# 自我介绍

**首先，自我介绍要有自信！把你的口头禅那个那个给去掉！！！！**

面试官您好，我叫唐娜，目前是研二的学生，就读于 吉林大学 汽车工程学院的 车辆工程专业。我的研究方向是 无人车的激光SLAM （就是激光雷达同时定位与建图）。从研一上学期我进入吉林大学的汽车仿真与控制 国家重点实验室参与项目，主要包括基于rikirobot的两轮驱动机器人的室内定位和建图、以及后轮轮毂电动车的控制项目。另外基于AVP场景的无人车定位建图的算法研究这个项目是目前还在做的项目，这个同时也是我的毕业论文的方向。

**问吴师兄的问题总结：**

1. ☑我想要下载库或者包，在官网上，怎么样才能让他下载的快一些呢，因为直接用浏览器下载实在是太慢了。解决方法：直接把链接发到迅雷 用迅雷下载
2. ☑Ubuntu16.04下载不下来文件。（上不了网络啊）
3. 怎么使用Gmapping里面的传感器融合
4. 需要改什么参数，LEGO-LOAM 里面由什么需要改才能把octoree给俺进去

标定需要学习的内容：

Automatic laser calibration, mapping, and localization for autonomous vehicles， 2011

# 面试会问到的问题！

## 恒润科技：

1. 工作的话倾向于哪方面的工作：回答：SLAM相关的定位和建图都可以。**这里可能会问道：定位建图和感知方面的区别在哪里？（这个地方主要说出来自己想做哪方面的工作就行废话不要说，什么挺感兴趣的，乱七八糟的话，他问你为什么选这个方向的时候你再扯，否则会丢分，拉低印象分！！！）、**
2. **目前我做的方向主要是和室内有关的项目，有没有尝试过室外场景的SLAM吗？--》这个地地方你要注意，最好的回答应该是：尝试过。并且把你尝试的内容，也就是你的基于LOAM 跑的包给拿过来，这方面非常的有用。这个尝试室外场景的时候，遇到过哪些问题以及相应的解决方法一定要说明白，这块主要是看你有没有真正的做过目前的项目。**
3. **用到的是哪个LOAM ？loam是一篇论文，用的是哪个包？GIthub上下载的velodyne\_Loam 这个包里 每个文件是用来干什么的？(wailedai)**
4. **听说过Advance -LOAM和Lego-LOAM吗？**
5. **对LOAM 的了解和学习上的收获部分（一个简要的说明包括流程，用到的算法 包主要分为那几块和论文主要的思想是什么都要能够说出，最好是有一个完整清晰的流程：你LOAM 从论文方面看主要用到了\*\*\*\*方法，这种发发主要是）》》》》》》**
6. **从实际跑的包的过程来看主要分为那几个文件，每个文件是用来干什么的 启动的过程是什么？需要修改哪些参数？每个参数对应的内容是什么？跑的源码是什么？效果不好的原因是什么？）原**
7. **LEGO-LOAM 坐标系垂直的的问题 怎样去解决的？？？和Octomap c++基础可能不调好，核心改的是什么参数》》》**
8. **用的是Velodyne VLP16线雷达 是水平放置的还是呈一定角度放置?-->水平放置**
9. **室内建图主要用的是Gmapping吗？cartographer也用过**
10. **能够支撑你工作的点（未来能够胜任你在SLAM上的点是什么？）**
11. **SLAM 方面上的松耦合和紧耦合的说法？**
12. **SLAM 主要分为前端和后端，你主要想做或者做的是前端还是后端的部分？--》回答的方法主要向往前端去做还是后端？为什么就说一下前端和后端的优缺点和主要处理的内容。然后说一下自己为什么想做前端或者为什么想做后端？**
13. **后端的话为什么一开始选择粒子滤波的方式吗？粒子滤波属于SLAM 方式中的松耦合的方式很少使用滤波器的方式去处理了，现在课题的方向**
14. **SLAM 目前的前沿方法或者想要达到更好的方法的话为什么还要用粒子滤波的方法？**
15. **小车上简历上写着使用GPS和IMU 的东西去使用相关的东西，怎么用的？主要是大车和小车都是怎么用的？在rikirobot上是怎么用的？IMU 的芯片主要是接收什么信息，里程计主要是接收什么信息？IMU的芯片用的什么量级的？型号？积分过程是自己做的还是直接有ROS的库直接用的？IMU的结果和激光里程计之间做的是松耦合？用粒子滤波做的处理？里程计不是耦合完以后的里程计不准。**
16. **消除LOAM 建图过程中的充盈和累积误差的问题，用图优化的方法。**
17. **图优化的了解多少。--》非线性最小二乘问题。图和边分别是什么？说明白了！！使用过图优化相关的东西吗？cartographer里面有图优化的东西 图优化相关的库有哪些？**
18. **Cartographer图优化用到的库有哪些？cere库 G2O的库 Gmapping和cartographer的优缺点，坏掉是指哪些方面坏掉了。Gmapping不能回环和走走廊的时候定位不准确的原因是什么？怎么样进行解决？？**
19. **Cartographer在跑的过程中是因为加了什么东西去解决这个问题的。**
20. **关于室内cartographer跑出的效果比Gmapping 好的原因：**
21. **滤波器的方法通常基于几点假设，首先是**
22. **假设噪声服从高斯分布，**
23. **其次假设马尔科夫性，即假设当前状态只跟上一时刻的状态有关，而与其他时刻的状态无关，这就导致滤波器方法没有充分利用所有信息，很难生成全局一致性优良的地图；**
24. **然后考虑到实践中代码需要存储所有状态量的均值和协方差矩阵，空间复杂度为 O(n2)，如果将特征点也作为状态变量（一张图片中的特征点通常成百上千），那么随着地图规模的扩大，需要消耗的存储空间会非常大；**
25. **扩展卡尔曼滤波存在非线性误差，如果运动模型和观测模型的非线性程度很大，那么其非线性误差将会增大。**

## 备注

* **红色备注的部分是还没有弄明白需要实现的内容，在实现过程中可能会出现什么问题，一定要记录下来，改过什么参数一定要记录下来！！！**
* **黄色的部分是重点需要看的部分**
* **绿色部分是最后你需要变成小抄的部分。**
* **蓝色部分是自己预估可能会被问到的部分**
* **一定要让吴师兄做一次模拟面试！！！**

## 驭势科技面试题目：

## 可能会问到的问题： Q&A

1. 仅应用VLP16线激光雷达进行三维点云地图构建。VLP水平放置非角度（不会标定）
2. OctoMAP地图的生成
3. 使用OXTS设备进行GNSS-INS组合导航，具体型号是哪一款？具体的厘米及精度是多少？
4. 建图的后端优化包括哪些？
5. 相机模型的原理是什么？
6. 最小二乘优化算法
7. **基于rikirobot两轮驱动机器人定位建图，硬件包括哪些部分**
8. **通讯和调试主要分为那些**
9. **怎么调试的**
10. **GMapping算法中二维栅格地图的构建、Gmapping算法的原理**
11. **粒子滤波算法的原理**
12. **AMCL蒙特卡洛算法的原理**
13. **卡尔曼滤波算法的原理**
14. **Cartographer原理**
15. **图优化的原理**
16. **Ceres的使用**

# 项目介绍

## 基于AVP场景的无人车定位建图的算法研究（参考陈贵宾的论文）

### 项目介绍

应用课题组改装的线控电动车，在GPS信号缺失条件下仅应用IMU及Velodyne VLP16线激光雷达进行三维点云地图构建，并将生成的点云地图处理得到OctoMap，生成可以用来导航的地图。

主要工作（已经完成的部分）：

1. 应用LOAM算法进行三维激光SLAM建图，并针对三维SLAM建图方法的回环问题进行后端优化。

### 算法细节

在有GPS信号，或者信号强的时候用RTK，弱的时候用激光雷达Carla软件的使用

老师的项目：崔祥坡、孙浩、靡沛文

神经网络学习-》TPU 谷歌：TensorFlow

#### 电动车改装

VLP16 /PUCK

加装激光雷达，传感器上有信号线的一端冲车窗的位置，激光雷达上连有电源连接盒，盒子上有电源线接口和网线接口和GPS口，连接盒安装在副驾驶的工具箱上，用3M胶固定。电源线缠在副驾驶遮光板上。激光雷达电源连接盒内有绿色的LED灯常亮。没有工控机，我们的工控机的容量好像是4个G，不好用，直接用的笔记本电脑。（如果工控机启动有小圆点，如果黑屏，可以使用CRTL+alt+delete重启）

一般会有launch文件进行加载，找到Map的 launch所在的文件夹

地图导入结束

坐标系有base\_link坐标系（存在的坐标变换）

* **系统使用的是ubuntu16.04 \*64**
* **Ros版本：kinetic**

**官网https://www.velodynelidar.com/vlp-16.html**

横向视角360”，纵向视角30°

编译过程：

|  |
| --- |
| **🖝安装ROS依赖：**  **sudo apt-get install ros-kinetic-velodyne**  **🖝创建ROS工程**  **mkdir -p catkin\_velodyne/src**  **cd catkin\_velodyne/src**  **git clone https://github.com/ros-drivers/velodyne.git**  **cd ..**  **rosdep install --from-paths src --ignore-src --rosdistro kinetic -y**  **catkin\_make**  **source devel/setup.bash** |
| **连接好网线之后查看192.168.1.201** |
|  |

把velodyne XML文件转成ROS节点的YAML文件。（VLP-16.xml文件是激光雷达附赠的U盘中提供，转换后生成VLP-16.yaml）

|  |
| --- |
| **查看节点运行情况**  **rosnode list**  **订阅points主题，显示点的坐标**  **rostopic echo /velodyne\_pionts** |
| **rosrun velodyne\_pointcloud gen\_calibration.py ~/Desktop/VLP-16.xml**  **roslaunch velodyne\_pointcloud VLP16\_points.launch calibration:=/home/phd/VLP-16.yaml**  **🖝roslaunch velodyne\_pointcloud VLP16\_points.launch calibration:=/home/riki/catkin\_velodyne/VLP-16.yaml**  **🖝实时显示点云：**  **rosrun rviz rviz -f velodyne**  **🖝在rviz中点add 增加PointCloud2 再在PointCloud2下点topic输入/velodyne\_points。这样就可以实时显示获取的3D点云图。** |

**运行**

|  |
| --- |
| **roslaunch velodyne\_pointcloud VLP16\_points.launch 启动**  **rosrun rviz rviz 实时显示点云**  **add pointcloud2 topic 选velodyne\_points**  **右侧最上面Global Options列表下的Fixedfram 输入velodyne**  **rosbag record -o out /velodyne\_points 录像 ctrl+c结束 生成out\_[time].bag**只保存/velodyne\_points这个topic的数据（可以用rostopic list -v开看当前可用的topic），保存在当前目录的out.bag。  **rosbag play ~/out\_[time].bag 显示某time的录像** |
| **补充：** |
| **[VeloView](http://www.paraview.org/Wiki/VeloView)也是可以实时显示3D LiDAR激光雷达点云图的，保存格式是pcap。** |
| **rostopic list -v开看当前可用的topic** |

#### RTK设备的配置与使用

使用OXTS设备进行厘米级的定位。RTK进行安装需要两个接收器之间相隔至少1m左右。，同时需要安装基站，基站安装的完成后进行平均，最高精度可达到2cm左右 最低精度可达到20cm左右。

#### LOAM 的配置

**1.ros环境搭建**

|  |
| --- |
| **mkdir -p ~/catkin\_ws/src**  **cd ~/catkin\_ws/**  **catkin\_make**  **source devel/setup.bash** |

**2.下载loam编译**

|  |
| --- |
| **cd ~/catkin\_ws/src**  **git clone https://github.com/laboshinl/loam\_velodyne.git**  **cd ..**  **catkin\_make -DCMAKE\_BUILD\_TYPE=Release**  **source ~/catkin\_ws/devel/setup.bash** |

**3.运行Loam**

|  |
| --- |
| **🖝roslaunch loam\_velodyne loam\_velodyne.launch**  **rosbag play 文件名.bag**  **或者可以这样**  **🖝rosbag record -o Trista -a /laser\_surround** |

**4.保存地图**

|  |
| --- |
| **选择右侧最上面Global Options列表下的Fixed Frame选择Laser \_odom**  **选择pointClouds列表下的Topic选择laser\_cloud\_surround**  **打开另一个窗口 rosbag record -o out /laser\_cloud\_surround 录制**  **ctrl+C结束录制生成out\_（时间）.bag**  **转PCD**  **🖝rosrun pcl\_ros bag\_to\_pcd out\_（时间）.bag /laser\_cloud\_surround pcd** |

1. **查看**

|  |
| --- |
| **🖝cd pcd**  **🖝pcl\_viewer （文件名）如：pcd/1566443218.880078077.pcd** |

1. **将PCD文件转换成二进制文件.ot 或者.bt**

|  |
| --- |
| **🖝cd ~/octomap-devel/octomap\_tutor-master/bin**  **🖝 pcd2octomap data /sample.pcd data /sample.bt 文件目录（或者.ot文件）**  **🖝riki@ubuntu:~/octomap-devel/bin$octovis/home/riki/octomap-devel/octomap/share/data/geb079.bt**  **🖝bin/pcd2colorOctomap data/sample.pcd data/sample.ot** |

**其中会出现坐标系与建出的图相互垂直的问题，解决方法 修改Plane的参考平面就可以了**

|  |
| --- |
| **WARNING: disk usage in log directory [/home/.../.ros/log] is over 1GB. 问题解决办法**  **解决方法：**  **这个警告是由于主文件夹下/.ros中的日志文件夹/log超过了１G而产生的提醒，这些log并没有什么实际用处，清空只需执行：**  **🖝rosclean purge**  **然后在两个y/n里都输入y即可消除这一警告。当然，手动去目录当中删除同样有效。** |

**常见的问题：**

|  |
| --- |
| **\*\*\* Error in `octovis': realloc(): invalid pointer: 0x00007faca21a1820 \*\*\*** |

**解决方法：**

|  |
| --- |
| **重新安装一遍octomap:**  先重装了一下 依赖库  **🖝sudo apt-get install libqglviewer-dev-qt4**  **重新编译一下：**  **一定要在安装目录下进行重新编译**  **cmake..**  **make**  **sudo make install**  **重新运行一下指令就可以了**  **octovis octomap.bt** |

注意：

|  |
| --- |
| 网上说装libqglviewer-dev-qt4后删掉的libqglviewer-dev会对g2o库有影响,暂时没发现什么问题,先记录一下.  在编译安装OctoMap后，可以再重新安装回libqglviewer-dev，以便为其他功能模块，如g2o提供依赖项。  sudo apt-get install libqglviewer-dev  因为libqglviewer-dev-qt4和libqglviewer-dev只能存在一个，但libqglviewer-dev-qt4并不能支持g2o。安装回libqglviewer-dev是不会影响octovis的使用的，因为它影响的只是OctoMap的编译安装。 |

#### LOAM 论文的理解

论文是使用一个三维空间中运动的**两轴激光雷达**来构建**实时激光里程计。**

文章提出了构建实时里程计的难点在于**点云不是相同时间获得**的。

因为点云中的点随着激光雷达运动会产生**运动畸变**，会造成点云在匹配时发生错误，从而不能正确获得两帧点云的**相对位置关系**也就无法获得正确的里程计信息。

核心思想是将定位和建图的分割，通过两个算法：**一个是高频率但是低精度的的里程计运动估计（定位），另一个算法在一个低的频率执行匹配和注册点云信息（建图和校正里程计）**。将这两个算法结合就获得了高精度、实时性的激光里程计。

首先GPS信息是提供了位置，惯导提供了什么信息？里程计包括什么？

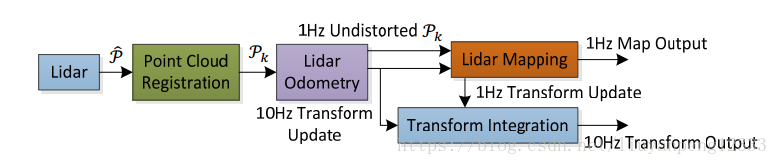
常见的scan-to-scan、map-to-map、scan-to-map匹配的方法

##### LOAM的优点就是很好地利用各个匹配的优缺点，

1. 首先，虽然scan-to-scan匹配精度差，但是我们可以只是使用它做一个获取粗的里程计，计算量较小因此我们可以高频执行，去除匀速运动造成的运动畸变。
2. 因为map-to-map匹配存在计算量大的问题，执行的频率低的地图匹配。这样的高低频率结合就保证了计算量的同时又兼具了精度。

当使用多传感器时可以使用卡尔曼进行滤波。（**卡尔曼滤波**）这样就可以获得实时的地图用于机器人导航时的路径规划和避障。

**激光的分辨率为0.25度**，频率为40HZ。



激光里程计主要分四部分完成。首先是获得激光雷达坐标系下的点云数据P^，然后把第k次扫描获得的点云组成一帧数据Pk。然后将Pk在两个算法中进行处理，也就是上面Liar Odometry节点和Lidar Mapping节点。Liar Odometry节点的作用是获取两帧连续点云数据间的运动，估计出来的运动用于去除Pk中的运动畸变。这个节点执行的频率为10Hz，作用相当于scan-to-scan匹配获得粗糙的运动估计用于去除匀速运动造成的运动畸变，并将处理后的结果给了Lidar Mapping节点做进一步处理。Lidar Mapping节点使用地图去匹配和注册没有畸变的点云数据以1Hz的频率。最后由Transform integration节点接收前面两个节点输出的Transform信息并将其进行融合处理以活动频率为10Hz的Transform信息即里程计。

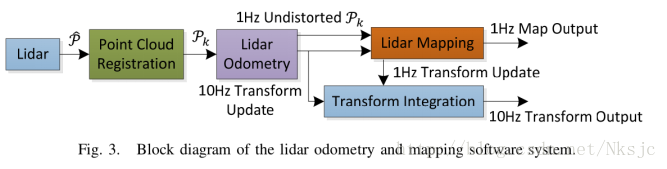
选取点时注意的部分：

**已经选过点周围的点**或者是**在于激光线接近平行的平面上的点**，时我们也**避免可能遮挡点**，**激光雷达的运动是匀速的**

#### Loam\_velodyne的代码解析

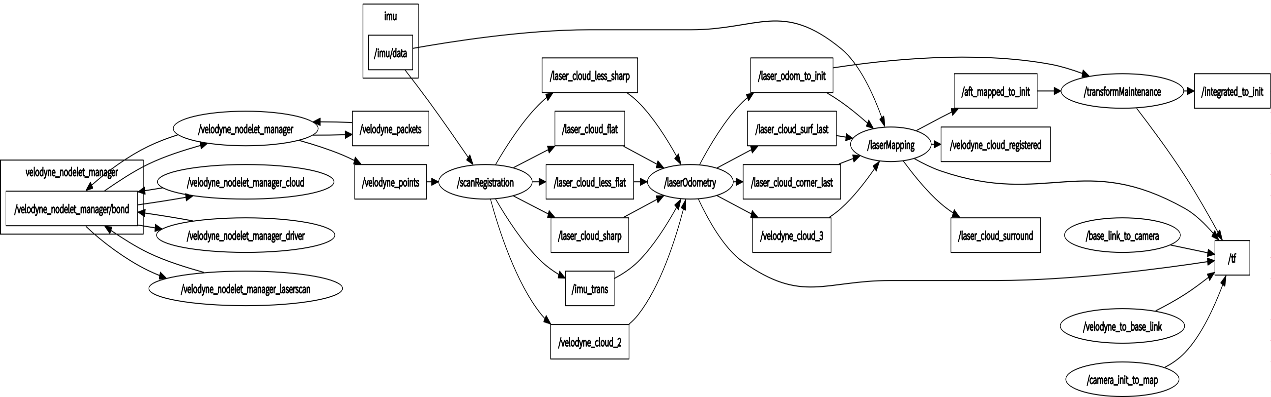
在github上下载的**laboshinl/loam\_velodyne** 820个星标的那个<https://github.com/laboshinl/loam_velodyne>

LOAM源码主要由四个节点构成，主要包括**ScanRegistration特征点提取、LaserOdometry高频低精度odom(10Hz)（从特征点中估计运动发送能够给lasermapping）、LaserMapping低频高精度odom(1Hz)、transformMaintenance(双频odom融合)**，每个节点以**rosnode**的形式存在， 也就是说是独立的进程，进程间通过rostopic传递点云， odom等数据。



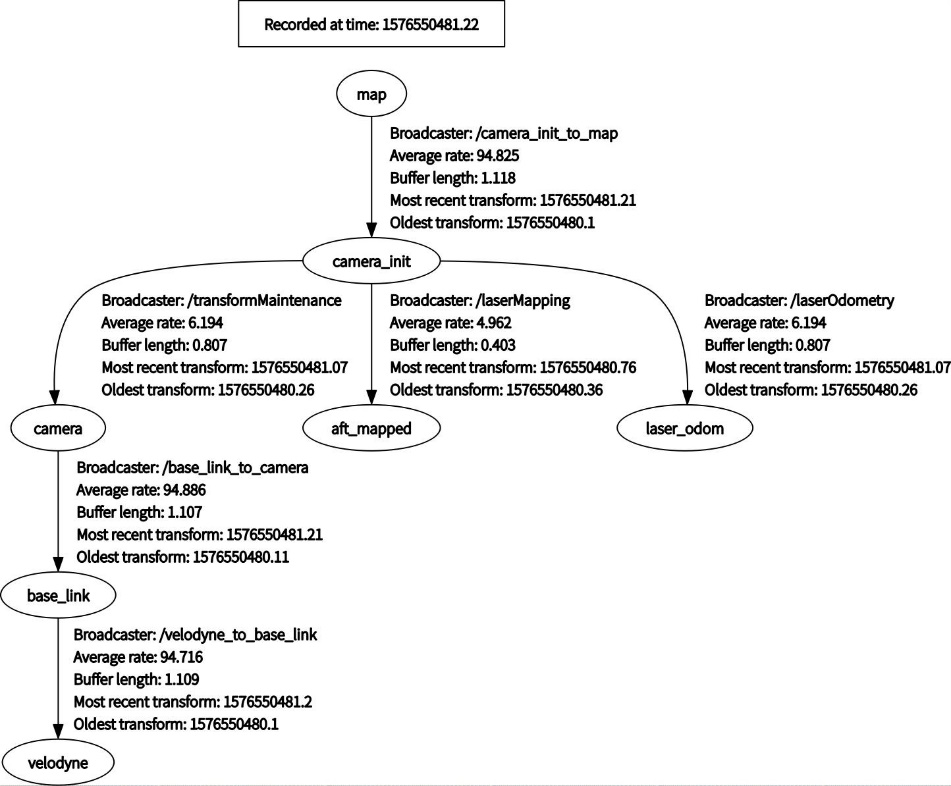
具体的rqt\_graph(命令：rosrun rqt\_)

|  |
| --- |
| **riki@ubuntu:~$ rosrun rqt\_graph rqt\_graph** |



Tf-tree的结构：

|  |
| --- |
| **riki@ubuntu:~$ rosrun rqt\_tf\_tree rqt\_tf\_tree** |



话题链接

**LOAM的原理：提取特征点匹配后计算坐标变换。**

**为什么用的多线激光雷达,但是LOAM是针对单个Scan提取特征点的？**

**原因**：这里主要考虑到**线束间角分辨率**(竖直分辨率)与**单个线内点间角分辨率**(水平分辨率)存在的差异.VLP16的竖直分辨率为2°，水平分辨率为0.4°。

#### vaserKvaser读取CAN总线上的信号，包括CAN总线上的信号包括哪些

#### IMU的测量值包括哪些，车辆级别的IMU的测量的优点和缺点RTK

1 由**陀螺仪**计算出**惯性组件的姿态**（即求出四元数或者旋转矩阵）；

2 根据惯性组件的姿态将**加速度**测得的**载体坐标系**的比力分量转换到**导航坐标系**中（坐标系转换）；

3 在导航坐标系下，通过**积分求解**比力方程，获得载体相对地球的速度（**去除地球引力加速度**）；

4 在位置参考坐标系下，由**载体速度积分**得到**位置**。

#### LOAM建图的优点和缺点，针对缺点进行回环处理，重影的问题，参考陈贵宾的毕业论文

**优点**：LOAM 是目前公认的比较优秀的激光里程计，在小规模建图时表现良好，但是因为要保证实时在线建图，没有添加回环检测模块，也就**没有使用图优化技术**对构建的地图进行统一优化，在某些场景建图时，会因为提取的特征点不够多而最终导致明显的建图“重影”问题。

**缺点**：存在里程计固有的“累计误差”

**方法优化**：算法采用图优化 SLAM 中的 Pose-Graph （位姿图）作为系统框架，采用 G2O 作为优化函数库，将 LOAM 计算得到的初始车辆位姿设置为位姿图的顶点，使用 ICP 点云配准算法结合顶点之间的欧式距离为位姿图建立局部回环边和全局回环边的约束。位姿图构建完毕后，通过 G2O 内置的优化求解器来计算尽量满足所有约束的最优车辆位姿，进而消除 LOAM 直接建图的“重影”和“累计误差”问题，提高建图精度。

**位姿图本质上是一个优化问题，顶点是优化变量，算法的优化求解目标是尽量满足所有约束的所有车辆位姿。边是不同车辆位姿之间的转换关系，本质是残差函数。本质是对残差方程构成的超定方程组进行求解，就变成了非线性最小二乘问题。**

**目标为一组残差方程的残差和最小为优化目标，求解最优的车辆位姿，将激光点云从局部坐标系（雷达坐标系）变换到全局坐标系（以第一帧位子为原点的坐标系）。**

将位姿图顶点和边的类型都设置为具有六自由度的 SE3 类型

进行坐标变换：

激光雷达坐标—》全局坐标

前一秒叫做目标点云、当前秒的点云叫做源点云。配准结束后，ICP算法输出源点云

#### LOAM用到的优化方法包括哪些 我想进行哪些改进

https://blog.csdn.net/orange\_littlegirl/article/details/89067017

#### 激光雷达、*毫米波*、摄像头的测距原理

##### 激光雷达信号处理算法

###### SLAM是什么？--状态估计的问题

定位： 在给定地图的情况下，估计机器人的位姿

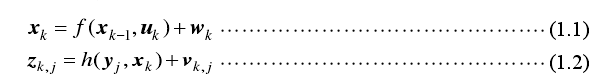
建图: 给定机器人位姿的情况下估计环境地图

SLAM： **同时**估计机器人的位姿和环境地图

IEEE上首次提出SLAM问题

**SLAM解决的问题主要包括**：

1. 机器人在环境中的**位姿**
2. 导航过程中需要的**环境地图**

**SLAM 问题本质上是一个状态估计问题，即通过系统控制输入 uk 以及观测数据 zk,j 来估计车辆的运动轨迹（车辆位姿）xk 以及地图中路标（特征点）的位置 yj，**

###### SLAM地图的分类？

对于静态环境的地图的分类主要分为尺度地图（建图方法主要包括基于滤波器的方法、基于图优化的方法）、拓扑地图、混合地图。

###### SLAM的框架

求解方法包括**滤波器**和**图优化**两种方法。

###### 图优化

图优化是什么？

* 目前激光SLAM中比较好用的是**图优化**（Graph-based）的方法。
* **图优化（又称为全SLAM\Full-SLAM）**是离线的方式记录下**所有时刻**的**位姿**和**特征点**，根据所有的信息来优化并估计计算出最优的车辆位姿和特征点位置。利用稀疏性
* **图优化中的图**指的是图论中的图的定义，给定顶点和连接不同顶点之间的连线所构成的图形。
* **安装好G2O函数库，只需要设置好顶点和边就能得到全局最优的车辆位姿及特征点位置。节点**表示**车辆的位姿（摄像头或者激光雷达的位姿以及特征点的位姿）**；**边**表示**位姿之间的空间约束关系（一般是里程计或者匹配算出来的转换矩阵）**。**目标函数**是误差函数（总的误差），对误差函数进行**优化**。因为机器人在行使的过程中，里程计进行位姿估计，但是里程计有累计误差，会导致出现不精确（例如在跑一个回环的时候闭合不好，后面就会发散，图就会坏掉）。（图优化部分可以调整全局一致性）。

图优化的总体框架



* **基于图优化的方法整体包括前端和后端。**
* **前端：**主要是对传感器的输入数据进行**数据关联**，包括**里程计（局部数据的关联）**和**闭环检测（全局数据的关联）（主要作用是消除里程计的累积误差，协调统一全局的作用）**；
* **后端：**主要是对前端构建的图进行**非线性优化**，调整前端构建的图中所有的顶点的位置，尽量满足**所有边的约束**，得到所有边约束下的**最优顶点位置**。

里程计

里程计分为**机械式里程计**和**基于匹配算法的里程计**。

机械式里程计一般通过安装在两个车后轮的编码器和安装在车体上的惯导设备（IMU）来获取车辆在不同采样时刻之间的位姿变换，但是精度低且易受车轮滑移率的影响。（二维场景，不直接将里程计的输出结果作为顶点之间的约束，而是将其与视觉或者激光里程计融合使用。）

###### 滤波器

* **基于滤波器**的方法主要包括状态预测（里程计）、测量预测、测量、数据关联、状态更新和地图更新。（典型的是卡尔曼估计、扩展卡尔曼滤波器、粒子滤波器、信息滤波器等进行求解）（这里已知前验概率和后验概率的部分）。在比较早的的SLAM中的占据主导地位。

**缺点：基于几点假设：**

* 噪声服从**高斯噪声**分布
* 假设**马尔科夫性**，即假设当前状态只跟上一时刻的状态有关，而与其他时刻的状态无关，这就导致滤波器方法不能充分利用所有信息，很难生成**全局一致性**优良的地图。
* 代码需要存储所有**状态量的均值和协方差矩阵**，空间复杂度为 O(n2)，如果将特征点也作为状态变量（一张图片中的特征点通常成百上千），那么随着地图规模的扩大，需要消耗的**存储空间会非常大。**
* 扩展卡尔曼滤波存在**非线性误差**，如果运动模型和观测模型的非线性程度很大，那么其**非线性误差将会增大**。

卡尔曼滤波器

扩展卡尔曼滤波器

**粒子滤波器**：

FastSLAM、FastSLAM2.0在室内、室外小规模场景下都有不错的效果

Gmapping可以实时构建室内地图，在构建小场景地图所需的计算量较小且精度较高。随着场景增大所需的粒子增加，因为每个粒子都携带一幅地图，因此在构建大地图时所需内存和计算量都会增加。因此不适合构建大场景地图。并且没有回环检测，因此在回环闭合时可能会造成地图错位，

粒子退化

**粒子退化**主要指**正确的粒子被丢弃**和**粒子多样性减小，而频繁重采样则加剧了正确的粒子被丢弃的可能性**和**粒子多样性减小速率**。

**重采样**

**重采样的知识，我们知道在执行重采样之前会计算每个粒子数的权重，有时会因为环境相似度高或是由于测量噪声的影响会使接近正确状态的粒子数权重较小而错误状态的粒子的权重反而会大。重采样是依据粒子权重来重新采粒子的，这样正确的粒子就很有可能会被丢弃，频繁的重采样更加剧了正确但权重较小粒子被丢弃的可能性。这也就是粒子退化原因之一。**

**频繁重采样导致的粒子多样性减小的速率加大**，什么是**粒子多样性**呢？就是粒子的不同，就像最开始有十个粒子，如果发生重采样后其中有五个粒子被丢弃，剩下的五个粒子复制出五个粒子，这时十个粒子中只有五个粒子是不同的也就是粒子多样性减小。为了防止粒子退化就要减少重采样的次数。

为了**减小粒子数**Gmapping提出了**改进提议分布**，为了**减少重采样的次数**Gmapping提出了**选择性重采样**。

如何改进提议分布？**里程计采样的高斯分布和激光进行观测后获得状态的高斯分布**利用最近的一次观测来模拟目标分布。RBpf使用粒子滤波来估计机器人位姿，而粒子滤波中最常用的是**重要性重采样算法**。这个算法通过不断迭代来估计每一时刻机器人的位姿。算法总共包括四个步骤：**采样- 计算权重-重采样-地图估计**。

信息滤波器

###### 区别

* 基于滤波器的SLAM与基于图优化的SLAM方法的**区别**在于，基于滤波器的SLAM方法只顾及x\_t当前时刻的位置，而不管之前的位置。就会导致前面的位置一旦产生误差，误差就几乎不可能会修复。例如Gmapping的方法就是这样。

###### 激光雷达数学模型介绍

**激光雷达的数学模型主要分为两种：**

* **光束模型（波束模型）**
* **似然场模型**
* **基于特征的测量模型**
* **基于相关性的测量模型**

光束模型

其中**光束模型（beam model）**比较早，可能会受到**初值的干扰**；最后的观测值是四种概率密度的加权平均混合就可得到；激光的观测值有四种可能：主要包括

1. **小的测量噪声p\_hit**--（**高斯分布**，主要受传感器的有限分辨率(最大传感器测矩距离)、大气对测量的影响（主要考虑传感器受什么因素的干扰就是传感器的缺点））
2. **意外对象引起的误差p\_short**、（**指数分布**；动态环境例如人等，->处理方法：作为状态向量的一部分进行估计他们的位置或者直接作为传感器的噪声处理）
3. **由于未检测的对象引起的误差p\_max**（使用很窄的**均匀分布**假装|激光测距仪遇到黑色吸光对象或者强光下也会出现检测失效|镜面反射）
4. **随机意外噪声 p\_rand**（**均匀分布** 多次反射等其他噪声）

**光束模型的缺点：**

* **期望值的计算需要用到射线投射（raytracing\ray casting）来进行测量，一帧16束激光束,就需要进行16次raytracing。**
* **在非结构化的环境中，位姿改变就会导致期望值发生巨大的变化。**

光束模型(沿着一个波束测量距离—主要用于激光类的的优先选型)

超声波模型是在一个测量锥内测量距离（超声波的优先模型）

固有模型参数：

依赖数据：从实际数据中得到这些参数（似然最大化的方法—极大似然估计算法）

似然场模型（likelihood model）

**主要思想是将传感器扫描的终点映射到地图的全局坐标空间中，优点：**

* 对图像进行高斯平滑，在任何环境中期望值对于位姿都是平滑的；（大致的意思就是机器人如果有微小的变化对于测量的结果影响较小）
* 得分计算不需要经过ray tracing ，直接通过查表可得到，计算量低。（但是仅针对二维空间进行，增加了紧凑性）
* 适合结构化环境和非结构环境中。

**缺点：**

* 不能对引起短读数的其他动态物体进行建模
* 不能确定一个点的路径是否被地图上的一个障碍物所拦截
* 没有考虑地图的不确定性（unknown|高度不确定）

**观测值的噪声和不确定性**

1. **测量噪声p\_hit**
2. **测量失败p\_max**
3. **无法解释的随机测量p\_rand**

基于特征的测量模型

特征提取

主要提取直线、边、角、局部极小、分别与墙、墙角、树干等物体对应

地标的测量—仅适应于基于特征的地图

传感器测量地表相对于机器人局部坐标系的距离和方位

###### 运动畸变的去除

运动畸变产生的原因：

1. 激光点数据不是瞬间获得的
2. 激光测量时伴随着机器人的运动
3. 激光帧率较低时，机器人的运动不能忽略

**VLP16 10HZ -100ms**

**用于畸变去除的方法有：**

* **纯估计方法**
* **里程计辅助方法**
* **融合的方法**

**纯估计的方法**：类ICP的方法，作用是配准点云。核心是求解R和t,是的误差函数最小（非线性最小二乘法）

**已知对应点的求解方法：**

将对应点求解点云的中心部分，然后对两帧点云进行去中心化，求解R和t

**未知对应点的求解方法：（EM算法是什么意思）**

1. 寻找对应点（对应点是欧氏距离最近的点）
2. 根据此对应点计算R,T
3. 对点云进行转换，计算误差
4. 不断迭代，直至误差小于某一个值

ICP的缺点：

* 没有考虑激光的运动畸变
* 当前的激光数据是错误的

**VICP：**

* 考虑了机器人的运动（但是认为是匀速运动）
* 进行匹配的同时估计机器人的速度
* 两帧之间的位置处进行线性查知（LOAM中用到了）

缺点：

对于低帧率的激光（5HZ，匀速运动假设不成立）

数据预处理和状态估计过程耦合

**里程计辅助的方法**

* 较高的位子更新频率（200HZ.）可以比较准确的反应运动情况、
* 有较高的局部位姿估计，
* 跟状态估计完全解耦

**惯性测量单元（IMU）：**

直接测量**角速度**和**线加速度**

具有较高的**角速度测量精度**

**测量频率极高**（1kHZ-8kHZ）

**线加速度精度太差**，二次积分在局部的精度依旧很差

**轮式里程计：**

直接测量机器人的**位移和角度**

具有较高的**局部角度测量精度**

具有较高的**局部位置测量精度**

更新测量速度较高（100HZ-200HZ）

单片机用的是usb3.0吗

在处理器上处理，利用CPU读取激光雷达数据，同时在单片机上传里程计积分数据，两者进行时间同步。在CPU上统一进行运动畸变去除。

体系清晰，不会产生延时，但是会进行时间同步、需要进行位姿插值。

轮式里程计

1. 求解当前帧激光数据中每一个激光点对应的机器人位姿，求解起始时刻之间这一段时间的的机器人位姿
2. 根据求解的位姿把所有激光点转换到同一坐标系下
3. 重新封装成一帧数据发布出去

ROS中自带的机制可以用于同步

（进行线性差值）

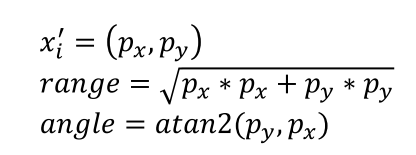
或者可以进行二次差值：

在一帧激光数据之间，认为机器人是匀加速运动（这个假设更加合理，可信度更高一些）

机器人的位姿就是关于时间的二次函数

也可以进行二次曲线的拟合

用分段线性函数对二次曲线进行近似得到位姿，进行坐标变换，然后转换后的坐标转换为激光数据发不出去



融合的方法去除运动畸变

**（这里注意位置误差的线性假设比位置的线性假设更加合理）**

1. 用**里程计方法进行校正**，去除大部分的运动畸变
2. 假设**误差值线性分布**
3. 用ICP 的方法进行匹配，**匹配结果作为正确值**，得到里程计的误差值
4. 把误差值均摊到每一个点上，重新进行激光点位置修正
5. 再进行ICP 迭代，直到收敛

###### 激光的前端配准（前端匹配、帧间匹配）算法

**常见的帧间匹配方法：（自上而下精度逐渐增加）**

1. **ICP匹配方法（原始ICP方法）**
2. **PL-ICP匹配方法**
3. **NICP匹配方法**
4. **IMLS-ICP匹配方法（隐式最小二乘方法）**

ICP匹配方法

**点匹配：欧氏距离配准两个点云**

**已知对应点的求解方法：**

将对应点求解点云的中心部分，然后对两帧点云进行去中心化，求解R和t

**未知对应点的求解方法：（EM算法是什么意思）**

1. 寻找对应点（对应点是欧氏距离最近的点）
2. 根据此对应点计算R,T
3. 对点云进行转换，计算误差
4. 不断迭代，直至误差小于某一个值

PL-ICP匹配方法

**激光点是对实际环境中曲面的离散采样**

**重要的不是激光点，而是隐藏在激光点中的曲面**

**误差尺度位当前激光点到实际曲面的距离，关键的问题在于如何恢复曲面**

**PL-ICP的方法是分段线性的方法来进行实际曲面进行近似，从而定义当前帧激光点到曲面的距离。（点到直线的距离）**

**已知条件：**

**参考帧**

**当前帧**

**参考激光真生成的曲面**

**初始解**

**代求数据：**

**两帧激光之间的相对位姿关系q\***

**算法流程：**

1. **将当前帧的数据根据初始位姿投影到参考帧坐标系下**
2. **对于当前帧的点，在参考帧中找到最近点两个点（确定一条直线）**
3. **计算直线误差，去除误差过大的点**
4. **最小化误差函数**

与ICP的区别

**误差函数形式不同：**

1. ICP对点对点的距离作为误差，PL-ICP为点到线距离作为误差；PL-ICP的误差形式更符合实际情况
2. 收敛速度不同，ICP是一阶收敛，PL-ICP是二阶收敛
3. PL-ICP的求解精度高于ICP,特别实在结构化环境（人造环境、直线特征边角特征明显的环境）中
4. PL-ICP对初始值更敏感（该方法一般和里程计结合在一起，CSM+PL-ICP）

NICP匹配方法

**NICP基本思想：**

1. 替换原始ICP的方法中对应的点匹配方法
2. 充分利用实际曲面的特征来对错误的点匹配进行滤除，主要特征为法向量（最小特征值对应的特征向量）和曲率。
3. 误差项处理考虑对应点的欧氏距离之外，同时还考虑对应点的法向量的角度差
4. 点匹配的原则：

* 两点之间的距离大于阈值，拒绝
* 两点之间的曲率差距大于阈值，拒绝
* 法向量角度之差大于阈值，拒绝

**算法流程：**

1. 计算激光参考帧与当前帧中的每一个点的法向量和曲率
2. 根据当前解，把当前激光帧的点转换到参考坐标系中，并且根据欧氏距离、法向量、曲率等信息来选择匹配点（也可能没有匹配点）
3. 用LM的方法进行迭代求解，迭代收敛即可得到两帧激光数据之间的相对位姿。

IMLS-ICP匹配方法（implicit moving least square）-ICP匹配

**基本思想：**

1. **选择具有代表性的激光束进行匹配，减少计算量，减少激光点分布不均匀导致的计算结果出现偏移**
2. **点云中隐藏着真实的曲面，从参考帧点云将曲面重建出来（最小二乘的方法）**
3. **曲面重建的越准确，对真实世界描述越准确，匹配精度越高**

###### 帧间匹配算法

**帧间匹配算法主要包括三类：**

* **高斯牛顿优化方法**
* **NDT方法**
* **相关匹配方法及分支定界加速**

**高斯牛顿优化法（又称最快速度下降法）**

非线性优化方法整理

**非线性优化方法（参考视觉SLAM14讲）：**

* **牛顿法**
* **高斯牛顿迭代法**
* **LM法**

**状态估计问题：**

* **最大后验与最大似然**

**概念：**

**最大后验MAP：**最大后验概率等于最大似然\*最大先验

**最大似然估计：**在什么样的状态下，最可能产生现在观测到的数据。

**最大后验等于最大似然\*先验**

* **最小二乘**
* **非线性最小二乘**

**相机与图像**

**相机成像的原理：**

**三维空间中的物体反射光线或者发出光线，穿过相机光心后投影在相机的成像平面上。相机的感光器件接收到光线后，产生测量值，得到像素，形成照片。**

###### 激光SLAM（KITTI排名目前是什么）

**传感器输入**：惯性测量单元IMU、轮式里程计（Wheel Odometry）、雷达Lidar

**生成的地图类型**：覆盖栅格的地图、点云地图

**输出：**覆盖栅格地图、机器人的轨迹（pose graph）

###### 帧间匹配算法：2D:

1. **PP-ICP(迭代最近点算法)、**
2. **PI-ICP（点到线迭代最近点算法）、3Dpoint-to-plane ICP、PCL库里的GICP（plane-to-plane ICP）NICP(不但要考虑点的距离还要考虑法向量与点的曲率) IMLS-ICP（隐式最小二乘ICP implicit moving least-square ICP）基本思想是：点云数据对应的是表面，理论厚度是0，先拟合一个曲面，在和点云进行匹配。其中的隐式是没有曲面表达式，通过点来获得。**
3. **NDT（三维纯定位比较多）、**
4. **CSM（correlation Scan Match）算法（暴力求解的方法，可以通过加速策略进行降低计算量）优点：对初值不敏感、精度受限于分辨率**
5. **梯度优化方法：Hector-SLAM——对初始值敏感、非线性最小二乘优化方法主要包括LM（裂纹伯格优化方法）、高斯牛顿迭代方法、梯度下降方法**
6. **2D:State of Art :CSM+梯度优化的方法（cartograper使用的就是这种方法）**
7. **Feature-based method 特征点描述方法**

###### 回环检测

1. **Scan-to-scan（容易匹配错）(3D)**
2. **Scan-to-map（cartographer 子图匹配上容易回环）（cartographer在几何对称的环境下比较容易出现回环错误）**
3. **Map-to-map（将最近几帧的激光聚合成一个子图，与过去的子图进行匹配）**
4. **Branch and bound &lazy decision(分支定界的方法|延迟决定)构造一个树（搜索空间），进行分支，叶子节点是最细栅格；定义一个代价函数，父节点的代价是子节点的上界，假设是最优，子树全部截掉，减少计算量。**

**关于滤波器的方法：**

* **EKF-SLAM ->生成的是feature-based Map 不能用于后面导航**
* **FastSLAM 02-03**
* **Gmapping (计算量非常大)|Gride-fastSLAM+scan-matching模块 这个方法非常依赖里程计（将所有的粒子都一视同仁，每个粒子都携带一个地图，存储量非常大）**
* **Optimal RBPF 10年**
* **MPRT粒子传播几次得到n个粒子，选一个最优的粒子作为真实的粒子（相当于给每个粒子几次机会）**

**关于图优化的方法**

**自从意识到激光点云的稀疏性的时候主流就开始了图优化的方法**

* **Karto SLAM**
* **Cartograper 局部匹配scan-matching\全局矫正、回环检测Map-to-map（缺点是预先构建子图浪费时间）（这个算法的原理图）**

###### 数据预处理

* **轮式里程计的标定//offline出厂标定online在线标定。负载过大，**
* **激光雷达的运动畸变去除 ||同一时刻取得，旋转需要实践100ms**
* **不同系统之间的时间同步 ||请求数据时间**

**实际环境中的问题：动态物体、环境的变化、几何结构相似的环境、全局定位、地面凹凸不平的问题、机器人载重变化（对车辆由什么影响）的问题等等**

**与视觉进行融合的问题主要包括：**

**解决集合环境相似的问题，利用信息量丰富的视觉地图进行全局定位。利用高精度的历程信息可以解决凹凸不平的地面问题。**

**其实是可以直接用点云地图进行导航的（高飞的论文、沈邵劼论文里有相关的内容）**

y、z都有**平移、旋转、缩放、倾斜**四种变换；**射影变换**有15个自由度，都有其次矩阵的**16个元素扣去一个全局尺度**就是他的自由度数。

**李群李代数**

什么样的相机位姿最符合当前观测数据的问题；构建成优化的问题，求解最优的R和t，使得误差最小化。

凤姐咬你脚（**封闭性、结合律、幺元、逆、交**）

###### LOAM

* 纯激光 SLAM 相当于里程计的作用，有匀速运动假设，没有回环。
* V-LOAM视觉激光融合 漂移匀速假设 -》无回环
* VELO-视觉激光融合，无运动畸变假设，有回环
* LEGO-LOAM 有回环

激光和视觉融合

* 3D激光雷达为视觉特征提供相对准确的深度信息
* 视觉辅助激光雷达进行畸变的去除
* 视觉辅助回环检测
* 视觉提供精确的里程信息

##### LOAM进行中激光里程计模块的总结：

1. **特征点检测（检测边缘点、平面点）；**
2. **寻找匹配点（对于每一个edge point找到对应匹配的线|对于每一个planar point 找到对应匹配的面）**
3. **构建非线性方程组（目标函数：误差函数 点到线的距离、点到面的距离）**
4. **求解非线性方程组（进行迭代）**
5. **如果到了当前帧的末尾，则投影到当前帧的时刻，开始进行下一帧的求解**

##### LOAM中激光建图模块：

1. **雷达里程计的输出与地图进行匹配：输出主要包括当前的位姿和去畸变之后的完整的一帧数据；（这个地方的匹配的特征点的数量更多；当前帧的特征点与去畸变的一帧的附近的立方体中所有相应的特征点）**
2. **直线匹配：对于edge point边角特征点，cubic中所有的边缘点会按照直线进行分布，根据特征点求解出直线的方程；**
3. **根据cubic中的边缘特征点，计算位姿的协方差矩阵**
4. **对协方差矩阵进行特征值分解**
5. **最大特征值对应的特征向量即为直线的方向向量；**
6. **该直线通过所有的Edge Point的几何中心**

**平面匹配：**

1. **对于平面特征点，cubic中的所有平面点会按照平面进行分布。根据这些特征点求解出平面的方程。**
2. **根据cubic中的平面特征点planar point ，计算位姿协方差矩阵。**
3. **对协方差矩阵进行特征值分解**
4. **最小特征值对应的特征向量，即为平面的法向量**
5. **该平面通过所有planar point的几何中心**

**由我来问的问题：**

**陈贵宾的毕业论文**

**我如果能够去贵公司实习，我主要做什么？主要怎么做的？**

**传感器算法实习的主要内容：**

* **研究摄像头、毫米波雷达、激光雷达的性能**
* **传感器融合CornerCase处理算法的开发**
* **传感器性能的测试及评估工作**
* **了解摄像头、毫米波雷达、激光雷达测量原理及相应的信号处理算法，具备实际项目研究经验**
* **Matlab/Simulink进行仿真开发**
* **三个月左右**

**自动化辅助工具开发实习生**

* **职责：**
* **自动化辅助工具的设计与开发**
* **负责根据需求文档设计并实现ADAS功能继承辅助工具，并验证**
* **负责设计文档的撰写**
* **精通Matlab编程语言，了解CAN协议和CAPL编程者优先**

**智能驾驶定位算法实习生**

* **熟练使用Matlab/Simulink进行仿真开发**
* **有环境感知定位算法有深入了解，具备实际项目研究经验**

**环境感知算法实习生**

* **基于雷达进行物体检测、识别、跟踪、场景分割、理解、高精度定位、立体视觉等领域具有研发经验优先**
* **掌握C++/c和脚本语言编程，熟悉大规模并行计算的基本原理并具有实现并行计算算法的基本能力**

**ROS软件实习生**

* **岗位职责：负责基于ROS的应用程序进行开发**
* **负责传感器数据可视化模块，毫米波雷达/激光雷达等**
* **熟悉C++,熟练掌握Ubuntu等常用指令**
* **由RVIZ应用、CAN/Ethernet等相关经验者优先**
* **机器学习：**
* **由机器学习、深度学习基础、了解TensorFlow等平台**

**驭势实习需要学习的资料和总结：**

**卡尔曼滤波（扩展卡尔曼滤波）可以用于定位的方法（扩展卡尔曼滤波定位和摄像头的处理也会用到这方面的内容）**

###### 激光SLAM中的问题

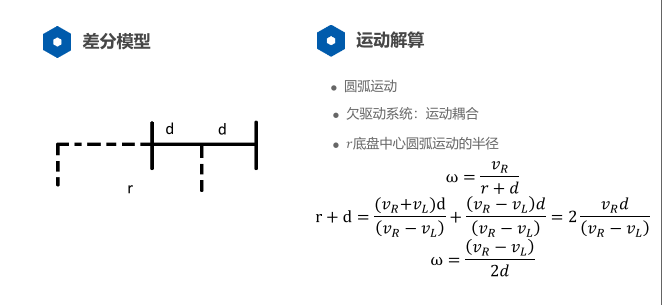
* 退化环境（走廊缩短2-3m）
* 地图的实时动态更新
* 全局定位
* 动态环境定位

**LCM**

**MHT**

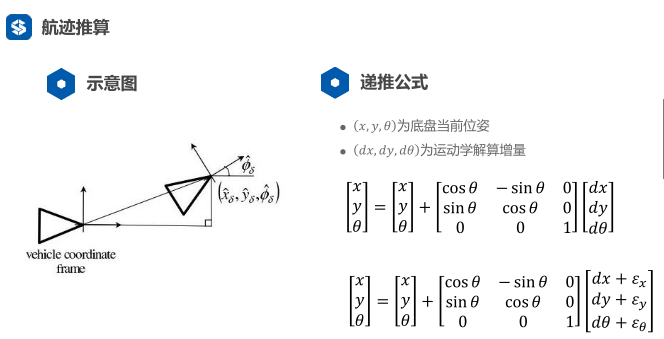
###### 传感器数据处理：里程计运动模型及标定

**两轮差分底盘的运动学模型：差分模型**



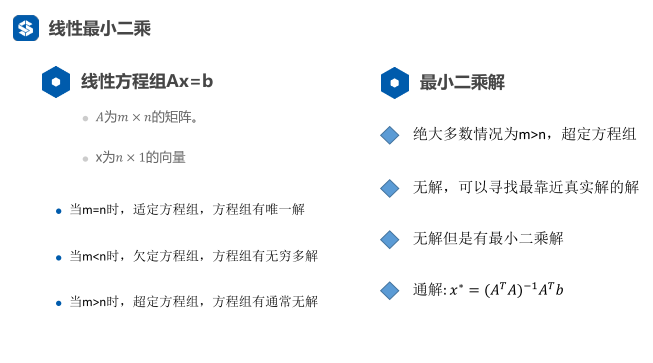


**航迹推算：**



**里程计标定：**

**线性最小二乘的原理：求解方程AX=b**



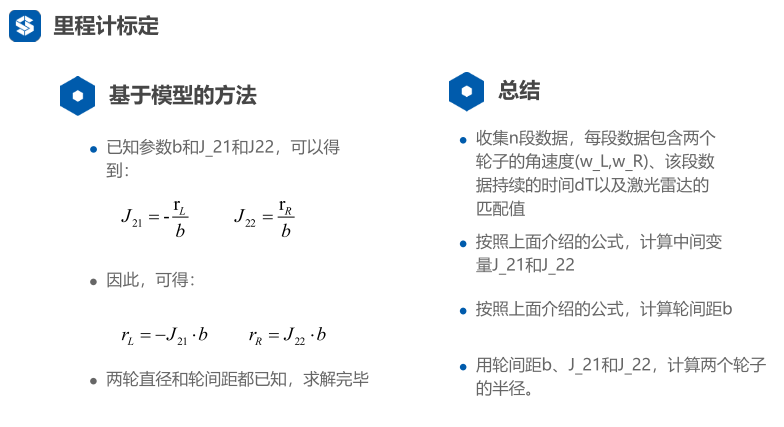
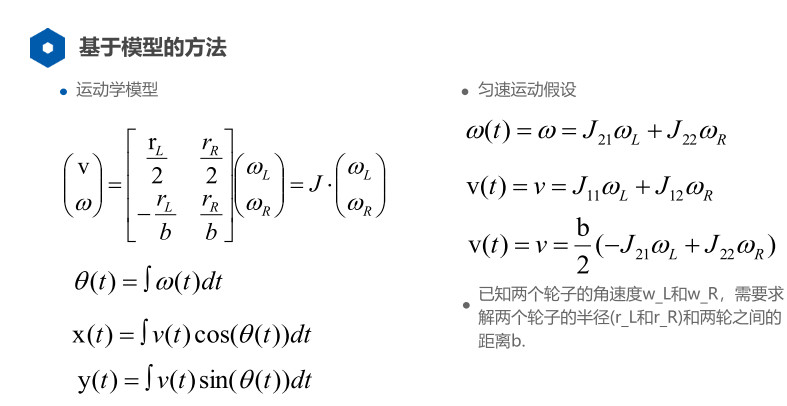
**最小二乘线性拟合**

**最小二乘在里程计标定中的应用**

**分为两种应用：**

* **直接线性方法**：通用性强、精度不高、实现比较简单||用激光了我打的scan-match数据作为真值；里程计测得的数据作为控制量
* **基于模型的方法**：精度高、实现复杂、特异性高

**基于模型的方法：**

1. 假设激光雷达位于车体的正中心
2. 激光雷达的匹配值作为观测值
3. 里程计的积分作为预测值
4. 通过最小化预测值和观测值的差即可得到里程计的参数
5. 

###### 激光雷达运动畸变的去除

###### 激光雷达传感器介绍

激光雷达测距原理（与结构光类似）：

* **三角测距**
* **TOF（飞行时间测距）**

三角测距

其中**三角测距：激光器发射激光，打到物体上。反射光由线性CCD接收，激光器和探测器间隔的及处理**类似于双目测距，和基线成反比，基线越短有效测距范围就越大（通常是10cm左右）这种雷达一般国产的比较多，像思岚、镭神等。

CCD是Charge Coupled Device(电荷耦合器件)的缩写,它是一种半导体成像器件。

TOF

激光器发射一个激光脉冲，计时器记录下来出射的时间，接收器接收返回的激光，记录下回返的时间。两个时间做差就是激光的飞行时间，光速是一定且已知的3\*10 -8，就可以计算出距离。

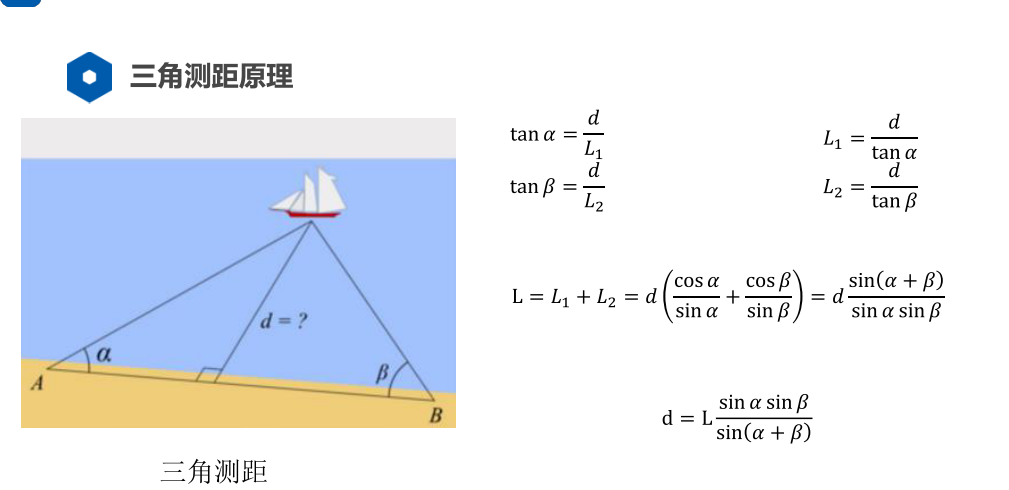
难点：

1. 计时。计时系统的要求很高
2. 脉冲信号的处理（回波和脉冲的问题）

优点：

1. 实时性好，测量距离远；
2. 采样率更高。转速一定情况下，每一帧点云的数量和点云的角分辨率。三角雷达20K以下，TOF更高
3. 测量精度不随长度增加而变化，所以一般在几十米的范围内都能保持几个厘米的精度。
4. 转速（帧率）三角雷达的最高转速通常在20HZ以下，TOF通常在30-50HZ.。三角雷达通常采用上下分体的结构，上面转的部分负责激光发射、接收和采集；下面负责电机的驱动进而供电等。TOF雷达转速更高。（转速高就可以减少畸变的影响）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测距原理 | 测量距离 | 价格 | 抗干扰能力 | 应用场景 | 精度 |
| 三角测距 | 中近距离精度较高 | 便宜 | 易受干扰（容易受到强光的干扰） | 室内使用 | 远距离的时候精度较低 |
| TOF | 测距范围广 | 贵 | 抗干扰能力强 | 室内室外都行 | 测距精度高 |



TOF波形相位差

##### 毫米波雷达信号处理算法

###### 毫米波测量原理

**毫米波雷达测距原理（77GhZ FMCW）**：

调频连续波测距的基本原理：

1、发射波TX为高频连续波，其频率随时间按一定规律规律变化。

2、发射波TX遇到物体之后反射，接收器接收到反射波RX。

3、信号的发射到接收，产生一定的时间间隔 t。由这个时间间隔，得到频率差值信号IF signal。

4、对频率差值信号，进行FFT变换，得到对应的频谱。频谱的峰值处对应的频率 f 和距离 d 具有对应关系，进而得到距离d。

5、测距分辨率的分析。

6、测距范围的分析。

##### 摄像头信号处理算法（参考视觉十四讲的内容）

###### 摄像头测量原理—主要包括深度相机的原理

摄像头测量原理：

**相机原理分为三类：**

(1)结构光(Structured-light)，代表公司有奥比中光，苹果(Prime Sense)，微软 Kinect-1，英特尔 RealSense, Mantis Vision 等。

(2)双目视觉(Stereo)，代表公司 Leap Motion， ZED， 大疆;

(3)光飞行时间法(TOF)，代表公司微软 Kinect-2，PMD，SoftKinect， 联想 Phab。

1. **红外结构光：结构光投射到待测物体上被待测物体的高度调制，被调制的结构光被摄像系统采集，传至计算机内分析计算后可得到被测物的三位面形数据。（调制方法分别为时间调制和空间调制）空间调制方法为结构光场的相位、光强等性质被待测物体的高度调制后都会发生变化。**。**通常采用特定波长的不可见的红外激光作为光源，它发射出来的光经过 一定的编码投影在物体上，通过一定算法来计算返回的编码图案的畸变来得到物体的位置和深度信息。**

条纹结构光，代表传感器 enshape ，

编码结构光，代表传感器 Mantis Vision, Realsense(F200),

散斑结构光 ，代表传感器 apple(primesense), 奥比中光。

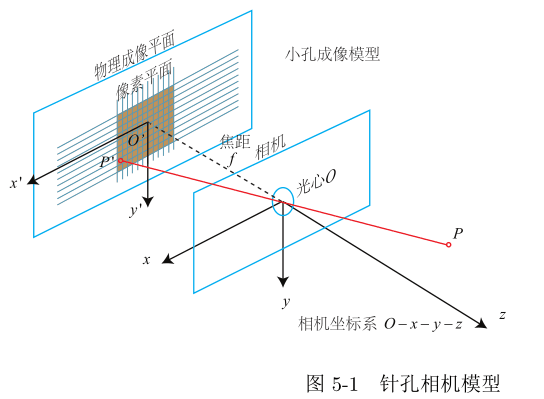
1. **飞行时间法**

**输出彩色图和深度图，计算像素中的3D相机坐标，生成点云。RGB-D相机的成本是多少。**

**优点：**实时测量每个像素点的距离，但是适用范围有限、红外进行深度测量的相机，容易受日光或者其他传感器发射的红外光干扰，不能在室外使用，同时使用多个会相互干扰，透射材料会失效**。**

###### 简单的相机模型：

针孔相机模型

* ****

主要包括**外参**和**内参**两个参数。内参通常出厂的时候是确定的或者使用网上的教程也可以进行标定，它包括像素坐标系的缩放倍数和坐标原点移动组成的矩阵K；外参主要包括旋转矩阵R和t；也是机器人的轨迹（待估计的目标）

畸变 由透镜形状引起的畸变称为**径向畸变**：径向畸变分为桶形畸变（图像的放大率随着离光轴的距离增加而减小）和枕形畸变

**切向畸变**（由于在组装过程中不能是透镜和成像面严格平行）

**世界坐标：正常的世界坐标系**

**像素平面：**物理成像平面，x-\y-\z-

**对称的成像平面：**将物理成像平面对称到相机前面

**归一化平面：**将对称的成像平面归一化，距离相机光心的距离为1

**相机坐标系：**z轴指向相机前方，x向右，y向下，相机光心为原点

主要包括五个畸变系数找到这个点在像素平面的正确位置：

1. 对三维空间点投影到归一化图像平面。设它的归一化坐标。
2. 对归一化平面上的点进行径向畸变和切向畸变纠正。
3. 将纠正后的点通过内参数矩阵投影到像素平面，得到该点在图像上的正确位置。

单目相机成像过程：

1. 世界坐标系下一个固定点P，世界坐标P\_w
2. 由于相机在运动，运动可以有R,t或者变换矩阵描述。
3. 投影到归一化平面
4. 归一化后的坐标经过内参后，得到对应的像素坐标



双目相机模型

单目相机无法测量点在空间中的具体位置。一般由左眼和右眼水平放置的相机组成它们之间的距离叫做双目相机的基线。视差和距离成反比，视差越大，距离越近。双目深度需要用到GPU或者FPGA

RGB-D相机模型

**相机原理分为两类：**

1. **红外结构光：结构光投射到待测物体上被待测物体的高度调制，被调制的结构光被摄像系统采集，传至计算机内分析计算计算后可得到被测物的三位面形数据。（调制方法分别为时间调制和空间调制）空间调制方法为结构光场的相位、光强等性质被待测物体的高度调制后都会发生变化。**
2. **飞行时间法**

**输出彩色图和深度图，计算像素中的3D相机坐标，生成点云。RGB-D相机的成本是多少。**

**优点：**实时测量每个像素点的距离，但是适用范围有限、红外进行深度测量的相机，容易受日光或者其他传感器发射的红外光干扰，不能在室外使用，同时使用多个会相互干扰，透射材料会失效**。**

鱼眼相机

**相机畸变处理（畸变校正）**

**常用的两种去畸变处理：**

1. **先对整张图象进行去畸变，得到去畸变后的图像，然后讨论图像上的点的空间位置（视觉SLAM常用）**
2. **先考虑图像中的某个点，然后按照去畸变方程，讨论去畸变后的空间位置。**

**图像**

在图像中，数组的行数对应图像的高度，列数对应图像的宽度。彩色图像的表示需要通道的概念来表示。通道的顺序可以自由定义，默认顺序为BGR、如果要表示透明度，则默认顺序是RGBA

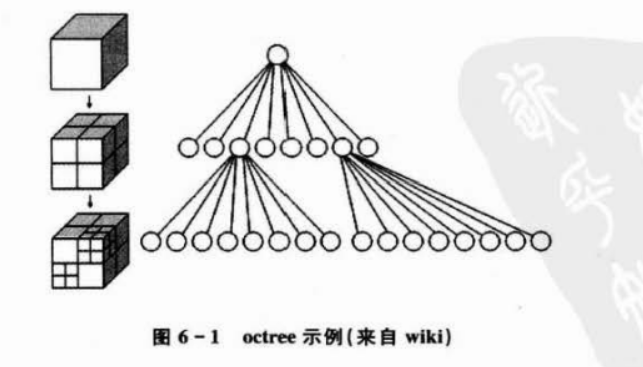
###### 八叉树地图

常见空间索引一般是**自顶向下**逐级划分空间的各种空间索引结构，比较有代表性的包括 **BSP 树、KD 树、KDB 树、R 树、R＋树、CELL 树、四叉树和八叉树**等索引结构，而在这些结构中 **KD 树和八叉树** 在 3D点云数据组织中应用较为广泛。

八叉树定义Octo-map地图—Octo-tree八叉树

|  |  |
| --- | --- |
| **八叉树地图的优点是灵活、压缩、随时可更新的形式，格式是.bt和.ot。** |  |
| **压缩--八叉树为什么要比点云地图节省空间？** | 原因是现实中很多物体都是连成一片的，例如一片空白和一片物体，一片得物体不需要展开，直接用一个节点表示就可以了。 |
| **八叉树地图的表示方式** | 1．0-1表示可以用一个比特来存储，节省空间。 |
| 2.由于有噪声，用概率来表达某个节点是否被占据的事情。浮点数0-1初始0.5不断更新 |
| 3.方式更新过程中超界，所以使用概率对数值来描述。 |
|  |  |

**八叉树结构**通过对三维空间的几何实体进行**体元剖分**，每个体元具有相同的时间和空间复杂 度，通过循环递归的划分方法对大小为 Zn × 2n×2n的三维空间的几何对象进行剖分，从而构成一个**具有根节点的方向图。**



PCL中的octree模块及类介绍

Octree模块中的16个类：

classpcl::octree::Octree2BufBase<DataT,LeafT>

**点云地图**有了三维结构，也进行了**体素滤波**调整**分辨率，**但是点云有几个**明显的缺陷：（把八叉树的论文翻译整理一下）**

1. **点云地图的规模很大，pcd（这是什么文件，怎样进行保存）文件也会很大，占用的内存非常大。**
2. **点云地图无法处理运动物体（不够实用）**

**八叉树地图**是一种灵活的、压缩的、随时更新的地图形式**。八叉树地图**就是**把三维空间建模为许多小方块（体素），将一个小方块切成8块，（将小方块分到最小空间）就是八叉树（Octo-tree），它的体积和深度呈指数关系。**

**为什么更节省空间：因为在实际建图过程中，物体都是一块区域一块区域出现的，八叉树的中，如果一个方块的子节点全部占用或者全部不占用，就没必要对他进行展开到叶节点。（比特是什么）一般是0.1表示，但是实际会有噪声可能会出现一会是1、一会是0；或者0和1之间不断的跳转切换；所以考虑使用概率来表示通常每个方格初始的取值是0.5；如果不断被占据就增加大于0.5；如果不断空闲，就减小小于0.5；所以用概率对数值（log-odd）来表示；**

**Logit模型和逆logit模型来做概率变换**

**点云的体素滤波器是什么？**

**生成的地图存成octomap.bt文件**

**Octovis是可视化界面，包括地图的查看、旋转、缩放等等。Octomap可以查询任意点的占据概率，以此设计在地图中进行导航的方法。**

**注意实时三维重建于SLAM 的区别：**

|  |  |  |
| --- | --- | --- |
| **方法** | **联系** | **区别** |
| **SLAM** | **都是基本上的实时定位与建图算法** | * **轻量级、小型化发展，有些方案甚至放弃了建图和回环检测的部分例如激光里程计LOAM、视觉里程计** * **主要是以定位为主体，定位算法（稀疏特征或稀疏直接法）可以满足实时性的需求，而地图的加工可以在关键帧处进行处理，无需实时响应** |
| **实时三维重建** | * **大规模、大场景的重建方向发展** * **主要是以建图为主体GPU进行加速** * **TSDF 截断符号距离函数，不严谨的叫法是TSDF地图** |
| **TSDF地图整个存储在GPU显存当中，而不是内存；并行进行计算和更新（鲁棒性更加好）** | **定位：主要讲当前的RGBD像级图像和GPU中的TSDF进行比较，估计相机的位姿；**  **建图：根据相机的位姿，对TSDF地图进行更新。（双边贝叶斯滤波，去除深度图中的噪声）** |  |

## 基于rikirobot的两轮驱动机器人定位、建图

### 项目介绍

基于装有Velodyne VLP16线激光雷达的rikiRobot，实现二维栅格地图的构建，并基于已知地图实现车辆的定位。

1.在Linux/ROS环境下搭建了rikiRobot机器人平台，完成ROS和MATLAB\Simulink的通讯调试。

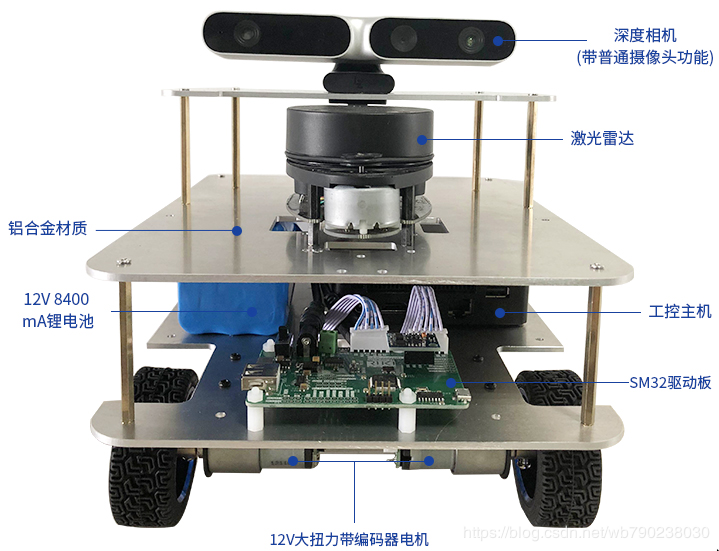
2.使用基于粒子滤波的Gmapping算法进行二维栅格地图的构建，使用AMCL蒙特卡洛定位算法进行位置跟踪，实现基于二维地图的定位。

3. 实现Cartographer等二维的SLAM建图方法以及相应的算法。

### 算法细节

#### 一、RikiROBOT平台的组成和搭建

两驱三轮智能机器人平台，（主要包括深度摄像头+激光雷达+工控机+SM32驱动板+小车平台-12V大扭力带编码器电机）



平台由三层空间组成，系统主要分为**执行机构、驱动系统、传感系统和控制系统**。

1. **执行机构**：人体的手和脚，直接面向工作对象的**机械装置**
2. **驱动系统**：人体的肌肉和筋络，负责驱动执行机构，**将控制系统下达的命令转换成执行机构需要的信号**；
3. **传感系统**：人体的感官和神经，主要完成信号的**输入和反馈**，包括内部传感器（里程计、IMU等）和外部传感器系统（摄像头、激光雷达等）；
4. **控制系统**：人体的大脑，**实现任务即信息的处理，输出控制命令信号**；控制系统需要基于处理器（树莓派、**工控机**、TK2等）实现，在处理器之上，控制系统需要完成机器人的算法处理、关节控制、人机交互等丰富的功能；

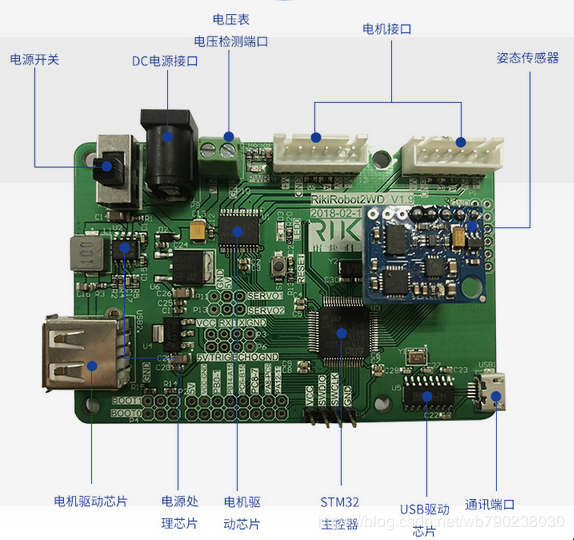


##### 1\*执行机构



##### 2\*驱动系统

riki 的平台搭载了如图所示的**底层驱动板**，其中集成了**电源驱动**、**电机驱动**、**姿态传感器接口**等底层驱动功能；



###### 1、电源子系统

一般的机器人平台需要有个电源子系统，但是riki家的电源是 **12V** 2800mah的锂电池，直接给驱动板供电；驱动板通过**USB micro**的通讯端口给工控机供电、通过**电机接口**给两个主动轮供电、通过**传感器接口**给9轴IMU供电。

###### 2、电机驱动子系统

直流电机的驱动子系统自上而下可分为两部分，第一部分是**与电机直连的电机驱动模块，**可将上层下达的控制信号转换为电机需要的电源信号；第二部分是**电机控制模块**，接收控制系统的运动命令，实现对电机的**闭环驱动控制**；

###### 3、传感器接口（9轴IMU、里程计）

riki将部分传感器接口集成到驱动板上，可以处理9轴IMU、里程计等传感器的信号；

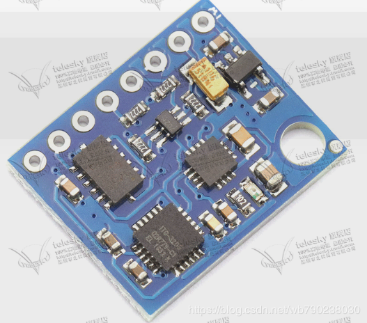
##### 3\*传感器系统

###### 内部传感器

riki的内部传感器包括GY-85 九轴IMU以及集成在小车主动轮上的编码器；

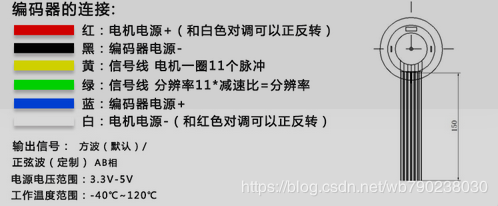
GY-85 九轴IMU

GY-85 九轴IMU链接：<https://detail.tmall.com/item.htm?spm=a230r.1.14.20.c6312336FAOwMf&id=41337816280&ns=1&abbucket=12>



霍尔编码器

霍尔码器链接：<https://item.taobao.com/item.htm?spm=a230r.1.14.60.6b2a181dCRD5Qt&id=556442142947&ns=1&abbucket=12#detail>



###### 外部传感器

激光雷达之前是用的思岚A1，但是我们课题组有VelodyneVLP16所以直接用VelodyneVLP16替换掉，因为后面会用这个激光雷达上实车。

思岚A1





Velodyne VLP16参数：

传感器的有效范围为100M,16个通道、大约3000000（30万）点/秒，360度水平视角，30度垂直视角（+-）度，工作电压大约为9-32V

##### 4\*控制系统

###### 硬件方案

控制系统硬件选择上有两种方案：**单处理器和多处理器。**

**单处理器**，就是使用PC作为控制系统平台，控制系统的功能在**PC上使用ROS系统实现**，通过**USB串口**与**机器人驱动板**直接通信，采集机器人信息并控制机器人移动，这种方案简单易用处理器性能强大，可很快实现ROS功能，但是PC机体大、灵活性差（必须用USB线跟驱动板相连接）、接口种类少、且无法进行远程监控，**不作为推荐方案。（可以用VNC Viewer进行远程控制）**

**多处理器，**针对第一种方案，做ROS开发的基本会选择“**PC+嵌入式系统**”的方案，我选择的是“PC+工控机Intel3865”，**将工控机搭载在机器人上作为机器人本体的控制系统**，PC通过无线网络与工控机连接，实现对机器人人的远程监控、图形化显示，以及处理复杂功能的运算。

1. 如何实现工控机与PC机的通信？

**设置IP地址，**工控机与PC必须在同一个局域网下，分别在计算机roswing-K45VD、syw上使用ifconfig命令查看计算机的局域网IP地址，分别在两台计算机的/etc/hosts 文件中加入对方的IP地址和对应的计算机名，然后分别在两台计算机上ping下对方，看看网络是否畅通。

采用工控机+PC进行双控制模式（直接使用VNC 进行远程控制）

控制手柄

主要用的是罗技F710进行控制

#### 二、ROS\SIMULINK 的调试通讯、对ROS 进行网络环境配置

##### 1.ROS与Matlab 进行通讯连接

Matlab/Simulink可作为ROS master或者普通节点，实现在ROS中对节点的管理和通讯

要求：

1. ROS主机和Matlab 安装的从机要在同一个局域网下，

测试环境Ubuntu 16.04 ROS版本是Ros Kinect Matlab2017b

查看IP地址的方法：输入ifconfig

Windows查看IP地址的方法是在CMD命令窗口输入ipconfig

1. 在ubantu的~/.bashrc文件末尾添加

$export ROS\_IP=windows.local（Ubuntu 的IP）

$export ROS\_HOSTNAME=ubuntu.local

$export ROS\_MASTER\_URI=http://ubuntu.local:11311

1. 在ROS端输入roscore;
2. 在matlab命令窗口输入命令，连接到ROS网络并创建节点

* setenv('ROS\_MASTER\_URI','http://192.168.43.136:11311')-》Ubuntude IP地址
* setenv('ROS\_IP','192.168.1.103')；
* rosinit

1. 连接成功以后，在matlab中输入命令创建话题

* chatpub = rospublisher('/talker话题名', 'std\_msgs/String消息类型');
* msg = rosmessage(chatpub);

1. 初始化指定了话题名和消息的类型；
2. 在消息中放入数据，并通过连接到ROS网络的结点话题发送消息

* latchpub = rospublisher('/talker', 'IsLatching', true);
* msg.Data = 'Hello, This is Matlab发布消息';
* send(chatpub,msg);

1. rostopic echo /talker

补充：

想使用Matlab/Simulink进行ROS的调试和开发，在每次启动Matlab时，如下两句都需要运行一次：

* setenv('ROS\_MASTER\_URI','http://192.168.43.136:11311')-》Ubuntude IP地址
* rosinit

rosnode list可以看见Matlab节点

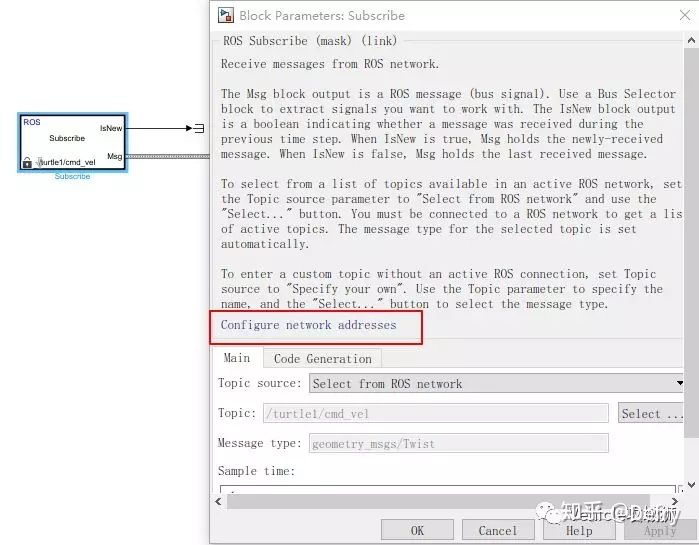
Rostopic list可以看到创建的/talker话题

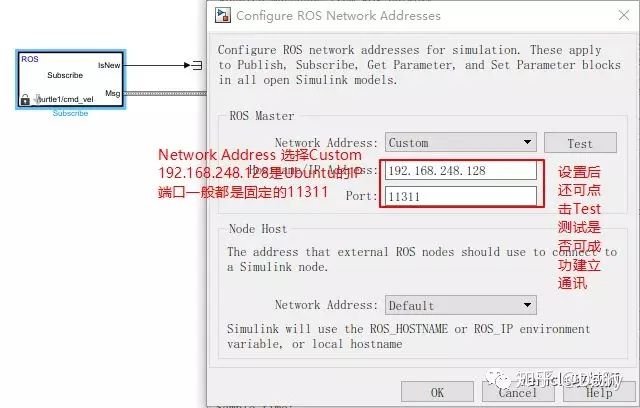
Rosrun rqt\_graph rqt\_graph可以查看节点之间的关系图

##### 2.Ros与Simulink进行通讯连接

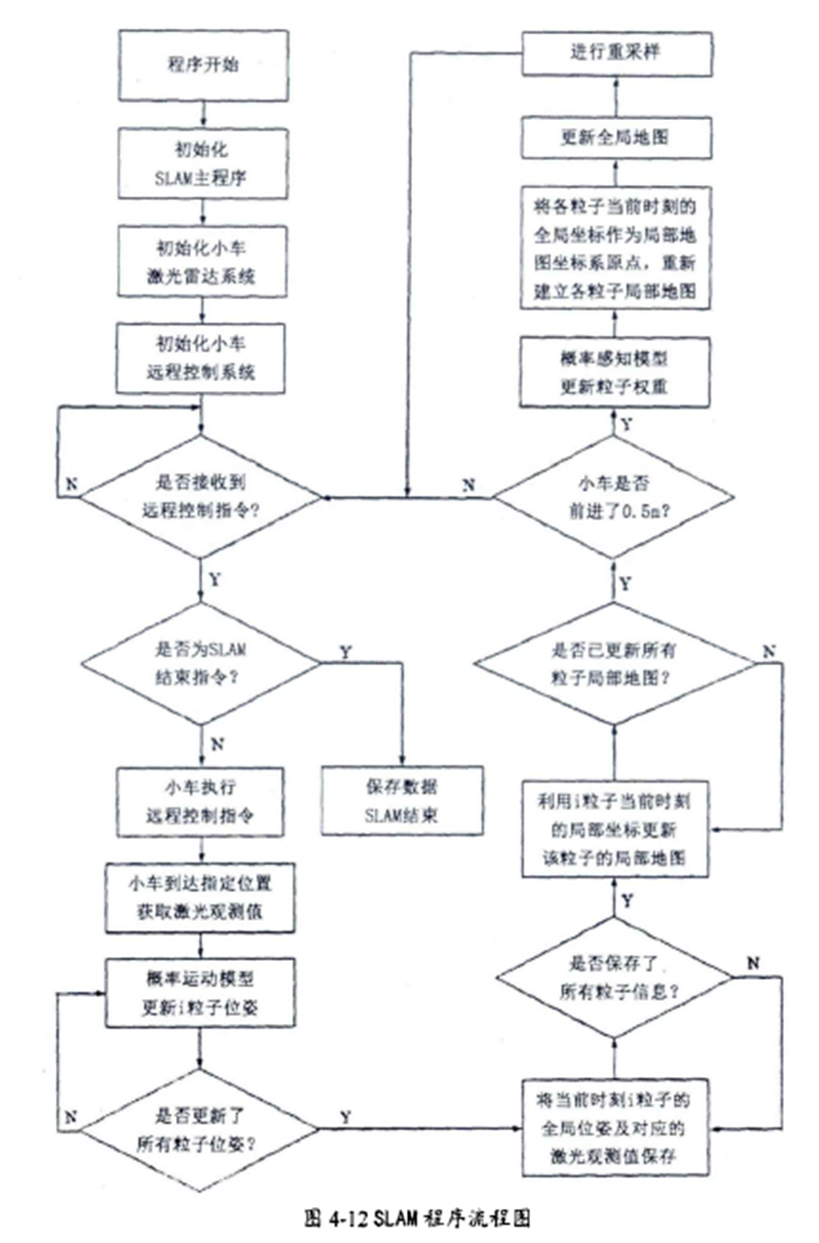
若Simulink与运行在虚拟机Ubuntu中的ROS主节点建立通讯，除了运行如上两句，还需要进行进一步的设置，在虚拟机Ubuntu中启动主节点roscore，运行节点（例如TURTLEbotism节点，通过键盘控制其运动，turtlebosim 通过/turtle1/cmd\_vel的主题发布出来），在Simulink建立模型并订阅这个话题，

ROS Subscribe（mask）(link)的模块，打开Configure network address进行通讯设置





#### 三、基于粒子滤波的Gmapping算法流程（没弄完）



##### 1.坐标系变换

通常包括三个坐标系：

* map（全局地图坐标系）
* base\_link(机器人坐标系)
* base\_laser(传感器坐标系)

三个坐标系通过一个变换矩阵可以将不同坐标系下的坐标信息放到同一个坐标系下。

##### 2.Gmapping的应用条件

###### Gmapping的订阅的话题

* tf ([tf/tfMessage](http://docs.ros.org/api/tf/html/msg/tfMessage.html))
* scan ([sensor\_msgs/LaserScan](http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html))

###### Gmapping发布的话题

map\_metadata ([nav\_msgs/MapMetaData](http://docs.ros.org/api/nav_msgs/html/msg/MapMetaData.html))

map ([nav\_msgs/OccupancyGrid](http://docs.ros.org/api/nav_msgs/html/msg/OccupancyGrid.html))

~entropy ([std\_msgs/Float64](http://docs.ros.org/api/std_msgs/html/msg/Float64.html)) 估计机器人姿势上的分布的熵（值越大表示不确定性越大）。

###### Gmapping的服务（services）

dynamic\_map ([nav\_msgs/GetMap](http://docs.ros.org/api/nav_msgs/html/srv/GetMap.html))

###### Gmapping的Parameters

###### Gmapping需要的tf Transforms

<the frame attached to incoming scans> → base\_link

base\_link → odom

###### Gmapping提供的tf Transforms

map → odom

##### 基于粒子滤波定位的方法和原理

所谓粒子滤波就是指：通过寻找一组在状态空间中传播的**随机样本**来近似的表示**概率密度函数**，**用样本均值代替积分运算**，进而获得系统状态的最小方差估计的过程，这些样本被形象的称为“粒子”，故而叫粒子滤波。

贝叶斯滤波

**贝叶斯估计的方法**

**状态估计问题**（目标跟踪、信号滤波）就是根据之前一系列的已有数据 y\_1:k（后验知识）递推的计算出当前状态x\_k的可信度。

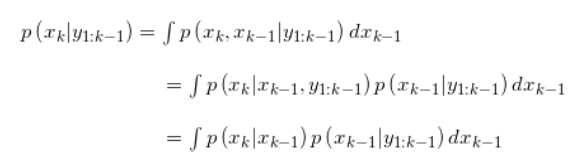
过程：

这个可信度的概率公式是条件概率，通过**预测**和**更新**两个步骤递推计算。

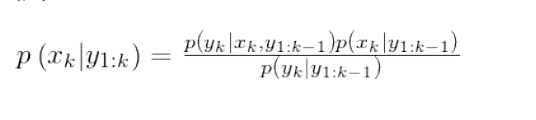
**状态方程：**

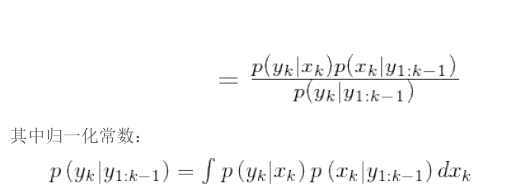
**测量方程：**

* **预测过程：存在假设（一阶马尔科夫假设）：当前时刻的状态下x\_k只与上一时刻的状态x\_k-1有关。**P(X\_k-1|y\_k-1)🡪P(x\_k|y\_1:k-1)



* **更新：k时刻测量的数据y\_k只与当前状态x\_k有关。**P(x\_k|y\_1:k-1)🡪P(x\_k|y\_1:k)





公式推导见印象笔记中粒子滤波部分[白巧克力唯心的粒子滤波.pdf](https://app.yinxiang.com/shard/s50/nl/13289595/41a79bf6-5e6f-4222-af4d-782ad8834e33)

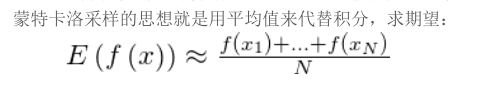
**缺点：**是用到积分，对于非线性、非高斯系统比较难。

蒙特卡洛采样

方法：蒙特卡洛采样的核心就是用平均值来代替积分，求期望。

过程：

假设从一个目标概率分布总采样得到一系列的样本（粒子），可以用样本去估计这个分布的某些函数的期望值。



优点：

贝叶斯后验概率的计算里要用到积分，为了解决这个积分难的问题，可以用蒙特卡洛采样来代替计算后验概率。从后验概率中采样很多粒子，用他们的状态求平均得到滤波结果。

缺点是：后验概率未知

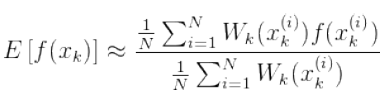
重要性采样

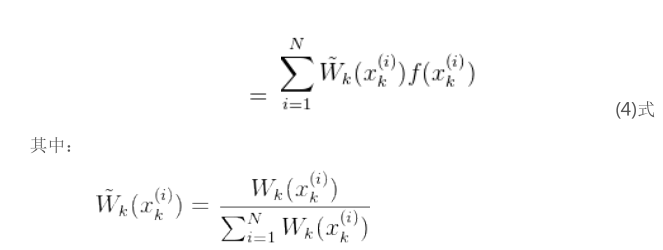
方法和优点：

既然无法从目标分布中采样，从已知概率分布去采样也行，得到所有采样粒子状态的加权平均值。

过程：

P（x\_k|y(1:k)）





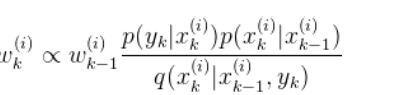
缺点是效率低（每增加一次采样就得计算一次后验）

序贯重要性采样（SIS）

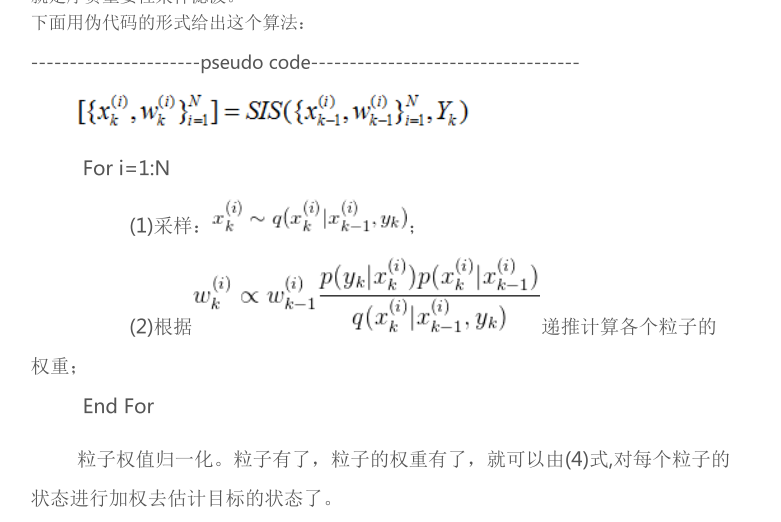
**方法和优点**：为了解决重要性采样中每增加一次采样都要重新计算，因此以递推的形式计算权重，这就是序贯重要性采样（SIS），粒子滤波的原型。

**过程：**

重要性概率密度函数不需要用积分。粒子权值递归形式。



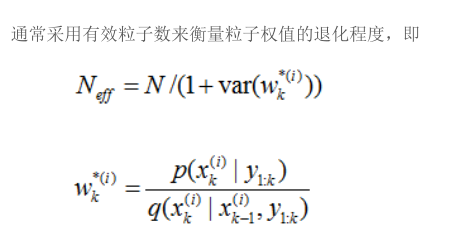
存在假设：**重要性分布只与前一时刻的状态下x（k-1）以及测量y(k)有关。**



**缺点：粒子权重退化问题，**

重采样

**优点和方法**：为了解决SIS中粒子权重退化的问题，粒子权值的方差随时间增大，无效采样粒子数目增加导致估计性能下降。



有效粒子数越小，即权重的方差越大，也就是说权重大的和权重小的之间差距大，表明权值退化越严重。

**解决方法**：

* 增加粒子数量（但是会增加计算量，实时性也比较差）。
* **选择合适的重要性概率密度函数。**
* **在序贯重要性采样后采用重采样的方法。**

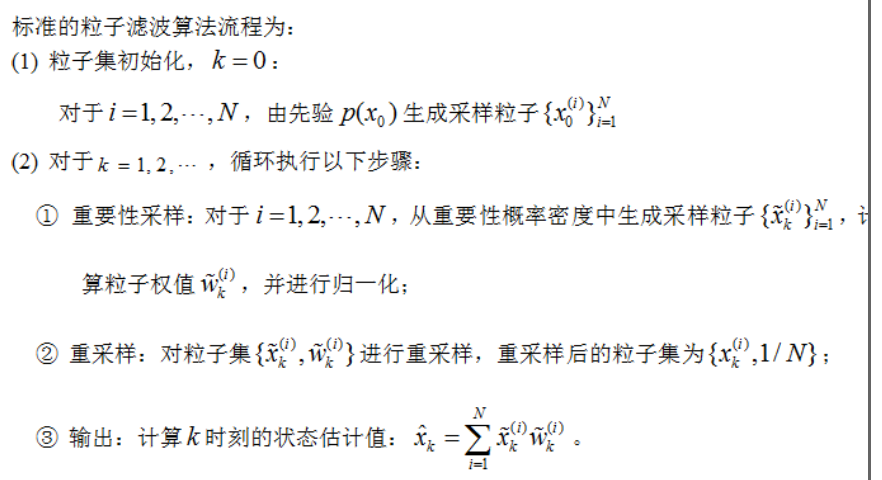
思路：**丢弃掉掉权重小的粒子，保持粒子数目不变，用权重大的粒子去复制。补充求其掉的权重小的粒子的位置。**

**缺点：权重大分为两种情况，一种是分子大就是后验概率大，这种很优秀，正确。**

**但是如果权重大是因为分母小，就不优秀了。重采样会导致粒子的多样性丢失。**

**解决方法：正则粒子滤波**

粒子滤波算法流程



整个流程图



SIR滤波Sampling Importance Resampling Filter

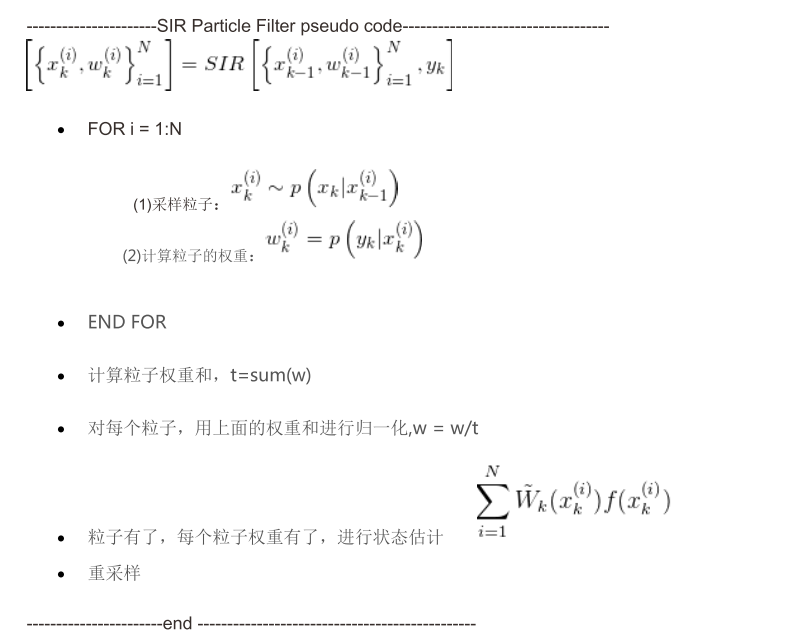
**方法：**

针对粒子的**重要性概率密度函数**做出特定的选择，将其选取为**先验概率**。（先验概率可以由状态方程得到），带入到SIS推导出的权重公式，

**优点：**

* **SIR只和系统状态方程有关，不需要自己去设计概率密度函数。**
* **简单易用。**

**过程：**



**缺点**：

* 效率不高
* 对奇异点敏感的问题。
* 没有就加入测量，只是凭先验概率进行采样，而不是修正后的后验概率。

粒子滤波的实际考虑和特性

密度提取：粒子滤波下的样本集表示连续置信度的高度离散化。从这样的样本中提取一个密度

卡尔曼滤波和粒子滤波的区别

|  |  |  |
| --- | --- | --- |
| 算法的区别 | 卡尔曼滤波 | 粒子滤波 |
| 使用场景 |  |  |
|  |  |  |

#### 四、AMCL定位算法的原理和实现

AMCL定位算法的原理

#### 五、卡尔曼滤波方法和实现

##### 卡尔曼滤波器的原理和公式最好能罗列出来

#### 六、Gmapping包跑出来的效果图+需要设置的参数

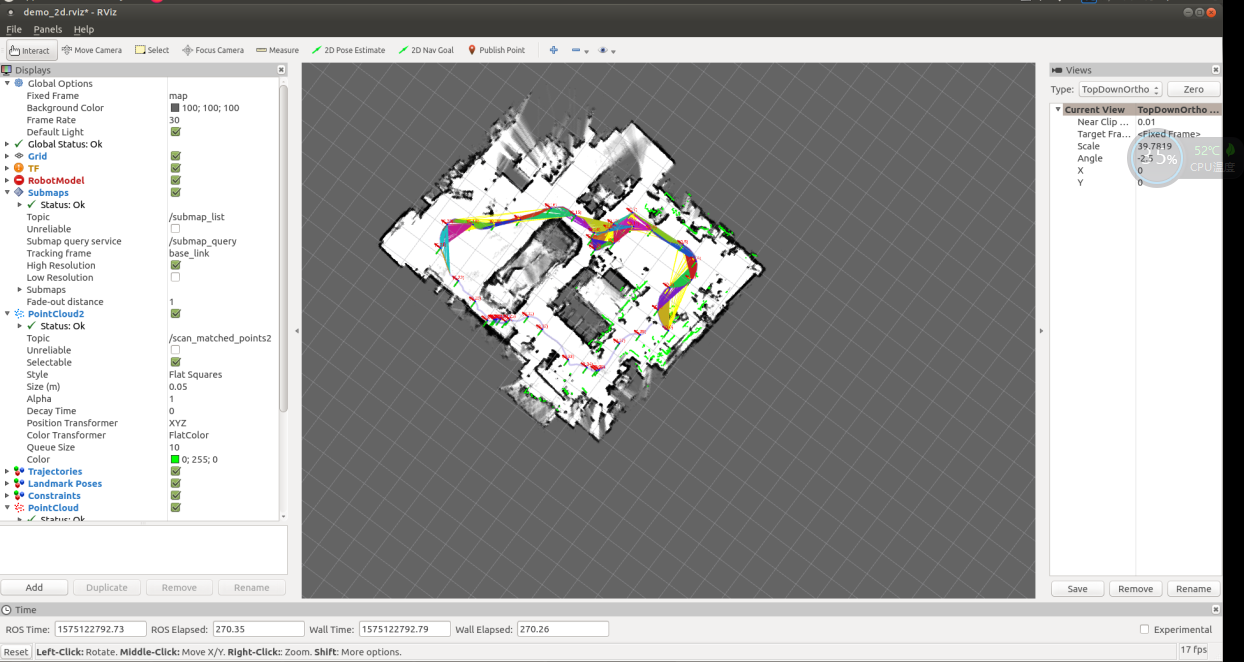
#### 七、Cartographer的原理和算法（不需要讲详细的代码完全讲出来，但是需要讲包括哪几部分说出来）

#### 八、Cartographer跑出来的效果图

##### Cartographer启动程序指令：

* roslaunch rikirobot bringup.launch
* roslaunch racecar\_telelop joystick.launch
* roslaunch velodyne\_pointcloud VLP16\_lab.launch
* 在桌面启动roslaunch cartographer\_2d

##### Cartographer实际建出的栅格地图：



## HKPC-In-wheelMotor Vehicle project

### 项目介绍

应用MATLAB/Simulink进行基于模型设计的嵌入式控制系统开发，以实现后轮轮毂电机汽车的纵向运动控制；进一步进行了分布式轮毂电机驱动/四轮独立转向全线控电动汽车的研究。

主要工作：

1.在CarSim和PanoSim中建立轮毂电机汽车的纵向动力学模型及仿真场景；基于MATLAB/Simulink实现纵向车速控制；充分利用轮毂电机电动汽车车轮驱动力精确可控的优点，提出多模式切换控制策略，实现不同工况下的控制。

2.参与上位机与控制器的通讯架构设计；应用MicroAutoBox进行快速原型设计并进行联合调试；基于ControlBase自动代码生成嵌入式开发工具链开发控制器，并在实车对上述开发的控制策略进行试验验证。

3.接续上述工作，进一步参与开发了分布式轮毂电机驱动/四轮独立转向全线控电动试验平台，对驱动电机的控制逻辑进行设计。

### 算法细节

#### HKPC项目的逻辑切换部分弄完,怎样进行控制的 包括王鹏的那些文件

#### MicroAutoBox的使用步骤和作用、联合调试

### CarSim的使用和Simulink的联合仿真

**CarSim8.0**中主要包括**车辆参数和仿真工况的设置**、**数学模型的求解**、**运算结果的后处理**（做图和视频观看）

### PanoSim的使用和Simulink的仿真

PanoSim 和Simulink 进行联合仿真

### PID调节

PID是通过系统误差利用**比例**、**积分**和**微分**计算出控制量进行控制的。

P主要起的是主导的作用，积分I主要是辅助调节的作用，微分主要是补偿的作用。

1. 关掉积分和微分的，**先调P**，P由小往大调，看能不能快速响应，**缩小静差不产生振荡**，调节稳态误差（静差），过大的比例系数会是系统产生超调，并产生震荡，稳定性变差。
2. 如果**静差**不满足，必须加入I**积分环节**，作用是减少超调，**使系统稳定增加**，**减少震荡**再调**I**，I从大往小调，最好是能够快速响应，超一点点也行。
3. 最后调**D**。加快速度的响应速度，使**系统超调量减少，稳定性增加。**一般系统D为0,1或者2（不能再大了）通常影响滞后效果。
4. 在看整体的相应，进行**微调**，超调震荡就得看是P还是I还是D了。

**比例控制**是一种最简单的控制方式。其控制器的**输出与输入误差信号**成比例关系。当仅有比例控制时系统输出存在稳态误差。比例调节作用快，系统一出现偏差，调节器立即将偏差放大输出。但系统存在余差，比例度越大，过渡过程越平稳，但余差越大，比例度越小，过渡过程易振荡，比例度太小时，就可能出现发散振荡。

**积分调节**作用的输出不仅取决于**偏差信号的大小**，还取决于**偏差存在的时间**，只要有偏差存在,尽管偏差可能很小，但它存在的时间越长，输出信号就越大，只有消除偏差，输出才停止变化。

**比例+积分(PI)**控制器，可以使系统在进入稳态后**无稳态误差**。

**在微分控制中**，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。由于积分调节器输出值大小与误差的持续时间有关，就会使系统的调节过程变慢，带来的系统动态性能变差。此外，当系统受到冲击式偏差时，偏差的变化率很大，而PI调节器调节速度很慢，就会造成系统的振荡。

**比例+微分**的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而**避免了被控量的严重超调**。

随着**Kp**值的增大，系统的**稳态误差逐渐减小**，稳态性能得到很好的改善，但是，Kp的增大，使系统的超调量同时增加，系统的动态性能变差，稳定性下降。

在比例控制的基础上增加**微分控制**并**不会影响系统的稳态误差**，而**增大微分常数Kd**可以的减小系统的**超调量**和**调节时间**，在不影响系统的稳态性能的基础上改善了系统的动态性能。

**增大积分环节中的增益Ki**则会使系统的**超调量增加**，系统的**震荡加剧**，从而破坏了系统的动态性能。要使系统各项性能尽可能的好，

**只有一边增大Ki加快系统消除稳态误差的时间，一边减小Kp来改善系统的动态性能。**

PID控制中的Ki；与PI控制器的设计相同，但是为了满足超调量和上升时间这两个性能指标，比例增益Kp和微分增益Kd应同时调节。

增大Kd可以有利于加快系统的响应速度，使系统超调量减小，稳定性增加，同时增大Kp可以进一步加快系统的响应速度，使系统更快速。

### 张佳旭师兄的留下来的资料看一看

### Control base的使用

#### Control Base自动代码生成

### 上位机与控制器的通讯框架设计部分

制动试验台、驾驶模拟器和操作台之间是通过网线进行通讯的。驾驶模拟器将实时的车辆状态信息、之动态将制动主缸和轮缸信息通过网线传递到操作台中，并将之动态的操作界面实时显示。操作台发出控制指令通过网线传递给驾驶模拟器和制动试验台。

# 竞赛

## “认证杯”研究生数学建模竞赛

我们组的题目是A题。假设纸飞机的翼面是精确的平面，建立数学模型。根据纸飞机的三视图和中心位置，计算投掷出手的合理角度使飞行距离尽可能远。

我主要负责使用Simulink对纸飞机进行建模，主要建立了一种离散迭代模型，求解得到飞行距离和手持位置、初始高度等的关系。

## 华为杯数学建模竞赛

华为杯我们做的是D题。

题目主要内容是：（包括论文和源程序m文件）

我主要负责**第二问**的运动学片段的划分和**第三问**行驶工况的构造。

**第二问**：对运动学片段进行定义，采用开始于怠速起点到终止于下一个怠速起点。将预处理后的数据根据加速度、减速度、怠速和匀速进行划分片段。

**第三问：**使用题目中定义的19个特征参数，使用Matlab软件按照特征参数值进行公式编写。先利用**SPSS**软进行主因子分析（这个方法就是根据数据的均值和标准差进行标准化，让处理后的数据符合正态分布），对特征参数进行降维处理。利用Matlab中改进**K-Means**的算法和遗传聚类的算法对主成分进行聚类。构建汽车行驶工况，进行分析验证。

**K-Means**算法步骤如下：

1. 将所有对象随机分配到k个非空的簇中；
2. 计算每个簇的平均值，并用该平均值代表相应的值；
3. 根据每个对象与各个簇中心的距离，分配给最近的簇；
4. 重新计算每个簇的平均值，不断重复直到满足某个准则函数或者终止条件。

常见的K-MEANS聚类算法对**初值敏感**，容易陷入**局部最优**，尤其是对于分界面模糊的样本聚类能力较差。，在初始阶段的**变异操作中采用K-MEANS变异算子**，可以加快初始阶段的搜索速度，在提高搜索速度的同时，保持种群的多样性，可以同时获得快速搜索和全局最优的双重功效。

**按照平均速度的低速、中速和高速进行划分**。

# 专业知识

## 后端

状态估计的概率解释：

* 批量的
* 渐进的

如果没有测量运动的装置，就可能没有运动方程。我们可以认为没有运动方程：类似于SFM问题，通过一组图像来恢复运动和结构。

与SFM不同的是SLAM在时间上有先后顺序（Structure From Motion），而SFM中允许使用无关的图像。

## 常见的定位方法

定位也称为位置估计，是已知地图环境的机器人位姿（进行坐标变换）。基于位置的地图是例如占用栅格地图。定位问题是以最初及运行期间可供使用信息的类型为特征的。

影响定位的因素主要分为四个方面：

* 全局|局部|最初及运行期间可供使用的信息的类型
* 环境
* 定位算法是否控制机器人运动
* 涉及的机器人的数目

大致分为三种：（局部定位与全局定位）

* 位置跟踪（初始位姿已知，高斯分布来近似位姿的不确定性）
* 全局定位（初始位姿未知，不能假定位姿误差的有界性）
* 绑架机器人问题（全局定位的一个变种，通过该方法可以测试一个定位算法，衡量该算法在全局定位失效中恢复的能力）

**静态环境与动态环境**

**对于动态环境处理方法：**

1. **状态向量里可能包括动态实体（调整马尔可夫假设）**
2. **滤除掉传感器数据（当作噪声处理）**

**被动方法与主动方法**

**马尔科夫定位**

**扩展卡尔曼滤波定位**

**估计一致性**

**解决全局定位的方法：**

**栅格定位**

**蒙特卡洛定位方法：适用于全局定位和局部定位。在重采样的过程中可能会丢失正确的位姿附近的粒子**

**置信度是以可获得的数据为条件的关于状态变量的后验概率-**

**常见的地图构建方法（参考两本神书）**

**在线SLAM问题和全SLAM 的问题。在线SLAM问题是对过去全SLAM 问题的过去位姿积分的结果。**

### 蒙特卡洛定位

用粒子表示置信度的定位算法（适用于局部定位和全局定位）

优点：比较容易实现、对于大范围的局部定位问题工作良好。

#### KLD（库尔贝克-莱布勒散度采样）采样

KLD是指两个概率分布差的测度。

采样的思想基础：基于采样的近似质量的统计界限来确定粒子数。每次粒子滤波迭代，KLD采样以概率1-&确定样本数，使得真实的后验与基于采样的近似之间的误差小于e.

#### 动态环境下的定位

概率滤波框架能够容纳的传感器噪声必须在各时间步间相互独立，但是动态引起的对传感器测量值的影响会覆盖多个时间步。

##### 处理动态环境的基本技术：

* **状态增广技术**（将隐藏的状态输入滤波器中，动态的物体数目变化、物体的速度、位置等）--》计算法复杂性高
* 异常值去除技术

## Ubuntu的常用的指令和ROS中间件RVIZ等

### Ubuntu 16.04常用的指令：

选择指令+鼠标中键 –》相当于复制

root 用户权限、

sudopasswd命令设置密码

简单命令：

安装 QQsudo dpkg -ilinux11\_1.0-preview1\_i386.deb

打开终端 Ctrl+Alt+T

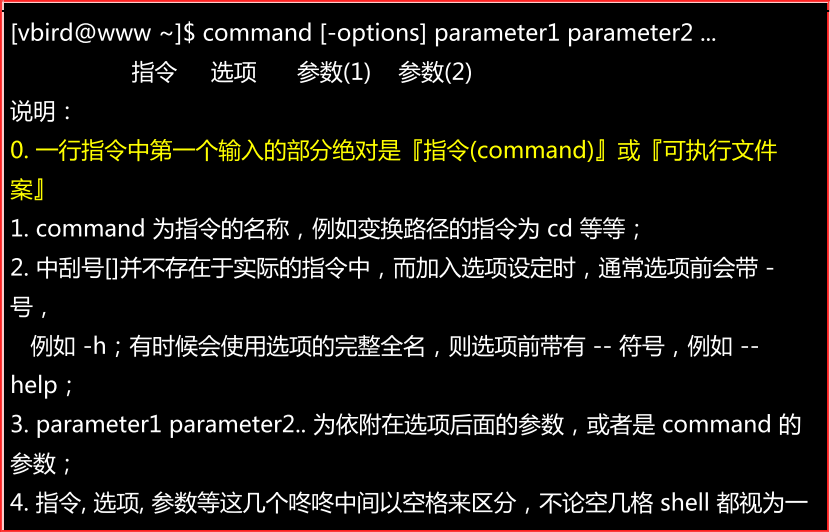
复制 ctrl+C

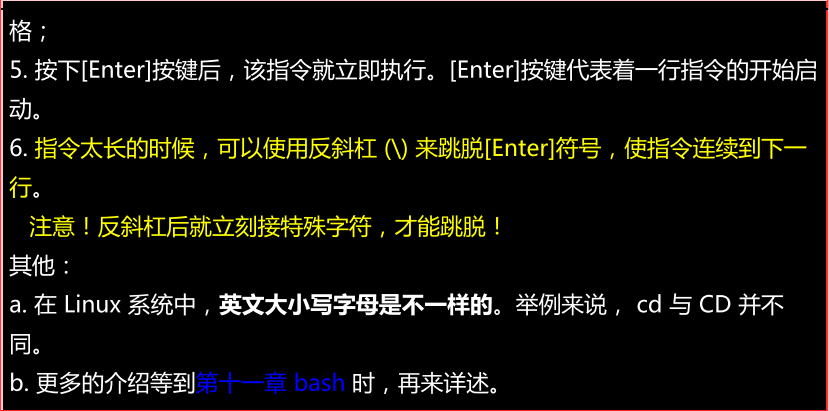
粘贴 Ctrl+shift+v

Tab键补全信息|

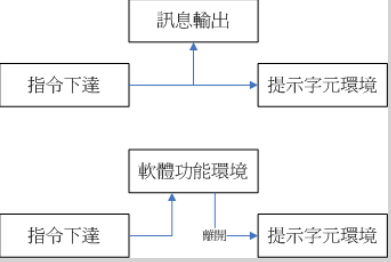
Cd 到当前目录，vim 文件名 就可以修改CmakeList文件了

|  |  |
| --- | --- |
| vim命令 |  |
| vim 文件名 | 进入文件 |
| i | 插入文件 |
| esc | 退出插入 |
| ：wq | 保存并退出 |
| ：x | 保存并退出 |
| shift+zz | 保存并退出（或者大写） |
| cat 文件名 | 查看文件内容 |
| esc : q! | 不保存退出 |
| esc : ! | 强制退出（不建议） |





|  |  |  |
| --- | --- | --- |
| 命令 | 命令格式 | 命令作用 |
| chmod | **chmod u+x topic\_publisher.py** | **增加运行权限** |
| 工作目录切换命令 | | |
| pwd | **pwd[选项]** | **显示当先用户所处的工作目录** |
| cd | **cd[目录名称]** | **切换工作路径** |
| **cd -** | **返回上一次所处的目录** |
| cd ~ | **切换至当前用户的家目录** |
| cd ~username | **切换至其他用户的家目录** |
| cd .. | **进入上级目录** |
| cd /etc | **进入/etc目录** |
| cd /bin | **进入/bin目录** |
| ls | ls[选项][文件] | **显示目录中的文件信息** |
| ls -a | **查看所有文件（包括隐藏文件）** |
| ls -l | **查看文件的而属性、大小等信息** |
| ls -d | **查看目录属性信息** |
| 文本编辑命令 | | |
| .bash文件-shell程序 | | |
|  |  |  |
| 调整显示的语系 |  |  |
| echo $LANG |  | **显示当前所支持的语系** |
| LANG=en\_US | 注意：注销Ubuntu后就没有这个指令了 | **修改语系为英文语系没有空格！！** |
| date |  | **显示日期与时间** |
| cal |  | **显示日历的指令** |
| bc |  | **简单好用的计算器** |
| scale=3（小数点保留的位数） | **quit退出** |
|  |  |  |



将消息发布到话题上：

|  |  |  |  |
| --- | --- | --- | --- |
| 普通的话题文件topic\_publisher.py | 语句名字 | 作用 |  |
| #!/usr/bin/env python | shebang | 告诉系统这是一个python文件，应该被传递一个Python解释器 | 需要使用chmod-linux指令 |
| import rospy | 出现在每一个ROS python中 | 负责导入我们将要用到的基础功能 |  |
| from std\_msgs.msg（包名.msg） import Int32 | 导入我们使用的而消息定义 |  |  |
| <depend package=”std\_msgs” /> | 在package.xml文件中添加一个依赖 |  |  |
| rospy.init\_node(‘topic\_publisher’) |  | 初始化节点 |  |
| pub = rospy.Publisher(‘counter’,Int32) |  | 赋予话题一个**名字**(counter)并说明该话题上发布的**消息类型**（Int 32） |  |
| rate = rospy.Rate(2) | 设置速率，单位是HZ，表示我们发布消息的速率 | 在话题上发布消息 | 这句话的意思是：每秒发布两次消息 |
| count = 0 |  | 计数器初始值为0 |  |
| while not rospy.is\_shutdown():  pub.publish(count)  count +=1  rate.sleep() | 如果节点已经准备好要被关闭了，则is\_shutdown()函数返回一个True,反之返回一个False | 决定是否退出循环 | rate.sleep（）让程序睡眠一会，从而保证while循环体执行频率大约为2Hz |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 订阅一个话题 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

解压 tar -xzvf DBow3

|  |  |
| --- | --- |
| Cmake |  |
| svn |  |
| make |  |
| c++编译链接 |  |

### ROS中间件

#### ROS中的roscore

向节点提供连接消息，方便节点之间相互传递信息的服务程序。

|  |  |  |
| --- | --- | --- |
| ROS指令 | 指令格式 | 指令作用 |
| roslaunch | roslaunch Package(包名) launch\_file（launch文件名是个XML文件后缀为.launch） | 用于自动启动一系列ROS节点的命令行工具 |
| Ctrl+C |  | 结束所有节点 |
| tab |  | 补全命令 |
| rosrun | rosrun PACKAGE EXECUTABLE（可执行程序） [args]向这个程序传递任何参数 | 寻找程序包中的可执行程序并且向这个程序传递任何参数 |
|  |  |  |
|  |  |  |
|  |  |  |

Tf坐标系转换

ROS命名空间和重映射

Topic（话题）是一种发布/订阅通讯机制（publish/subscribe），定义了类型的消息流。

### 可视化RVIZ

**运行指令**：Rosrun rviz rviz

关于RVIZ 界面的介绍：

**其中红色的是x轴，蓝色是Y轴，绿色是z轴**

**Grid**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | --- | | **Displays a grid along the ground plane, centered at the origin of the target frame of reference. More Information.** | | **Reference Frame**  **The TF frame this grid will use for its origin.** | |  | |  | | **Alpha** The amount of transparency to apply to the grid lines. | | **Plane** The plane to draw the grid along. | |  | |
|  | |  | | --- | | **Amount of transparency to apply to the points. Note that this is experimental and does not always look correct.** | |  | |  | |  | |  | |  | |  | |  | |

**出现的错误**

**Fixed Frame [velodyne] does not exist**

### 列举ROS的编译过程





**对于OpenCV这样的库，需要先make 一下，调用sudo 马克install将OpenCV安装在电脑上，然后编译，而且还需要添加依赖项**

**Make-j4 调用多线程进行编译（j后边的参数使用线程的数量）**

**将包安装在不同的位置**

### RVIZ的使用

安装好了以后直接ros rviz rviz就可以了

## 三维里面常用的定位方法的原理弄明白

高精定位：将车辆在线获取的环境特征与实现建好的高精度地图进行复杂的特征匹配。=，来获得当前车辆在地图的确切位置。自动驾驶车辆通过摄像头、激光雷达等传感器进行环境感知的能力有限，高精度地图上的三维语义信息如红绿灯、交通标志牌等可以缩小目标的检测范围、降低感知难度。

高精度地图是一种具有厘米级精度、包含路网结构信息的特殊地图，主要应用于高精定位、环境感知、路径规划以及仿真实验。

高精度地图的制作过程主要包括**数据采集**、**数据处理**、**语义特征提取**、**人工质检**、**发布**等过程。

## DBoW3的使用

**库安装源：**

* Github上直接下载最新的DBoW3
* 在高翔SLAM14讲中3rdparty直接有这个的包

**安装过程中遇到的问题：**

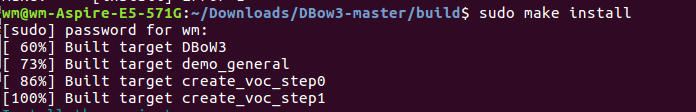
|  |
| --- |
| /home/riki/slambook/ch12/feature\_training.cpp:1:25: fatal error: DBoW3/DBoW3.h: 没有那个文件或目录  compilation terminated.  CMakeFiles/feature\_training.dir/build.make:62: recipe for target 'CMakeFiles/feature\_training.dir/feature\_training.cpp.o' failed  make[2]: \*\*\* [CMakeFiles/feature\_training.dir/feature\_training.cpp.o] Error 1  CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/feature\_training.dir/all' failed  make[1]: \*\*\* [CMakeFiles/feature\_training.dir/all] Error 2  Makefile:83: recipe for target 'all' failed  make: \*\*\* [all] Error 2 |

**解决方法：**

**在/home/xxx/slambook/ch12/CMakeLists.txt中添加：#include "DBoW3/DBoW3.h"（亲测无效）🡪之后出现的报错：**

|  |
| --- |
| [ 16%] Building CXX object CMakeFiles/feature\_training.dir/feature\_training.cpp.o  /home/riki/slambook/ch12/feature\_training.cpp:1:25: fatal error: DBoW3/DBoW3.h: 没有那个文件或目录  compilation terminated.  CMakeFiles/feature\_training.dir/build.make:62: recipe for target 'CMakeFiles/feature\_training.dir/feature\_training.cpp.o' failed  make[2]: \*\*\* [CMakeFiles/feature\_training.dir/feature\_training.cpp.o] Error 1  CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/feature\_training.dir/all' failed  make[1]: \*\*\* [CMakeFiles/feature\_training.dir/all] Error 2  Makefile:83: recipe for target 'all' failed  make: \*\*\* [all] Error 2 |

**解决方法：进入**/usr/local/include后根本没有发现DBow3所以在安装DBow3时，在进行sudo make install即可。



亲测依旧报错—》》》》》》》

|  |
| --- |
| **[ 16%] Building CXX object CMakeFiles/feature\_training.dir/feature\_training.cpp.o**  **make[2]: \*\*\* No rule to make target '/usr/local/lib/libDBoW3.a', needed by 'feature\_training'。 停止。**  **CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/feature\_training.dir/all' failed**  **make[1]: \*\*\* [CMakeFiles/feature\_training.dir/all] Error 2**  **Makefile:83: recipe for target 'all' failed**  **make: \*\*\* [all] Error 2** |

**去掉Cmakelists里面这句话：set( DBoW3\_LIBS "/usr/local/lib/libDBoW3.a" )**

**报错：**

|  |
| --- |
| **[ 16%] Linking CXX executable feature\_training**  **CMakeFiles/feature\_training.dir/feature\_training.cpp.o：在函数‘main’中：**  **feature\_training.cpp:(.text.startup+0x90a)：对‘DBoW3::Vocabulary::Vocabulary(int, int, DBoW3::WeightingType, DBoW3::ScoringType)’未定义的引用**  **feature\_training.cpp:(.text.startup+0x919)：对‘DBoW3::Vocabulary::create(std::vector<cv::Mat, std::allocator<cv::Mat> > const&)’未定义的引用**  **feature\_training.cpp:(.text.startup+0x93a)：对‘DBoW3::operator<<(std::ostream&, DBoW3::Vocabulary const&)’未定义的引用**  **feature\_training.cpp:(.text.startup+0x9c7)：对‘DBoW3::Vocabulary::save(std::\_\_cxx11::basic\_string<char, std::char\_traits<char>, std::allocator<char> > const&, bool) const’未定义的引用**  **feature\_training.cpp:(.text.startup+0xa06)：对‘DBoW3::Vocabulary::~Vocabulary()’未定义的引用**  **feature\_training.cpp:(.text.startup+0xc4d)：对‘DBoW3::Vocabulary::~Vocabulary()’未定义的引用**  **collect2: error: ld returned 1 exit status**  **CMakeFiles/feature\_training.dir/build.make:112: recipe for target 'feature\_training' failed**  **make[2]: \*\*\* [feature\_training] Error 1**  **CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/feature\_training.dir/all' failed**  **make[1]: \*\*\* [CMakeFiles/feature\_training.dir/all] Error 2**  **Makefile:83: recipe for target 'all' failed**  **make: \*\*\* [all] Error 2** |

**然后我进行了神操作，将动态库和静态库的名字改了一下。**

## 三维空间刚体运动

三维空间的刚体运动描述方式：**旋转矩阵R**、**变换矩阵T**、四元数、欧拉角。

位于空间中（x,y,z）处，并且朝向方向。

坐标系的欧式变换—》设定一个惯性坐标系（世界坐标系）、相机坐标系、变换矩阵T

**欧式变换**🡪刚体运动，保证同一个向量在各个坐标系下的长度和夹角都不会发生变化。

旋转矩阵的逆代表的是相反的旋转。

旋转向量和欧拉角：

任何旋转都可以用一个旋转轴和一个旋转角来刻画。于是可以使用一个向量，其方向与旋转轴一致，长度等于旋转角。（这种向量成为旋转向量或轴角）--只需要一个三维向量就可以描述旋转。对于变换矩阵，使用一个旋转向量和一个平移向量既可以表达一次变换。维数为6维的。旋转向量就是李代数。

**罗德里格斯公式**

**欧拉角：常用的rpy角的旋转顺序是ZYX，代表分别绕着X、Y、Z轴旋转。**

* **欧拉角的重大缺点：万向锁的问题导致欧拉角不适合插值和迭代，往往只应用于人机交互中。**

**四元数**

|  |  |  |
| --- | --- | --- |
| **表示方法** | **优点** | **缺点** |
| **欧拉角** |  |  |
| **旋转矩阵** |  |  |
| **变换矩阵** |  |  |
| **角轴** |  |  |
| **四元数** |  |  |

## 李群和李代数

相似变换群Sim(3)🡪一般为单目相机下的尺度不确定性造成的。对于空间中的点，在相机坐标系下要经过一个**相似变换**，而非欧式变换。将尺度因子s标记出来。

## Eigen库的使用

是一个C++ 的库，类似于矩阵运算的Matlab

学习资料包括Eigen（C++）的中文文档

* **Eigen库知乎翻译中文文档：**<https://zhuanlan.zhihu.com/p/87613088>
* **官方网站** ：<http://eigen.tuxfamily.org/index.php?title=Main_Page>
* **Eigen官方文档：**<http://eigen.tuxfamily.org/dox/modules.html>
* **印象笔记有备份**
* **Ubuntu中自带的Eigen安装方法：sudo apt-get install libeigen3-dev**
* **下载Eigen3.3.7的地址：**https://gitlab.com/libeigen/eigen#

### Eigen库需要注意的细节部分：

1. 矩阵和向量在Eigen中不同，不能混合两种不同类型的矩阵，这样是错误的。Eigen矩阵不支持自动类型提升例如

|  |
| --- |
| **Eigen::Matrix<double, 2, 1> result\_wrong\_type = matrix\_23 \* v\_3d;** |

1. 不能搞错矩阵的维度否则会报错

|  |
| --- |
| **YOU\_MIXED\_MATRICES\_OF\_DIFFERENT\_SIZES’ is not a member of ‘Eigen::internal::static\_assertion<false>’** |

1. 注意断点调试的含义是什么：

Cmakelist文件中 需要将release改写成debug就可以进行断点调试了。-03是优化选项。



**装好以后放在什么位置，一般像头文件的库一般会装在/usr/include/eigen3/**

**查看内容**

|  |
| --- |
| **less /usr/include/eigen3/Eigen/Core 查看内容** |

**编译后不会出现二进制文件，不需要link 不需要link，只需要头文件就可以了。**

## Sophus库的使用

**需要学习的知识点**：

* □SLAM 视觉十四讲中的李群和李代数部分内容（包括课后习题、pdf）、
* □高翔深蓝学院视频、（包括课后习题、ppt）
* □Sophus的 usesophus.cpp代码的理解（结合之前的课本进行理解和注释）
* https://github.com/strasdat/ScaViSLAM

### 1.Sophus库的介绍

**Github**上的李群和李代数的库。早期的Sophus只是普通的sophus（可以从slambook(视觉SLAM14讲中的代码得到/3rdparty部分，包括.h文件和.cpp文件)），后来将sophus改编成了模板类（hpp形式给出，只有头文件的库，没有cpp文件，不需要编译的）

早期的Sophus需要编译，最近更新的Sophus不需要编译，直接引入就可以了。

最近更新的sophus的链接：<https://github.com/strasdat/Sophus>

早期的sophus的链接：<https://github.com/TristaClover/slambook>

### 2.Sophus库的编译过程：

|  |
| --- |
| **编译过程**： |
| cd 3rdparty/Sophus/sophus/ |
| cd .. |
| ls |
| mkdir build |
| cd build |
| cmake .. |
| make |

如果用的是新的版本：

将usesophus.cpp文件里面进行更改：

#include “sophus/so3.h”改成#include “souphus/so3.hpp”

#include “sophus/se3.h”改成#include “souphus/se3.hpp”

指明sophus/so3d还是sophus/so3f

使用这个库需要的知识点包括SLAM 视觉十四讲中的李群和李代数部分内容、高翔深蓝学院视频、sophus的 usesophus.cpp代码的理解（结合之前的课本进行理解和注释+深蓝学院的课后习题和李群和李代数的课后习题部分都要弄明白）

## G2O库的求解

Github上直接搜索G2O然后下载就可以了，对C++11进行了改变。用的是图模型来进行优化的

遇到的G2O问题：

G2O 全称（general graphic optimization,G2O）图优化理论🡪图用顶点表示优化的变量，用边表示误差项（对于非线性最小二乘的问题，可以构建一个图）如果以激光SLAM为例，图优化的节点就是激光雷达的位姿

## VIO

* 自动驾驶通常使用**IMU+GPS/差分**  **GPS/RTK的融合方案**，形成**GNSS-INS**组合导航系统，达到厘米组定位精度。
* 头戴式**AR/VR头盔**多使用**视觉+IMU**的VIO定位系统，形成**高帧率定位方案**。

### 1.紧耦合和松耦合

* **紧耦合**是融合本身会影响视觉和IMU中的参数（如IMU的零偏和视觉的尺度等。）典型方案是**MSCKF**和**非线性优化**。
* **松耦合**是将IMU定位与视觉/GNSS的位姿直接进行融合，融合过程对二者本身不产生影响，**作为后处理方式输出**。典型的方案是**卡尔曼滤波器**。
* **为什么要使用紧耦合？**

1. **单凭单目视觉或者IMU不具备估计Pose的能力：视觉存在尺度不确定性问题IMU存在零偏导致漂移。**
2. **松耦合中，视觉内部BA没有IMU的信息，在整体层面来看不是最优的。**
3. **紧耦合可以一次性建模所有的运动和测量信息，更容易达到最优。**

## Cere Solver库的求解

### 安装：

Ceres solver.org/installation.html进行安装<http://www.ceres-solver.org/installation.html>

http：//github.com/search?q=ceres

git clone <https://ceres-solver.googlesource.com/ceres-solver>

slambook包里面也提供了ceres

## Opencv库的使用

### 安装方式有两种：

1. 直接在Ubuntu16.04下安装  **sudo apt-get install libopencv-dev**  （预编译的版本）|装到系统目录下面（2.几的版本）
2. 3.0以上的，在opencv 的官网进行安装：建议下载最新的，因为库通常是**用新不用旧**。安装下载的官网为[**https://opencv.org/releases/**](https://opencv.org/releases/)
3. 分为两个不同的版本，我用的是3.4.8（2019-11-12）最新的应该是4.1.2（2019-11-12）特征和区别在哪里
4. 下载source源代码

### 学习教程：

1. <http://opencv.org/books.html>
2. <https://docs.opencv.org/master/d9/df8/tutorial_root.html>学习教程
3. https;//docs.opencv.org/
4. 谷歌 搜索opencv教程
5. 看opencv的时候可以直接在google里搜索opencv的函数名 这样就可以直接跳到这个函数的教程里 如：opencv imread
6. Vim cmake 可以直接修改CMAKElist.txt文件，另外在find\_package（opencv 3.0以上的版本号）

Opencv的contrib比较好用（第三方很好用的库）

学会搜函数的文档和函数的用法。

### 编译方式（常用的）：

|  |
| --- |
| 解压 |
| ls |
| 当前的文件夹下 |
| mkdir build |
| cmake .. |
| make |

**编译和使用过程中出现的问题和解决的方法**：

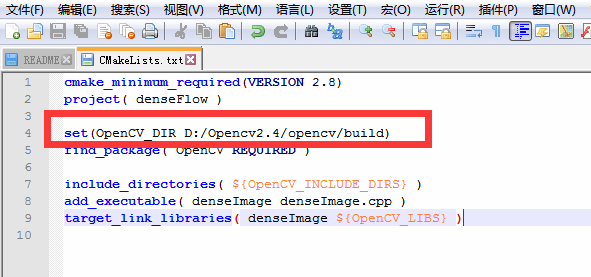
**报错原因：**

|  |
| --- |
| **CMake Error at CMakeLists.txt:3 (find\_package):**  **By not providing "FindOpenCV.cmake" in CMAKE\_MODULE\_PATH this project has**  **asked CMake to find a package configuration file provided by "OpenCV", but**  **CMake did not find one.**  **Could not find a package configuration file provided by "OpenCV" with any**  **of the following names:**  **OpenCVConfig.cmake**  **opencv-config.cmake**  **Add the installation prefix of "OpenCV" to CMAKE\_PREFIX\_PATH or set**  **"OpenCV\_DIR" to a directory containing one of the above files. If "OpenCV"**  **provides a separate development package or SDK, be sure it has been**  **installed.** |

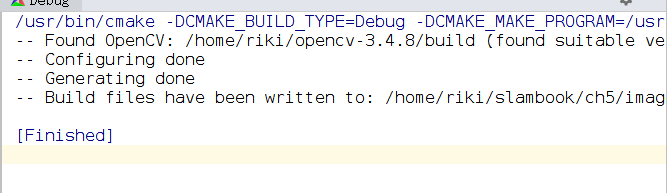
**解决方法**

原文链接：<https://blog.csdn.net/kz2313456/article/details/86481165>

打开你的CMakeLists.txt，添加set(OpenCV\_DIR D:/Opencv2.4/opencv/build) 注意这是你的OpenCVConfig.cmake的路径（**注意路径中的斜杠方向**）



再编译就可以啦



## PCL架构图

1. 官方网站：<http://pointclouds.org/>
2. 安装方式主要分为两种：从预编译包安装和从源代码安装。在Ubuntu16.04以后就直接在里面自带的仓库里面了。可以通过Sudo apt-get install libpcl-dev装上1.7版本的PCL。

安装pcl的工具 ：

Sudo apt-get update

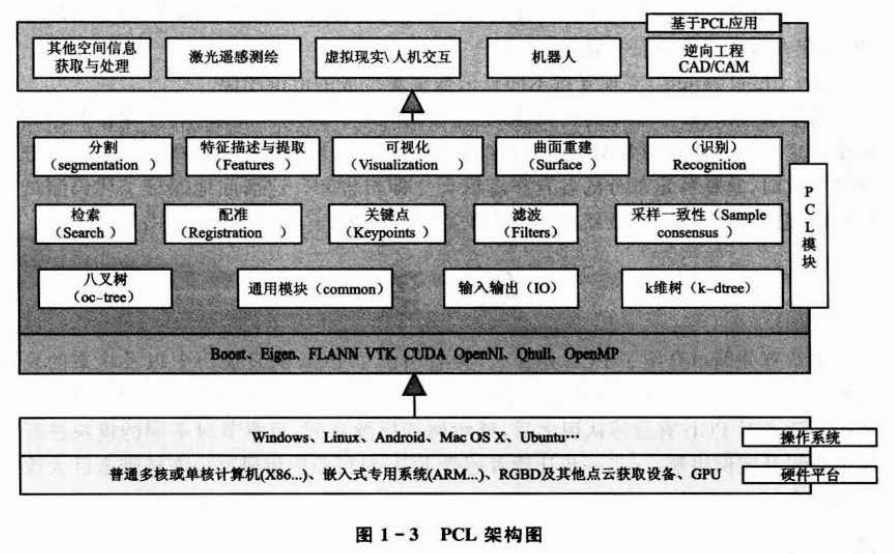
sudo apt-get install libpcl-dev pcl-all

sudo apt-get install libpcl-dev pcl-tools

1804很好 1604好像还有bug 在Google中搜索解决

**点云的拼接过程**：

|  |  |
| --- | --- |
| cd ../joinMap |  |
| ls |  |
| rm-rf build |  |
| mkdir build |  |
| cmake .. |  |
| make |  |
| ./build/joinMap |  |
| ls |  |
| 将图像转成map.pcd文件 |  |
| pcl\_viewer map.pcd | 打开地图文件 |



图片来自点云库PCL学习教程

Boost 、 Eigen 、 FLANN 、 VTK 、 CUDA 、 Open NI 、 QHull，实现点云相关的获取、滤波、分割、配准、检索、特征提取、识别、追 踪、曲面重建、可视化等。

第三方库有很多：

**FLANN**：（Fast Library for Approximate Nearest Neighbors）K近邻搜索的操作

**Boost：**PCL所有的模块和算法都是通过boost共享指针来进行传送数据的，因而避免了多次复制系统中已存在的数据的需要，从 0.6 版本开始，**PCL** 就已经被移入到Windows 、 Mac OS 和 **Linux 系统**

**一个处理管道的基本接口程序：**

1. 创建处理对象
2. 使用setInputCloud通过输入点云数据，处理模块。
3. 设置算法相关参数
4. 调用计算，得到输出

**PCL** 是指纳入了多种操作点云数据的三维处理算法**，**其中包括 **过滤、特征估计、表面重建、模型拟合和分割、定位搜索等。**

**PCL c++编程规范**

|  |  |
| --- | --- |
| **文件命名** |  |
| **文件名单词之间用下划线隔开** | **unordered\_map.hpp** |
| **头文件的扩展名** | **.h** |
| **模板类实现文件的扩展名** | **.hpp** |
| **源文件的扩展名** | **.cpp** |
| **目录命名** |  |
| **头文件都应****放在源码目录中的include/下** |  |
| **模板类实现文件放在源码目录中的include/impl/下** |  |
| **源文件都应该放在src/下** |  |
| **Include语句** |  |
| **#include在同一个目录下用双引号，其他情况用<>** | **#include** |
| **宏定义大写格式，为头文件所定义的宏后面要加上下划线** |  |
| **命名空间：多于一个单词的，单词之间应该用下划线连接** |  |
| **类/结构命名** | **驼峰命名|如果包含缩写，缩写部分全部大写。类名和结构名最好是名词组成的名字** |
| **函数和成员函数命名** |  |
| **变量命名** |  |
| **常量命名** |  |
| **迭代子变量命名** | **名字应该大写** |
| **常量命名** | **常量的名字** |
| **成员变量命名** | **类的成员变量命名单词之间用下划线隔开并且以下划线结尾。** |
| **Return语句** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### PCL点云库的公共字段

点云类实际上是一个点云容器，这个容器里包含了所有点有嗲云需要的公共信息，而不管点是什么类型。

|  |  |  |
| --- | --- | --- |
|  | 字段 | 功能 |
| header | pcl:PCLHeader | 指定点云的获取时间 |
|  |  |  |
| points | std::vector<PointT,…> | 存储所有点的容器。其中vector定义的PointT对应于类的模板参数，即点的类型 |
| is\_dense |  | 指定点云中是否有无效值 |
|  |  |  |
|  |  |  |
|  |  |  |

**不同的点云类型pcl::PointCloud类的模板参数**

|  |  |
| --- | --- |
| PointT类型 | 描述 |
| pcl::PointXYZ |  |
| pcl::PointXYZI | 包含一个描述点亮度intensity的字段 |
| pcl::PointWithRange | 有一个字段用来存储距离（视点到采样点） |
| pcl::InterestPoint | 有一个字段用来存储强度 |
| pcl::PointXYZRGBA | 存储颜色RGB和透明度A（Alpha） |
| **pcl::PointNormal** | 曲面上给定点处的法线以及测量的曲率也包括XYZ 编译类型为PointXYZRGBNormal或者PointXYZINormal |
| **pcl::Normal（最常用的点的类型）** | 曲面上给定点处的法线以及测量的曲率 |

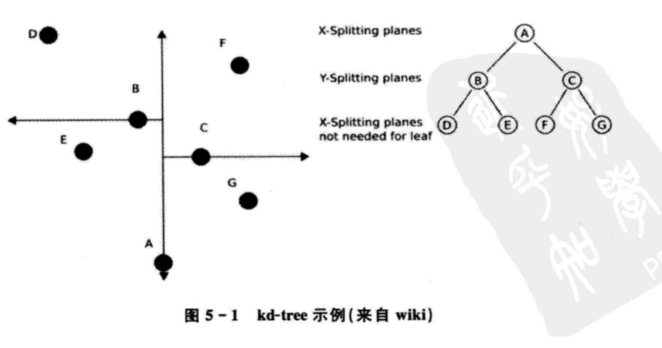
**PCL中的算法**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

### 常见的点云空间索引方法KD-tree

#### KD-Tree

**kd-tree** 或者 k 维树是计算机科学中使用的一种数据结构, 它是一种带有其他约束条件的二分查找树。kd-tree 对于区间和近邻搜索十分有用。

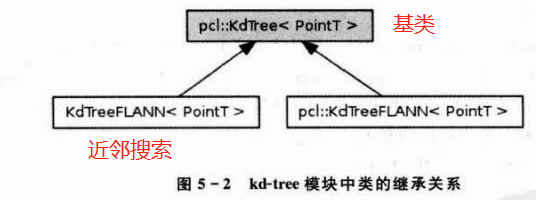


建立 **kd-tree** 最高效的方法是，像快速分类一样使用**分割法**，把**指定维度的值放在根上**，在该维度上包含较小数值的在左子树，较大的在右子树。然后分别在左边和右边的子树上重复这个过程，**直到用户准备分类的最后一个树仅仅由一个元素组成**。

#### PCL中的kd-tree模块及类介绍

**依赖于pcl\_common模块**

* **class pcl::KdTree<PointT>**



**ROS下安装terminator的指令：**

**Terminal 详见印象笔记ros下的terminator的安装**

**常见的linux 常用指令查找—详见印象笔记—Linux指令的压缩指令**

|  |  |
| --- | --- |
| **指令** | **命令作用** |
| **zip -r <压缩后的文件名><准备压缩的文减名>** |  |
| zip -r abcdef.zip abc def.txt  zip -r <压缩后的文件名><待压缩文件名1><带压缩文件名2> | 将文件夹abc和文件def.txt压缩成一个压缩包abcdef.zip |
|  |  |

## MATLAB\Simukink的使用

### Matlab常用指令|Simulink 常用模块

### SIMULINK自动驾驶工具箱

## CarSim 和PanoSim的使用

## dSPACE快速原型系统

#### MicroAutoBox的使用步骤和作用、联合调试

## C++基础

## 汽车系统动力学

## 现代控制理论的那本书

|  |
| --- |
| roslaunch loam\_velodyne loam\_velodyne.launch |
| rosbag record -o out /velodyne\_points |
| rosbag play nsh\_indoor\_outdoor.bag |
| 最后会记录一个名为out\_(记录时间)的.bag包 |
| 然后将.bag包转化为 .pcd文件 |
| rosrun pcl\_ros bag\_to\_pcd  out\_.bag(上面记录的那个包) /cloud  pcd |
| 此时转化为一个名为cloud的pcd集，进入pcd那个文件夹 |
| 用pcl\_viewer last.pcd（最后一帧pcd）然后就看到了最终的结果 |