第 1 章 电机控制

1.1. 直流电机控制策略：

针对本文所研究的智能车来说，车体速度是大惯性的被控对象。算法输出的控制量只能对电机输出力进行控制。而有一定负载时电机的输出力无论对车轮的转速还是车体的形式速度都是不成正比的，车在刚开始启动的时候速度是零，而电机的输出可能很大；车在匀速行驶的时候速度很快，而电机的输出可能并不是很大。而且电池电量、车体重量都会对车速造成影响。因此只有用闭环才能对车速进行良好的控制。在车轮对地面不打滑的情况下车体的速度和后轮的转动速度是成正比的。因此我们可以直接用光电码盘对后轮的转速进行控制。

对于这样一个大惯性系统，我们选用 **PID 和鲁棒**相结合的办法进行速度控制。回路的设定值由经验值确定。考虑到速度控制通道的时间滞后比较小，因此采用 PID 控制方案，并在进行加减速控制时，引入了“棒棒控制”。

$$U(k+1) = U(K) + P1 * e(k) + P2 * (e(k) - e(k-1)) + P3 * ((e(k) - e(k-1) - (e(k-1) - e(k-2)));$$

公式 1 1 PID 的公式

其中第一项为积分项；第二项为比例项；第三项为微分项。考虑到被控对象（车体速度）本身是一个大的积分环节，公式中可以将第一项省略，即采用 PD 控制。E 为误差。

同时设定误差门限，在误差比较大的时候采用大输出控制电机，将误差在最短时间内减小到所要求的范围，这就是**鲁棒控制**的思想。

电机控制策略，其实就是对模型车速度的控制策略，它是继路径识别之后非常重要的策略，直接关系到整个模型车比赛的性能。速度控制得当，小车才能以最好的状态，在最短的时间完成比赛路程。

1.1.1. 弯道速度控制

在模型车入弯时刻出于稳定性考虑做减速控制。减速原则是在原来直道速度设定值的基础上，减小速度设定值到低速挡，保证模型车可以安全入弯。另一方面，入弯之后，为了过弯时模型车能够不明显的左右摆动并采用较好的姿态通过弯路，

令车速与偏差成线性关系，偏差越大速度设定值越小。

速度设定方法如公式 1 2。

$$v_s(k) = v - k_1 * |e(k)| \quad \text{公式 1 2}$$

其中：$v_s(k)$ 为速度闭环设定值；

v 模型车全程运动平均速度设定值；

$e(k)$ 车体偏离理想轨迹的当前偏差值；

k_1 为减速控制比例系数。

同时，通过实验发现模型车入弯之后，令模型车以某一线性规律加速运行可以使车在不冲出轨道的前提下以更短的时间通过弯道。控制规律如.公式 1 3。

$$v_s(k) = v + Ck * \sqrt{Cp} \quad \text{公式 1 3}$$

其中：Ck 为弯道加速系数；

Cp 弯道加速变量；

Ck 为常数，初始化时设定。Cp 入弯时刻初始化为 0，每个控制周期累加 1。

1.1.2. 直道速度控制

直道采用匀速控制方案，速度设定在此控制系统下可以控制的最大入弯速度。在从弯道驶出进入直道后须进行加速控制，增大速度设定值。因速度控制回路有一定的惯性，所以在提速时为了使速度迅速改变，加入了“棒棒控制”。控制规律如公式 1 4。

$$v_s(k) = v + 100 - K_a * \sqrt{k_p} \quad \text{公式 1 4}$$

其中：K_a 补偿效果控制系数；

K_p 调速补偿变量。

K_a 为常数，初始化时设定。K_p 出弯时刻初始化为 0，每个控制周期累加 1。

1.1.3. PID 闭环控制算法

根据路径识别的情况，如果当前路径为直道，则需要加速；若是弯道，则需要降速，而且根据不同的弯道速度也是有所区别。系统利用测速模块反馈的当前速度值，通过增量式 PID 算法进行调节，从而控制直流电机对当前路径进行快速反应。

增量式 PID 的算式为：

$$\Delta u(k) = K_P\left[e(k) - e(k-1)\right] + K_I \cdot e(k) + K_D\left[e(k) - 2e(k-1) + e(k-2)\right]$$

在增量式 PID 处理的过程中，有一个步骤需要注意，即在算完△u (k)后，需要把它赋值给电机控制对应的 PWM 通道信号 ，这时要判断该△u (k)的值，如果它小于 0，则把 PWM 信号赋值为 0，如果它大于 PWM 信号的最大值，即大于 PWM 整个周期所对应的数值时，要把 PWM 信号赋值为该最大值。然后，再存储本次△u (k)，和上次△u (k)。

```
error=speed_v-infrared_value7;
pwmtemp=PWMDTY23+PID_P*(error-last_error)+PID_I*(error)+PID_D*(error+pre_error-2*last_error);
```

由上述代码中，speed_v 表示标准速度(期望速度)，infrared_value7 表示 ATD1 转换的经公式 4.5 计算出的即时速度值，通过计算它们的差值 error，再利用增量 PID 控制算法计算出 pwmtemp，再对 pwmtemp 进行处理，最后 PID 程序处理的结果是给出一个 PWM 信号，而这个信号就是驱动电机的，当 error 大，即标准速度和即时速度相差的比较多时，pwmtemp 的绝对值相对比较大，则给电机的 PWM 信号相对较大，这样电机转速相对较快；反之，电机转速则相对较慢。由此可以看出， PID 算法主要的功能是，在闭环系统中，利用即时速度的反馈，使得即时速度逼近标准速度的时间尽量变短，这样小车就可以根据路径识别得出的速度标准值，及时调整自己的速度，以适应各种路况。例如，小车正在直道上行进，而且直道的时速应该是各种赛道中最快的，当系统的路径识别算法察觉到前方出现弧度大的弯道后，系统会根据事先调试的结果，给出大弧度弯道的速度标准值，然后 PID 算法根据速度标准值 speed_v 和即时速度反馈值 infrared_value7 对电机的输入信号 PWM 进行及时调整，反映到实际中，就是小车及时减速并顺利通过弯道。

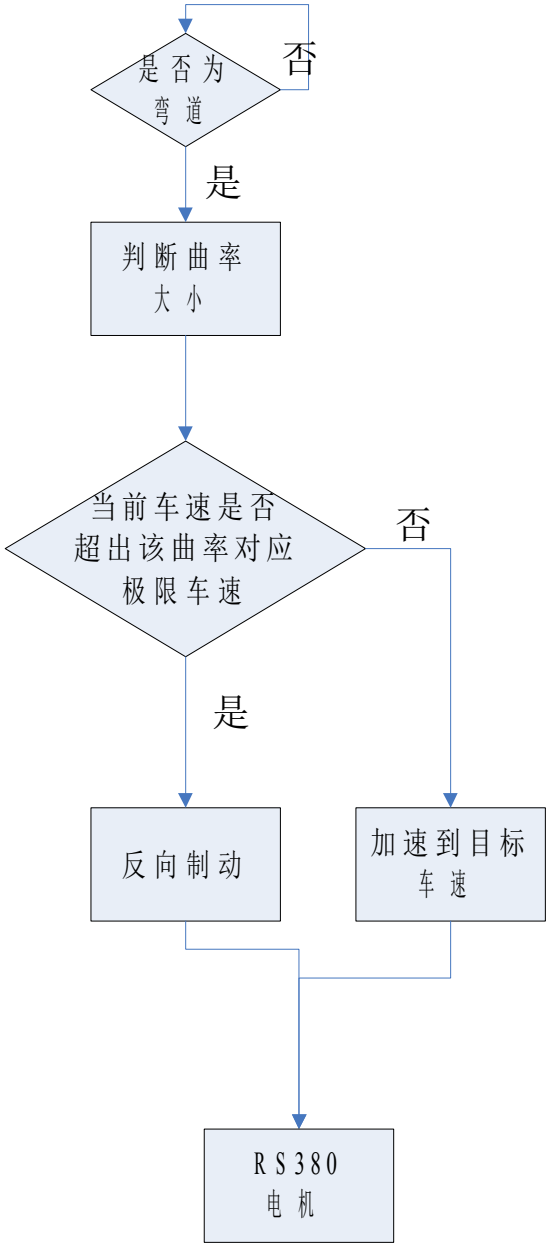
在增量式 PID 处理的过程中，有一个步骤需要注意，即在算完△u (k)后，需要把它赋值给电机控制对应的 PWM 通道信号，这时要判断该△u (k)的值，如果它小于 0，则把 PWM 信号赋值为 0，如果它大于 PWM 信号的最大值，即大于 PWM 整个周期所对应的数值时，要把 PWM 信号赋值为该最大值。

1.2. 反向制动算法

要使赛车在较短的时间完成比赛，速度自然越高越好，显然速度太高弯道是过不去的，如果以弯道的极限速度匀速跑，又浪费了直道的时间。所以最佳的策略是直道以较高的速度跑，到弯道时再尽快将速度降下来。在入弯减速时如果只靠赛道

的摩擦阻力效果显然是不够理想的。为此我们引入反向制动算法。

由于MC33886芯片集成的H桥驱动电路本身就具有反向制动功能，所以不需再外加硬件电路。反向制动流程图如图：



在制动时，电机反向电动势对整个系统电路造成的冲击，从而引起单片机工作不稳定，电流过大导致电机过热、电机频繁换向导致电刷打火剧烈导致电机寿命缩短等一系列问题都是需要考虑的。

1.3.速度控制策略

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

定，道路越接近直道，设定速度越高，反之越低。

车模行驶中的最低速度是这样确定的：令车模以较低的速度匀速行使，在保证无犯规行为出现的前提下，逐渐提高匀速行使的速度，直到车模出现犯规行为，此速度再减去一个安全量，即为所需的最低速度。简单的说，变速行使的最低速度等于匀速行使的最高速度。

车模行驶中的最高速度是这样确定的：在确定最低速度以后，加入变速策略，不断提高最高速度的设定值，直到模型车出现犯规行为，此速度再减去一个安全量，即为所需的最高速度。

车模行驶过程中难免出现“失去道路”的情况，对此需要采取一定的安全策略，防止赛车“盲跑”而导致犯规。

2、针对本文所研究的智能车来说，车体速度是大惯性的被控对象。算法输出的控制量只能对电机输出力进行控制。而有一定负载时电机的输出力无论对车轮的转速还是车体的形式速度都是不成正比的，车在刚开始启动的时候速度是零，而电机的输出可能很大；车在匀速行驶的时候速度很快，而电机的输出可能并不是很大。而且电池电量、车体重量都会对车速造成影响。因此只有用闭环才能对车速进行良好的控制。在车轮对地面不打滑的情况下车体的速度和后轮的转动速度是成正比的。因此我们可以直接用光电码盘对后轮的转速进行控制。

对于这样一个大惯性系统，我们选用 PID 和棒棒相结合的办法进行速度控制。回路的设定值由经验值确定。考虑到速度控制通道的时间滞后比较小，因此采用 PID 控制方案，并在进行加减速控制时，引入了“棒棒控制”。

$$U(k+1) = U(K) + P1 * e(k) + P2 * (e(k) - e(k-1)) + P3 * ((e(k) - e(k-1) - (e(k-1) - e(k-2)));$$

公式 4 3 PID 的公式

其中第一项为积分项；第二项为比例项；第三项为微分项。考虑到被控对象（车体速度）本身是一个大的积分环节，公式中可以将第一项省略，即采用 PD 控制。E 为误差。同时设定误差门限，在误差比较大的时候采用大输出控制电机，将误差在最短时间内减小到所要求的范围，这就是棒棒控制的思想。

电机控制策略，其实就是对模型车速度的控制策略，它是继路径识别之后非常重要的策略，直接关系到整个模型车比赛的性能。速度控制得当，小车才能以最好的状态，在最短的时间完成比赛路程。

1.3.1. 弯道速度控制

在模型车入弯时刻出于稳定性考虑做减速控制。减速原则是在原来直道速度设定值的基础上，减小速度设定值到低速挡，保证模型车可以安全入弯。另一方面，入弯之后，为了过弯时模型车能够不明显的左右摆动并采用较好的姿态通过弯路，令车速与偏差成线性关系，偏差越大速度设定值越小。速度设定方法如……公式 4 4。

……公式 4 4

其中： 为速度闭环设定值；

模型车全程运动平均速度设定值；

车体偏离理想轨迹的当前偏差值；为减速控制比例系数同时，通过实验发现模型车入弯之后，令模型车以某一线性规律加速运行可以使车在不冲出轨道的前提下以更短的时间通过弯道。控制规律如.公式 4 5。

……公式 4 5

其中： 为弯道加速系数；

弯道加速变量；

为常数，初始化时设定。 入弯时刻初始化为 0，每个控制周期累加 1。

1.3.2. 直道速度控制

直道采用匀速控制方案，速度设定在此控制系统下可以控制的最大入弯速度。在从弯道驶出进入直道后须进行加速控制，增大速度设定值。因速度控制回路有一定的惯性，所以在提速时为了使速度迅速改变，加入了“棒棒控制”。控制规律如……公式 4 6。

……公式 4 6

其中： 补偿效果控制系数；

调速补偿变量。

为常数，初始化时设定。 出弯时刻初始化为 ，每个控制周期累加 。

1.3.3. 速度及加速度信号采集子程序

当图像数据采集处理完后，我们读取速度传感器和加速度传感器的值，采集这两个传感器信号的频率也为 60Hz。流程图如图 4.3 所示。

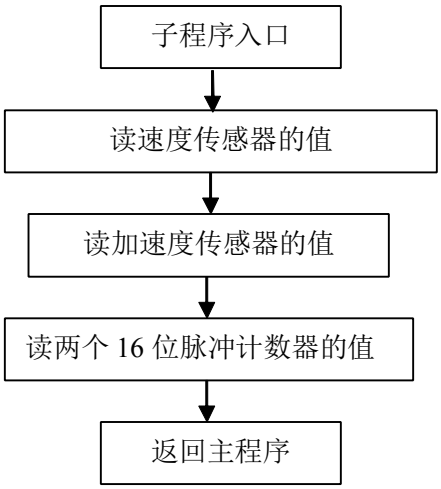


图 4.3 速度及加速度信号采集子程序流程图

1.4. 速度 PID 算法

根据偏差的比例（Proportional）、积分（Integral）、微分（Derivative）的线性组合进行反馈控制（简称 PID 控制），数字 PID 控制算法是电机微机控制中常用的一种基本控制算法[9]。

在连续系统中，模拟 PID 调节器是一种线性调节器，控制系统原理框图如图 5.10。

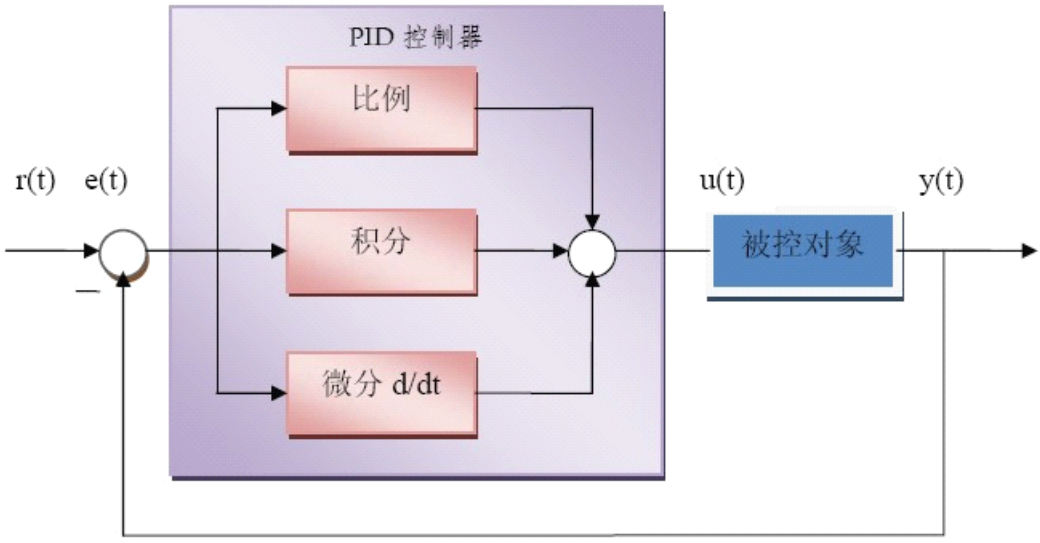


图 5.10 PID 控制系统原理框图

控制规律为：

$$u(t)=K_P[e(t)+\frac{1}{T_I}\int_0^t e(t)dt+T_D\frac{de(t)}{dt}] \quad (5.1)$$

式中：K_p：比例增益，K_p的倒数称为比例带；T_I：积分时间常数；T_D：微分时间常数；$u(t)$：控制量；$e(t)$：偏差，等于给定量与反馈量的差。

在计算机控制系统中，数字 PID 控制算法通常又分为位置式 PID 和增量式 PID。

本次设计中，我们采用增量式 PID。

增量型算法与位置型算法相比，具有以下优点：增量型算法不需要做累加，增量的确定仅与最近几次偏差采样值有关，计算精度对控制量的计算影响较小，而位置型算法要用到过去偏差的累加值，容易产生大的累加误差；增量型算法得出的是控制量的增量，而位置型算法的输出是控制量的全量输出，误动作影响大；采用增量型算法，易于实现手动到自动的无冲击切换。

在实际应用中，采样的反馈值 $y(k)$ 即为脉冲累加器中的 PACN32 中的脉冲数，预设门限值 A 在参数整定时根据实际情况调节，输出 $u(k)$ 并不能直接用来控制电机，需要将其转换为控制 PWM 占空比，然后用增大或减小 PWM 占空比的方法来实现对电机的加减速的控制。换句话说，在求偏差量时，实际上用的是每 20ms 电机转过的齿轮数和实际期望电机转过的齿轮数，通过二者的差值，再乘以相应的系数，即 K_P、K_I、K_D 的协调控制，计算出相应的 PWM 占空比，实际上用的是 PWMDTY 的值[10]。

本设计中综合考虑各种因素，最后选用的采样周期为 20ms，即每 20ms 对电机进行一次 PID 调节。由于在程序中，对图像的采集使用的是 PH 口的中断程序，因此，PID 采样周期的选择实际上是受限制与图像采集，因为每行的扫描周期为 64μs，有效扫描时间为 52μs，采用的是隔行扫描的方式，即每隔 6 行采集一行图像的信息，如果在每行之间加入 PID 调节的话，那么处理 PID 子程序的时间必须控制在 64*5＝320μs 之内，另外图像采集只是采集了奇场中的行数，在偶场中没有采集，因此 PID 子程序的执行是不均匀的，并没有达到预期的效果，同时还可能会影响到视频采集，因此，经过分析，最终决定将 PID 的采样周期定为 20ms，即当进行一次场采集进行一次 PID 调节。而且经过最终的检验，这样能够满足对速度控制的需要。

电机调速

本作品采用增量式的 PI 控制。

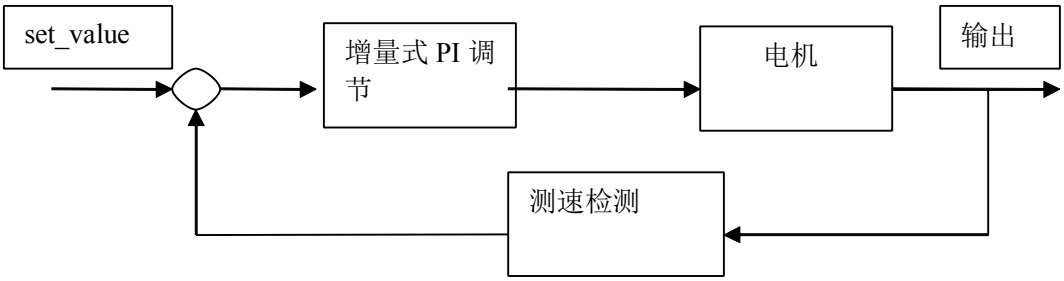


图 4.12

增量式 PI 算法:

pwm=Kp*(errork0-errork1)/100+Ki*errork0/100+pwm;

1.5.直流驱动电机控制子程序

本设计通过调整方波的占空比来实现电机的调速。经过初始化子程序的设置，PWM0 和 PWM5 都能输出频率为 5KHz 的方波。由硬件部分可知，PWM0 产生的方波经 MC33886 功率放大后接到电机的负极，PWM5 产生的方波经 MC33886 功率放大后接到电机的正极。车模向前行驶时，PWM0 输出方波的占空比为零，通过调整 PWM5 输出方波的占空比实现加速或缓慢减速。当需要快速减速时，PWM5 输出方波的占空比为零，PWM0 输出方波的占空比为合适的值，电机反转实现车模制动。该部分程序工作流程图如图 4.7 所示。

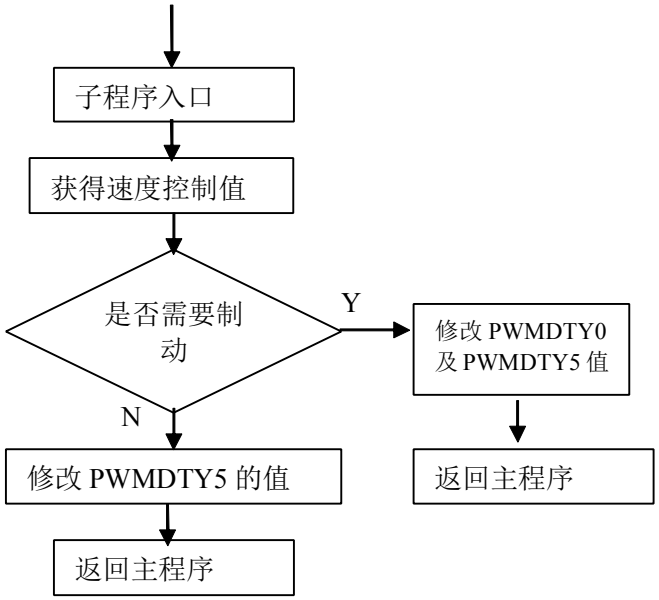


图 4.7 驱动电机控制子程序流程图

1.6. 舵机控制

1.6.1. 舵机转角的决策

由于 CCD 图像传感器的应用，大大提高了车的前瞻性。经测量，本系统预测距离可达 1 米以上。但是，考虑到各部件延时严重且具有很大不确定性，所以预测距离越远，控制算法就会越复杂。综合各种因素考虑，我们最终将预测距离定在 80~100cm。

黑线中心位置轨迹中包含着两方面的信息，一方面是反映前方道路趋势的信息，另一方面是反映车体与轨道偏离程度的信息。经过反复试验，我们发现，前者可以通过计算黑线的斜率来很好的描述。而后者，恰恰就是屏幕最下端黑线的位置。最后用这二者的线性组合得到舵机转角。有了这两方面信息，可以使智能车既能充分利用前瞻性，又能不过分地偏离轨道。从而得到了较理想的转角控制策略解决方案。

1.6.2. 偏航距离的计算

由于已经获得了赛道中心线的位置，所以计算偏航距离的问题是选取何处的中心线的距离为当前的偏航距离。控制算法的执行周期为 40ms，如果赛车的速度为 2m/s，则在两次控制算法的执行中间，赛车要前进 8cm，赛车所处的环境将发生比

较大的改变，所以赛车的控制只能算是半实时控制，这是所有使用摄像头作为主要寻线传感器的参赛队都避免不了的问题。因为算法的滞后性，赛车需要将“当前位置”进行适当前移。前移量应该跟赛车当前速度成正比，但实际中我们发现，适当增加一些前移距离是有好处的，因为可以在入弯处提前转弯，使得赛车沿弯道内侧行驶，缩短了过弯距离。

1.6.3. 舵机参数测定

通过实验测出已安装在模型车上的舵机的中位PWM值（Servo_center），模型车转向轮左右极限时舵机PWM（Servo_left、Servo_right）值是舵机控制的基础。

测试舵机中位时，先在舵机控制模块中给舵机PWM设计一个全局变量（Pwm_Servo），将赛车放在长直道上，保证车身放正并处于赛道正中间，利用BDM查看此时Pwm_Servo的值，将此值直接赋给舵机PWM，并将赛车放到直道上跑，看是否跑偏，微调Pwm_Servo的值，直到赛车在直道上不再跑偏为止。

测定舵机左右极限时，可根据转向轮的左右极限转角，利用BDM查看舵机Servo_left和Servo_right的值。注意转向轮在左右极限转角时，转向轮不可被赛车底盘卡住，这样不但会增加转向阻力，严重时还可能使转向轮抱死，失去转向能力。

舵机的PWM计算公式为：

Pwm_Servo=Servo_center+(Servo_left-Servo_right)/Simage_number*offline_one；……公式四

式中Simage_number为每行采集的点数，

offline_one为黑线在整幅图像中的位置。

1.6.4. 路径规划

要想以最短的时间完成比赛，并不需要严格按照黑色引导线行驶，我们需要找出相对较短的行驶路径，即路径规划。例如 过普通弯道的时候，以同样的速度走内弯显然比走中间或外弯所花时间少，过小S形弯时。直接过去和严格循迹相比速度快，行驶距离短等等。（图）

如果能提前预知前方赛道信息，那么我们就有可能对路径进行规划。由于我们采用的是摄像头，扫描距离较大，这就为我们进行路径规划提供了条件，这也是我们选择摄像头的原因之一。

由于摄像头看得较远，每幅图像反映的赛道信息量较大，我们需要针对赛车在不同位置，对其有侧重的采用。具体就是将采到的15行划分为几个区域，针对不同路况对每个区域赋以不同的权值。其原理图如图：

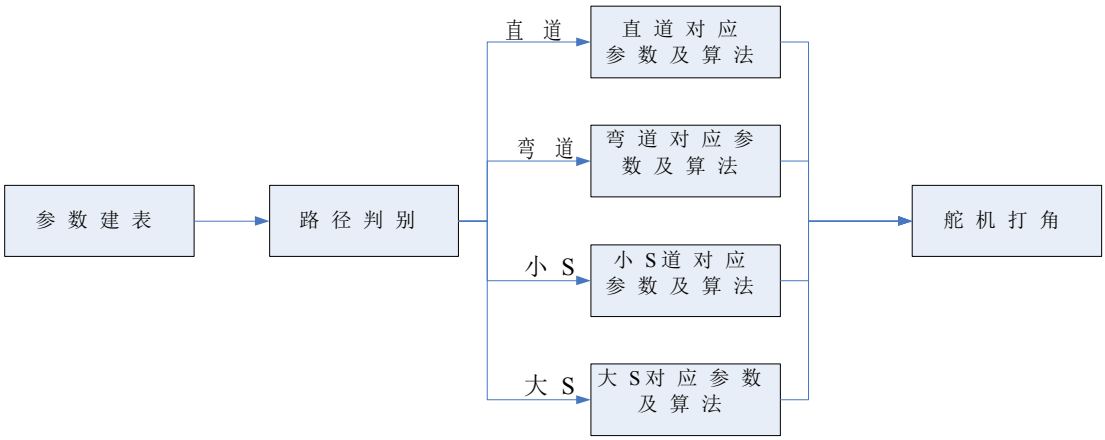


图5.10 路径规划示意图

1.7. 偏航距离的计算

由于已经获得了赛道中心线的位置，所以计算偏航距离的问题是选取何处的中心线的距离为当前的偏航距离。控制算法的执行周期为 40ms，如果赛车的速度为 2m/s，则在两次控制算法的执行中间，赛车要前进 8cm，赛车所处的环境将发生比较大的改变，所以赛车的控制只能算是半实时控制，这是所有使用摄像头作为主要寻线传感器的参赛队都避免不了的问题。因为算法的滞后性，赛车需要将“当前位置”进行适当前移。前移量应该跟赛车当前速度成正比，但实际中我们发现，适当

增加一些前移距离是有好处的，因为可以在入弯处提前转弯，使得赛车沿弯道内侧行驶，缩短了过弯距离。

1.7.1. 偏航角度的计算

计算偏航角度的实质是直线拟合问题，因为赛道中心线所在的直线确定了，而直线的斜率与偏航角度一一对应。直线拟合最有效的方法是最小二乘法[7]，但是直接应用存在一个问题，即如何确定进行直线拟合的区间？在整个成功识别出赛道的区间内进行直线拟合显然是欠缺考虑的，因为在弯道的情况下，这种方法拟合出的是一条弦线，而不是当前该弯道处的切线。摄像头视野越大，弯道曲率越大，弦线偏离切线的程度也就越大。为了能够在直道和弯道上都能正确的拟合出正确的直线，我们采用了直线检测的方法，即首先根据残差的大小确定直线的范围，然后在这一范围内进行直线拟合。

1.7.2. 曲率的计算

如果说斜率的计算需要某种技巧的话，计算曲率则更是一种技巧的应用。首届时很多参赛队针对各自的实际需要，提出了自己的方法。其中最普遍的是根据斜率的导数来计算曲率[8]。但是斜率的计算本身就很不准确，特别是某个点的斜率，对斜率求导就更不准确，所以使用这种方法只能得出一个大致的结果。本文作者提出了另外一种方法，首先对获得的路径进行滤波，使得路径尽可能平滑，然后取其两个端点和中间点，计算这 3 个点组成的三角形的外接圆的半径，半径的倒数就是这段路径的曲率。经过多次实验，这种方法的误差一般不大于 20%，对智能车的控制来说已经足够了。

图 5.8 是实验赛道的照片，让赛车从起跑线开始，在赛道上行驶一圈，记录下每个时刻的曲率，如图 5.9。从图 5.9 可以看出，计算出的曲率能较为正确的反映实际赛道的弯曲情况。但是这种方法得出的曲率不是摄像头所看到的当前位置的曲率，而是摄像头所看到的

路径的整体曲率，因为这种算法仅仅与路径中三个点的位置有关。然而实验中却发现，这种特点反而给赛车带来了一个好处：即摄像头在小S型弯道时舵机几乎不跟随路径的摆动而摆动，而是直线冲过。这是因为较小的S型路径其弯曲部分能完整的显现在摄像头的视野中，而算法中对路径进行了滤波，滤除了中间部分的弯曲，使得路径变直了，赛车因此就直线走过。

1.8. 车辆横向控制算法

本设计采用的横向控制算法是模拟人驾驶汽车的方式控制车模转向。首先，根据图像数据处理后的数据判断车模偏离黑线的距离。其次，综合考虑车速传感器获得的当前速度值，计算目标转角。然后，在下一次采样时，与上次采样值比较，判断偏离距离的变化趋势，再次计算目标转角，当车模回到中线位置时，前轮回正。该部分程序流程图如图 4. 4 所示。

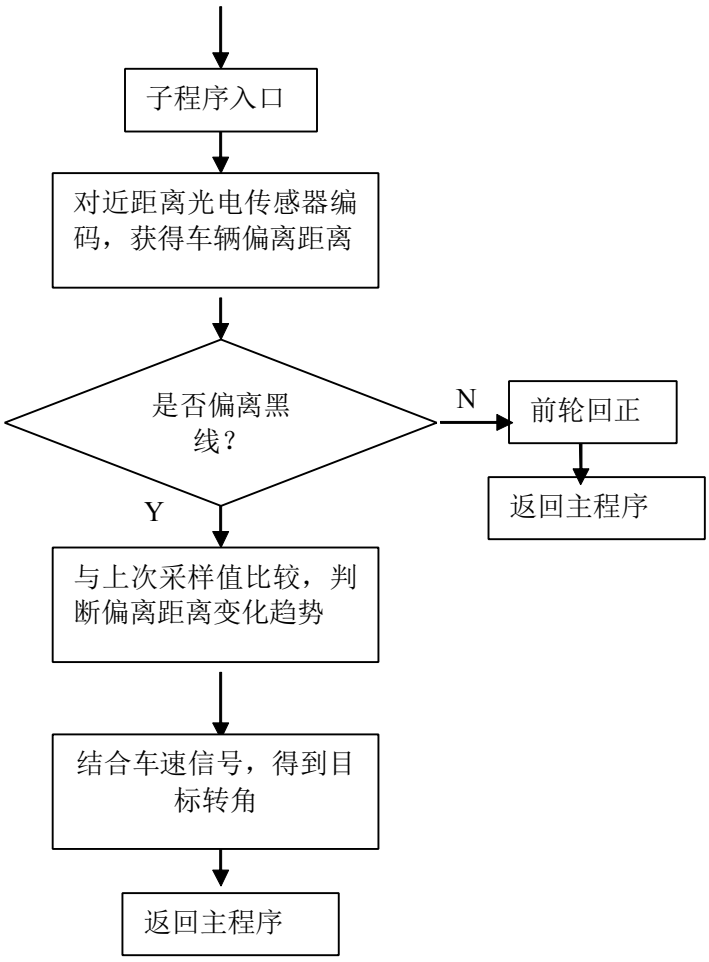


图 4.4 横向控制算法流程图

1.9.车辆纵向控制算法

本设计中模型车的速度控制采用 PID 控制算法。首先，图像数据处理后的数据判断车模当前行驶的路况，得到目标速度值。其次，根据车速度传感器的值，判断车速是否达到目标速度，进行加减速。第三，结合加速度传感器的值判断模型车是否发生滑移，如果滑移进行防滑处理。该部分程序流程图如图 4.5 所示。

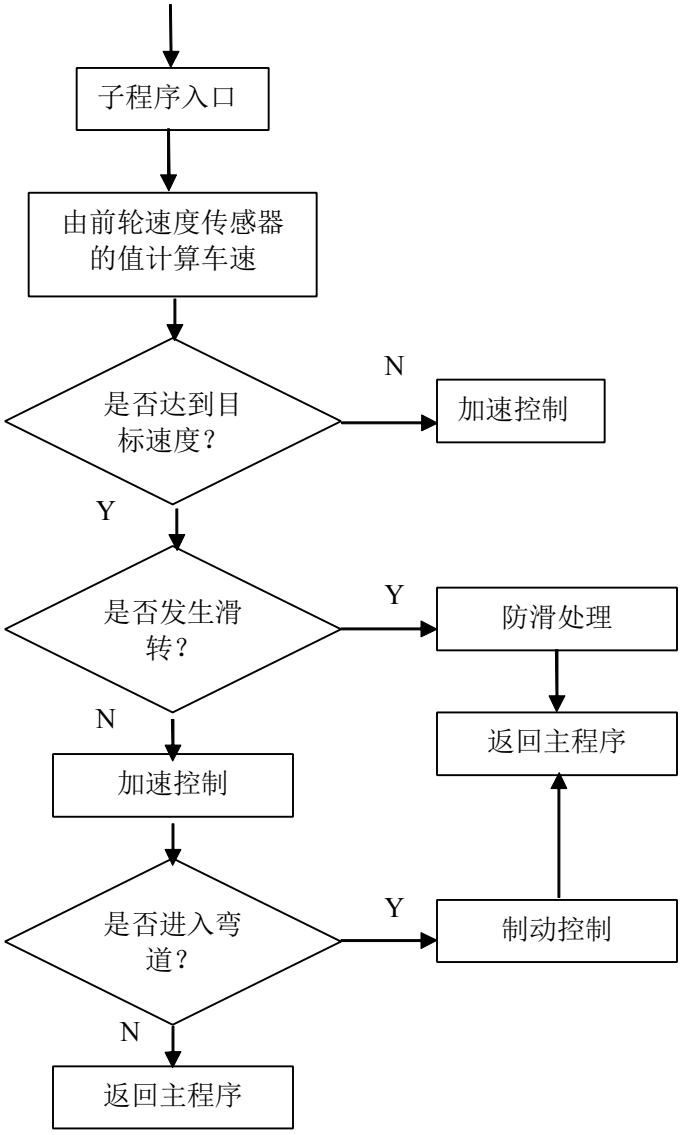


图 4.5 纵向控制算法流程图

1.10. 转向控制子程序

由于舵机反应速度的局限，本设计每 10ms 对舵机控制一次。可知 PWM23 已经在

初始化子程序中设置完毕，其能产生周期为 20ms 的方波，在此我们只需修改 PWMDTY23 寄存器即可改变方波的占空比，进而控制舵机转向。本车舵机的安装位置和 PWM23 通道的设置使得前轮在中间位置时，PWMDTY23 的值为 4960，左极限位置时值为 5300，右极限位置时为 4640，即将左极限位置到右极限位置的转向级别分为 660 级。为了防止舵机转向超过极限位置，发生卡死现象，损坏舵机，本设计通过软件限定极限转角。该子程序工作过程如图 4.6 所示。

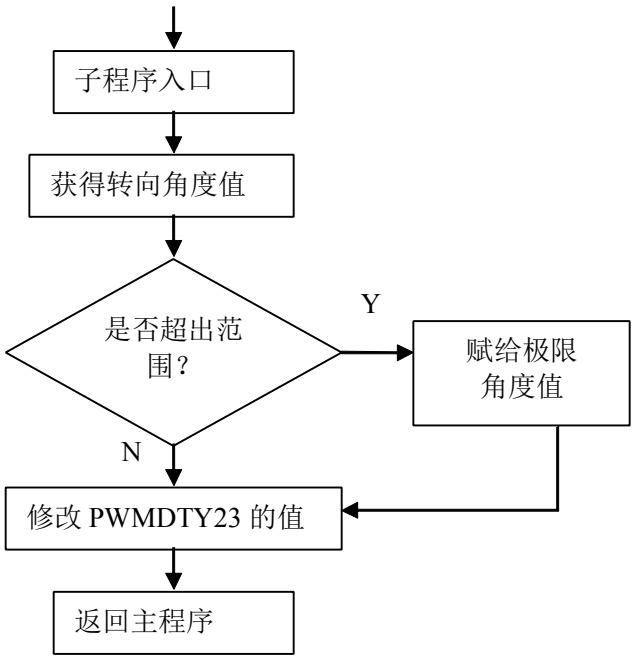


图 4.6 舵机控制子程序流程图

1.11. 车体控制策略：

对于硬件结果已经确定的智能车来说，算法控制的实际上就是车体的方向和速度，即前轮舵机的转角和后轮的转速。如果前面图像识别和虑噪环节做的好的话，对于控制量的给出的算法就相对容易，但是需要花很大精力进行参数的调整，使得车的行驶轨迹更加优化，并且让速度尽可能的提高。

对于前轮的控制来说，由于执行机构（舵机）和被控对象的惯性相对较小，换句话说算法给出的控制量在误差允许 的范围内可以认为由执行机构准确执行，可以进行开环的控制[2]。

我们的控制算法是在图像中分别取上、中、下三个点，即在 23 行的数组中取 3、12、21 三个点，计算出图像的位置，然后用最上面一点和最下面一点的差做为图像的斜率。然后用两者的线性组合算出舵机的转角。用下面公式给出：

$$Warp = P1 * (Average[2] + Average[11] + Average[20]) + P2 * (Average[20] – Average[2])$$

公式 1 5 舵机的转角公式

其中，warp 是车体对于黑线的误差，也是舵机应该转动的角度。

同时，我们发现在车运行过程当中经常会出现图像中只有半幅图像有线，或者整幅图像都没有黑线的情况。在半幅图像有线的情况下，我们就用图像的在视野中的终点、图像最下面的点及两者的中点进行控制；如果整幅图像都没有黑线，就让车的舵机保持上一次的转角，一直到图像重新恢复到视野。

对于前轮的控制来说，由于执行机构（舵机）和被控对象的惯性相对较小，换句话说算法给出的控制量在误差允许 的范围内可以认为由执行机构准确执行，可以进行开环的控制。

第 2 章　　图象采集

2.1.采集

1、一场图像采集完成后，得到一幅 20*20 的原始二值图像信息，如图 4.3（A）所示，同时给标志变量 VideoFinFlag 置 1。主程序中检测到此标志的时候，启动滤噪程序。考虑到图像中有效的信息为简单的黑色线条，必然具有很强的连续性。所以，可以采用从图像一侧到另一侧，逐行邻域查询的办法。具体来说，就是根据已经确定的前行黑线中心位置，在本行的一个邻域内查找黑点。超出这个邻域的黑点都认为是无效的。这样，只要能够确定某一行黑线的中心位置，就能逐行确定各行黑线的中心位置，从而，也就提取出了图像中最基本的信息——黑线中心位置轨迹（如图 4.3（B）所示）。

原始图像信息	处理后的信息
00000000000010000000	2
00000000000010000000	2
00000000000100000000	1
00000000000100000000	1
00000000001000000000	0
00011000001000000000	0
00010000001000000000	0
00000000011000000000	0
00000000011000000000	0
00000000110000000000	-1
00000000110000000000	-1
00000000111000000000	0
00000000111000000000	0
00000000111000000000	0
00000000111000000000	-1
00000000111000000000	-1
00000001110000010000	-2
00000001110000000000	-2
00000001110000000000	-2
00000011110000000000	-3
(A)	(B)

图 4.3　处理前后的图像信息

2、阈值分割法是一种基于区域的分割技术，它对物体与背景有较强对比的景物的分割特别有用。它计算简单，而且总能用封闭且连通的边界定义不交叠的区域。阈值分割法的关键在于阈值的确定。如果阈值是不随时间和空间而变的，称为静态阈值；如果阈值随时间或空间而变化，称为而动态阈值。基于静态阈值的分割方法算法简单，计算量小，但是适应性差。基于动态阈值的分割方法其复杂程度取决于动态阈值的计算方法[6]。针对本智能车系统，普通的双峰法就能适合绝大部分情况，因为智能车的运行环境是比赛赛道，背景和前景区分明显，且背景简约。但是实验环境并不理想，由于受到光线斜射的影响，有时背景和前景的对比十分不明显。结

合实际需要，提出了一种新的计算阈值的方法，这种方法的思想与迭代法有些相似。

首先我们假定，智能车系统运行时，开始时采集的第一幅图像是良好的。这个“良好”含义是：第一行也就是最近处的一行，完整的包含了导航线，并且使用“双峰法”能正确提取。其次，我们对每一行都定义一个阈值 Tr，每一行都用该行的阈值 Tr 进行分割。在此基础上按照以下规则进行阈值传递：

a) 如果第 r 行分割出来的黑色区域（线段）是连续的，那么计算目标区域和背景区域的平均灰度值，并取其中值，做为当前行和下一行的 Tr。

b) 如果第 r 行分割出来的黑色区域是空集或者不连续，则保持当前行和下一行的阈值不变。

下滤波算法：当某点的灰度值与其左边和右边点的灰度值的差同时大于某个正阈值或同时小于某个负阈值时，认为该点是干扰点，取其左右两边灰度值的平均值作为该点的灰度值。

2.2. 图像抗干扰处理

抗干扰处理包括以下方面的内容：

1）消除信号产生和传输的过程中造成的噪声；

2）消除赛道不理想或光线不均匀形成的干扰；

3）消除赛道中的交叉线、断续线；

4）识别起跑线。

信号产生和传输的过程中造成的噪声具有普遍性，但是只有当其非常严重时才对路径的识别造成影响。所以采取以下滤波算法：当某点的灰度值与其左边和右边点的灰度值的差同时大于某个正阈值或同时小于某个负阈值时，认为该点是干扰点，取其左右两边灰度值的平均值作为该点的灰度值。对于赛道不理想或光线不均匀形成的干扰，其一般特点是散布于赛道黑线的两边，所以采用以下算法处理：

步骤 1 从最近的第一行开始，寻找这样一个行：该行只有一段连续黑色线段，且线段的左右边均不是图像的左右边。标记该行为 Start 行。

步骤 2 在下一行中寻找这样一段连续黑色线段，满足这样的条件：其左边缘不在上一行黑线右边缘的右边，且其右边缘不在上一行黑线左边缘的左边。

步骤 3 如果存在步骤 2 中的黑色线段，认为该段黑线是赛道黑线，其余黑点（线段）均认为是干扰，跳步骤 2。否则，标记该行为 End 行，并计算该行与 Start 行之差，如果大于 3，则结束搜索，如果小于等于 3，则返回步骤 1。

这种算法的实质是搜索连续的黑线。因为赛道是连续的，而干扰却大部分是离散分布的。

对于赛道中的交叉线，可以统计一行中黑点的个数，因为正常的黑线只有 2.5cm，在图像中不超过 6 个点。所以把 6 作为一个阈值 TN，如果一行黑点的个数超过 TN 个点，就可以认为是交叉干扰。实际上，我们做了更复杂的处理，就是根据当前赛道的倾斜程度来调节 TN，因为当赛道倾斜时其宽度将发生变化。并且，我们不只是识别出交叉干扰，而且“砍掉”它的左右两支，保留一条连续的赛道黑线。

起跑线是在赛道黑线的左右两边，对称存在两条长度各为 10cm，相距 6cm，且与赛道黑线垂直的黑色线段。正确的识别起跑线是应用更高级控制算法如记忆算法的基础。我们使用如下算法识别起跑线：在已识别出赛道黑线的行，保证该行不是交叉行（前面已经识别出），然后确定其左右两边各一个矩形区域，统计两个区域中黑点的个数，如果均大于某一阈值，则判断该行出现了起跑线。

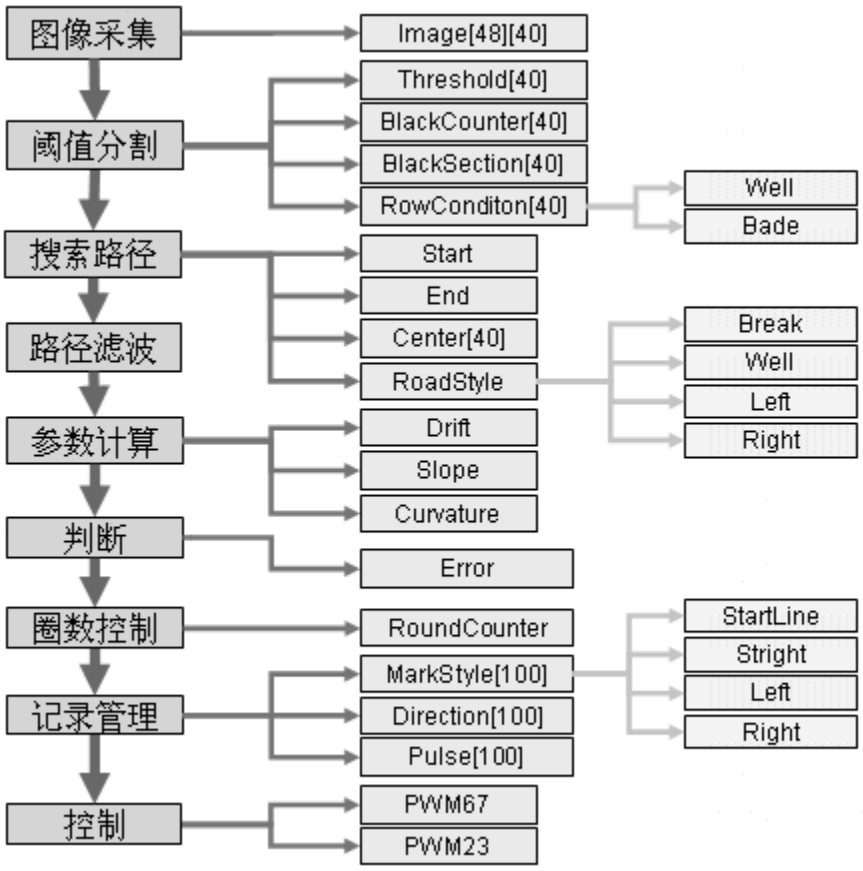
比赛中将设置限速段，限速段的一种可能的标记法就是把赛道黑线做成断续的，

断续宽度 2cm，间隔 2cm，所以并不是所有不连续的黑点都是干扰。这给参赛选手带来了很大的挑战。我们采取的应对算法是，在遇到断续的行时，按当前已识别出的黑线的走向趋势延伸黑线到该行，如果在 3 行内发现了可以连接的黑线，则认为延伸成功，否则结束搜索。

经过以上处理，我们得到了如下信息：

首先是两个标记 Start 和 End，这两个标记标记了成功识别出的赛道的最近一行和最远一行；其次是在 Start 和 End 之间的每一行，记录了其赛道的左边缘位置和右边缘位置，并且通过坐标变换算法，计算出每行赛道中心线的实际位置。最后留下了一些其它标志，比如每行是否是交叉、起跑线或者断续线。这些标志在我们调试的过程中都用某种方式以图像的形式反映出来。整个处理过程如图 5.6。如果说斜率的计算需要某种技巧的话，计算曲率则更是一种技巧的应用。首届时很多参赛队针对各自的实际需要，提出了自己的方法。其中最普遍的是根据斜率的导数来计算曲率[8]。但是斜率的计算本身就很不准确，特别是某个点的斜率，对斜率求导就更不准确，所以使用这种方法只能得出一个大致的结果。本文作者提出了另外一种方法，首先对获得的路径进行滤波，使得路径尽可能平滑，然后取其两个端点和中间点，计算这 3 个点组成的三角形的外接圆的半径，半径的倒数就是这段路径的曲率。经过多次实验，这种方法的误差一般不大于 20%，对智能车的控制来说已经足够了。

图 5.8 是实验赛道的照片，让赛车从起跑线开始，在赛道上行驶一圈，记录下每个时刻的曲率，如图 5.9。从图 5.9 可以看出，计算出的曲率能较为正确的反映实际赛道的弯曲情况。但是这种方法得出的曲率不是摄像头所看到的当前位置的曲率，而是摄像头所看到的路径的整体曲率，因为这种算法仅仅与路径中三个点的位置有关。然而实验中却发现，这种特点反而给赛车带来了一个好处：即摄像头在小 S 型弯道时舵机几乎不跟随路径的摆动而摆动，而是直线冲过。这是因为较小的 S 型路径其弯曲部分能完整的显现在摄像头的视野中，而算法中对路径进行了滤波，滤除了中间部分的弯曲，使得路径变直了，赛车因此就直线走过。



2.3. 图像滤波

找出二值化以后的图像每一行中的黑色区域，如果只有一个黑色区域，则记下黑色区域起始点对应的列号和结束点对应的列号，求平均值即为黑线中心，如果黑色区域大于或者等于两个该行图像信息有错，则利用该行相邻行找出正确的黑线位置。 如果连续几行都为错，则放弃这几行，直道找到正确行为止，再将刚才放弃的部分黑线补上。

2.4. 梯形失真校正

梯形失真会导致路进识别不准甚至出错，我们对梯形失真进行了简单校正。我们采用比例法来矫正梯形失真。由于摄像头视野最远处的宽度和最近出的宽度显然是不一样的，而摄像头的行分辨率是相同的，这就导致两个采样点之间的实际距离是不一样的，而单片机却认为他们是一样的，这也是导致梯形失真的原因。我们测量出摄像头扫描赛道最近处和最远处视野宽度，算出相邻两个采样点所代表的实际距离，再量出视野长度，中间各行相邻两采样点之间的距离根据其行号按比例计算出来。

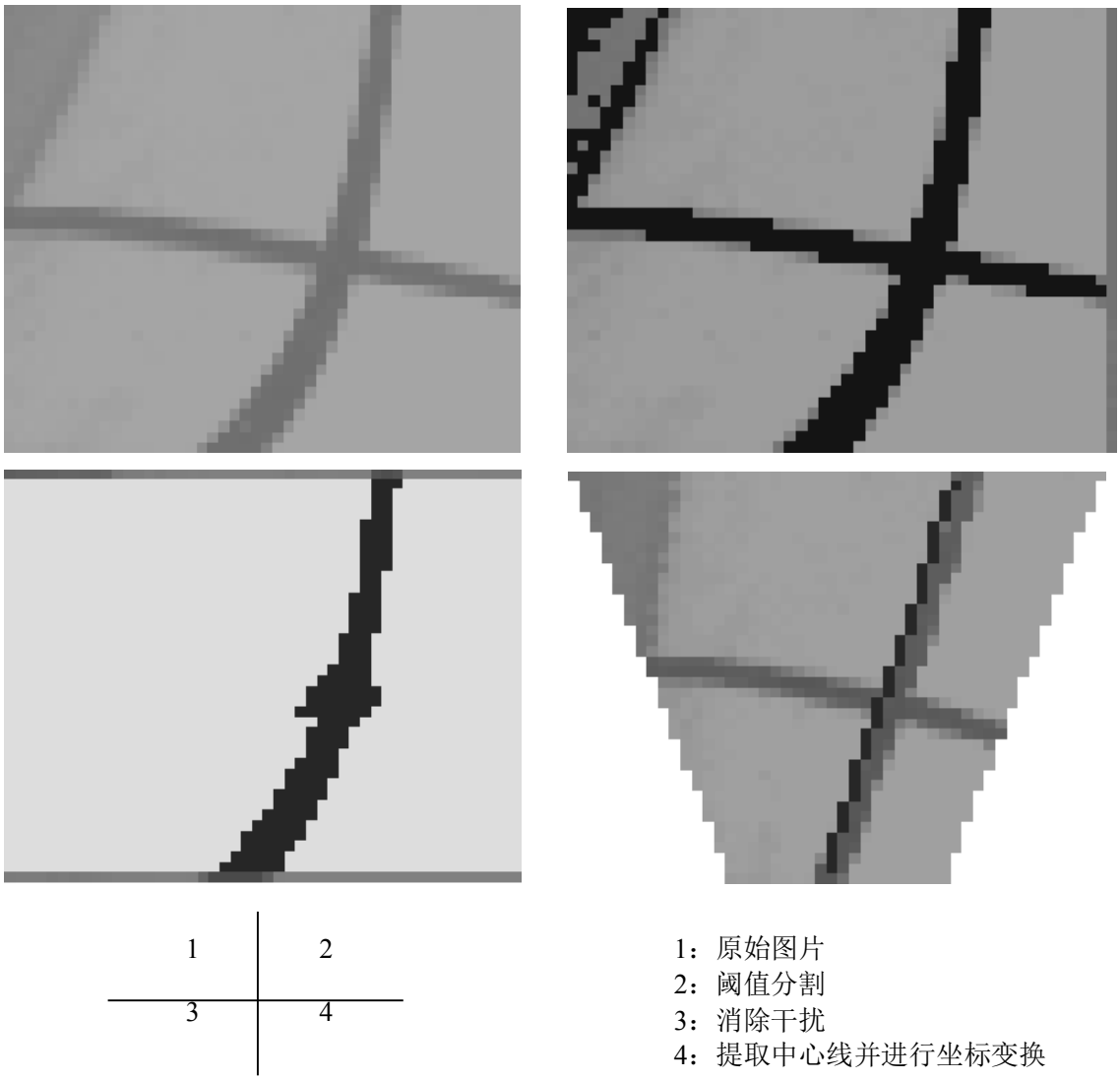


图 5.7 赛道抗干扰处理

2.5. 路径识别

路径识别是整个算法的重点，几乎所有的控制算法都是建立在路径识别上的。路径识别主要是普通弯道、小S形弯道、大S形弯道、十字交叉道的识别。

普通弯道是最容易识别的，可以根据偏移量、曲率等来识别。我们采用前一种，这种方法不但算法简单，而且还可以用于识别多种路况。其具体思路就是将整幅图像的引导线位置相加再求平均值，再根据我们试验得到的数据设定阈值：直道和小S形弯道判断的阈值（我们将小S形弯道也处理成直道），只要平均值小于该值就为直道，大于该值则为普通弯道或大S形弯道，再根据平均值的大小和当前车速做加减速控制和舵机打角控制。

十字交叉道识别也较重要，该处处理不好赛车则不会按预定路线行进，图为我们处理前后的交叉道的情况：

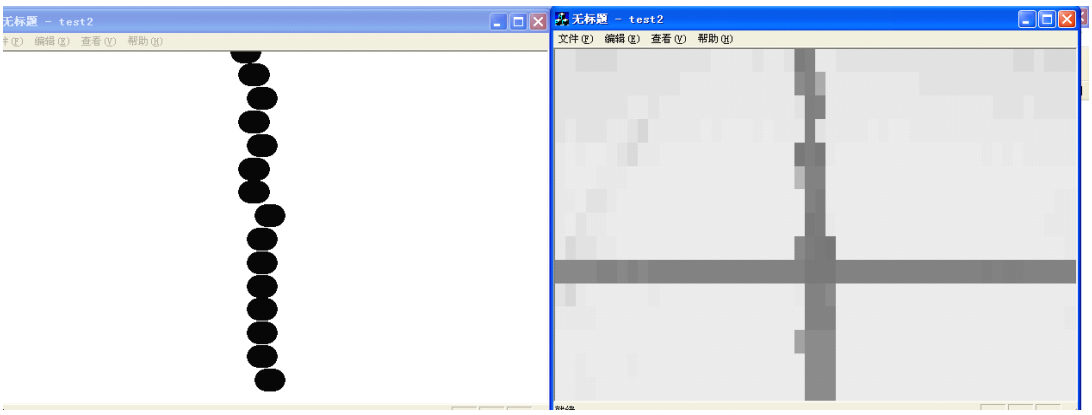


图5.7　路径识别前后的交叉道

2. 图像平滑（去噪）：

用信号处理的理论来解释，我们的做法实现的是一种简单的低通滤波器（low pass filter）。在灰度连续变化的图象中，如果出现了与相邻像素的灰度相差很大的点，比如说一片暗区中突然出现了一个早点，这种情况被认为是一种噪声。灰度突变在频域中代表了一种高频分量，低通滤波器的作用就是滤掉高频分量，从而达到减少图象噪声的目的。

为了方便的叙述上面所说的“将原图中的每一点的灰度和它周围八个点的灰度相加，然后除以 9，作为新图中对应点的灰度”这一操作，我们用如下的表示方法:

图 2 1 模板表示方法

这种表示方法有点象矩阵，我们称其为模板(template)。中间的黑点表示中心元素(此处用*表示)，即，用哪个元素做为处理后的元素。例如 表示将自身的 2 倍加上右边的元素作为新值，而 表示将自身加上左边元素的 2 倍作为新值。通常，模板不允许移出边界，所以结果图象会比原图小。模板操作实现了一种邻域运算(Neighborhood Operation)，即，某个像素点的结果不仅和本像素灰度有关，而且和其邻域点的值有关。在以后介绍的细化算法中，我们还将接触到邻域运算。模板运算的数学涵义是一种卷积（或互相关）运算。这样可以滤除图像中大部分的噪点。

同时，我们根据赛道的具体信息进行虑噪：图像中黑线必然是连续的，上一行中黑线位置和本行黑线位置相差太远就认为本行信息是错误的，将其保持为上一行

的信息。事实证明这种算法可以比较好的适应赛道的具体情况并产生良好的虑噪效果。

3. 畸变校正：

由于摄像头看赛道是有一定角度的，实际上是把赛道上一个梯形映射为一个矩形存储到单片机中了，畸变校正的目的就是将原来的梯形还原出来。畸变校正实际上是一种坐标系的转换。考虑到图形从上到下基本是线性的畸变，所以只要每行的点的横坐标乘以不同的系数就可以达到畸变校正的目的。如我们的设想头看到梯形的上底边 80cm,下底边 35cm 只要将最下面的点的横坐标 乘以 16，最上面点的横坐标乘以 7，并由线性算出中间点的横坐标乘的系数即可将图像中的点较好的还原到原来的位置。当然，算法中没有考虑纵坐标的畸变，是有一定误差的，但是这种误差是控制算法可以接受的。

4. 图像边缘提取：

识别一个对象是从其边缘开始的，一幅图像不同部分的边缘是模式识别最重要的特征。在边沿检测中，常用的一种模板是 Sobel 算子。Sobel 边缘算子是一种一阶差分算子，它可以有效地消除道路图像中的大部分无用信息离散 Sobel 算法的定义如下:

$$\begin{cases} \nabla_x f(x,y) = [f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)] \\ \qquad\qquad -[f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)] \\ \nabla_y f(x,y) = [f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)] \\ \qquad\qquad -f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)] \end{cases}$$

公式 2 6 离散 Sobel 算法

为了编程容易实现，将 Sobel 算子转换为两个卷积核，分别为水平核和垂直核，图像中的每个点都用这两个核做卷积，两处卷积的最大值作为该点的输出值，运算结果即为经过边缘增强的图像。

5. 计算黑线位置：

提取出图像的边缘以后只要将每行图像的边缘进行平均就是图像中黑线的重心位置(Average[23])。能够准确的辨识黑线位置对于下面控制量的给出是至关重要的。

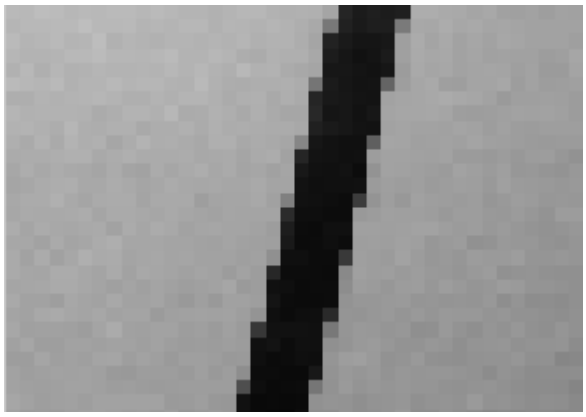


图 2 1 采集到的图像

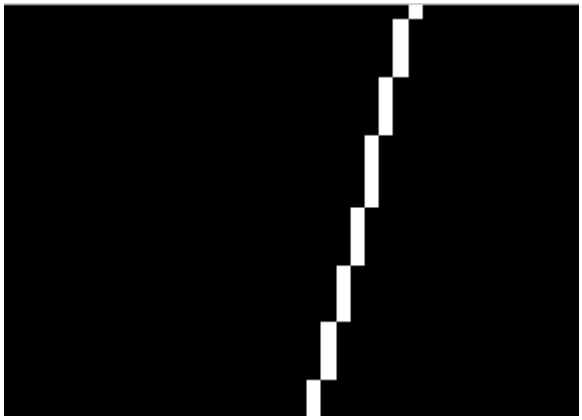


图 2 2 经过处理后的图像

1、路径信息采集处理方法

智能车通过 CCD 视频传感器采集到的路径信息都是模拟值，然后将一场数据的模拟值通过 MC9S12DG128 的 ATD0 通道转换成数字量，这样，这一场的数据就变成了一个 260*33 的二维矩阵，其中的每个数字代表着一场图像中对应位置点的象素灰度值。但是，单片机处理这么多的数据会十分消耗资源，而且单片机的内存也不可能给这个存放图像数据的二维矩阵分配太多的存储空间。所以，为了节省单片机的资源和减少处理时间，就必须对每场的图像做一些处理，本队采取的方法是，每隔 10 行进行一次处理，这样，该场图像的数据就变成了一个 26*33 二维矩阵。图 5.3 就是一条直线的二维矩阵图。

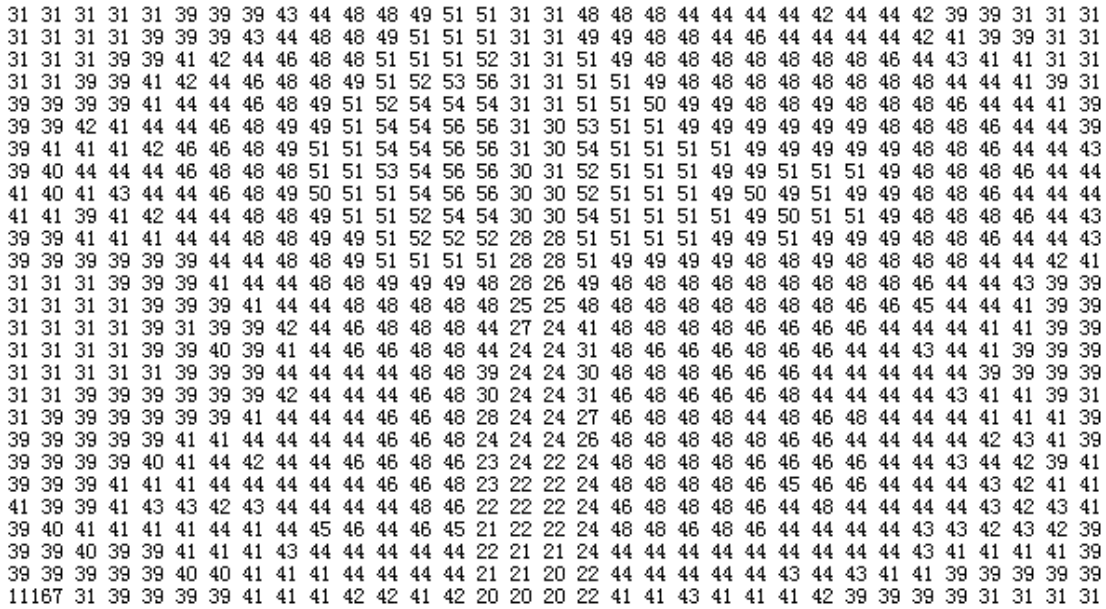


图 5.3 直线赛道图像数据图

接着，就要对 CCD 视频传感器采集的视频数据进行路径判断处理，这时，需要确定视频阈值的大小。如果象素点灰度值小于该阈值，则确定该点为黑点，否则，确定其为白点。这样，就可以在一场数据中分辨出象素点是黑色还是白色。处理后的结果如图 5.4 所示。

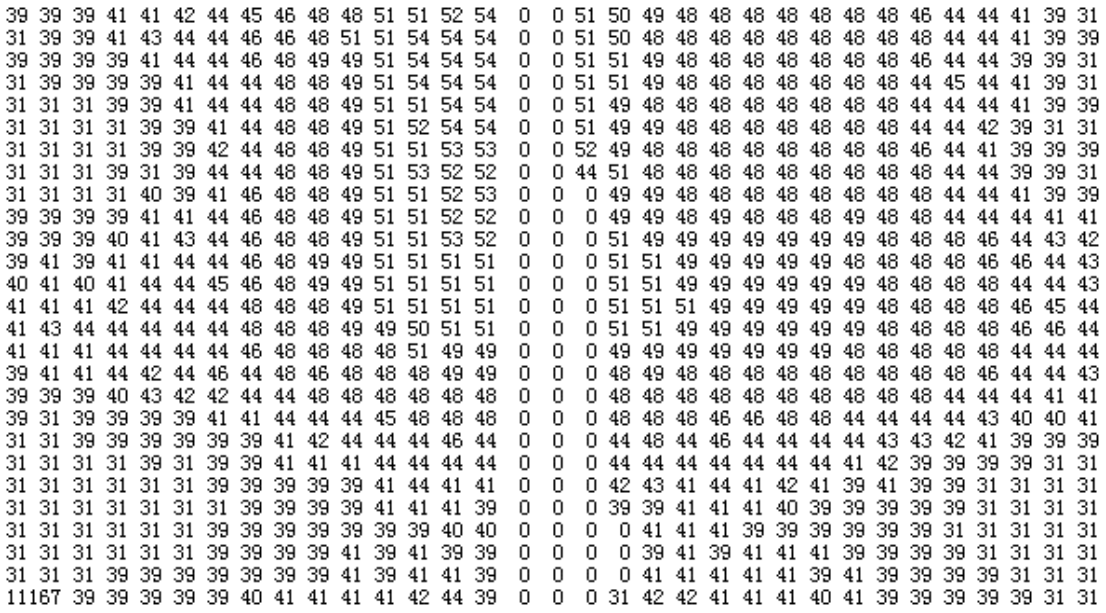


图 5.4 处理后的直线赛道图像数据图

由图 5.3 可知，图像近处的灰度阈值为 31，所以，灰度值小于 31 的点都被判

定为黑点，为了调试方便，把黑点对应的灰度值都置为 0。但是，在上数第 8 行时出现了 31 的灰度值，如果按照近处的灰度阈值来判断的话，该点会被判定为白点，而实际图像中该点的位置却为黑点。因此，为了得到正确的判定结果，可以将远处（图 5.3 中为第 8 行以上）的数据的灰度阈值稍作调整，将其加 1，变为 32，这样，远处的点也可以被判定为黑点，也就符合实际图像中的情况了。这里提到的“远处”开始的位置是通过实践测定的结果，是一个经验值。这也　是灰度阈值自适应的一种实现方法。在确定某点是黑（白）点之后，就可以确定黑线的位置了，下面介绍一下具体的实现方法。因为每行有 33 个象素点，所以把这 33 个点分别标号，记为 0～32。第一次处理的时候，先按照从左向右的顺序，进行比较，当遇到第一个黑点的时候，记下该点的序号 A，接着继续比较，当遇到第一个白点的时候，也记下该点的序号 B，其后的数据可以不处理。这样，将序号（B－A）/2，就可以得到该行黑点的中心位置坐标 C，（B－A）即为该行的黑点个数，记为 D。以上就是第一行的数据，为了提高系统处理的速度，降低系统资源的消耗，以后各行数据的处理方式与第一行有所区别。在处理后面的行时，可以根据上一行得出的黑点中心坐标 C 决定比较的顺序。例如，如果上一行的黑点中心坐标 C 大于行中心坐标 16，那么就从右向左开始比较。再者，通过实践测试观察，摄像头采集的数据中，近处的数据一直是比较准确的，因此，处理的顺序是从近向远。通过实践测试，近 7 行（对应原始图像的前 70 行）作为近处数据，其它的数据作为远处数据。对处理过程再作进一步的优化方法为，远处数据的处理范围为，第 8 行数据参照第 7 行数据得出的黑点中心坐标 C 的位置，将其左右各 5 个点作为第 8 行的待处理数据，然后在遵照上面的方法进行比较。第 9 行再参照第 8 行得出的黑点中心坐标……这样，就可以大大减少待处理的数据，从而降低了数据处理的时间，加快了系统处理数据的速度。

经过上述图像处理后，可以获得两个主要的参数：每行黑点的中心位置坐标和每行的黑点总数。依据这两个参数，可以判断当前的路径信息。

经过条件判断，确定出可靠的最远处的一行数据作为路径判断的依据，即该行的黑点中心位置坐标。

2、舵机转向控制算法

经过上面的图像处理方法，得到了路径识别的结果，记为可靠的最远处的一行

数据黑点中心位置坐标。根据此坐标，来控制舵机的转向。因为这个数据是远处的，所以有一些前瞻性，因此，可以提前做出转向的动作，这样，在急转弯的时候将会有很大的优势。

为了更好的解决转向的问题，本队采用查表的方法进行控制。设置一个一维数组，其中有 33 个元素，对应每行的 33 行坐标位置，不同的坐标对应不同的转角。而这 33 个转角的大小都是通过实际测量实验得到的。

2.6.有一定抗干扰和抗反光能力的黑线提取算法

(1)原始图像特点：

原始图像是一个将模拟图像经A/D转换得到的120×30的二位数据矩阵，矩阵的每一个元素对应一个像素点的灰度值，黑色的灰度值低，白色的灰度值高。图像的第一行对应最远处，大约150cm ，图像的最低部一行对应最近处，大约20cm。远处的图像小，近处的图像大，黑线为梯形状。

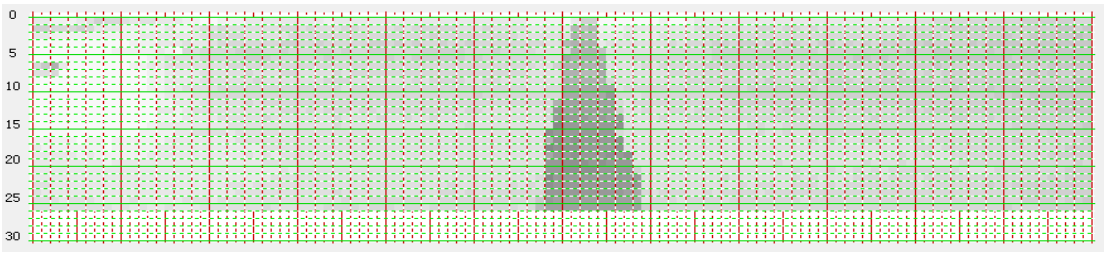


图 4.4 直道

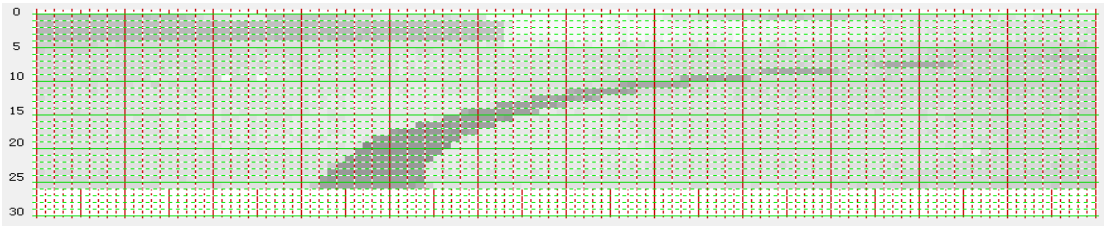


图 4.5 弯道

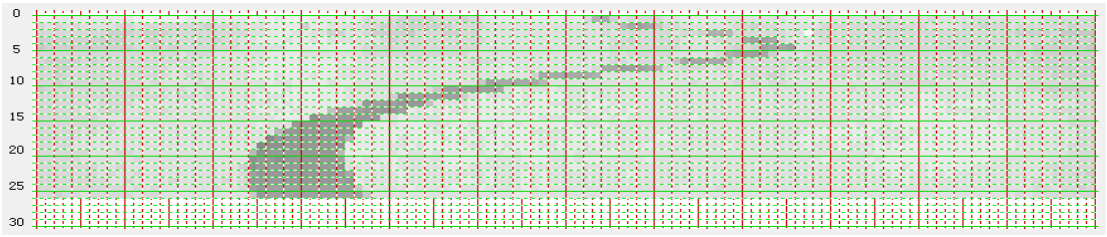


图 4.6 S 弯

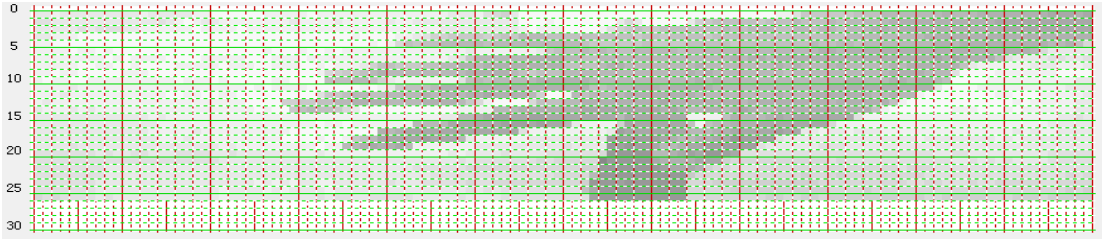


图 4.7 手

(2)基本思想：

a) 直接利用原始图像，因为如果采用3×3均值滤波或中值滤波会占用大量的计算时间，况且在阈值选择合理的情况下原始图像黑线比较清晰，容易提取。

b) 逐行扫描，小于阈值的点判断为黑点。

c) 黑线宽度有一个范围，在确定的黑线宽度范围内才认为黑线有效，这样可以滤除不在宽度范围内的黑点干扰。

d) 利用黑线的连续性来确定求出的黑线中心是否有效。

e) 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确定。

f) 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

g) 图像数据量大（120×30）,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，避免了整行逐点扫描，节约了时间。

(3)算法程序流程图：

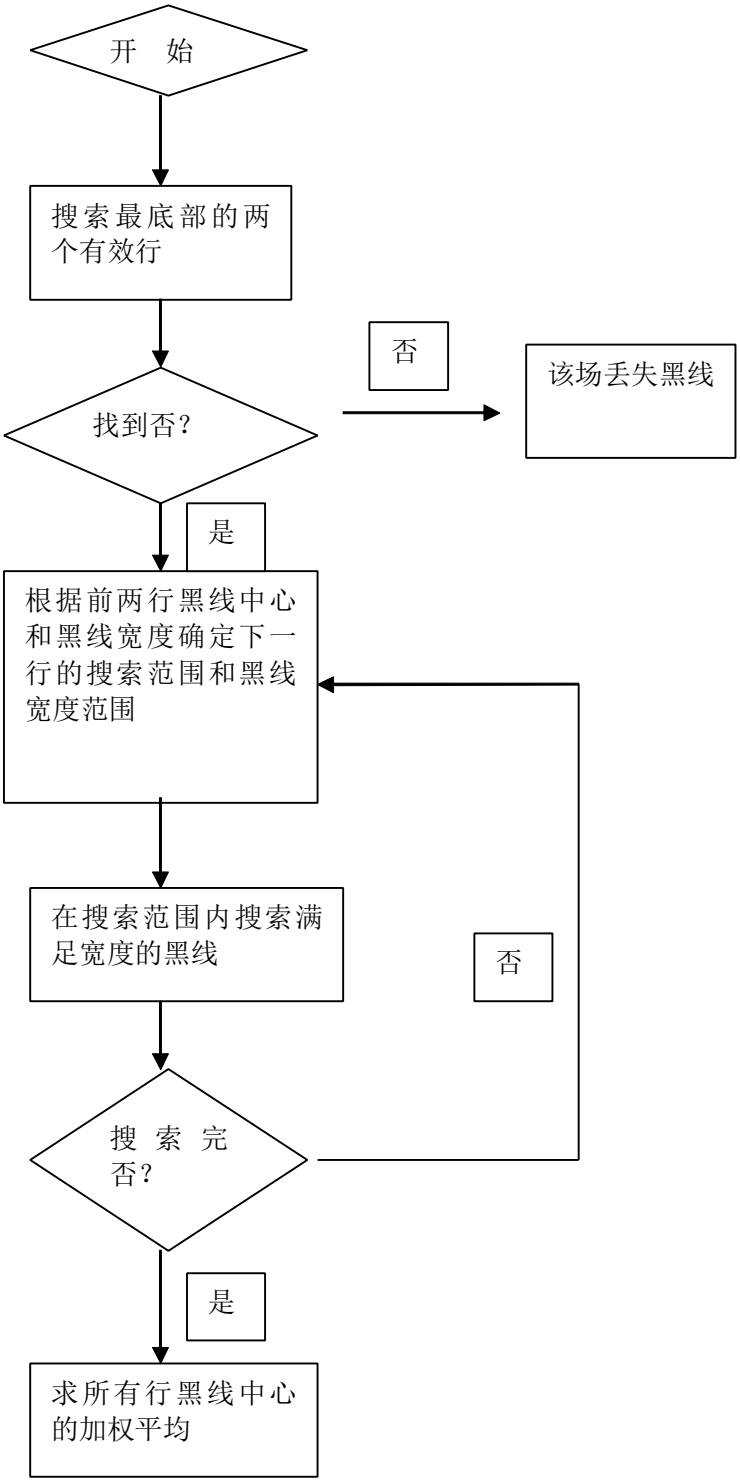


图 4.8 算法流程图

(4)处理后求出的黑线中心点

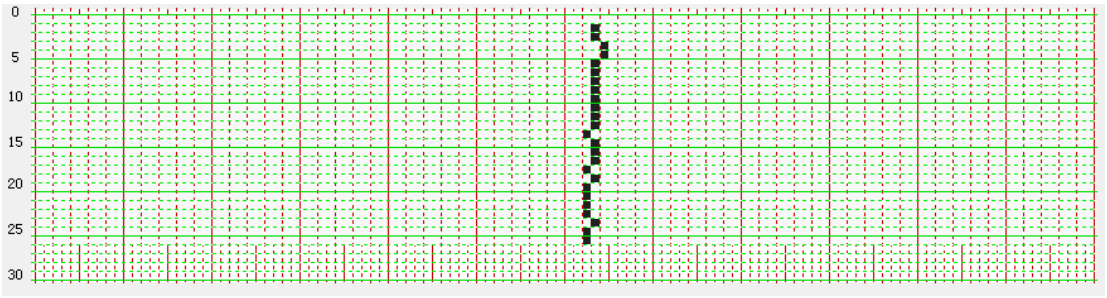


图 4.9 直线

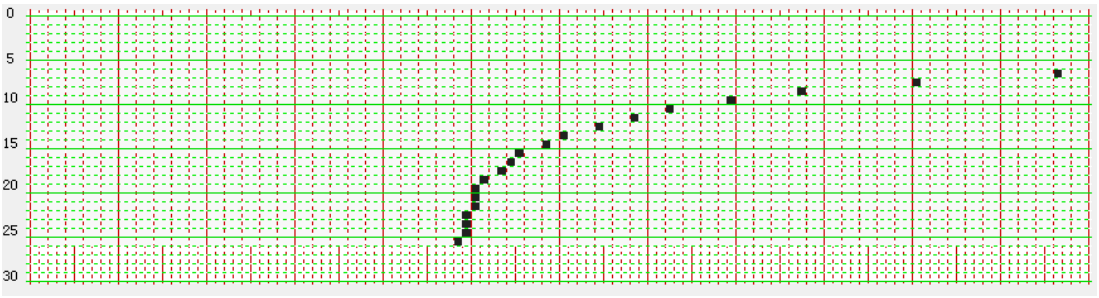


图 4.10 弯道

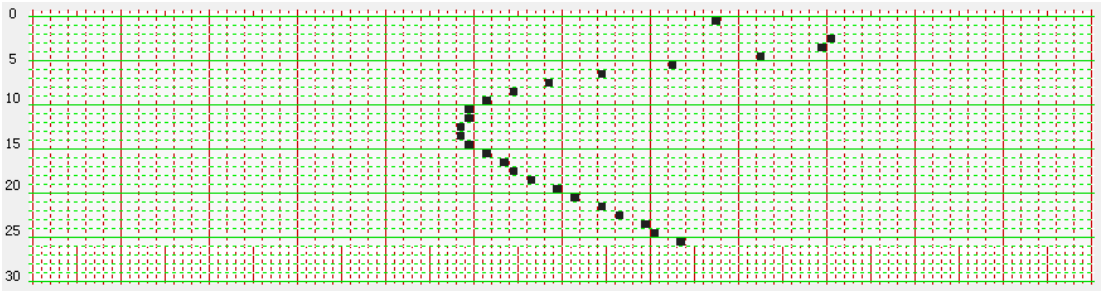


图 4.11 S 弯

2.7.图像采集子程序

本设计将摄像头设置为输出 QQCIF(88×72)格式的图像，输出数据格式为 YUV(4:2:2)。我们只取 Y 量，即可得到黑白图像。为了便于图像处理，我们需开辟两个缓存区，一块用于保存当前图像，一块用于处理上一幅图像。这样，单片机的 RAM 就不够用了。我们将图像抽行提取，最终保存的图像大小为 40×72，既满足控制精度的要求，又能解决 RAM 不足的问题。图像采集子程序的工作流程图如图 4.2 所示

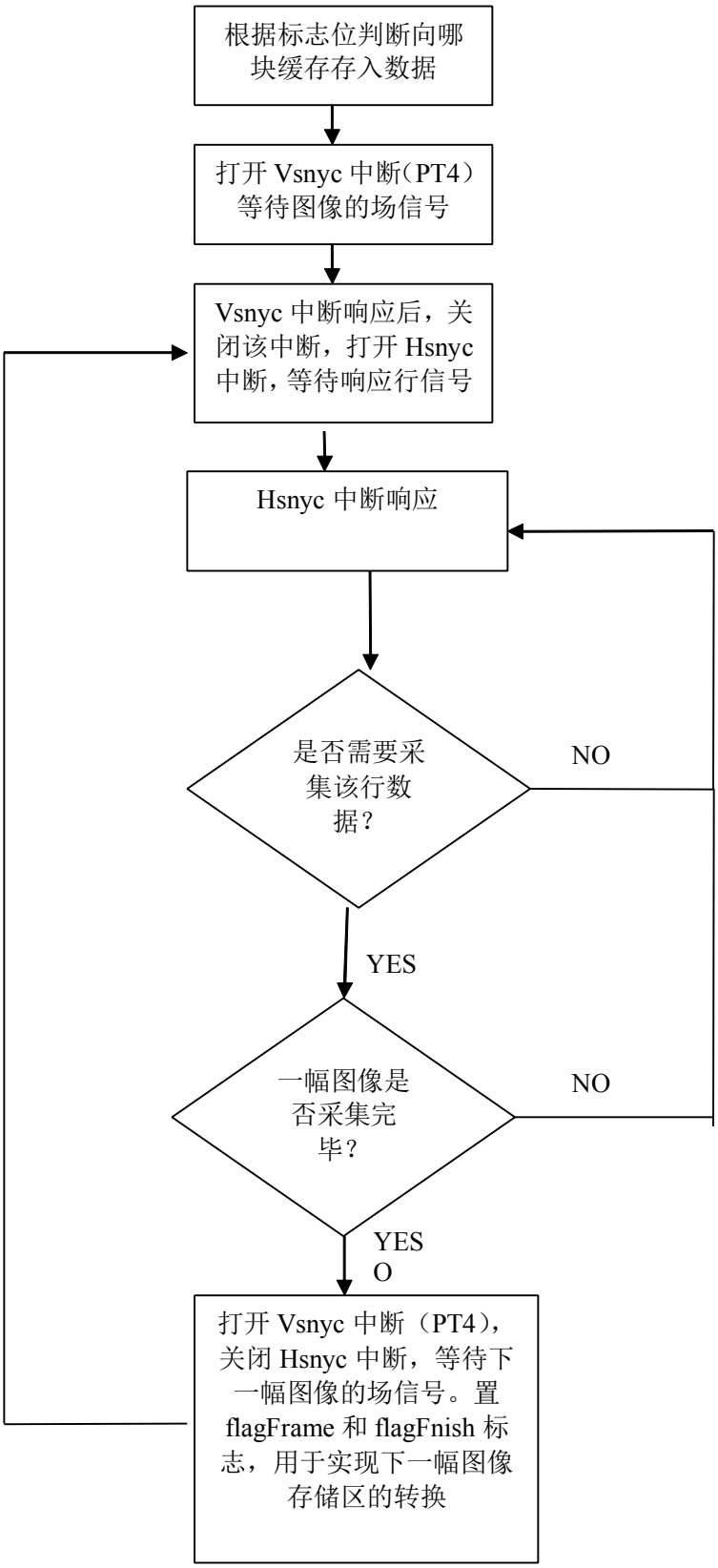


图 4.2　图像采集子程序工作流程图

2.8. 图像处理子程序

1、阈值的确定

受单片机处理速度的限制，要实现动态的阈值确定比较困难。我们只能在系统初始化后采集几幅图像，找到整幅图像中的像素最小值和相隔两个像素的最大像素差分值，将像素最小值加上最大差分值的三分之一作为整幅图像的阈值。在以后的图像处理中都用该阈值对图像进行二值化。

2、黑线的识别

根据上面得到的阈值将图像二值化。对二值化后的图像进行分析，计算每行中连续黑像素的个数，当该值超过二时，将该位置记录下来。当一行扫描完毕时将连续黑像素最多的点的中间位置作为该行的识别结果，即得到黑线的位置。

3、道路特征的提取

本设计将道路分为四类，分别为直道、S 道、左转弯道和右转弯道。在采集到的图像中提取等距离的五行黑线的位置，然后根据每一行黑线位置偏差值的增减趋势判断道路特征为以上四种中的哪一种。

2.9. 速度及加速度信号采集子程序

当图像数据采集处理完后，我们读取速度传感器和加速度传感器的值，采集这两个传感器信号的频率也为 60Hz。流程图如图 4.3 所示。

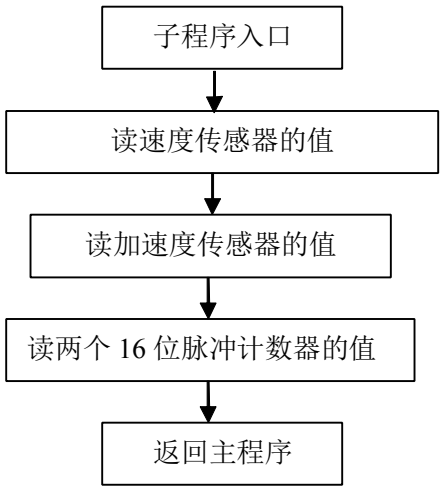


图 4.3 速度及加速度信号采集子程序流程图

2.10. 数据处理子程序

在数据采集子程序中，我们得到的数据有：图像数据、车速传感器的信号、加速度传感器信号。

将这三种信号作为输入量，经综合分析处理，得到舵机控制量和驱动电机控制量两个输出量，实现车辆的横向控制和纵向控制

2.11. 图相采集和数据处理

摄像头的信号经过 LM1881 视频分离芯片分离出行信号和场信号，场信号代表一个图像开始标志，行信号是单片机进行 AD 采集标志。本作品采用 DG128 片内 AD 采集，系统时钟为 24MHz(由于稳压电路不是太好，所以超频的话，小车容易死机。) 。首先，用实时中断跳过 800us 的消影区，然后用 I/O 口中断隔 5 行采一次共采集 51 行，由于第一行容易出错，所以我们只用到了后面 50 行数据。把 AD 采集的数据放入一个二为数组 AD_Value_Video[50][40]中，受到采集频率的限制每行有效数据为 34 个（如果使用广角镜头的话，需要超频才能才到较远处的的黑线）。

每采集完一行就用边沿提取法提取出黑点的位置，

（1)该算法主要过成如下从有效数据最左端的第 2 个点开始一次向右进行阈值判断：通过分析采集到的数据满足阈值的要相隔 1～2 个点（如图 4.2 所示）。

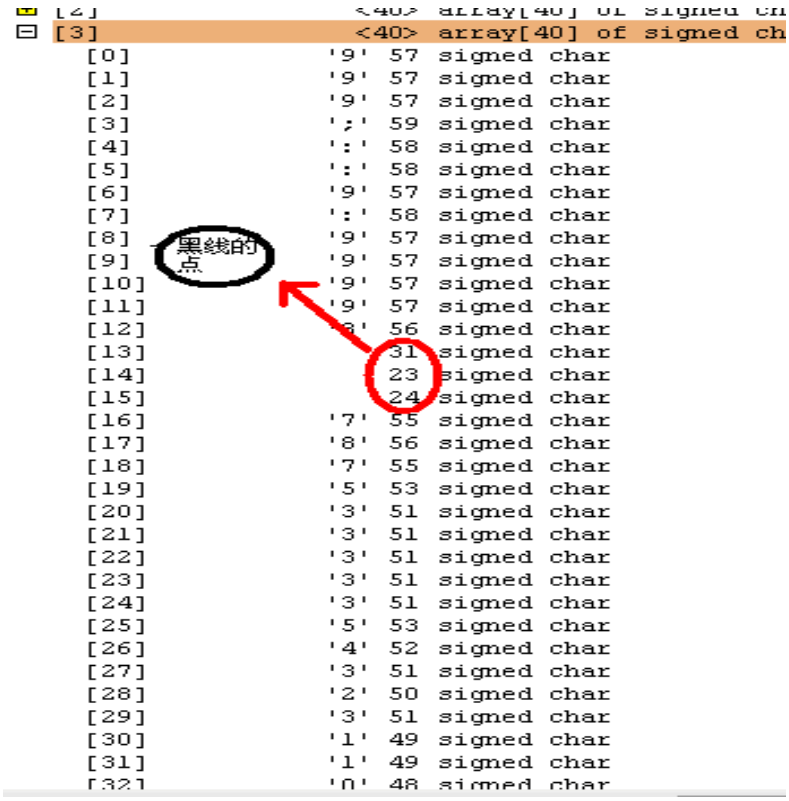


图 4. 4

（2）设 AD_Value_Video[row][i]为原点，判断和 AD_Value_Video[row][i+2]的差值是否大于阈值。如果是，则将 i+2 放入数组 data_temp[]中，通过分析采集的数据，黑线的点最多为 3 个，所以从第 i=i+5 个位置继续判断是否满足条件，如果满足条件，则将该点放入数组 data_temp[]中;

（3）判断 data_temp[i]中的点的 AD 值，是否满足小于两边的点的 AD 值，如果满足 data_temp[i]值不变，否则 data_temp[i]=0;

（4）找出 data_temp[i]中点不为 0 的个数，如果个数为 1；则这个点迹为黑线的位置并把它放入 Line_Place[row]中。如果大于 1，则比较两点的 AD 值，取 AD 值小的为有效值并把该值放入到 Line_Place[row]中（滤掉毛刺）；如果个数为 0;则 Line_Place[row]＝0；

利用该算法所得到的黑线提取效果不仅可靠，而且实时性好；在失去黑线目标后能够记住是从左侧还右侧超出视野，从而控制舵机转向让赛车回到正常赛道。

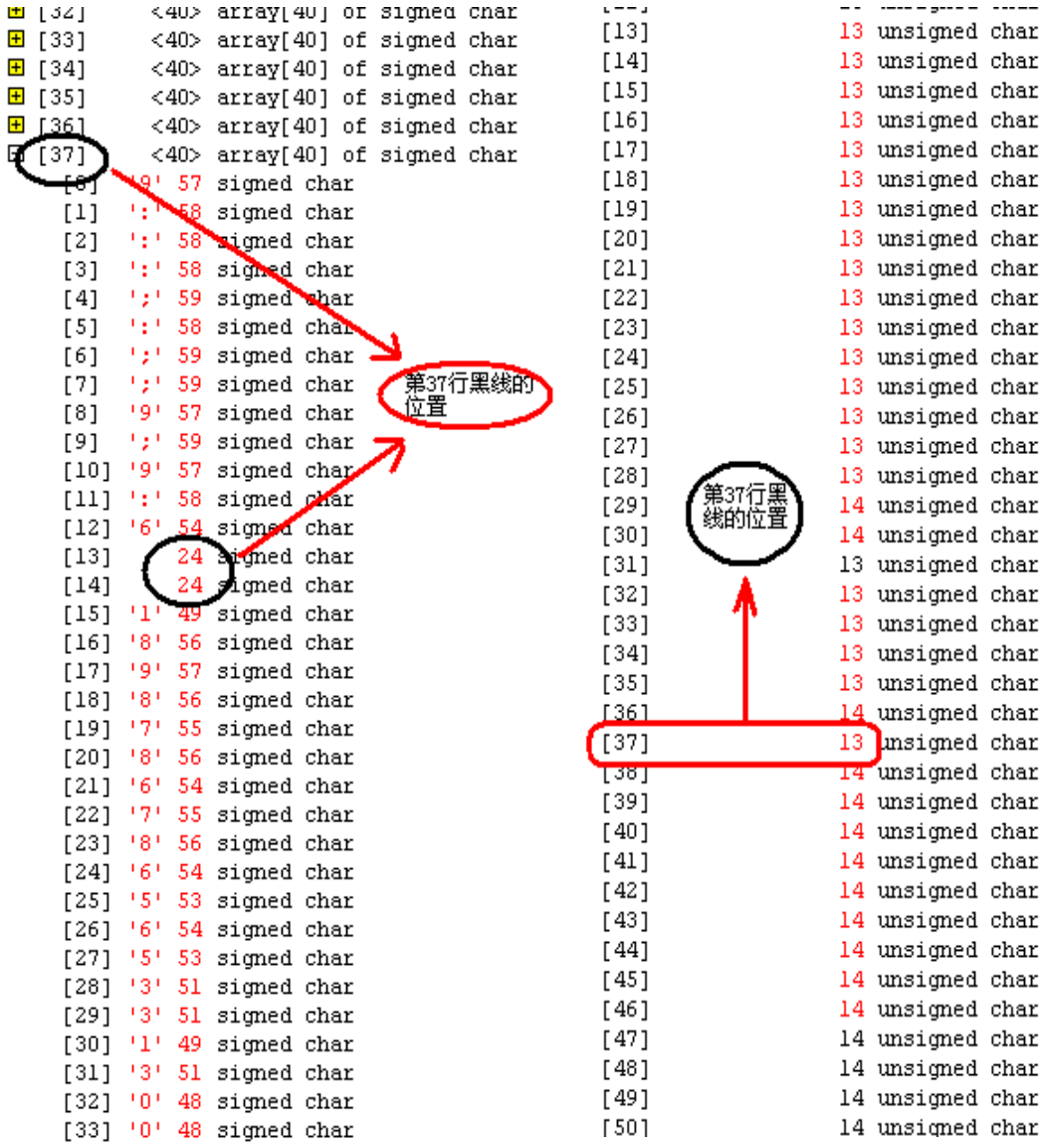


图 4.5

2.12. 图相处理和速度控制结合

本组小车的循迹控制周期为 20ms，调速周期为 10ms，其中调速程序用到了循迹控制中的速度标志变量 Speed_flag，由于 Speed_flag 20ms 更新一次且调速和循迹都为中断程序，所以调速频率要大于循迹中断率。

我们把图象分为三个部分 Video0,，Video1 ,，Video2，其中 Video0 部分为了控制速度提供一个速度标志 Speed_flag, Video1 和 Video2 两部分用来循机迹得出一个 Turn_Video, 它作为舵机输出。通过 Speed_flag 和 Turn_Video 联合判断小车前方道路状况，通过实际测算基本上能判断出大 S 弯和 U 形弯道，而小 S 弯道可通过 Video1 和 Video2 的求取虑成直线，然后通过增量 PI 实现调速。这样就可以实现良好的循迹和加减速控制。

Video0 代表前 15 行的部分；

Video1 代表是整个 50 行的图象；

Video2 代表中间的 10～35 行的图象

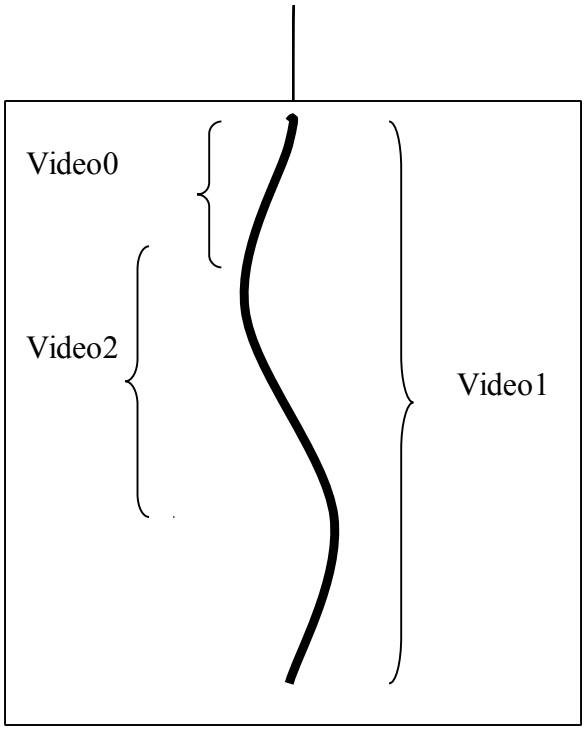


图 4.6

2.12.1. 速度标志的求取

在求取 Speed_flag 时我们用了前 15 行数据（也就是 Video0 部分），求平均值的方法得出黑线的相对位置 Speed_flag，

首先算出数组 Line_place [1]～Line_place [15]中不为 0 的点的个数 temp。如果 temp<=5，规定前 15 行已经超出了黑线，这样 Speed_flag 的输出就不经过下面的式子了，调用上一次的 Speed_flag ，如果 Speed_flag 大于直线范围，则 Speed_flag 每次加 value，直到最大。如果 Speed_flag 小于直线范围，则 Speed_flag 每次减 value，直到最小。

Speed_flag=（Line_place[1]+ Line_place[2]+ ••• ＋Line_place[15]）/temp；

在通过分段线性关系:

k2<k1;

Speed_flag*=k1 ;　　Speed_flag<9;

Speed_flag*＝k2;　　Speed_flag <＝19&&Speed_flag>=9;

Speed_flag*=k1;　　Speed_flag>19;

2.12.2. 舵机输出控制

Video1 和 Video2 也是通过平均值法求出黑线的相对位置 Turn_Video1 和 Turn_Video2。

首先算出数组 Line_place [1]～Line_place [50]中不为 0 的点的个数 temp。如果 temp<=10，规定黑线已在小车的视野之外，这样 Turn_Video1 的输出就不经过下面的式子了，调用上一次的 Turn_Video1 ，如果 Turn_Video1 大于直线范围，则 Turn_Video1 每次加 value，直到最大。如果 Turn_Video1 小于直线范围，则 Turn_Video1 每次减 value，直到最小。

Turn_Video1=（Line_Place[1]+ Line_Place[2]+ ••• ＋Line_Place[50]）/temp;

算出数组 Line_place [10]～Line_place [35]中不为 0 的点的个数 temp。如果 temp<＝7，规定以超出了黑线，这样 Turn_Video2 输出就不经过下面的式子了，调用上一次的 Turn_Video2，如果 Turn_Video2 于直线范围，则 Turn_Video2 次加 value，直到最大。如果 Turn_Video2 于直线范围，则 Turn_Video2 次减 value，直到最小。

Turn_Video2=（Line_Place[10]+ Line_Place[11]+ ••• ＋Line_Place[35]）/temp;

（5）再通过分段线性关系：

$$\begin{cases} Turn\_Video1*=k1; & Turn\_Video1<9; \\ Turn\_Video1=turn[Turn\_Video1]; & 9<=Turn\_Video1<=19; \\ Turn\_Video1=Turn\_temp+k1*Turn\_Video1; & Turn\_Video1>19; \end{cases}$$

$$\begin{cases} Turn\_Video2*=k1; & Turn\_Video2<9; \\ Turn\_Video2=turn[Turn\_Video2]; & 9<=Turn\_Video2<=19; \\ Turn\_Video2=Turn\_temp+k1*Turn\_Video2; & Turn\_Video2>19; \end{cases}$$

以上用到的 9 和 19 都是经验值，本作品规定的一行黑线有效位置为 0～28

其中 9～19 规定为直线范围，以外为弯道范围，在求取 Speed_flag 中直线的比例要小于弯道比例，是为了增大直线加速的范围，而在 Video1 和 Video2 中求取 Turn_Video1 和 Turn_Video2 直线输出采用查表的方式，是为了把小 S 弯道虑成直线，然后调用直线的跑法提高速度。

通过实践直线的比例要小，过大的话高速直线会左右振荡。弯道的比例要适中过大会撞到塞到外的杆子。过小的话过弯就会增加行进路程，同时比例过小也不利于高速过弯。

以上通过 Turn_Video1 和 Turn_Video2 共同输出到舵机公示如下：

Turn_Video＝(K1*Turn_Video1＋K2*Turn_Video2)/(K1+K2);

最后再判断 Turn_Video 是否超出了舵机的输出范围（我们规定舵机的输出范围为 1370 ～ 1430 ）。 如 果 Turn_Video<=1370, 则 Turn_Video ＝ 1370 ； 如 果 Turn_Video>=1430,则 Turn_Video=1430。

2.12.3. 速度控制

采用 Speed_flag 和 Turn_Video 联合识别前方路况，为调速提供依据。

（1） 直线：Speed_flag 在直线范围&&Turn_Video 在直线范围，如图 4.5：

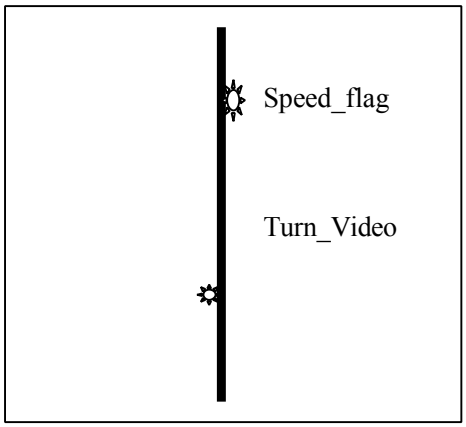


图 4.7

（2）入弯：Speed_flag 在弯道范围&&Turn_Video 在直线范围，如图 4.6：

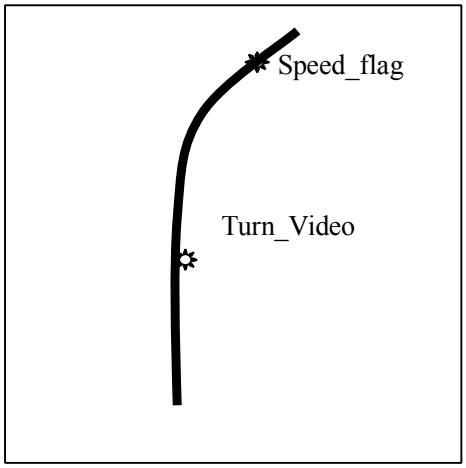


图 4.8

（3）大 U 形弯：Speed_flag 在弯道范围内&&Turn_Video 相同侧的弯道范围 并且在行使 0.7s 后 Speed_flag 和 Turn_Video 同上次侧相同，如图 4.7：

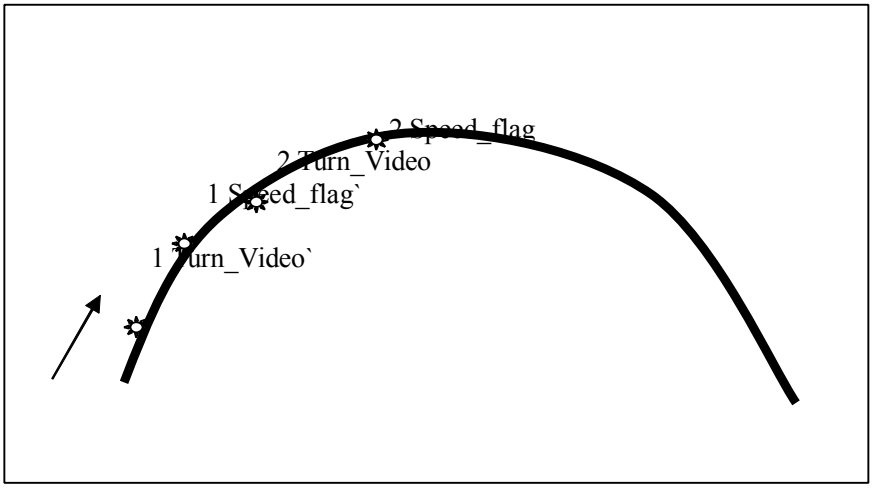
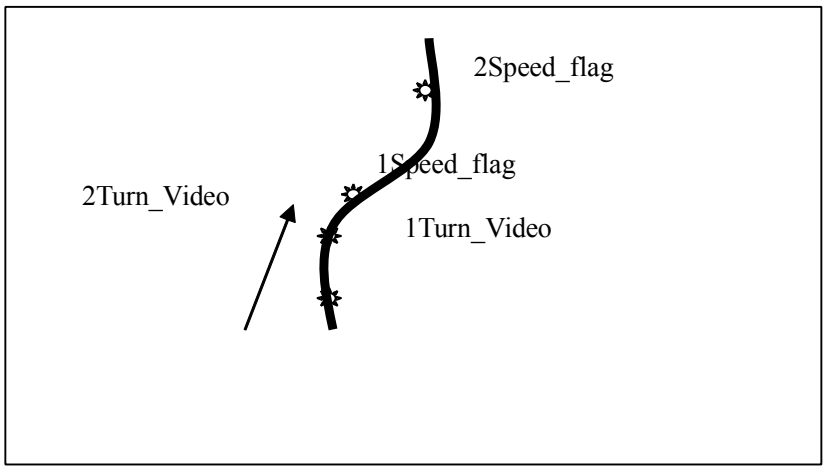


图 4.9

（4）大 S 出弯或出弯：Speed_flag 在弯道范围内&&Turn_Video 相同侧的弯道范围，但在行使 0.7s 后 Speed_flag 和 Turn_Video 同上次范围不同。



(5)弯道：Turn_Video 在弯道范围&&Speed_flag 在弯道范围内。

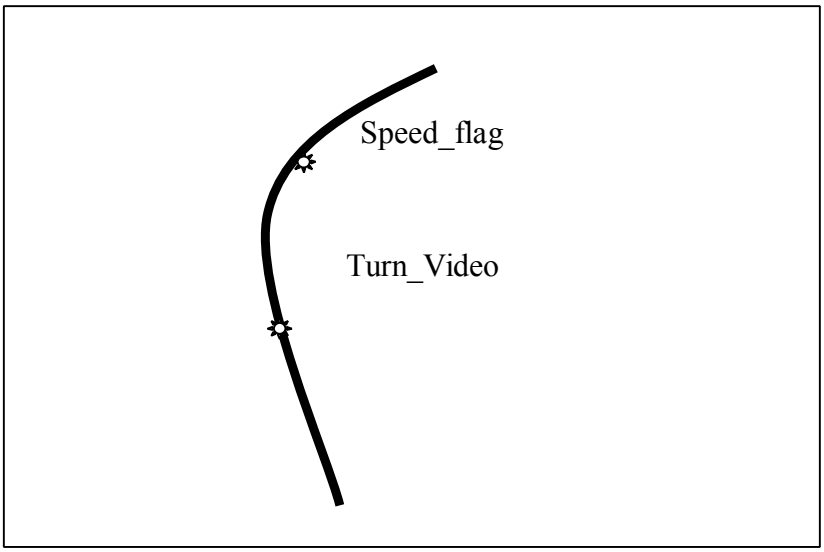


图 4.11

通过以上的条件可以对速度进行合理的控制。

2.13. 路径规划

要想以最短的时间完成比赛，并不需要严格按照黑色引导线行驶，我们需要找出相对较短的行驶路径，即路径规划。例如 过普通弯道的时候，以同样的速度走内弯显然比走中间或外弯所花时间少，过小S形弯时。直接过去和严格循迹相比速度快，

行驶距离短等等。（图）

如果能提前预知前方赛道信息，那么我们就有可能对路径进行规划。由于我们采用的是摄像头，扫描距离较大，这就为我们进行路径规划提供了条件，这也是我们选择摄像头的原因之一。

由于摄像头看得较远，每幅图像反映的赛道信息量较大，我们需要针对赛车在不同位置，对其有侧重的采用。具体就是将采到的15行划分为几个区域，针对不同路况对每个区域赋以不同的权值。其原理图如图：

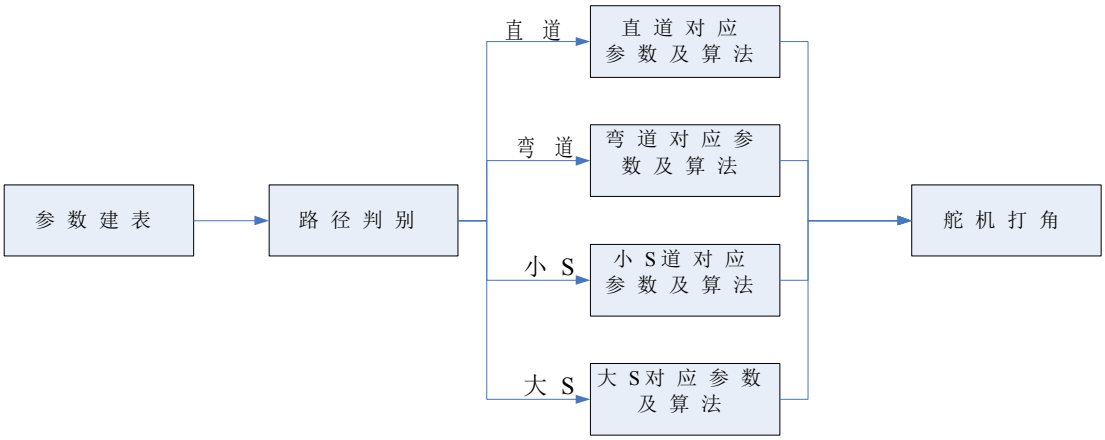


图5.10 路径规划示意图

2.14. 图像处理（上海交大一部分）

一场图像采集完成后，得到一幅 20*20 的原始二值图像信息，如图 4.3（A）所示，同时给标志变量 VideoFinFlag 置 1。主程序中检测到此标志的时候，启动滤噪程序。考虑到图像中有效的信息为简单的黑色线条，必然具有很强的连续性。所以，可以采用从图像一侧到另一侧，逐行邻域查询的办法。具体来说，就是根据已经确定的前行黑线中心位置，在本行的一个邻域内查找黑点。超出这个邻域的黑点都认为是无效的。这样，只要能够确定某一行黑线的中心位置，就能逐行确定各行黑线的中心位置，从而，也就提取出了图像中最基本的信息——黑线中心位置轨迹（如图 4.3（B）所示）。

原始图像信息	处理后的信息
00000000000010000000	2
00000000000010000000	2
00000000001100000000	1
00000000001100000000	1
00000000010000000000	0
00011000001000000000	0
00010000001000000000	0
00000000011100000000	0
00000000011100000000	0
00000000111000000000	-1
00000000111000000000	-1
00000000111100000000	0
00000000111100000000	0
00000000111100000000	0
00000001111000000000	-1
00000001111000000000	-1
00000011110000010000	-2
00000011110000000000	-2
00000011110000000000	-2
00000111110000000000	-3
(A)	(B)

图 4.3 处理前后的图像信息

2.15. ccd 路径识别与自适应阈值计算

所谓路径识别，简单的理解就是把图像中反映路径的部分提取出来。这是一个图像分割的过程。图像分割是计算机进行图像处理与分析中的一个重要环节， 是一种基本的计算机视觉技术。在图像分割中，把要提取的部分称为“物体（Object）”，把其余的部分称为“背景（Background）”。分割图像的基本依据和条件有以下 4 个方面：

1）分割的图像区域应具有同质性，如灰度级别相近、纹理相似等；

2）区域内部平整，不存在很小的小空洞；

3）相近区域之间对选定的某种同质判据而言，应存在显著的差异性；

4）每个分割区域边界应具有齐整性和空间位置的平整性。

现在的大多数图像分割方法只是部分满足上述判据。如果加强分割区域的同性质约束，分割区域很容易产生大量小空洞和不规整边缘；若强调不同区域间性质差异的显著性，则极易造成非同质区域的合并和有意义的边界丢失。不同的图像分割方法总是为了满足某种需要在各种约束条件之间找到适当的平衡点。

图像分割的基本方法可以分为两大类：基于边缘检测的图像分割和基于区域的图像分割。

边缘是指图像局部亮度变化最显著的地方，因此边缘检测的主要依据是图像的

一阶导数和二阶导数。但是导数的计算对噪声敏感，所以在进行边缘检测前需要对图像滤波。大多数的滤波算法在滤除噪声的同时，也降低了边缘的强度。此外，几乎所有的滤波算法都避免不了卷积运算，对于智能车系统来说，这种运算的计算量是 S12 单片机系统所无法承受的。

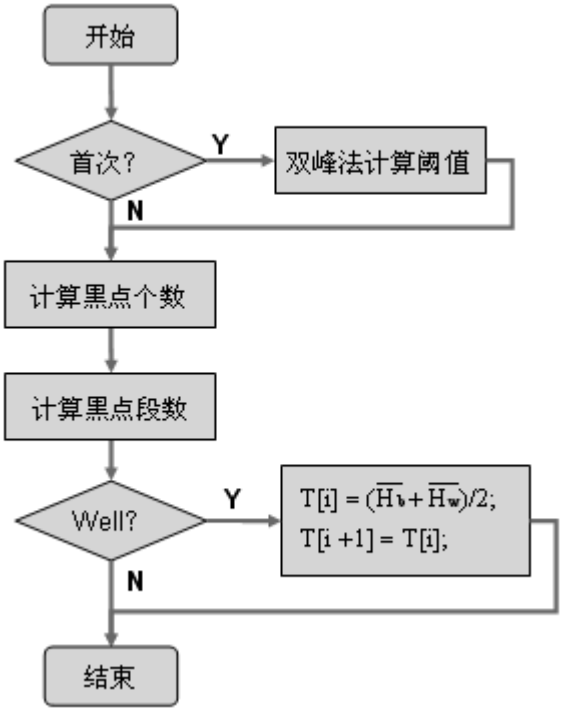
阈值分割法是一种基于区域的分割技术，它对物体与背景有较强对比的景物的分割特别有用。它计算简单，而且总能用封闭且连通的边界定义不交叠的区域。阈值分割法的关键在于阈值的确定。如果阈值是不随时间和空间而变的，称为静态阈值；如果阈值随时间或空间而变化，称为而动态阈值。基于静态阈值的分割方法算法简单，计算量小，但是适应性差。基于动态阈值的分割方法其复杂程度取决于动态阈值的计算方法[6]。

针对本智能车系统，普通的双峰法就能适合绝大部分情况，因为智能车的运行环境是比赛赛道，背景和前景区分明显，且背景简约。但是实验环境并不理想，由于受到光线斜射的影响，有时背景和前景的对比十分不明显。结合实际需要，提出了一种新的计算阈值的方法，这种方法的思想与迭代法有些相似。

首先我们假定，智能车系统运行时，开始时采集的第一幅图像是良好的。这个“良好”含义是：第一行也就是最近处的一行，完整的包含了导航线，并且使用“双峰法”能正确提取。其次，我们对每一行都定义一个阈值 Tr，每一行都用该行的阈值 Tr 进行分割。在此基础上按照以下规则进行阈值传递：

c) 如果第 r 行分割出来的黑色区域（线段）是连续的，那么计算目标区域和背景区域的平均灰度值，并取其中值，做为当前行和下一行的 Tr。

d) 如果第 r 行分割出来的黑色区域是空集或者不连续，则保持当前行和下一行的阈值不变。



这种算法的特点是运算量小，但是能跟踪环境光线的逐渐变化。由于环境光线不管在时间上还是空间上都是逐渐变化的，所以这种分割算法产生严重错误的概率极小，小于 1/10000。这个数据的来源是，我们让智能车在实验赛道上运行，累计时间超过 30 分钟，没有一次智能车因为阈值分割错误而停车。图 5.2 是智能车因错误而停车时的即时照片，在上面几行中，赛道黑线与背景间的差别已十分微弱，但图 5.3 显示，这种情况下阈值分割的结果仍然是正确的，包含了识别赛道的足够信息，导致智能车停止的是其它的程序错误。图 5.3 的最右边一行显示的是阈值，图 5.4 则集中显示了阈值从近处到远处的变化。

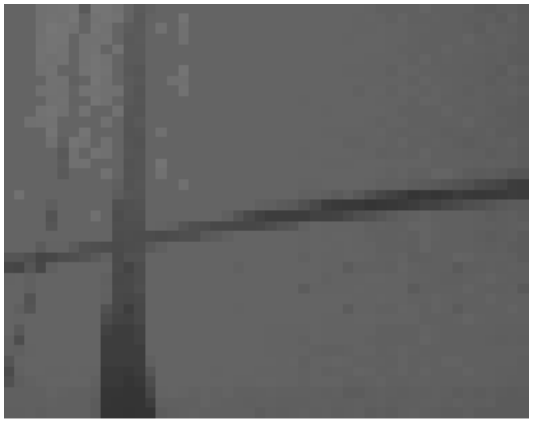


图 5.3 干扰比较严重的赛道图片

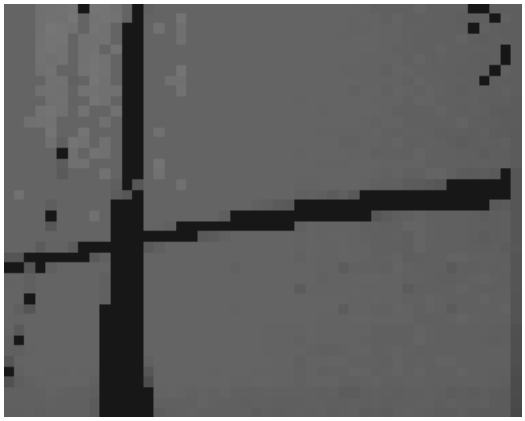


图 5.4 动态阈值分割的结果，保留了背景

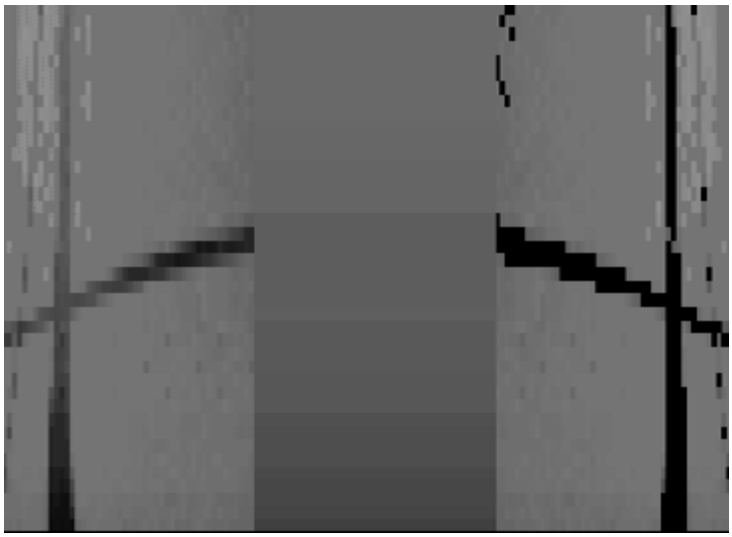


图 5.5 左：原始图像；中：阈值；右：分割的图像

2.16. 图像处理

由于摄像头采集回来的图像干扰信息很多，图像的梯形失真也需要校正。这就要求我们对采集回来的原始图像进行滤波等处理。

1、二值化处理

二值化处理最重要的便是阈值的设定。说为的阈值就是将灰度或彩色图像转换为高对比度的黑白图像。所有比阈值亮的像素转换为白色；而所有比阈值暗的像素转换为黑色。有两种阈值方法。固定阈值和动态阈值，前者又分为全局阈值法和局部阈值法。全局阈值即是整幅图像都使用一个阈值，局部阈值法是在不同区域采用不同阈值。动态阈值是根据环境的变化，针对采集回来的图像信息计算阈值。全局阈值发在图像处理中应用比较多，我们也采用此法。

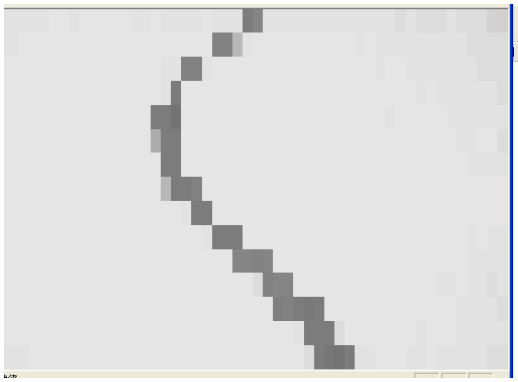


图5.5 二值化前图像

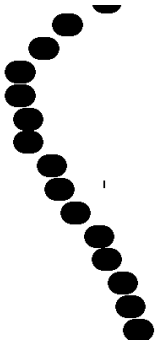


图5.6 二值化后图像

2、图像滤波

找出二值化以后的图像每一行中的黑色区域，如果只有一个黑色区域，则记下黑色区域起始点对应的列号和结束点对应的列号，求平均值即为黑线中心，如果黑色区域大于或者等于两个该行图像信息有错，则利用该行相邻行找出正确的黑线位置。 如果连续几行都为错，则放弃这几行，直道找到正确行为止，再将刚才放弃的部分黑线补上。

3、梯形失真校正

梯形失真会导致路进识别不准甚至出错，我们对梯形失真进行了简单校正。我们采用比例法来矫正梯形失真。由于摄像头视野最远处的宽度和最近出的宽度显然是不一样的，而摄像头的行分辨率是相同的，这就导致两个采样点之间的实际距离是不一样的，而单片机却认为他们是一样的，这也是导致梯形失真的原因。我们测量出摄像头扫描赛道最近处和最远处视野宽度，算出相邻两个采样点所代表的实际距离，再量出视野长度，中间各行相邻两采样点之间的距离根据其行号按比例计算出来。

2.17. 路径识别

路径识别是整个算法的重点，几乎所有的控制算法都是建立在路径识别上的。路径识别主要是普通弯道、小S形弯道、大S形弯道、十字交叉道的识别。

普通弯道是最容易识别的，可以根据偏移量、曲率等来识别。我们采用前一种，这种方法不但算法简单，而且还可以用于识别多种路况。其具体思路就是将整幅图像的引导线位置相加再求平均值，再根据我们试验得到的数据设定阈值：直道和小S形弯道判断的阈值（我们将小S形弯道也处理成直道），只要平均值小于该值就为直道，大于该值则为普通弯道或大S形弯道，再根据平均值的大小和当前车速做加减速控制和舵机打角控制。

十字交叉道识别也较重要，该处处理不好赛车则不会按预定路线行进，图为我们处理前后的交叉道的情况：

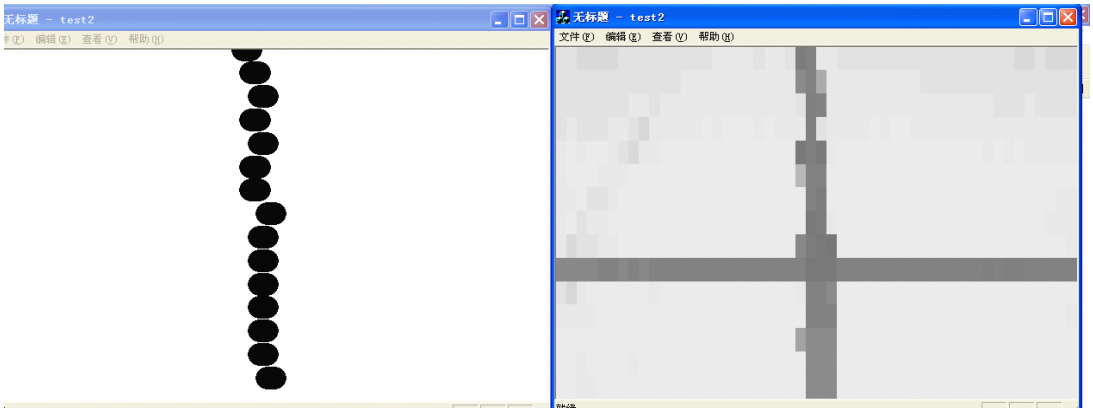


图5.7　路径识别前后的交叉道

第 3 章　　图像采集模块

小车最终方案选取了 CMOS 摄像头方案，CMOS 摄像头分为黑白和彩色，由于彩色摄像头的输出信号比较复杂，不易采集处理，比赛时只需分辨赛道的灰度，黑白摄像头已经足够。

所选取的 CMOS 摄像头基本参数为：

表 5.1

performance index / Spex and model	resolution ratie	supply voltage consumed power	illumination	output standard	lens focus and visual angle
SS2000B white and black series	350 line	6-9v 90mW	0.5lux	PAL/NTSC	3.6mm 69°

为使提高这个系统速度和易于数据采集及处理，在选取摄像头是，我们尽量采用线数低的摄像头。

3.1. 摄像头结构、工作原理和输出信号制式

1、摄像头结构

摄像头一般由镜头、传感器芯片、外围电路组成。传感器芯片不能单独工作，还需外围电路支持，才能将光信号转换成电信号。光线经物体发射，通过镜头聚焦，焦点恰好在传感器芯片的感光元件上，从而转换成电信号。输出信号制式为 PAL 电视机制式的摄像头，一般有四条线：正极，地线，音频输出线，视频输出线。CMOS 摄像头供电电压一般为 6-9V,视频信号峰峰值为 1V。

2、工作原理

针对于比赛，黑线吸收光线，将不会反射，传感器感光元件阵列相对应的感光元件不产生电压信号，即一个低压信号，而对应白线反射光线，通过镜头聚焦，在相对应的感光元件出产生一个相对高电压信号，一般峰峰值为 1V。

常用的摄像头视频输出信号是 PAL 电视机制式，它的工作原理与电视机的工作原理相似：在一定分辨率下，每秒扫描 25 帧图像，每帧图像含有 625 行信息，分为奇、偶场，进行个隔行扫描，总共每秒 50 场信号，每场有 312.5 行信息，从奇数行开始扫描，即依次扫描第 1、3、5、7、9……，当扫描完奇数场后，再开始扫描偶数场，构成一帧图像。如图所示：

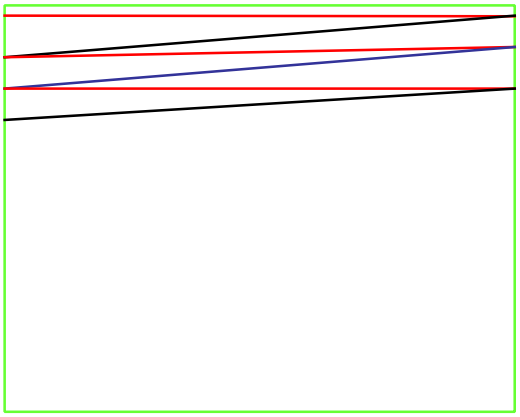


图 5.1　奇数场

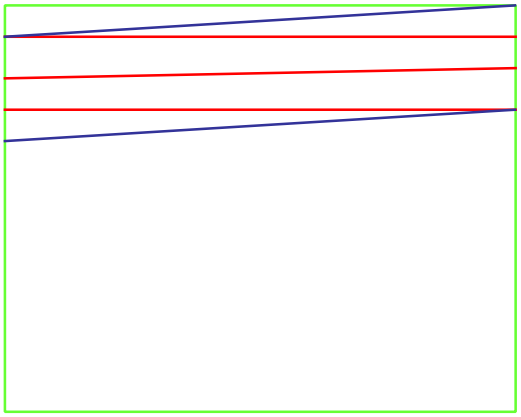


图 5.2　偶数场

3、输出图像结构

如图 1 、 2 所示，每一行有正程和逆程，每一场有正程和逆程，逆程是每一行和每一场的消隐，行消隐是电视机的电子枪从电视机屏幕的最左端回到最右端所需的时间，场消隐是电视机的电子枪从电视机屏幕的最末段到起始段所需要的时间。

输出信号主要包含三个部分：

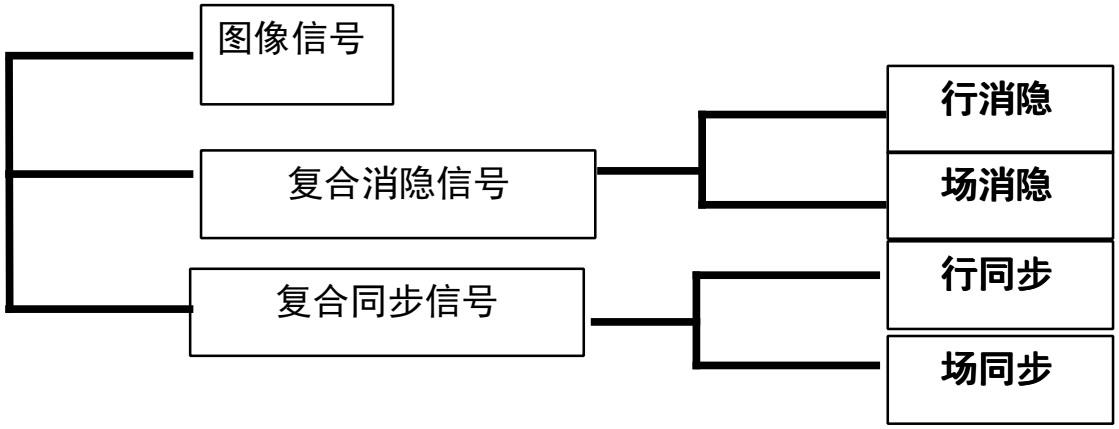


图 5.3　输出信号组成部分

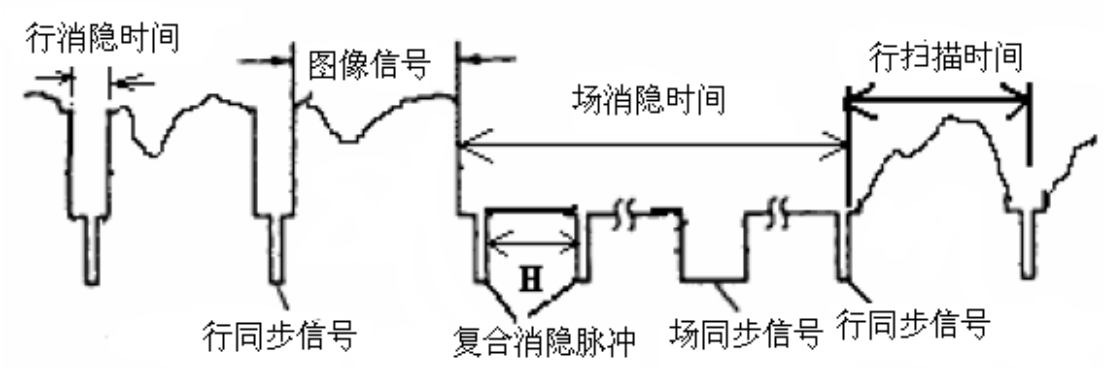


图 5.4　输出信号波形组成

图像信号、复合消隐信号、复合同步信号，同时其中包括行消隐、场消隐、行同步、场同步。

行信号

每一行各个信号时间为：

行扫描时间为	64 μs
图像信号	52.2 μs
消隐信号	11.8 μs +250 ns
行同步信号宽度	4.7 μs+100 ns
行延迟	1.3 μs+250 ns

如图所示：

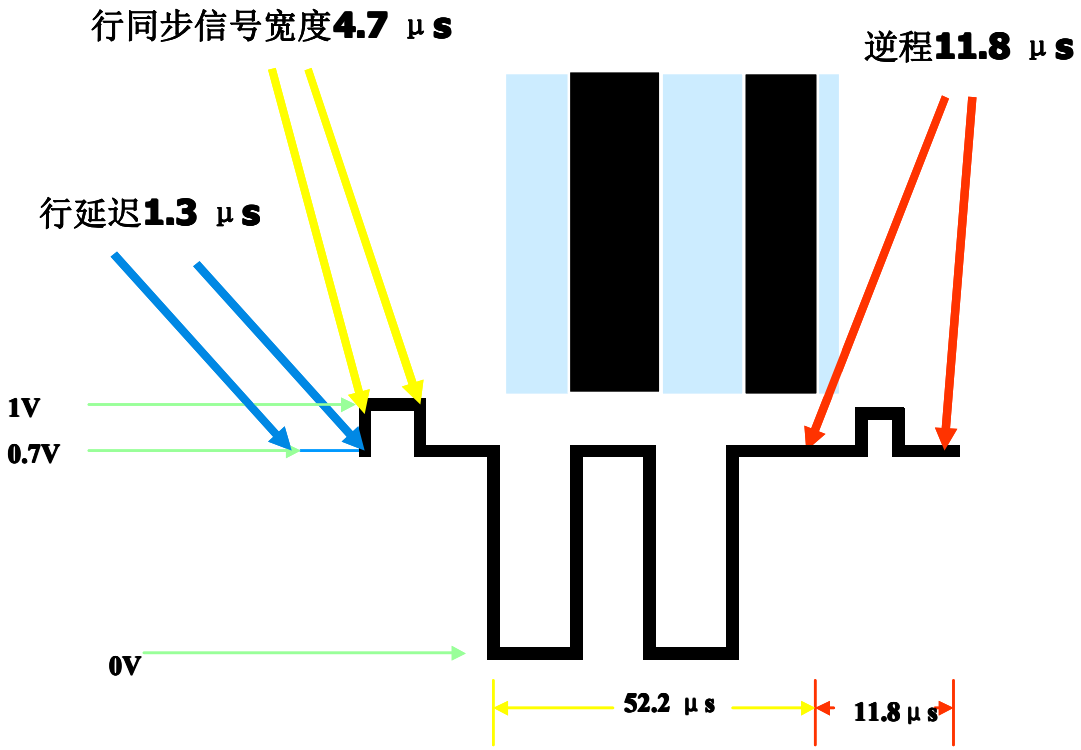


图 5.5　行信号分析图

场信号

每一场信号包括 312.5 行， 其中有场消隐信号和场同步信号，各个信号的时间计算如下：

一帧 = **奇数场** + **偶数场**

625 行= 312.5 + 312.5

场消隐信号 = 25 × 64 + 11.8

= 1600 + 11.8

= 1611.8 μs

场同步 = 3 × 64

= 192 μs

信号峰峰值

标准电视机信号峰峰值为 1V，以同步信号作为 100%，黑电平和消隐电平为 70%，

白电平为 0%，图像信号介于白电平和黑电平之间，根据图像的灰度而变化。

在标准的 1V 全电平信号中

同步信号　0.7V--1V　0.3V+9mV

图像信号　0V--0.7V　0.7V+20mV

通过示波器观测到的波形是其倒像，

同步信号　0.0V--0.3V　0.3V+9mV

图像信号　0.3V--1V　0.7V+20mV

4、检测黑线原理

路径识别中，黑线和白色底板存在很多大灰度比，在图像信号上会形成相应高低不同的电压值，当检测到黑线是，图像信号中将形成一个“凹”形槽，通过 DG128 对信号进行采集，并对数据处理，就可以得到黑线相对小车的位置，一帧完成后，能计算出路径的整个斜率和偏离量。

3.2.CMOS 摄像头的选取

在掌握 3.1 中 CMOS 摄像头的基本知识后，我们到市场上选购所需的摄像头。目前，市场上的摄像头一般都是在 320 线以上，而根据 MC9S12DG128 Datesheet 中的说明，DG128 的 AD 最高采集频率为 2MHz，可以采集到的像素点有：

$$64/（12 \times 1/2）= 10 \quad （个）$$

由计算可知，在最高采集频率，8 位 AD 的情况下，只能采集到 300 以上个像素中的 10 个，横向上的分辨率十分低，不足以满足识别路径的要求。所以在选购 CMOS 摄像头时，尽量选取像素较低的，同时考虑到功耗，我们选购了 6-9V 供电的 350 线 SS2000B 黑白摄像头。

3.3.LM1881 视频分离模块

MC9S12DG128 要对摄像头输出的视频信号进行采集，必须将信号中的行同步信号和场同步信号分离出来，作为中断，使 MC9S12DG128 AD 模块采集视频信号。

LM1881 可以从摄像头的视频信号中分离出行同步、场同步信号、奇偶场信号等。LM1881 引脚图如下：

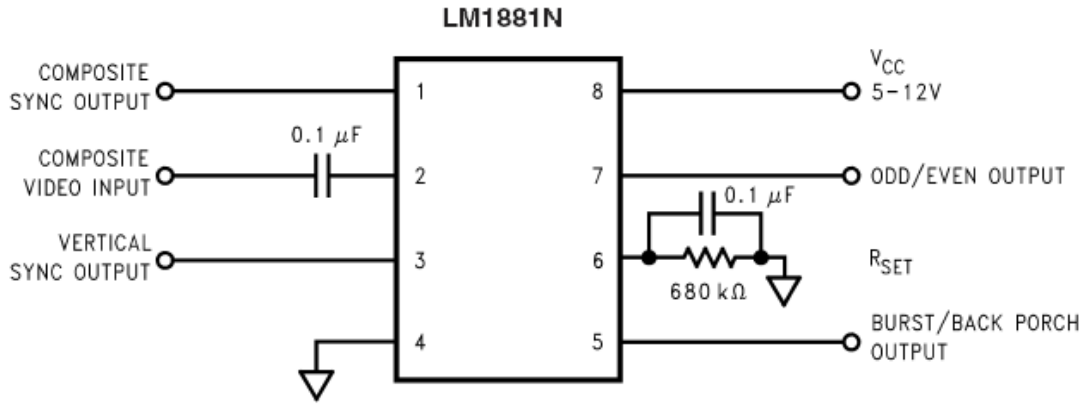


图 5.6　LM1881 引脚图

引脚 2 位视频信号输入端，经 LM1881 芯片处理，引脚 1 输出行同步信号，引脚3 输出场同步信号，引脚 7 输出奇偶场信号，引脚 5 输出滞后行信号 5.8μs 的后肩区同步信号。它们的输出波形如下：

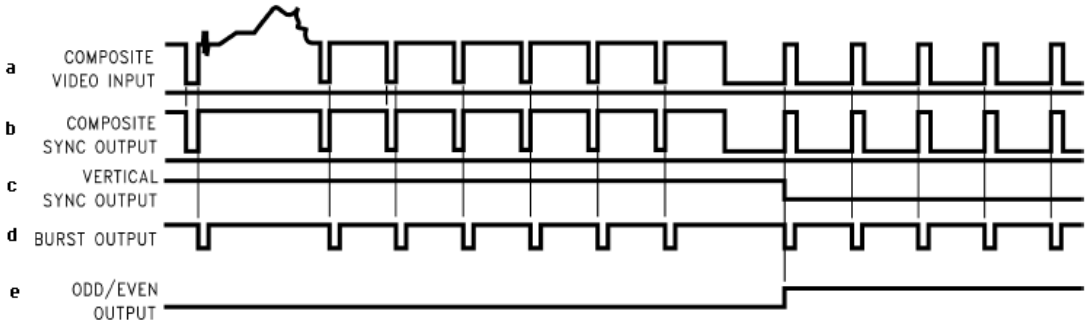


图 5.5　视频分离信号

（a）视频信号　（b）行同步信号　（c）场同步信号

（d）后肩区同步信号　（e）奇偶场同步信号

行同步信号对应与视频信号，它是判断是否进行 AD 采集的标志，图像信号滞后行同步信号的上升沿 5.8μs；后肩区同步信号上升沿滞后行同步信号 5.8μs，所以它与图像信号同步。奇偶场同步信号中，偶场为低电平，奇场为高电平，场同步信号的下降沿与奇偶场同步信号上升沿同步。我们可以用场同步信号或奇偶场同步信号作为场的变换中断，用行同步 信号或后肩区同步信号作为行变换的中断。

3.4.SS2000B 黑白摄像头测试

购买 SS2000B 黑白摄像头后，我们对它的输出信号参数进行了相应测试。将摄像头视频输出接在 LM1881 引脚 2 上，示波器探头分别悬挂在 LM1881 引脚 1、3、5、7 进行测试，测试结果如下表：

表 5.2

行信号时间	64μs
行同步脉冲时间	4.4μs
图像信号滞后行脉冲时间	5.8μs
场信号时间	19.8ms
图像信号起始行	第 25 行
图像信号截止行	第 300 行

在程序测试中，发现当场中断产生时，行中断标志从零开始计数，当行中断标志为 23 时开始采集的数据才为有效数据，当行中断标志为 300 时，为有效数据的截止行。

3.5.视频信号数据采集

3.5.1. 视频采集电路

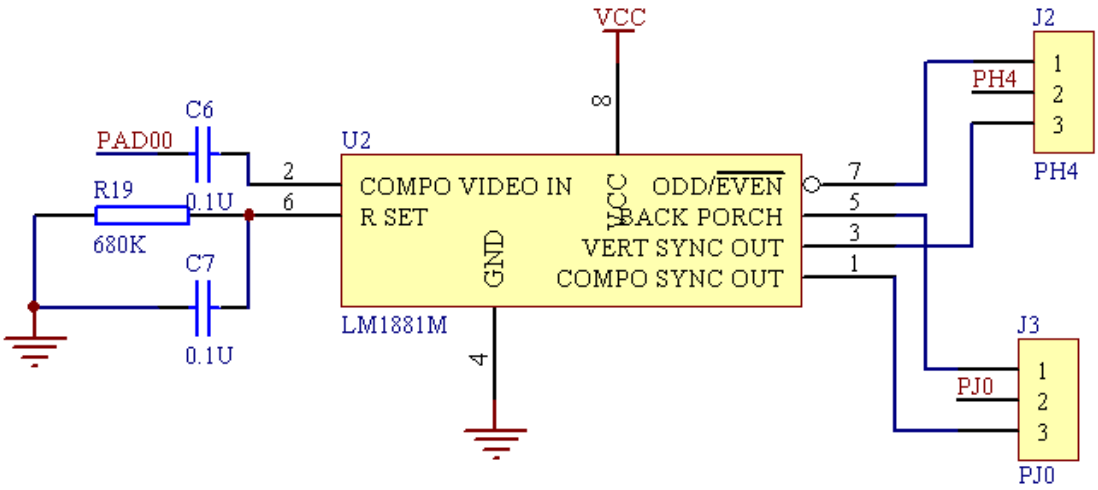


图 5.8 视频采集电路

LM1881 提取出行同步信号和场同步信号，将其作为 MC9S12DG128 AD 采集的中断，AD 模块能正确采集图像信号。LM1881 与 MC9S12DG128 的电路连接如图 5.8 所示：

视频信号接在 LM1881 的视频输入端，同时接在 MC9S12DG128 AD 采集端口（PAD0 口）。如图所示，我们将能产生场中断的场同步信号和奇偶场同步信号分别连接到 J2 的 1、3 插针上，2 号插针与芯片的 PH4 口相连接，将能产生行中断的行同步信号和后肩区同步信号分别连接到 J3 的 1、3 插针上，2 号插针与芯片的 PJ0 口相连接。在该方案中我们使用场同步信号作为场中断，行同步信号作为行中断（用跳线冒短接 J2 的 2、3，用跳线冒短接 J3 的 2、3），因为图像信号滞后行同步信号 5.8μs，当行中断产生时，需要运行一部分判断程序，大概需要 5.8μs，当运行完这部分程序后，恰好打开 AD 中断，对视频信号进行采集，即采集到有效图像信息。

在这里 PH4 I/O 口 PJ0 I/O 口都以中断的方式接收 LM1881 分离出的行同步信号和场同步信号，这样很方便的就能使芯片对视频信息中的图像信号进行准确的采集。

3.5.2. 软件采集流程

纵向采集个数测试

由 3.4 中的测试可得有效视频信号是从第 25 行开始，到第 300 行结束，如果采集每一行信息，将超出 MC9S12DG128 的处理能力和存储空间。同时在纵向不需要怎

么高的分辨率。同时我们对 2.5cm 宽的黑线采集点数进行了测试。测试方法如下：

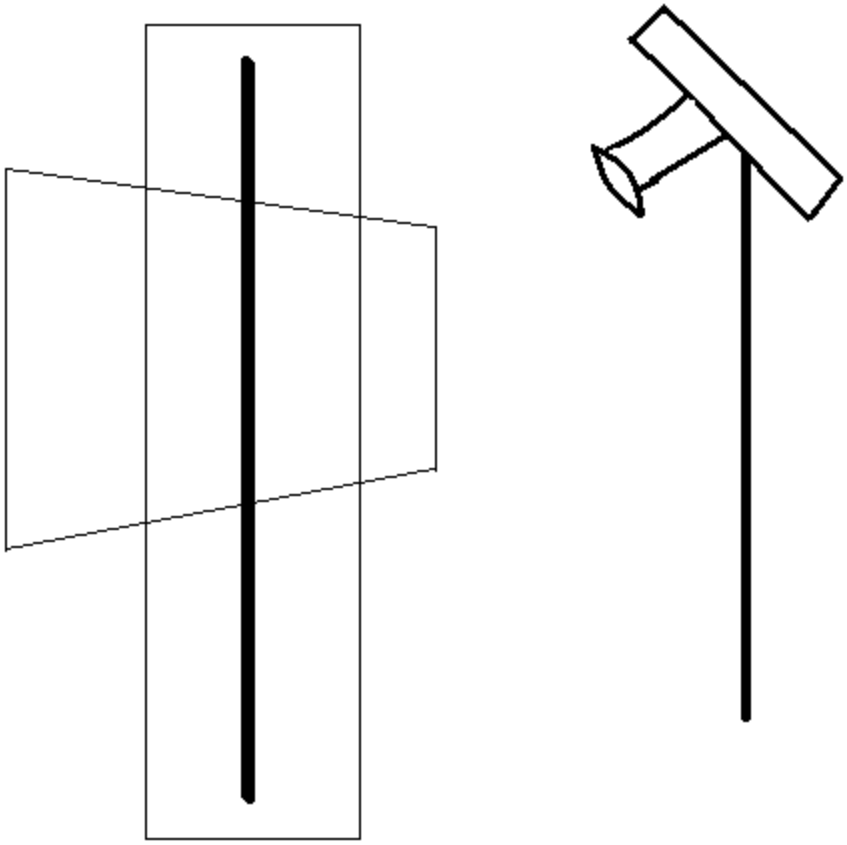
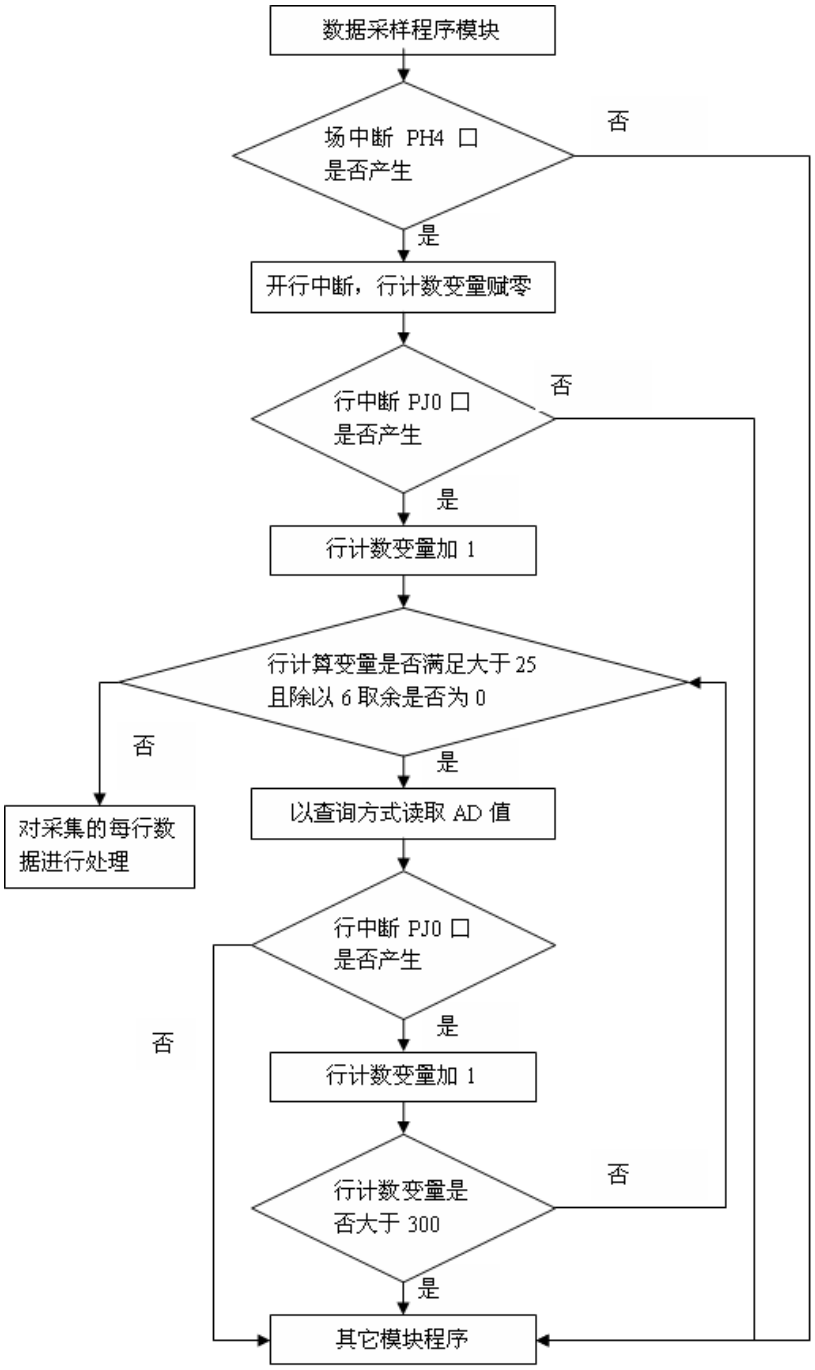


图 5.9　采集示意图

将贴有黑线的跑道与摄像头的纵向垂直，对每行视频信号信号进行采集，经采集分析，总共有 8 行数据采集到 2.5cm 的黑线，故我们编写程序从第 25 行开始每 6 行采集一次，一共能采集 46 行有效数据。

数据采集流程：



如上面的流程图所示，整个程序中，场信号和行信号都以中断的方式产生，这样可以优化程序的运行时间。当 PH4 端口产生场中断时，运行场中断函数，行计算变量清零，打开行中断，若 PJ0 端口产生了行中断，执行行中断函数，行计算变量加一，判断行计算变量是否满足大于 25 且除以 6 取余是否为 0，若满足说明可以对视频信号进行采集。程序中采取在行中断中以查询的方式采集视频信号，而不使用 AD 中断方式对数据进行采集，因为用 AD 中断方式将增多中断数量，各个中断的时

序不以控制，以致于出现错误采集、漏采集数据的情况，同时可以提高 AD 采集的速度。当采集完以后数据后，每隔 6 行中间大概有 0.3ms 的时间，在这段时间里，我们用来对采集的行信号进行数据处理，提取黑线位置，并存储在一个一维数组中。在一场中，行中断产生一次行计数变量增加 1 一次，一直到行计数变量大于 300 为止，当其大于 300 时，说明该场结束，进入面处理函数模块，对采集到的 46 行黑线位置的数据进行处理，并控制舵机、后轮电机运作。之后，准备下一场数据的采集。

3.6.图像处理与路径识别别算法

图像处理与路径是别是整个算法的核心和重点，能否在短时间内将前方距离较远的黑线准确识别出来，对于高速度行驶状态下的控制决策给出是至关重要的。摄像头在这方面较红外有着很大的优势。

图像识别与处理步骤[1]：
1. 图像的采集：

赛道信息有普通的 COMS 摄像头采集回来,经过 1881 的视频信号分离和单片机自身的 AD 转换,将赛道的图像信息转化为内存当中的一个数组，考虑到图像的分辨率，AD 转化速度和算法处理速度等等多方面因素，我们将这个数组定义为 43 * 23.

2. 图像平滑（去噪）：

用信号处理的理论来解释，我们的做法实现的是一种简单的低通滤波器（low pass filter）。在灰度连续变化的图象中，如果出现了与相邻像素的灰度相差很大的点，比如说一片暗区中突然出现了一个早点，这种情况被认为是一种噪声。灰度突变在频域中代表了一种高频分量，低通滤波器的作用就是滤掉高频分量，从而达到减少图象噪声的目的。

为了方便的叙述上面所说的“将原图中的每一点的灰度和它周围八个点的灰度相加，然后除以 9，作为新图中对应点的灰度”这一操作，我们用如下的表示方法:

图 4 1 模板表示方法

这种表示方法有点象矩阵，我们称其为模板(template)。中间的黑点表示中心元素(此处用*表示)，即，用哪个元素做为处理后的元素。例如 表示将自身的 2 倍加上右边的元素作为新值，而 表示将自身加上左边元素的 2 倍作为新值。通常，模板不允许移出边界，所以结果图象会比原图小。模板操作实现了一种邻域运算

(Neighborhood Operation)，即，某个像素点的结果不仅和本像素灰度有关，而且和其邻域点的值有关。在以后介绍的细化算法中，我们还将接触到邻域运算。模板运算的数学涵义是一种卷积（或互相关）运算。这样可以滤除图像中大部分的噪点。

同时，我们根据赛道的具体信息进行虑噪：图像中黑线必然是连续的，上一行中黑线位置和本行黑线位置相差太远就认为本行信息是错误的，将其保持为上一行的信息。事实证明这种算法可以比较好的适应赛道的具体情况并产生良好的虑噪效果。

3. 畸变校正：

由于摄像头看赛道是有一定角度的，实际上是把赛道上一个梯形映射为一个矩形存储到单片机中了，畸变校正的目的就是将原来的梯形还原出来。畸变校正实际上是一种坐标系的转换。考虑到图形从上到下基本是线性的畸变，所以只要每行的点的横坐标乘以不同的系数就可以达到畸变校正的目的。如我们的设想头看到梯形的上底边 80cm,下底边 35cm 只要将最下面的点的横坐标 乘以 16，最上面点的横坐标乘以 7，并由线性算出中间点的横坐标乘的系数即可将图像中的点较好的还原到原来的位置。当然，算法中没有考虑纵坐标的畸变，是有一定误差的，但是这种误差是控制算法可以接受的。

4. 图像边缘提取：

识别一个对象是从其边缘开始的，一幅图像不同部分的边缘是模式识别最重要的特征。在边沿检测中，常用的一种模板是 Sobel 算子。Sobel 边缘算子是一种一阶差分算子，它可以有效地消除道路图像中的大部分无用信息离散 Sobel 算法的定义如下:

$$\begin{cases} \nabla_x f(x,y) = [f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)] \\ \qquad\qquad -[f(x-1,y-1)+2f(x,y-1)+f(x+1,y-1)] \\ \nabla_y f(x,y) = [f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)] \\ \qquad\qquad -f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)] \end{cases}$$

公式 4 1 离散 Sobel 算法

为了编程容易实现，将 Sobel 算子转换为两个卷积核，分别为水平核和垂直核，图像中的每个点都用这两个核做卷积，两处卷积的最大值作为该点的输出值，运算

结果即为经过边缘增强的图像。

4. 计算黑线位置：

提取出图像的边缘以后只要将每行图像的边缘进行平均就是

图像中黑线的重心位置(Average[23])。能够准确的辨识黑线位置对于下面控制量的给出是至关重要的。

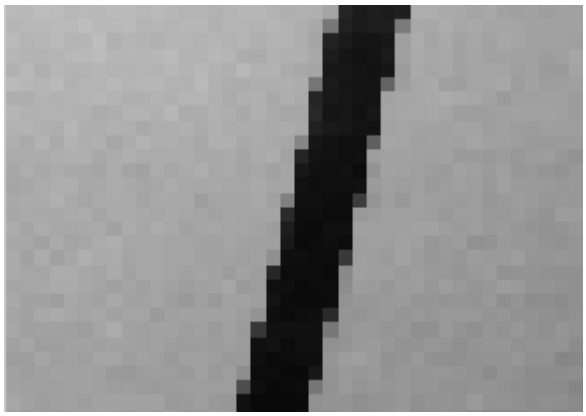


图 4 1 采集到的图像

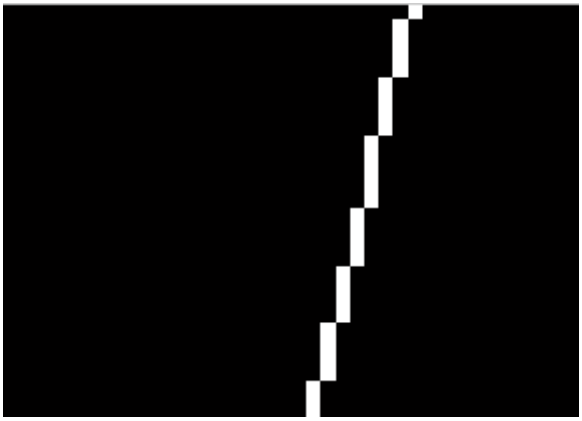


图 4 2 经过处理后的图像

3.7. 摄像头图像数据采集以及处理(上海交大 1 部分)

（1） 图像数据获取

图像数据以黑线位置的形式获取。具体流程图如下：

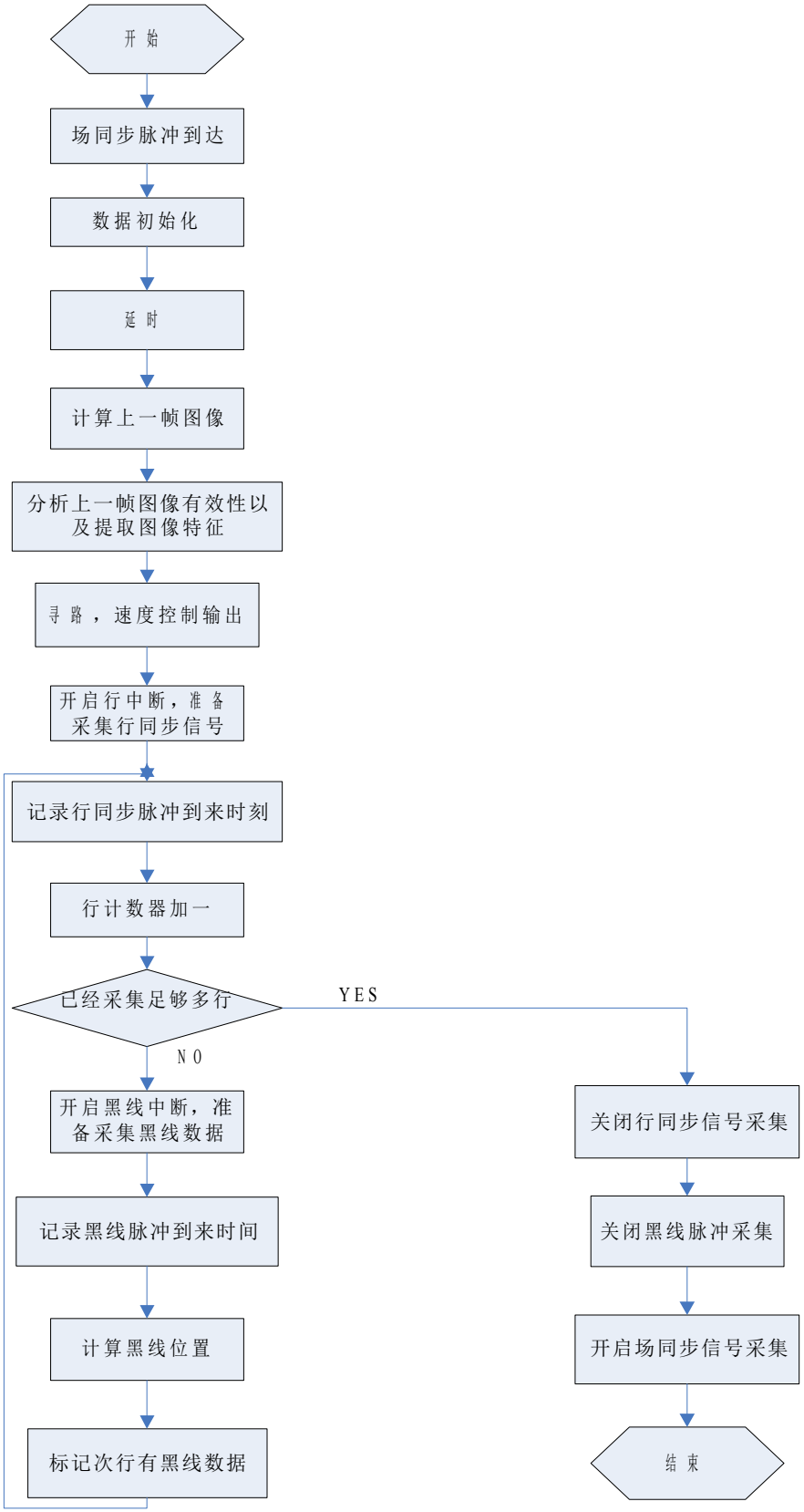

开 始
场同步脉冲到达
数据初始化
延 时
计算上一帧图像
分析上一帧图像有效性以及提取图像特征
寻 路 ，速度控制输出
开启行中断，准 备采集行同步信号
记录行同步脉冲到来时刻
行计数器加一
已经采集足够多行
YES
NO
开启黑线中断，准备采集黑线数据
记录黑线脉冲到来时间
计算黑线位置
标记次行有黑线数据
关闭行同步信号采集
关闭黑线脉冲采集
开启场同步信号采集
结 束

图 4.3　图像获取流程图

具体程序及注释如下：

A．场中断服务程序

场中断服务流程图如下图所示：

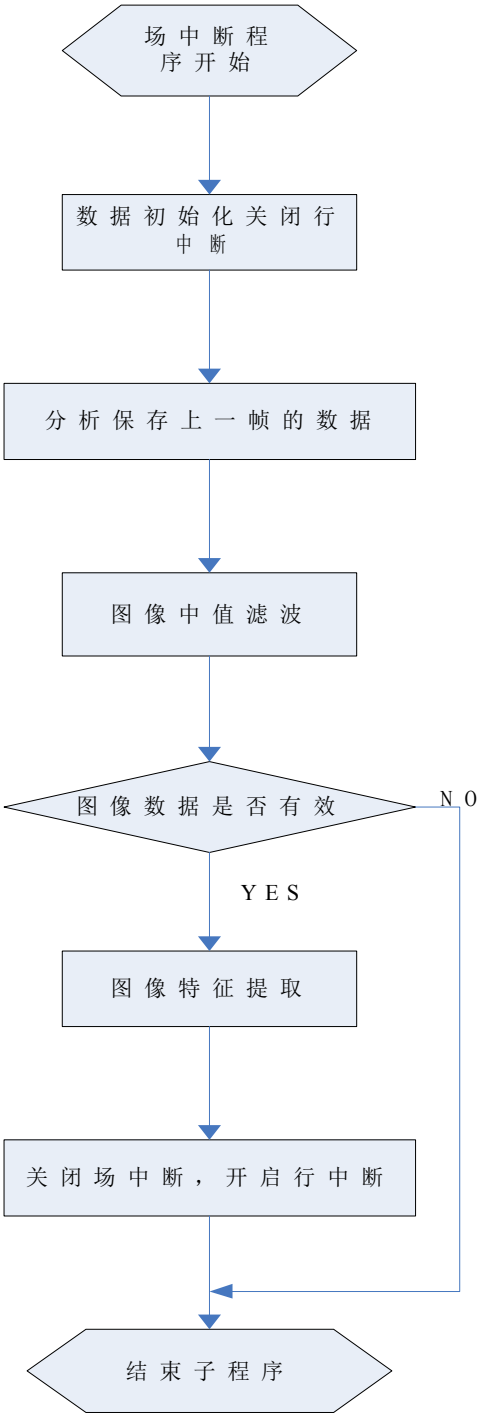


图 4.4　场同步信号中断服务流程图

当新的一场到来时，ECT 已经将脉冲到来时刻时的计数器值送入了 TC6 寄存器里而，由于处理算法在开始时并不关心这个值具体为多少，所以，在程序的第一步，应当将中断标志清除，以防止再次重复的相应这一个中断。在初始化时标识位快速清零有效的情况下，只需要读出 TC6 的值即可。随后，由于所需要的数据不是在这一帧的开始，所以处理算法主场同步开始后获取的数据，因此在此期间关闭行中断，同时调用处理算法完成对上一帧图像的处理以及寻路、速度控制算法的计算、决策等等。但是这一部分不属于图像的采集，所以该部分将会在以后阐述。

经过一定时间后，电视信号已经扫描到了采集位置，此时打开行脉冲信号的所在通道的中断使能，准备采集行脉冲到来时的时间值。与此同时，在程序里而设定了一个行计数器，用于记录当前所得到的行信息是在那一帧里而开始采集以来的第几行。在场同步信号的脉冲造成的中断服务程序中，需要对此记数器做清零处理。为了保险起见，场中断使能在采集到了足够的数据以后才重新打开。

B．行中断服务程序

行中断服务程序流程图如下所示：

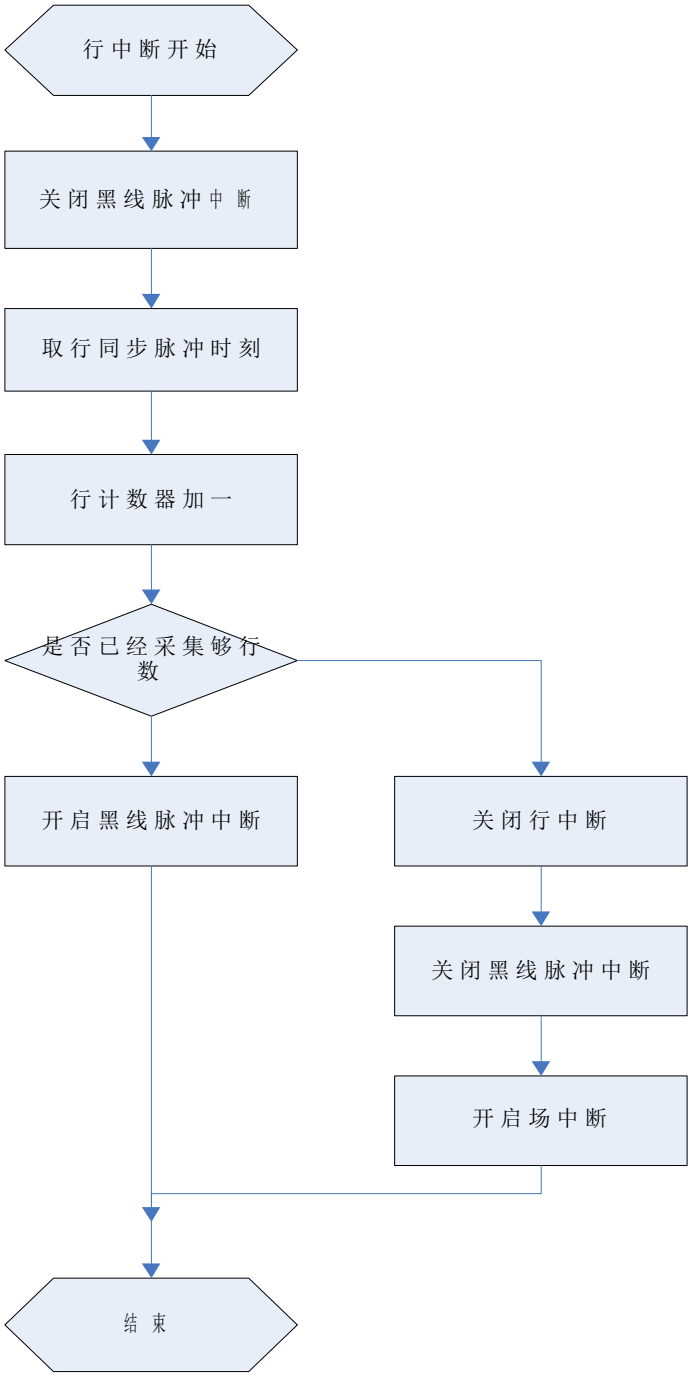


图 4.5　行中断程序服务流程图

此时，ECT 将脉冲到来时刻的系统主定时计数器值送到了 TC4 寄存器当中。当保存这个值，系统自动清除了行中断标记。为了回避初期的信号毛刺，我们仍然需要暂时的关闭黑线采集的中断，在很短的一段时间过后再次打开。另外，在采集黑线行信号计数器中做加一的处理，区别同一帧中不同行的信息。由于本方案并不需

要采集图像信号的全部，在场中断服务程序中回避了一帧中前而的若干行以后，当行中断服务程序中判断当前行计数器值已经超过了采集范围，通过关闭行中断使能来回避后而的若干行。最后，程序打开黑线脉冲信号所在通道的使能，逐个采集黑线脉冲。

C．黑线脉冲中断服务程序

黑线脉冲中断服务程序流程图如下：

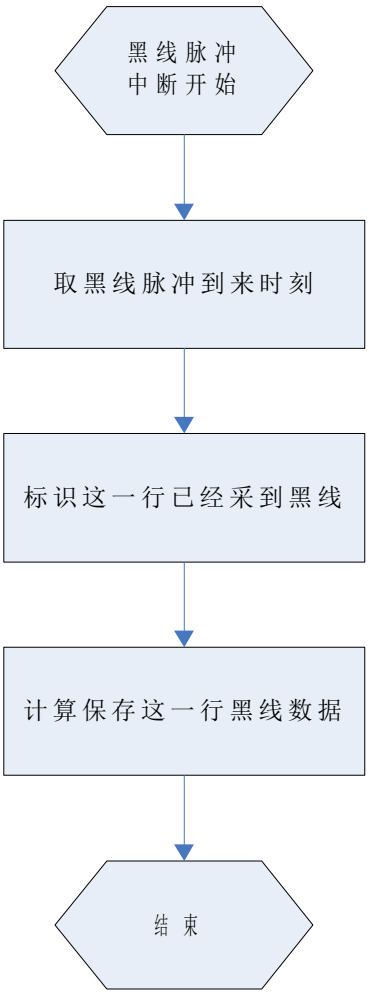


图 4.6　黑线脉冲中断服务程序流程图

与行中断情况相同，ECT 将黑线脉冲到来时刻的系统主定时计数器的值送到了 TCO 寄存器当中。将该值之与在其以前发生的相邻的行同步脉冲到来时间做减法，即得到在当前行中黑线所在的位置。不过这样做有一个问题，主定时计数器可能在自由运行时，可能会在行同步信号与黑线脉冲之间的这段时间段内溢出，这样的话如果直接做减法会得到错误的数据，所以在程序中先要做个判断。正常情况下，黑

线脉冲时间值应当大于对应行同步信号时间值，此时直接相减。发生溢出时，黑线脉冲小于行同步脉冲，黑线位置时间值应当为：

黑线位置时间=计数器最大值—行同步信号时间值+黑线脉冲时间值。

此外，摄像头在这一行没有照射到黑线的时候，在这一行的有效时间内没有黑线脉冲，小会运行中断服务程序。图像数据的变量值不会刷新，这样会造成在获取图像时的错误。所以在每一行的中断服务程序里而（包括行同步脉冲中断服务程序与黑线脉冲中断服务程序），我们设定了一个全局的标记，当这一行产生了黑线中断时，标记有效，表示这一行有黑线，这样在每一帧结束的时候通过判断这一标记的有效性，就可以对那些没有通过黑线脉冲中断服务程序做出刷新的数据值做刷新处理。

本方案采用了一个很简单的鉴别算法。因为系统采用的是存储黑线的位置信息，而不是每一帧的象素信息，所以数据量得到了大大的减少。在这种情况下，在程序中存储了当前帧之前的 3 帧的信息。鉴于发生上述的将赛道边沿误判断成黑线的情况，黑线数据一定是从图像的一边逐渐消失，由突然从图像的另外一边出现。所以，才发生错误时，摄像头所认定的黑线位置与传感器探测到的黑线位置一定是相反的。所以在程序算法中加入了一个判断当前黑线方向的标志，通过判断这个标识，可以提出坏帧。另外，在正常情况下，由于车的连续运动，我们连续采样，而黑线又一定是连续的。所以可以认定，图像的相邻两帧的差距不应当很大，只需要对其做一个差值，对于变化过大的相邻两帧数据我们都认定为无效，这样就可以很简单的剔除了坏帧，判断出黑线数据的有效性。

3.8. 路径信息采集处理方法

智能车通过 CCD 视频传感器采集到的路径信息都是模拟值，然后将一场数据的模拟值通过 MC9S12DG128 的 ATD0 通道转换成数字量，这样，这一场的数据就变成了一个 260*33 的二维矩阵，其中的每个数字代表着一场图像中对应位置点的象素灰度值。但是，单片机处理这么多的数据会十分消耗资源，而且单片机的内存也不可能给这个存放图像数据的二维矩阵分配太多的存储空间。所以，为了节省单片

机的资源和减少处理时间，就必须对每场的图像做一些处理，本队采取的方法是，每隔 10 行进行一次处理，这样，该场图像的数据就变成了一个 26*33 二维矩阵。图 5.3 就是一条直线的二维矩阵图。

```
31 31 31 31 31 39 39 39 43 44 48 48 49 51 51 31 31 48 48 48 44 44 44 44 42 44 44 42 39 39 31 31 31
31 31 31 31 39 39 39 43 44 48 48 49 51 51 51 31 31 49 49 48 48 44 46 44 44 44 44 42 41 39 39 31 31
31 31 31 39 39 41 42 44 46 48 48 51 51 51 52 31 31 51 49 48 48 48 48 48 48 48 46 44 43 41 41 31 31
31 31 39 39 41 42 44 46 48 48 49 51 52 53 56 31 31 51 51 49 48 48 48 48 48 48 48 48 44 44 41 39 31
39 39 39 39 41 44 44 46 48 49 51 52 54 54 54 31 31 51 51 50 49 49 48 48 49 48 48 48 46 44 44 41 39
39 39 42 41 44 44 46 48 49 49 51 54 54 56 56 31 30 53 51 51 49 49 49 49 49 48 48 48 46 44 44 44 39
39 41 41 41 42 46 46 48 49 51 51 54 54 56 56 31 30 54 51 51 51 51 49 49 49 49 49 48 48 46 44 44 43
39 40 44 44 44 46 48 48 48 51 51 53 54 56 56 30 31 52 51 51 51 49 49 51 51 51 49 48 48 48 46 44 44
41 40 41 43 44 44 46 48 49 50 51 51 54 56 56 30 30 52 51 51 51 49 50 49 51 49 49 48 48 46 44 44 44
41 41 39 41 42 44 44 48 48 49 51 51 52 54 54 30 30 54 51 51 51 51 49 50 51 51 49 48 48 48 46 44 43
39 39 41 41 41 44 44 48 48 49 49 51 52 52 52 28 28 51 51 51 51 49 49 51 49 49 49 48 48 46 44 44 43
39 39 39 39 39 39 44 44 48 48 49 51 51 51 51 28 28 51 49 49 49 49 48 48 49 48 48 48 48 44 44 42 41
31 31 31 39 39 39 41 44 44 48 48 49 49 49 48 28 26 49 48 48 48 48 48 48 48 48 48 48 46 44 44 43 39 39
31 31 31 31 39 39 39 41 44 44 48 48 48 48 48 25 25 48 48 48 48 48 48 48 48 48 46 46 45 44 44 41 39 39
31 31 31 31 39 31 39 39 42 44 46 48 48 48 44 27 24 41 48 48 48 48 46 46 46 46 44 44 44 41 41 39 39
31 31 31 31 39 39 40 39 41 44 46 46 48 48 44 24 24 31 48 46 46 46 48 46 46 44 44 43 44 41 39 39 39
31 31 31 31 31 39 39 39 44 44 44 44 48 48 39 24 24 30 48 48 48 46 46 46 44 44 44 44 44 39 39 39 39
31 31 39 39 39 39 39 39 42 44 44 44 46 48 30 24 24 31 46 48 46 46 46 48 44 44 44 44 43 41 41 39 31
31 39 39 39 39 39 39 41 44 44 44 46 46 48 28 24 24 27 46 48 48 48 44 48 46 48 44 44 44 41 41 41 39
39 39 39 39 39 41 41 44 44 44 44 46 46 48 24 24 24 26 48 48 48 48 48 46 46 44 44 44 44 42 43 41 39
39 39 39 39 40 41 44 42 44 44 46 46 48 46 23 24 22 24 48 48 48 48 46 46 46 46 44 44 43 44 42 39 41
39 39 39 41 41 41 44 44 44 44 44 46 46 48 23 22 22 24 48 48 48 48 46 45 46 46 44 44 44 43 42 41 41
41 39 39 41 43 43 42 43 44 44 44 44 48 46 22 22 22 24 46 48 48 48 46 44 48 44 44 44 44 43 42 43 41
39 40 41 41 41 41 44 41 44 45 46 44 46 45 21 22 22 24 48 48 46 48 46 44 44 44 44 43 43 42 43 42 39
39 39 40 39 39 41 41 41 43 44 44 44 44 44 22 21 21 24 44 44 44 44 44 44 44 44 44 44 43 41 41 41 41 39
39 39 39 39 39 40 40 41 41 41 44 44 44 44 21 21 20 22 44 44 44 44 44 43 44 43 41 41 39 39 39 39 39
11167 31 39 39 39 39 41 41 41 42 42 41 42 20 20 20 22 41 41 43 41 41 41 42 39 39 39 39 31 31 31 31
```

图 5.3 直线赛道图像数据图

接着，就要对 CCD 视频传感器采集的视频数据进行路径判断处理，这时，需要确定视频阈值的大小。如果象素点灰度值小于该阈值，则确定该点为黑点，否则，确定其为白点。这样，就可以在一场数据中分辨出象素点是黑色还是白色。处理后的结果如图 5.4 所示。

```
39 39 39 41 41 42 44 45 46 48 48 51 51 52 54  0  0 51 50 49 48 48 48 48 48 48 48 46 44 44 41 39 31
31 39 39 41 43 44 44 46 46 48 51 51 54 54 54  0  0 51 50 48 48 48 48 48 48 48 48 44 44 41 39 39
39 39 39 39 41 44 44 46 48 49 49 51 54 54 54  0  0 51 51 49 48 48 48 48 48 48 48 46 44 44 39 39 31
31 39 39 39 39 41 44 44 48 48 49 51 54 54 54  0  0 51 51 49 48 48 48 48 48 48 48 44 45 44 41 39 31
31 31 31 39 39 41 44 44 48 48 49 51 51 54 54  0  0 51 49 48 48 48 48 48 48 48 48 44 44 44 41 39 39
31 31 31 31 39 39 41 44 48 48 49 51 52 54 54  0  0 51 49 49 48 48 48 48 48 48 48 44 44 42 39 31 31
31 31 31 31 39 39 42 44 48 48 49 51 51 53 53  0  0 52 49 48 48 48 48 48 48 48 48 46 44 41 39 39 39
31 31 31 39 31 39 44 44 48 48 49 51 53 52 52  0  0 44 51 48 48 48 48 48 48 48 48 48 44 44 39 39 31
31 31 31 31 40 39 41 46 48 48 49 51 51 52 53  0  0  0 49 49 48 48 48 48 48 48 48 48 44 44 41 39 39
39 39 39 39 41 41 44 46 48 48 49 51 51 52 52  0  0  0 49 49 48 49 48 48 48 49 48 48 44 44 44 41 41
39 39 39 40 41 43 44 46 48 48 49 51 51 53 52  0  0  0 51 49 49 49 49 49 49 49 48 48 48 46 44 43 42
39 41 39 41 41 44 44 46 48 49 49 51 51 51 51  0  0  0 51 51 49 49 49 49 49 48 48 48 48 46 46 44 43
40 41 40 41 44 44 45 46 48 49 49 51 51 51 51  0  0  0 51 51 49 49 49 49 49 49 48 48 48 48 44 44 43
41 41 41 42 44 44 44 48 48 48 49 51 51 51 51  0  0  0 51 51 51 49 49 49 49 49 48 48 48 48 46 45 44
41 43 44 44 44 44 44 48 48 48 49 49 50 51 51  0  0  0 51 51 49 49 49 49 49 49 48 48 48 48 46 46 44
41 41 41 44 44 44 44 46 48 48 48 48 51 49 49  0  0  0 49 49 49 49 49 49 49 49 48 48 48 48 48 44 44 44
39 41 41 44 42 44 46 44 48 46 48 48 48 49 49  0  0  0 48 49 48 48 48 48 48 48 48 48 48 46 44 44 43
39 39 39 40 43 42 42 44 44 48 48 48 48 48 48  0  0  0 48 48 48 48 48 48 48 48 48 48 44 44 44 41 41
39 31 39 39 39 39 41 41 44 44 44 45 48 48 48  0  0  0 48 48 48 46 46 48 48 44 44 44 44 43 40 40 41
31 31 39 39 39 39 39 39 41 42 44 44 44 46 44  0  0  0 44 48 44 46 44 44 44 44 43 43 42 41 39 39 39
31 31 31 31 39 31 39 39 41 41 41 44 44 44 44  0  0  0 44 44 44 44 44 44 44 41 42 39 39 39 39 31 31
31 31 31 31 31 31 39 39 39 39 39 41 44 41 41  0  0  0 42 43 41 44 41 42 41 39 41 39 39 31 31 31 31
31 31 31 31 31 31 31 39 39 39 39 41 41 41 39  0  0  0 39 39 41 41 41 40 39 39 39 39 39 31 31 31 31
31 31 31 31 31 31 39 39 39 39 39 39 39 40 40  0  0  0  0 41 41 41 39 39 39 39 39 39 31 31 31 31 31
31 31 31 31 31 31 39 39 39 39 41 39 41 39 39  0  0  0  0 39 41 39 41 41 41 39 39 39 39 31 31 31 31
31 31 31 39 39 39 39 39 39 39 41 39 41 41 39  0  0  0  0 41 41 41 41 41 41 39 41 39 39 39 39 31 31 31
11167 39 39 39 39 39 40 41 41 41 41 42 44 39  0  0  0 31 42 42 41 41 41 40 41 39 39 39 39 39 31 31
```

图 5.4 处理后的直线赛道图像数据图

由图 5.3 可知，图像近处的灰度阈值为 31，所以，灰度值小于 31 的点都被判定为黑点，为了调试方便，把黑点对应的灰度值都置为 0。但是，在上数第 8 行时出现了 31 的灰度值，如果按照近处的灰度阈值来判断的话，该点会被判定为白点，而实际图像中该点的位置却为黑点。因此，为了得到正确的判定结果，可以将远处（图 5.3 中为第 8 行以上）的数据的灰度阈值稍作调整，将其加 1，变为 32，这样，远处的点也可以被判定为黑点，也就符合实际图像中的情况了。这里提到的“远处”开始的位置是通过实践测定的结果，是一个经验值。这也是灰度阈值自适应的一种实现方法。

在确定某点是黑（白）点之后，就可以确定黑线的位置了，下面介绍一下具体的实现方法。

因为每行有 33 个象素点，所以把这 33 个点分别标号，记为 0～32。第一次处理的时候，先按照从左向右的顺序，进行比较，当遇到第一个黑点的时候，记下该点的序号 A，接着继续比较，当遇到第一个白点的时候，也记下该点的序号 B，其后的数据可以不处理。这样，将序号（B－A）/2，就可以得到该行黑点的中心位置坐标 C，（B－A）即为该行的黑点个数，记为 D。以上就是第一行的数据，为了提高系统处理的速度，降低系统资源的消耗，以后各行数据的处理方式与第一行有所

区别。在处理后面的行时，可以根据上一行得出的黑点中心坐标 C 决定比较的顺序。例如，如果上一行的黑点中心坐标 C 大于行中心坐标 16，那么就从右向左开始比较。再者，通过实践测试观察，摄像头采集的数据中，近处的数据一直是比较准确的，因此，处理的顺序是从近向远。通过实践测试，近 7 行（对应原始图像的前 70 行）作为近处数据，其它的数据作为远处数据。对处理过程再作进一步的优化方法为，远处数据的处理范围为，第 8 行数据参照第 7 行数据得出的黑点中心坐标 C 的位置，将其左右各 5 个点作为第 8 行的待处理数据，然后在遵照上面的方法进行比较。第 9 行再参照第 8 行得出的黑点中心坐标……这样，就可以大大减少待处理的数据，从而降低了数据处理的时间，加快了系统处理数据的速度。

经过上述图像处理后，可以获得两个主要的参数：每行黑点的中心位置坐标和每行的黑点总数。依据这两个参数，可以判断当前的路径信息。

经过条件判断，确定出可靠的最远处的一行数据作为路径判断的依据，即该行的黑点中心位置坐标。

11、经过上面的图像处理方法，得到了路径识别的结果，记为可靠的最远处的一行数据黑点中心位置坐标。根据此坐标，来控制舵机的转向。因为这个数据是远处的，所以有一些前瞻性，因此，可以提前做出转向的动作，这样，在急转弯的时候将会有很大的优势。

为了更好的解决转向的问题，本队采用查表的方法进行控制。设置一个一维数组，其中有33个元素，对应每行的33行坐标位置，不同的坐标对应不同的转角。而这33个转角的大小都是通过实际测量实验得到的。