首先，“大 S 型”道中的大部分段的弯曲程度 curve_degree 都是非常大的，但在两弯相交处也会出现弯曲程度 curve_degree 比较小的情况，而且“大 S 型”道有一个很显然的特征就是计算出的弯曲程度 curve_degree 有正有负，而且正负都会出现很大的数，因为“大 S 型”道不可能只向一个方向弯，这也是所有“S 型”道所共有的特点，因此，只要计算出道路中的各段的弯曲程度 curve_degree 出现了一个很大的正数和一个很大的负数，那么就可以判断为“大 S 型”道。赛道中“大 S 型”道的形状如图 4.4 所示。

（5）“小 S 型”道的识别

上面已经对“大 S 型”道进行了识别，而“小 S 型”道的识别与“大 S 型”道的识别非常类似，唯一不同的地方就是“小 S 型”道的计算的弯曲程度 curve_degree 的绝对值不会出现很大的数，而是一些比较适中的数，但是它“大 S 型”道的共同特点就是计算的弯曲程度 curve_degree 都是有正有负，而且“小 S 型”道的整幅图像中最左边的黑线和最右边的黑线不应该偏离太多。因此，只要计算出道路中的各段的弯曲程度 curve_degree 的最大值和最小值分别为一个较适中的正数和一个较适中的负数，而且整幅图像中最左边的黑线和最右边的黑线偏离不是太多，那么就可以判断为“小 S 型”道。赛道中“小 S 型”道的形状如图 4.5 所示。

（6）起跑线的识别

全国大学生智能车竞赛规定，智能车必须能够识别起跑线，并在跑完一圈以后能够自动停止在 3 米以内的赛道中，否则在本来的时间基础上加上 1 秒，如果加上 1 秒那必须将速度加大很多才能挽回。然而，当速度本来就较快时，由于智能车本身的结构原因，很难

再进行提速，因为速度过快可能会导致翻车等一系列不利现象。因此，对于起跑线的精准识别显得至关重要。观察图 4.9 可知，起跑线的颜色从左到右依次为“黑”→“白”→“黑”→“白”→“黑”→“白”→“黑”，对应于微控制器内的数字信号即是 0→1→0→1→0→1→0，因此我们可以利用这个特征来识别起跑线。即当这部分图像中出现这样的颜色规律时就判定为是起跑线。当跑完第二圈，第三次检测到起跑线时，表明已经完成了比赛，要在冲过起跑线后刹车，在三米范围内停车。

4.3 分类进行方向控制算法

4.3.1 直道的方向控制算法

对于赛道中的直道，是方向控制中最好处理的一种情况，因为只要小车没有偏离赛道，就可以不进行偏转，而当智能车偏离赛道时也只需要一个较小的偏转，让智能车能缓慢回归赛道就可以了，具体的方法是计算扫描到黑线的偏离中心线的平均值和黑线的斜率，再将这两个数分别乘上各自的比例系数，加上舵机偏向中心位置时需要给出的高电平的值，作为 PWM 波的高电平送入给舵机，就可以实现直道上的智能车方向控制了。

4.3.2 大弯的方向控制算法

在道路中遇到大弯时需要转弯，转向角只受弯道的弯曲程度影响，但是距离大弯多远就开始转弯却需要受到当前速度的影响，因为舵机在转向时有一定的延时，舵机转 60 度大约需要 110 毫秒，而智能车的最快速度可以达到 10 米/秒，如果智能车当时运行的速度就是 10 米/秒，那么当舵机转过 60 度时智能车已经走过 1 米多了，所以不能等到了弯道才开始转弯，而是需要提前转弯，而且速度越快，越是要提前得多，所以智能车转弯提前多少就由速度控制，用比例控制就能收到很好的效果，而转弯时转多少却仅仅受弯道的弯曲程度控制，弯曲程度越大，转向角就也越大，弯曲程度越小，转向角也就越小。具体的对应关系通过实验获得，在此没有列举出具体的数据。

4.3.3 急弯的方向控制算法

急弯的方向控制算法与大弯的方向控制算法相似，也是需要提前转弯，而且提前多少转弯也要受当前速度的控制，速度越大，越要提前转弯，转向角也是用弯道的弯道程度来决定的，急弯比大弯的弯曲程度更加大，因此需要的转向角也要相应增大。急弯的方向控制比大弯更难，很可能出线，因此在急弯的方向控制时更应多加注意。

4.3.4 “大S型”道的方向控制算法

“大 S 型”道一般是指“S 型”道最靠左的黑线和最靠右的黑线之间相差的距离大于 60 厘米，这是因为全国大学生智能车竞赛中的赛道为 60 厘米，冲出赛道就算违规，因此“大 S 型”道是不能直接冲的，但是我们可以转尽量小的弯来使智能车通过的路线为最短，而且转更小的弯也可以使得速度不至于减少太多，实现转尽量小弯的方法是扫描最左边和最右边的点，并得出这两个点中离智能车最近的点是左边还是右边的，如果这个点是最右边的，那么智能车的最佳行驶方向就为这个点靠左 30 厘米的方向，如果这个点是最左边的，那么智能车的最佳行驶方向就为这个点靠右 30 厘米的方向。利用这种算法可以实现智能车走最少的路程，转最小的弯。但是用这种算法智能车有可能偏离赛道的距离超过 30 厘米，为保险起见，这个阈值最好设低一点，根据赛车的运行情况来看，将阈值设为 25 厘米就比较保险了。

4.3.5 “小S型”道的方向控制算法

“小 S 型”道一般是指“S 型”道最靠左的黑线和最靠右的黑线之间相差的距离不大于 60 厘米，智能车过这种弯道时，基本可以从中间直接通过，而不需要转弯和减速。因此，找准“小 S 型”道的正中心显得至关重要，因为只有找准了“小 S 型”道的正中心，智能车才能不转弯不减速地通过“小 S 型”道，如果“小 S 型”道的正中心找得不准，那么智能车就很容易冲出赛道。找“小 S 型”道的正中心的方法为：先找出“小 S 型”道最左边和最右边的坐标值，然后将最左边和最右边的坐标值相加之后再除以 2，就得到了“小 S 型”道的正中心坐标，智能车就可以沿着这个方向前进，而不需要转弯。但是，由于 CMOS 图像传感器扫描到的图像

可能出线误差，那么智能车就有可能产生误判，而冲出赛道。因此为了保险起见，必须当“小 S 型”道足够小时，才能采用直接冲的办法。通过本智能车的调试，建议当“S 型”道最靠左的黑线和最靠右的黑线之间相差的距离不大于 40 到 50 厘米才能采用这种算法。

4.3.8 起跑线的方向控制算法

起跑线的作用是让智能车可以在跑完两圈以后自动停车，也即是第三次检测到时，当检测到的次数不足三次时，应该直接忽略，当检测次数到达三次时就应该直接停车，也不应该影响到智能车的方向控制。

4.4 速度控制方案

4.4.1 速度控制的重要性

要想智能车以最短的时间跑完整个赛道，不但要求优越的方向控制算法，让智能车尽量走更近的路，转更少的弯。而且对于速度控制算法也有很高的要求，智能车在行驶过程中，不能够完全匀速，在直道上应该提速，而在弯道上应该减速。

在智能车沿着黑线行驶的过程中，速度不能够恒定不变，而应该根据道路的具体形状进行适当的调整。当然，最好的方法是让智能车始终以最快的速度通过整个赛道，但是如果速度始终调得太高，当小车遇到急弯或者是圆道时必然无法转过来，就会冲出赛道。因此不能够始终给最快的速度。然而，如果速度调得始终过低，就无法以最短的时间通过赛道。因此，速度应该要随着检测到的道路形状变化而变化。例如当前方道路为直道或类似直道时，应该将 PWM 波提得最高，让通过主电机的电流达到最大，从而让智能车以最快的速度行驶。当进入弯道时，应该将 PWM 波降到最低，让通过主电机的反向电流达到最大，从而让智能车以最短的时间将速度降到理想状态。

4.4.2 速度控制的算法设计

智能车速度控制的总体思路与智能车方向控制的思路完全相同，也是将道路的形状进行分类，对于不同的类给以不同的速度。而在某一时刻给出智能车

的理想运行速度以后，采用棒－棒控制与 PID 控制相结合的算法实现对智能车的速度调节，以最短的时间使智能车达到这一理想速度。从而实现希望智能车的速度完全由道路情况所控制。

智能车速度控制的程序流程图如图 4.6 所示。

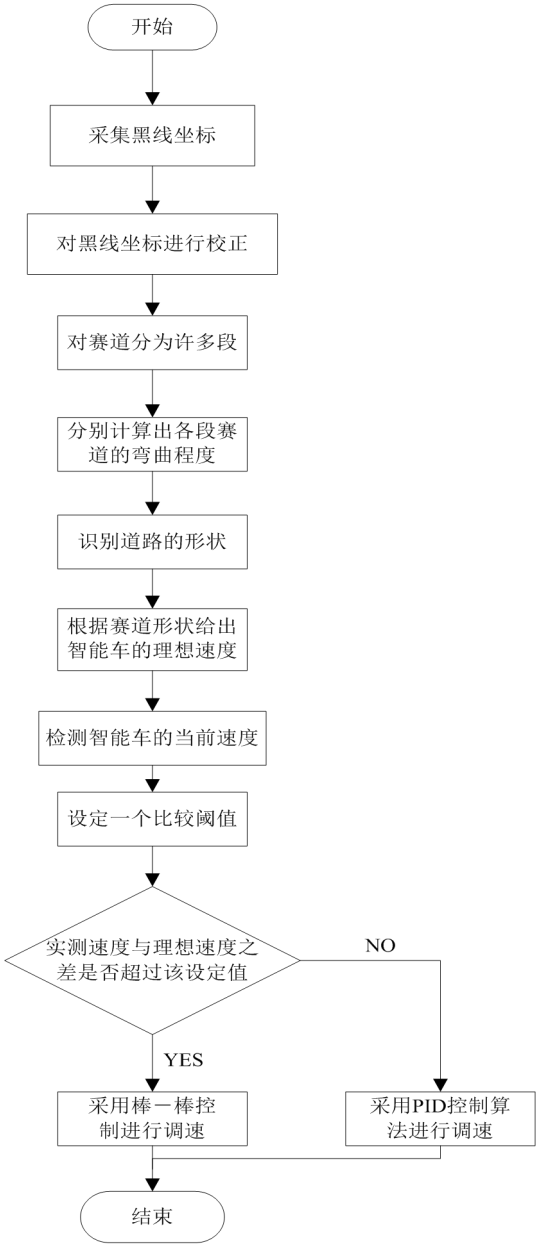

图 4.6 智能车速度控制总体流程图

4.5 棒－棒控制算法

4.5.1 棒－棒控制算法简介

棒－棒控制的基本思路是当检测到的输出量大于理想输入量时，就急剧将

控制量增加到最大，让输出量以最快的速度减少，当检测到的输出量小于理想的输入量时，就急剧将控制量降低到最小，让输出量以最快的速度减少。

4.5.2 棒－棒控制算法用于该赛车系统

本赛车系统的速度控制采用了棒－棒控制与PID控制相结合的方式。其基本思路为：智能车的理想速度由当前的道路信息决定。当光电编码器检测到的速度与理想速度相差较远时，采用棒－棒控制；当光电编码器检测到的速度与智能车的理想速度逐渐接近以后，就采用PID控制。对主电机的控制是通过一个H桥，将H桥的两个口分别接单片机的两个管脚，在本系统中，H桥的两端分别接单片机口中的PTP3和PTP1，当PTP1为高电平，PTP3为高电平时，主电机全速正转，当PTP1为低电平，PTP3为低电平时，主电机全速反转。采用棒－棒控制进行速度控制的原理就是检测智能车的当前速度，如果当前速度低于理想的速度，就PTP3置为高电平，从PTP1口输出一路占空比很高的PWM波，电机就能以最大的加速度升速。反之，就把PTP3置为高电平，从PTP1口输出一路占空比很大的PWM波，从而让电机以最大的加速度降速。

4.5.3 棒－棒控制与PID控制两种方法结合的必要性

棒－棒控制算法由于其快速性非常好，而本赛车系统的速度调节也正好要求很好的快速性，因而在本智能车的调速过程中用到了棒－棒控制算法也就理所当然，但是，棒－棒控制算法在用于本赛车调速时也存在着一定的不足，由于主电机的数学模型为一个惯性环节，存在着一定时间的延时，因此如果仅仅使用棒－棒控制一种算法可能导致超调，也不可能达到稳定，且在任何时刻通过主电机的电流都始终为最大，只是在不同的时刻可能会出现一正一反。正是出现一正一反的这种情况，对主电机以及主电机驱动电路的冲击最大，既造成整个智能车功耗的增加，又可能导致主电机及主电机驱动芯片因电流冲击过大而烧坏。因而在智能车的实际检测速度与理想速度接近时，可以采用 PID 控制算法，这样就既能实现调速的快速性，又能实现速度的平稳性，降低智能车的整体功耗，达到很好的调速效果。

4.6 PID 控制算法

4.6.1 PID控制算法简介

PID 控制是工程实际中应用最为广泛的调节器控制规律。问世至今 70 多年来，它以其结构简单、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。单位反馈的 PID 控制原理框图如图 4.7 所示。

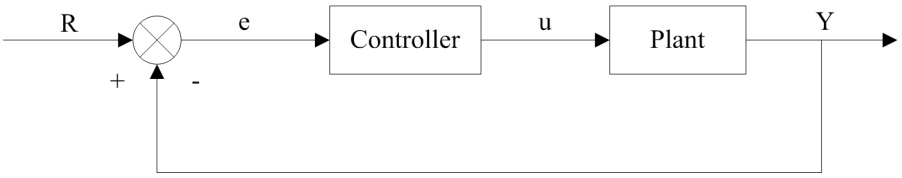

图 4.7 单位反馈的 PID 控制原理图

单位反馈 e 代表理想输入与实际输出的误差，这个误差信号被送到控制器，控制器算出误差信号的积分值和微分值，并将它们与原误差信号进行线性组合，得到输出量 u。

$$u(t) = k_P[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de}{dt}] \tag{4.6}$$

对应的模拟 PID 调节器的传递函数为

$$D(s) = \frac{U(s)}{E(s)} = K_P(1 + \frac{1}{T_I s} + T_D s) \tag{4.7}$$

其中，K_P 为比例增益，K_P 与比例带 δ 成倒数关系即 $K_P = \frac{1}{\delta}$，T_I 为积分时间常数，T_D 为微分时间常数，$u(t)$ 为控制量，$e(t)$ 为偏差。

4.6.2 数字PID控制算法

1. 数字PID位置型控制算法

在单片机控制系统中，PID 控制规律的实现必须用数值逼近的方法。当采样周期相当短时，用求和代替积分，用后向差分代替微分，使模拟 PID 离散化变为差分方程。

为了便于单片机实现，必须把公式(4.6)变换成为差分方程，为此可作如下近似

$$\int_0^t e(t)dt \approx \sum_{i=0}^{k} Te(i) \tag{4.8}$$

$$\frac{de(t)}{dt} \approx \frac{e(k)-e(k-1)}{T} \tag{4.9}$$

式中，T 为采样周期，k 为采样序号。

由公式(4.6)、公式(4.7)、公式(4.8)可得数字 PID 位置型控制算式为

$$u(k) = K_P[e(k) + \frac{T}{T_I}\sum_{i=0}^{k} e(i) + T_D \frac{e(k)-e(k-1)}{T}] \tag{4.10}$$

公式(4.10)表示的控制算法提供了执行机构的位置$u(k)$，所以被称为数字 PID 位置型控制算式。

2. 数字PID增量型控制算法

由公式(4.10)可以看出，位置型控制算法不够方便，这是因为要累加偏差$e(i)$，不仅要占用较多的存储单元，而且不便于编写程序，为此可对公式(4.5)进行改进。

根据公式(4.10)不难写出$u(k-1)$的表达式[8]，即

$$u(k-1) = K_P[e(k-1) + \frac{T}{T_I}\sum_{i=0}^{k-1} e(i) + T_D \frac{e(k-1)-e(k-2)}{T}] \tag{4.11}$$

将公式(5.10)和公式(5.11)相减，即得数字 PID 增量型控制算法为

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

$$= K_P[e(k)-e(k-1)] + K_I e(k) + K_D[e(k-1) - 2e(k-1) + e(k-2)] \tag{4.12}$$

其中 $K_P = \frac{1}{\delta}$ 称为比例增益

$K_D = K_P \frac{T}{T_I}$ 称为积分系数

$K_D = K_P \frac{T_D}{T}$ 称为微分系数

我们在实际代码实现时，处理成

error[0]=0；

error[1]=0；

error[2]=0；

PreU=0；

for(；；)

{error[0]=error[1]；

error[1]=error[2]；

error[2]=r-y；

PreU+= K_P *(error[2]-error[1])+ K_I *error[2]；

PreU+= K_D *（error[2]-2*error[1]+error[0]）；

……

}

其中，error[2]为本次的偏差，error[1]为上一次的偏差，error[0]为上上次的偏差，PreU 为输出的控制量。

这种算法用来控制步进电机特别方便，对直流电机也可以采用，其实如果对控制有更高的要求或者干扰因素较多，我们可以对 PID 算法做各种改进，比如用梯形法做数值积分以提高精度，将差分改成一阶数字滤波等等，在实际调车的过程中，我们确实遇到过由于光电编码器采样得到的脉冲上升下降沿不够陡，使得速度采样出现不稳定和失真，但由于这些附加处理比较耗费代码的运行时间，出于代码效率和实际效果的比较，我们没有采用这些改进的方案，另外可以考虑加反向器来整波形得到较为理想的方波。

运用 PID 控制的关键是调整三个比例系数，即参数整定。PID 整定的方法有两大类：一是理论计算整定法。它主要是依据系统的数学模型，经过理论计算确定控制器参数。由于智能车整个系统是机电高耦合的分布参数系统，并且要考虑赛道具体环境，要建立精确的智能车运动控制数学模型有一定难度，而且我们对车身机械结构经常进行不断修正，模型参数变化较频繁，可操作性不强；二是工程整定方法，它主要依赖工程经验，直接在控制系统的试验中进行，且方法简单，我们采用了这种方法，同时，我们先后实验了几种动态改变 PID 参数的控制方法[8]。

4.7 经典 PID 算法实现速度的控制

4.7.1 经典PID算法实现速度控制的计算机仿真

智能车的主电机可以看着是一个惰性单元，应用 PID 控制算法可以达到较好的控制效果，该智能车在 MATLAB 上进行仿真得到的单位价跃响应如图 4.12 所示，因为在本赛车速度控制系统中，其中一个最重要的指标就是快速性，而对于超调量和过渡过程时间要求并不严格。特别是过渡过程时间，该智能车的速度时时刻刻都需要发生变化，所以还等不到速度稳定便又要变化，由仿真结果可见，该智能车速度调节的快速性很好，能够完全满足调速的要求。

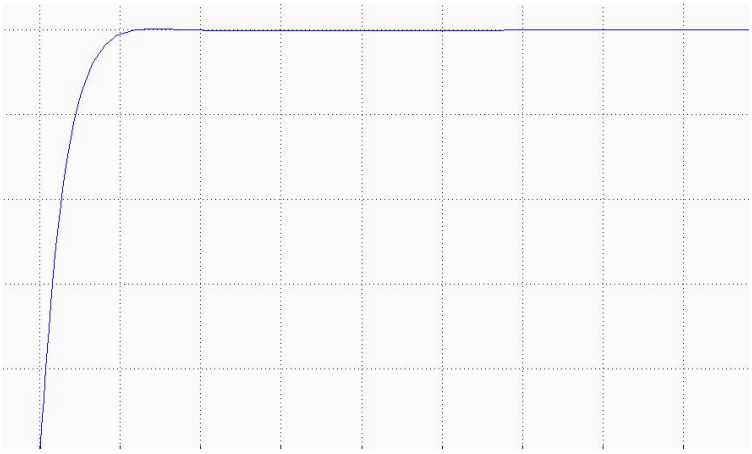

图 4.8　PID 速度控制算法的 MATLAB 仿真

4.7.2 经典PID算法实现速度控制的具体实现

我们对速度的控制采用了增量式 PID 算法。该算法的基本策略是急弯降速，直道和小弯提速，由于我们的摄像头扫描到的范围较宽，所以在过急弯道时可以提前减速，先把扫描到的图像进行处理，然后得出经过反复实验，将图像经过算法处理后得到的黑线位置和对应的速度 PID 参照速度处理成二次曲线的关系。在实际测试中，发现小车直道和弯道相互过度时加减速比较灵敏，与舵机转向控制配合较好。

但是，存在的局限一方面是车在弯道进直道后的加速和直道入弯道的速度控制并达不到最好的控制效果，弯道入直道减速不够快速，直道入弯道加速得时机也不够及时。我们做了进一步的改进，根据入弯时黑线位置的特点动态改变二次曲线中最高点（直道的最高速度）和最低点（弯道的最低速度）的大小，使得控制效果更合理。

另一方面是没有考虑到实际比赛中长直道急速冲刺的情况，赛前在程序中人为设定直线速度不够灵活合理，所以在程序中根据赛道状态动态提高直线速度，使得我们可以在长直道的赛场充分发挥车的潜能。

4.7.3 PID参数调节

比例控制能迅速反应误差，从而减少误差，但是比例控制不能消除静态误差，K_P 的加大，会引起系统的不稳定；积分控制的作用是，只要系统存在误差，积分控制作用就不断地积累，输出控制量以消除误差，因而，只要有足够的时间，积分控制将能完全消除误差，积分作用太强会使系统超调加大，甚至使系统出现振荡现象；微分控制可以减小超调量，克服振荡，使系统的稳定性提高，同时加快系统的动态响应速度，减少调整时间，从而改善系统的动态性能。而在该智能车系统调速的过程中，由于快速性是最重要的指标之一，因而应主要考虑其快速性，在调试的过程中，把 K_P 参数调得较大，这是为了增大速度调节的快速性，将 K_D 参数调得较为适中，因为除了快速性以外，最重要的就是要减少系统的超调量，而将 K_I 参数调得较小，因为在该速度调节中，并不需要达到静态，由于理想速度就在时时刻刻发生变化，所以也不可能达到静态。实验结果表明，K_P 参数取 8，K_D 参数取为 2，而 K_I 参数取得尽可能小，大约为 1 时，可以取得较好的效果。

4.8 控制系统实时性能分析

智能车系统的控制周期为 20ms，也就是每一场对车的转向和速度进行控制。新一场信号采集的同时，单片机既要相应行中断采集当前场的数据，又要在相应中断剩下的时间分析出上一场数据，集合相应的算法进行控制量的输出。在这 20ms 中单片机能否完成上面所说的任务是需要通过分析所得的。如果单片机不能在 20ms 内完成上述的任务，系统控制性能就会下降，甚至出现错误。如果可以在低于 20ms 的时间内完成任务，那么系统就能够按照一定的时序有条不紊的运行。引入多任务下单片机负载的概念[9]：

$$L=\frac{T_{t1}}{T_{p1}}+\frac{T_{t2}}{T_{p2}}+\frac{T_{t3}}{T_{p3}}+....+\frac{T_{t4}}{T_{p4}} \qquad (4.13\ )$$

下面是各个任务执行时间：

数据的计算 4ms-6ms 、摄像头数据采集 6ms－ 8ms、速度采集 1us、速度控制 1us、舵机控制 1us。可以计算出系统的负载 L 大约为 0.7，能够满足系统稳定性的要求。

第五章 总结

自去年10月开始学校的智能车选拔，我们已经走过12个月的历程，一路拼搏，一起参加了校赛、赛区赛，这一路上我们历经风风雨雨，走到现在。

在小车的各方面设计，我们都有详细的分析和思考，并尽自己的最大能力将整车的结构与算法调整至最佳水平。与此之外我们在各个细节方面进行了优化。现在，面对即将到来的大赛，我们充满了信心，在这几个月的备战中，非常感谢实验室的大力支持，在此特别感谢一直支持和关注智能车比赛并给我们悉心指导的老师和学长。同时也感谢比赛组委会能组织这样一项很有意义的比赛。

第六章 车模规格

赛车基本参数	长： 30cm
	宽： 19cm
	高： 32cm
车重	1260g

功耗	空载: 8W
	带载：大于10W
电容总容量	1700uF
传感器	编码器
	CMOS数字摄像头
	红外对管4个
除了车模原有的驱动电机、舵机之外伺服电机个数	0
赛道信息检测	视野范围（近瞻/远瞻）40cm/150cm
	精度(近/远) 2/10cm
	频率 100Hz

参考文献

[1] 邵贝贝. 嵌入式实时操作系统[LC / OS-Ⅱ(第 2 版)[M]. 北京. 清华大学出版社. 2004

[2] 谭浩强，C语言程序设计[M].清华大学出版社，2006.1

[3]新型PID控制及其应用2002年作者：陶永华主编

[4]周金萍等《MATLAB6.5 实践与提高》，中国电力出版社

[5] 易大义,“计算方法[M]”, 浙江大学出版社，1995

[6] 俞斯乐著，“电视原理”，国防工业出版社，2005.8.1

[7] 余永权,“单片机在控制系统中的应用”，电子工业出版社，2003.10

[8] 卓晴，黄开胜，邵贝贝，“学做智能车”，北京航空航天大学出版社

[9] 邵贝贝著，“单片机嵌入式应用的在线开发方法”

[10] 余志生，“汽车理论-(第 4 版)”， 机械工业

[11] 邓兆祥，褚志刚等，“汽车前轮定位参数优化设计”，重庆大学机械传动国家重点实验室

[12] 邓鲁华等，“数字图像处理（原书第 3 版）”，机械工业出版社

[13]Michael J.Young， “Visua l C++6 从入门到精通”，电子工业出版社

附录

附录 A 部分源程序

```
/**********************************  总  初  始  化  函  数
****************************************/
/*******************************  以  及  中  断  服  务  函  数
****************************************/


#include "common.h"
#include "include.h"
#include "init_all.h"


//蜂鸣器
#define BUZZER PTB18
//#define Buzzer_init gpio_init(BUZZER,GPO,1)

void Mie(void)  {GPIO_PDOR_REG(GPIOX_BASE(BUZZER))  |= (1 << PTn(BUZZER));}   //
开
void Bi(void)  {GPIO_PDOR_REG(GPIOX_BASE(BUZZER)) &= ~(1 << PTn(BUZZER));}  //
灭

extern uint8 end;
extern int16 turn;
extern uint8 inf_delay;
extern int16 right_speed_count;
extern int16 left_speed_count;
uint8 buff[CAMERA_SIZE];              //原始图像
float uart_temp = 0;                  //参数变量，方便调参数

uint8 img_flag;                       //采集完成标志
uint8 pit_flag, steer_flag, motor_flag;              //pit 标志位 1 为 10ms

//int32 expect_speed = 0;      //初始化期望速度           速度控制放在 PID 里
```

```
extern reg_s ov7725_eagle_reg[];

void Stop()
{
      if(((gpio_get       (PTB17)==1)&&(gpio_get       (PTD15)==1))||((gpio_get
(PTB21)==1)&&(gpio_get      (PTD5)==1))||((gpio_get      (PTB21)==1)&&(gpio_get
(PTD15)==1))||((gpio_get (PTB17)==1)&&(gpio_get (PTD5)==1))&&turn<50&&turn>-50)
    {
//      gpio_init(PTB18,GPO,1);
        end = 1;
    }
}

void init_all()         //开关中断都在里面
{
     uart_init (UART3,9600);
   //led_init(LED_MAX);                //四颗灯。。。。PTB20, PTB21, PTB22, PTB23
   // led_init(LED0);
  //  gpio_init(PTB23, GPO, 1);
  //  gpio_init(BUZZER,GPO,1);  //蜂鸣器
    key_init(KEY_MAX);
    LED_Init();     //OLED 初始化
    gpio_init(PTD3, GPI, 0);           //左轮行驶方向
    gpio_init(PTC7, GPI, 0);           //右轮
    gpio_init(PTC6, GPI, 1);           //灯塔
    gpio_init(PTA4, GPO, 0);           //静电
    gpio_init(PTC2, GPO, 0);        //方向
    gpio_init(PTC1, GPO, 0);        //使能，0 有效
    gpio_init(PTA19, GPI, 0);
    gpio_init(PTA24, GPI, 0);
    gpio_init(PTA25, GPI, 0);
    gpio_init(PTA27, GPI, 0);
    gpio_init(PTB17, GPI, 0);
    gpio_init(PTB21, GPI, 0);
    gpio_init(PTD5, GPI, 0);
    gpio_init(PTD15, GPI, 0);
    pwm_init();
    FTM_QUAD_Init(FTM2);                                        //FTM2 正交解码初
始化（所用的管脚可查 vcan_port_cfg.h ）
```

```
    FTM_QUAD_clean(FTM2);

 //   lptmr_delay_ms();
    lptmr_pulse_init(LPT0_ALT2, COUNT, LPT_Rising);          //初始化脉冲计数器，
用 PTA19 输入,上升沿触发
    lptmr_pulse_clean();
    set_irq_priority(PIT1_IRQn,0);
    set_irq_priority(PORTB_VECTORn,1);
    set_irq_priority(PORTD_VECTORn,2);



    port_init(PTB17, IRQ_RISING | ALT1);
    port_init(PTB21, IRQ_RISING | ALT1);
    port_init(PTD5, IRQ_RISING | ALT1);
    port_init(PTD15, IRQ_RISING | ALT1);
  //  flash_init();
    /*
    while(1)
    {
         if(key_check(KEY_D) == KEY_DOWN) //检测 key 状态（带延时消抖）
         {
           DELAY_MS(100);          //防止数据跳动
           if(key_check(KEY_D) == KEY_DOWN)
           {
              ov7725_eagle_reg[45].val++;
           }
         }

         LED_PrintValueI(83,1,ov7725_eagle_reg[45].val);



         if(key_check(KEY_ENTER) == KEY_DOWN)
         {
               DELAY_MS(100);          //防止数据跳动
               if(key_check(KEY_ENTER) == KEY_DOWN)
               {
                 LED_Fill(0x00);
                 break;
               }
```

```
        }
    }*/
    camera_init(buff);              //摄像头初始化    目的地址 buff     SCCB（PTA25 SDA   PTA26 SCL）
                        //DMA(PTA27 PCLK(默认上升沿),源地址为PTB_B0_IN:PTB0~PTB7) PTA29 场中断

    set_vector_handler(PORTB_VECTORn, PORTB_IRQHandler);       //设置 PORTB 的中断

    set_vector_handler(DMA0_VECTORn, DMA0_IRQHandler);         //设置 DMA0 的中断

    set_vector_handler(PORTD_VECTORn, PORTD_IRQHandler);       //设置 PORTD 的中断

    set_vector_handler(PORTE_VECTORn, PORTE_IRQHandler);

 //    set_vector_handler(UART3_RX_TX_VECTORn,UART3_IRQHandler);  //设置串口接受中断 TX PTC17  RX PTC16



 //  pit_init_ms(PIT0, 5);             //PIT 定时器初始化,20ms 定时，脉冲读取  调试用
    pit_init_us(PIT1, 500);               //start 后用 0.5ms


 // set_vector_handler(PIT0_VECTORn ,PIT0_IRQHandler);       //设置 PIT0 的中断服务函数为 PIT0_IRQHandler
   set_vector_handler(PIT1_VECTORn ,PIT1_IRQHandler);       //设置 PIT0 的中断服务函数为 PIT0_IRQHandler
 //  uart_rx_irq_en (UART3);                                   //开串口接收中断


  //  enable_irq (PIT0_IRQn);                                 //使能 PIT0 中断，调试

    camera_get_img();                   //取图像

  //   flash_init();            //flash 初始化
```

```
   // while(1)
   // show_team_name();          //开机图片

    brick_init();               //砖头参数初始化
    Bi();
    DELAY_MS(10);
    Mie();

    select();              //主界面  按键调试

    PID_init();                         //PID 参数初始化


  //  show_final();               //显示最终参数
  //  LED_Fill(0x00);
  //  disable_irq(PIT0_IRQn);           //关掉 PIT0，调试结束

    enable_irq(PORTB_IRQn);
    enable_irq(PORTD_IRQn);
    enable_irq (PIT1_IRQn);           //使能 PIT1，真正的开跑
}



/*****************************  以 下 均 为 中 断 函 数
*****************************/

extern int32 c;

void PORTB_IRQHandler()
{
     uint8 n = 0;
     uint32 flag = PORTB_ISFR;
     PORTB_ISFR = ~0;
     n = 17;
     if(flag & (1 << n))
     {
        Stop();
     }
```

```
    n = 21;
    if(flag & (1 << n))
    {
   Stop();
    }
}

void PORTD_IRQHandler()
{
    uint8 n = 0;
    uint32 flag = PORTD_ISFR;
    PORTD_ISFR = ~0;
    n = 5;
    if(flag & (1 << n))
    {
//        PORTD_ISFR  = (1 << n);      //写 1 清中断标志位
    Stop();
    }                        //用不上
    n = 15;
    if(flag & (1 << n))
    {
//   PORTD_ISFR  = (1 << n);      //写 1 清中断标志位
    Stop();
    }
}

void PORTE_IRQHandler()   //PORTE 中断服务函数（鹰眼的场中断）
{
    uint8 n = 0;
    uint32 flag = PORTE_ISFR;
  //   PORTA_ISFR = ~0;      //写 1 清中断标志位

    n = 9;
    if(flag & (1 << n))
    {

       camera_vsync();   //场中断
    }
#if  0                       //用不上
    n = 28;
```

```
     if(flag & (1 << n))
     {
        camera_href();    //行中断
     }
#endif
}

void DMA0_IRQHandler()     //DMA0 中断
{
     camera_dma();
     img_flag = 1;        //采集完成
}


void UART3_IRQHandler()        //串口中断接收函数，调节 PID 参数用的
{
    char temp;

    UARTn_e uratn = UART3;

    if(UART_S1_REG(UARTN[uratn]) & UART_S1_RDRF_MASK)   //接收数据寄存器满
    {

        uart_getchar   (UART3, &temp);                      //无限等待接受 1 个字
节

    }
}




void PIT0_IRQHandler()        //定时器 0 中断函数，调试
{

  // img_flag++;
  if(img_flag>5)
  {
    img_flag = 0;
```

```
  }

    if(gpio_get(PTC7))                                  //前进
    left_speed_count = -lptmr_pulse_get();              //右轮速度
    else
    left_speed_count = lptmr_pulse_get();              //右轮速度

    if(gpio_get(PTD3))                                 //左轮方向
    right_speed_count =  -FTM_QUAD_get(FTM2);          //左轮速度
    else
    right_speed_count =  FTM_QUAD_get(FTM2);          //左轮速度

    FTM_QUAD_clean(FTM2);
    lptmr_pulse_clean();
 /*

    PIT_Flag_Clear(PIT0);       //清中断标志位

}
```