**第十七届全国大学生智能汽车竞赛**

**室外ROS无人车竞速赛（高教组）**

**技 术 手 册**

学 校：XXXX学院

队伍名称：XX

参赛队员：XX、XX、XX

带队教师：赵娟

关于技术手册使用授权的说明

本人完全了解全国大学生智能汽车竞赛关于保留、使用技术手册和研究论文的规定，即：参赛技术手册内容著作权归参赛者本人，比赛组委会和赞助公司可以在相关主页上收录并公开参赛作品的设计方案、技术手册、源码以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：

带队教师签名：

日 期： 2022.\*\*.\*\*

（本页需要签字后，替换为扫描页）

**技术手册撰写注意事项：**

了解到实验室队伍老生带新生容易断带、知识不能很好的积累沉淀转化等痛点，本次技术手册的内容更加聚焦于技术总结，知识整理。力图将技术手册打造为老队员、新队员、指导老师之间共用的知识纽带。通过几年的打磨，便于带队老师开设自动化控制、自动驾驶、人工智能、软件编程等相关课程以及开展新队员的筛选工作，便于新队员更快的传承已有的知识储备，使得实验室队伍建设构成闭环。

技术手册内容主要是将智能车作为硬件载体，以自动化控制、自动驾驶、人工智能、软件编程等专业为导向，构建一套用于学习的实验体系。书写的内容包括：

(1)参赛队员在备赛期间，结合比赛要求，结合机械、电控、编程等课本上的知识对智能车进行量化测试、局部优化的内容;

(2)参赛队员在比赛规则以外的其他方面结合智能车做的一些技术探索;

(3)参赛队员在实现比赛功能过程中，做的技术探索。

技术手册书写注意事项如下：

（1）提交形式为**实验指导书+源码；**

(2) 所有内容，需要以智能车作为唯一硬件载体；

(3) 提供不少于5个实验，以**实验指导书+源码**的形式提交；

(4) 每一个实验都要保证可以在**原版智能车**(学校最初拿到车时候的硬件、系统)上进行复现测试；

(5) 实验指导书书写时需要体现如何调用提供的源码，便于复现验证；

(6) 实验内容不得照搬原有智能车使用手册和视频资料，若与已有知识储备内容完全重复，视为该实验未书写；

技术手册总分30分，评判标准如下：

(1)每个实验最高分6分，总体不少于5个实验；

(2)每个实验得分根据实验难易程度、书写清晰度、实验间是否有关联性、创新性等因素进行判定；

(3)若书写多于5个实验，逐个实验判分后，以分数最高的5个实验的成绩加和作为最终成绩。

### 目录

实验一 车模底盘的控制实验 2

实验二 tf 静态坐标变换 7

实验三 第一圈：雷达数据分析，确定锥桶坐标实验 15

实验四 第一圈：基于锥桶坐标的PID控制器设计实验 22

实验五 第一圈：bag工具实验 28

实验六 第一圈：对控制器稳定性加强实验 32

实验七 第一圈与第二圈衔接处理实验 37

实验八 第二圈：栅格地图滤波及连线实验 41

实验九 第二圈：规划器选择及其参数调整实验 46

实验十 第二圈：整体两圈复现指令及整体流程 53

[实验十一 SLAM功能调试](#_bookmark0) 58

[实验十二 视觉功能调试](#_bookmark0) 63

[实验十三 opencv数据处理](#_bookmark0) 65

[实验十四 机器人运动学原理](#_bookmark0) 67

... ...

**(以上目录仅供参考，具体实验名称及实验内容由学生自主设定)**

实验案例书写具体要求如下：

实验案例需言简意赅，便于后续师弟师妹消化传承；

实验案例撰写可自行创新发挥，新颖的案例更加容易获得高分；

每个实验需要包含实验目的、实验内容、实验器材、实验原理、实验步骤这几部分；

实验目的：参考模板，提炼若干条

实验内容：针对实验目的，开展的具体操作总结

实验器材：罗列实验中用到的器材设备、传感器、工具、测量仪器等

实验原理：描述实验中硬件实现原理、物理相关基础、算法实现原理等

实验步骤：实验开展过程中具体的运行指令、操作方式、测试方式、代码讲解等

### 实验一 车模底盘的控制实验

**实验目的：**

1、熟悉 Linux命令

2、熟悉 ROS 话题通信；

3、了解车模工作流程

4、实现一个发布者；

**实验内容：**

1、通过键盘控制小车进行

2、向“~/car/cmd\_vel”话题发布 Twist 信息；

**实验仪器：**

ROS 智能车、电脑

**实验原理：**

通过修改 Twist 信息，并将其发布至“~/car/cmd\_vel”话题，来实现对小车舵机与电机的控制。

**实验步骤：**

**<1> 启动车**

roslaunch racecar Run\_car.launch

**<2> 启动键盘控制**

cd racecar/src/racecar/src

python3 racecar\_teleop.py

**具体控制方法如下：**

**加减速为 W，S.**

**U：向左转向并前进**

**I：前进**

**O：向右转向并前进**

**J：左转**

**K：停止**

**L：右转**

**<3> 在自己虚拟机上实现一个发布者**

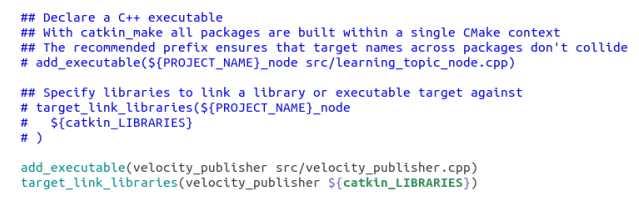
**发布turtle1/cmd vel话题，消息类型geometry\_msgs::Twist**



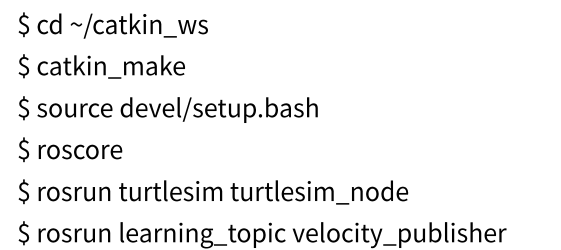
**配置发布者代码编译规则**

设置需要编译的代码和生成的可执行文件

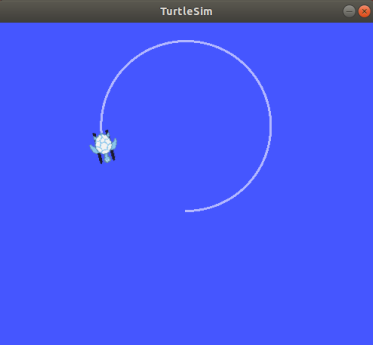
设置链接库:



**编译并运行发布者**



**实验效果**



之后还可以去创建一个发布者代码 (Python)

**实验二 tf 静态坐标变换**

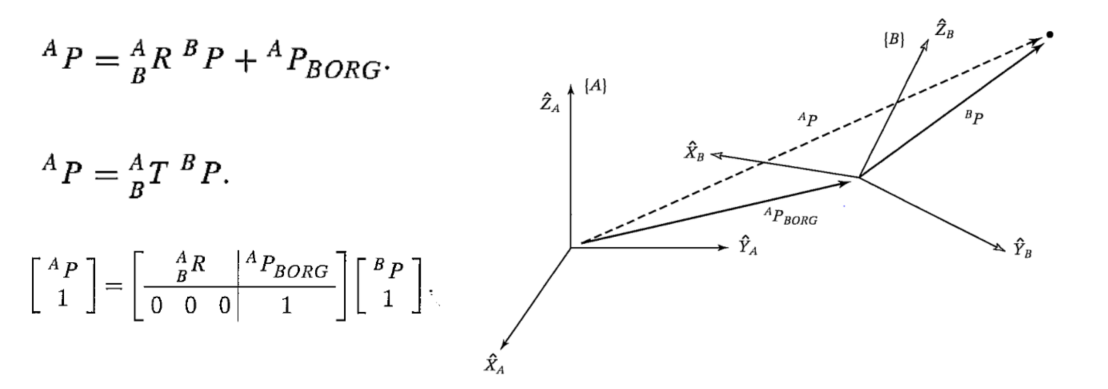
**实验目的：**实现不同坐标系下的点或者物体之间位置的转换。

**实验内容：**通过调用 ros 系统中封装好的包（把旋转量矩阵转换为四元数），将当前坐标进行变换。

**实验仪器：**ROS 智能车、电脑。

**实验原理：**

**机器人中的坐标变换**

****

**TF功能包功能：（小车中不需要全部用到）**

五秒钟之前，机器人头部坐标系相对于全局坐标系的关系是什么样的?

机器人夹取的物体相对于机器人中心坐标系的位置在哪里?

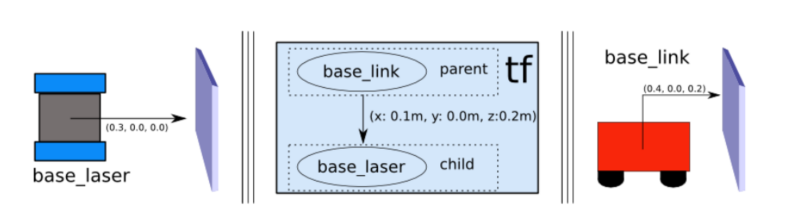
机器人中心坐标系相对于全局坐标系的位置在哪里?

**TF坐标变换实现**

广播TF变换

监听TF变换

**坐标系之间的数据变换**



**实验步骤：**

**1、 获取坐标参数**

通过打印坐标相关信息，获取坐标的参数，其中较为常用的消息类型是：

geometry\_msgs/TransformStamped 和geometry\_msgs/PointStamped,

本次实验使用的是

geometry\_msgs/TransformStamped,

在终端 terminator 中输入 rosmsginfo geometry\_msgs/TransformStamped

打印输出后得到结果为:

std msgs/Header header

uint32 seq

time stamp

string frame idstring child frame idgeometry\_msgs/Transform transformgeometry\_msgs/vector3 translation

float64 xfloat64 y

float64 z

geometry\_msgs/Quaternion rotation

float64 x

float64 y

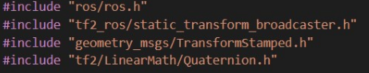
float64 z

float64 w

**2、 实现静态坐标变换**

在智能车工作空间下建立一个 cpp 文件以建立发布者

首先，对以下头文件进行包含

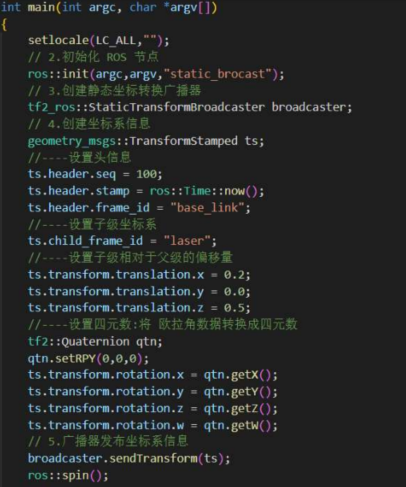


另外建立一个 cpp 文件用于作为接收者，下面是接收者要包含的头文件：

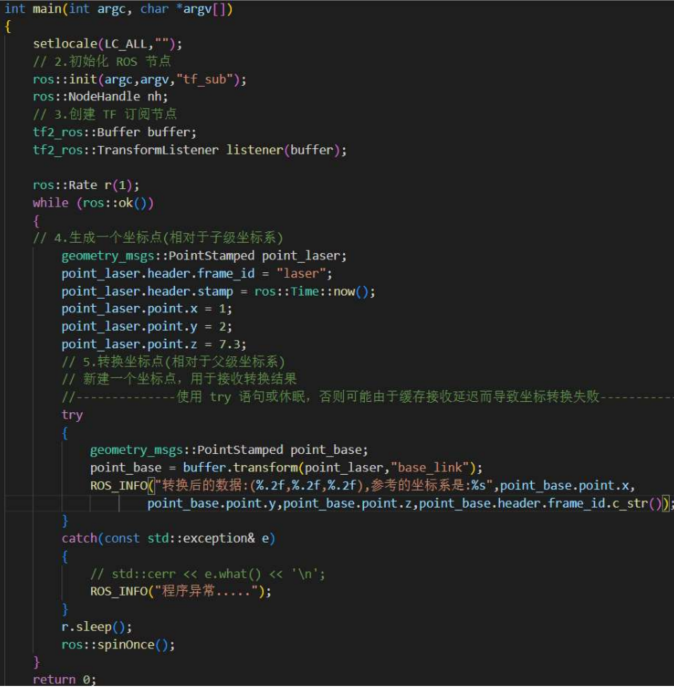


下面进行主体代码的编写：

发布者主体代码编写，请注意注释内容



接收者主体代码编写，请注意注释内容



之后这两个文件加入编译文件中生成可执行文件，调用，这样就能输出每个相对与子坐标系的坐标现在在父坐标系下的坐标了。

**实验三 PID 算法实验**

**实验目的：**

1、熟悉PID算法的两种形式，即位置式 PID 、增量式 PID

2、根据算法公式，参考所给代码，自己实现一个PID 控制程序；

3、在理解PID 算法原理的基础上，进行参数的整定

**实验内容：**

1.PID 算法的基本原理；

2.位置式 PID 与增量式 PID；

3.PID 算法的 python 代码实现；

4. PID 算法的可视化演示。

**实验仪器：**

电脑

**实验原理：**

PID 控制器（Proportional-Integral-Derivative Controller）是一种控制系

统中常用的控制方式，它由三部分构成：

比例 P（proportional）

我们先说PID中最简单的比例控制，抛开其他两个不谈。还是用一个经典的例子吧。假设我有一个水缸，最终的控制目的是要保证水缸里的水位永远的维持在1米的高度。

假设初始时刻，水缸里的水位是0.2米，那么当前时刻的水位和目标水位之间是存在一个误差的error，且error为0.8.这个时候，假设旁边站着一个人，这个人通过往缸里加水的方式来控制水位。如果单纯的用比例控制算法，就是指加入的水量u和误差error是成正比的。即  
u=kp\*error（假设kp取0.5，）

那么t=1时（表示第1次加水，也就是第一次对系统施加控制），那么u=0.5\*0.8=0.4，所以这一次加入的水量会使水位在0.2的基础上上升0.4，达到0.6.

接着，t=2时刻（第2次施加控制），当前水位是0.6，所以error是0.4。u=0.5\*0.4=0.2，会使水位再次上升0.2，达到0.8.

如此这么循环下去，就是比例控制算法的运行方法。  
可以看到，最终水位会达到我们需要的1米。

但是，单单的比例控制存在着一些不足，其中一点就是 –稳态误差！

像上述的例子，根据kp取值不同，系统最后都会达到1米，只不过kp大了到达的快，kp小了到达的慢一些。不会有稳态误差

。

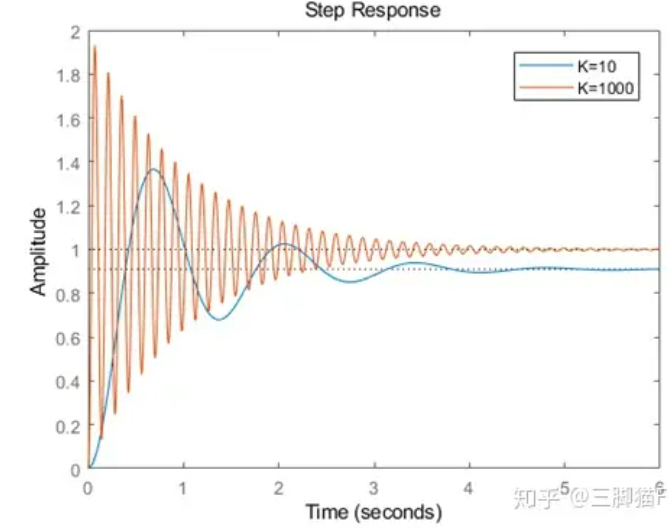
但是，考虑另外一种情况，假设这个水缸在加水的过程中，存在漏水的情况，假设每次加水的过程，都会漏掉0.1米高度的水。仍然假设kp取0.5，那么会存在着某种情况，假设经过几次加水，水缸中的水位到0.8时，水位将不会再变换！

因为，水位为0.8，则误差error=0.2. 所以每次往水缸中加水的量为u=0.5\*0.2=0.1.同时，每次加水，缸里又会流出去0.1米的水！！！加入的水和流出的水相抵消，水位将不再变化！

也就是说，我的目标是1米，但是最后系统达到0.8米的水位就不再变化了，且系统已经达到稳定。由此产生的误差就是稳态误差了。

（在小车控制中，摩擦阻力就相当于是“漏水”，）  
所以，单独的比例控制，在很多时候并不能满足要求。

下面是利用matlab进行仿真时展现的不同K值时闭环系统的step response。



显然K只要不断增大，稳态值就会无限趋近于1，稳态误差不断减小。但给定任意一个K，其最终的稳态值与1之间永远存在一个差值。

**积分 I（integral）**

还是用上面的例子，如果仅仅用比例，可以发现存在暂态误差，最后的水位就卡在0.8了。

于是，在控制中，我们再引入一个分量，该分量和误差的积分是正比关系。所以，比例+积分控制算法为：

**u=kp\*error+ ki\*∫ error**

还是用上面的例子来说明，第一次的误差error是0.8，第二次的误差是0.4，至此，误差的积分（离散情况下积分其实就是做累加），**∫error=0.8+0.4=1.2**.。

这个时候的控制量，除了比例的那一部分，还有一部分就是一个系数ki乘以这个积分项。

由于这个积分项会将前面若干次的误差进行累计，所以可以很好的消除稳态误差（假设在仅有比例项的情况下，系统卡在稳态误差了，即上例中的0.8，由于加入了积分项的存在，会让输入增大，从而使得水缸的水位可以大于0.8，渐渐到达目标的1.0.）这就是积分项的作用。

**微分 D（derivative）**

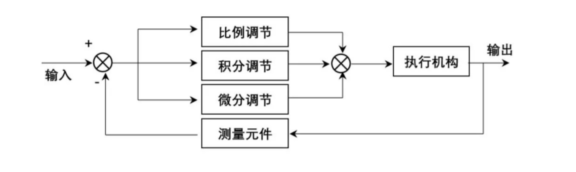
换一个另外的例子，考虑路上行驶汽车的刹车情况。平稳的驾驶车辆，当发现前面有红灯时，为了使得行车平稳，基本上提前几十米就放松油门并踩刹车了。当车辆离停车线非常近的时候，则使劲踩刹车，使车辆停下来。整个过程可以看做一个加入微分的控制策略。

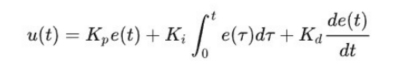
微分，说白了在离散情况下，就是error的差值，就是t时刻和t-1时刻error的差，即u=kd\*（error（t）-error（t-1）），其中的kd是一个系数项。可以看到，在刹车过程中，因为error是越来越小的，所以这个微分控制项一定是负数，在控制中加入一个负数项，他存在的作用就是为了防止汽车由于刹车不及时而闯过了线。

从常识上可以理解，越是靠近停车线，越是应该注意踩刹车，不能让车过线，所以这个微分项的作用，就可以理解为刹车，当车离停车线很近并且车速还很快时，这个微分项的绝对值（实际上是一个负数）就会很大，从而表示应该用力踩刹车才能让车停下来。

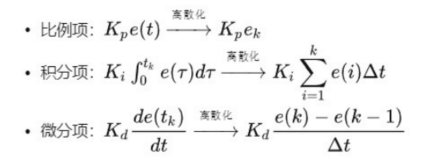
切换到上面给水缸加水的例子，就是当发现水缸里的水快要接近1的时候，加入微分项，可以防止给水缸里的水加到超过1米的高度，说白了就是减少控制过程中的震荡。

PID控制：

PID 算法的公式：



将其离散化，得到：



位置式 PID：



求增量



增量式 PID：



**增量式与位置式各自的优缺点**

位置式 PID 优缺点

**优点：**

位置式 PID 是一种非递推式算法，可直接控制执行机构（如平衡小

车）

u(k)的值和执行机构的实际位置（如小车当前角度）是一一对应的，因此

在执行机构不带积分部件的对象中可以很好应用

**缺点：**

每次输出均与过去的状态有关，计算时要对 e(k)进行累加，运算工作

量大。

增量式 PID 优缺点

**优点：**

误动作时影响小，必要时可用逻辑判断的方法去掉出错数据。

手动/自动切换时冲击小，便于实现无扰动切换。当计算机故障时，仍能保

持原值。

算式中不需要累加。控制增量Δu(k)的确定仅与最近 3 次的采样值有

关。

**缺点：**

积分截断效应大，有稳态误差；溢出的影响大。

有的被控对象用增量式则不太好；

**实验步骤：**

**1. 搭建实验环境**

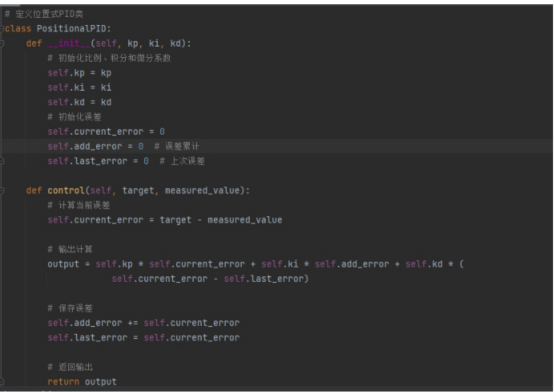
为了方便进行移植，实验环境为 **Python3 +Matplotlib**

其中 Matplotlib 是 Python 中类似 MATLAB 的绘图工具，可实现 PID 可视化功能，协助 PID 算法调参**。**

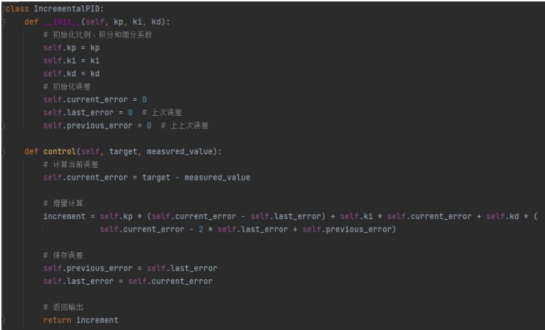
**2. 位置式 PID 与增量式 PID 的 python 算法实现**

**需要自己结合公式复现，不要照抄**

**位置式 PID**

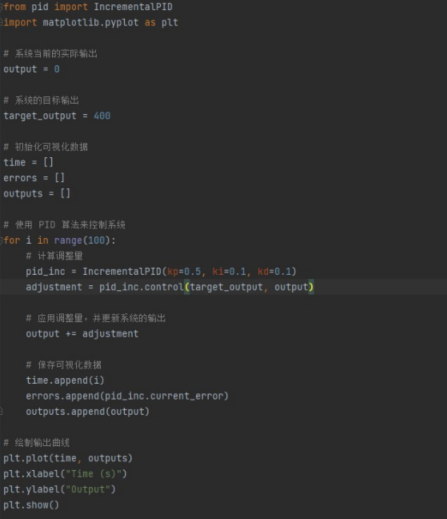
****

**增量式 PID**

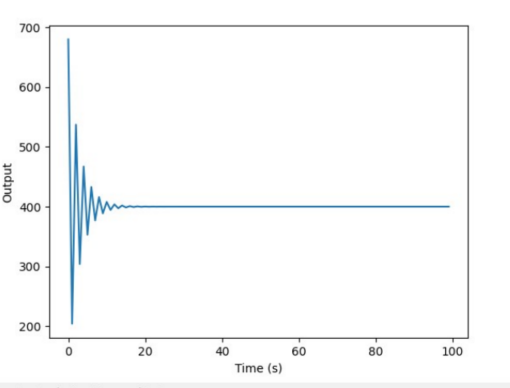
**2. 3.位置式 PID 与增量式 PID 的 python 算法实现**

由于用 python 写的话不像是在单片机上，单片机编程有一个重要的函数中断函数．Python中，我们使用 time 库来控制函数的运行时间的间隔。

一个增量式 PID 算法可视化程序：



当 **kp=1，ki=0.5，kd=0.2** 时运行后输出下述图像



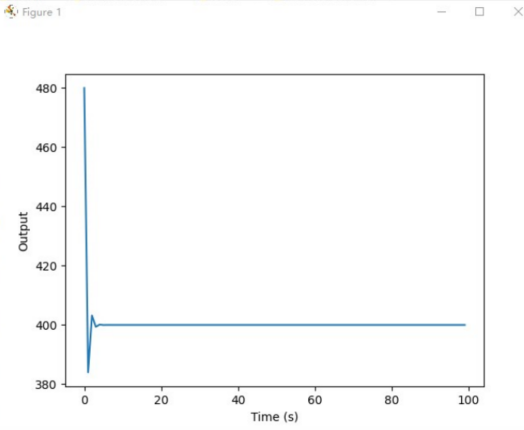
通过可视化可以清晰观察目标数到达期望值过程中产生了震荡，我们需要调参来

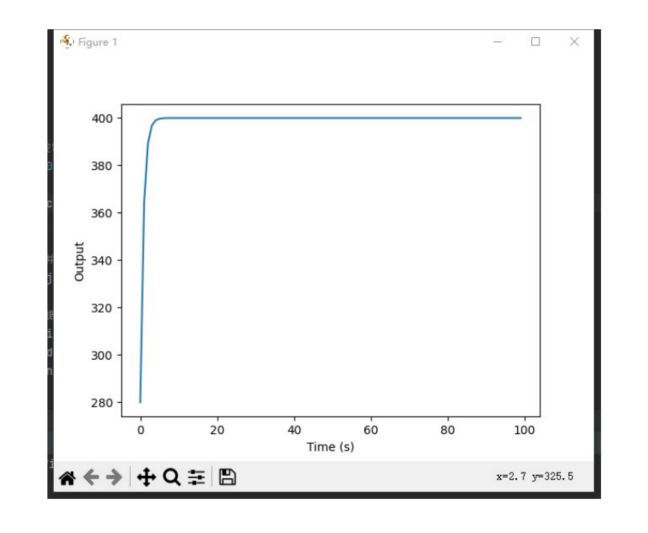
使目标值顺滑地到达期望值。

PID整定口诀

参数整定找最佳，从小到大顺序查  
先是比例后积分，最后再把微分加  
曲线振荡很频繁，比例度盘要放大  
曲线漂浮绕大湾，比例度盘往小扳  
曲线偏离回复慢，积分时间往下降  
曲线波动周期长，积分时间再加长  
曲线振荡频率快，先把微分降下来  
动差大来波动慢，微分时间应加长  
理想曲线两个波，前高后低四比一  
一看二调多分析，调节质量不会低

下面是通过调整参数后的图像。





代码文件：

pid.py,

PIDVisualization.py

使用方法：使用 Visual Studio Code 或 PyCharm 将两个文件放置在同一工程

下，直接编译运行，在此过程可通过调整期望值（参数 target\_output）和 PID

算法的三个参数进行编译输出可视化，实现 PID 算法调参。

下面介绍在小车上实际应用时参数的调节方法

**4. 熟悉 PID 算法在小车上实际应用时参数的调节方法**

**·kp**

该参数在 PID 算法中的作用最主要体现在反应速度方面，提高该参数的值可

以显著提高小车的反应速度，同时也能够起到消除静态误差的作用。然而，该参

数过大时，将会造成小车在寻路时明显摆动。

当发现小车反应速度过慢时，应增大该参数；当发现小车左右摆动明显时，

应减小该参数。

**·ki**

该参数在 PID 算法中起到消除静态误差的作用，主要在转弯时体现。但在实

际应用中，累积的误差可能导致小车反应速度大幅降低，因此，在实际应用中，

必须对积分值进行限幅，减小累积误差对反应速度的影响。

当发现小车转弯时容易碰撞外圈时，应增大该参数；转弯时碰撞内圈，或是

转弯后反应速度明显降低，则应减小该参数。

**·kd**

该参数在 PID 算法中主要起到减少抖动的作用。

在小车左右晃动明显时，可以适当增大该参数；当小车出现转弯困难等现象时，可以适当减小该参数

**实验九 激光雷达寻路实验（C++实现）**

**实验目的：**

掌握采用激光雷达识别锥桶并寻路的方法。

**实验内容：**

1、获取激光雷达的测量数据并处理；

2、使用获取到的数据进行寻路操作。

**实验仪器：**

ROS 智能车、激光雷达。

**实验原理：**

利用激光雷达的测量数据识别锥桶，判断锥桶所处的位置，并通过锥桶位置

进行寻路。

**实验步骤：**

**1. 了解激光雷达测量数据**

激光雷达传入的测量数据消息类型为 sensor\_msgs 下的 Laserscan，其

主要由以下几部分构成：

· Header header，包含 seq、stamp、frame\_id，seq，该内容非本

实验重要内容，故不详述；

· float32 angle\_increment，每次扫描增加的角度

· float32 time\_increment，扫描的时间间隔

· float32 scan\_time，扫描的时间间隔

· float32 range\_min，距离最小值

．54．

· float32 range\_max，距离最大值

· float32 angle\_min，开始扫描时的角度

· float32 angle\_max，扫描结束时的角度

· float32[] ranges，距离信息

· float32[] intensities，强度数据

其中，我们需要的信息是 angle\_min、angle\_max、angle\_increment

和 ranges。

通过 angle\_min、angle\_max、angle\_increment 可获知数组长度，通

过 ranges 中各数据的位置、angle\_min 和 angle\_increment 即可获知周围

各角度的距离信息，即可获得周围任意边界点的极坐标信息。

例：设激光雷达所处位置为原点，其正前方为 x 轴正方向，正右方为 y

轴正方向，由上往下看时激光雷达顺时针旋转，则可根据 ranges[i]的信息

获知，此时由激光雷达正前方顺时针旋转 theta=i\*angle\_increment +

angle\_min（弧度）的方向上，边界与原点的距离为 ranges[i]。

由此，我们便可得到激光雷达获取到的任意边界点的极坐标信息。

**2. 代码实现获取激光雷达信息并处理**

首先输入以下命令，并输入密码，安装 sensor\_msgs 依赖包：

sudo apt install ros-sensor-msgs

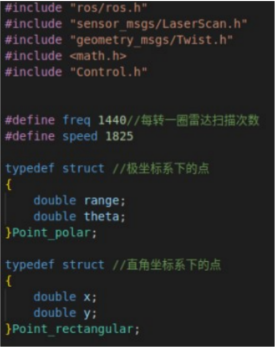
安装完成后，编写 C++代码

由于本实验中需实现在回调函数中发布数据，因此需定义一个类如下：



其中，/scan 为雷达话题，/car/cmd\_vel 为小车底盘控制话题

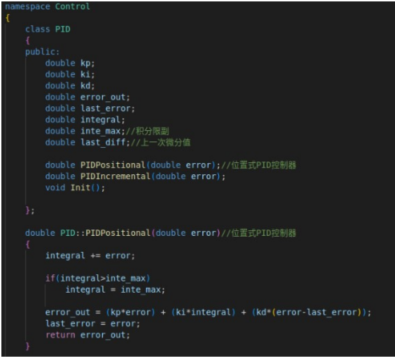
随后包含头文件、定义所需变量并编写 main 函数如下：

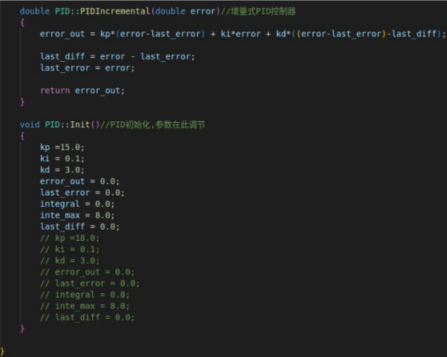




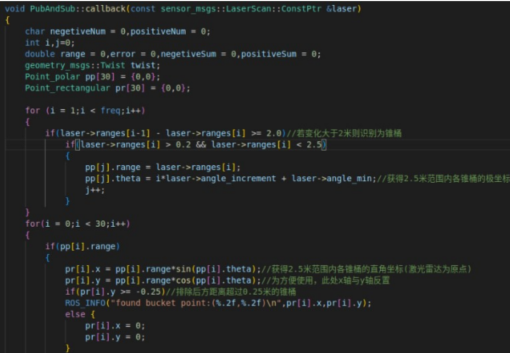
其中 Control.h 为编写的 PID 文件，内容为 PID 控制器（如下），在此

处不作细讲。





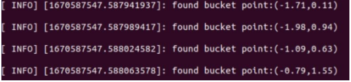
随后进行回调函数的内容编写，如下：



关于识别锥桶，在无外物干扰的情况下，对 ranges 进行遍历，若距离产

生突变，则说明此处极有可能是锥桶，本实验采取该方法，当距离的变化大于 2

米时识别为锥桶。运行节点，能够正常输出锥桶坐标如下，则证明代码无误：



至此，我们已经可以获取周围 2.5 米内的锥桶的直角坐标信息。

**3. 代码实现激光雷达寻路**

通过锥桶的坐标信息实现寻路的方法有多种，以下是一种极其简单且实用

的方法：

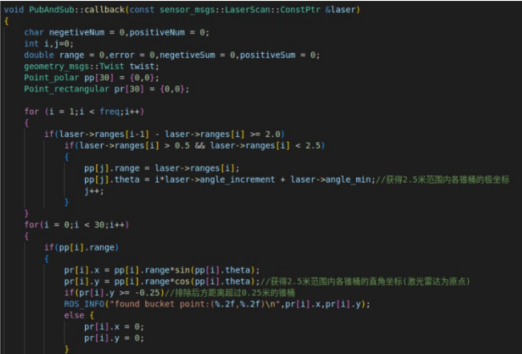
在进行一些筛选项的前提下，将锥桶的 x 坐标相加，得出偏差，再通过

PID 控制器处理并输出。

具体步骤如下：

首先初步处理数据，对上面编写的回调函数进行修改和补充，排除距离激光雷达 0.5 米以内的激光点云信息，以使小车不会过于靠近锥桶（当发现小车

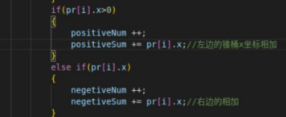
时常碰撞锥桶时，应增大该参数），并使锥桶的 x 坐标相加

修改内容为：



同时在第二个 for 循环内第一个 if 语句（即：if（pp[i].range))中加入以

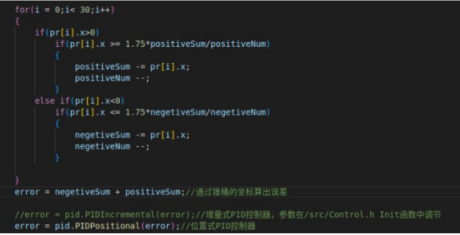
下代码：



此时我们分别得到了左右锥桶的 x 坐标之和与其数量，但此时数据中被远端的锥桶影响较多，容易出现直角弯转不过的问题。因此，我们还需滤除部分信息。

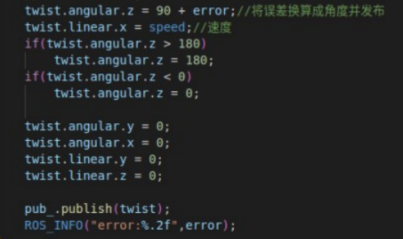
本实验采用的是一种简单的方式：将 x 坐标超过平均值 1.75 倍的锥桶

信息滤除。代码实现如下：



该参数可调节，具体视赛道状况而定。经过实验，该参数在 1.65-1.85 范围内均可，过小可能会导致数据丢失，过大可能会造成干扰。

最后，在回调函数中将数据组织并发布即可，代码如下：



将小车放在赛道中，打开 Run\_car.launch 和本实验编写的节点，即可实

现激光雷达自主寻路。