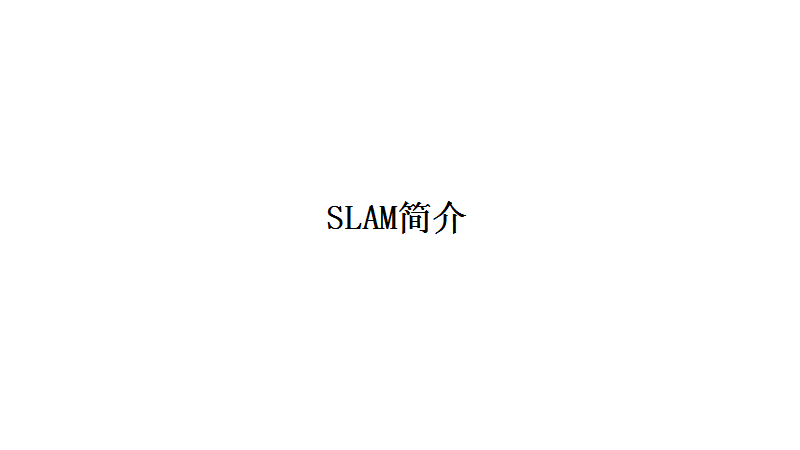
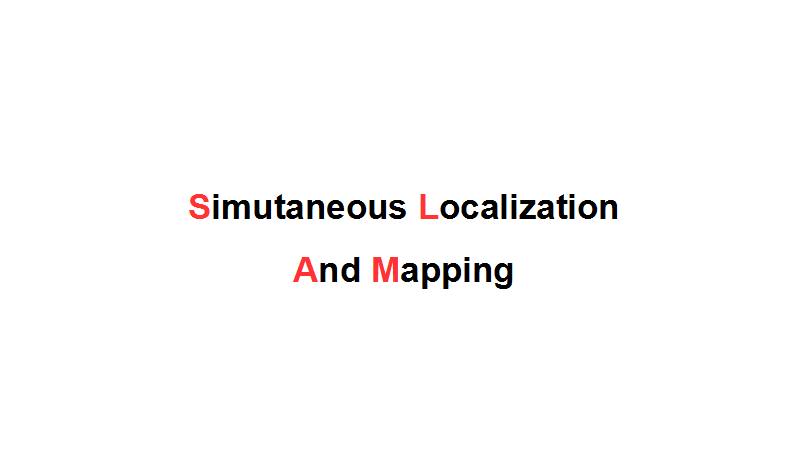
SLAM讲稿



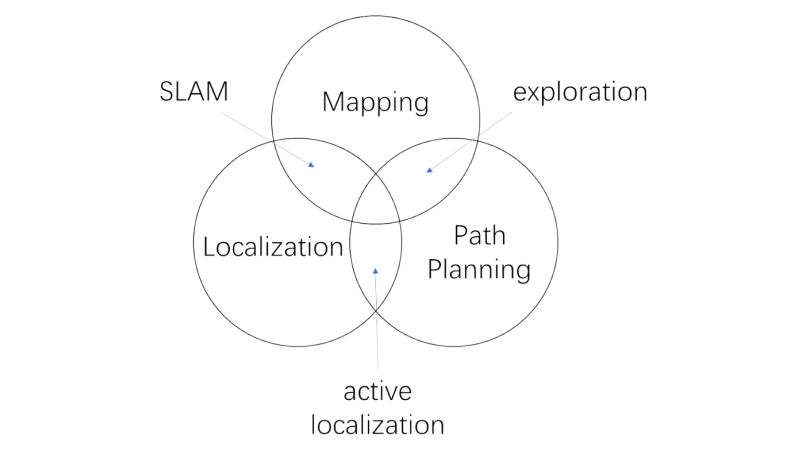
今天我们介绍ros应用的一个很大的方面，就是SLAM。相信大家用ros的目的，肯定不仅仅满足把一个机器人跑起来这么简单，主要还是基于ros进行一个开发和工程应用。ros是一个工具，SLAM是一种功能，可以用ros来实现SLAM功能。因为我们课程主要介绍ros这个工具，因此本节课讲述的重点在于SLAM这个功能是干什么的，有什么用以及如何用ros在工程上实现基本SLAM算法，而不会去介绍SLAM算法的原理。



首先，我们对SLAM做一个简单的介绍



SLAM全称是Simutaneous Location And Mapping， 用中文翻译就是同步定位和建图，这里面有两个关键任务，一个是定位，一个是建图。并且这两个任务要满足同时的条件。所以SLAM就是在未知的环境中创建地图，并且对自身进行定位。



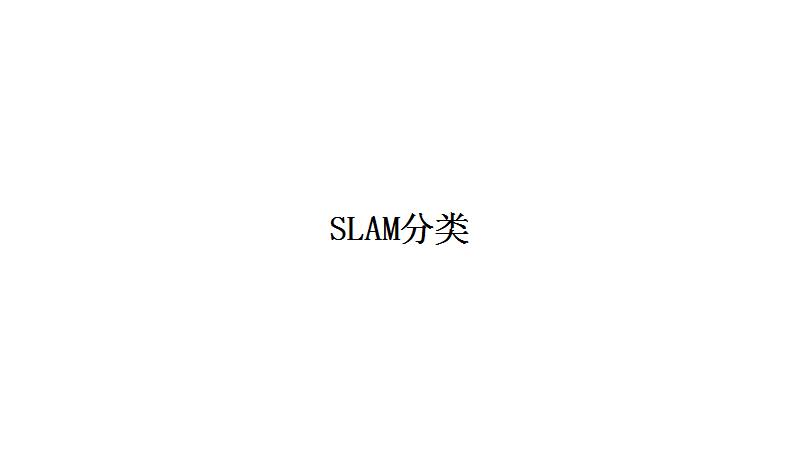
从任务上我们可以看看这张图，移动机器人他的任务通常包括建图，定位和路径规划这三个方面。而SLAM就在定位和建图的交集这里，从严格意义上来讲，SLAM是不包括路径规划的。一般我们研究SLAM问题时就假设路径规划问题时已经解决的问题。但这只是理想情况，在实际过程中还是需要考虑路径规划的问题，路径规划是我们下一章节的主题，在后面我们会详细给大家指导。针对这些问题，ros为我们提供了许多的帮助，首先ros里面有许多SLAM的开源算法包，其中最常用的是Gmapping，当然也有kartoslam，Hectorslam以及谷歌的cartographer。在定位方面呢，ros主要为我们提供了AMCL蒙氏卡罗定位，在路径规划方面呢有一个navigation，这是一个导航功能包集，里面包含了许多算法。

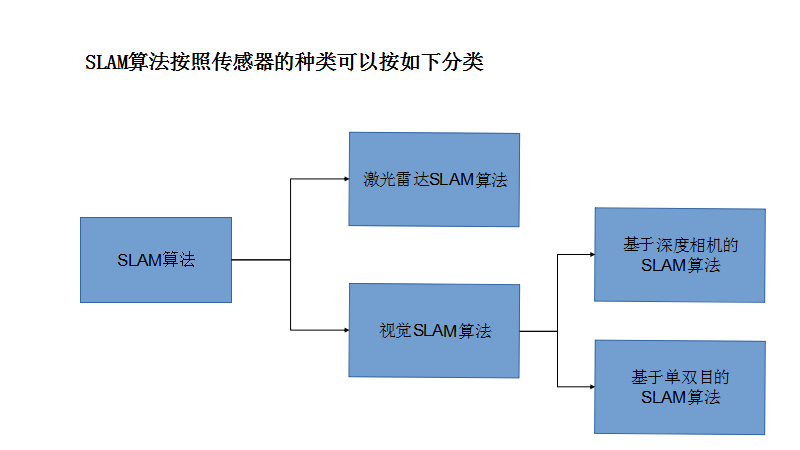


那说了这么多，在陌生环境中创建地图和定位究竟有什么用呢？



其实SLAM在当前热门的方向中都有应用：  
在VR/AR方面，根据SLAM得到地图和当前视角对叠加虚拟物体做相应渲染，这样做可以使得叠加的虚拟物体看起来比较真实，没有违和感。  
在无人机领域，可以使用SLAM构建局部地图，辅助无人机进行自主避障、规划路径。  
在无人驾驶方面，可以使用SLAM技术提供视觉里程计功能，然后跟其他的定位方式融合。  
机器人定位导航方面，SLAM可以用于生成环境的地图。基于这个地图，机器人执行路径规划、自主探索、导航等任务。

SLAM作为一种基础技术，从最早的军事用途，像核潜艇海底定位就有了SLAM的雏形。到今天，已经逐步走入人们的视野，过去几年扫地机器人的盛行让它名声大噪。可以说近几年SLAM算法也得到了飞速的发展，算法种类也渐渐丰富了起来。



SLAM算法主要按照接受环境信息的传感器不同可以分为激光雷达SLAM算法和视觉SLAM算法两大类。

激光雷达SLAM激光SLAM脱胎于早期的基于测距的定位方法(如超声和红外单点测距)。激光雷达(Light Detection And Ranging)的出现和普及使得测量更快更准，信息更丰富。激光雷达采集到的物体信息呈现出一系列分散的、具有准确角度和距离信息的点，被称为点云。通常，激光SLAM系统通过对不同时刻两片点云的匹配与比对，计算激光雷达相对运动的距离和姿态的改变，也就完成了对机器人自身的定位。

激光雷达距离测量比较准确，误差模型简单，在强光直射以外的环境中运行稳定，点云的处理也比较容易。同时，点云信息本身包含直接的几何关系，使得机器人的路径规划和导航变得直观。激光SLAM理论研究也相对成熟，落地产品更丰富。

眼睛是人类获取外界信息的主要来源。视觉SLAM也具有类似特点，它可以从环境中获取海量的信息，拥有超强的场景辨识能力。早期的视觉SLAM基于滤波理论，其计算量非常巨大。近年来，随着算法的优化革新以及相机技术、计算性能的进步，实时运行的视觉SLAM已经不再是梦想。

其中视觉SLAM算法又因为视觉传感器的不同，可以分为基于深度相机的SLAM算法和基于单双目的SLAM算法，这就不继续深入细讲了，有兴趣的同学可以课下自己去学习。



对于激光雷达SLAM和视觉SLAM可以说是优缺点分明

在适应性上，激光SLAM依据所使用的激光雷达的档次基本被分为泾渭分明的室内应用和室外应用，而VSLAM在室内外都有丰富的应用环境。VSLAM的主要挑战是光照变化，例如在室外正午和夜间的跨时间定位与地图构建，其工作稳定性不如高端室外多线激光雷达。近年来，光照模型修正和基于深度学习的高鲁棒性特征点被广泛应用于视觉SLAM的研究中，体现出良好的效果，应当说VSLAM随着这些技术的进步将会在光照变化的环境中拥有更稳定的表现。

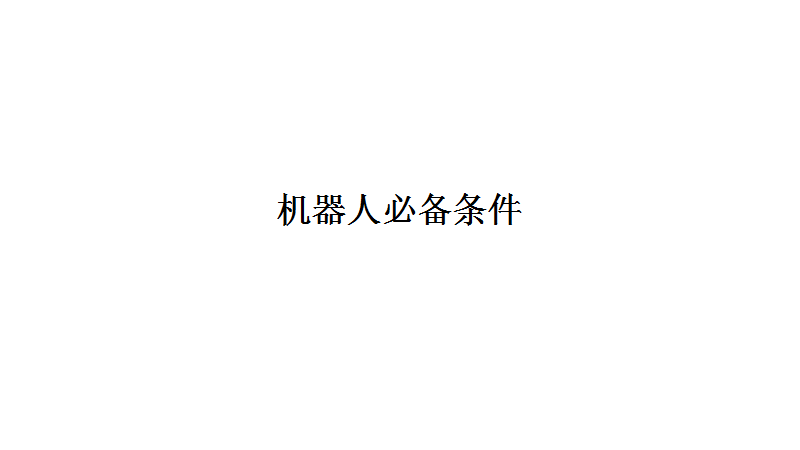
在稳定性上，激光SLAM不擅长动态环境中的定位，比如有大量人员遮挡其测量的环境，也不擅长在类似的几何环境中工作，比如在一个又长又直、两侧是墙壁的环境。由于重定位能力较差，激光SLAM在追踪丢失后很难重新回到工作状态。而视觉SLAM在无纹理环境(比如面对整洁的白墙面)，以及光照特别弱的环境中，表现较差。

在精度上，在静态且简单的环境中，激光SLAM定位总体来讲优于视觉SLAM；但在较大尺度且动态的环境中，视觉SLAM因为其具有的纹理信息，表现出更好的效果。在地图构建上，激光SLAM的特点是单点和单次测量都更精确，但地图信息量更小；视觉SLAM特别是通过三角测距计算距离的方法，在单点和单次测量精度上表现总体来讲不如激光雷达，但可以通过重复观测反复提高精度，同时拥有更丰富的地图信息。

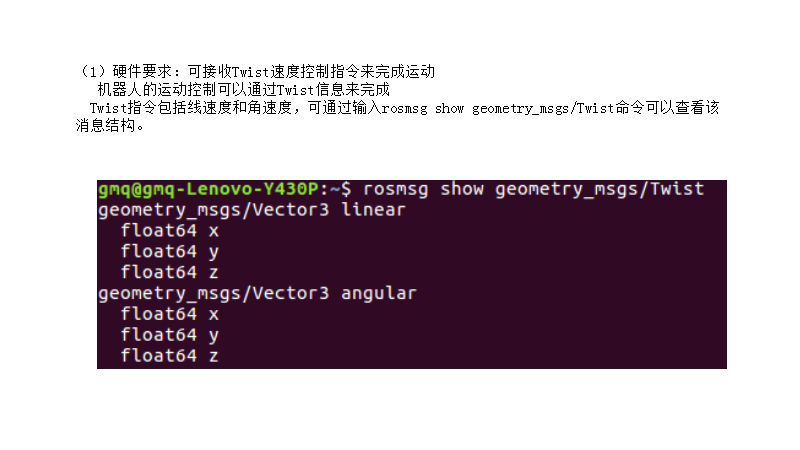
在经济性上，激光雷达事实上有许多档次，成本都高于视觉传感器。最昂贵如Velodyne的室外远距离多线雷达动辄数十万元人民币，而室外使用的高端中远距离平面雷达如SICK和Hokuyo大约在数万元人民币的等级。室内应用较广的中低端近距离平面激光雷达也需要千元级—，其价格相当于比较高端的工业级摄像头和感光芯片。激光雷达量产后成本可能会大幅下降，但能否降到同档次摄像头的水平仍有一个大大的问号。

在计算要求上，激光SLAM的计算性能需求大大低于视觉SLAM。主流的激光SLAM可以在普通ARM CPU上实时运行，而视觉SLAM基本都需要较为强劲的准桌面级CPU或者GPU支持。但业界也看到了这其中蕴藏的巨大机会，为视觉处理定制的ASICS市场已经蠢蠢欲动。一个很好的例子是Intel旗下的Movidius，他们设计了一种特殊的架构来进行图像、视频与深度神经网络的处理，在瓦级的超低功耗下达到桌面级GPU才拥有的吞吐量。DJI的精灵4系列产品就是使用这类专用芯片，实现了高速低功耗的视觉计算，为无人机避障和近地面场景导航提供根据。

可以看到ppt中的图表清晰的列出两者的优缺点，可以说激光SLAM和视觉SLAM各擅胜场，单独使用都有其局限性，而融合使用则可能具有巨大的取长补短的潜力，这也是SLAM算法发展的方向。



机器人要想在ros上实现SLAM算法，是需要具备一些条件的，主要呢是以下三条，硬件要求，深度信息参数设置和里程计信息的提供。



首先第一点是硬件要求，这个硬件要求仅限于ros中常用的功能包对机器人的硬件要求，而不是说要去实现这样一个功能必须要满足这些硬件要求。虽然ros系统一直以它的通用性好著称，但是并不能适用到各个具体的机器人上去，尤其是现在机器人既有天上飞的，又有地上跑的，还有水里游的。现在机器人在功能上面会有许多的差异，因此硬件上面也有许多差异，这就导致我们的一些功能和算法不可能完全适用到所有机器人上面去。因此我们这里只是介绍怎样使用最常用的功能包完成相应的工作。ros功能包对硬件的要求有下面三点，首先第一点要求机器人是差分轮式机器人，可使用Twist速度指令控制并且运动的，Twist指令包括线速度和角速度，可通过输入rosmsg show geometry\_msgs/Twist命令可以查看该 消息结构。所以当我的机器人接收到运动速度指令之后，如果向左转，就是左轮比右轮旋转的慢，这样就可以向左转，若是左轮旋转得快，就是向右转。通过这种形式就可以控制机器人做运动。当然这部分电机的控制是需要我们去在嵌入式开发板或者运动控制器上完成相应的工作，驱动这一部分是ros没办法为我们提供的，是需要我们自己去实现的。



第二个要求是机器人必须安装激光雷达等测距设备，可以获取环境的深度信息。这是很好理解的，因为我们机器人要去做SLAM建图，要去做导航，它首先要知道周围环境的障碍物距离机器人本体有多远，那自然就需要去给机器人装备一些测距的设备来感知这个距离，而这种设备之前已经提到过了，就是激光雷达和视觉传感器。这是激光雷达的一些图片。



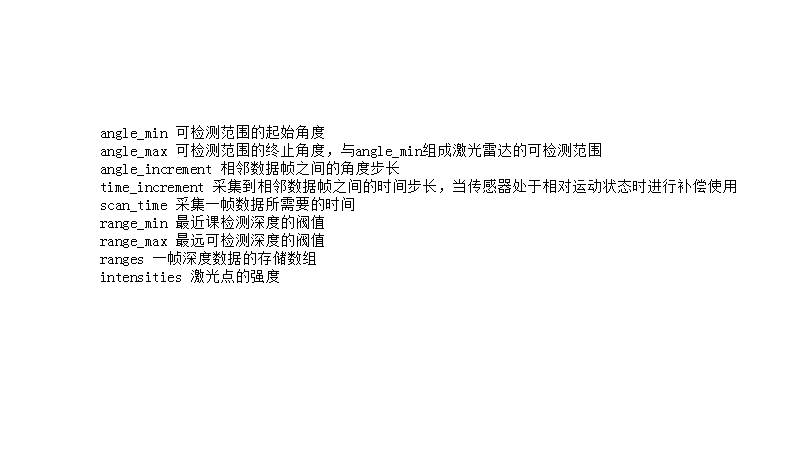
这是视觉传感器，包括深度相机，单目以及双目摄像头。



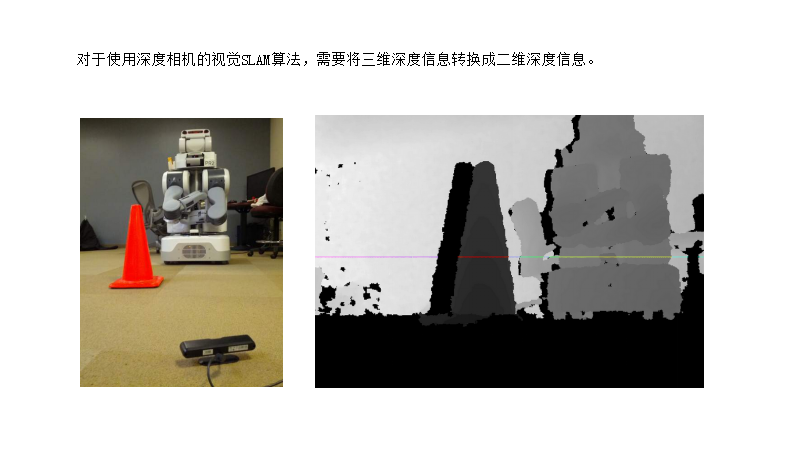
第三点就是最好我们机器人使用正方形或者圆形外形的机器人，其他外形的机器人也可以正常使用，比如说人形机器人，但是效果要比圆形或者方形机器人差很多。这是因为这些算法刚开始被开发出来的时候，都是应用于pr2或TurtieBot等圆形轮式机器人上的，所以当这些算法运用到其他外形的机器人上时，效果就达不到当初设计的算法效果。以上三点就是使用ros常用算法包时对硬件的要求。



接下来必备条件是机器人的深度信息，这是机器人感知周围环境的一个很重要的传感器信息。不同的传感器的深度信息不一样，我们先以激光雷达为例，可以用rosmsg show sensor\_msgs/LaserScan命令来查看消息结构，如图所示



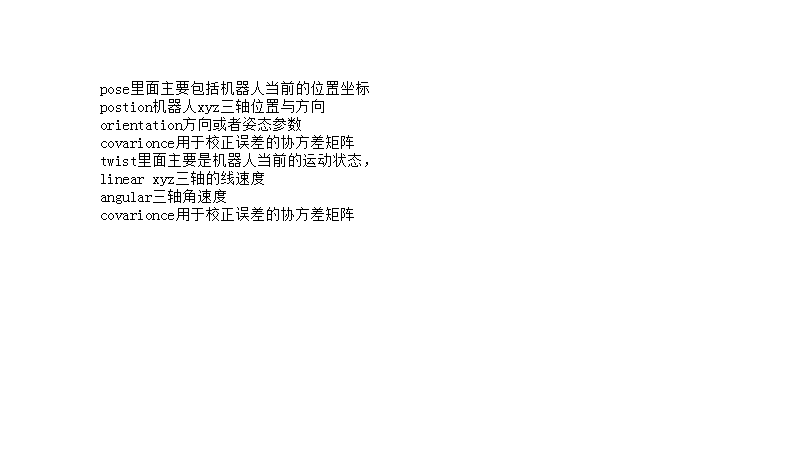
首先最上面的四行是header，包括序列，时间戳和坐标系，而后面是激光雷达的一些数据，包括激光雷达的起始角和终止角，这两个参数结合起来就是激光雷达可检测的一个范围。像国内低成本的雷达Rplidar，它的起始角度可以设置为-180度，终止角度设置为180度，所以它的可检测范围就是从-180到180这样360度的范围，另外数据中还包括相邻数据帧之间的角度步长，就是每两条激光线之间的一个角度，其他参数依次为采集到相邻数据帧之间的时间步长、采集一帧数据所需要的时间、最近可检测深度和最远可检测深度等信息，都是描述了一些激光雷达的基本参数。而ranges中存储的就是深度信息，是以数组的形式进行存储，将所有的数据都储存到这里面去。最下面还有个intensities这个参数，这个参数指代的是激光雷达每个激光点的强度，也就是激光发射出去再反射回来所接收的激光的一个强度，通过这个强度信息我们可以感知到这个障碍物的一个反光属性啊等一些性质，但是一般我们常用的雷达在强度这部分是用不到的，只有一些高端的雷达才会使用到这个强度信息作为一个辅助的判断



而使用带有深度相机的机器人获取的深度信息是三维的，但是ros中的导航和SLAM功能包是在二维实现的，这就需要将三维的深度信息转换为二维的深度信息。我们举实际生活当中的一个例子。我们可以看到左图中有一个白色机器人，一个红色路障和一个kinect摄像头，Kinect摄像头产生的右图的点云的数据，那我们怎么去在这个点云的数据当中去采集一行的深度信息，其实很简单，只需要在某一行做一个切面，比如图中那条彩色的线，这条线是在某一个高度上做一个横切面，这个横切面取出来的一行的点就作为是最终从三维转换到二维的一个激光数据，其中红色代表离kinect距离比较近，紫色代表离kinect比较远。关于这部分的功能，ros是直接给我们提供一个功能包，功能包的名字叫做depthimage\_to\_laserscan。所以说不管是激光雷达还是kinect深度相机，都可以产生一个二维的深度信息。



另外一个很重要的传感器信息就是里程计信息，这也是许多ros功能包都需要去输入的一个传感器信息，但并不是所有功能包都需要这个信息。在之后的实训课程当中，有一些是不需需要这些里程计信息的。里程计信息的结构可以通过命令rosmsg show nav\_msgs/Odometry命令去查看。



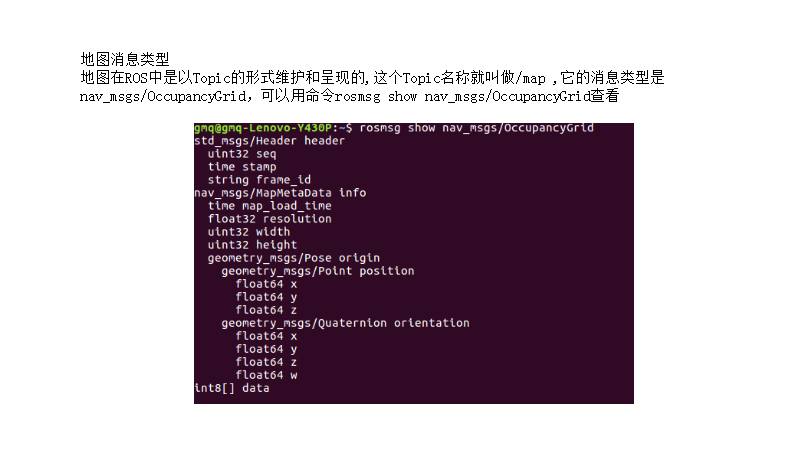
里程计信息主要分为两部分，一个是pose，一个Twist。其中Twist在之间说过了，里面主要包括线速度和角速度，就是机器人当前的实时速度。而pose是机器人当前的一个实时位置，包括机器人的xyz方向上的一个位置以及机器人的姿态位置，是以一个四元数来表示不管是在速度还是在位置信息里面，都会包含一个协方差矩阵，这个矩阵的作用可以帮我们做里程计的位置估算，但如果我们是用编码器作为里程计信息的获取的传感器，是不需要用这个矩阵的。



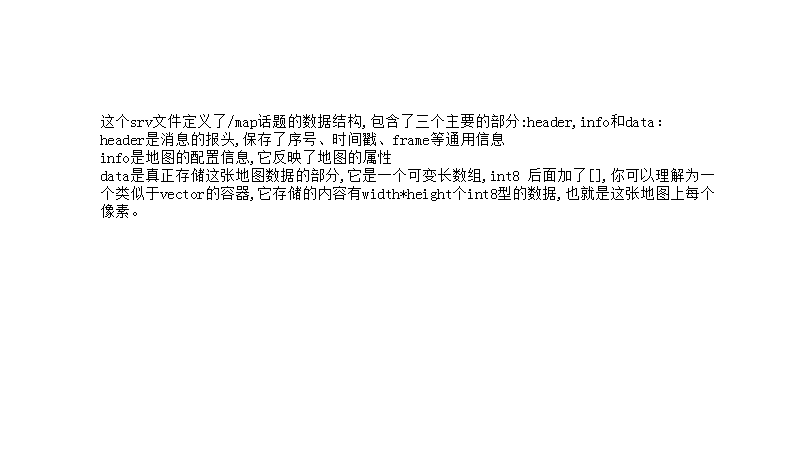
在SLAM中，地图是不可避免的话题，也是SLAM功能最终输出的结果，现在我们将详细地来讲解。



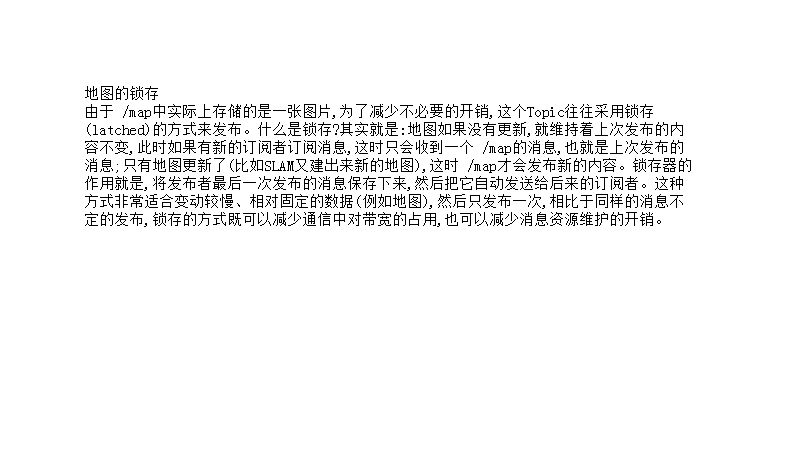
在ros中，建立的地图就是这个样子的，这是什么呢，其实地图就是一个topic。虽然是一张图片，但是是以topic的形式呈现的这个topic的名字就是map，对应的类型是OccupancyGrid，这个其实就是一个栅格地图，就是说其实这个地图是由一个个方形的栅格拼接起来的，你可以理解为，这个地图是由一个个小格子组成的。这个message是定义在nav\_msgs这个package里面的



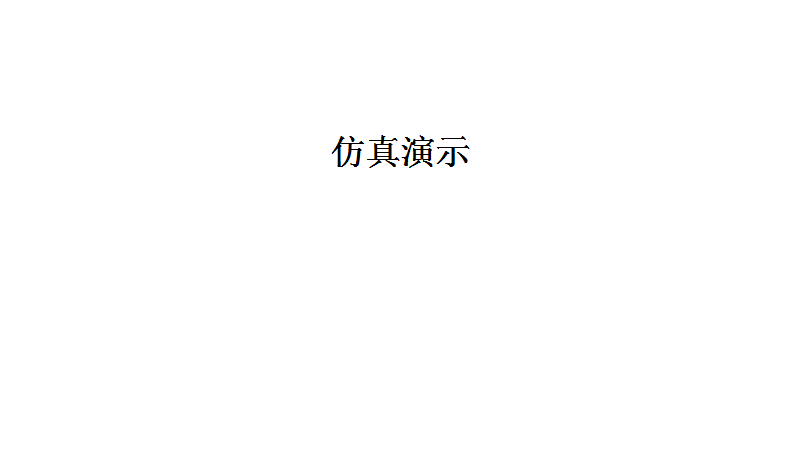
该消息可以用命令rosmsg show nav\_msgs/OccupancyGrid来查看，我们可以来看一下这个消息的结构是什么样的



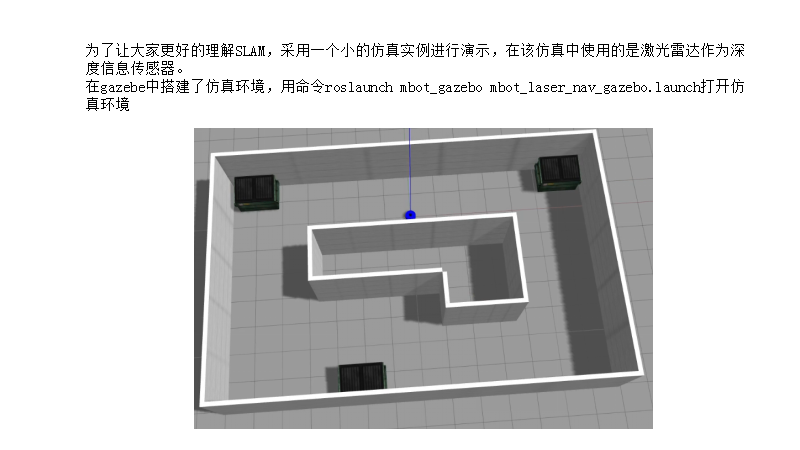
整个数据结构呢主要包含三部分，第一部分是header，也就是消息的报头，其中保存了序号，时间戳，坐标系等信息，其中坐标系呢一般就是名为map的坐标系了，这个坐标系指的就是tf上的坐标系，因此地图这个消息就是绑定在map这个坐标系上的，第二部分Info中是存储了当前地图中的一些相关信息，它并不是存储地图的数据而是相关信息，比如地图加载时间，地图的分辨率，地图的宽度和高度。其中分辨率的单位m/pixd，由于地图是栅格地图，所以分辨率的意思就是每一个方格边长实际是多少m。高和宽单位都是pixd，也就是高维度上有多少个像素，宽维度上有多少个像素。还有我们熟悉的pose和origin，就是机器人的位置和姿态信息。那整个消息最重要的就是最下面这个data信息，这部分才是我们实际地图每一格的地图，这里他用一个数组来表示。那大家可以想一下这个数组尺寸有多大呢，其实很简单，就是上面参数的width\*heigth，也就是宽度乘以高度。因为宽度和高度表示的是这两个维度上有多少个像素，所以他两相乘就是整个地图上的像素数。那每一个像素呢就是一个字节。



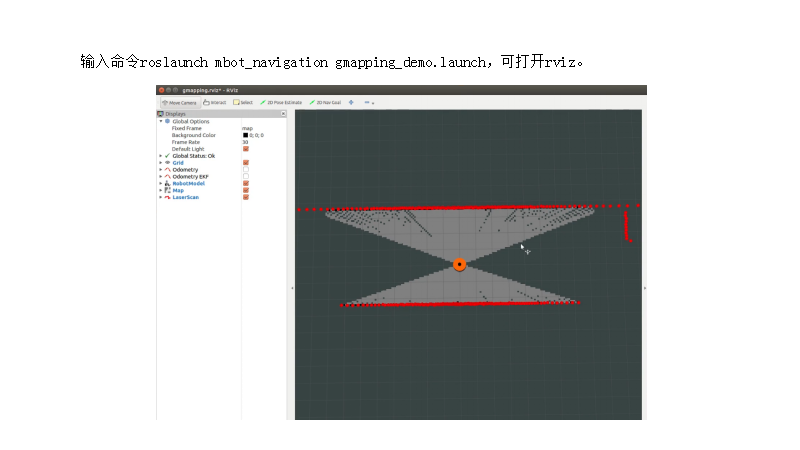
由于 /map中实际上存储的是一张图片,为了减少不必要的开销,这个Topic往往采用锁存(latched)的方式来发布。什么是锁存?其实就是:地图如果没有更新,就维持着上次发布的内容不变,此时如果有新的订阅者订阅消息,这时只会收到一个 /map的消息,也就是上次发布的消息;只有地图更新了(比如SLAM又建出来新的地图),这时 /map才会发布新的内容。锁存器的作用就是,将发布者最后一次发布的消息保存下来,然后把它自动发送给后来的订阅者。这种方式非常适合变动较慢、相对固定的数据(例如地图),然后只发布一次,相比于同样的消息不定的发布,锁存的方式既可以减少通信中对带宽的占用,也可以减少消息资源维护的开销。



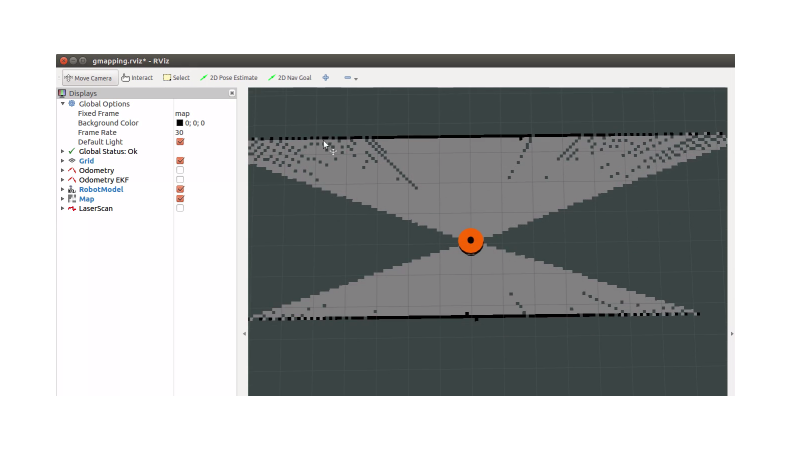
为了让同学们能够更好地理解SLAM，我们在Gazebe上进行一个仿真演示，首先将文件拷到自己创建的工作目录下，并进行编译，编译成功后，进行到下一步。



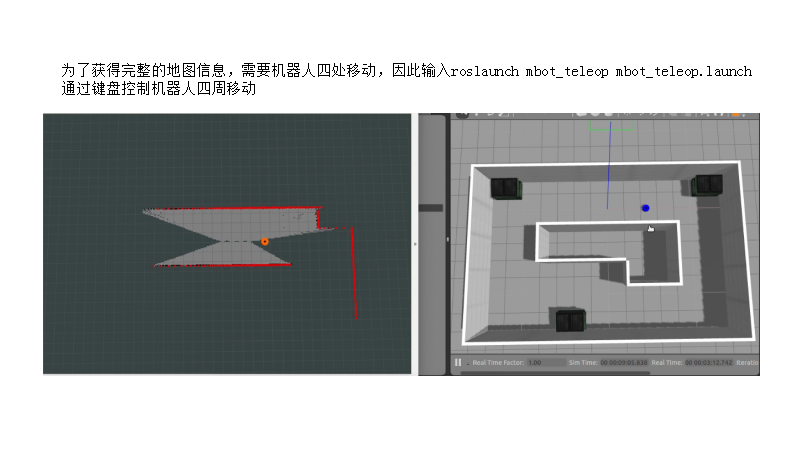
使用命令roslaunch mbot\_gazebo mbot\_laser\_nav\_gazebo.launch打开Gazebe仿真器，仿真器中就会呈现已经建立好的仿真地图，我们可以看到这是一个封闭的方形空间，里面还有障碍物，环境中蓝色的就是移动机器人。



继续输入命令roslaunch mbot\_navigation gmapping\_demo.launch，就是启动SLAM算法gmapping节点，并且自动弹出rviz界面，我们可以在rviz中看到Gazebe环境中机器人的模型，就是橙色的圆，机器人的周围会有很多红色的点，这些红色的点就是我们机器人上面放置的激光雷达感知的深度信息，大家可以跟我们的Gazebe仿真环境做一下对比，这些红色的点就是周围的墙体，同时我们在rviz当中还可以看到一些浅灰色的信息，这些信息就是目前机器人已经创建完成的地图，也就是说这些地方已经是地图信息了。

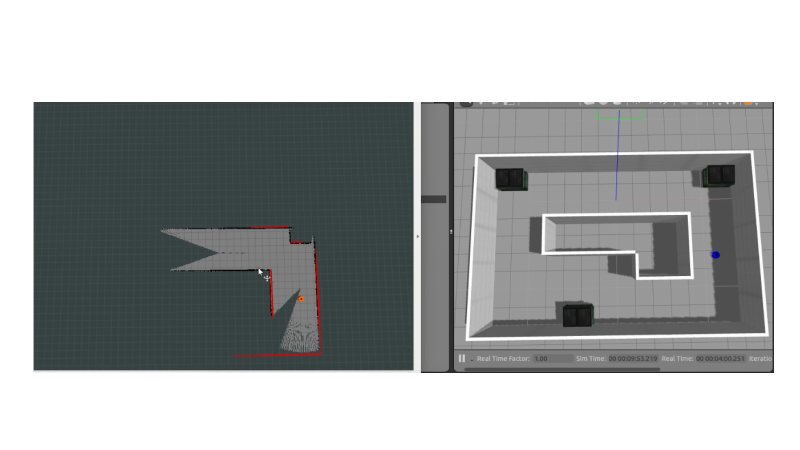


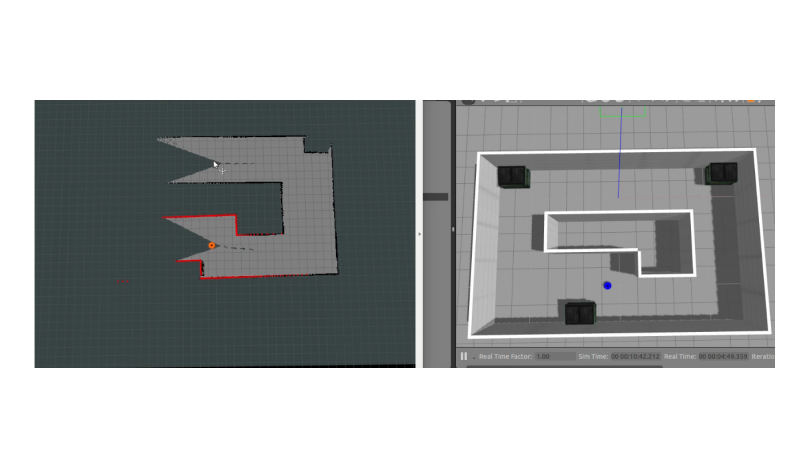
我们可以先把激光雷达的点先关闭，我们可以看到在最边缘是一些黑色的点，这些点是已经建立完成的地图中的一些障碍物，而浅灰色代表的是没有障碍物，所以浅灰色部分就是障碍物可以运动的，像黑色部分就是有障碍物存在的，和我们的实际环境非常相似。那就下来就是要让机器人在仿真环境当中去做运动，它在运动过程当中会把周围的环境信息给建立出来，建立出一张完整的二维栅格地图。

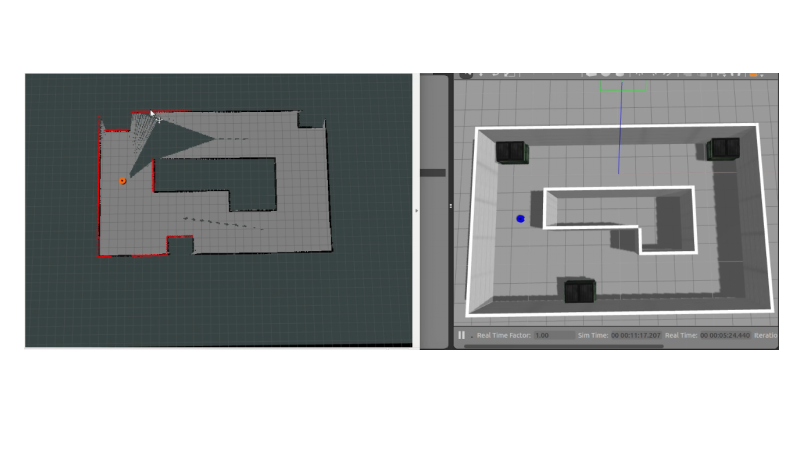


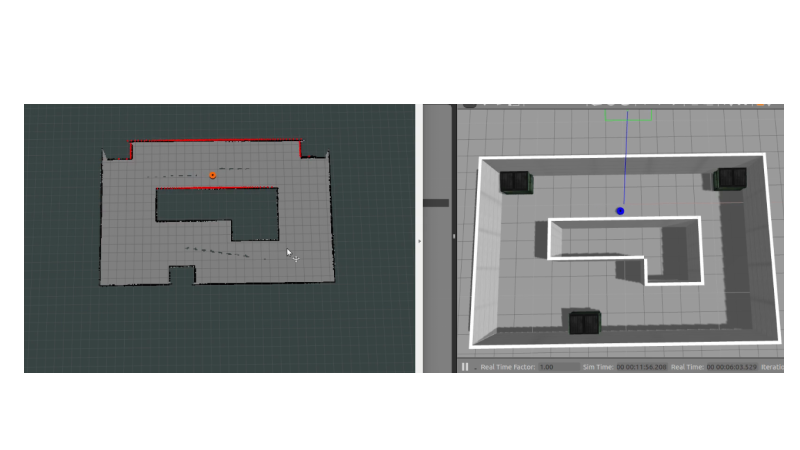
因此要做的第三步就是来启动键盘控制节点，我们这里首先是使用键盘控制这种人为的方式来控制机器人在环境中做运动。

我们在键盘上摁住向前运动，可以看到机器人在rviz这个图里面已经开始向前运动了，Gazebe也在向前运动，两者是保持完全一致的运动状态，因为我们是控制的一个仿真机器人，rviz可以帮我们实现的是整个机器人系统的数据显示，比如显示机器人模型，显示障碍物信息，以及地图信息。所以我们可以吧Gazebe环境想象成一个真实的机器人在做运动。大家可以在自己的电脑上做控制，控制自己的电脑上机器人做运动，来实时地看到建图的一个进程，可以看到激光雷达检测的范围是非常大的，也是360度这个范围，所以根据激光雷达可以做自身的一个定位校准的，然后我们可以看到整个地图在校准过程当中会发生整体的调整，这就是因为SLAM算法在整体的建图过程当中会同时完成机器人定位的校准，我们来控制机器人在环境中运行一周，大概去看一下整个地图的建立情况，最后能否形成一个封闭的地图。

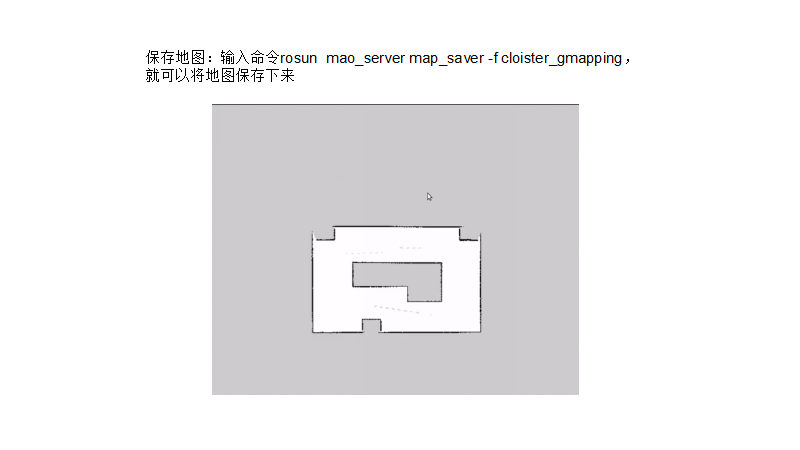








好的，大家可以来看一下地图，基本是完全一致的，可以看到整个建图过程还是很快的，基本上走完一圈以后，地图就建立出来了。



可以通过输入命令rosun mao\_server map\_saver -f cloister\_gmapping，将创建的地图信息保存下来，保存的路径是在打开的这个终端的根目录下，打开根目录后就会看到这就是一个图片。