速响应起跑线信号。

另外，干簧管是一种有触点的无源元件，比使用霍尔元件能省去一个电源为其供电，对于使用充电电池的竞赛车模节省了能源，也是保证车模电力系统稳定的一个方面。根据起跑线永磁铁安放的位置，对于干簧管必须进行单独放置，不和磁感应线圈一起放置。所有的干簧管采用并联的形式，占用单片机系统资源少，当任何一个干簧管检测磁场信号时便会给单片机 I/O 口一个检测到起跑线的信号，用来控制停车。另外，可以在系统刚开始的一段时间不检测 I/O 口，过一段时间后再检测，这样更加简单、精确。干簧管电路图如图 3.6.3 所示。

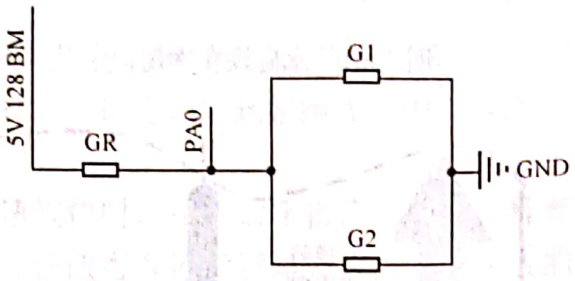

图 3.6.3　干簧管电路图

3.6.3　光电组

参照摄像头组。

3.7　PID 控制算法和应用

3.7.1　PID 控制算法

1. PID 控制简介

在工程实际中应用最为广泛的调节器控制规律为比例、积分、微分控制，简称 PID 控制，又称 PID 调节。PID 控制器问世至今已有近 70 年的历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握或得不到精确的数学模型时，控制理论的其他技术难以采用，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便，即当不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用 PID 控制技术。PID 控制在实际应用中也有 PI 和 PD 控制。如图 3.7.1 所示为 PID 控制系统原理图。

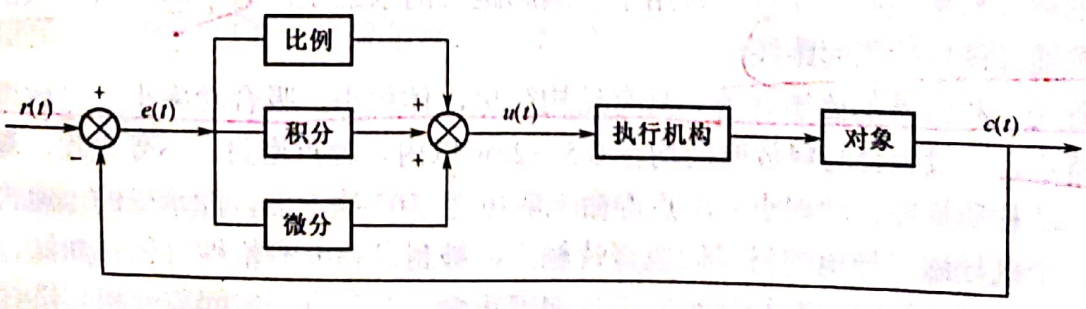

图 3.7.1　PID 控制系统原理图

（1）比例调节作用是按比例反映系统的偏差，系统一旦出现偏差，比例调节作用用来减小偏差。比例作用大，可以加快调节，减小误差，但是过大的比例使系统的稳定性下降，甚至造成系统的不稳定。

（2）积分调节作用是使系统消除稳态误差，提高误差度。当有误差时，积分调节就进行，直至无误差时积分调节才停止，积分调节输出常值。积分作用的强弱取决于积分时间 T_1，T_1 越小，积分作用就越强。反之 T_1 大则积分作用弱，加入积分调节可使系统稳定性下降，动态响应变慢。

（3）微分调节作用反映系统偏差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。在微分时间选择合适的情况下，可以减少超调，减少调节时间。微分作用对噪声干扰有放大作用，因此过强地加微分调节，对系统抗干扰不利。

智能汽车的结构和参数受各个方面的影响，所以系统得不到精确的数学模型，控制理论的其他技术也难以采用，系统控制器的结构和参数必须依靠经验和现场调试来确定，因此 PID 控制技术最为方便。

数字 PID 有多种改进形式，如位置式 PID、增量式 PID、PI 控制及 PD 控制，应视系统的需求进行选择。

2. PID 算法分类

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。增量型算法与位置型算法相比，具有以下优点：增量型算法不需要作累加，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。

1）位置式 PID

$$u(k)=K_{\mathrm{p}}\left\{e(k)+\frac{T}{T_{\mathrm{i}}}\sum_{j=0}^{k}e(j)+\frac{T_{\mathrm{d}}}{T}[e(k)-e(k-1)]\right\}$$

模拟调节器的调节动作是连续的，任何瞬间的输出控制量 u 都对应于执行机构（如调节阀）的位置。由上式可知，数字控制器的输出控制量也和阀门位置相对应，故称为位置型算式（简称位置式）。相应的算法流程图如图 3.7.2 所示。

由流程图可以看出，因为积分的作用是对一段时间内偏差信号的累加，因此利用计算机实现位置型算法不是很方便，不仅需要占用较多的存储单元，而且编程也不方便，下面介绍其改进式——增量型算法。

2）增量式 PID

在实际电动机编程控制时，采用了基于 PWM 脉宽调制的 PID 闭环控制，主要实现对智能汽车模直流电动机速度的闭环控制。

控制编程依据：

$$u(n)=u(n-1)+K_{\mathrm{p}}[e(n)-e(n-1)]+K_{\mathrm{i}}e(n)+K_{\mathrm{d}}[e(n-2)-2e(n-1)+e(n)]$$

式中，$u(n)$ 为第 n 次输出控制量；$u(n-1)$ 为第 $n-1$ 次输出控制量；$e(n)$ 为第 n 次偏差；$e(n-1)$

为第 $n-1$ 次偏差；$e(n-2)$ 为第 $n-2$ 次偏差；K_p 为比例增益系数；K_i 为积分增益系数；K_d 为微分增益系数。

电动机 PID 控制子程序流程如图 3.7.3 所示。

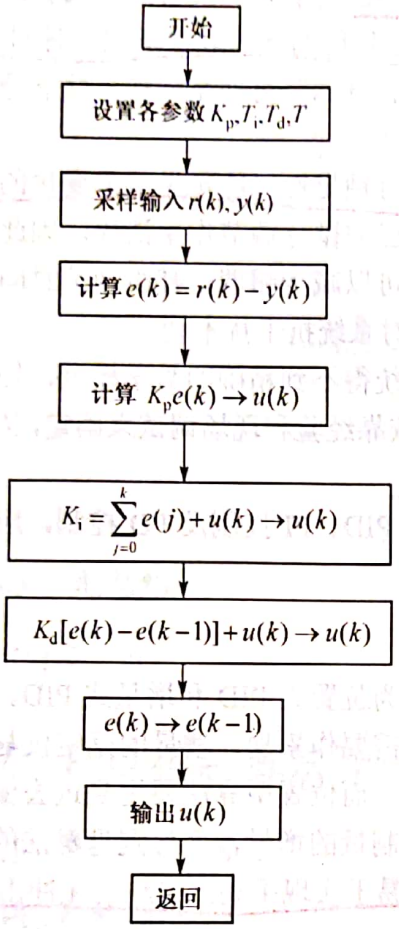

图 3.7.2　位置式 PID 算法流程

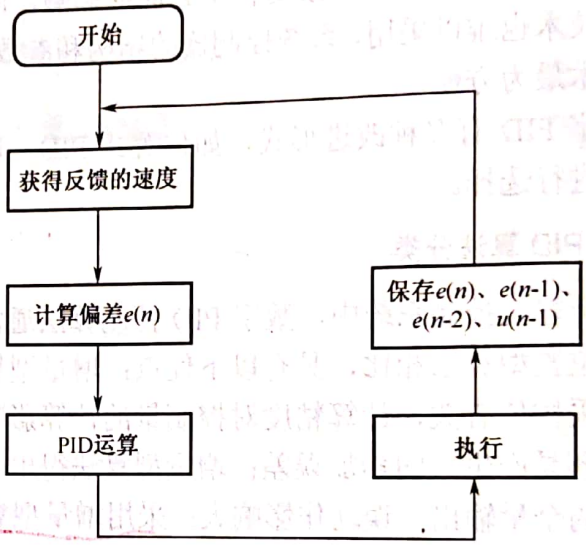

图 3.7.3　电动机 PID 控制子程序流程

3）不完全微分 PID

将微分环节引入智能汽车的方向和速度控制明显地改善了系统的动态性能，但对于误差干扰突变也特别敏感，对系统的稳定性有一定的不良影响。为了克服上述缺点，本节在 PID 算法中加入了一阶惯性环节，不完全微分 PID 算法结构如图 3.7.4 所示。

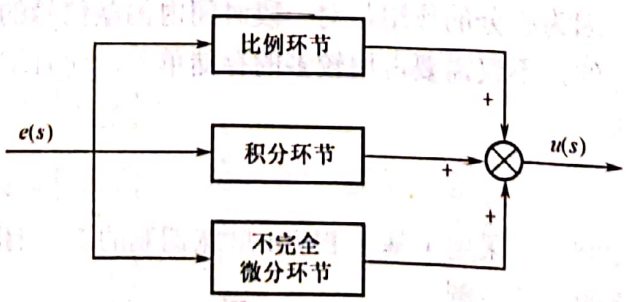

图 3.7.4　不完全微分 PID 算法结构图

将一阶惯性环节直接加到微分环节上，可以得到系统的传递函数为：

$$U(s)=\left(K_p+\frac{K_i}{s}+\frac{K_d\cdot s}{1+T\cdot s}\right)=U_p(s)+U_i(s)+U_d(s) \tag{3.2}$$

将式（3.2）的微分项推导并整理，得到方程如下：

$$u_d(k)=K_d(1-\alpha)[e(k)-e(k-1)]+\alpha\cdot u_d(k-1) \tag{3.3}$$

式中，α 是由系统的时间常数和一阶惯性环节时间常数决定的一个常数。

为了编程方便，可以将式（3.3）写成如下形式：

$$\begin{aligned}u_d(k)&=K_d(1-\alpha)e(k)+e\alpha\cdot u_d(k-1)-K_d(1-\alpha)e(k-1)\\&=K_d(1-\alpha)e(k)+H(k-1)\end{aligned} \tag{3.4}$$

式中，$H(k-1)=\alpha u(k-1)-K(1-\alpha)e(k-1)$ 。

分析式（3.4）可知，引入不完全微分以后，微分输出在第一个采样周期内被减小了，此后又按照一定比例衰减[3][4]。实验表明，不完全微分有效克服了智能汽车的偏差干扰给速度控制带来的不良影响，具有较好的控制效果。图 3.7.5 为不完全微分 PID 算法的程序流程图。

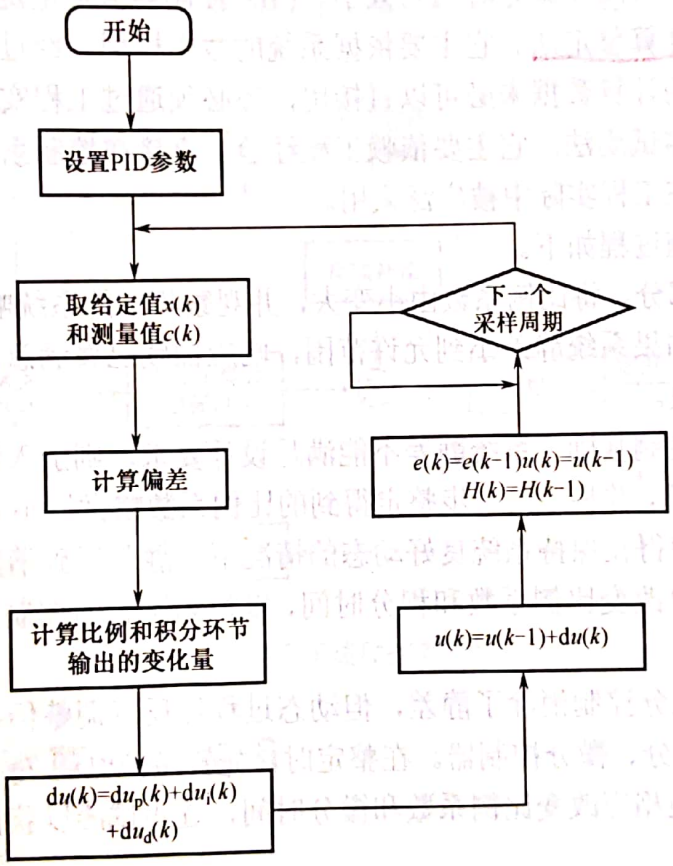

图 3.7.5　不完全微分 PID 算法的程序流程图

4）微分先行 PID

由于智能汽车在跑道上行驶时经常会遇到转弯的情况，所以智能汽车的速度设定值和方向设定值都会发生频繁的变化，从而造成系统振荡。为了解决设定值的频繁变化给系统带来的不良影响，本节在智能汽车的速度和方向控制上引入微分先行 PID 算法，其特点是只对输出量进行微分，即只对速度测量值和舵机偏转量进行微分，而不对速度和方向的设定值进行微分。这样在设定值发生变化时，输出量并不会改变，而被控量的变化相对来说是比较缓和的，这就很好地避免了设定值的频繁变化给系统造成的振荡，明显改善了系统的动态性能。

图 3.7.6 是微分先行 PID 控制的结构图，微分先行的增量控制算式如下。

$$\Delta u(k) = K_p[e(k) - e(k-1)] + Ke(k) - K[c(k) - 2c(k-1) + c(k-2)] - K_d[c(k) - c(k-1)] \tag{3.5}$$

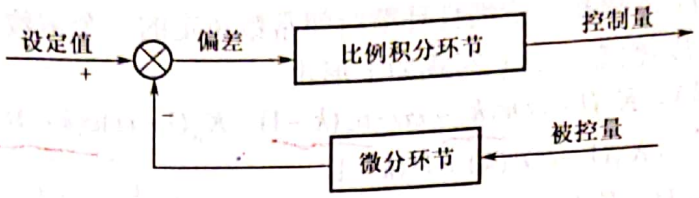

图 3.7.6 微分先行 PID 控制结构图

3. PID 参数的整定

PID 控制器的参数整定是控制系统设计的核心内容。它根据被控过程的特性确定 PID 控制器的比例系数、积分时间和微分时间的大小。PID 控制器参数整定的方法很多，概括起来有两大类：一是理论计算整定法。它主要依据系统的数学模型，经过理论计算确定控制器参数。这种方法所得到的计算数据未必可以直接用，还必须通过工程实际进行调整和修改。二是工程整定方法，俗称试凑法。它主要依赖工程经验，直接在控制系统的试验中进行，且方法简单、易于掌握，在工程实际中被广泛采用。

试凑法的具体实施过程如下。

（1）在整定比例部分，将比例系数由小变大，并观察相应的系统响应，直至得到反应快、超调小的响应曲线。如果系统静差小到允许范围，响应曲线已属满意，那么只需比例控制即可，由此确定比例系数。

（2）如果在比例控制基础上系统静差不能满足设计要求，则加入积分环节，整定时先置积分时间 T_I 为很大的值，并将经第一步整定得到的比例系数略微缩小（如缩小为原值的 0.8），然后减少积分时间，使得在保持系统良好动态的情况下，静差得到消除，在此过程中，可根据响应曲线的好坏反复改变比例系数和积分时间，以得到满意的控制过程，从而得到整定参数。

（3）若使用比例积分控制消除了静差，但动态过程经反复调整仍不能满意，则可加入微分环节，构成比例、积分、微分控制器。在整定时，先置微分时间 T_d 为零，在第二步整定的基础上增大 T_d，同样地相应改变比例系数和微分时间，逐步试凑以获得满意的调节效果和控制参数。

3.7.2 PID 控制在智能汽车上的实现

1. 舵机 PD 控制

舵机的 PD 控制也是历届参赛队伍中使用比较多的方法。

表达式为：

$$\text{output} = K_p e + K_i \sum e + K_d \Delta e$$

式中，e 为车身与跑道的偏移量；K_p 为比例系数；K_i 为积分系数；K_d 为微分系数；output 为舵机 PWM 输出量；Δe 为上一次偏移量与当前偏移量之差。

其中起最主要作用的是比例项；积分项会导致舵机转向延时，故常常将积分项系数 K_i 置零；适当调节微分系数 K_d 能使车模优化路径，达到更好的转向效果。

2. 电动机 PID 控制

为了使智能汽车在直道上以较快速度运行，在转弯时防止智能汽车冲出跑道，则必须将智能汽车的速度降低，这就要求智能汽车的速度控制系统具有很好的加/减速性能。当智能汽车经过连续转弯的跑道时，路线偏差的频繁变化会造成速度设定的频繁变化，这会引起速度控制系统的振荡，并且微分环节对误差突变干扰很敏感，容易造成系统的不稳定。为了解决上述存在的问题，本节对数字 PID 算法进行了改进，将不完全微分和微分先行引入到 PID 算法中，大大改善了速度控制系统的动态性能。

图 3.7.7 是智能汽车速度控制系统结构图。由于赛道路况和智能汽车的姿态会经常变化，所以速度控制系统的模型也是不定的，为了提高系统的适应性，本节速度控制系统中采用了模糊 PID 算法。将速度设定和实际速度进行模糊分挡[5][6]，通过调试得到不同情况下相对最优的 PID 参数，保证了速度控制系统在不同情况下都有较好的控制效果。

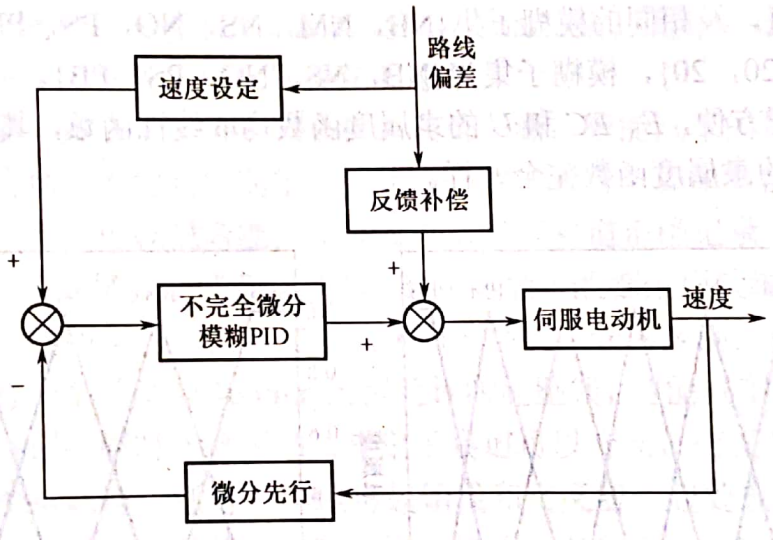

图 3.7.7　智能汽车速度控制系统结构图

3.8　其他控制算法和应用

3.8.1　模糊控制

在转向控制算法中，如果采用 PID 控制算法，为了使车模在整个过程中跑得比较顺畅，要求在小偏差时缓慢调节，甚至不调节；在遇到弯道产生大偏差时，舵机能够迅速动作。 这就要求偏差量比例系数 P 是一个变量，即一个非线性的过程。但由于 P 是一个离散的数值，势必造成转向调节的阶跃式变化，对路径变化反应不灵敏，同时容易产生超调及振荡现象。而对于追求高速性能的高车速、短决策周期控制策略来说，很可能因为舵机响应不及时或超调过大而造成控制失效。 因而该系统的设计采用模糊控制算法，以增强系统的控制响应速度及对不确定性因素的适应性，其原理如图 3.8.1 所示。

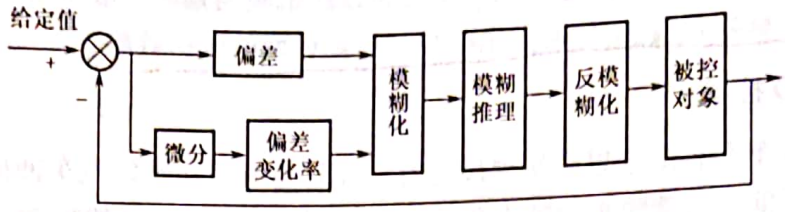

图 3.8.1 模糊控制原理

模糊控制采用二维模糊控制器，输入变量是摄像头判断出的黑线相对于中心线的横向偏差 E 和偏差的变化率 EC，其输出量是智能汽车舵机的转向角 U。同时由于系统所采用的单片机具有模糊控制指令，算法的实现较容易。

1．模糊子集的选取

模糊化即把精确的输入量按照输入隶属度转换为一个模糊矢量并作为模糊控制器的输入变量。在该系统中，输入的模糊变量为偏差 E 和偏差变化率 EC，输出的模糊变量是舵机转向角 U，其中偏差 E 和舵机转向角 U 具有相同的论域：

$E=U=\{-40，40\}$，及相同的模糊子集{NB，NM，NS，NO，PS，PM，PB}，偏差变化率 EC 的论域为 $EC=\{-20，20\}$，模糊子集为{NB，NS，NO，PS，PB}。

为了实现和处理方便，E、EC 和 U 的隶属度函数均取线性函数，其隶属度函数如图 3.8.2 所示，其中 U 和 E 的隶属度函数完全一样。

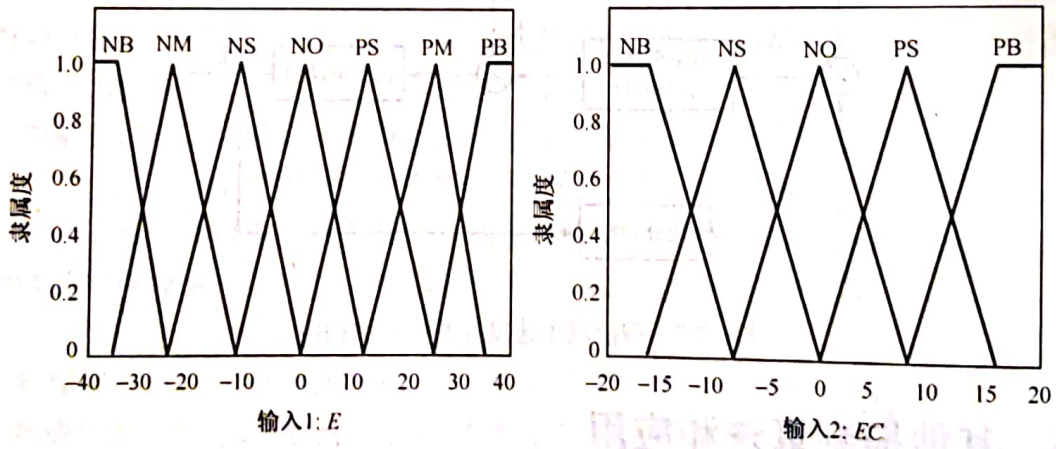

图 3.8.2 模糊子集

2．模糊规则库的设计

模糊规则表是模糊推理的依据，也是模糊控制器设计的核心工作，根据智能汽车的运行特性，并通过实际调试效果不断调整，最终设计出模糊控制器模糊规则表。

3．模糊推理及反模糊化

该系统中模糊推理采用 MAX-MIN 法：当相同后件的规则强度不同时，模糊输出取其大者。设输入变量 E、EC、U 的隶属度分别为 μE、μEC、μU，则有：

$$\mu_{U_k}(E,EC)=\min\{\mu_{E_i},\mu_{EC_j}\}$$

$$\mu_U(E,EC)=\max\{\mu_{U_k}(E,EC)\}$$

式中，i,j= NB, NM, NS, NO, PS, PM, PB 。反模糊化即为将模糊推理后得到的模糊集转化为用作控制的数字值的过程。为了能够得到更加准确的控制量，设计的反模糊化算法采用重心法（COG 法），能够较好地表达输出隶属度函数的计算结果，即采用下式确定控制输出 v_0：

$$v_0 = \frac{\sum_{i=1}^{m} v_i k_i}{\sum_{i=1}^{m} k_i}$$

式中，m 为输出模糊子集数；v_i 为各输出模糊子集对应模糊单点集的值；k_i 为输出对该子集的隶属度；v_0 为控制器的精确输出控制量。

3.8.2　赛道记忆算法

全国大学生智能汽车竞赛的规则为记忆算法在智能汽车控制系统中的应用提供了条件。比赛中智能汽车要在赛道上连续跑两圈，并以其中最好圈的成绩作为比赛成绩。赛道记忆算法的基本方法是：在第一圈以最安全的速度慢速驶过，将赛道的信息保存下来，第二圈根据保存下来的信息进行车速和转向控制策略的最优化，从而在第二圈取得好成绩。

要想成功实现赛道记忆算法，必须具备以下几点。

（1）智能汽车在第一圈必须安全走完全程。在第一圈，智能汽车的最主要目的并不是跑得快，而是采集赛道信息。因此它的期望运行轨迹与赛道重合，如果智能汽车在第一圈出现跑出场外等特殊情况，那么可以说智能汽车并没有真正将赛道正确记录下来，此时赛道记忆已经失败。因此使用赛道记忆算法的智能汽车，在第一圈一般都采用较缓慢的匀速走完全程。

（2）智能汽车必须能明确分辨出赛道起始线和十字交叉线的区别。根据比赛规则，在赛道的计时起始点两边有一个长度为 100mm 的黑色计时起跑线，智能汽车前端通过起始线作为比赛开始或者结束的时刻；同时比赛规则还规定了赛道可以交叉，交叉角为 90° 。不采用赛道记忆算法的智能汽车有时可以通过简单地将起始线和交叉线一起处理来回避这个问题，而采用赛道记忆算法的智能汽车则必须要判断出第一圈的结束和赛道十字交叉线的区别。

（3）智能汽车必须有足够记录一圈赛道数据的内存空间。由于单片机芯片的 RAM 容量有限，因此这个问题也不能忽视。算法设计者需要对算法进行优化处理，改变保存的格式，从而节省内存空间。

以上是成功实现记忆算法的条件，但实现记忆的难度远不止这些，主要体现在对第一圈记忆的赛道信息的分析和处理，这往往是记忆算法成功与否的关键。记忆算法的难度虽然大，但若成功，它取得的效果是不言而喻的，而且它有很大的发展空间。

3.9　计算机辅助调试

3.9.1　开发软件介绍

本实例采用 Kinetis K60 芯片，开发环境为 CodeWarrrior10.1，CodeWarrrior10.1 是 CodeWarrrior 公司为 Kinetis 系列微处理器开发的一个集成开发环境。