１．安装了ROS，探索了消息，服务的用法，总结一下：

　　　　消息，在包下面的src文件夹创建x.msg，内容自定

1 打开包的 package.xml,添加如下两行,如果已经存在则不用重复添加:

<build\_depend>message\_generation</build\_depend>

<exec\_depend>message\_runtime</exec\_depend>

删掉#，去除对下边语句的注释:

add\_service\_files(

FILES

Service1.srv

Service2.srv

)

generate\_messages(

DEPENDENCIES

std\_msgs

)

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

message\_generation（添加这句话）

)

catkin\_package(

CATKIN\_DEPENDS message\_runtime

)

还有就是CMakelist.txt的修改

cmake\_minimum\_required(VERSION 2.8.3)

project(beginner\_tutorials)

## Find catkin and any catkin packages

find\_package(catkin REQUIRED COMPONENTS roscpp rospy std\_msgs genmsg)

## Declare ROS messages and services

add\_message\_files(FILES Num.msg)

add\_service\_files(FILES AddTwoInts.srv)

## Generate added messages and services

generate\_messages(DEPENDENCIES std\_msgs)

## Declare a catkin package

catkin\_package()

## Build talker and listener

include\_directories(include ${catkin\_INCLUDE\_DIRS})

add\_executable(talker src/talker.cpp)

target\_link\_libraries(talker ${catkin\_LIBRARIES})

add\_dependencies(talker beginner\_tutorials\_generate\_messages\_cpp)

add\_executable(listener src/listener.cpp)

target\_link\_libraries(listener ${catkin\_LIBRARIES})

add\_dependencies(listener beginner\_tutorials\_generate\_messages\_cpp)

（http://wiki.ros.org/cn/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29）

看看对应的改下

２．尝试了在KDevelop上开发ＲＯＳ（）

遇到的问题就是catkin\_make的路径没确定好，要确定在/home/wwh/catkin/cmake

还有就是选择可执行文件的路径，

举例：放在~/catkin\_ws/K\_ws/devel/lib/d\_pkg (dpkg 是包的名称)

3 自己找了一种方法，将rm 重定向为trash 这样就把文件删除到回收站里，还有一次挽救的机会，然后把真正的rm重定向为rm!这个就是真正的删除，不可挽回的了。

4vim用法; :x保存并退出;(http://blog.csdn.net/u010956473/article/details/74087608)

5安装了git,(<http://www.jb51.net/LINUXjishu/457371.html>)

(http://blog.csdn.net/tina\_ttl/article/details/51326684)

(http://blog.csdn.net/ajianyingxiaoqinghan/article/details/70544159)

教程

环境配置就不说了,参见上面链接.(git remote rm origin 删除远程仓库)

Git文件(a.txt)的流程.

首先 git add a.txt (将 a.txt进入缓冲区.使用git rm --cached a.txt 删除)

git add ./文件夹(添加文件夹的)

2.git commit -m a.txt

(或git commit -m 'add new readme file')

(这个指令可能出错,原因是没有告诉它user.name等) 方法: git config --global user.email ["youremail@example.com"](mailto:\"youremail@example.com\")

git config --global user.name "yourusername"

3.git push ori master (把它推到远程仓库上)

4.可能的问题:就是我在仓库上加了东西,本地没有更新,此时git上去会报错,得先同步;

方法:(前几步同上)

git fetch ori master

Git log -p master..ori/master

Git merge ori/master

再git push ori master

从仓库上clone:

1. git clone [gt@github.com:weihegood/ori.git(对应仓库的网址)](mailto:gt@github.com:weihegood/ori.git(对应仓库的网址))

三1.argc argv解析

argc 是 argument count的缩写，表示传入main函数的参数个数；

argv 是 argument vector的缩写，表示传入main函数的参数序列或指针，并且第一个参数argv[0]一定是程序的名称，并且包含了程序所在的完整路径，所以可以用来传路径参数所以确切的说需要我们输入的main函数的参数个数应该是argc-1个；

注: char \*argv[]“也可以替换为 “char \*\*argv“

２.

消息和服务的区别．消息是发到主题上，然后其他节点订阅这个主题，就能收到主题上的消息．属于多对多的关系，间接通信。

而服务是属于节点和节点之间的直接通信，为一对多关系。

犯了个错，在尝试一个服务节点给多个节点提供服务时，没有把client的ｎｏｄｅ名字改，导致两个节点同一个名字，出错（ ros::init(argc, argv, "client2"); client2就是节点名字）

查看软件版本： 略

３添加桌面可执行图标；

在左上角点击Ｕｂｕｎｔｕ图标，然后搜索叫“启动应用程序的”程序

进行配置，点击添加，名字自取，命令的话选择软件对应的可执行文件的路径(不知道百度)　同时加上　bash -i -c　再加个空格与路径分开，然后拖到桌面。桌面会出现一个后缀为.desktop的文件，右击属性，勾选为可执行文件。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Qt(community)是最新的Qt5.9对应的qt\_creator

1. 学习ROS下消息的记录与回放。

　相关命令：rosbag record -O x.bag(将记录消息的文件记为x.bag) /包名／消息名

Rostopic list -v(看有谁在发信息)

Rosbag play (-r 2 (两倍频率回放消息))　x.bag

Rosbag info x.bag(look the information of the news)

(注意，rosbag记录的数据有些差别，故回放时不一定达到原来的效果。)

5学习了roswtf来检测ros的错误，当build or communication出错时，可以用这个指令来看看有什么问题。（roscore得跑着）

6解决了一种问题。

/usr/lib/x86\_64-linux-gnu/libQt5Qml.so.5: version `Qt\_5' not found (required　by /opt/qt59/bin/../lib/x86\_64-linux-gnu/qtcreator/libUtils.so.4)

方法

把export LD\_LIBRARY\_PATH=/home/wwh/Qt5.9.1/5.9.1/gcc\_64/lib/写入到.bashrc中

7学习rqt\_console and rqt\_logger\_lever

Rqt\_console (原来显示节点的输出消息)

Rqt\_logger\_lever(修改日志等级，有什么用呢，就是可以获取某等级即该等级以上的日志消息，便于ｄｅｂｕｇ)

８使用roslaunch

学到一个东西，用roscd直接到达包的路径，例子：

Roscd qq(直接到了/hame/wwh/qqq/src/qq)

首先创建launch文件夹

Mkdir launch 再gedit qq.launch

接着：

<launch>

<node pkg="qq" name="temptalker" type="talker" output="screen">

</node>

<node> pkg="qq" name="templistener" type="listener" output="screen">

</node>

</launch>

看得懂吧

3.12

学习了卡尔曼滤波。

它的思想就是测量和预测之间如何取权重，从而得到一个最佳的估计。权重的计算便是其核心了，采用了一种递归的方法。

Rand()函数的使用，

例子：生成一个在０～１之间的伪随机数：

　　　Rand()%100/(double)101;

3.13学习了matlab下karman滤波器的编写；

掌握了randn(N,1)用法（生成[N,1]维度的高斯分布数组）

体会到Ｑ R的意义，（Ｑ 系统状态的噪声，即预测量的方差

R　测量噪声。　根据数据图改变他们的大小），不过还是没调的那么好。

3.14学习了粒子滤波，不过还不太明白，但是已经能以代码实现，能够理解它的基础思想，便是蒙特卡洛采样，以样本的频率来代替状态的概率。重点是理解重采样的方式。http://blog.csdn.net/qq\_27923041/article/details/56008756

<http://blog.csdn.net/heyijia0327/article/details/41122125>

以及知乎.

实验了一下，发现滤波器的效果要看采集的数据特征而定，而且，滤波器的参数也要调好，盲目的套用并不一定能得到很好的滤波效果，注意，滤波不仅是数据的处理，更多的意味着数据的估计。

3.15

学习了《概率机器人》第一，二章，书确实很好，理解了概率论的许多基础知识，不过怕忘了。贝叶斯滤波！

同时，开始学习ros的gazebo章节，进行机器人的建模仿真。

使用软链接　ln -s /weihe/good /home/weihe/good;

把/home/weihe/good/链接到/weihe/good下。

3.16 学习了urdf，利用rviz进行仿真查看。不过学得比较慢。。。

解决了一个问题，就是利用rviz观看urdf文件时，一开始是不会显示机器人模型，需要先　add robotmodel　，再在global option 下的FIxed frame 下选择机器人的连接(如base\_link)；这样就能看到模型了。

3.17加载了世界模型，第一次加载的是一个仓库，嫌复杂，自己改为厨房，如何更改呢？？

先去改/opt/ros/kinetic/share/gazebo\_ros/launch/willowgarage\_world.launch里面world value(现在我改的是1.world，原来是willowgarage.world)其实这里有两种改法，第一种是我这种，改world value另一种是到usr/share/Gazebo-7.0/worlds下用１.world替代willowgarage.wrold，（即重命名），第一种1.world就是放在这个目录下的。如果出现robot 和wrold干涉的话，可以把房子的坐标改一下。

# 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

3.23

折腾了一天，终于把C++下txt 转为数组。

涉及到文件的读取，数据类型的转换。

直接上代码　：

Ifstream infile;

Infile.open(“filename”);

While(!infile.eof()){

Infile>>laser.angle>>laser.distance;

}

Infile.close();

同时，学习了K-means聚类算法。

逻辑比较简单，就是先人为确定类的数目（这是一个缺点）；

然后在样本范围内随机确定K个中心点，计算样本到这些点的欧式距离（即评估相似性的一种标准），然后在得到的K个类中，再找出Ｋ个中心点，再以这Ｋ个中心点继续分类，以此循环。

此外，把采样到的激光雷达视觉通过ROS下的rviz显示出来，开始来做行人检测。