第七届全国大学生“飞思卡尔”杯
智能汽车竞赛

技 术 报 告

学　　校：　　兰州交通大学

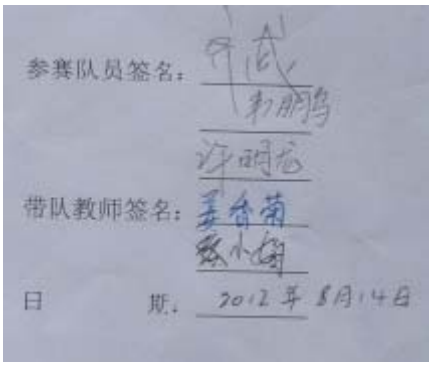
队伍名称：　　神舟 1 号

参赛队员：　齐威　许明龙　韦鹏

带队教师：　　姜香菊　路小娟

关于技术报告和研究论文使用授权的说明

本人完全了解第七届“飞思卡尔”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会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。



摘　　要

本文介绍了在准备第七届“飞思卡尔”杯全国大学生智能汽车竞赛过程中“神舟 1 号”的队员们的设计工作。

硬件平台采用 K60 系列微控制器作为系统控制处理器，以大赛组委会统一规定的 A 型车模为系统车模，主要的控制对象为转向舵机和驱动电机，软件的调试平台采用 Keil MDK 进行软件开发，并用 Visual C++6.0 开发了智能车上位机进行辅助调试。

文中介绍了智能小车控制系统的软硬件结构和开发流程。整个系统涉及车模机械结构调整、单片机控制电路的设计、摄像头图像信号的处理、车模控制算法和控制策略优化等多个方面。为了提高智能车的行驶速度和稳定性，借助蓝牙串口技术进行上下位机通信，经过大量的底层调试和上层仿真，最终确定了现有的系统结构和各项控制参数。

关键词：智能车；摄像头；图像处理

Abstract

This article describes the results of the Shenzhou1 players' work in the process of preparing the seventh Freescale smart car Competition.

The system uses the K60 series microcontroller to control the motor and steering servos to uniform provisions of the Competition Organizing Committee for Model A car. Keil MDK software platform is used to debug software and monitoring software platform developed by VC 6.0 to debug the system.

The hardware and software architecture and development process of the intelligent car control system are introduced in the paper. The whole system includes car model mechanical structural adjustment, microprocessor control circuit design, the camera image signal processing, control algorithms and optimal strategies and other aspects. In order to improve the speed and stability of the intelligent car, the upper and lower computer communication is completed with the Bluetooth serial port technology. Ultimately the system structure and the control parameters are determined after a large number debugging and simulation.

Key words: smart car, Camera, Image Processing

目　　录

第一章　引　言

本文详细介绍了为第七届“飞思卡尔”杯全国大学生智能汽车竞赛而准备的智能车系统方案。该系统以 Freescale 32 位 K60 系列微控制器 MK60N512VMD100 作为系统控制处理器，采用基于摄像头的图像采样模块获取赛道图像信息，通过最优路径选择算法提取赛道黑线，识别当前所处赛道，算出小车与赛道中心的位置偏差，采用 PID 方式对舵机转向进行控制。通过光电编码器实时获取小车速度，形成速度闭环控制。调试过程中应用蓝牙模块与上位机结合进行辅助调试。小车还将通过特定算法分析出前方的路况，并根据路况的不同而为小车分配以不同的速度。

文中主要分为四个部分：

第一章引言；

第二章硬件设计；主要介绍机械结构和调整方法，赛车转向模块和驱动模块的设计、参数和有关测试，图像采样模块的摄像头工作机制以及安装选型、采样电路设计和采样策略。

第三章软件设计；主要分析系统的舵机转向策略、速度闭环控制与速度分配策略。

第四章调试与性能分析；详细叙述该系统开发过程中所用到的开发工具、软件以及各种调试、测试手段方法。

第二章　硬件设计

智能车比赛是一场实时比赛[1]，要求智能车在规定的时间内表现出最佳的状态，最佳状态要求系统具有以下特性：硬件电路的稳定性、机械结构参数的最优化和软件设计的高效性。

智能车的设计主要可以分为两个部分：硬件设计和软件设计。本章主要介绍硬件设计。硬件设计主要包括机械调整和硬件电路两大部分。图 2.1 为基于摄像头进行路径识别的智能车结构总图。

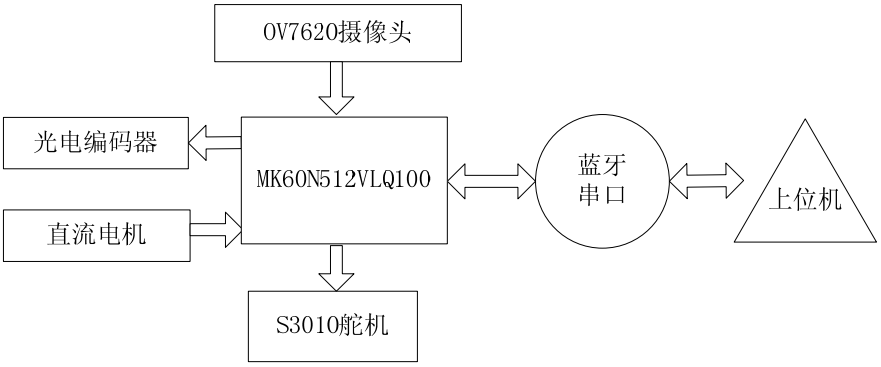


图 2.1　系统结构总图

2.1 系统硬件结构

硬件的设计不仅影响到智能车的整体性能，而且也关系到制作成本。在进行硬件的选择时，应该考虑以下主要因素：首先，硬件设计的可靠性要求较高；其次，处理图像信息速度高也要求采用合适的摄像头；此外，应该合理确定传感器的性能以满足智能车功能的要求。

因为该系统采用摄像头 OV7620 为路径检测的传感器，相比与其他两个组别在硬件设计上难度有所减低。而单片机则采用今年比赛规则中所设计的 32 位微处理器 K60 系列单片机，这样在硬件设计上难度有所增大，涉及到了电平转换。图 2.2 为智能车硬件的主要组成部分。

2.2 机械结构

机械调整的目的是为了提高车模的稳定性。低速时主要侧重于机械的对称性，硬件的对称性可以使得软件设计在高速时侧重于过弯性能和加减速的灵活性。实现方法是指在机械结构改造时，找到各种机械结构参数的最优化值，其中包括前轮定位参数、转向舵机、底盘重心位置调整等各种机械参数。

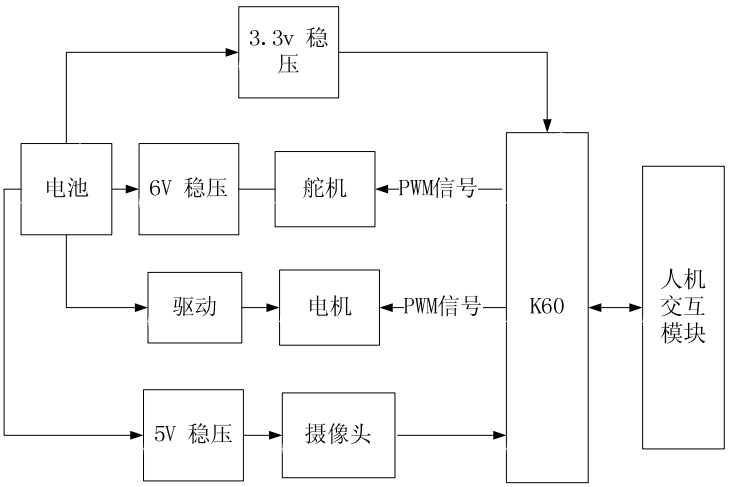


图 2.2　系统硬件结构图

大赛规定的 A 车模底盘相对较轻，如果摄像头的安装位置有问题，在稳定性和过弯性能上很难达到要求，而且摄像头的安装位置不同其所采集到的图像的视野也不同，对软件算法的影响也较大，所以对摄像头的安装专门进行了分析说明。

2.2.1 摄像头定位

摄像头的安装主要从以下几个方面进行考虑。摄像头视野的前瞻和盲区、图像的梯形失真程度、摄像头的横向覆盖范围以及对车模重心的影响。系统的理想状况是要达到：车模的重心低、摄像头的前瞻至少稳定采集到 1m、盲区不超过 10cm、图像失真小、横向跨度达到算法要求。但很难协调达到上述要求，所以在软件算法决定图像的采集范围后，尽量降低摄像头的高度，以达到减低重心的目的。摄像头整体安装效果图如图 2.3 所示。

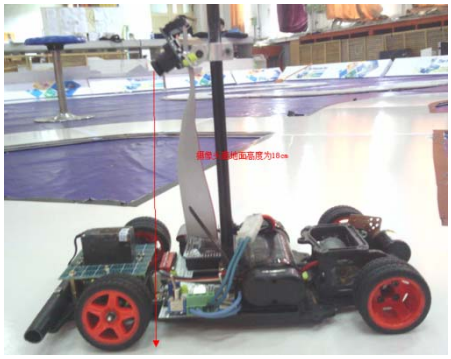


图 2.3　摄像头安装效果图

2.2.2 前轮机械

该车模系统中，为使其转向灵活，主要通过调整前轮主销的后倾角和内倾角来实现。主销后倾角是指主销在汽车的纵向平面内与竖直线所成的一个夹角。由于主销后倾角的存在，使车轮偏转后产生一个回位力矩，纠正车轮的偏转通过增减上横臂轴上的黄色垫片的数量可以改变主销后倾角大小。通常后倾角值应设定在 1 至 3 度。但为了使模型车转向灵活，该系统中将主销后倾角设为 0 度。图 2.4 主销后倾角示意图。

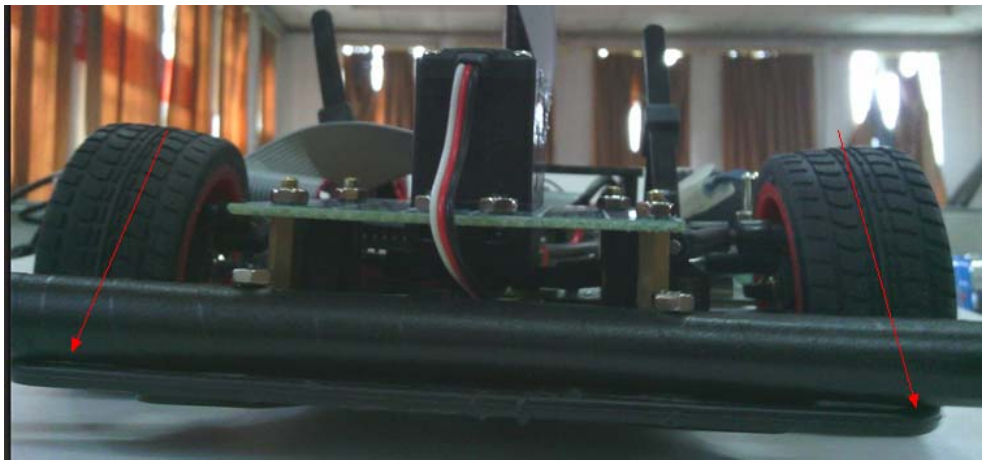


图 2.4　后倾角示意图

主销内倾角也有使车轮自动回正的作用。当前轮在外力作用下发生偏转时，车轮连同整个汽车的前部被抬起一定高度；当外力消失后，车轮在重力作用下恢复到原来中间位置。主销内倾角不宜过大，角度越大前轮自动回正的作用就越强烈，但转向时也越费力，轮胎磨损增大；反之，角度越小前轮自动回正的作用就越弱，因此这个主销内倾角都有一个范围，约 5 至 8 度之间。利用试验法确定即可。本系统将主销内倾角调节为 7 左右。这样可以减小赛道面作用于前轮的阻力矩，使舵机转向更加轻便，同时车轮自动回正的速度加快。

2.2.3 舵机的安装

组委会提供的舵机型号为 S3010，原来是安装在底板上的，但由于力臂不够大，导致舵机反应不够灵敏，鉴于舵机响应速度过慢对小车的控制不利，队员改变了舵机用于控制转向的力臂长度。通过自行设计的舵机转动机构，有效的提高了转向的相应时间，对后续小车的控制起到了一定的优化作用。

本系统中将舵机偏前放置，从而增加了舵机到连杆之间的摆杆长度，这样与以前的长度相比让前轮转过同样的角度舵机只需转过比以前更小的角度，虽然舵机本身的动作的速度没有变，但对于转向来说则比以前更快了。其次，舵

机水平放置，使舵机位于两轮的中心线上，小车前部相对于舵机左右对称，从而左右横拉杆长度相同，舵机左右转向时受力比较均匀，左右两轮转角相同，使转弯更加稳定可靠。安装效果见图 2.5。通过实践测试调整，发现舵机的响应速度提高许多，稳定性增强，为快速灵巧的转向提供了硬件上的保证。舵机力臂示意图如图 2.6 所示。

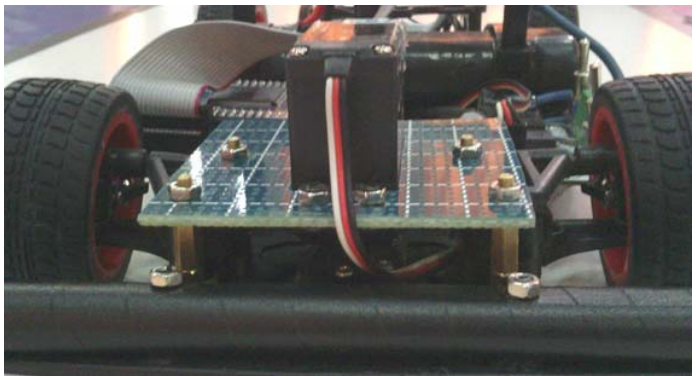


图 2.5　舵机的安装图

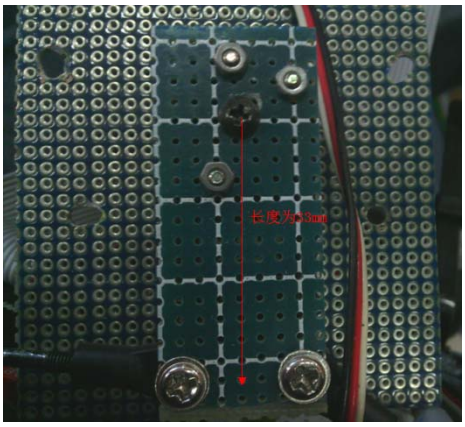


图 2.6　舵机力臂示意图

2.2.4 重心调整

智能车的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的调整以及前后位置的调整。理论上，赛车重心越低稳定性越好。因此除了摄像头可以装得稍高以外，其他各个部件的安装高度都很低。

模型车正常行驶必须满足驱动一附着条件：即所需的驱动力必须大于等于滚动阻力、坡度阻力、空气阻力之和等于前轮驱动轮的附着力。附着力与路面

附着系数以及驱动轴的轴荷有关，而驱动轴的轴荷将取决于重心的水平位置，故重心位置必须保证驱动轮能够提供足够的附着力。所以仅从这方面考虑，重心越靠近驱动轴越好。

其次，重心对通过性的影响。由于坡道的影响，在较陡侧坡行驶或高速急转弯行驶时，模型车将有发生侧向倾覆的可能，为了避免这种危险，重心应在保证最小离地间隙的前提下尽量降低。

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。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；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。因此，确定合适的车体重心，让车模更加适应比赛赛道是非常关键的。经过实际调试，将摄像头安装在车体中后部，使车体重心位于中间偏后一些的地方，可以使得赛车转向灵敏度、稳定性以及加减速性能都达到较佳水平。

为了降低重心设计替换了后轮支撑套，采用了下高上低的支撑套，于此同时，设计还在前轮支撑位置加上垫片，从而降低小车前部重心，通过以上两种方式，降低了整车的重心，进而提高小车运行过程中的稳定性，小车底盘如图 2.7 所示。

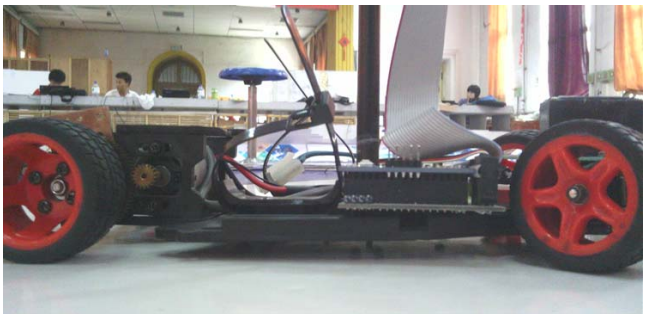


图 2.7　小车底盘示意图

2.3 电路设计

智能车硬件电路设计共包括六大模块：控制处理芯片、舵机驱动模块，电机驱动模块，图像采集模块，速度采集模块，人机交互模块和电源模块。其中 K60 单片机是系统的核心部分。它负责接收道路图像数据、行驶速度等反馈信息，并对这些信息进行恰当的处理，变成控制量来对舵机与驱动电机进行控制。舵机模块和驱动模块分别用于实现车模的转向和驱动。图像采集模块由摄像头和电平转换电路组成。其功能是获取前方路径的图像数据，以供 K60 作进一步分

析处理。速度采集模块由光电编码器提供，通过检测脉冲累积数来求得车模的速度值。电源模块主要用于为系统其他各个模块提供所需要的电源。人机交互模块主要是进行人机交互，包括串口通信，拨码开关等组成，功能是辅助调试或者在现场对参数进行微调[2]。

2.3.1 单片机稳压电路

因为单片机供电电压为 3.3V，考虑到单片机系统的稳定性，所以设计采用了串联型稳压芯片。该芯片具备功耗小，波纹小，电流大等特点。单片机稳压电路原理图如图 2.8 所示。

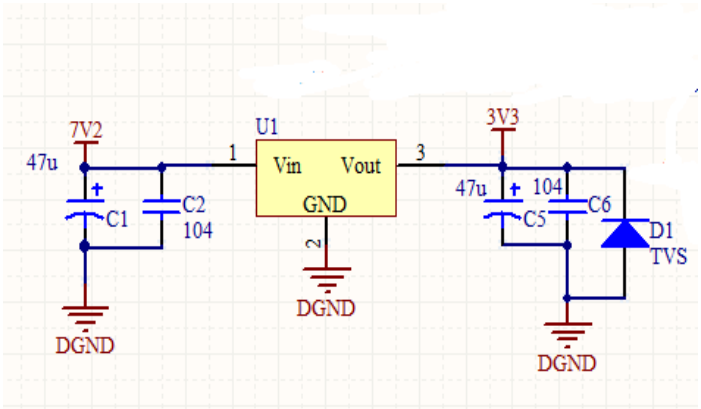


图 2.8　单片机稳压电路

2.3.2 摄像头稳压电路

摄像头供电电压为 5V，由于摄像头对波纹系数要求较高，因此，设计采用了波纹系数较小的线性稳压芯片，电路图 2.9 所示。

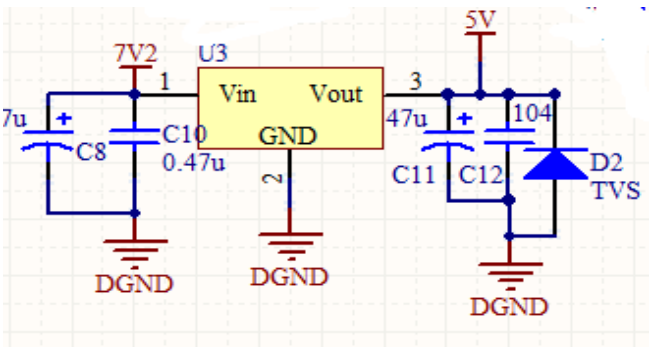


图 2.9　摄像头稳压电路

2.3.3 舵机稳压电路

系统设计采用的是模拟舵机 S3010，舵机供电电压为 6V，通过测试发现，舵机的供电电压在 6.5V 的时候，舵机反应速度有所提高，因此 在设计过程中采用可调线性稳压芯片，该芯片具有电流大，波纹小，可调节等特点，电路图 2.10 所示。

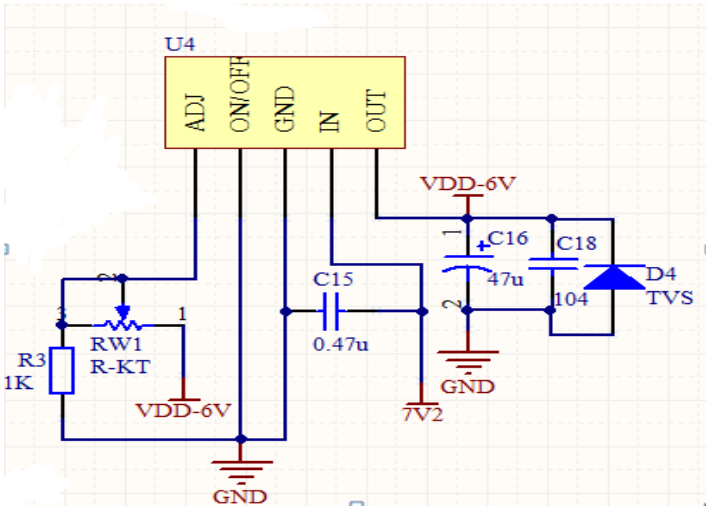


图 2.10　舵机稳压电路

2.4 电机驱动设计

图 2.11 为电机驱动电路[3]。

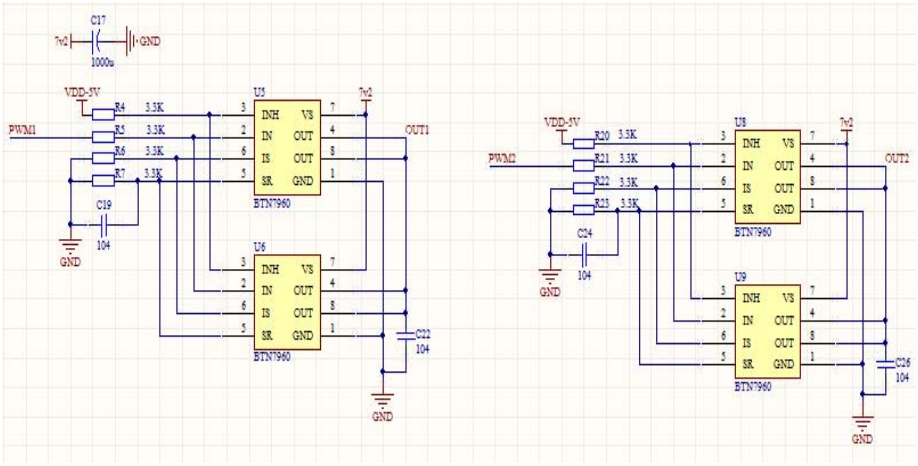


图 2.11　电机驱动电路

2.5 其他辅助电路

2.5.1 摄像头接口电路

K60 是 3.3V 供电，输入输出电压都为 3.3V，然而摄像头 OV7620 端口输出电压为 5V，因此为了保护单片机，防止单片机长时间过压，在 OV7620 输出引脚都做了保护，从而实现系统的稳定运行，电路图 2.12 示。

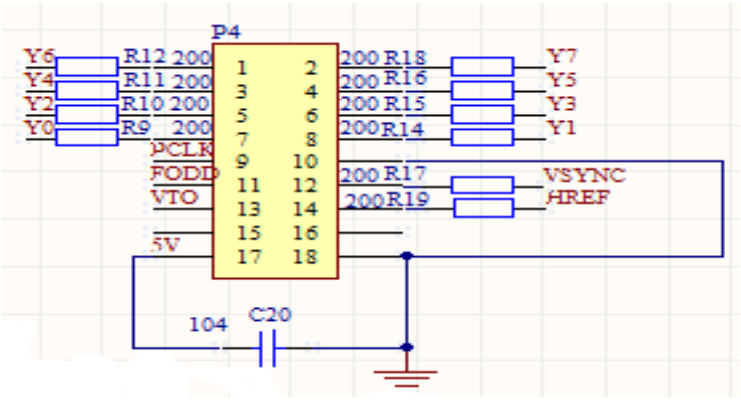


图 2.12　摄像头接口电路

2.5.2 驱动隔离电路

因为设计所采用的驱动芯片是 BTN7960，为了防止驱动芯片出现回流而把单片机烧毁，在驱动的 PWM控制端，以及舵机的控制端都采用了 74LS244 芯片，进行隔离，从而保证单片机安全可靠工作，电路图 2.13 所示。

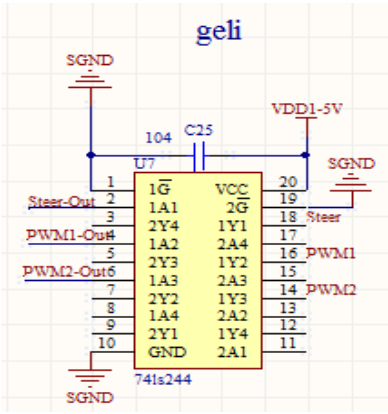


图 2.13　舵机稳压电路

总体电路的 PCB 板设计图见附录 A。

第三章　软件设计

3.1 系统软件结构

3.1.1 系统软件结构

当系统的硬件部分都设计定型后，建立在其上的软件系统便直接决定了整个系统的使用性能和智能化程度。软件系统的设计已经有很多成熟的方法和技术，今年的赛道为识别双线，智能车软件设计结构图如图 3.1 所示。

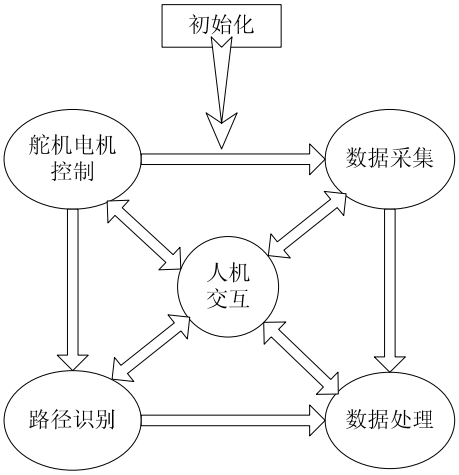


图 3.1　系统软件结构图

3.1.2 主程序程图

智能车要想稳定、高速的跑完整个赛道，软件设计过程中的算法显得十分重要。

这就是求一个最优解问题：

1.在最短的时间跑完整个赛道；

2.利用最优的搜索算法完成对赛道的识别；

3.保证智能车在跑道上的稳定性和对噪点反映的快速性等。

智能车必须有自主驾驶系统特性，在软件上，系统包含系统初始化、摄像头的动态配置、图像采集、赛道中心线的提取及搜索最优路径的规划等，图 3.2 为主程序流程图。程序代码见附录 B。

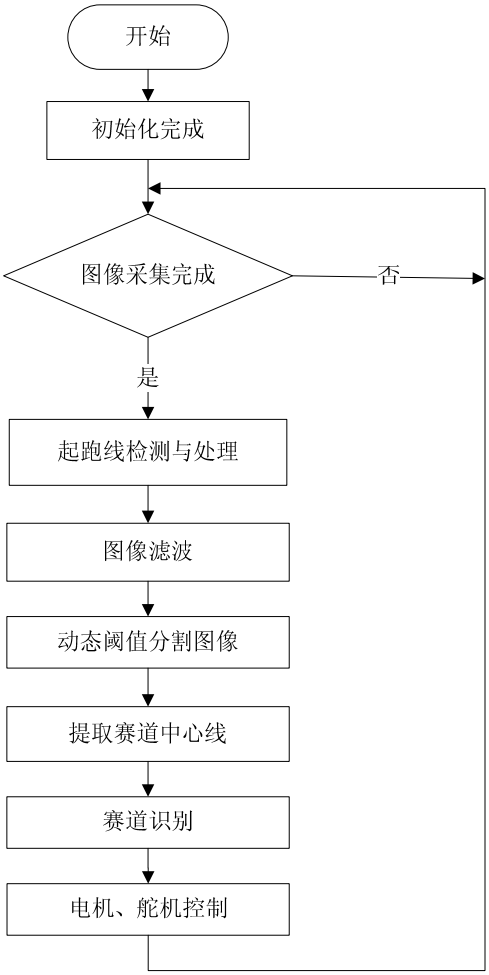


图 3.2　主程序流程图

3.2 最佳路线识别算法

对路径的确定采用 RANSAC（Random Sample Consensus，随机抽样一致性算法）是一种鲁棒性很强的模板估计方法，其实质是一个反复测试、不断迭代的过程[4]。在图像拼接中，RANSAC 算法的模板是 3*3 变换矩阵(选择 8 参数投影变换，至少需要四个特征点对来生成)，而目标函数是计算经变换矩阵变换后的特征点与其匹配特征点之间的误差。算法中，迭代的时间取决于抽样点的数量和线内点所占的比例。可以用公式（1）在理论[5]上进行估算：

$$N=\frac{\log(1-p)}{\log[1-(1-\varepsilon)^4]}$$

公式 1

其中 N 为采样的次数，p 为取样 N 次得到正确样本的概率，一般要求必须

大于 95%，ε 为外点的比例。随着 ε 的增加需要随机采样次数的变化情况如表 3.1 所示。

表 3.1　随机采样次数与集合中外点比例关系

集合中外点的比例 ε（%）						
	20	30	40	50	60	70
随机采样数 N（次）	9	17	34	72	178	567

从表 3.1 数据可以看出，当线外点的比例为 70%时，迭代次数高达 567 次。实际中由于图像噪声等原因线内点的比例可能会非常低，这样就大大增加了迭代次数，影响了算法整体效率。为此本研究提出用聚类的方法预筛选数据，以提高线“内点”的比例，进而减少迭代次数，提高算法效率。

预筛选的基本思路是根据拼接图像待匹配点之间的连线斜率应相同或相近的特性，将两幅待匹配图像置于同一坐标系下，然后以某一斜率值为中心，设定一邻域阀值，若在较小邻域阀值内包含最多的斜率相近的特征点对,则这些点就是最精确匹配的相关点对。据此可以排除大部分误差较大的错误匹配。上述较小邻域根据经验值一般可取 0.01~0.05。聚类法预筛选的数学原理如图 3.3 所示。

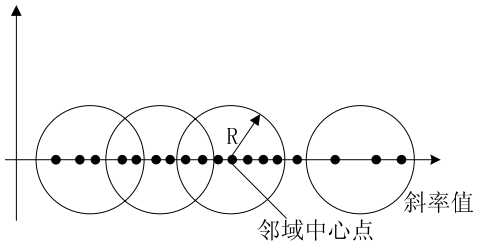


图3.3　聚类预筛选的数学原理图

3.2.1 算法的主要步骤

SIFT 算法提取特征点后，以一幅图像的特征点为参考基由 BBF(best-bin-first)搜索算法在另一幅图像中查找每个特征点相对应的最近邻与次近邻点，通过判定欧式距离的比值得到初匹配，这时每个匹配的特征点对都包含有位置，尺度，方向及 128 维的特征描述符信息，初匹配中含有大量的“外点”。将二幅待匹配图像变换到同一坐标系中并重叠在一起，然后把两幅图像中的匹配特征点用直线连接，则这些直线的斜率应相同或相近（当镜头畸变被忽略时）。运用简单聚类的方法，以某斜率值为中心找到一个包含最多斜率值的小邻域：

$$M = \max\{\sum_{i=1}^{n}\sum_{j=1}^{n} f(|s_i - s_j|)\} \quad \text{公式 2}$$

$$f(s) = \begin{cases} 1 & s \le t \\ 0 & s > t \end{cases} \quad \text{公式 3}$$

其中，S_i，S_j 分别代表第 i 和第 j 条直线的斜率，t 为设定邻域的阀值。

具体部分算法分为以下 4 个部分：

（1） 计算成对待匹配特征点对数据集中所有直线的斜率值。

（2） 根据经验值，设定邻域阀值为 t=0.01，依据公式(2)、(3)通过循环计算得到 M 的值，并同时记下此时 i 的值，然后依此计算出此时领域内的点对集，作为预选出的待匹配点。删除在邻域外的点对集。

（3） 使用 RANSAC 算法对预选出的点进行二次筛选，得到的最大结果集即为匹配的特征点对。

（4） 使用匹配的特征点计算变换矩阵，并采用 Levenberg-Marquardt(L-M)算法进行优化。

3.2.2 最佳路径的确定

一条理想的路径应是在两幅图像重叠区域的差值图像上颜色、结构强度差值最小的一条线，因此定义路径求解准则如下：

$$E(x,y) = E_{color}(x,y)^2 + E_{gemetry}(x,y) \quad \text{公式 4}$$

其中 $E_{color}(x,y)^2$ 表示重叠像素点的颜色值之差，$E_{gemetry}(x,y)$ 表示重叠像素点的结构差值。$E_{gemetry}(x,y)$ 是通过修改梯度计算 Sobel 算子实现的。利用 Sobel 算子进行梯度计算时，计算在 x 方向和 y 方向的梯度分别采用模板：

$$S_x=\begin{pmatrix} -2 & 0 & 2 \\ -1 & 0 & 1 \\ -2 & 0 & 2 \end{pmatrix} \text{和} S_y=\begin{pmatrix} -2 & -1 & 2 \\ 0 & 0 & 0 \\ -2 & 1 & 2 \end{pmatrix}。$$

假定两幅原始图像为 f1 和 f2，则 $E_{\text{gemetry}}(x,y) = \text{Diff}\left(\text{f1}(x,y), \text{f2}(x,y)\right)$，Diff 的求解是通过计算两幅图像 f1 和 f2 在 x 和 y 方向的梯度之差的积得到的。

根据上述准则，对差值图像运用动态规划的思想从重叠区域的第一行出发，建立以该行上每一个像素为起点的路径，最后从这些路径中寻找一个最佳的路径[4]。

具体步骤如下：

1. 初始化。第一行各列像素点为每一条路径的起点，其强度值为各个点的准则值，按既定的扩展方法向下一行扩展；

2. 扩展路径。已经计算过路径强度的一行向下扩展，直到最后一行为止。扩展的方法是将每一条路径的当前点与该点紧邻的下一行中的三个像素准则值进行比较，取对应的强度值最小的像素点作为该路径的扩展方向，更新此路径的强度值，并将路径的当前点更新为得到最小强度值所在的下一行中的紧邻像素值所在的列；

3. 选择最佳路径。从所有的路径中选择强度值最小的一条作为最佳路径。

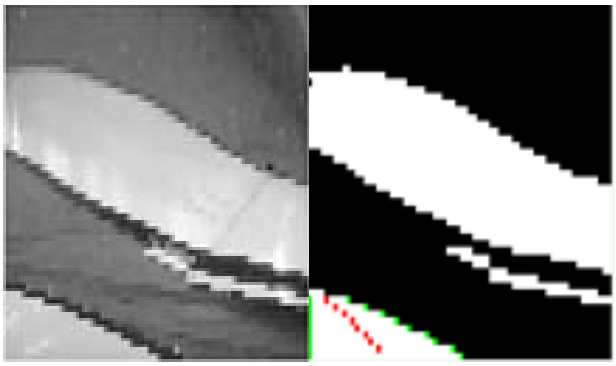


图 3.4　通过最佳路径搜索前后比较图

图 3.4 为最佳路径搜索算法示意图，上述算法有其结构合理性，一定程度上避开了场景中运动物体，减少了鬼影出现的概率。但仔细分析会发现它只是一个局部最优解，还有可能出现鬼影现象。因为可能某条路径的大部分误差很小，仅有个别几个点的误差较大，但整条路径的误差最小则这条路径就被错误的认为是最佳路径。

3.2.3 基于特征点的最佳路径多分辨率融合

为了使最佳路径完全避开两幅图像上的差异对象和运动物体，必须对路径的扩展方向加以限制。经过匹配，过滤之后的匹配特征点对，肯定不会出现在两幅图像中的差异对象上，结合特征点的性质，可以利用特征点对限制路径的扩展方向，并对特征点处的准则值适当加权（权值应大于 0 小于 1），这样可加大包含特征点的路径被选择的概率。同时，为了使路径尽可能多的包含特征点，可以适当拓宽路径的扩展范围，采用将下一行中与该点紧邻的 5 个像素点作为这条拼接缝的最佳扩展方向。求出路径后，因为两边图像可能存在曝光差异，为此对合成图像采用多分辨率样条算法[6]二次融合，以实现平滑无鬼影拼接。

3.3 图像去噪

图像去噪是为了去除干扰点更加方便的提取边缘线和中心线并且能提高提取边缘线精确度，对采集回来的图像的灰度值分析后，发现 0-10 245-255 这两段极端区间的灰度值一般是采集不到的，如果发现这些灰度值存在就认为该点为噪点，如图 3.5 所标注出来的点就是噪点，需要进行处理。处理的算法是扫描整幅图像，判断该点灰度值如果是极端灰度值，该点灰度值就给值为四周 8 个点灰度值的均值。处理后图如图 3.6 所示。

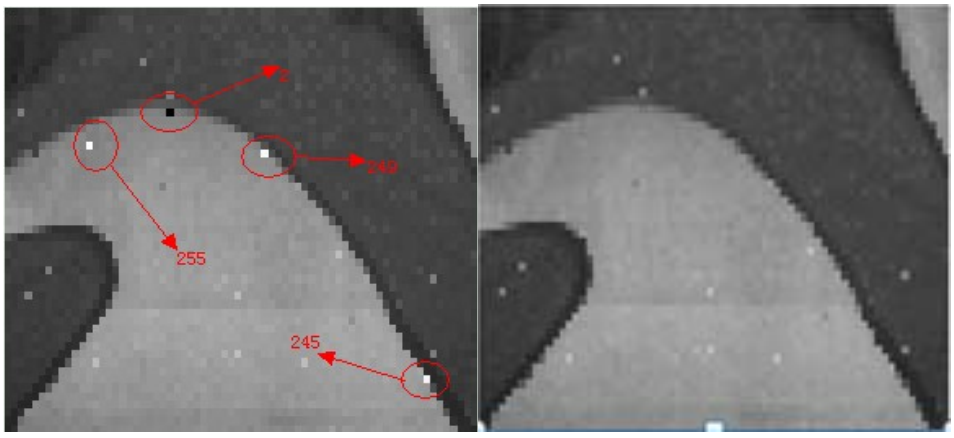


图 3.5　噪点处理前图　　图 3.6　噪点处理后图

图像去噪流程图如图 3.7 所示。

3.4 赛道识别

进行识别和处理的赛道类型有：直道、小 S 道、普通弯道、急弯、直入弯。通过赛道识别，为了能够更好的控制舵机和电机，使小车在快速行驶过程中仍然能够保持良好的路线。初期的赛道识别做得不是很完善，存在识别不出来以及识别出错的情况，导致在快速情况下直道上出现连续左右摇摆晃动的情况，在弯道会冲出赛道或者打滑甩尾，严重影响了整体车速。所以对赛道识别又做了进一步的研究。

赛道识别提取的主要参数有：中心线偏差和（即相邻两行间中线位置做差然后求和）：中心线偏差大小在加上中心线的长度这个条件就可以知道赛道的弧度大小，正负恰好能反映左右趋势。

中心线的方差：中心线的方差能反映其波动的情况，如果在偏差和很小的情况下，尚且有较明显的波动（即方差较大），因此就可以知道是小 S 类型的赛

道。

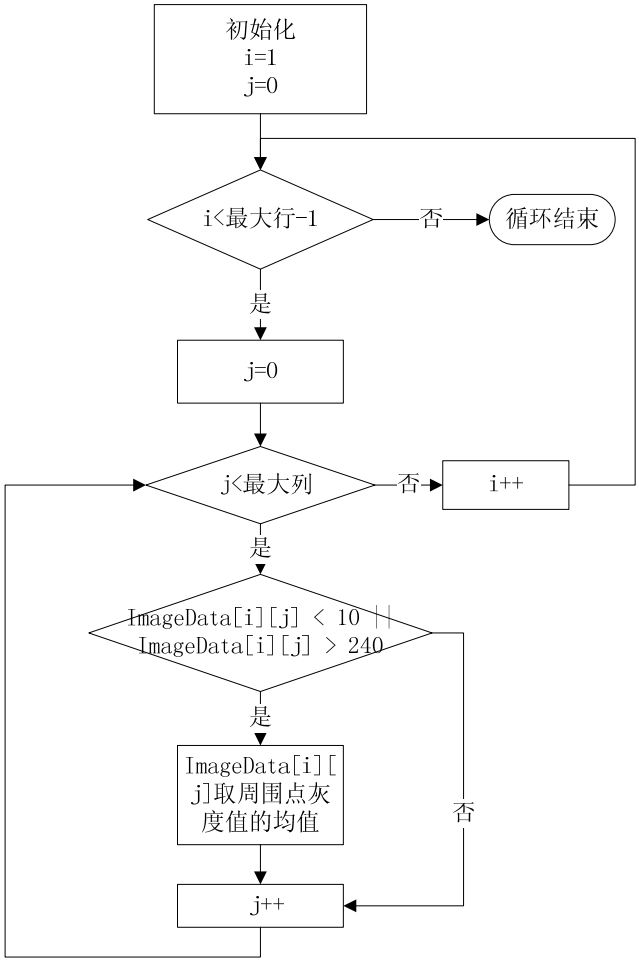


图 3.7　图像去噪流程图

从图 3.8 直道中心线数据提取前后的比较可可以看出直道偏差和小，方差小。

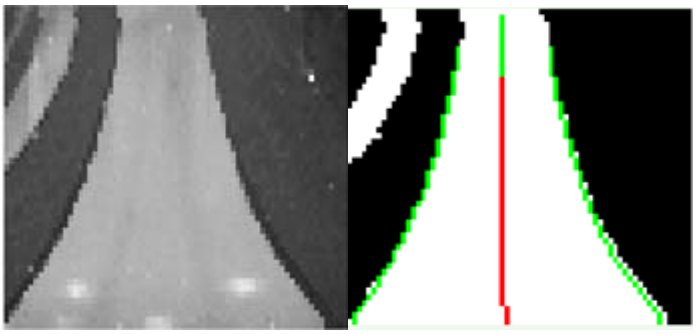


图 3.8　直道中心线提取前后比较图

从图 3.9 小 S 道 9 中心线数据提取前后的比较 9 可以看出小 S 道偏差和小，方差较大。

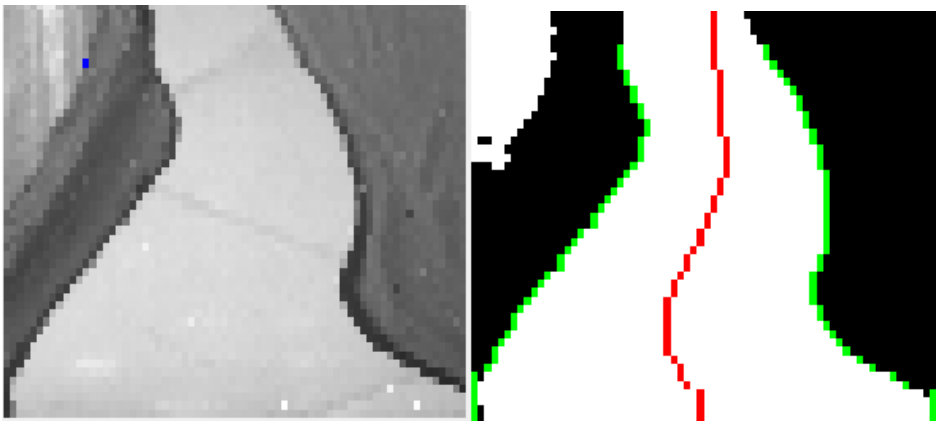


图 3.9　小 S 道中心线提取前后比较图

直入弯：直入弯比较难准确判断，也比较难解决误判的情况，在中心线提取稳定的情况下，对中心线进行分段求偏差和，根据每段的偏差情况可以判断出直入弯，但是这种方法误判率较高，所以又加入了限制条件，当前实际速度以及历史最近的几次赛道类型是否是直道就可以解决误判的情况。

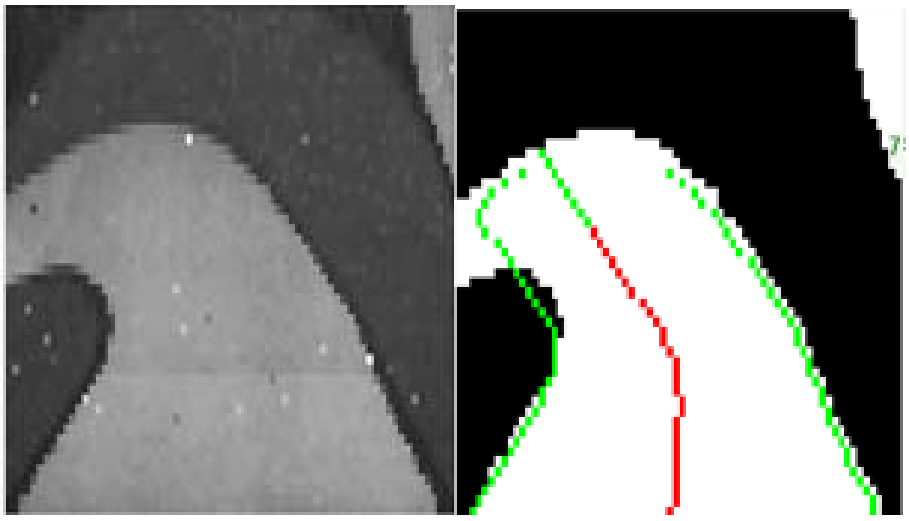


图 3.10　直入弯中心线提取前后比较图

急弯：中心线的有效长度很小，并且平均偏差（平均偏差=偏差和/中心线的长度）较大。

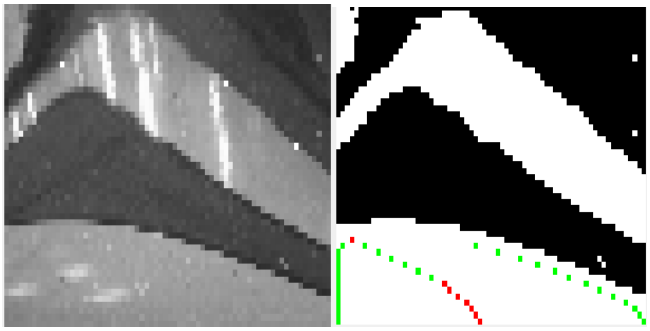


图 3.11　急弯中心线提取前后比较图

普通弯道：中心线长度较长，偏差和大，并且方差大。

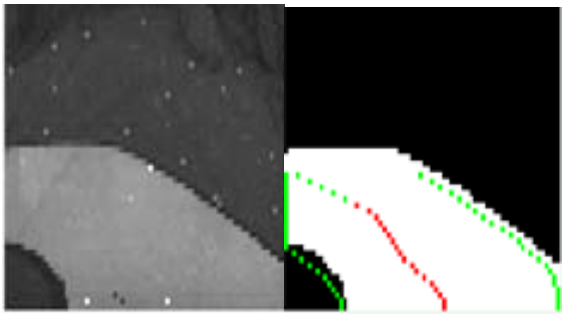


图 3.12　普通弯道中心线提取前后比较图

十字交叉道：前期对十字交叉道不做特殊的处理，小车在不跑偏的情况下都能顺利通过，但是这种不处理的处理办法还是会让小车比较频繁的在十字交叉道冲出赛道或者在十字交叉道晃动，影响了小车的整体稳定性。所以后期对十字交叉道又进行了处理。

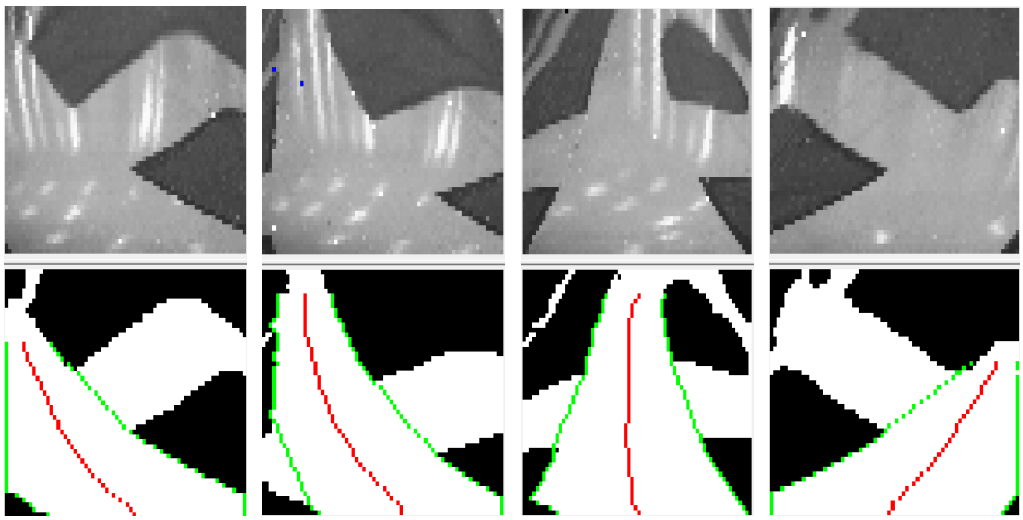


图 3.12　十字交叉线中心线提取前后比较图

十字交叉道的识别是比较容易的，但是识别出来后舵机的偏差值却是比较难确定的，所以就对识别出来的十字交叉道进行补线处理，补线处理后做一般的控制。

由于十字交叉道干扰边缘非常多，所以对搜索十字交叉道两边缘的搜索要十分小心才能搜索到正确的左右边缘。首先，要确定的是十字交叉道是左倾还是右倾还是正的十字道，

通过图像左、中、右黑线的截止行就可以判断，左边黑线为三者最大的是左倾，右边黑线截止最大的是右倾，中间截止最大的是正十字。对于左倾的十字道，主要对右边缘进行拟合，拟合首先要找到右边缘的拐点，所谓的拐点就是原来一直往左方向的线突然往右拐了，并且连续向右方向拐了好几个点。确定了拐点好了以后接下来就要确定右边缘的另外一个拟合点了，对于左倾的十字来说，另外一个拐点必然在第一个拐点的左边，所以从第一个拐点再往前搜索，搜索到的第一个比第一个拐点小的就是另外一个拟合点了，找到的两个拟合点间拟合连线即可。对于正十字和右偏的十字处理思路基本一致。

3.5 电机控制

控制算法：PID 控制。

PID 控制是工程实际中应用最为广泛的调节器控制规律，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[3]。

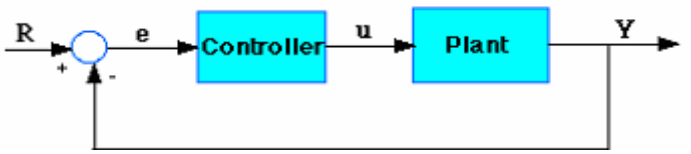


图3.13　PID控制原理图

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。

$$u = k_p e + k_i \int e dt + k_d \frac{de}{dt}$$

其中，k_p、k_i、k_d 分别称为比例系数、积分系数、微分系数。u 接着被送到了执行机构，这样就获得了新的输出信号 u,这个新的输出信号被再次送到感应器以发现新的误差信号，这个过程就这样周而复始地进行。

PID 各个参数作用基本介绍：

增大微分项系数可以加快动态系统响应，但容易引起震荡。一般增大比例系数能够减小上升时间，但不能消除稳态误差。增大积分系数能够消除稳态误差，但会使瞬时响应变差。增大微分系数能够增强系统的稳定特性，减小超调，并且改善瞬时响应。

对连续系统中的积分项和微分项在计算机上的实现，是将上式转换成差分方程，由此实现数字 PID 调节器。

采用位置式 PID 控制算法，用矩形数值积分代替上式中的积分项，对导数项用后向差分逼近，得到数字 PID 控制器的基本算式（位置算式）：

$$u_n = k_p(e_n + \frac{1}{T_i}\sum_{k=1}^{n} e_k T + T_d \frac{e_n - e_{n-1}}{T})$$

其中 T 是采样时间，k_p、T_i、T_d 为三个待调参数，在实际代码实现算法时，处理成以下形式：

$$\text{PreU} = K_p * \text{error} + K_i * \text{Integral} + K_d * \text{derror}$$

增量式 PID 控制算法

对位置式加以变换，可以得到 PID 算法的另一种实现形式（增量式）：

$$\Delta u_n = u_n - u_{n-1} = k_p[(e_n - e_{n-1}) + \frac{1}{T_i}e_n + \frac{T_d}{T}(e_n - 2e_{n-1} + e_{n-2})]$$

设计在实际代码实现时，处理成 PreU +=（K_p * d_error + K_i*error + Kd*dd_error) 的形式。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。根据不同赛道类型给定不同的速度，如果速度变化过大会引起小车的不稳定，所以加入了平滑速度的控制处理。

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，因此采用了这种方法，同时，也先后实验了几种动态改变 PID 参数的控制方法。

3.6 舵机控制

控制算法：PD 控制。

舵机偏角输入量的确定：

设计尝试过两种办法：

第一种是选取中心线的其中一行的偏差作为输入量。

第二种是求多行中心线的平均偏差作为输入量。

测试发现求多行中心线偏差的方法更加良好。在对参考行的选取上队员做了大量的测试，是根据不同的赛道类型给定参考行。在多行求平均偏差的过程中使用了加权求平均的方法，根据提取每行的中心线质量情况给定一个权重，再根据赛道类型给中线远端和近端给一个权重，通过两个权重加权平均而得的偏差能够实现在不同速度下对舵机准确圆滑的控制。

Error = (k1[i]*k2[i]*MidError[i]+ k1[i+1]*k2[i+1]*MidError[i+1]+……+ k1[i+n]*k2[i+n]*MidError[i+n]) / (k1[i]*k2[i]+ k1[i+1]*k2[i+1]+ ……+ k1[i+n]*k2[i+n])

第四章　调试与性能分析

4.1 开发平台

开发环境：Windows7 32 位 + VC6.0 。

1. 使用 MFC 类库做界面开发。

2. 串口通信技术，本软件使用微软提供的 MSCOMM32.OCX 控件进行串口操作。

3. 多线程技术，本软件使用三条线程进行处理，一条线程负责接收数据，一条负责数据处理操作最后一条线程进行界面更新操作。使得在高速接收串口数据的情况下，软件运行顺畅。

4. 缓冲区机制，本软件使用三级缓冲机制，每个缓冲区采用双向循环链表实现。缓冲机制有效减少数据丢失的情况，双向链表可以很灵活的对这些数据进行操作处理。

5. DLL 动态链接库技术，实现算法库的动态添加，动态算法库使用 C 语言编写，编译生成 DLL 文件，加载到软件形成算法插件，执行算法插件后，就可以看到处理后的效果，可以实现静态的仿真。

6. VC 对位图的基本操作技术。

4.2 调试分析

上位机实现对数据的采集和处理，上位机示波器检测界面如图 4.1 所示。

图 4.1　上位机示波器检测界面

图像处理界面如图 4.2 所示。

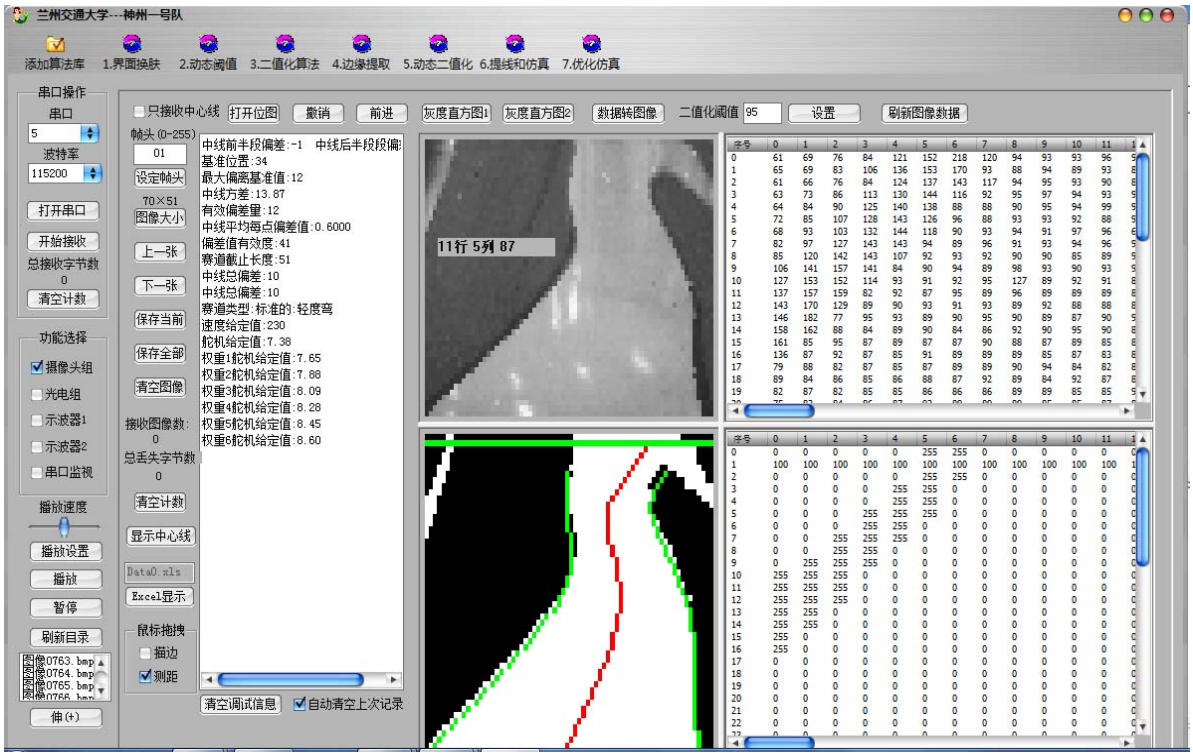


图 4.2　上位机对采集的图像处理界面

4.3 上位机图像仿真

下位机通过串口把采集好的图像实时发送到上位机，上位机接收到图像数据后根据协议判断是否是一副完整的图像，如果是完整的图像则自动保存成 BMP 格式的位图。上位机通过算法插件可以进行全程仿真。上位机采集的赛道信息如图 4.3 所示，图 4.4 是上位机采集的速度受到干扰时 PID 调节图。

图 4.3　上位机采集的赛道信息

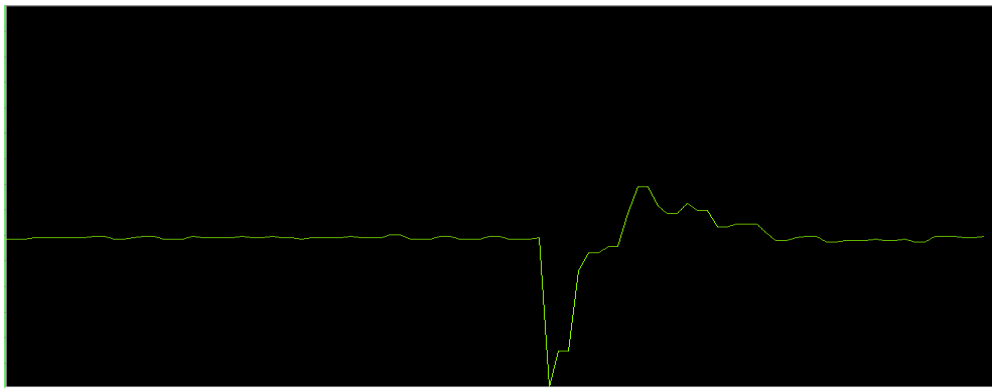


图 4.4　速度受到干扰时 PID 调节图

第五章　赛车主要技术参数

根据组委会的相关规定和设计方案的要求具体最终确定模型车的主要技术参数如表 5.1 所示。

表 5.1 主要技术参数

赛车基本参数	长	28.5cm
	宽	16.8cm
	高	25.2cm
车重	0.75kg	
功耗	空载	8.8W
	带载	大于15W
电容总容量		1501.1uF
传感器	CMOS摄像头1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/远）	5/130cm
	精度(近/远)	3/12.5mm
	频率	60Hz

第六章　总　结

经过半年多的工作，对硬件的选型和软件算法的选取及最终的实现过程在前面的几章中都做了全面地介绍。在此对设计过程做一个总结。

总体上，系统的开发工作分为三期：

第一个阶段：是对主要开发工作最全面的熟悉和准备；

第二个阶段：根据需求分析和系统实现的过程做深入的分析；

第三个阶段：在前两个阶段工作的基础上选择硬件和软件算法，重点比较了各种算法的实际效果，确定了最终方案。

回顾整个过程，基本保证了开发的有序与可控性，完成了设计任务。

本设计基于利用图像采集模块得到的路况信息使智能车实现循迹跟踪，采用 PWM 技术控制电机的转速和转向，装备 CCD 摄像头传感器后车速有了更大的提高空间，直线与普通弯道的识别均可以达到预期的效果。在图像处理中主要包括以下几个方面的内容：图像去噪及动态二值化、边缘线和中线线提取、赛道识别。在控制算法中，主要用到了以下三方面的内容：舵机转向控制采用 PD 算法， PID 控制算法主要用来对智能车的快速加速、减速和速度的平稳进行控制。

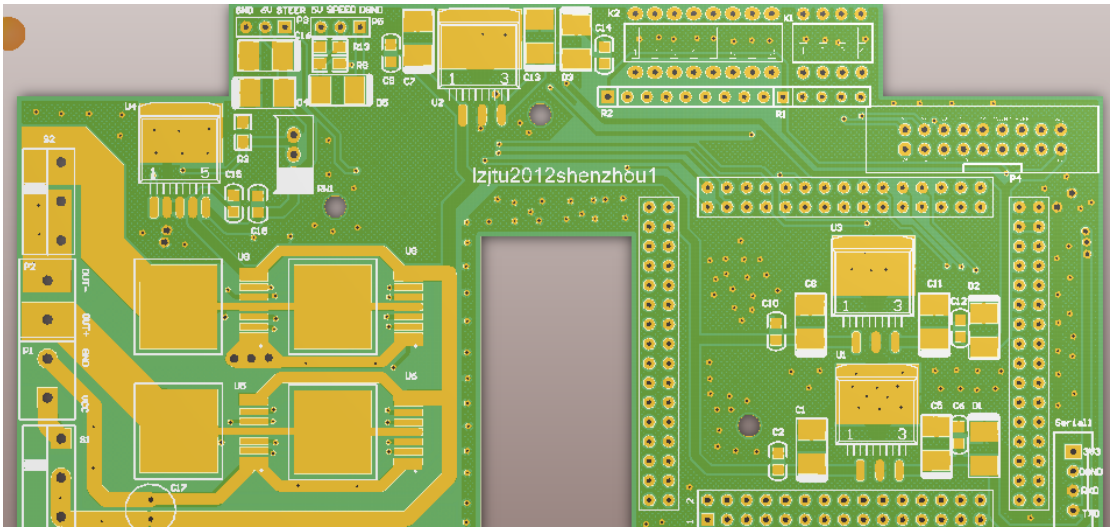
第七章　致　谢

经过了一年多的准备，我们成功的冲进了全国总决赛。在此我们全队感谢一直以来关注我们小车并且提供便利的自动化系主任王思明教授；感谢姜香菊老师；感谢路小娟老师；感谢汉鹏武老师；研究生季聪学长；以及为西部赛而一起奋斗的智能车战友们，感谢你们大量的建议、帮助和支持。也对一直默默关心、支持兰州交通大学智能车的各位领导、老师、同学表示感谢，感谢你们对车队的关注与支持，我们的每一点进步，都源自你们的帮助。希望你们继续关注兰州交大智能汽车，继续关注兰州交通大学神舟一号智能车队，谢谢。

参考文献

[1]卓晴,黄开胜,邵贝贝.学做智能车——挑战“飞思卡尔”杯[M].北京:北京航空航天大学出版社,2007.
[2]王宜怀,刘晓升.嵌入式应用技术基础教程[M].北京:清华大学出版社.2005.
[3]陈伯时.电力拖动自动控制系统——运动控制系统[M].北京:机械工业出版社,2005.
[4]党建武,宗岩,王阳萍.基于SIFT特征检测的图像拼接及优化算法研究[J].中国图象图形学报.2012,29(1):329-332.
[5]张媛,高冠东,贾克斌 运用特征点匹配的柱面全景图像快速拼接算法[J].中国图象图形学报.2009,3(3):247-250.
[6]范鹏轩.虚拟现实系统关键技术和算法的研究及应用[D].重庆:重庆交通大学,2010.

附录 A　总体电路的 PCB 板设计图



附录 B　程序代码

```
#include "MK60N512MD100.h"
#include "delay.h"
#include "serial.h"
#include "pwm.h"
#include "pit.h"
#include "pid.h"
#include "function.h"
#include "define.h"
#include "image.h"
#include "wdog.h"
#include "lcd.h"
#include "DataProcess.h"
#include "Control.h"
#include "boma.h"
#include "adc.h"
unsigned char SendByte=0;
unsigned char LineCount;//列计数值
unsigned char RowCount; //行计数值
unsigned char LineCount1;//列计数值
unsigned char RowCount1; //行计数值
unsigned char ImageFlag=0;
int16_t SteerCounter=0;
int16_t MotorRunFlag=0;
int16_t MotorPidVal=0;      //电机 PID 计算后的值
uint16_t    MotorSpeed=0;
uint8_t SerialBuf[24]={0};
uint8_t SerialAdd=0;
uint8_t SerialFlag=0;
unsigned char    SendStartLineFlag=0;
unsigned char SendADFlag=0;
int StraightSpeed = 200;
int    SmallSSpeed    = 200;
int    BigSSpeed    = 130;
int    BendSpeed = 130;
int    CommonSpeed = 130;
int    StraightToBendSpeed = 130;
int SpecialSpeed = 120;
int CanSpeedUp = 0;       //长直道提速使能
```

```
int CanFullSpeed = 0;//可全速使能
int CanCheck = 0; //使能检测起跑线
float StraightK = 0.3;
float SmallSK = 0.5;
float BigSK = 1.5;
float BendK = 1.6;
float CommonK = 1.0;
unsigned char StraightFS=35;//直道前瞻
unsigned char SmallSFS=35;//小 s 前瞻
unsigned char BigFS=22;//大 S 前瞻
unsigned char BendFS=20;//急弯前瞻
unsigned char CommonFS=25;//普通前瞻
unsigned char FarWeight = 0;
//  图像原始数据
unsigned char ImageData[RowMax][ColumnMax];     //图像数据数组
unsigned char StartLineBuf[RowMax][ColumnMax];  //起跑线数据数组
unsigned char ThresholdData[RowMax][ColumnMax];   //二值化数据数组
signed char LeftBlack[RowMax]; //左边缘黑线数组
signed char RightBlack[RowMax];//右边缘黑线数组
signed char BlackLineData[RowMax];//提取黑线值数据
signed char RoadType = -1;
signed char LastRoadType = 0;
unsigned char IsCrossing=0;
double CrossingError = 0.0;
int ThresholdValue=0;                                   //二值化阈值
int ThresholdValue1 = 150;
int ThresholdAvg = 150;
int ThresholdArray[5]={0};
unsigned char bCalThresholdAvg = 0;
int DirectData;                                         //舵机值
unsigned char BeginCheck=0;//开始检测起跑线标志
unsigned int TimeCount = 0;//检测起跑线的时间延迟
unsigned char SlectedCheckStartLine=0;//拨码开关检测起跑线标志
unsigned char IsOutRoad=0;
volatile unsigned char IsStartLine = 0;
unsigned short ADVal[5]={0};
int FullSpeedTime=0;    //起跑满占空时间
int CheckTime=0;//检测时间
int FullSpeedCount = 0;
int FullSpeedFlag = 1;
```

```
int Boma1=0;
int Boma2=0;
int Boma3=0;
unsigned char ImageOK = 0;
float SteerErrorArray[5] = {0.0,0.0,0.0,0.0,0.0};
int SteerTimeCount = 0;
unsigned char SteerErrorOk = 0;
int aaa=100;
int main (void)
{
    __disable_irq();
    SystemCoreClockUpdate();   /* Get Core Clock
      Frequency */
    //IO 口初始化
    GPIO_Init();
//  //串口初始化
    Serial_Init (UART3,90000000,115200);
//  UART3->C2|=UART_C2_RIE_MASK;      //使能接收中断
//  NVIC_EnableIRQ(UART3_RX_TX_IRQn);//打开串口中断
//  Ser_PutChar(UART3,0x01);   //串口验证看门口
    //电机 PWM 初始化
    FTM0_Init();
    FTM1_Init();//舵机初始化
    Counter_Init();
    //打开场中断
    NVIC_EnableIRQ(PORTA_IRQn);
    NVIC_DisableIRQ(PORTB_IRQn);
    NVIC_SetPriority(PORTA_IRQn,1);
    NVIC_SetPriority(PORTB_IRQn,2);
    NVIC_SetPriority(PIT0_IRQn,3);
    NVIC_EnableIRQ(PIT0_IRQn);//开启定时器
    PIT_Init(PIT0_IRQn,450000);//定时器时间
    SteerUpdatePWM(0,DirectMiddle);//
    Boma1Switch();   // 控制速度档拨码开关
    Boma2Switch();   // 控制满占空拨码开关
    __enable_irq();
    Delay_MS(2000);
    while(1)
    {
        ImageProcess();//图像处理滤波+动态阈值二值化
```

```
            StartLineProcess();
            GetImageParam();//提取图像特征参数
            MidLineProcess();//提取中心线并处理
            RTRecognition();   //赛道识别
            CheckStartLine(); //检测起跑线
            SpeedCtrol();        //速度控制
            DirectionCtrol();     //舵机控制
            LedShow();             //LED 灯显示状态
        } // end while(1)
}
void PIT0_IRQHandler(void)//
{
    if((PIT->CHANNEL[0].TFLG&PIT_TFLG_TIF_MASK)!=0)
    {
        PIT->CHANNEL[0].TFLG |= PIT_TFLG_TIF_MASK ;//   Clear Timer Interrupt Flag
        MotorSpeed=FTM2->CNT;      //读取速度
        FTM2->CNT=0;                    //计数值清零
        MotorPidVal=MotorPidCal(SpeedPid);
        MotorRun(MotorPidVal);
    }
}
```