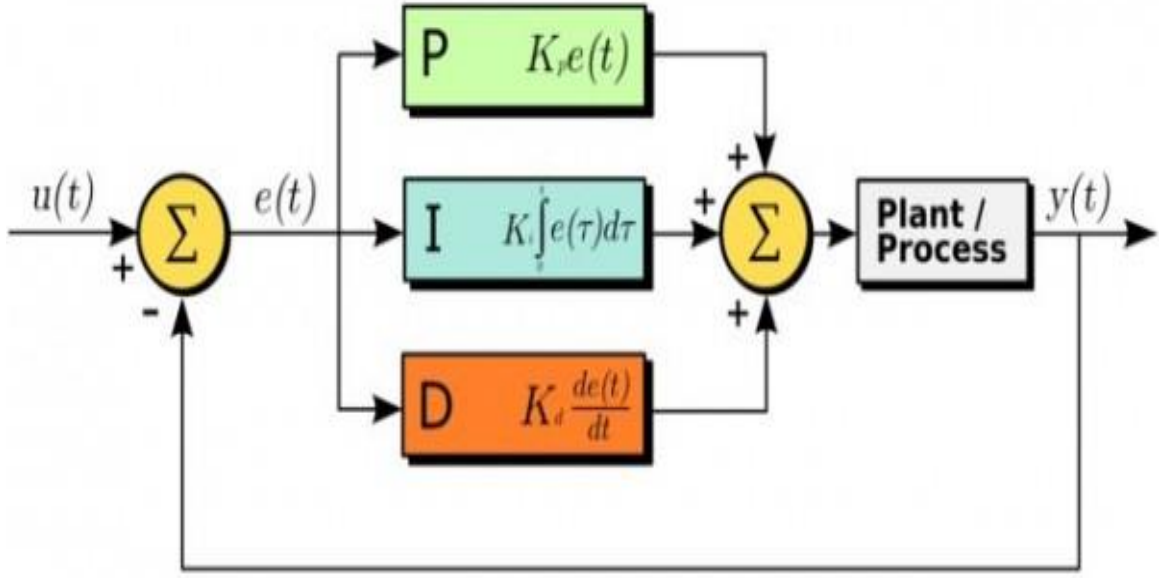
PID Control

Theory

$$\mathrm{u} = \underbrace{K_p e}_{\text{Proportional Term}} + \underbrace{K_i \int_0^t e\,dt}_{\text{Integral Term}} + \underbrace{K_d \frac{d}{dt} e}_{\text{Differential Term}}$$



目標位置

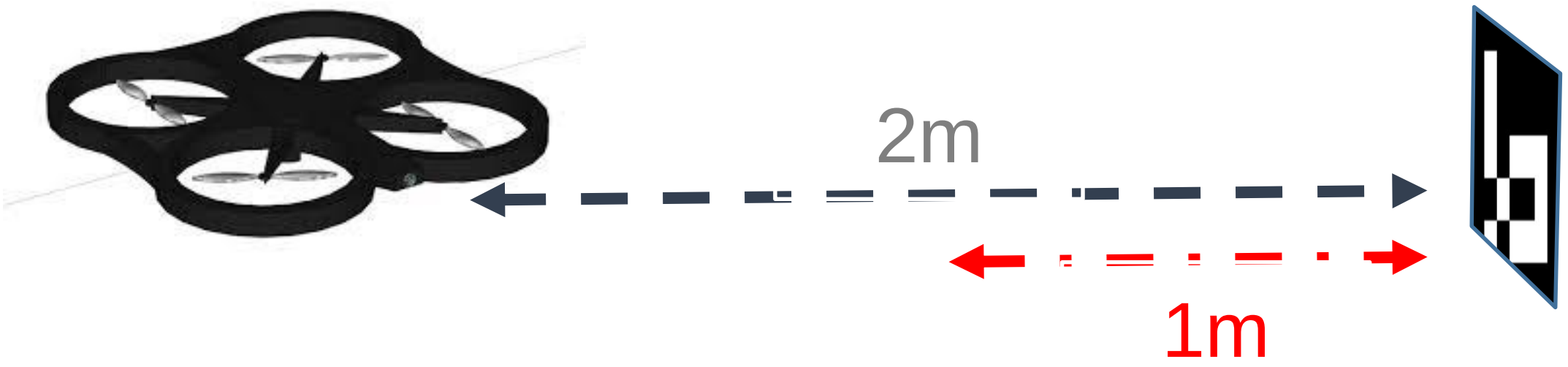
離目標還很遠時

速度大

往前飛，越來越接近目標後，
誤差變小，速度也變小

速度小

Step1:



欲修正的誤差(error): 2m – 1m = 1m

Step2:

利用PID去smooth原本的誤差

1m ⟶ 0.4m

將誤差轉換成速度給無人機

Step3:

根據無人機飛行的狀況調整PID

1. 先把I, D設為0, 先調整P
2. 再調整I, D

```cpp
void PIDManager::getCommand(Mat& _error, Mat& _output) {
    /*
     *  Input format :  Mat(4, 1, CV_64F)
     *      which stands for _error(x_error, y_error, z_error, r_error)
     *  Output format : Mat(4, 1, CV_64F)
     *      which stands for _output(x_out, y_out, z_out, r_out)
     */
    double dt = (getCurrentTime() - previous_time) / 1000.; // in "sec" unit
    Mat de = Mat::zeros(4, 1, CV_64F);
    Mat output = Mat::zeros(4, 1, CV_64F);
    if(mInit) {
        for(int i = 0; i < 4; i++) {
            // de
            de.at<double>(i, 0) = (_error.at<double>(i, 0) - previous_error.at<double>(i, 0)) / dt;
            error_integral.at<double>(i, 0) += de.at<double>(i, 0) * dt;
            // output
            Mat coeffs;
            if(i == 0)  coeffs = mX;
            else if(i == 1) coeffs = mY;
            else if(i == 2) coeffs = mZ;
            else        coeffs = mR;
            //cout << endl << i << endl;
            //cout << coeffs << endl;
            output.at<double>(i, 0) = _error.at<double>(i, 0) * coeffs.at<double>(0, 0)     // error * kp
                                    + error_integral.at<double>(i, 0) * coeffs.at<double>(1, 0)     // Sum(errors) * ki
                                    + de.at<double>(i, 0) * coeffs.at<double>(2, 0);        // de * kd
        }
    } else {
        mInit = true;
```