浙江大学

探究性实验项目总结报告

课程名称　智能控制系统设计与实践

实验项目　基于摄像头和电磁传感器的智能循迹控制系统设计

姓　　名　葛明阳

学　　号　3170105814

专　　业　电气工程及其自动化

时　　间　2020 年 08 月 30 日

一、 实验目的和要求

1. 了解车模构成，掌握车模组装方法；
2. 掌握 MCU 的驱动和软件开发；
3. 掌握常用传感器的驱动方式，如摄像头、编码器、陀螺仪等；
4. 掌握 PCB 的绘制和电路板的焊接、调试；
5. 掌握图像校正、提取黑线、赛道元素识别的方法；
6. 掌握模糊控制、PID 控制的原理和参数调试方法。

二、 实验内容

2.1 控制器选型

方案一：采用 LPC54606 芯片。该芯片采用 Cortex-M4 内核，主频 180MHz，SRAM 为 200K，Flash 为 512K，对于小车图像处理算法性能不足。

方案二：采用 i.MX RT1064 芯片。该芯片是 i.MX RT 系列，由 NXP 半导体公司推出，是一款跨界处理器芯片。该系列下又包括 i.MX RT1020、i.MX RT1050 及 i.MX RT1060 等子系列芯片。该芯片采用了微控制器的内核 Cortex-M7，对中断响应更快，具有良好的实时性，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。其主频高达 600M，且板载 4M 大容量片上 Flash 和 512KB 的 RAM，满足程序开发的需求。

因此最终选择方案二，采用逐飞 RT1064 核心板。

2.2 摄像头选型

方案一：采用 OV7725 二值化摄像头。虽然其数据量小，易于单片机处理，但由于竞赛中摄像头的高度要求 10cm，该摄像头已经不再满足新的赛道条件的需求。

方案二：采用 MT9V032 灰度摄像头。其拥有 140°广角视域，帧率高。因此图像更新速度快，采集信息量大，满足比赛需求。

因此结合上述分析，最终选择方案二，摄像头采用总钻风 MT9V032，帧率 150FPS。

在实际应用时，摄像头帧率设置过高会导致图像变暗，且过高的帧率会超出程序控制周期，即图像更新的信息在控制时可能不被采用。此时可通过配置摄像头增益等参数对图像进行优化，根据实验效果选择较优的配置。

2.3 屏幕选型

方案一： 采用 OLED 显示屏。传输数据量小，实时性高。但分辨率较低，尽可显示二值化图像。

方案二： 采用 IPS 显示屏。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 延时，图像处理程序仅为 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。

因此最终选择方案二，采用 IPS 显示屏。

2.4 智能车系统整体方案

1. 采用 MT9V032 摄像头，以 150fps 的帧率拍摄赛道图像。图像数据通过 DMA 自动传输（场中断传输和行中断传输）给 RT1064，然后运行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，并判断赛道类型。
2. 由规划好的路径，选择图像合适位置的参考行，计算出偏差，用模糊 PD 控制舵机转向；当进入坡道时，由于拍摄到赛道外图像，故切出摄像头状态，使小车保持直行，离开坡道恢复。
3. 通过编码器检测车速，RT1064 输入捕捉功能进行脉冲计数，2ms 的 pit 中断测量。通过脉冲计数及时间，可计算得到小车的实时速度。通过路径偏差计算出预期速度，并利用 PI 控制以及 bang-bang 控制算法调节电机输出，从而实现小车速度控制。
4. 采用陀螺仪进行坡道辅助判断。记小车车头朝向为 x 轴正方向，通过 I2C 协议向陀螺仪 ICM20602 的相关寄存器中实时读取当前 y 方向的角速度值，以获得小车的翻转角速度，从而判断当前是否正在经过坡道或者颠簸。由于小车撞到路肩时同样可能出现翻转角，因此加入防误判算法。
5. 向 UART 转 Wi-Fi 模块以 150000 的波特率发送串口数据。模块将数据传输回上位机，实现实时监控，从而获得小车运行过程中的各项观察指标，便于后续参数调节。
6. 采用拨码开关、按键以及 IPS 显示屏辅助进行控制策略选择和参数设置。

系统框图如下：

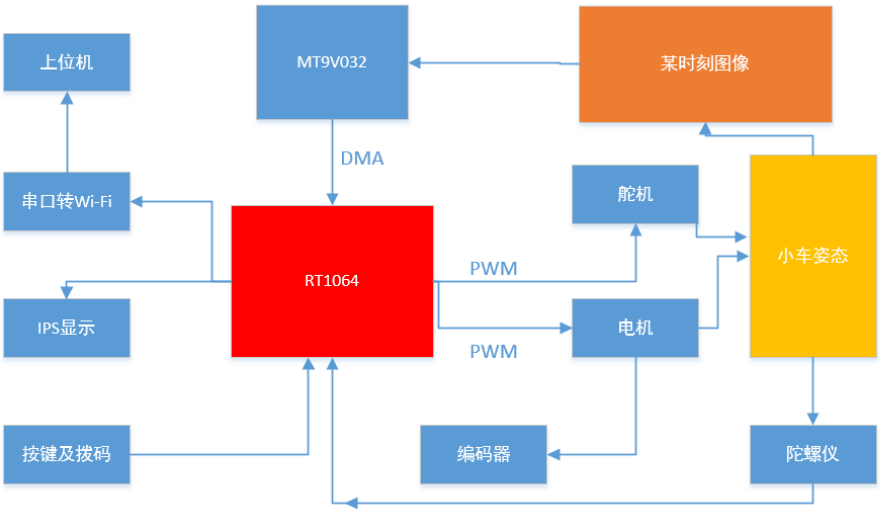

图 1 系统框图

三、 技术要点

第一节 图像信息获取

1.1 数字摄像头的使用

摄像头采用逐飞的 MT9V032，通过一个 51 单片机配置，通过 MCU 的串口传给 51 数据即可进行配置，可以对帧率、自动曝光、增益等进行配置。经过实践，二值化图或 IPS 原图的图像质量均符合要求。

1.1.1 数字摄像头介绍

- MT9V032 的优势有：
  - 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；
  - 高动态性能，90DB+的高动态范围。
  - FPS 可调，可自动曝光、曝光时间可调，不需要自己写驱动程序，也可通过取位的方式实现二值化。
- 图片大小、分辨率的选择

摄像头的大小和分辨率是 752*480，可以通过将两个或者四个像素点并成一个的方式，将图像进行降采样，以加快计算速度。经过裁剪，可以最终将图像变成 80*60 的图像，而视域基本不变，只需相应处理畸变、配合舵机控制程

序进行一定调整，使用起来相比 7725 等固定分辨率相机更加灵活

- 采用 140 度无畸变广角镜头

由于环岛等赛道元素的存在，且摄像头高度逐年的降低，因此采用广角的摄像头获取更宽的视野。但更宽的摄像头也存在图像过远和入弯补线错误等情况，需要同时调整高度和程序，以达到理想的效果。

1.1.2 数字摄像头的使用

数字摄像头用 SCCB 协议对摄像头的寄存器进行配置，单片机需要配置好中断、DMA 传输，自动获取摄像头图像。

1) 摄像头硬件连接

9V032 摄像头模块是 3.3V 供电，不能和 MCU 等用一个 3.3V 稳压芯片供电，应单独使用一路 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

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。在摄像头初始化的时，可以用 SCCB 协议对摄像头的寄存器进行配置。

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。

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。

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。

1.2 图像二值化阈值选取

图像二值化有很多算法。这里主要介绍大津法及平均值法两个方法

1）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。

1.3 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

1.4 $$\mu = \omega_0\mu_0 + \omega_1\mu_1 \quad \text{(公式 1)}$$

1.5 $$g = \omega_0(\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2 \quad \text{(公式 2)}$$

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

1.7 $$g = \omega_0\omega_1(\mu_0 - \mu_1)^2 \quad \text{(公式 3)}$$

使得类间方差g最大的阈值T_0'即为最佳分割阈值。

2）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$T_0 = T_0' + T_{offset}$。

1.3 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），导致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。

1.8

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

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。

1.4 边缘检测

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。因此采用 Sobel 算子进行边缘检测。

索贝尔算子（Sobel operator）主要用作边缘检测，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：

1.9 $$S_x=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix},\ S_y=\begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$ (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

1.10 $$G_x=S_x*A\ ,\ G_y=S_y*A$$ (公式 6)

于是得到图像梯度矢量大小为

1.11 $$G=\sqrt{G_x^2+G_y^2}$$ (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。

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。

1.5 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(x, y)，以及其对应的灰度值$i_{(x,y)}$，记二值图像对应值为$p_{(x,y)}$，则有：

1.12 $$p_{(x,y)} = \begin{cases} 0 , i_{(x,y)} < T_0 \\ 1 , i_{(x,y)} \geq T_0 \end{cases}$$ (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$G_{(x,y)}$，则有：

1.13 $$p_{(x,y)} = \begin{cases} 0 , G_{(x,y)} > G_0 \\ 1 , G_{(x,y)} \leq G_0 \end{cases}$$ (公式 9)

最终得到更新的二值化图像$p_{(x,y)}$，并用于后续图像处理等。

1.6 边线识别

根据得到的二值化图像，检测黑白的跳变点以期望找到每一行的赛道边线位置，为了克服噪音的影响，采用不同严格度的边线检测方法。

当初步搜索的边线与前一行边线连续，且符合赛道畸变特征，则可以放宽判定条件，以避免条件过于严苛而丢失边线。否则，需要逐渐使用更加严格的条件以避免将噪音误判为边线情况的发生。

而对于圆环，在进入圆环之前的时刻。摄像头能同时看到前进方向的赛道及进圆环方向的赛道，故也需要对边线进行重搜，以保证搜到的边线是正确的路径。

1.7 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来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，即经过旋转、平移及投影变换，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，即 Zg= 0，故可得到如下方程：

1.14

$$\begin{cases} x_c = \dfrac{(d\cos\theta + h)X_G}{Y_G\sin\theta + d\cos\theta} \\ y_c = \dfrac{(d\cos\theta + h)Y_G - hd\sin\theta}{Y_G\sin\theta + d\cos\theta} \end{cases} \qquad \text{(公式 10)}$$

其中 X_G，Y_G 是世界坐标系，x_c，y_c 是摄像头坐标系。而求解该方程主要有三个参数：焦距 d，摄像头高度 h，摄像头俯仰角 θ。

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。

第二节 方向控制

小车方向控制由两个部分构成：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），保证小车沿赛道前进；其二，即使小车沿着赛道行进，若其与赛道中心线位置有所偏移，会影响小车后续状态的稳定性，故需要控制舵机使得小车向中间修正，使得小车处于赛道的正中间（以下简称车正）。

控制行，即用来计算转向偏差的参考行，表征用于方向控制的图像行数。速度较快时，控制行需要选取较远行，从而保证舵机角度控制提前。由于舵机额定频率为 50Hz，舵机响应的滞后需要用前瞻来进行补偿。

2.1 车直转向偏差值的计算

根据阿克曼转向模型，小车行进曲率半径等于前方中心线曲率半径的转角，可以控制角度使得小车前轮平均打角应与中心线切线平行。

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做为车直偏差。

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呈正相关的，因此该斜率可作为车直转向偏差。

2.2 车正转向偏差值的计算

车正转向偏差，即近景行中心线与小车中心位置的偏差。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，因此在转弯时，对固有偏差做偏置处理，以减小误差。

2.3 车直舵机的 PD 控制

由于车直偏差与舵机打角存在线性关系，所以直接采用比例控制。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，且调试方便。该方法极大的提升了小车变速行驶时的稳定性。

由于纯比例控制会导致小车在直路上超调较大，故通过微分作用消除超调及震荡。

2.4 车正舵机的 PD 控制

由于车正偏差与舵机打角为非线性关系，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。

第三节 其他

此外，还有机械设计、电路设计、方向控制、电磁信息获取、速度控制、其他传感器等其他技术要点，此处不再赘述。

四、 实验成果

- 此次比赛中，获得全国大学生智能汽车竞赛（浙江赛区）第七名的成绩。
- 此次比赛中，共历时 11 个月，起止时间：2019.09.20-2020.08.10
- 此次比赛中，共撰写 RT1064 嵌入式 C 语言代码 2w 行（修改的库函数除外）。最终输出 C 语言代码 1.1w 行（库函数除外）。
- 此次比赛中，共撰写 Matlab 仿真代码 1k 行，Simulink 仿真工程 2 份。
- 此次比赛中，共输出比赛日志 100 天共 1.2k 行 2w 字。
- 此次比赛中，共输出电路板原理图及 PCB 工程文件 8 组。共输出电路板 100 余块，成品电路 20 余块。
- 此次比赛中，共输出 1w 字技术报告 1 份。

五、 实验体会

- 我本次主要参与整体搭建，完成视觉识别、车道类型判断、方向控制等内容等。在图像处理方面，10cm 高度摄像头在带来新的挑战的同时，也激励我们实现了更多的创新。理论推导逆透视方案、反光点的去除、在噪音丰富的情况下做边缘检测等都让我对图像处理有了更深的理解。通过对小车图像模型与运动模型的建模和实际操作观察，提高了数学建模的能力。同样，通过 Matlab 与 Simulink 仿真，也验证了逆透视、转向方案等多种可行性，从而选择较优的解决方案，加快了比赛准备进程。通过系统辨识的方

案，也加快了系统参数的整定。总而言之，这次实践中，自己对于智能控制系统的设计与实践能力得到了很大的提升。

- 在实践的过程中，虽然我们各自负责不同的模块，但车辆的整体调试是一个系统工作，各个控制器、常量参数都会对小车动态产生影响，本次实验深度加强了我们对经典控制理论、对模糊控制等控制方法的实践认识，同时在此基础上对嵌入式软件设计又有了新的理解。

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

```
uint8_t GetOSTU(void)
{
    int16_t i, j;
    uint32_t Amount = 0;
    uint32_t PixelBack = 0;
    uint32_t PixelIntegralBack = 0;
    uint32_t PixelIntegral = 0;
    int32_t PixelIntegralFore = 0;
    int32_t PixelFore = 0;
    double OmegaBack, OmegaFore, MicroBack, MicroFore, SigmaB, Sigma; // 类间方差;
    int16_t MinValue, MaxValue;
    uint8_t Threshold = 0;
    uint16 HistoGram[256]; //
    int16_t Mincount = 0, Maxcount = 0;

    for (j = 0; j < 256; j++)
    {
        HistoGram[j] = 0; //初始化灰度直方图
    }

    for (j = START_LINE; j < CAMERA_H; j++)
    {
        for (i = 0; i < CAMERA_W; i++)
        {
            HistoGram[Image_Use[j][i]]++; //统计灰度级中每个像素在整幅图像中的个数
        }
    }
```

```
28.
29.     //获取最小灰度的值
30.     for (MinValue = 0; MinValue < 256 && HistoGram[MinValue] <= 5; MinValue+
    +)
31.     {
32.         Mincount += HistoGram[MinValue];
33.         HistoGram[MinValue] = 0;
34.     }
35.     //获取最大灰度的值
36.     for (MaxValue = 255; MaxValue > MinValue && HistoGram[MaxValue] <= 15; M
    axValue--)
37.     {
38.         Maxcount += HistoGram[MaxValue];
39.         HistoGram[MaxValue] = 0;
40.     }
41.     //滤除反光点
42.     for (j = START_LINE; j < CAMERA_H; j++)
43.     {
44.         for (i = 1; i < CAMERA_W; i++)
45.         {
46.             if (Image_Use[j][i] > MaxValue - 8)
47.             {
48.                 Image_Use[j][i] = MaxValue - 8;
49.             }
50.         }
51.     }
52.
53.     HistoGram[MaxValue] += Maxcount;
54.     HistoGram[MinValue] += Mincount;
55.     if (MaxValue == MinValue)
56.     {
57.         return MaxValue; // 图像中只有一个颜色
58.     }
59.
60.     if (MinValue + 1 == MaxValue)
```

```
61.     {
62.         return MinValue; // 图像中只有二个颜色
63.     }
64.
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_dias     -
    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. }
```