# 机器人导航

技术报告

学 院：XXXX 学院

班 级：XX

姓 名：XX

学 号：X

**目录**

实验一 slam建图 [2](#bookmark1)

实验二 PID控制 [7](#bookmark2)

实验三 Move\_base导航：小车A点到B点 15

实验四 2D雷达数据处理 22

实验五 多点导航 28

**实验一** **slam建图**

**实验目的：**

1 、了解导航模块中的组成部分以及相关概念

2 、了解常用的 SLAM 实现方法，比如: gmapping、hector\_slam、cartographer等

3 、学习使用较为常见的gmapping构建地图；

**实验内容：**

1、搭建slam建图环境；

2、使用gmapping构建地图。

**实验仪器：**

gazebo仿真环境包robot\_task

gmapping 包(用于构建地图):sudo apt install ros-<ROS版本>-gmapping

地图服务包(用于保存与读取地图):sudo apt install ros-<ROS版本>-map-server

navigation 包(用于定位以及路径规划):sudo apt install ros-<ROS版本>-navigation

**实验原理：**

SLAM(simultaneous localization and mapping),也称为CML (Concurrent Mapping and Localization), 即时定位与地图构建，或并发建图与定位。SLAM问题可以描述为: 机器人在未知环境中从一个未知位置开始移动,在移动过程中根据位置估计和地图进行自身定位，同时在自身定位的基础上建造增量式地图，以绘制出外部环境的完全地图。

gmapping 是ROS开源社区中较为常用且比较成熟的SLAM算法之一，gmapping可以根据移动机器人里程计数据和激光雷达数据来绘制二维的栅格地图。

**实验步骤：**

**<1> 搭建仿真环境，编译功能包**

详细参考[Introduction · Autolabor-ROS机器人入门课程《ROS理论与实践》零基础教程](http://www.autolabor.com.cn/book/ROSTutorials/) 第七章内容

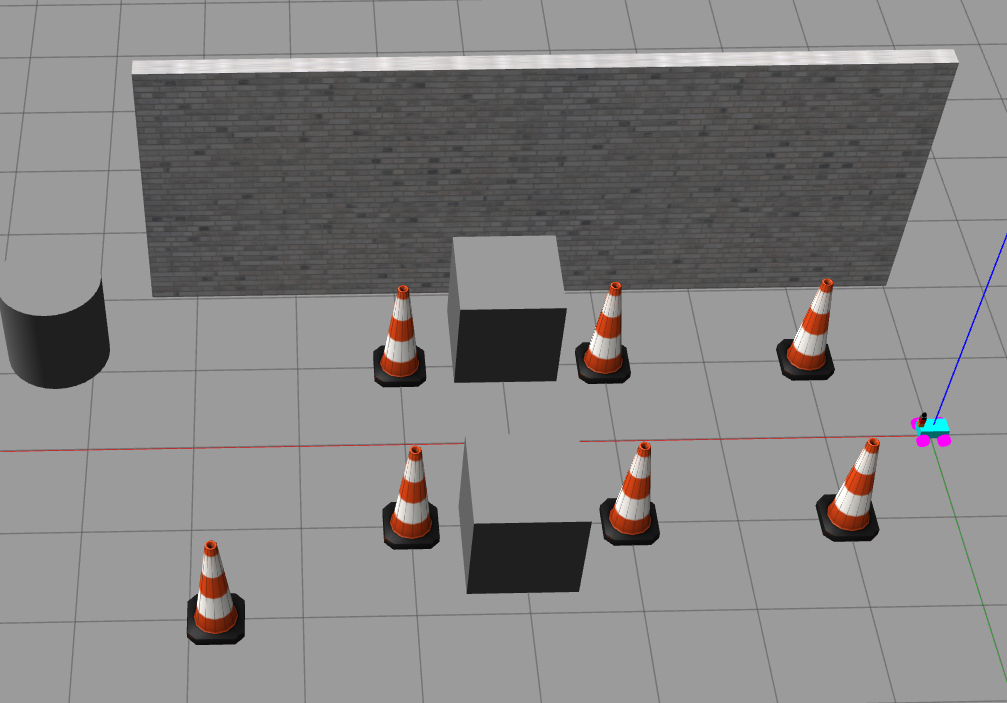
**<2> 打开三个终端刷新环境变量后，分别输入以下命令**

刷新环境变量

source devel/setup.bash

打开gazebo仿真环境

roslaunch robot\_task ucar\_robot.launch



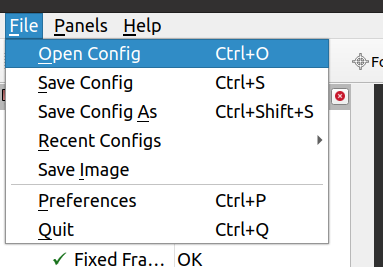
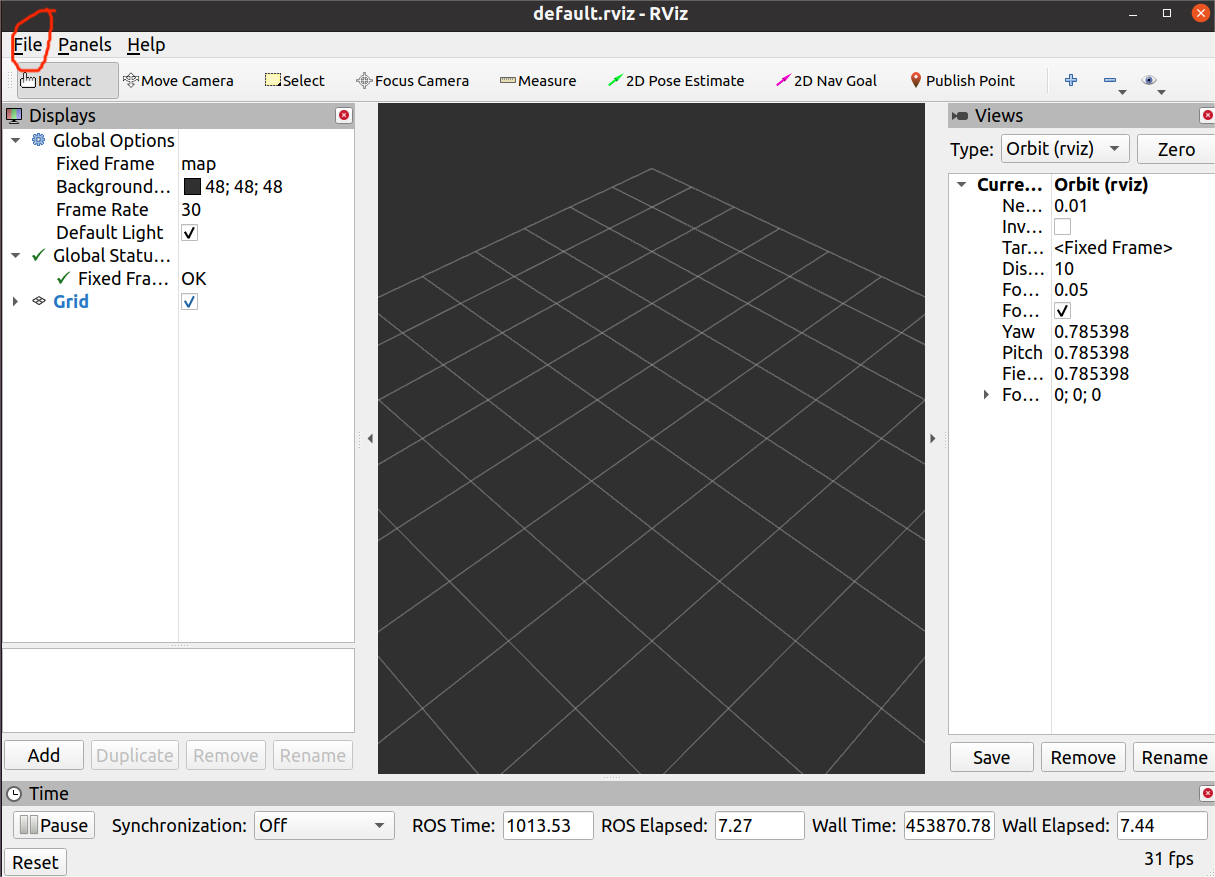
打开建图模块

roslaunch nav\_demo nav07\_slam\_auto.launch

打开rviz

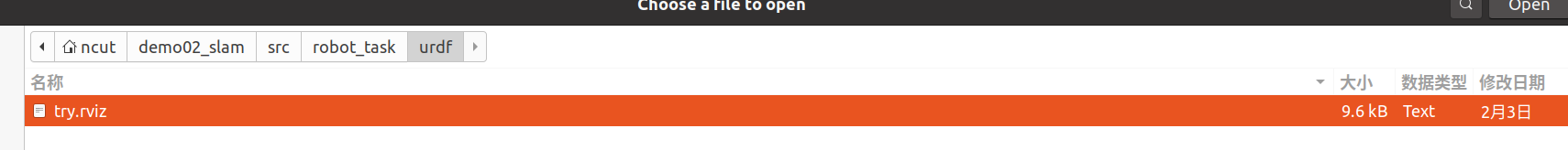
rviz

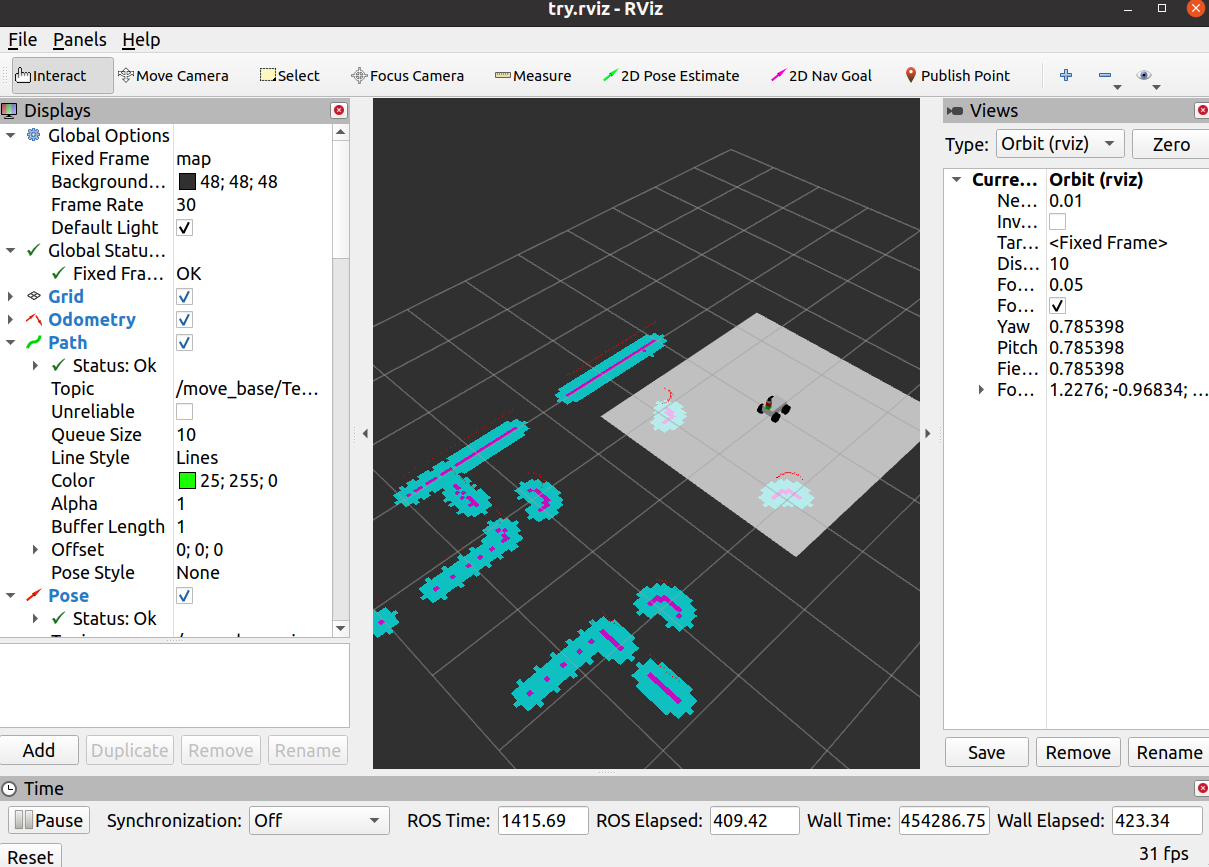
选择file/open config



选择 主目录/工作空间/src/robot\_task/urdf/try.rviz

打开





（可以将其写入launch文件打开，也可以自行配置rviz）

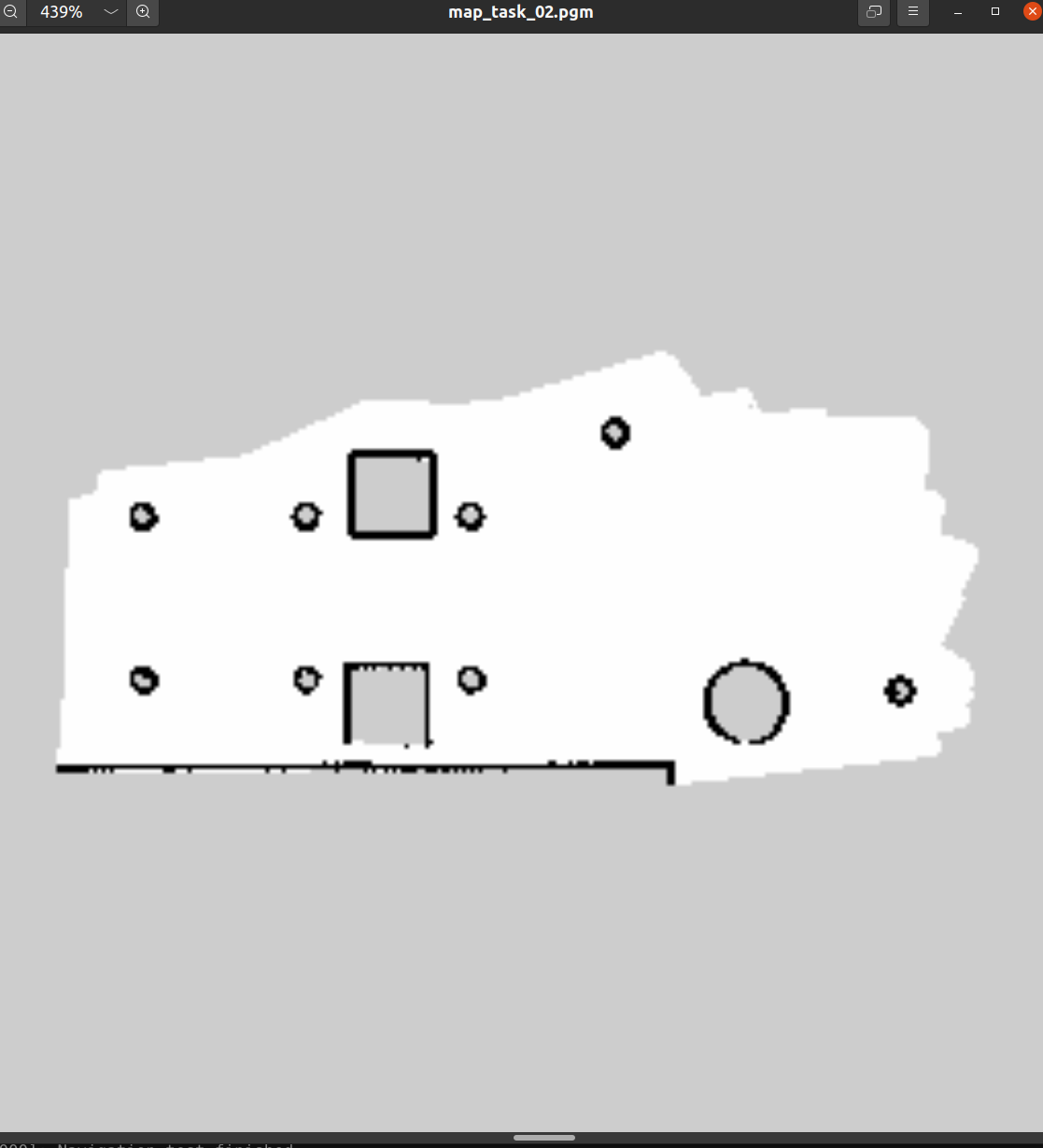
**<3> 使用 rviz 中的 2D Nav Goal 控制小车运动，进行建图**

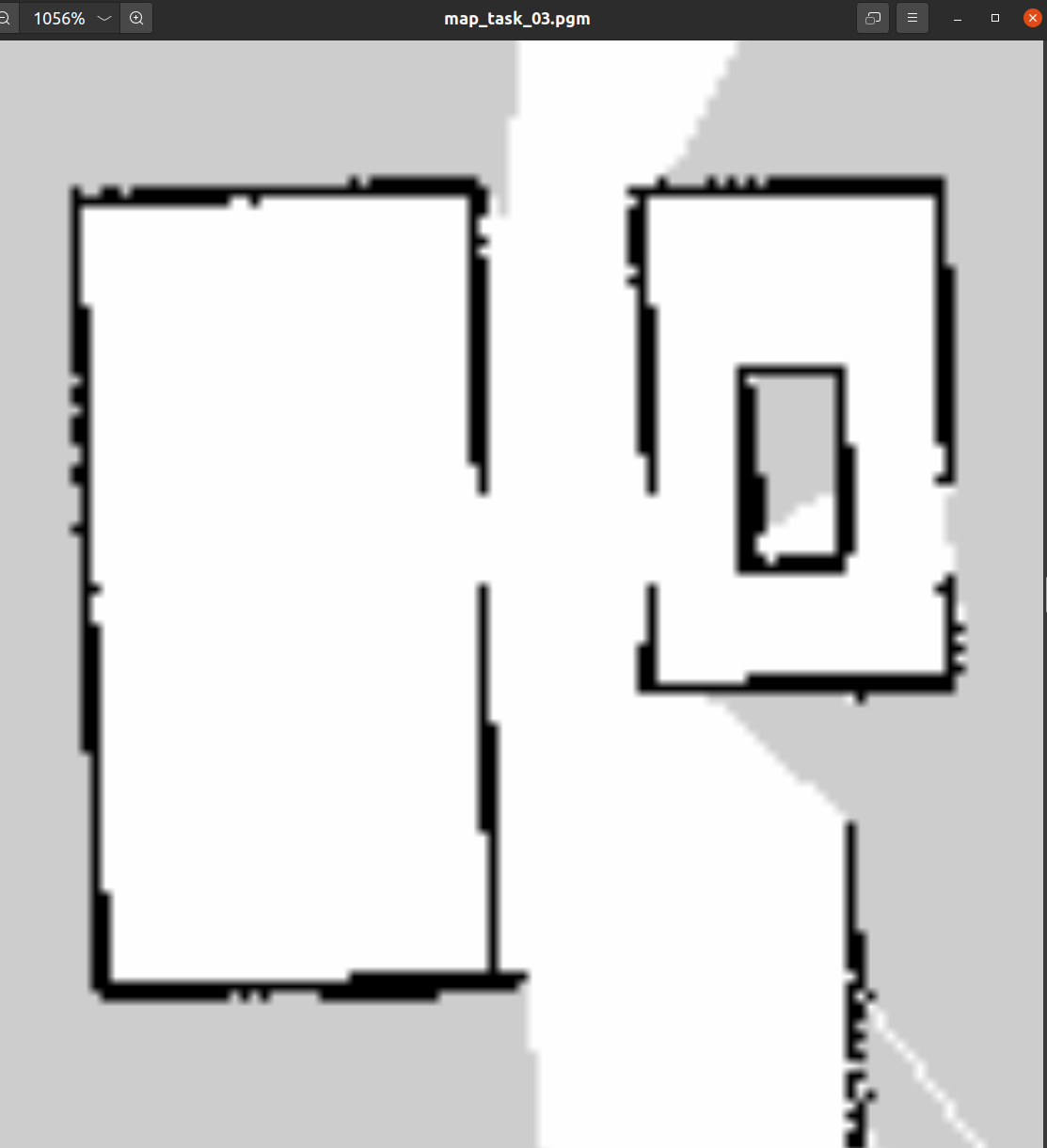
**<4> 建图完成后，保存地图**

roslaunch nav\_demo nav02\_map\_save.launch

保存好的地图后可查看







**其他建图方式：**

Hector SLAM

不需要里程计数据，使用高斯牛顿方法，直接使用激光雷达数据估算里程计信息。所以，只需要激光雷达数据，即可完成建图。

基于优化的算法（解最小二乘问题），优缺点：不需要里程计，但对于雷达帧率要求很高40Hz，估计6自由度位姿，可以适应空中或者地面不平坦的情况。初值的选择对结果影响很大，所以要求雷达帧率较高。在缓慢移动的情况下准确性较高，相对稳定。

Cartographer

累计误差较前两种算法低，能天然的输出协方差矩阵，后端优化的输入项。成本较低的雷达也能跑出不错的效果。适合构建大型地图。

Cartographer是基于图优化的方法建图算法，其主要通过闭环检测来消除构图过程中产生的累积误差。用于闭环检测的基本单元是submap。一个submap是由一定数量的laser scan构成。将一个laser scan插入其对应的submap时，会基于submap已有的laser scan及其它传感器数据估计其在该submap中的最佳位置。submap的创建在短时间内的误差累积被认为是足够小的。然而随着时间推移，越来越多的submap被创建后，submap间的误差累积则会越来越大。因此需要通过闭环检测适当的优化这些submap的位姿进而消除这些累积误差，这就将问题转化成一个位姿优化问题。当一个submap的构建完成时，也就是不会再有新的laser scan插入到该submap时，该submap就会加入到闭环检测中。闭环检测会考虑所有的已完成创建的submap。当一个新的laser scan加入到地图中时，如果该laser scan的估计位姿与地图中某个submap的某个laser scan的位姿比较接近的话，那么通过某种 scan match策略就会找到该闭环。Cartographer中的scan match策略通过在新加入地图的laser scan的估计位姿附近取一个窗口，进而在该窗口内寻找该laser scan的一个可能的匹配，如果找到了一个足够好的匹配，则会将该匹配的闭环约束加入到位姿优化问题中。Cartographer的重点内容就是融合多传感器数据的局部submap创建以及用于闭环检测的scan match策略的实现。

**实验二** **PID控制**

**实验目的：**

1 、了解pid相关概念

2 、学习pid控制仿真小车；

**实验内容：**

1、学习pid控制相关知识；

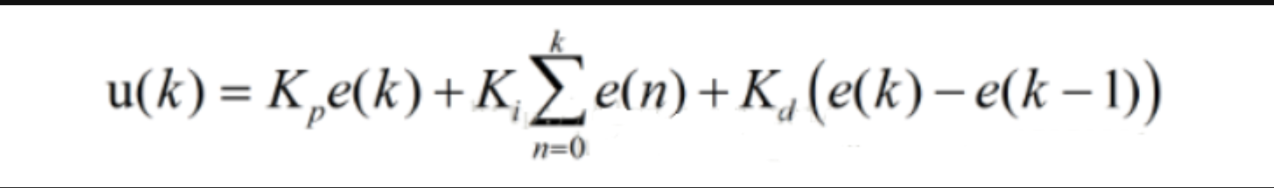
2、使用pid算法实现小车贴墙走，使小车由A到达B点。

**实验仪器：**

gazebo仿真环境包robot\_task

**实验原理：**

PID理论：

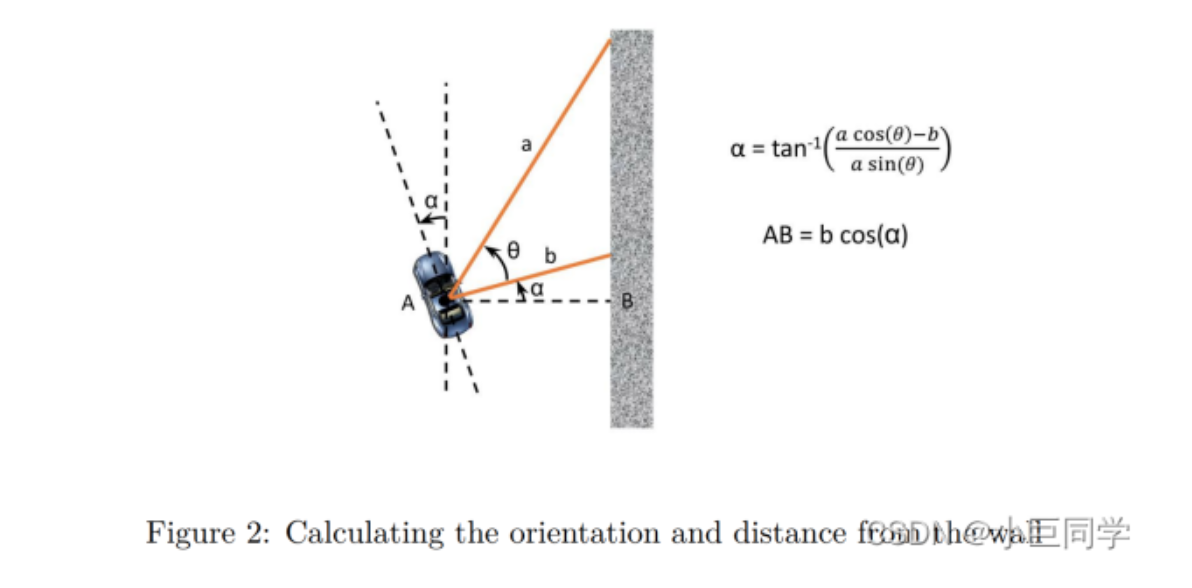


kp是比例系数

kd是微分系数，是对误差进行微分。误差error的微分是 (err2 - err1)

ki是积分系数，是对误差的积分（累加），也就是误差的无限和。作用是减小静态情况下的误差，让受控物理量尽可能接近目标值。

通过PID算法控制车始终与墙保持一定距离，沿着墙走，这个距离可根据这个公式得到，因为运动过程会产生误差，要把误差尽量消除掉，所以就要通过上面公式处理。



**实验步骤：**

<1>启动仿真环境

<2>运行pid控制代码

在工作空间下运行

rosrun run\_code test01\_pid.py

<3>调整参数，使小车能顺利到达B点

DESIRED\_DISTANCE\_RIGHT = 0.3 #激光雷达测量距离

小车在不同角度时的速度

if np.abs(steering\_angle) > np.pi / 180 \* 20.0:

speed = 0.1

elif np.abs(steering\_angle) > np.pi / 180 \* 10.0:

speed = 4

else:

speed = 5

**实验三 Move\_base导航：小车A点到B点**

**实验目的：**

1 、了解导航模块中的组成部分以及相关概念

2 、了解movebase导航框架

3 、了解teb等导航相关算法；

**实验内容：**

1、在以建好的地图中完成导航由A到B点；

**实验仪器：**

gazebo仿真环境包robot\_task

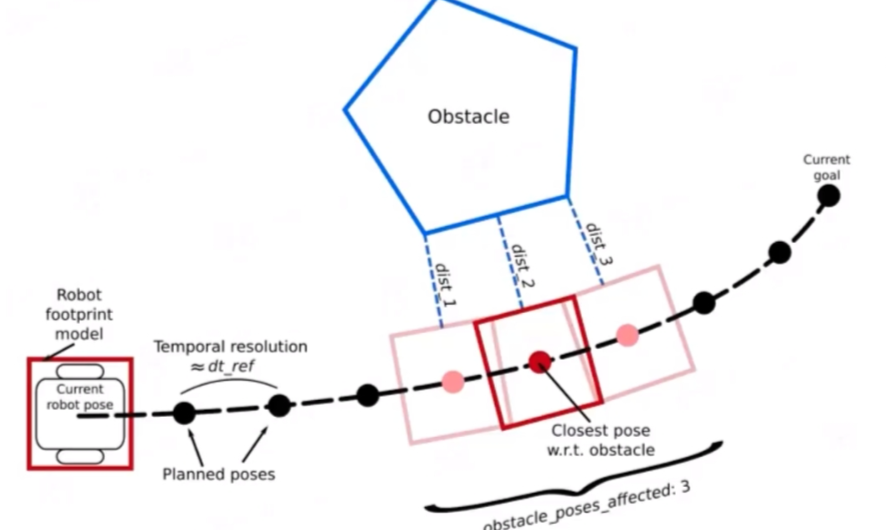
gmapping 包(用于构建地图):sudo apt install ros-<ROS版本>-gmapping

地图服务包(用于保存与读取地图):sudo apt install ros-<ROS版本>-map-server

navigation 包(用于定位以及路径规划):sudo apt install ros-<ROS版本>-navigation

**实验原理：**

**Teb算法，**像在起点和终点连一条橡皮筋，通过中途机器人的位姿约束这条皮筋，达到最快路径的效果。



**实验步骤：**

<1> 打开三个终端刷新环境变量后，分别输入以下命令

刷新环境变量

source devel/setup.bash

打开gazebo仿真环境

roslaunch robot\_task ucar\_robot.launch

打开导航模块

roslaunch nav\_demo nav08\_teb.launch

打开rviz

rviz

选择file/open config

选择 主目录/工作空间/src/robot\_task/urdf/try.rviz

打开

（可以将其写入launch文件打开，也可以自行配置rviz）

<2> 使用 rviz 中的 2D Nav Goal 设置目标点，完成导航

<3>调节参数

Teb中重要参数

max\_global\_plan\_lookahead\_dist，#向前规划最长距离

此距离1.应随车辆最大速度的增大而增大 2.不应超过激光雷达等传感器的可靠测量范围 3. 不应超过局部耗费地图的大小，即不能要求TEB对局部耗费地图以外的部分进行规划。

min\_obstacle\_dist #与障碍物最小期待距离

dt\_ref #最优路径上两姿态

此距离不固定，规划器自动根据速度大小调整这一距离，速度越大，相邻距离自然越大。

inflation\_dist #障碍物膨胀距离

max\_vel\_x\_backwards 最大倒车速度

将此速度设置为0或者负数将导致错误。禁止倒车应在penalty部分将前向行驶的权重设置得极高。

**实验四 2D雷达数据处理**

**实验目的：**

1 、了解雷达及相关概念

2 、学习处理雷达消息；

**实验内容：**

1、获取雷达消息内容；

2、依据雷达消息，实现小车定点停车。

**实验仪器：**

gazebo仿真环境包robot\_task

**实验原理：**

激光雷达是避障与 SLAM 常用的装备，ROS 中也对激光雷达作出了相应的支持，各个厂 家也有不少提供了自己的ROS 库供开发者使用，使得无论是初学者还是大神们都能轻松愉 快的使用激光雷达为自己的机器人导航。

ROS 中的 sensor\_msg s/LaserScan 提供了最基础的针对一线激光雷达消息格式：

|  |
| --- |
| # 一线激光雷达的一次扫描  Header header # timestamp 是扫描中第一束激光的获取时间  # 角度环绕 Z 轴（ Z 轴为激光雷达正上方） ，逆时钟方向增加，0 度角指向X 轴  float32 angle\_min # 扫描的起始角度 [rad]  float32 angle\_max # 扫描的结束角度 [rad]  float32 angle\_increment # 每次测量间的角度差 [rad]  float32 time\_increment # 每次测量间的时间差 [seconds]  # 这个数值会用来生成点云（而不是 scan\_time）  float32 scan\_time # 每次扫描的时间差 [seconds]  # 这个时间差是每次激光发射的时间差，上面那个是每次接收（测量）的时间差  float32 range\_min # 距离最小值 [m]  float32 range\_max # 距离最大值 [m]  float32[] ranges # 距离数据 [m]  float32[] intensities # 强度数据 [单位视设备标定]  #如果设备不产生强度数据，则该数据为空 |

**实验步骤：**

<1>启动仿真环境

刷新环境变量

source devel/setup.bash

打开gazebo仿真环境

roslaunch robot\_task ucar\_robot.launch

<2>运行程序获取雷达相关消息

rosrun run\_code test06\_lieda.py

该程序为获取距离雷达最近的障碍与雷达的距离

更改以下回调函数的内容，可获取其他雷达相关消息

def leida\_callback(data):

rospy.loginfo(min(data.ranges))

<3>运行程序完成定点停车

rosrun run\_code test04\_stop.py

**实验五 多点导航**

**实验目的：**

1 、了解导航模块中的组成部分以及相关概念

2 、了解movebase导航框架

3 、学习编写多点导航程序；

**实验内容：**

1、完成多点导航；

2、调整点位设置，已达到更好的导航效果。

**实验仪器：**

gazebo仿真环境包robot\_task

gmapping 包(用于构建地图):sudo apt install ros-<ROS版本>-gmapping

地图服务包(用于保存与读取地图):sudo apt install ros-<ROS版本>-map-server

navigation 包(用于定位以及路径规划):sudo apt install ros-<ROS版本>-navigation

**实验原理：**

[move\_base导航](https://blog.csdn.net/weixin_44917390/article/details/106624670" \t "https://cn.bing.com/_blank)

move\_base是ROS（机器人操作系统）中的一个功能包，用于实现机器人导航中的最优路径规划。它结合了全局路径规划和本地路径规划，确保机器人能够在动态环境中安全、高效地到达目标位置。

move\_base功能包

move\_base功能包主要由两个部分组成：全局路径规划和本地路径规划。

全局路径规划（global\_planner）

全局路径规划根据给定的目标位置和全局地图进行总体的路径规划。常用的算法包括Dijkstra和A\*算法，这些算法能够计算出从起点到终点的最优路径[1](https://blog.csdn.net/weixin_44917390/article/details/106624670" \t "https://cn.bing.com/_blank)。

本地路径规划（local\_planner）

本地路径规划在实际环境中执行，由于可能出现障碍物，机器人无法完全按照全局路径行驶。本地路径规划使用动态窗口方法（Dynamic Window Approach, DWA）算法，实时搜索避障路径，并计算行驶周期内的线速度和角速度，确保机器人能够避开动态障碍物[1](https://blog.csdn.net/weixin_44917390/article/details/106624670" \t "https://cn.bing.com/_blank)。

主要话题和服务

move\_base功能包提供了多个话题和服务，用于与其他ROS节点进行交互。例如：

/move\_base/goal：用于接收导航目标。

/move\_base/status：用于发布导航状态。

/move\_base/feedback：用于发布导航反馈信息

**实验步骤：**

**<1>打开三个终端刷新环境变量后，分别输入以下命令**

刷新环境变量

source devel/setup.bash

打开gazebo仿真环境

roslaunch robot\_task ucar\_robot.launch

打开导航模块

roslaunch nav\_demo nav08\_teb.launch

打开rviz

rviz

选择file/open config

选择 主目录/工作空间/src/robot\_task/urdf/try.rviz

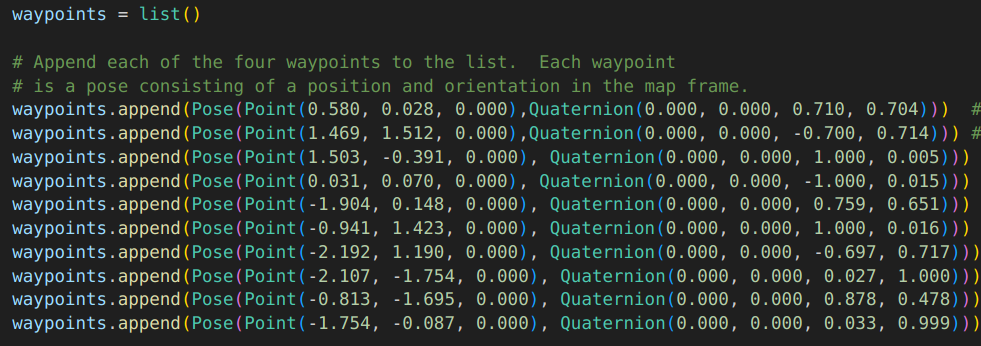
打开

（可以将其写入launch文件打开，也可以自行配置rviz）

**<2>使用 rviz 中的 2D Nav Goal 设置目标点**

**按顺序设置目标点，记录这些点的位姿信息**

**<3>将位姿信息写入程序内**



**<4>调试运行程序，调整点位信息或加入必要的中间点**

**rosrun run\_code test05\_xunhang.py**