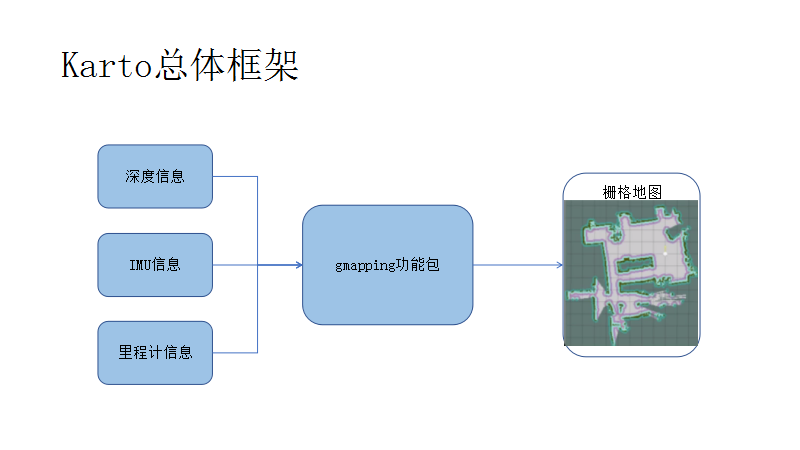


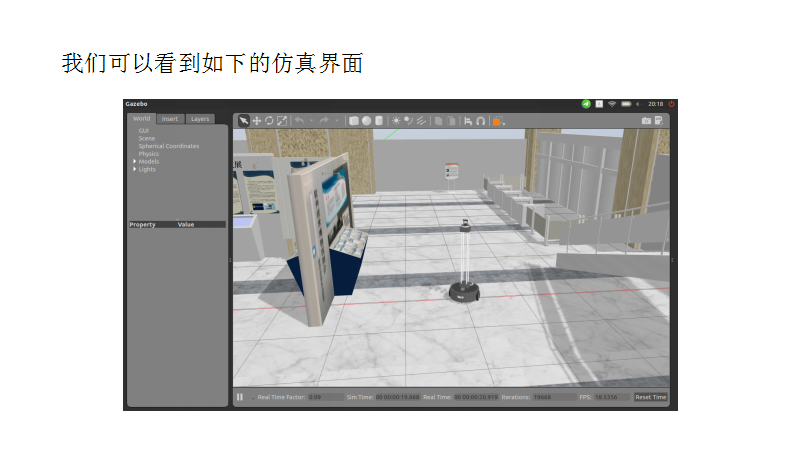
这节课我们来看另外一个SLAM算法，叫做Kartoslam，这个算法应该是比较早的一个算法了，从性能上讲，它可能要比gmapping要差一点，但是效果也还可以。Kartoslam也是基于激光雷达的深度信息来生成二维栅格地图的，它同gmapping一样需要机器人提供里程计信息，最后输出的话题保存在nav\_msgs/OccupancyGrid消息类型当中。



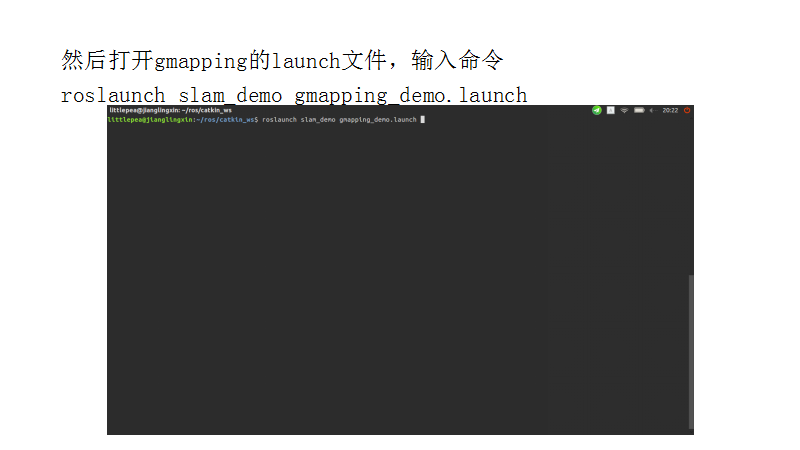
下面来看一下Karto的具体结构框架，和gmapping功能包的结构是一样的，需要输入深度信息，IMU信息以及里程计信息，最后生成一个栅格地图，唯一不同的是使用的算法原理不一样，这里我们就不去深究了。其实现在看了三种SLAM算法，我们可以发现结构都是大同小异，其实现在主流ROS的SLAM算法包都差不多，只是其中使用的SLAM算法不一样，如果以后对算法有更深入的研究，自己优化或者创新了一种算法，就可以将其中算法替换掉，输入和输出保持不变，当然这是更深层次的运用了。



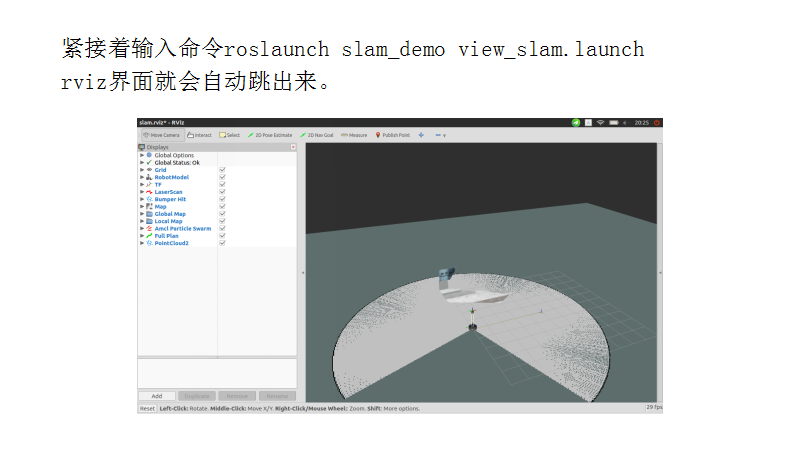
下面我们就在自己建立的仿真环境中运行gmapping功能包。gmapping功能包可以在GitHub网站下载或者课程资源中拷贝到当前的工作目录，编译成功后。首先打开之前建立的gazebo仿真环境，输入命令roslaunch robot\_sim\_demo robot\_spawn.lacunch



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。



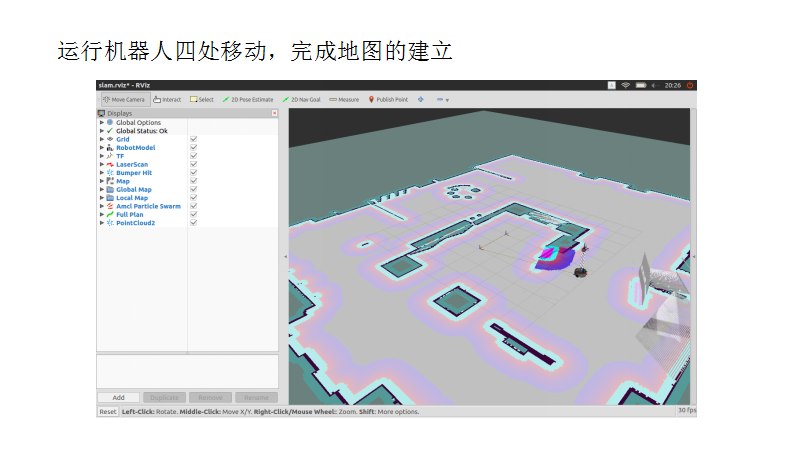
接着就是打开gmapping功能包的launch文件，输入命令roslaunch slam\_demo karto\_slam.launch。



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。



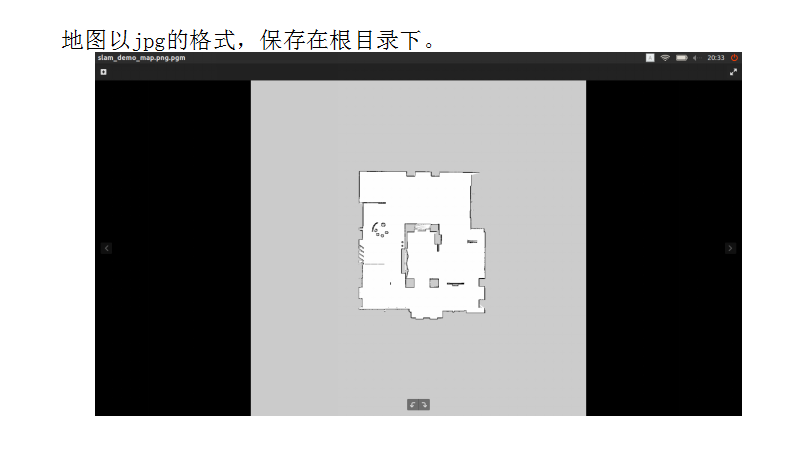
为了使机器人获得更多更加完整的地图信息，需要机器人在仿真环境中四处走动。因此打开键盘控制文件，即输入命令rosrun robot\_sim\_demo robot\_keyboard\_teleop.py。我们可以看到gazebo中的模型小车在我们的控制下运动，而rviz界面中的小车也随着控制在运动，并且浅灰色的范围越来越大。



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。



karto功能包节点的启动其实和gmapping差不多的，一样是要去启动这样一个launch文件，启动的时候也是要求配置一系列的参数，大家同样要去看一下参数里面hector节点订阅的激光雷达的话题是不是scan，最终建立的地图是在map坐标系下面的。其他的很多参数是跟算法直接相关的。如果大家对算法比较了解的话可以去调试这些参数，不是很了解的话也可以参考许多其他机器人的参数。