无人系统设计说明文档

本组进行无人系统设计的核心思想是本机速度不变，寻求并保持和敌机角度一致，从而击落敌机。但是，如何做到从与敌机角度完全不同到变为角度一致呢？

在大多数场合中，用“开关量”来控制一个物理量，就显得比较简单粗暴。有时候，是无法保持稳定的。因为单片机、传感器不是无限快的，采集、控制需要时间，而且，控制对象具有惯性。比如你将一个加热器拔掉，它的“余热”（即热惯性）可能还会使水温继续升高一小会。这时，就需要一种算法：它可以将需要控制的物理量带到目标附近，可以“预见”这个量的变化趋势，也可以消除因为散热、阻力等因素造成的静态误差。

PID 算法，是一种常见的“保持稳定”控制算法，就是比例（proportional）、积分（integral）和微分（derivative），。PID 控制算法是结合比例、积分和微分三种环节于一体的控制算法，根据输入的偏差值，按照比例、积分、微分的函数关系进行运算，运算结果用以控制输出，如图 1 所示。

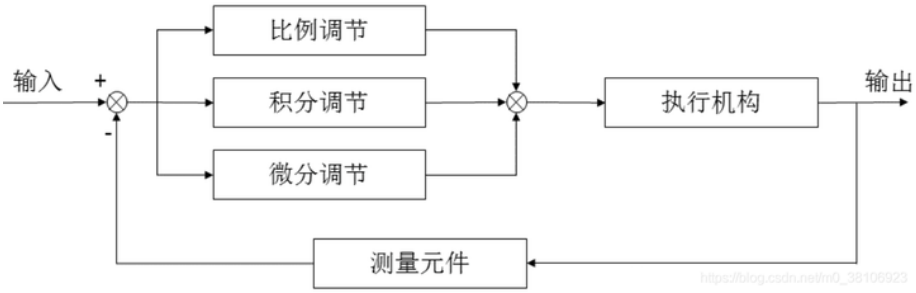

图 1

我们的设计是，先根据敌机和本机的 x、y、z 坐标计算出 dx、dy、dz 距离，再根据距离计算出 u、v、w 三个角度，最后计算出 du、dv、dw 三个角度差，如图 2 所示。

随后，将角度差（向量 y0）传入 PID 模块中，计算出三个角速度，从而控制飞机的俯仰、翻滚和偏航，如图 3 所示。

我们最终设定 p、i、d 的值为 3000、0 和 100，在调参的过程中，我们发现，由于 p 相当于改变的速度，i 相当于预判，d 相当于阻碍，当 p 过小的时候有可能根本追不上敌机的角度，所以 p 应该略大；i 过大时会有卡顿的现象，经过试验设为 0 也可以很好的打中，因此设为 0

虽然飞机的速度是一定的（400），但是三个方向的分速度 dx、dy、dz 与角速度有关，因此调整角速度实际上也就是调整三个方向的分速度，因此我们

的策略即为调整至与敌机角度一致之后保持方向不变直行直至追上敌机

```
/* Start_END */
}

void angle_extraction_Outputs_wrapper(const int32_T *u0,
                                      real_T *y0)
{
/* Output_BEGIN */
const int32_T *u1 = u0 + 12;
    if (u0[2] == 2) {
        const int32_T *tmp = u0;
        u0 = u1;
        u1 = tmp;
    }
    double dx = u1[3] - u0[3], dy = u1[4] - u0[4], dz = u1[5] - u0[5];
    double r = sqrt(dx * dx + dy * dy + dz * dz);
    double u = asin(dz / r);
    double v = atan2(dy, dx);
    double w = 0;
    double cu = u0[6] / 1000.0, cv = u0[8] / 1000.0, cw = u0[7] / 1000.0; // current
    if (cu > PI)
        cu -= PI2;
    double du = u - cu, dv = v - cv, dw = w - cw;
    trap(&dv);
    trap(&dw);

    y0[0] = du;
    y0[1] = dv;
    y0[2] = dw;
/* Output_END */
}

void angle_extraction_Terminate_wrapper(void)
{
/* Terminate_BEGIN */
/*
```

图 2

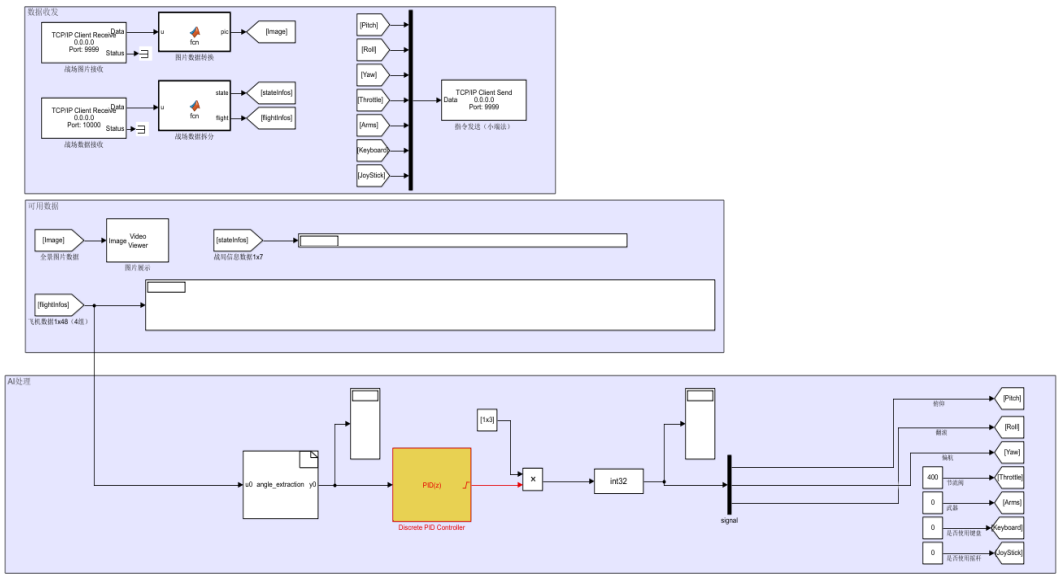

图 3