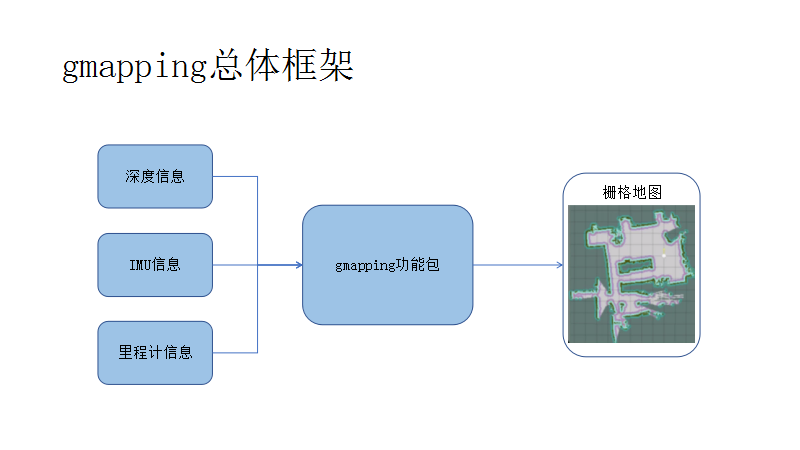
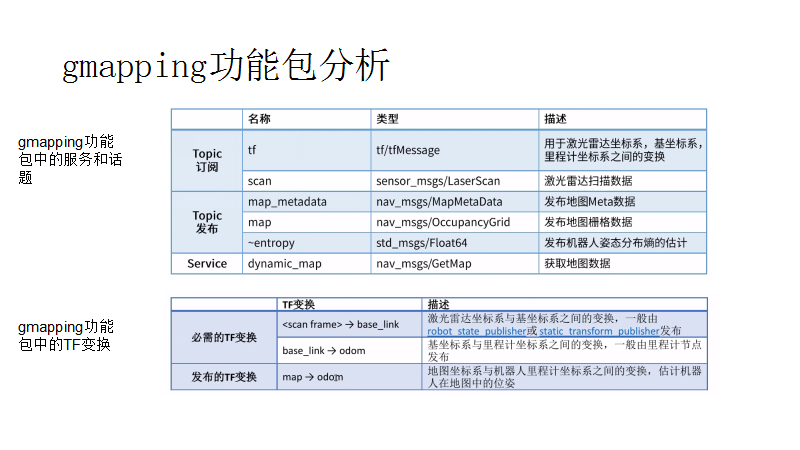


首先我们给大家介绍第一个SLAM功能包的使用方法，这个功能包的名字叫做gmapping。这是目前来说ros这种用的最成熟也是用的最多的SLAM功能包，gmapping功能包它是基于激光雷达信息的，然后使用粒子滤波的算法来创建二维栅格地图。它需要底层传感器来提供里程计信息便于机器人的定位。其实gmapping功能包的算法也是一个开源算法，这个算法来自于OpenSlam这样一个官方组织的，OpenSlam是一个官方网站，大家可以上网站去看一下，在这个网站中开源了许多种SLAM的算法，大家感兴趣的话可以去看一下算法的代码，gmapping是这种算法当中的一种，ros把这中算法提取出来以后，进行了一些封装，最终放置到了ros的社区里面去，成为了gmapping功能包。这个功能包最终的输出就是以话题nav\_msgs/OccupancyGrid形式的栅格地图。大家也可以在建图的过程当中去订阅这个地图信息，可以看到里面的数据是非常多的，会包含栅格地图当中每一个点它里面的栅格值。

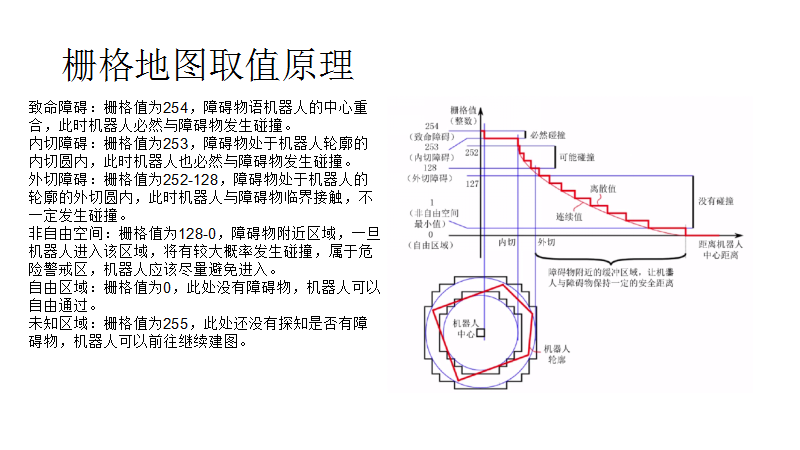


这是gmapping功能包的一个整体框架，这个功能包是需要我们三个信息的输入，这三个信息分别是深度信息，这个是必须的，无论是使用激光雷达还是使用Kinect传感器，都需要把它转换成二维的深度信息，发布给gmapping功能包。另外一个是IMU的信息，但这个信息是一个可选的信息。大家可以使用IMU配合里程计信息来达到更高精度的机器人定位。另外一个是机器人的里程计信息这个是需要大家给机器人去装配编码器或者视觉里程计或者是霍尔编码器这样一些传感器来达到里程计这样一个效果。所以这三个信息当中最重要的就是深度信息和里程计信息。那输入完成之后，我们再来看一下输出，这个功能包的输出就是一个栅格地图。这就是gmapping功能包的一个整体框架，最重要的是要弄清楚输入是什么样的，输出是什么样的。至于功能包中到底做了哪些算法处理，这并不是我们使用ros功能包中了解的重点。



关于gmapping功能包一些输入输出接口的可以来看一下，一般情况下我们需要去一个ros当中的功能包的使用方法的时候，最终的是要求关注一下功能包中的输入输出信息。这些输入输出信息主要包含的就是topic话题，输入输出服务，还有一些tf变换，还有一些param参数，这些是我们可以使用这些功能包做的一些配置选项。至于这个功能包里的具体算法，一般情况下我们是不太容易做修改的，所以我们可以把这个功能包当做一个黑盒，我们只需要知道这个黑盒的输入输出是什么样的，就可以把它给使用起来。那么关于gmapping功能包，它有订阅了几个话题，首先就是坐标系tf的变换。我们需要将激光雷达的坐标系，机器人的基坐标系以及里程计坐标系之间的一个变换，必须要发布给gmapping功能包来做处理。另外很重要的就是激光雷达信息，必须要发送给gmapping做障碍物的识别。关于这个功能包输出的话题，最重要的是map，最终发布的地图的栅格数据，同时还会发布一个机器人的姿态分布熵的估计，这个大家可以根据需要再做订阅。关于服务service，gmapping功能包有提供一个可以获取地图数据的service，只要我们去调用service，他就可以返回整个地图的数据内容。

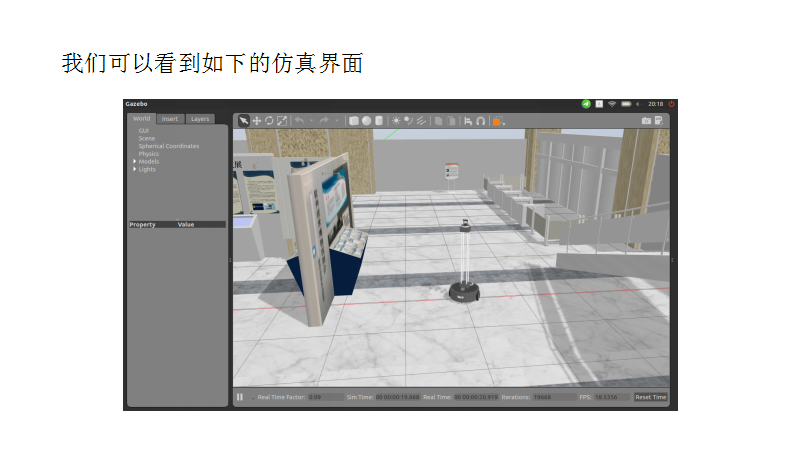
关于gmapping的坐标系变换，有以下的变换，它所需要我们给出的变换是激光雷达坐标系到整个机器人基坐标系的变换，另外就是机器人基坐标系和里程计坐标系之间的变换。gmapping它所发布的变换是地图和里程计之间的一个变换，它可以帮助我们去估计机器人在地图当中的一个位置，关于map和odom这样一个坐标系，其实初学者比较容易把它搞混了，这两个坐标系它都是全局坐标系，一般情况下，当我们机器人开机的时候，他会认为当前的位置是机器人里程计坐标系的一个零位置，然后机器人从零位置运动后，开始产生位置变化。但是我们机器人开机的位置不一定是我们map坐标系的零位置。map坐标系的零位置理论上是可以位于地图坐标系上的任意位置的，我只要搬动机器人把它放置到随意一个位置就可以了。那么这两个坐标系之间就存在着相对的坐标变换，这也就是gmapping给我们提供的坐标变换。



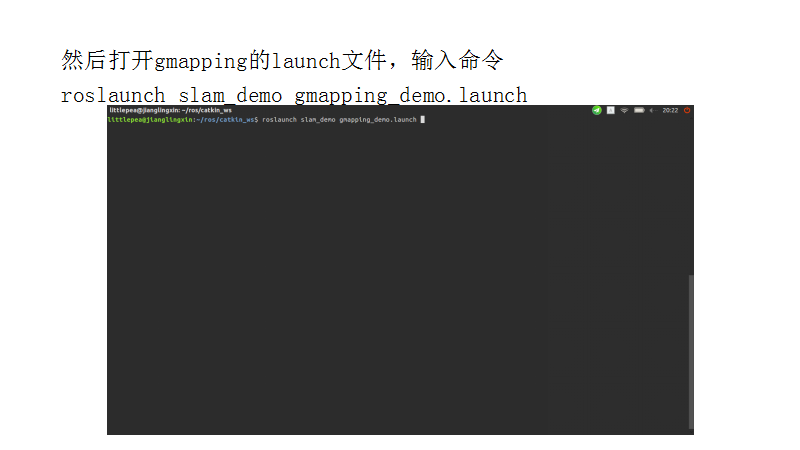
下面介绍一下栅格地图的取值原理，因为不管是gmapping功能包，还是Hector功能包，它最终创建的地图都是栅格地图这样一种形式，那栅格地图里面会有很多个栅格，每一个栅格会有具体的数值来代表栅格内是否有障碍物，那关于这个栅格的取值就会有一个取值的方法。我们可以看左边这张图上有一个红色的曲线，这条曲线代表的就是栅格的取值规则，大家可以看到横轴是障碍物和机器人中心的距离，而纵轴就是栅格的取值，然后从左边可以看到栅格具体取值的一个原则，假设我们的机器人上面装了一个激光传感器可以感知到机器人和障碍物的一个距离关系，如果我们发现障碍物和机器人中心的距离是0，也就是说障碍物和机器人中心重合的话，那说明机器人一定和障碍物发生了碰撞，此时我们给栅格的一个取值是254这个值。那这个栅格是253的话，代表障碍物处于机器人内切圆里面，红色的方框的机器人的外形，也说明障碍物和机器人是重合的。而如果障碍物是在外切圆以内，内切圆以外的话，那这样说明机器人和障碍物时有可能发生碰撞的，然后根据概率，给外切圆在252-128之间弄一个取值。如果是252的话，说明发生碰撞的概率是很大的，可能就在内部的区域，如果是128的话，可能是在外围，碰撞概率不会非常大。然后还有一个0-128的取值范围，就是一个缓冲区域，发生碰撞的概率不大，但是进入到这一区域后就要小心，机器人应避免进入这一区域。如果机器人和障碍物之间的区域是大于缓冲区域的距离时，这就说明障碍物和机器人有很大的距离，这时栅格的值就是0，表面此处没有障碍物，机器人可以自由通过。另外如果机器人它不知道这个栅格有没有障碍物或者是没有经过SLAM建图探索过的区域，它的栅格的取值全部都是255，就代表这一区域是未知区域，需要机器人去探索这一区域。以上就是栅格地图的取值原理，大家只要去记住基本的原则就是，如果这个栅格的值越大，就说明这个栅格是障碍物的可能性就越大



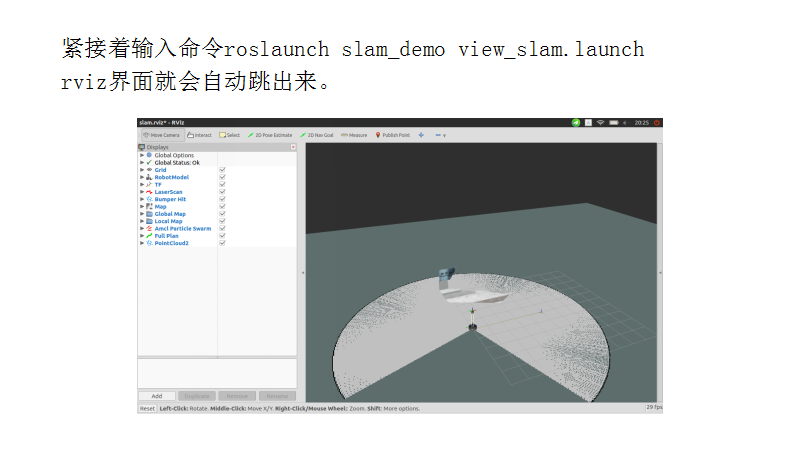
下面我们就在自己建立的仿真环境中运行gmapping功能包。gmapping功能包可以在GitHub网站下载或者课程资源中拷贝到当前的工作目录，编译成功后。首先打开之前建立的gazebo仿真环境，输入命令roslaunch robot\_sim\_demo robot\_spawn.lacunch



就可以打开如图所示的界面，我们可以看到xbot以及中科院软件所的环境。



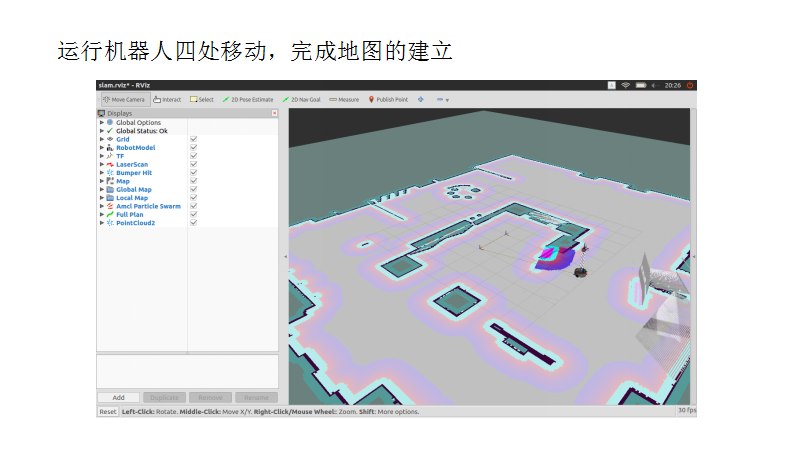
接着就是打开gmapping功能包的launch文件，输入命令roslaunch slam\_demo gmapping\_slam.launch。



接着输入命令roslaunch slam\_demo view\_slam.launch这样一个rviz的界面会自动弹出来，我们可以看到xbot的模型就在其中，另外浅灰色区域就是雷达已经探测出来的，而深灰色的区域则是还未探测的区域。



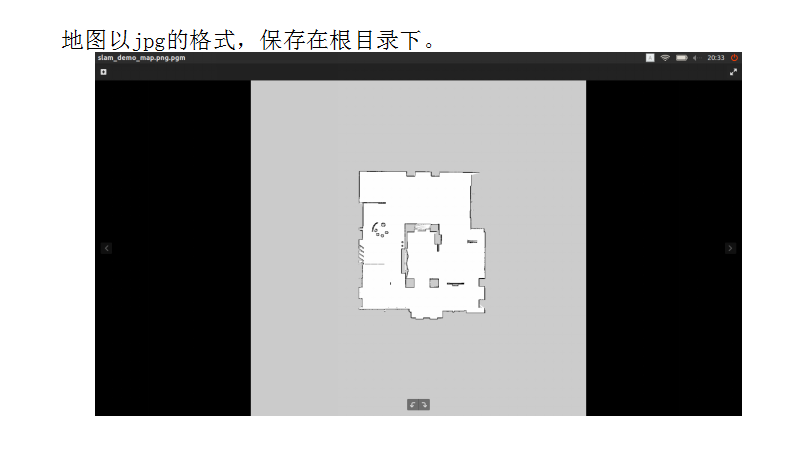
为了使机器人获得更多更加完整的地图信息，需要机器人在仿真环境中四处走动。因此打开键盘控制文件，即输入命令rosrun robot\_sim\_demo robot\_keyboard\_teleop.py。我们可以看到gazebo中的模型小车在我们的控制下运动，而rviz界面中的小车也随着控制在运动，并且浅灰色的范围越来越大。



在控制小车跑完整个地图之后，我们可以看到仿真的环境基本上出现在rviz中，基本和真实的环境相差不大，到这里基本slam的图已经建立完成了。



然后使用命令rosrun map\_server map\_server map.yaml，rosrun map\_server map\_saver -f slam\_demo.png将构建的地图保存下来。

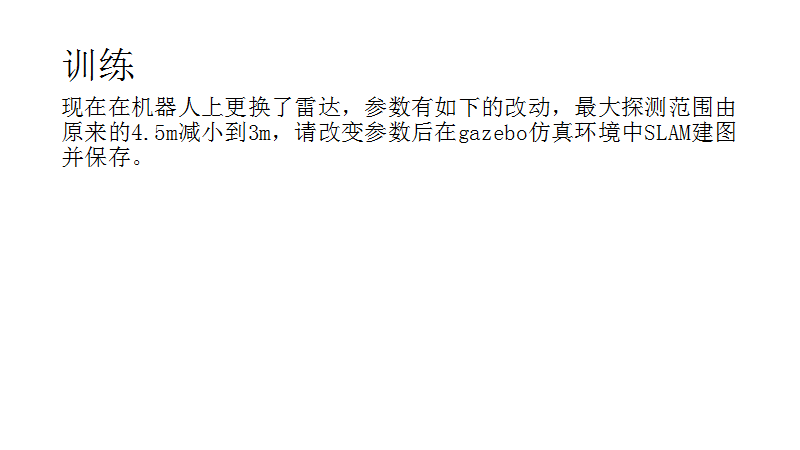


地图一般会直接保存到根目录下面，以jpg的形式，可以找到并打开看一下，我们可以对比一下建立的地图和真实的仿真环境，还是比较吻合的。



gmapping功能包里面有一个核心文件就是gmapping的launch文件，我们要运行gmapping的时候就要去启动这样一个launch文件，在launch文件中有一系列的参数需要去做配置，这些参数是我们使用功能包的一些接口，我们需要根据我们实际使用的雷达去改变gmapping的参数，从而改变建图的一个效果。从launch文件可以看到里面有非常多的一个参数，但是里面很多参数其实都不需要很深入的理解，除非大家是要去开发slam算法的话，才需要去看一下内部slam算法实现的原理，然后看完这些算法之后你就会知道这里面很多参数的意义是什么。但如果大家只是调用gmapping去跑自己的建图功能的话，其实不用去很深入地理解这些参数的，这些参数课算法是直接相关的。那这些参数我们可能是借鉴以及开发的搭载ros的机器人，像turtlebot等的参数，然后对里面比较关键的参数进行修改即可。当然如果你对这些参数很好奇，也可以去ros.wiki上查找，里面有详细的注解。这里面有几个参数可能是我们要去做修改的。第一个参数是odom\_frame,这个是我们要去订阅的里程计信息是在哪一个坐标系下的。一般情况下里程计的坐标系都是odom坐标系，这个根据你里程计坐标名的不同而改变。下面是地图更新的频率，比如说5秒钟跟新一次地图。因为我们建图的时候跟新频率非常快的话，那电脑的计算量非常大，会导致电脑运行非常卡，所以一般情况下都是5秒钟做一次更新。下面的参数依次是最大范围五米，激光雷达能检测到的最大范围是4.5米。关键是下面还有一个很重要的参数是scan这样一个参数是用来设置输入的激光雷达的话题是哪一个话题这个话题名我们通过上面的参数来配置，现在订阅的话题是scan这个话题，也就是一会我们激光雷达发布的激光数据。大家可以先用我们给的参数吧gmapping跑起来，使用这样的参数可能运行效果不是很好，与实际不搭配，但我们可以到实际情况中再做考虑。

如果是在其他机器人上使用或者使用不同参数的激光雷达传感器，只需要更改launch文件中的参数即可。如图



假设现在xbot上更新了激光雷达的，请对应参数更改程序，并在gazebo环境中slam建图，得到最后的环境地图并保存下来。