接下来我们要介绍的是ROS当中的第四个相对来讲比较常用的slam功能包，ORB\_SLAM。ORB\_SLAM跟我们之前的功能包最大的区别在于，他是使用单目摄像头来完成SLAM的。而且是三维点云的图，跟我们之前的二维地图是有很大区别的。它是靠摄像头来完成三维地图的构建。而且ORB\_SLAM这个功能包它也不需要机器人给它提供里程计信息，完全是使用摄像头来完成地图的构建的。很适用于手持设备，或者行驶当中连续图像的建图。比如说现在非常火的无人驾驶，像TESLA无人车就是使用摄像头来完成SLAM地图的构建的，在TESLA上是没有使用激光雷达的。而另外一派的百度的无人车是使用三维的激光雷达来完成整个地图的构建。两种方案最大的区别就是一个使用了摄像头，一个使用了激光雷达。那这两种传感器最大的区别就在于成本方面，一个激光雷达，它的成本可能在几十万这样一个量级，但是摄像头在成本上就会低非常多。所以说像TESLA这样的无人车它整个的成本就会低很多。

关于ORB\_SLAM功能包，它是国外的几个博士2015年发表在IEEE上的一个算法，同时封装成ROS当中的一个功能包，已经开源，大家可以下载他的源码，去尝试一下这个功能包SLAM建图的效果。

ORB\_SLAM功能包由于它只是提供了源码，并没有在ROS的软件源里面编译好安装包，所以我们需要去下载源码然后编译，整个编译过程相对来讲比较复杂，我们需要为它安装许多依赖库，主要包括eigen，g2o这些库，还有DboW库，然后就可以编译ORB\_SLAM的源码，编译完成之后还要去配置它的ROS环境。所以整个流程会稍微复杂一些，大家可以根据下面的编译步骤来进行编译，也可以在百度上搜到很多的教程，大家也可以通过网上的资源来做尝试。

安装链接库

$ sudo apt-get install libboost-all-dev libblas-dev liblapack-dev

下载源码

$ git clone <https://github.com/raulmur/ORB_SLAM2.git>

安装eigen3.2

<http://eigen.tuxfamily.org/index.php?title=Main_Page>下载源码包解压源码包，并进入目录编译安装：

$ mkdir build & cd build

$ cmake .. & make

$ sudo make install

编译g2o

$ cd ORB\_SLAM-master/Thirdparty/g2o/

$ mkdir build

$ cd build

$ cmake .. -DCMAKE\_BUILD-TYPE=Release

$ make

编译DBoW2

$ cd ORB\_SLAM

$ mkdir build & cd build

$ cmake .. -DROS\_BUILD\_TYPE=Release  
$ make

编译功能包

$export ROS\_PACKAGE\_PATH={ROS\_PACKAGE\_PATH}:ORB\_SLAM\_PATH

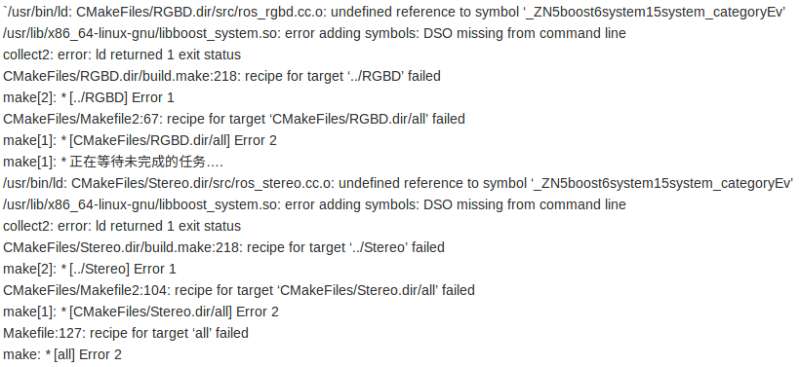
/ORB\_SLAM2/Example/ROS

$ chmod +x build\_ros.sh

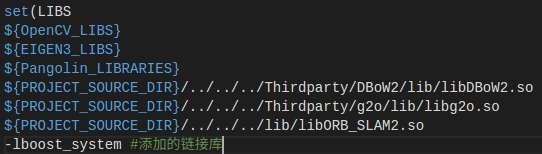
$ ./build\_ros.sh

$ source ORB\_SLAM2/Examples/ROS/ORB\_SLAM2/build/devel/setup.bash

大家在编译过程中4可能会发现一些问题，第一个呢，会报错usleep函数未定义，解决方法是在下面列出的文件当中添加#include<unistd.h>头文件。



第二个问题，报错未定义的引用，原因是系统编译的时候找不到boost的链接库，解决方法是修改ORB\_SLAM2/Example/ROS/ORB\_SLAM/CmakeLists.txt，添加一句-lboost.system。



这里主要给大家演示一下如何使用ORB\_SLAM功能包，首先第一步我们使用官方给我们提供的数据包，通过这个数据包里面提供的一些话题数据来完成ORB\_SLAM的功能。接下来我们来运行这个例程，首先我们来启动rosmaster,

$ roscore

然后我们启动ORB\_SLAM功能包当中的一个功能节点，这里需要加载一些数据文件，

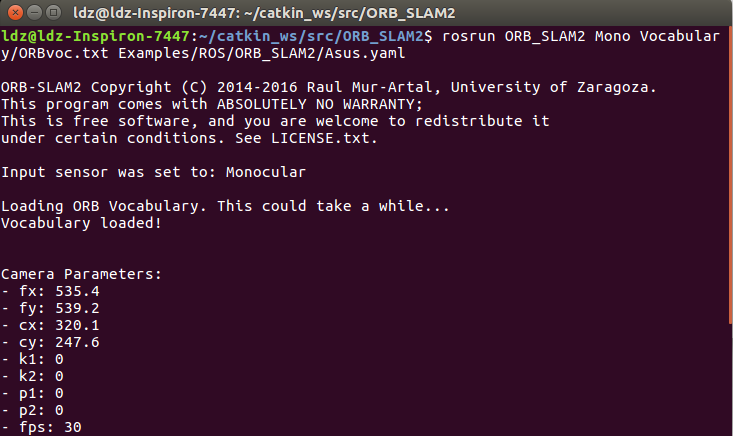
$ rosrun ORB\_SLAM2 Mono Vocabulary/ORBvoc.txt Examples/ROS/ORB\_SLAM2/Asus.yaml

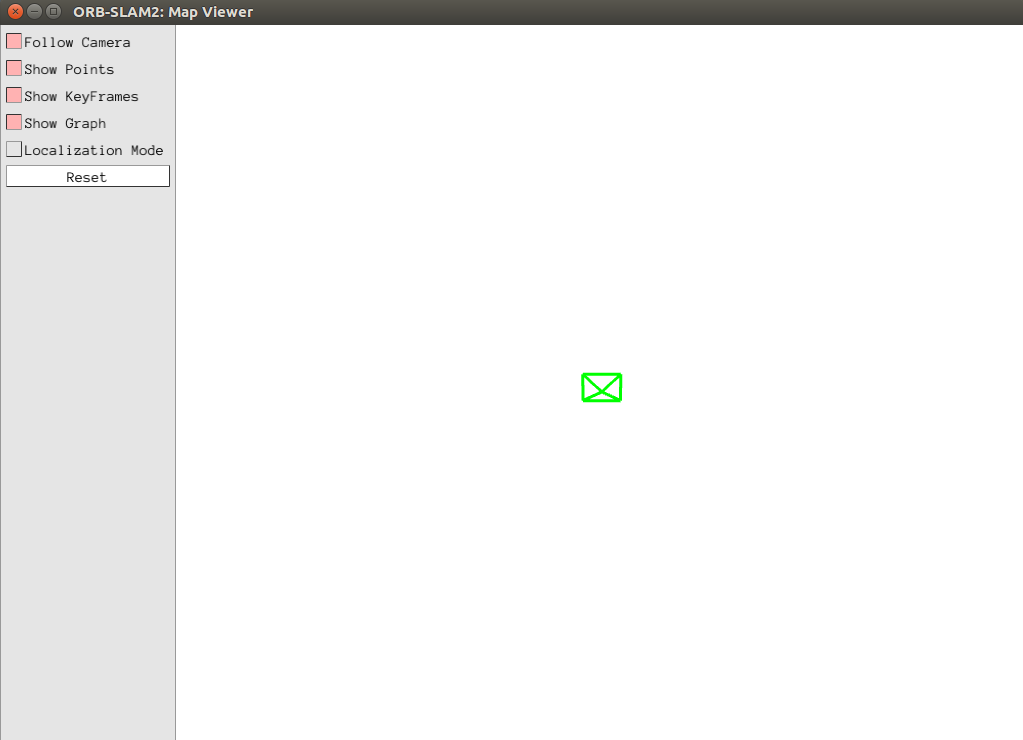
OK，现在我们启动了ORB\_SLAM的功能，这里要稍微等一下，因为系统需要加载一些数据文件，加载完成之后会显示出来这样的一个界面，当前因为还没有数据播放出来，所以它还不会产生任何的效果。

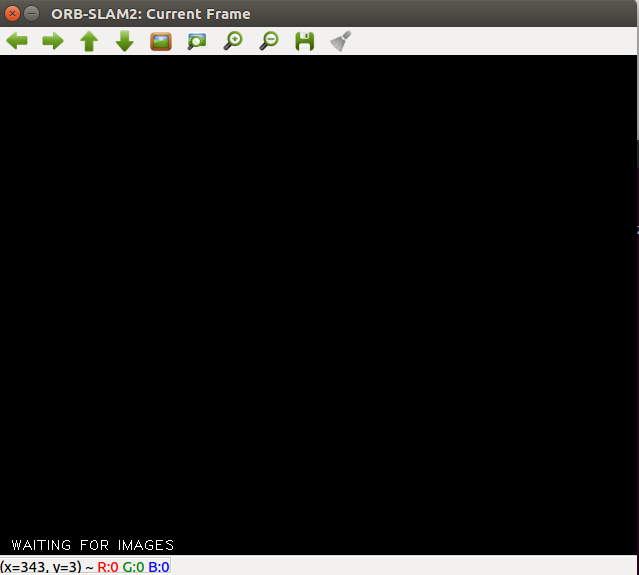
接下来我们去播放数据文件，这个数据文件是之前我在网上已经下载下来的，大家可以去这个网址上面去下载

然后我们需要通过rosbag命令来播放这个文件，需要注意的是，大家一定要在自己下载的数据文件所在目录下面去播放文件，否则会播放失败，

$ rosbag play rgbd\_dataset\_freiburg1\_desk.bag/camera/rgb/image\_color:=/camera/image\_raw





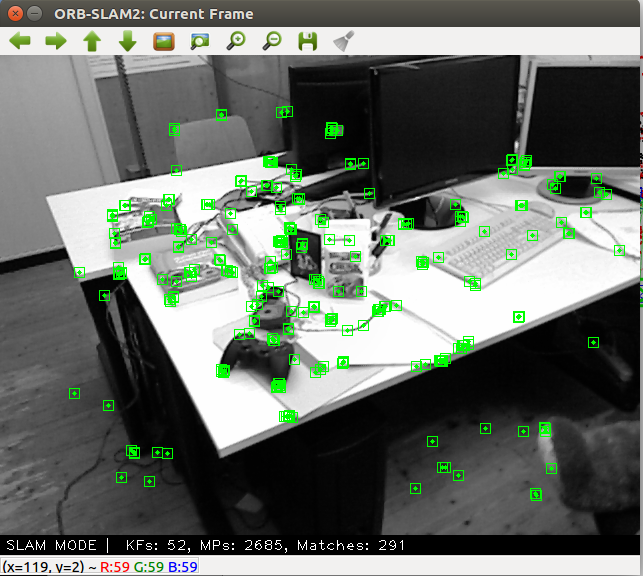


接下来我们播放数据

$ rosbag play rgbd\_dataset\_freiburg1\_desk.bag/camera/rgb/image\_color:=/camera/image\_raw

很快就可以看到可视化界面，左边就是我们这个摄像头，可以看到周围环境的信息，以及他在不断的匹配里面的一些特征点，这些绿色的点就是匹配出的特征点。右边这个界面是整个建图的流程。蓝色框代表摄像头移动的轨迹，这里面会出现很多点状的信息。这些点状的信息，就是最终建立好的，点云形态的一个地图信息，就是左边摄像头匹配出的很多的特征点。这就是一个三维的地图信息，可以使用鼠标来做拖拽，得知整个地图的形态。这样的一个地图信息看上去并不是十分的直观，因为它是由很多的特征点来组成的。

这个例程的实现效果就是这样的，它最终建立的地图就是一个点云地图。



接下来给大家演示的是如何在仿真环境中搭建的机器人上面使用ORB\_SLAM的功能。在我们提供的仿真环境中，机器人所使用的是能够获取周围环境深度信息的RGB-D摄像头，ORB\_SLAM2需要使用OpenCV处理图像以及特征，所以我们还要先安装OpenCV。

首先我们安装OpenCV的依赖包

$ sudo apt-get install build-essential

$ sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

$ sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

$ sudo apt-get install libxvidcore-dev libx264-dev

$ sudo apt-get install libatlas-base-dev gfortran

$ sudo apt-get install ffmpeg

然后分别下载OpenCV压缩包以及ippicv*\_linux\_*20151201.tgz

<https://github.com/opencv/opencv/archive/3.2.0.zip>

[https://raw.githubusercontent.com/opencv/opencv*\_3rdparty/81a676001ca8075ada498583e4166079e5744668/ippicv/ippicv\_*linux\_20151201.tgz](https://raw.githubusercontent.com/opencv/opencv_3rdparty/81a676001ca8075ada498583e4166079e5744668/ippicv/ippicv_linux_20151201.tgz)

编译安装OpenCV

$ unzip opencv-3.2.0.zip

$ cd opencv-3.2.0

$ mkdir build

$ cd build

$ cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local -D WITH\_TBB=ON -D BUILD\_NEW\_PYTHON\_SUPPORT=ON -D WITH\_V4L=ON -D WITH\_QT=ON -D WITH\_OPENGL=ON ..

在cmake下载过程中可能会出现 ICV:DOWNLONDING [ippicv\_linux\_20151201.tgz](http://download.csdn.net/detail/alphapii/9852916) 但是一直下载不下来，把自己下载的压缩包放到opencv3.2.0->ippic->Download下，然后重新cmake

安装

$ make

$ sudo make install

安装的速度可能会很慢，大家耐心等待即可。安装完之后还要配置环境：

$ sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'

$ sudo ldconfig

下面我们启动仿真环境下基于机器人携带的RGB-D摄像头的单目SLAM演示示例。首先还是先启动rosmaster

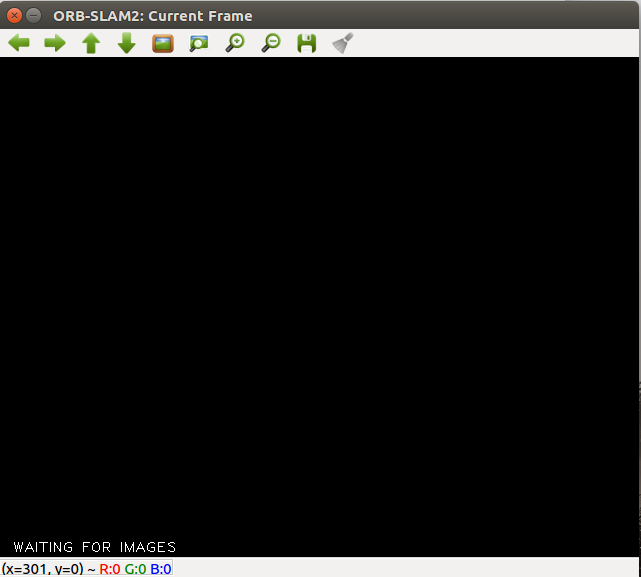
$ roscore

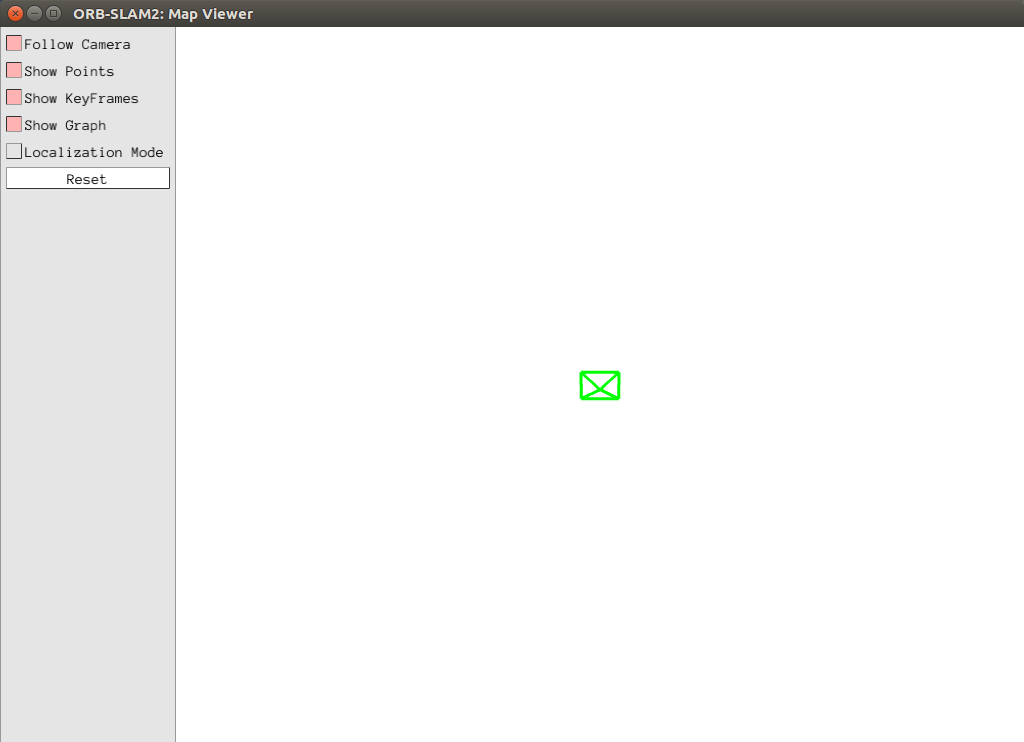
再启动ORB\_SLAM之前，我们需要修改一下ros\_rgbd.cc源文件，因为我们仿真环境中的机器人发布的话题和ORB\_SLAM算法订阅的话题不是相同的，我们要把/ORB\_SLAM2/Examples/ROS/ORB\_SLAM2/src/ros\_rgbd.cc文件中所订阅的话题名称修改为机器人所发布的图像信息话题名称。即将camera/depth\_registered/image\_raw改为camera/depth/image\_raw

接着打开一个新的终端，启动ORB\_SLAM算法程序

$ roslaunch orbslam2\_demo ros\_orbslam2.launch

同时会打开与刚才相同的可视化界面。

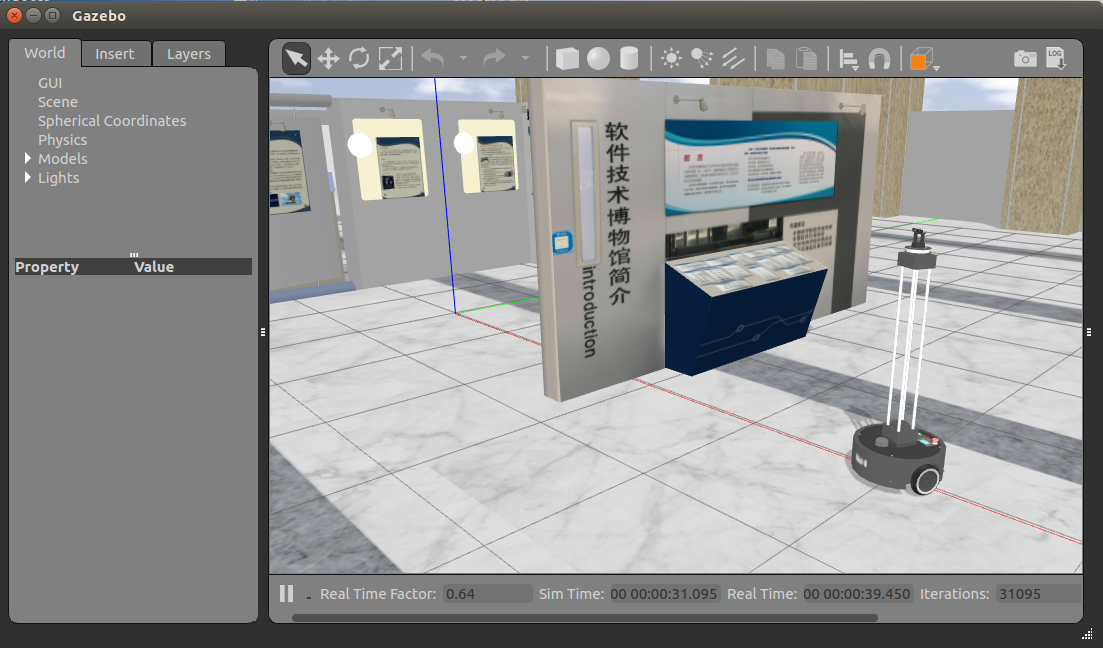


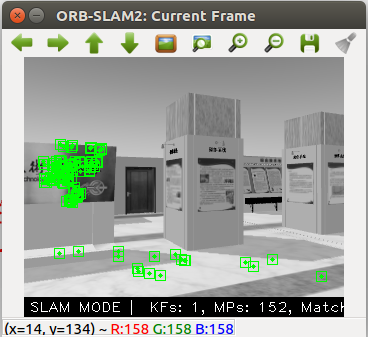


之后再打开新的终端输入命令打开gazebo仿真环境：

$ roslaunch robot\_sim\_demo robot\_spawn.launch

此时可以看到RGB-D摄像头视角中的仿真环境显示在图像信息窗口中。根据环境信息匹配得到的特征点也显示在地图创建窗口中。

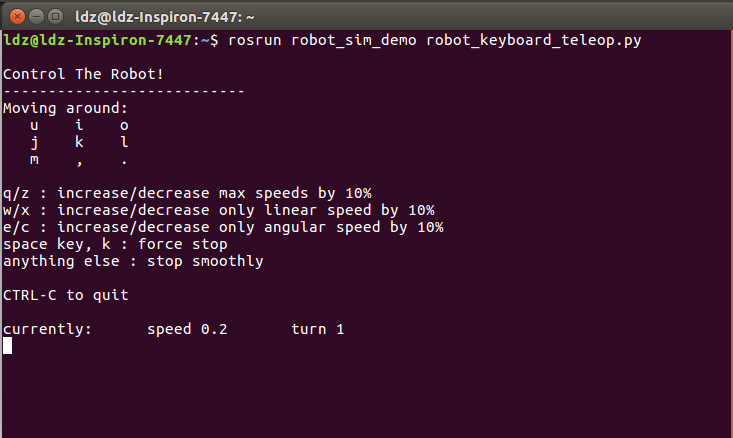




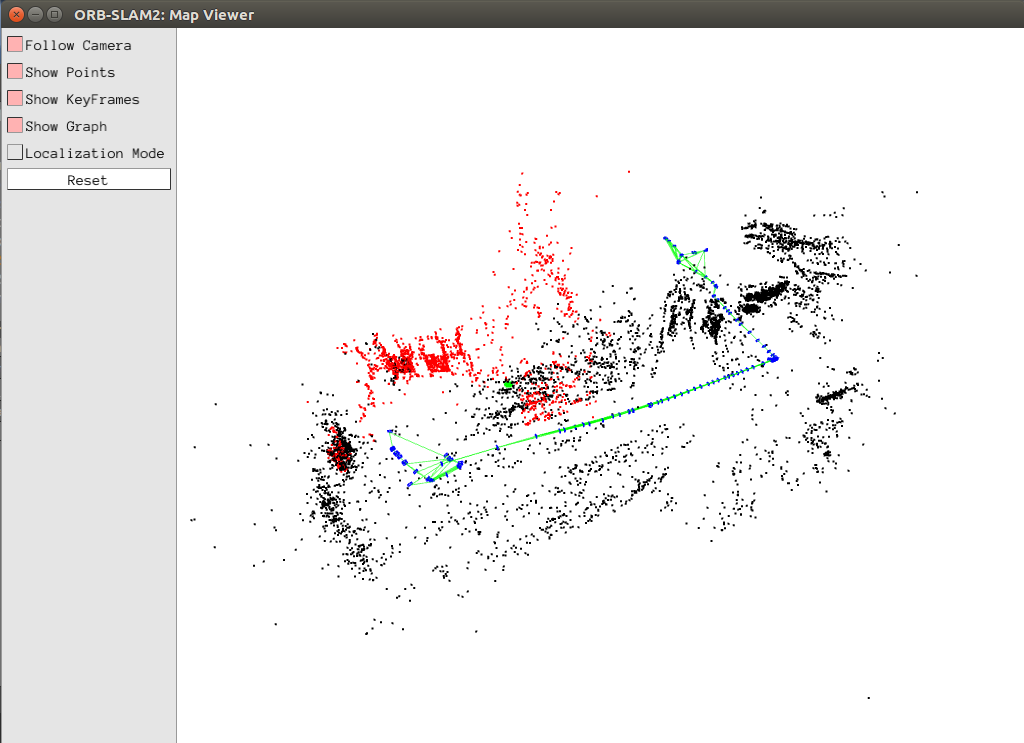
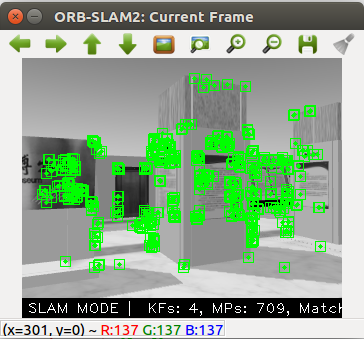
接下来我们打开键盘控制界面：

$ rosrun robot\_sim\_demo robot\_keyboard\_teleop.py

通过按键控制机器人移动



可以看到我们的摄像头中不断变化的环境信息以及匹配到的特征点，构建的三维点云地图也在不断更新，摄像头移动的轨迹也就是机器人运动的轨迹。只要我们控制机器人走完整个仿真环境，最终就可以构建出完整的点云形态的地图。这个呢是我们构建好的部分地图，大家可以清晰的看到机器人运行过的轨迹，但是环境特征点还是比较的抽象，我们无法直观的看到地图的环境是什么样的。且由于ORB\_SLAM输出的是一个三维点云形态的地图，功能包中没有提供保存地图的功能节点。我们无法像其他构建二维栅格地图的slam算法那样方便的保存地图，因为地图中包含关键帧(包括相机的pose,相机的内参,ORB特征),3D的地图点( 空间中3D位置,法线方向,ORB的描述子),词袋向量,共视图等信息。但也并不是没有保存的方法，有兴趣的同学可以上网查找一些相关博文，自己动手实现一下。



为了测试大家对本节内容掌握的情况，给大家准备了一个练习题，假如我们通过播放数据包的形式启动ORB\_SLAM功能时输入的命令为：

$ rosbag play rgbd\_dataset\_freiburg1\_desk.bag/camera/rgb/image\_color:=/camera/rgb/image\_raw

那么我们要如何修改ORB\_SLAM源文件才能够成功启动功能节点呢？

答案:只需将/ORB\_SLAM2/Examples/ROS/ORB\_SLAM2/src/ros\_mono.cc文件中所订阅的话题名称/camera/image\_raw修改为/camera/rgb/image\_raw然后重新编译运行即可