目　录

第一章　总体方案

智能车的核心单元是 RT1064 单片机。i.MX RT1064 是 i.MX RT 系列芯片，是由 NXP 半导体公司推出的跨界处理器芯片，该系列下又包括 i.MX RT1020、i.MX RT1050 及 i.MX RT1060 等子系列芯片。所谓“跨界”，是指它自身的定位既非传统的应用处理器也非传统的微控制器。[1]

传统的应用处理器，如手机主控芯片，通常采用 ARM 的 Cortex-A 系列内核，配合其芯片架构使得芯片能实现更高频率的运行。传统的微控制器也称为 MCU，通常采用 ARM 的 Cortex-M 系列内核，相对来说该内核对中断响应更快，所以具有良好的实时性，但其芯片架构特别是集成片内闪存带来了生产技术限制和成本负担，从而限制了其性能。而 i.MX RT 系列芯片集成了两者的优点，其基于应用处理器的芯片架构，采用了微控制器的内核 Cortex-M7，从而具有应用处理器的高性能及丰富的功能，又具备传统微控制器的易用、实时及低功耗的特性。

我们采用逐飞 RT1064 核心板，主频高达 600M，且板载 4M 大容量片上 Flash 和 512KB 的 RAM，满足程序开发的需求。之所以选择高性能的单片机，是因为十五届全国大学生智能汽车竞赛中摄像头的高度从 20cm 下降到 10cm。传统的基于 OV7725 等二值化摄像头的开发方式虽然数据量小，易于单片机处理，但是已经不再满足新的赛道条件的需求，我们彻底弃用 7725 的方案，而采用 MT9V034 灰度摄像头，使用边缘检测与二值化并用的方式进行图像的预处理，而更复杂的算法需要更强的单片机性能，同时使用跨界处理器对嵌入式系统的学习也大有裨益，所以综合各方面考虑，我们在练习过 LPC54606 后果

断选择了 RT1064 方案，摄像头采用总钻风 MT9V032，150FPS 下工作。高帧率带来的好处是图像更新速度更快，能够采集到的信息量更大，但缺点是图像亮度会受影响，且过快的帧率会超出控制的需求，即图像更新的信息在控制时可能不被采用，通过配置摄像头增益等参数可以对图像优化，需要根据实验效果做出取舍。

智能车系统的工作原理是：

1. 用 MT9V032 摄像头以 150fps 左右的帧率拍摄赛道图像，得到的数据以 DMA 自动传输（场中断传输和行中断传输）的方式输入 RT1064，通过一定的软件程序对图像进行反光点去除、边缘检测+二值化预处理、校正、分割并提取黑色边线，并经过路径规划得到小车的预期路径；通过检测是否为十字（全白行全白列或斜白行斜白列）或环路（一侧为环路特征，一侧近似直线），判断赛道类型。
2. 由规划好的路径，选择参考行，计算出偏差，用模糊 PD 控制舵机转向；当赛道为坡道时，改用电磁巡线，出坡道回到摄像头状态。
3. 通过编码器检测车速，通过 RT1064 输入捕捉功能进行脉冲计数，用 pit 中断获得一段精确的时间（2ms），脉冲除以时间，并进行一定的换算，即可得到小车的实时速度。用路径的偏差计算出预期的速度，并利用 PI 控制算法以及 bang-bang 控制算法调整电机来控制小车速度。
4. 智能车还辅有陀螺仪进行坡道判断，用 I2C 协议向陀螺仪 ICM20602 的相关寄存器中实时读取当前 y 方向（小车的左右方向）的角速度值，以获得小车前后翻转的角速度，从而判断当前是否正在经过坡道或者颠簸。由于智能车撞到路肩后可能出现坡道误判，还需加入防误判算法。
5. 向 UART 转 Wi-Fi 模块以 150000 的波特率发送串口数据，这个模块将这些

数据传输会上位机进行实时监控，从而可以让我们获得智能车运行过程中的各项观察指标，使得参数调节变得更加方便

6. 同时辅有拨码开关和按键以及 IPS 显示屏（OLED 显示屏分辨率较低，且为二值化显示，IPS 分辨率更高，能显示摄像头原图数据，但是数据是 16 位灰度数据，传输数据量非常大，在显示图像和变量时，由于软件 SPI 中的延时，导致主程序即图像处理程序运行仅仅为 20FPS，不满足智能车高速运行时的帧率需求，所以 IPS 屏仅能在静止调试时使用）进行控制策略选择和参数设置

系统框图如下：

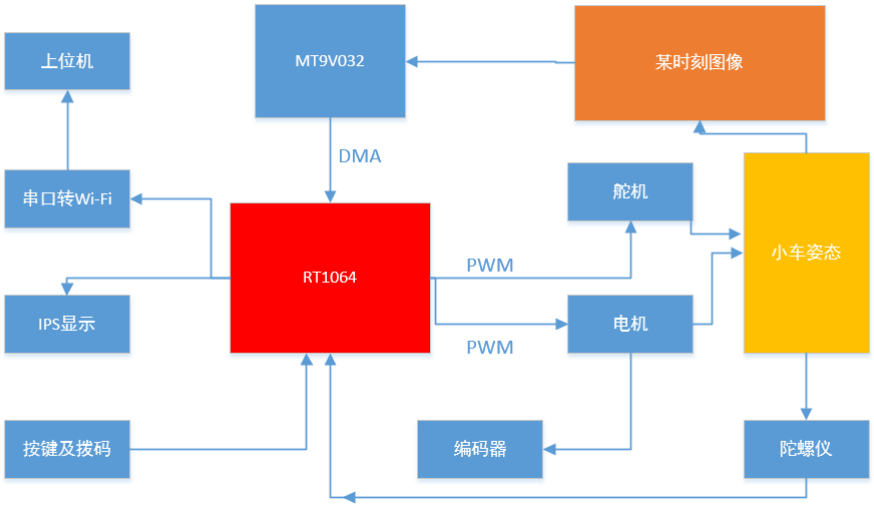

图 1 系统框图

第二章 机械安装

小车的机械决定了小车的最高速度，所以机械是很重要的一部分。机械的变化可以提升小车的速度上限，另外，机械的稳定性非常重要，例如小车在坡道摔下后，如果撞坏了电磁杆，那么整个比赛将功亏一篑。下面是我们在调整过程中总结出来的一些经验。

2.1 重心

小车的重心可以说是整个机械最重要的两个东西之一，首先重心要尽可能的低，当然不能擦到地面。在高速过弯的时候，重心较高的车更有可能会抬轮，就是弯侧的两个轮子由于离心力作用抬了起来。此时只有两个轮子着地，并且只有很小一部分接触了地面，所以就很容易发生侧滑、甩尾或者跳轮（轮子和地面不停撞击）的现象。

另一个是重心的前后位置，重心太靠前，后轮的抓地力不够，就会发生甩尾；重心太靠后，前轮抓地力不够，就会发生侧划（前轮滑动）的现象。而整辆车的重心的源泉，就是电池。采用锂电池后，电池重量减轻，但重量仍然不低。所以电池的位置很关键。

整车的重量要尽可能的轻，这样电机的加减速性能会比较好。前轮侧滑可以加几个配重块，但是不宜加太多。

2.2 舵机和前轮

前轮是这辆车中结构相对最为复杂的一部分机械了。前轮转向部分主要有

舵机、舵机摆杆、拉杆、主销、主销座和轮毂构成。

舵机横着装或者竖着装差别不大。摆杆不宜太短，一般取的尽量长一点，但也不要太长，可以缩短舵机滞后时间。拉杆一般装成略略带 V 的形状，这样可以在大弯的时候获得最大的拉力。

主销可以调的东西是最多的。通过调节拉杆的长度，我们可以调节主销的前束角，通过拧主销座上的两个螺丝，我们可以调节主销的内外倾角。一般前束角（从上往下看前轮是内八还是外八）取 0（两个轮子平行）就可以了，而内外倾角（从前往后看前轮是内八还是外八）一般取成负的，即从前完后看前轮的间距上大下小。当前轮左后晃动幅度较小时，倾角要小一点，当前轮由于损耗左右晃动幅度很大时，倾角可以稍大一点。这样可以时在弯中行驶时总有一个前轮是整个轮子都贴在地面上的，从而增大了前轮的摩擦力。

前轮的主销可以说是易损件，在行驶过程中遇到障碍或者坡道，对前轮都是一次巨大的损伤。一副新的主销，基本跑个几次后就松动了。松动之后可以将主销内倾角加大一点来补偿松动。

2.3 摄像头安装

i. 首先是摄像头架子的安装位置，一般是放在车的中间，但是由于电池位置需要往前移一点，摄像头位置要不就很靠前，要不就很靠后了。我个人倾向于往后放。因为理论上靠后放前瞻并不会拉近多少，而靠后放反而可以将视野范围扩大。小车在大弯中靠外圈行驶时，很可能会造成摄像头看不到或只看到一点点边线，而摄像头往后放可以让摄像头看的范围扩大。

ii. 摄像头的高度、角度。摄像头的高度角度，主要决定了摄像头能看到的最远端和最近端。一般来讲，摄像头的最近端越靠近车头越好，最远端当然是越远越好。此外前瞻行（一般处在车前 40cm~70cm 左右）尽量的靠近图像的下半边，因为图像的下半边最稳定，不容易发生边界跳动的情况。

iii. 有关摄像头支架，由于在小车行驶中，摄像头的杆子会晃动。我们为了减小这个晃动，就用两根细的碳素杆把主杆给支起来了，这样主杆强度也不用那么大，所以换成了细一点的主杆。这样摄像头不晃动了，但是另一个问题出现了，就是车的底板会因为摄像头支架的安装不对称而变形，有时即使一开始把支架安装好了，但是由于小车路过颠簸、坡道，把摄像头支架震变形了，一方面把底盘给搞变形了，另一方面把主杆给扯歪了。所以我们最后用的还是只用一根粗杆的方案。

第三章 电路设计

我们的电路设计做的比较早，我们改用了主控板和驱动板分开的方式，这样一方面可以一定程度上减少电机驱动部分电路对主控的干扰，另一方面缓解了布线压力，所有电路板都可以采用双面板，而嘉立创今后打双层板 5 片只需要 5 元钱，材质也满足智能车的需求，采用分立的模式值得选择。之后我们对电路板的设计进行了优化，将主板以及电机驱动板均做成了 4 层 PCB，驱动板的面积缩减为不到原来的 1/2，主板的面积缩减为原来的 3/4。更小的面积有利于电路板的安装。

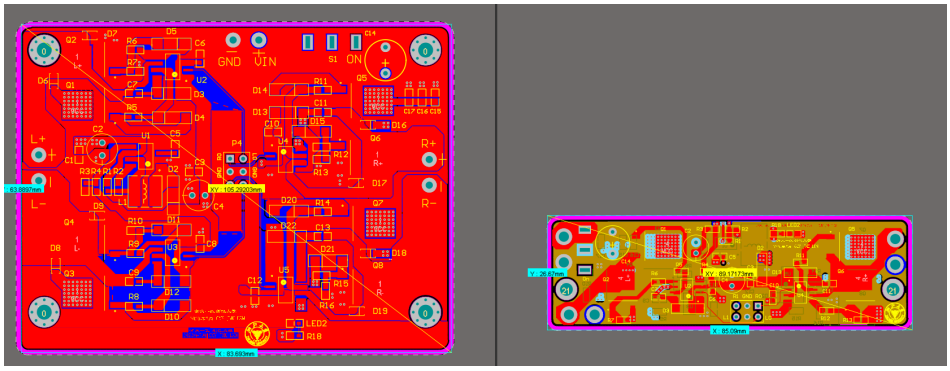

图 2 原驱动板（左）与新驱动板（右）

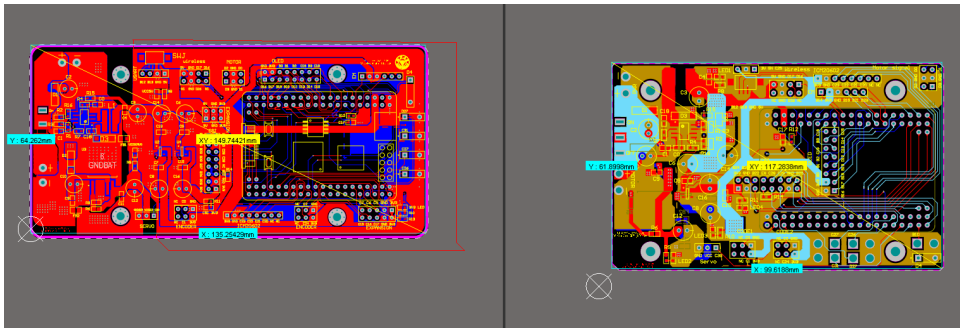

图 3 原主板（左）与新主板（右）

由于 RT1064 芯片外围电路较为复杂，主频较高，对电源质量、时钟信号走线都有很高要求。所以综合考虑下，相比绘制完整主板，直接采用官方核心板无疑是一个更好的方案。

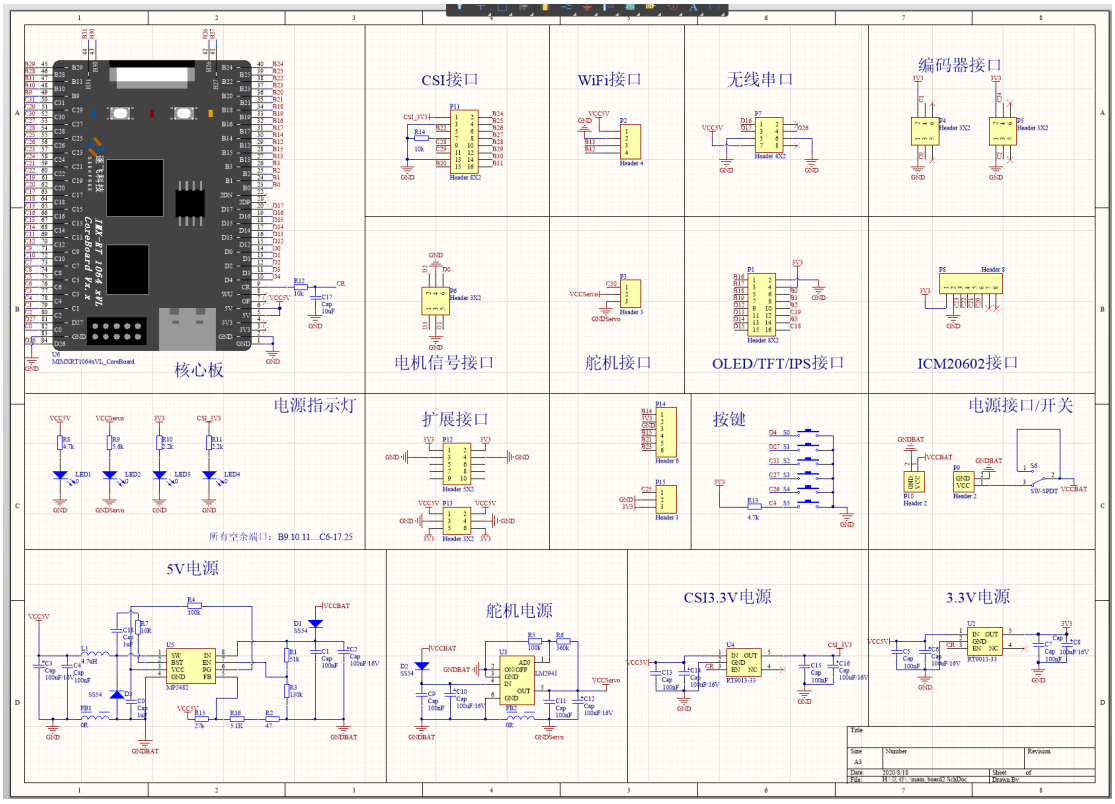

图 4 主板原理图

在主板上，我们首先采用 MP2482 进行电源电压到 5V 的稳压，相比以往采用的 LDO，该芯片为 buck-boost，在电源电压被电机堵转拉低时，还有较好的稳压能力和输出能力，该芯片相较于 TPS63070 最大的优势在于封装为 SOC-8，非常易于焊接，而 63070 的封装为 VQFN，即无引脚封装，焊接起来存在一定困难。其输入电压最低为 4.5V，输入电压范围比 TPS63070 小，但已经可以基础四轮组对于电源的要求。

在 3.3V 稳压上，我们采用了两路 RT9013-33LDO 线性稳压器，一路供给摄像头，一路供给 IPS 屏幕等传感器。该稳压器电路十分简单，且易于焊接调试。

在舵机的电源上，我们采用了 LM2941 提供 6V 的电压，该方案能够提供的最大输出电流为 1A，LM2941 的限流能力能够很好的保护舵机。

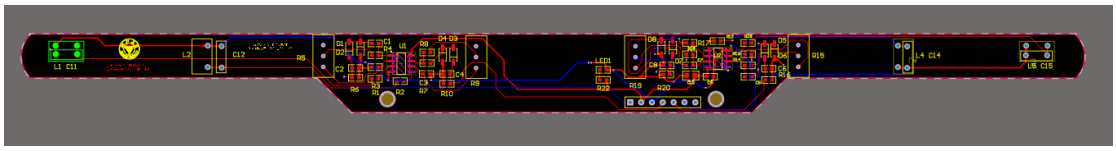

图 5 电磁杆

在电磁运放电路上，我们选择了 OPA 系列的运放，为节省空间将运放电路放置在电磁杆上，此设计可以节省主板空间。但通过实践，在实际调试过程中，电磁杆上的元件易因为撞击而撞裂焊盘，所以十分不建议将运放电路放在电磁杆上。

在驱动电路上，我们沿用最常用的方案，2104（驱动）+7843（MOS），性能稳定，驱动需要一个 12V，我们采用的是 34063，该升压电路的反馈电阻（0.22Ω）一定要保证阻值正确，且电容不能过大，否则电路工作不正常。另外，要保证通过大电流的通路的铺铜宽度足够。

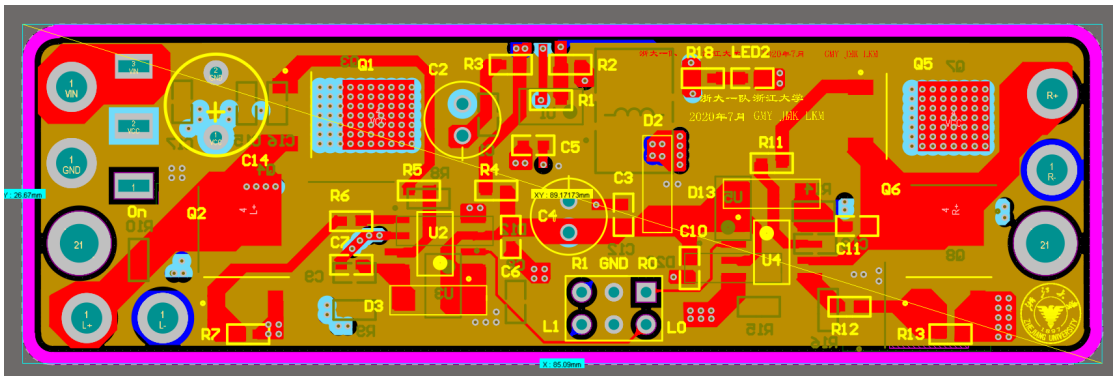

图 6 驱动板

总体来讲，电路设计以稳定、通用、成熟方案为宗旨，因为智能车电路不涉及过多的 PCB 设计知识，只要布局合理，电源处理恰当，即可工作非常正常，所以尽量不要追求过分个性化的设计，比如异形 PCB，过度密集的布局、过分追求更小等等，也不用过分参考智能车店家的设计，比如有的智能车主板设计了很多地平面的分割，实际并没有必要，徒增难度，甚至可能适得其反。

对于一般新手，零基础开始设计，一般要经历至少三版以上的返工，一定要总结经验，不能盲目设计。同时在焊接技术上，尽量采用质量更好的烙铁，锻炼更加熟练的焊接技能，对于数量充足、价格不高的元件，尽量不要采用拆机件，对于关键的元件，如电源、MOS 等，尽量在正规渠道购买，贪便宜吃大亏，尤其在比赛前，硬件损坏对心态有可能产生致命的打击。

第四章 图像信息获取

4.1 摄像头的配置和采集

在摄像头上，我们采用了逐飞的 MT9V032，MT9V032 和 MT9V034 性能较为接近，032 并不比 034 性能差，只是个别配置有微小的差别，而在智能车应用中没有影响，采用 032 的最大原因是，逐飞的 032 是通过一个 51 单片机配置，只需要通过 MCU 的串口传给 51 数据即可进行配置，配置代码清晰，需要配置的如帧率、自动曝光、增益等等都以一个结构体的方式呈现，修改起来非常简单（但相应的，我们也就无法得知 51 中对摄像头底层到底如何驱动的，SCCB 底层配置不开源），且经过实践，无论是显示二值化图，还是 IPS 显示原图，证明了该款摄像头的图像质量符合要求，所以我们选择了此款摄像头。

在第十五届全国大学生智能车竞赛中，逐飞和龙邱等零件供应商提供了 130°无畸变摄像头，相较于 140°有畸变的 MT9V032，去畸变的效果较为显著，也可以作为一种选择。

4.2 数字摄像头介绍

MT9V032 有两大非常突出的优势：一是全局快门，相比卷帘快门，在高速下依然可以保持非常好的图像质量；二是高动态性能，90DB+的高动态范围。在实际使用时，该款模块的 FPS 可调，可自动曝光（但一般不能使用），曝光时间可调（根据场地光条件决定），不需要我们自己写驱动程序（有利有弊，配置更加稳定，但对摄像头底层不够了解），且也可通过取位的方式实现二值化（我们没有使用）。

1) 图片大小、分辨率的选择

032 的大小和分辨率只能是 752*480，但是可以通过将两个或者四个像素点并成一个的方式，将图像变成 188*120，再用软件每两行/列取一个点（或者根据画幅，丢弃远处没用的行 从下面开始连续取 60 行）经过裁剪，可以最终将图像变成 80*60 的图像，而视域又没有太多的变窄，只需要相应处理畸变、配合舵机控制程序进行一定调整即可，使用起来相比 7725 等固定分辨率的更加灵活

2) 140 度无畸变广角镜头

由于库中我们只需要配置结构体，所以摄像头底层具体的细节我们没有过多的研究，在此也不做过多的介绍，但建议有余力的话可以对摄像头手册进行一定研究。除去底层，我们使用的摄像头最大的改变是广角镜头，由于环岛等赛道元素的存在，且摄像头高度逐年的降低（15 届为摄像头车身最高处不能超过 10cm，镜头中心实际仅为 9.9cm 左右），所以采用更广角的摄像头获取更宽的视野是有很大好处的，但更宽的摄像头也存在图像过远和入弯补线错误的情况，这需要同时调整高度和程序，以达到理想的效果（龙邱的 150 度镜头视野远没有逐飞的 140 度视野广，在实际选择时要根据个人需求以及程序的参数做出恰当、合理的选择）。

4.3 数字摄像头的使用

数字摄像头的使用步骤大致一样，如果没有 51 配置的话，是用 SCCB 协议对摄像头的寄存器进行配置，然后单片机需要配置好自身的中断和 DMA 传输，并自动获取摄像头图像。而我们采用的摄像头，仅需要使用库中的配置进

行一定修改，即通过串口向 51 发送数据，然后再通过 DMA 接收数据即可。

1) 摄像头硬件连接

9V032 摄像头模块是 3.3V 供电，这里，因为该模块上有一块 51 单片机，所以不能和 MCU 等用一个 3.3V 稳压芯片供电，应该单独使用一路 3.3V LDO 进行供电。除了 3.3V 和 GND，还有行场中断，串口（一般摄像头为 SCCB），8 位数据线。

2) SCCB 寄存器配置（一般摄像头）

SCCB(Serial Camera Control Bus)是和 I2C 相同的协议，一般由两根线组成，一根 SCL 时钟线和一根 SDA 数据线。我们在摄像头初始化的时候可以用 SCCB 协议对摄像头的寄存器进行配置(读寄存器或写寄存器)，需要注意的是 7725 的寄存器是八位的，而 9034 的寄存器是十六位的，所以写和读的时序有些许不同。它们具体的时序可以查阅相对应的芯片手册。

3) 中断传输

摄像头一般有两种传输方式，一种是场中断传输方式，一种是行中断传输方式，场中断即场中断信号来后开启 DMA 传输，直到一幅图传输完成；行中断即行中断信号来后开启 DMA 传输，有几行就需要进几次行中断。在初始化时需要配置好行/场的中断源、触发边沿和上/下拉电阻。然后在中断函数中清空标志位和打开 DMA 传输就可以了。

4) DMA 配置

数字摄像头可使用 DMA（Direct Memory Access）技术。DMA 是一种直接

存储器访问技术，工作过程中不需要 CPU 干预，也不需要像中断处理方式那样需要保留现场、恢复现场。

简而言之，若关闭 DMA，则需要占用 CPU 进程用于摄像头数据传输，无法执行主循环程序。开启 DMA 后，摄像头有数据过来，CPU 开启 DMA 记录摄像头数据，在 DMA 记录数据时，CPU 可以执行其他的任务，等 DMA 记录完数据后，CPU 再对数据进行图像处理，节约了时间。

4.4 图像二值化阈值选取

图像二值化有很多算法。这里主要讲述大津法及平均值法两个方法

a) 大津法(OTSU)

大津法又名最大类间差方法，是日本学者大津于 1979 年提出的一种自适应阈值分割法。它通过统计整个图像的直方图特性，使得阈值T_0'能将图像分为相差最大的两类。

设阈值T_0'将图像分割为目标图像和背景图像两个部分，属于目标图像的像素点数占整幅图像的比例为ω_0，其平均灰度为μ_0；属于背景图像的像素点占整幅图像的比例为ω_1，其平均灰度为μ_1。图像总平均灰度为μ，类间方差为g。则有

$$\mu = \omega_0\mu_0 + \omega_1\mu_1 \qquad \text{(公式 1)}$$

$$g = \omega_0(\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2 \qquad \text{(公式 2)}$$

其中，$\omega_0+\omega_1=1$。将(公式 1)代入(公式 2)有

$$g=\omega_0\omega_1(\mu_0-\mu_1)^2 \quad \text{(公式 3)}$$

使得类间方差g最大的阈值T_0'即为最佳分割阈值。

b) 平均值法

平均值法是指将算整场图像像素的灰度值的平均值μ作为图像的分割阈值T_0'。[1]

由于单片机主频高，算力足，这里采用效果更好的大津法来算去全局阈值。然而由于图像四周受畸变影响较重，图像相对正常图像较暗，加之工程实际与理论存在一定差别，为了给图像阈值的更多可调整空间，实际选取阈值还需在计算结果上加上一定偏置，即$T_0=T_0'+T_{offset}$。

4.5 反光点去除

由于赛道路面不完全平整，在灯光照射下会使得采集到的图像出现光强过高的“反光点”（即灰度值偏大），从而导致 2.1.1 中计算得到的阈值偏低。同时，此情况下也会影响后续边缘检测得到的赛道边线特征。

为解决此问题，我们对灰度图像进行了反光点去除处理。对于统计得到的图像直方图，可以获取每个灰度值 i（i=0,1,2,…,255）对应的像素点个数 N_i。由于反光点个数相对较少，故可以通过设置像素点个数阈值 N_{min} 来滤除该类型噪点，从而获取更为准确的灰度图像。

$$N_i = \begin{cases} N_i, N_i \geq N_{min} \\ 0, N_i < N_{min} \end{cases} \quad \text{(公式 4)}$$

然后将被滤除的噪点灰度值置为与其余像素点最大灰度值相同，并更新灰度直方图用于阈值选取，从而实现反光点的去除。

4.6 边缘检测

而不论是 OTSU 还是平均值法，都属于全局阈值，在赛道光线不均匀时，仍有使得图像不稳定的情况，，采用边缘检测的方式保证赛道边线提取的正确性就非常有必要。我们采用 Sobel 算子进行边缘检测。

索贝尔算子（Sobeloperator）主要用作边缘检测，在技术上，它是一离散性差分算子，用来运算图像亮度函数的灰度之近似值。分别用横向和纵向 sobel 算子对灰度图进行卷积运算，可以得到图像灰度矢量信息和其法矢量信息。[2]

sobel 算子有横向和纵向两种卷积核：

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \; S_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \text{(公式 5)}$$

将横向与纵向算子分别与图像平面卷积，得到横向及纵向的亮度差分值G_x及G_y。记灰度图像矩阵为 A，则有

$$G_x = S_x * A \; , \; G_y = S_y * A \quad \text{(公式 6)}$$

于是得到图像梯度矢量大小为

$$G = \sqrt{G_x^2 + G_y^2} \quad \text{(公式 7)}$$

由于赛道边缘点附近黑白梯度最大，故可认为大于临界阈值G_0的点为赛道边缘点。

又由于摄像头透视效果的影响，图像远景区域会将更多的图像信息投射到同一个像素点，这也就会造成图像远景区域的赛道边缘临界阈值更小，故临界阈值G_0应在图像上分段或动态给出。

4.7 图像二值化

依据上述阈值选取结果与边缘检测结果，对灰度图像进行二值化处理。

对于灰度图像中每一个像素点，首先根据灰度值与阈值对比进行预二值化，对于上述的阈值T_0，灰度图像中每个像素点$(\mathrm{x,y})$，以及其对应的灰度值$\mathrm{i}_{(\mathrm{x,y})}$，记二值图像对应值为$\mathrm{p}_{(\mathrm{x,y})}$，则有：

$$\mathrm{p}_{(\mathrm{x,y})}=\begin{cases}0\ ,\mathrm{i}_{(\mathrm{x,y})}<T_0\\1\ ,\mathrm{i}_{(\mathrm{x,y})}\geq T_0\end{cases} \quad \text{(公式 8)}$$

然后根据边缘检测结果，对二值图像结果为 1 的点（亮点）进行进一步检测，记每个像素点对应灰度梯度矢量大小为$\mathrm{G}_{(\mathrm{x,y})}$，则有：

$$\mathrm{p}_{(\mathrm{x,y})}=\begin{cases}0\ ,\mathrm{G}_{(\mathrm{x,y})}>G_0\\1\ ,\mathrm{G}_{(\mathrm{x,y})}\leq G_0\end{cases} \quad \text{(公式 9)}$$

最终得到更新的二值化图像$\mathrm{p}_{(\mathrm{x,y})}$，并用于后续图像处理等。

4.8 边线识别

由于最后得到的是二值化+边缘检测图像，所以黑线的识别相对比较简单。困难之处在于各种赛道类型的判断，以及一些黑线的有效性判断。

4.9 图像校正

由于摄像头本身存在外畸变，故对图像做逆透视变换已得到小车坐标下的标准图像。

透视现象实际上是世界坐标到摄像头坐标的转换，即经过一系列的旋转和平移，将每个点在小车坐标系的相对位置转换到摄像头坐标的相对位置，最后再经过一个投影，将摄像头坐标转换成平面坐标。由坐标变换的相关知识可以知道，坐标系的旋转、平移变换都可以用一个 3*3 的非奇异常数矩阵来表示，而投影可以用一个 3*3 的奇异矩阵表示。此外，由于智能车看到的图像都是平面的赛道，所以其实在世界坐标系中，只用到了 x 和 y 方向。所以总而言之，智能车中的透视可以说是一个二维坐标到二维坐标的线性变换。用齐次式来表示，可以将透视表示成：

$$\begin{bmatrix} \overline{x} \\ \overline{y} \\ \overline{s} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad \text{(公式 10)}$$

其中 X, Y 是世界坐标系，x, y 是摄像头坐标系。而这个 H 矩阵，就是要求的变换矩阵，而逆透视最关键的步骤，就是求解这个 H 矩阵。

求解逆透视主要有 3 个参数，由于理论计算、测量得到的参数精度不够，所以采用系统辨识的方式得到关键参数是一个很好的选择。

由于对整幅图像进行逆透视计算量较大，所以只对关键信息——边线进行逆透视是一个更好的选择。于是通过逆透视矫正，将赛道矫正为宽为 300 像素的标准小车坐标下的赛道。

第五章 电磁信息获取

针对此届智能车大赛的新规则，电磁传感器放置位置受限于摄像头视野范围，无法距离车头很远，实际最大只能达到 3-5cm 左右的前瞻。为了矫正传统偏差计算方法带来的误差，我们首先来看下经典电磁模型下的传感器状态。

首先一根无限长通电导线的磁感应强度是和导线距离的平方成反比的：

$$B = \frac{B_0}{r^2} \tag{公式 11}$$

对于赛道我们可以将其电磁线在小尺度下看作是无限长通电导线。又因为赛道电磁线是通交流电，所以通过电感的磁感应强度也是交变的，对应感生电流也是交变的，只要得知电感到电磁线距离 l，就可以得到感生电流的大小：

$$I_{感应} \propto 1/l^2 \tag{公式 12}$$

接下来我们就用距离平方的反比来代替小车测得的电感电压。实际系统中将交变电流做了交-直变换后得到了电感电压，但是这是一个线性的变换，所以测量值依旧和距离平方反比成正比例关系。

在我们的小车机械机构上，电感距离地面电磁线具有一定高度 h，而左右电感的相对距离是个定值 n。所以可以得到两侧电感测量值对应表达式：

$$\begin{cases} u_{测量\,1} \propto 1/(h^2 + a^2) \\ u_{测量\,2} \propto 1/(h^2 + b^2) \\ a + b = n \end{cases} \tag{公式 13}$$

但是值得注意的是以上公式未考虑电感并没有完全垂直电磁线的情况，事实上因为新规则的引入，电感相较于往届更容易随车头而左右晃动较大的偏

角，使得测量值出现较大波动，此时如果不针对角度量执行矫正会出现较大的测量误差。

在引入角度偏差的情况下，上述公式修正为[3]：

$$\begin{cases} \mathrm{u}_{测量\,1} \propto \frac{1}{h^2+a^2} * \cos(\theta) \\ \mathrm{u}_{测量\,2} \propto \frac{1}{h^2+b^2} * \cos(\theta) \\ a+b=n \end{cases} \quad (公式\ 14)$$

其中 θ 代表电磁线偏离小车电磁感法向量的角度，接下来我们讨论下如何测量这个角度。

假设在测量点引入另一个电感，这个电感垂直于原电感，那么同理应该有类似的电感测量值公式：

$$\begin{cases} \mathrm{u}_{测量\,1}{}' \propto \frac{1}{h^2+a^2} * \sin(\theta) \\ \mathrm{u}_{测量\,2}{}' \propto \frac{1}{h^2+b^2} * \sin(\theta) \\ a+b=n \end{cases} \quad (公式\ 15)$$

因此理论上只需要将两个测量值相除直接可得 θ 的正切值，但是实际系统中由于并不是理想的无限长直导线，以及测量值较小时存在较大测量误差，因此应用中选择测量值相对大的一组电感相除来计算 θ 角。由此我们可以得到这个偏差角，此外如果假设车辆轨迹是理想的电磁线切线的情况下，这个角度反馈的是弯道的曲率，就可以用于前文所述主动差速的输入了。

因为在上述方程组中仅有 a 和 b 是未知量，而求取$\frac{(a-b)}{2}$也就是中线偏差是主要目的，所以可以直接得到这个偏差正比于一个量：

$$\frac{(a-b)}{2} \propto \left(\frac{1}{\mathrm{u}_{测量\,1}} - \frac{1}{\mathrm{u}_{测量\,2}} \right) * \frac{\cos(\theta)}{n} \quad (公式\ 16)$$

再结合前述 θ 表达式：

$$\cot(\theta)=\begin{cases}\dfrac{u_{测量1}}{u_{测量1}'} & u_{测量1}{}^{2}+u_{测量1}{}'^{2}>u_{测量2}{}^{2}+u_{测量2}{}'^{2}\\ \dfrac{u_{测量2}}{u_{测量2}'} & u_{测量1}{}^{2}+u_{测量1}{}'^{2}<u_{测量2}{}^{2}+u_{测量2}{}'^{2}\end{cases} \quad \text{(公式 17)}$$

即可求得中线偏差从而给后级舵机 PD 控制使用。

然而在这个模型中依然存在一些误差，首先两个放置很近的电感存在无法抵消的互感现象，实际使用中尝试过对称布局三个电感、使用铝箔电磁屏蔽等等方法，均存在问题于是作罢。另一方面实际上不可能将两个电感测量同一点场强，因为电感体积较大。

第六章 方向控制

对于 C 车模，舵机的控制算法需要在速度为 2.0m/s 以下调节，当速度继续提升时，需要依靠主动调节电机的差速进行转向，若是单纯调节舵机，小车将无法正常通过不同类型的赛道。舵机的控制首先需要获得当前道路偏差，所以需要先讲一讲偏差是怎么算出来的，然后再说舵机的 PD 控制。

小车转向由两个部分构成，其一是使得小车行进方向与赛道切线方向平行（以下简称车直），保证小车是沿着赛道前进的；然而，即使小车沿着赛道行进，但小车可能不在赛道的正中，这会影响小车的后续状态的稳定性，所以也需要控制舵机使得小车向中间修正，即这一部分的目的是使得小车处于赛道的正中间，一下简称车正。

控制行就是用来计算转向偏差的参考行，前瞻远时，舵机打角就会提前。由于舵机的响应受硬件的限制，一般响应时间是 20ms，舵机响应的滞后需要用前瞻来进行补偿。

6.1 车直转向偏差值的计算

从理论上来说，小车前轮平均打角应与中心线切线平行，或者说根据阿克曼转向模型使得小车行进曲率半径等于前方中心线曲率半径的打角。

然而实际中，由于舵机打角滞后、中心线曲率拟合准确率低或计算量大、小车车轮与地面存在打滑等因素。所以，为了快速且准确的计算舵机打角，可以考虑将中心线的斜率来做为车直偏差是一种方案。

通过最小二乘法计算出中心线斜率，在入弯、弯心、出弯等过程中，

与小车前轮需要的打角之间都是单调的，所以此斜率是可以作为车直转向偏差的。

6.2 车正转向偏差值的计算

车正转向偏差的计算比较简单，即近景行中心线与小车中心位置的差，由于小车在转弯时赛道中心线与小车中心位置存在固有偏差，所以在转弯时，也需要对此做特殊处理。

6.3 车直舵机的 PD 控制

由于车直偏差与舵机打角属于线性关系，所以直接采用 P 控制。然而，实际中，由于小车与地面存在打滑现象，所以当速度增加时，舵机打角也需要增大，为了使小车更稳定，我们通过将 P 参数和速度进行耦合，解决了不同速度下舵机打角不同的问题，且调试参数较少。小车在行驶过程中，速度会出现变化，将速度和 P 耦合的做法极大的提升了小车的稳定性。

又由于纯 P 控制会导致小车在直路上存在超调，故在小车在直路上加以 D 控制以消除超调及震荡。

6.4 舵机的 PD 控制

由于车直偏差与舵机打角属于非线性关系，所以直接采用 PD 控制效果不佳。而将非线性偏差映射到线性偏差，也是困难且不必要的。所以，采用模糊 PD 控制，设置线性 PD 参数表，并和速度进行耦合，来使得小车快速且无超调的恢复到赛道中心位置。

第七章 速度控制

7.1 速度的获取

速度一般就用编码器加装在后轮差速器齿轮上进行测速。需要注意的是，编码器有三线编码器和四线编码器。三线编码器只能测量速度，并不能测量速度的方向。而四线编码器（我们用的是 512 线 ABI 编码器）可以测量速度的方向。

四线编码器由电源线、地线和 A，B 两相的输入组成。A，B 两相处于正交编码状态，即正转或者反转时 A，B 相的相位差会不一样。一般的单片机都有正交编码功能，可以自动的对 A，B 相进行解码（计数），计数的结果是有正有负的。

计数值乘以一定的系数后可以转换成距离，这个系数得到的方法可以用手推小车一米，看计数值是多少，然后就可以计算出这个系数。距离得到了，还差一个时间，我们使用的是 PIT 定时中断，即初始化完成后经过一定的时间（最终采用了 2ms），它会进一次中断，在中断中读一下计数值，再换算成距离，除以时间，最终就可以得到速度值。

这里需要注意的是，我们的系统中除了 PIT 中断，还存在着很多中断，其中，摄像头的场中断和 DMA 中断应该是系统中最重要的，所以相应的优先级应该排在前两位，PIT 中断选择一个比较靠前的合适的位置即可，虽然可能被打断，但测速的误差在智能车系统中可以接受。

7.2 速度控制算法

在速度较低时，速度控制的重要性不显，但是当小车速度越来越高时，速度控制显得尤为重要。并且当速度的性能要求较高时，传统 PI 控制算法已经满足不了需要。故我们对速度控制策略进行了优化。

根据我们的经验，在速度 2m/s 以下，基本不需要进行电机速度控制，在速度 2m/s 以上，则必须加入速度控制算法[4]，例如 PI 控制、PI 差速控制、增量 bang-bang 差速控制、定 PWM 差速控制等。

首先是速度控制周期的选取，一般图像采集一场的周期是 6~7ms，速度控制周期应与之相近。速度控制周期太长，会导致控制不及时，速度会产生很大的超调；周期太短，由于在一个周期内各个量变化较小，可能会导致速度测量的误差影响增大。

速度控制一般采用 PI 算法，并且是一个严格意义上的随动系统。首先根据路径计算出当前的期望速度。我们采用与舵机偏差相似的计算方法。由于电机作用的滞后比舵机更加严重，其应该具有更远的前瞻性。获得期望速度后，与测得的速度比较，得到偏差 error，然后就用标准的 PI 控制对电机进行控制。

P 分量：为了让小车可以更快的加速或者减速，一般需要将 P 调大，但会导致减速时极易产生超调。如小车刚入弯时，速度明显减慢，但小车在弯心时，由于电机控制超调，小车反而出现了速度回升。所以 P 要满足在可承受的超调下尽量大。

I 分量：I 分量的目的是消除静态误差。为了防止积分饱和，需要给 I 分量

加上限幅。但是实际上有了 I 分量后，也存在着一定的静差。

补偿：由于电机存在着反电动势，当转速越快时，反电动势就越大，在整体表现上就显得不灵敏，在低速时增大 1V 和高速时增大 1V 效果是不同的。所以这个反电动势对控制是不利的。为了补偿这个电动势，我们可以在电机的输出中加一项正比于速度的项来补偿反电动势，再加一项常数项来抵消一些阻力。由于这些补偿属于正反馈的范畴，所以不宜加的太大。

7.3 差速控制算法

由于我们采用的车模采用双电机驱动，因此如果采用同样的速度目标去设定电机的目标速度会导致和经典的四轮车转向模型存在矛盾，导致舵机转向的不灵敏，相当于舵机系统滞后性更强。为了实现真车类似的差速结构，根据经典转向模型（车辆转向中心在后轮延长线上并且后轮滚动摩擦），假定前轮内侧车轮为滚动摩擦，前轮外侧车轮存在一定横向滑动[5]，就能通过车身尺寸信息结合舵机转向角度计算出转向半径进而计算出合适的车轮差速，但是这里只能提供关于后轮两个速度的一个方程，另一个方程这里选择了两侧车轮速度之和等于设定车速乘二，进而希望车辆的重心运动速度保持恒定值，减少纵向加速度负担，进而尽可能使得车辆过弯时候保持相对匀速。

纵使我们采用了这样的差速模型，能够计算得到的也是理论的实际车轮速度，但是在小车系统中我们无法直接干预车子的实际轮速，因此通过模型计算得到的理想轮速只能是两侧车轮 PI 控制器的设定值，带来了较大的滞后性。为了弥补这个滞后，我们在差速模型的后级串接了一个 PD 控制器来更好的预测速度设定值的变化进而提供更为激进的差速，这样做不可避免的出现超调和震

荡，为了抑制之，在设置参数时尽可能减少 D 的值。

以上仅仅是基于经典模型提供的差速策略，接下来将其简称为“机械差速”，代表为了代替其他车模以及旧车模上的机械差速器结构所作的努力。但是实际赛道上由于小车速度高，地面抓地力状况并不乐观，导致即便是后轮也并不是完全的滚动摩擦，此时仅使用经典模型会出现极大的错误。为了减小这个因素的影响，策略有二：提高轮胎抓地力；修正经典模型。我们在两方面都做了努力，针对前者我们适当的对车轮进行磨损，提高车轮的抓地力使得在现有速度下尽可能保证滚动摩擦。针对后者我们又提出了新的模型并加入到我们的小车控制系统中。

因为所谓后轮的滑动只是因为后轮的实际速度要小于车轮的滚动速度，也就是码盘的测速值偏高，这会导致控制系统控制的后轮速度偏低。为了矫正这个问题，通过在两侧车轮的设定速度上加上一个 delta 的偏差量，进而去抵消这种偏差。注意到之前提到的机械差速模型中两侧车轮的设定速度是通过计算得出的，在此基础上加减一个值必然需要进行归一化，因此就等价于在车辆的重心速度上加一个 delta，从而最终的单侧后轮速度设定值应该由以下公式计算得出：

$$\mathrm{v}_{\text{设定}} = \left(v_{\text{重心}} * (1 \pm delta\%)\right) * (1 \pm p\%) \qquad \text{(公式 18)}$$

其中 p%代表经典模型中计算得出的单侧差速值，delta 的存在进一步扩大了两侧车轮的差速值，达到了预期的归一化和修正模型的目的。为了提供这个 delta 的值，我们还需要获取当前车轮打滑的程度，但是很遗憾目前的测量系统无法对这个变量给出较为准确的测量值。因此我们只能通过经验手段大致估计

合适的值，首先我们可以确定这样一个事实：舵机打角较小的时候代表弯道较缓或者处于直道，此时车辆抓地力应该较好，也就是不是很需要矫正经典模型，而在舵机处于大幅度打角时可以确定此时车轮打滑严重需要介入矫正经典模型。由于以上事实类似一个“主动提供一定差速值”的过程，因此称其为“主动差速”，用公式表达出来就是一个舵机打角的函数：

$$\mathrm{delta}\% = A * \mathrm{foo}(\text{舵机打角}); \qquad \text{(公式 19)}$$

其中 foo 函数是一个非线性映射，需要人工制定曲线，实际应用为了减少工作量仅仅尝试了几种定义在 0 到 1 的不同凹凸性的函数映射并且根据性能选取其中最好一种。A 表示一个介入量，范围是 0 到 1。可用于控制主动差速的介入程度。

实际系统中由于舵机打角并不能很好的反映实际弯道状况，因为舵机打角也不是完全贴合车轮偏差赛道中线的量去变化的。因此在不同系统中这个 foo 函数的输入会有所不同。在电磁巡线下，这里的输入采用了计算得出的电磁线法向量偏差角度的归一化结果。在光电巡线下，这个输入采用了识别得出的中线偏差值的归一化结果。这样就可以尽可能真实的反映赛道和车轮的动态关系。

第八章 其他传感器

陀螺仪在四轮光电组中速度控制方面并不是很重要，但是在上坡识别中，可以采用图像、激光和陀螺仪的方法，其中采用图像较为麻烦，采用激光又存在误判的情况，且在上坡过程中，采用陀螺仪积分的方式配合电磁打角比其他方式更加具有可控性，所以采用陀螺仪是非常有必要的。

我们选用的数字式陀螺仪 ICM20602，它支持 I2C 和 SPI 接口。我们采用库中的硬件 SPI 配置和采集方式，数据稳定，事实证明 ICM20602 比常见的 MPU6050 性能稍好，同时，我们还采用了卡尔曼滤波[4]（相对费时但单片机算力足够），对裸数据进行了标定，实际数据准确，上坡检测效果好。陀螺仪判定坡道会存在两个问题，第一个问题是小车压到路肩时会出现误判，需要加入误判检测，另一个问题是陀螺仪判定坡道时需要对角度进行判定，梯形坡道和圆弧坡道的特征不同，可能需要在比赛时根据实际情况通过屏幕和按键现场修改参数。事实上，光电管也许是个检测坡道的不错的传感器，但由于时间限制，以及我们的陀螺仪算法处理的效果很好，因此我们没有选择更换传感器。

第九章 开发工具及调试过程

开发工具选用 IAR (IAR Embedded Workbench)开发环境，它是一款应用于嵌入式系统的开发工具、C/C++的编译环境和调试器。其特征在于高效的 PROMable 代码；完全标准 C 兼容；内建对应芯片的程序速度和大小优化器；便捷的中断处理和模拟；以及高效浮点支持等。通过 JLink 等硬件仿真器，可以实现在线仿真调试。

硬件仿真器采用 DAP 下载器。DAPLINK 是一个可以在 ARM CORTEM MCU 上进行编程和调试的测试器，其具有 CDC 虚拟串口功能，可以在调试的同时进行串口通信；同时兼容两种下载调试模式：HID 免驱动兼容模式，WINUSB 高速模式。

参考文献

[1]卢守义,万星,卜令坤.基于 CMOS 摄像头的智能车设计[J].自动化应用,2018(04):138-140.

[2]李永,冯伟峰,李思光,王俊人.基于 MT9V032 摄像头的智能车软件设计[J].花炮科技与市场,2020(01):250.

[3]陈国定,张晓峰,柳正扬.电磁智能车电感排布方案[J].浙江工业大学学报,2016,44(02):124-128.

[4] Sui Jinxue. " NXP Cup Intelligent car Design and Example Tutorial [M]. Beijing: Electronics Industry Press. 2018.8.

[5] C. Kinnaird, "Digital adjustment of DC motor drive circuit parameters," 2016 IEEE Dallas Circuits and Systems Conference, Arlington, TX, pp. 1-4, 2016.

附 A　模型车的主要技术参数说明

改造后的车模总重约 882 克，长约 25 厘米，宽约 19.5 厘米，高约 11.5 厘米（其中摄像头中心高度约 9.5 厘米）。

车模静止状态下电路平均功耗 1.75W，电容总容量 2mF。

车模包含四类传感器：摄像头（1 个）、编码器（2 个）、陀螺仪（1 个）以及电磁传感器（含 4 个电感）。

除车模自带的驱动电机以及舵机外，未使用其它的伺服电机。

赛道信息检测频率与摄像头采样频率相同，约 150Hz；检测精度良好。

附 B　主要代码

1 大津法图像二值化阈值计算

```
1.  uint8_t GetOSTU(void)
2.  {
3.      int16_t i, j;
4.      uint32_t Amount = 0;
5.      uint32_t PixelBack = 0;
6.      uint32_t PixelIntegralBack = 0;
7.      uint32_t PixelIntegral = 0;
8.      int32_t PixelIntegralFore = 0;
9.      int32_t PixelFore = 0;
10.     double OmegaBack, OmegaFore, MicroBack, MicroFore, SigmaB, Sigma; // 类间方差;
11.     int16_t MinValue, MaxValue;
12.     uint8_t Threshold = 0;
13.     uint16 HistoGram[256]; //
14.     int16_t Mincount = 0, Maxcount = 0;
15.
16.     for (j = 0; j < 256; j++)
17.     {
18.         HistoGram[j] = 0; //初始化灰度直方图
19.     }
20.
21.     for (j = START_LINE; j < CAMERA_H; j++)
22.     {
23.         for (i = 0; i < CAMERA_W; i++)
24.         {
25.             HistoGram[Image_Use[j][i]]++; //统计灰度级中每个像素在整幅图像中的个数
26.         }
27.     }
28.
29.     //获取最小灰度的值
```

```
    for (MinValue = 0; MinValue < 256 && HistoGram[MinValue] <= 5; MinValue++)
    {
        Mincount += HistoGram[MinValue];
        HistoGram[MinValue] = 0;
    }
    //获取最大灰度的值
    for (MaxValue = 255; MaxValue > MinValue && HistoGram[MaxValue] <= 15; MaxValue--)
    {
        Maxcount += HistoGram[MaxValue];
        HistoGram[MaxValue] = 0;
    }
    //滤除反光点
    for (j = START_LINE; j < CAMERA_H; j++)
    {
        for (i = 1; i < CAMERA_W; i++)
        {
            if (Image_Use[j][i] > MaxValue - 8)
            {
                Image_Use[j][i] = MaxValue - 8;
            }
        }
    }

    HistoGram[MaxValue] += Maxcount;
    HistoGram[MinValue] += Mincount;
    if (MaxValue == MinValue)
    {
        return MaxValue; // 图像中只有一个颜色
    }

    if (MinValue + 1 == MaxValue)
    {
        return MinValue; // 图像中只有二个颜色
    }

```

```
65.     for (j = MinValue; j <= MaxValue; j++)
66.     {
67.         Amount += HistoGram[j]; //  像素总数
68.     }
69.
70.     PixelIntegral = 0;
71.     for (j = MinValue; j <= MaxValue; j++)
72.     {
73.         PixelIntegral += HistoGram[j] * j; //灰度值总数
74.     }
75.
76.     SigmaB = -1;
77.
78.     for (j = MinValue; j < MaxValue; j++)
79.     {
80.         PixelBack = PixelBack + HistoGram[j];
                    //前景像素点数
81.         PixelFore = Amount - PixelBack;
                    //背景像素点数
82.         OmegaBack = (double)PixelBack / Amount;
                    //前景像素百分比
83.         OmegaFore = (double)PixelFore / Amount;
                    //背景像素百分比
84.         PixelIntegralBack += HistoGram[j] * j;
                    //前景灰度值
85.         PixelIntegralFore = PixelIntegral - PixelIntegralBack;
                    //背景灰度值
86.         MicroBack = (double)PixelIntegralBack / PixelBack;
                    //前景灰度百分比
87.         MicroFore = (double)PixelIntegralFore / PixelFore;
                    //背景灰度百分比
88.         Sigma = OmegaBack * OmegaFore * (MicroBack - MicroFore) * (MicroBack
    - MicroFore); //计算类间方差
89.         if (Sigma > SigmaB)
                    //遍历最大的类间方差 g //找出最大类间方差以及对应的阈值
90.         {
91.             SigmaB = Sigma;
```

```
92.             Threshold = j;
93.         }
94.     }
95.     return Threshold; //返回最佳阈值;
96. }
```

2 转向控制策略

```
1.  void Turn_Cam_dias(void)
2.  {
3.    float temp;
4.    static float car_straight_dias_old;
5.
6.    car_straight_dias = M_Slope_fig() * SERVO_DIVIDE_ANGLE_SCALE;
7.
8.    Straight_offset_filter();
9.    car_center_dias = car_center();
10.   Center_offset_filter();
11.   if(Road == 0 && Road0_flag == 0 && Road0_flag0_flag && fabs(car_straight_d
    ias - car_straight_dias_old) < 30 )
12.   {
13.     temp = car_straight_dias + PID_CAR_STRAIGHT_CAM.D * (car_straight_dias -
     car_straight_dias_old);
14.   }
15.   else
16.   {
17.     temp = car_straight_dias;
18.   }
19.
20.   car_straight_dias_old = car_straight_dias;
21.   car_straight_dias = temp;
22.   if (fabs(car_center_dias) < 10)
23.   {
24.     car_center_dias = 0;
25.   }
26. }
```