无人驾驶汽车系统入门（四）——反馈控制入门，PID控制

无人驾驶汽车系统入门（四）——反馈控制入门，PID控制

说明：

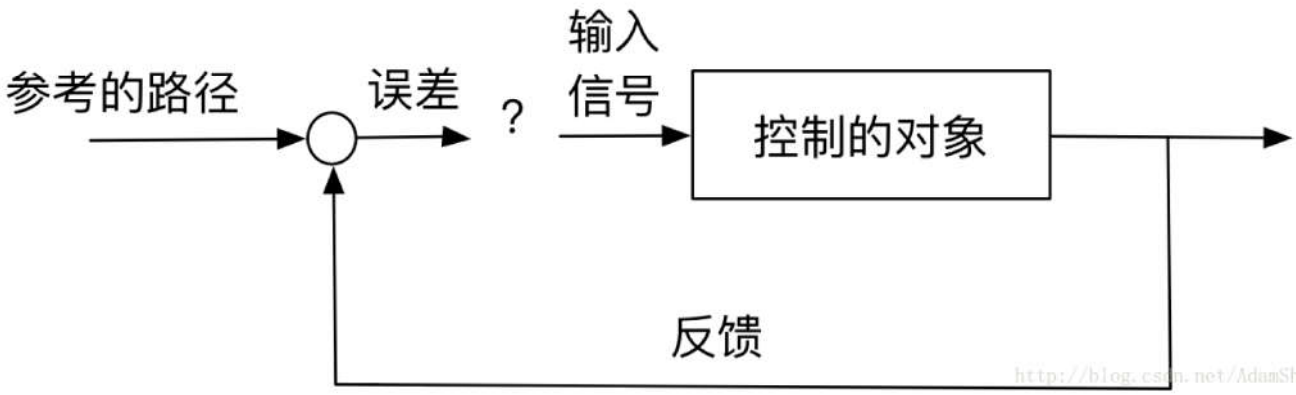
> 前面几篇博客介绍了卡尔曼滤波的一些基本算法，其实目标追踪，定位，传感器融合还有很多问题要处理，这些我们在以后的系列博客中在进一步细讲，现在我想给大家介绍一下无人驾驶汽车系统开发中需要的控制相关的理论和技术，还是和第一篇说的那样，我想到哪就写到哪，追踪和定位等更高级的算法我在后面会继续写。所以感兴趣的同学可以关注我的博客，无人驾驶汽车系统入门系列博客会一直更新下去。

为什么需要控制理论

试想有如下场景，当你驾驶一辆汽车通过这个弯道的时候，假设你已经知道你要开的路线，那么你会怎么去操作控制你的车呢？

显然，如果你不是专业的选手的话，你无法做到一步到位的控制，你需要一边观察车辆相对于你想要开的路线的相对偏差，一边调整你的方向盘的角度和油门踏板的力度，这种基于环境反馈的控制我们称为 反馈控制 。

反馈控制是现代控制理论的基础，这是反馈控制的一般思路：



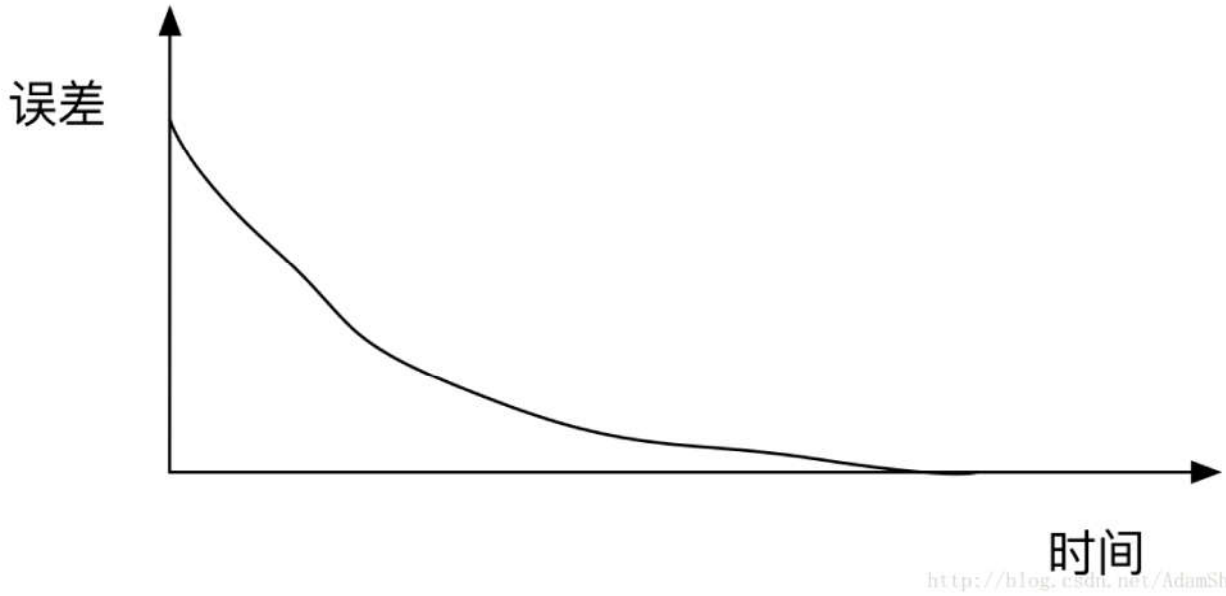
我们希望我们控制的对象（无人车）能够按照我们希望（规划好）的路径行驶，

我们会将环境当前给我们的反馈（我们当前的位置）和参考线进行比较，得到我们当前偏离参考线的距离（误差），

基于这个误差，我们设计一定的算法来产生输出信号，使得这个误差不断的变小，这样的过程就是反馈控制的一般过程。

那么我们如何基于这个误差来产生控制指令呢?

我们最直观的感觉就是要让误差在我们的控制下逐渐变小直到为0：



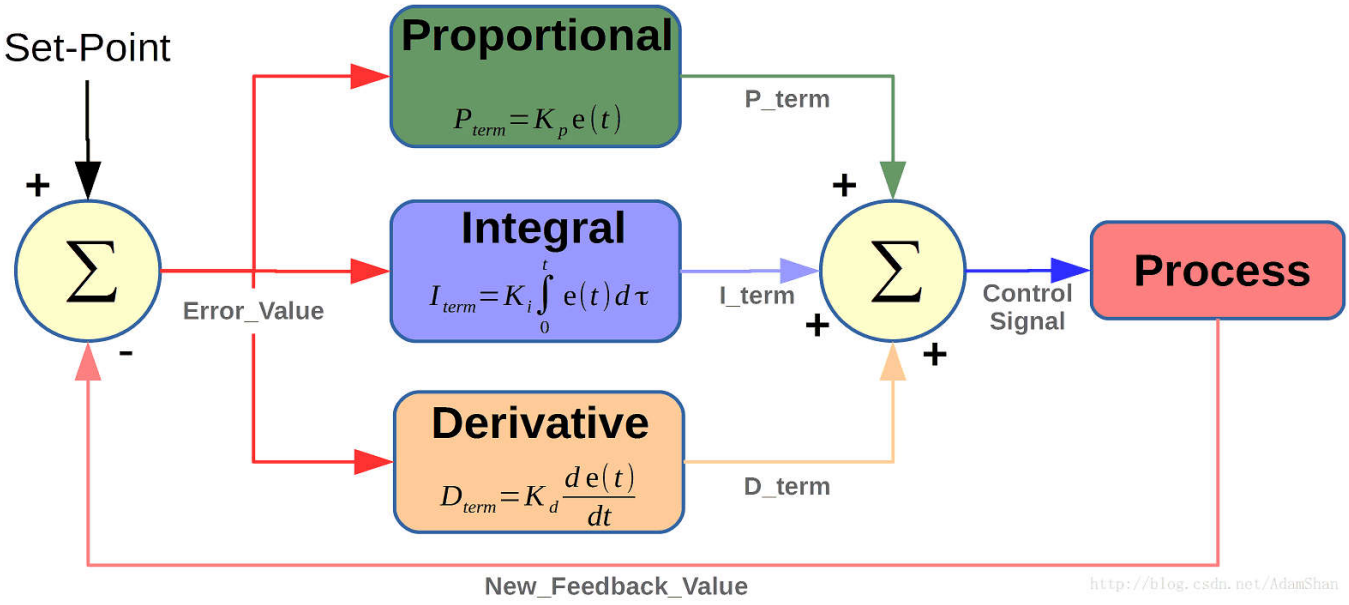
0误差就意味着车一直在你想让它开的路径上开。如何减少误差就是我们这几篇博客要向大家介绍的内容。

为了了解反馈控制，我先向大家介绍 PID控制，PID控制是目前利用最为广泛的控制理论，我们以它为出发点讨论控制理论。

比例，积分和导数

PID就是指 比例（proportion）、积分（integral）、导数（derivative），

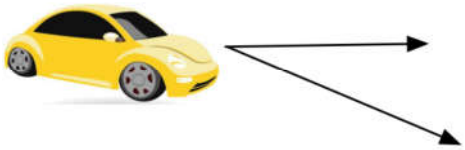
这三项表示我们如何使用我们的误差来产生控制指令，整个流程如下：



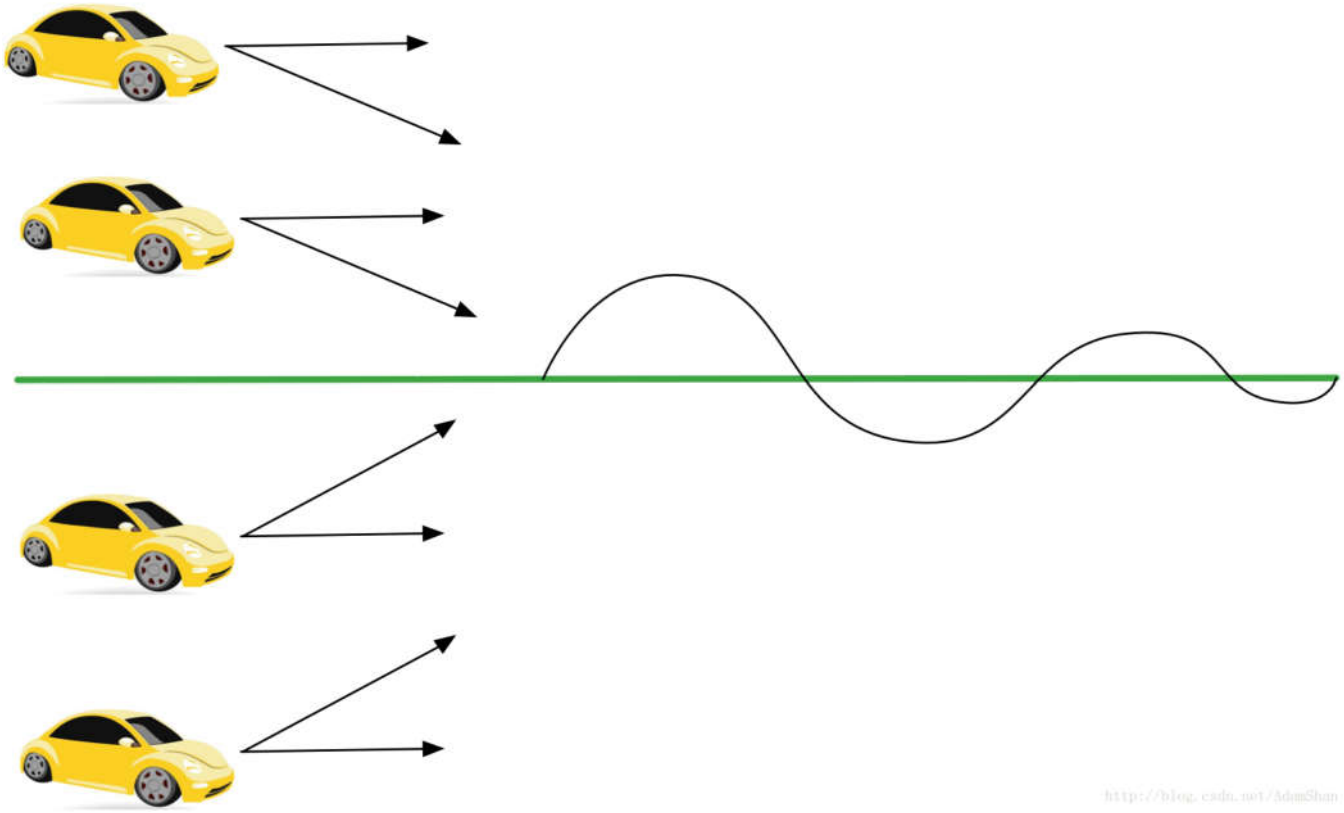
首先是根据反馈和参考值求出误差，这里的误差根据具体的情况可以是各种度量，比如说控制车辆按照指定的路径形式，那么就是车辆当前位置和参考线的距离，控制车辆的速度在设定的值，那么就是当前速度和设定速度的差值，求出误差以后，再根据误差求比例，积分和微分三项，其中 K_p，K_i，和 K_d 是三项的系数，它们决定着这三项对最后输出的影响的比重。将 P, I, D 三项求和作为最后的输出信号。我们分别讨论这三项的意义。

1.P控制

考虑一个简单的情况，假设我们希望无人车按照图中绿线行驶，但是我们的车在如图所示的位置：

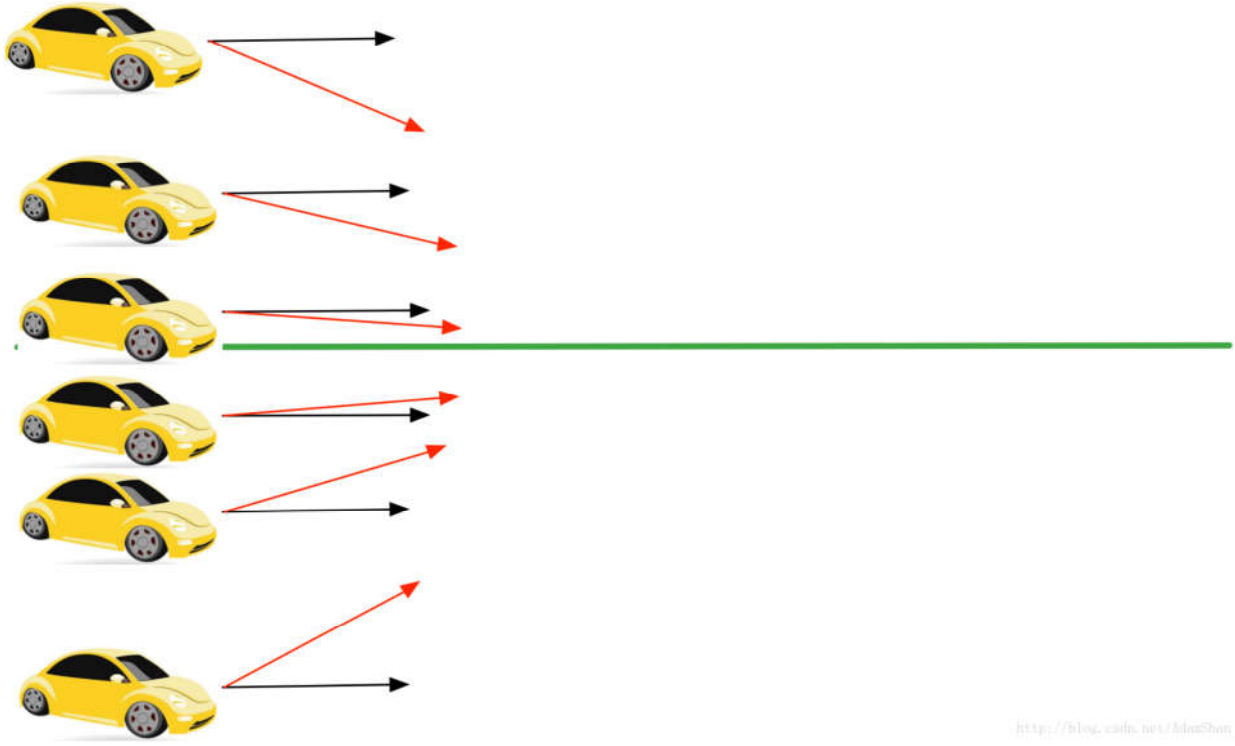


那么我们要转多少度角呢？如果都按照固定的角度转（如下图），那么车的轨迹将如图中所示：



那么显然坐这样的车是不舒服的。一个直观的解决方法就是使用比例控制。

如图所示，当偏差大的时候，我们偏转更多的角度，当偏差小的时候，则偏转小一点。

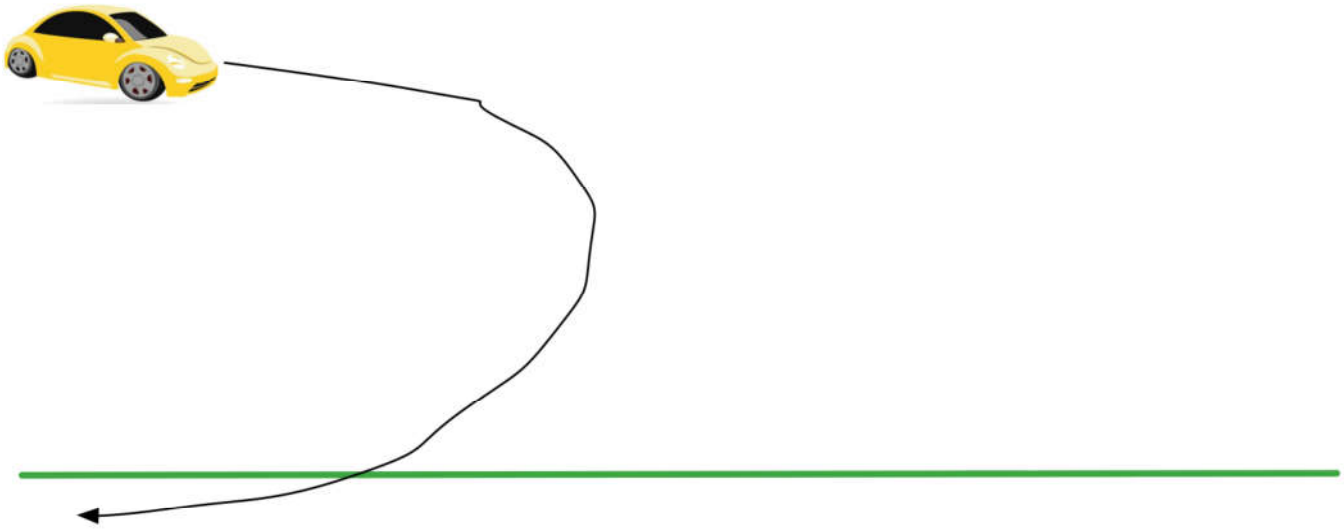


那么这就是P control（比例控制）这里我们使用 CTE（Cross Track Error） 作为偏差度量，

CTE就是我们到参考线的距离。那么这个时候转角就变成了：

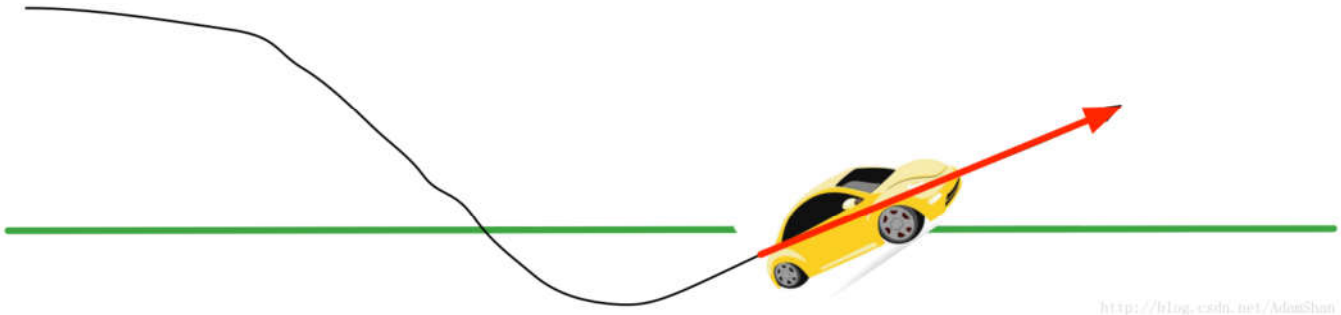
$$steering\ angle = K_p \cdot e(t)$$

其中的 $e(t)$ 就是在t时刻的CTE，在P控制中系数 K_p 会直接影响到实际的控制效果，在合理的数值范围内 K_p 越大控制的效果越好（越快速的回到参考线附近），但是，当本身位置和参考线相距很远且 K_p 系数较大的时候，就会出现车辆失去控制的情况：



失去控制

所以说，如果 K_p 参数设计合理的话，P控制要比固定控制要更好，但是还是不能控制的很好，因为P控制的车辆容易0值的影响，如图所示：



此时车辆虽然在参考线上，但是并不是我们希望的状态（它在下一刻就会偏离），但是对于P控制而言，这是理想状态，此时控制转角为0,因此，P控制会一次又一次的超过参考线（overshot），为了矫正这种overshot，我们需要考虑一个额外的误差项——CTE变化率。

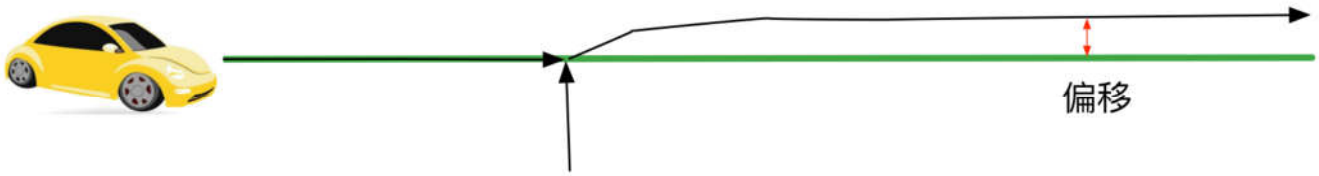
2.PD控制

CTE的变化率描述了我们的无人车向着参考线方向移动的有多快，如果我们的无人车一直都完美的在参考线上运动的话，那么我们的CTE变化率就为0。那么这一项（描述误差的变化率）就可以用导数来表示，那么，现在我们的控制输出就变成了比例项和导数项求和的形式：

$$steering\ angle = K_p \cdot e + K_d \frac{d\,e(t)}{dt}$$

其中的 K_d 就是导数项的系数，它的大小决定了CTE变化率对于反馈控制的影响。此时我们的控制叫做PD控制，在PD控制中，我们有两个系数需要调整，直观上来看，增大 P 系数将会增大无人车向着参考线方向运动的倾向；增大 D 系数将会增大无人车快速向参考线方向的运动的“抵抗力”从而使得向参考线方向的运动变得更加平滑。使用过大的 P 系数，过小的 D 系数的系统我们称之为 **欠阻尼的（underdamped）**，这种情况的无人车将沿着参考线震荡前进，反之，如果P系数过小，D系数过大，那么我们称之为 **过阻尼的（overdamped）**，这将使得无人车要较长的时间才能纠正其误差。合适地选择 P，D参数可以使无人车能快速回到参考线上的同时很好的维持在参考线上运动。

PD控制似乎已经能够胜任良好的反馈控制了，但其实还不够，PD控制器可以保证正常的控制的需求，但是当环境存在扰动的时候，比如说下面这种情况：



在这个位置受到环境
的干扰偏离了参考线

车在受力发生轻微偏移以后，由于PD控制中下 P 项倾向于向参考线方向运动，而 D 项则尝试抵消这种倾向，造成无人车始终都无法沿着参考线运动，这个问题叫做 steady state error 为了解决这个问题，我们再引入一项——积分项。

3.PID控制

我们将积分项也就如到我们的控制输出函数中，这个时候，无人车的转角就可以表示为：

$$steering\ angle = K_p \cdot e + K_d \frac{d\,e(t)}{dt} + K_i \int_0^t e(t)dt$$

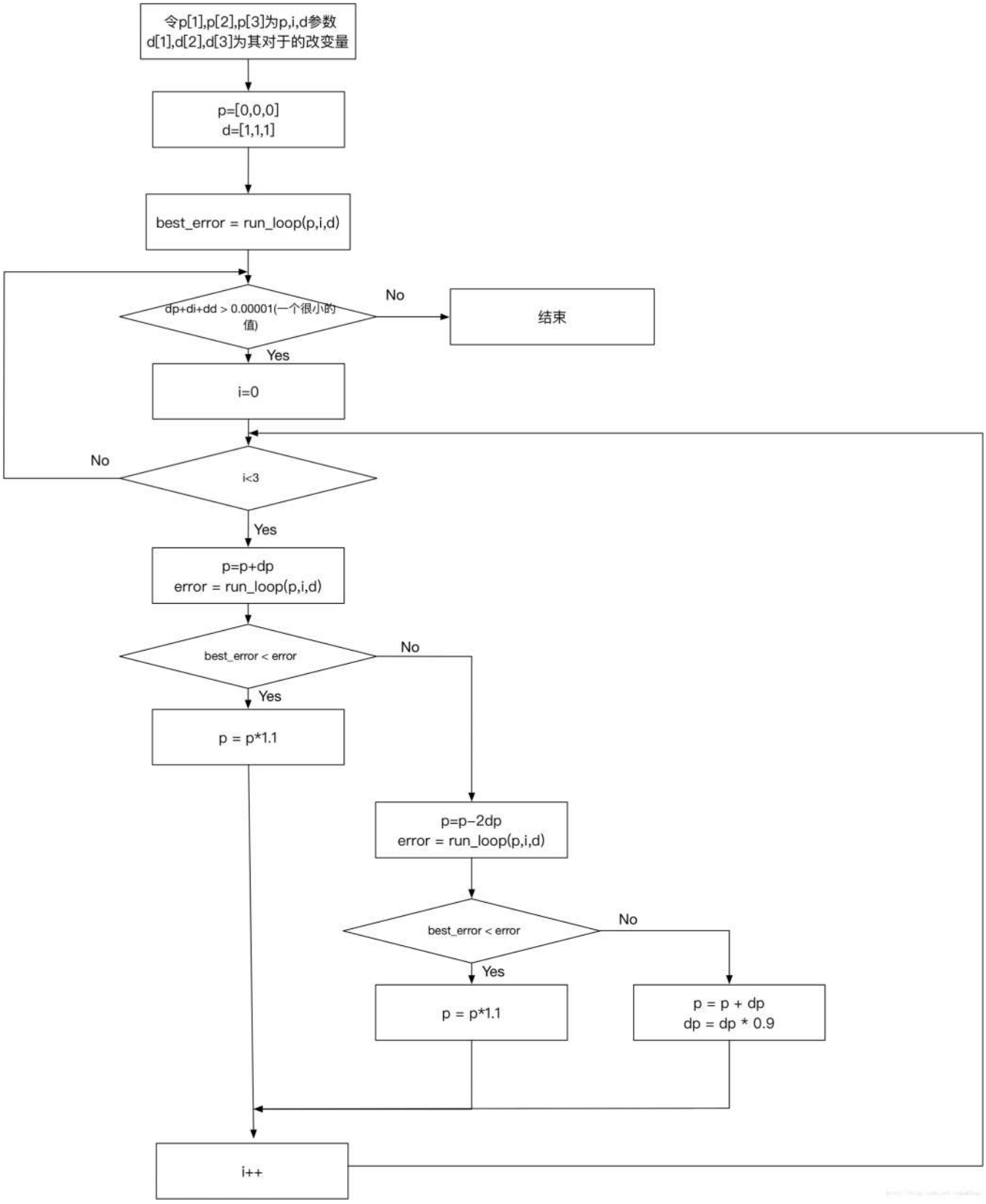
其中 K_i 就是积分项系数，积分项在我们这个例子中其实很好理解，本质就是车的实际路线到参考线的图形的面积，加入积分项以后，控制函数会尽可能使车辆路线的积分尽可能小（也就是使车辆路线和实际运动参考线之间形成的形状的面积尽可能小），那么也就避免了steady state这种情况了。

同样的，这里的积分项系数的大小也会影响我们整个控制系统的稳定性，过大的 K_i 会使控制系统“震荡”地运行，过小的 K_i 又会使控制的车辆在遇到扰动以后（处于steady state）要很久才能回到参考线上，这在某些情况下势必会使车辆处于一个危险的境况。

PID控制就是由这三项共同决定的，还有其他应用于无人驾驶汽车的高级控制算法，但是他们都和我们介绍的PID控制的原理相似。

我们发现其实PID实现确实不难，但是三个系数的选择却很难，那么如何选择PID系数呢?

我们可以在我们的控制循环中通过一定的算法不断尝试，下面我提供给大家一种寻找参数的算法：



具体的算法见我的C++代码实例。

PID C++代码

1.pid.cpp

```
#include <limits>
#include <iostream>
#include "PID.h"
```

```
//using namespace std;

PID::PID() {}

PID::~PID() {}

void PID::Init(double Kp, double Ki, double Kd) {
    parameter.push_back(Kp);
    parameter.push_back(Ki);
    parameter.push_back(Kd);

    this->p_error = 99999999.;
    this->d_error = 0.0;
    this->i_error = 0.0;
```

2. pid.h

```
 #ifndef PID_H
#define PID_H

#include <cmath>
#include <vector>

class PID {
private:
    int step;
    std::vector<double> changes;
    double best_error;
    double total_error;
    int index_param;

    int val_step;
    int test_step;

    int fail_counter;

    void IndexMove();
```

3.用法

在你的实际控制循环中，调用：

```
 PID pid;
pid.Init(0.3345, 0.0011011, 2.662); //your init parameters

for (in your control loop) {
  pid.UpdateError(cte);
  steer_value = pid.TotalError();
}
```

参考：