2.熟悉Linux

（1）如何在Ubuntu中安装软件（命令行界⾯）？它们通常被安装在什么地方？

打开终端(ctrl+alt+t)，输入命令sudo apt-get install packagename安装软件。

软件一般安装在/usr下，包含很多文件夹，根据文件的类型，分门别类，不是一个软件一个文件夹。老版本的Linux一般安装放在/usr/local目录下。部分软件放在/opt下， /opt目录专门是用来给第三方软件放置文件的，比如一些压缩包解压的软件都放在这里。

可以使用命令dpkg -L softwarename或whereis softwarename查询安装路径。

（2）Linux的环境变量是什么？如何定义新的环境变量？

环境变量是一个具有特定名字的对象，它包含了一个或者多个应用程序所将使用到的信息。

使用export设置一个新的环境变量export HELLO = "hello"，其中Hello为变量名，hello为变量的值。

（3）Linux 根⽬录下⾯的⽬录结构是什么样的？至少说出3 个⽬录的用途。

Linux文件系统被组织成一个有层次的树形结构。文件系统的最上层是 /，或称为根目录。在Linux 的设计理念中，一切皆为文件——包括硬盘、分区和可插拔介质。这就意味着所有其它文件和目录（包括其它硬盘和分区）都位于根目录中。例如：/home/jebediah/cheeses.odt 给出了正确的完整路径，它指向 cheeses.odt 文件，而该文件位于 jebediah 目录下，该目录又位于 home 目录，最后，home 目录又位于根(/) 目录下。

在根 (/) 目录下，有一组重要的系统目录，在大部分 Linux 发行版里都通用。直接位于根 (/) 目录下的常见目录列表如下：

/boot - 启动 (boot) 配置文件/dev - 设备 (device) 文件/etc - 配置文件、启动脚本等 (etc)/home - 本地用户主 (home) 目录/lib - 系统库 (libraries) 文件/lost+found - 在根 (/) 目录下提供一个遗失+查找(lost+found) 系统/media - 挂载可移动介质 (media)，诸如 CD、数码相机等/mnt - 挂载 (mounted) 文件系统/opt - 提供一个供可选的 (optional) 应用程序安装目录/proc - 特殊的动态目录，用以维护系统信息和状态，包括当前运行中进程 (processes) 信息。/root - root (root) 用户主文件夹，读作“slash-root”/sbin - 重要的系统二进制 (system binaries) 文件/sys - 系统 (system) 文件/tmp - 临时(temporary)文件/usr - 包含绝大部分所有用户(users)都能访问的应用程序和文件/var - 经常变化的(variable)文件，诸如日志或数据库等

（4）假设我要给a.sh 加上可执⾏权限，该输⼊什么命令？

chmod +x a.sh

（5）假设我要将a.sh ⽂件的所有者改成xiang:xiang，该输⼊什么命令？

chmod xiang:xiang a.sh

1. SLAM综述文献阅读

（1）SLAM 会在哪些场合中⽤到？⾄少列举三个⽅向。

在没有环境先验信息，同时又需要知道自身位置的场合。

室内扫地机和移动机器人、自动驾驶汽车、无人机、虚拟现实和增强现实。

（2）SLAM 中定位与建图是什么关系？为什么在定位的同时需要建图？

定位与建图是相互耦合的，准确的定位需要精确的地图，精确的地图来自准确的定位。

（3）SLAM 发展历史如何？我们可以将它划分成哪⼏个阶段？

SLAM已有30年的研究历史，前20年SLAM问题的回顾在Durrant-Whyte和Bailey的两篇论文中有详细论述。这个报告我们称之为“古典年代”（1986-2004），古典年代时期，引入了SLAM概率论推导方法，包括基于扩展卡尔曼滤波、粒子滤波和最大似然估计；而且，这里的第一个挑战是效率和数据关联的鲁棒性问题。接下来的年代，我们称之为“算法分析”年代（2004-2015）。在算法分析的年代，有许多SLAM基本特性的研究，包括可观测性，收敛性和一致性。在这一时期，研究者们理解了稀疏特征在高效SLAM解决方案中的重要角色，开发了主要开源SLAM库。

（4）列举三篇在SLAM 领域的经典⽂献。

[1] Mur-Artal R, Tardós J D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras[J]. IEEE Transactions on Robotics, 2016, 33(5):1255-1262.

[2] Engel J, Schöps T, Cremers D. LSD-SLAM: Large-Scale Direct Monocular SLAM[M]// Computer Vision – ECCV 2014. Springer International Publishing, 2014:834-849.

[3] Davison A J, Reid I D, Molton N D, et al. MonoSLAM: Real-Time Single Camera SLAM[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2007, 29(6):1052.

1. CMake练习

书写⼀个由cmake 组织的C++ ⼯程，要求如下：

1. include/hello.h 和src/hello.c 构成了libhello.so 库。hello.c 中提供⼀个函数sayHello()，调⽤此函数时往屏幕输出⼀⾏“Hello SLAM”。我们已经为你准备了hello.h 和hello.c 这两个⽂件，见“code/”⽬录下。

2. ⽂件useHello.c 中含有⼀个main 函数，它可以编译成⼀个可执⾏⽂件，名为“sayhello”。

3. 默认⽤Release 模式编译这个⼯程。

4. 如果⽤户使⽤sudo make install，那么将hello.h 放⾄/usr/local/include/下，将libhello.so 放⾄/usr/local/lib/下。

请按照上述要求组织源代码⽂件，并书写CMakeLists.txt。

# 声明要求的 cmake 最低版本

cmake\_minimun\_required(VERSION 2.8)

# 声明一个 cmake 工程

project(HelloSLAM)

#设置编译模式

IF(NOT CMAKE\_BUILD\_TYPE)

SET(CMAKE\_BUILD\_TYPE Release)

EDNIF()

# 添加一个库

add\_library(libhello SHARED hello.c)

#将hello.h 放⾄/usr/local/include/下

INSTALL(FILES hello DESTINATION /usr/local/include)

#将libhello.so 放⾄/usr/local/lib/下

INSTALL(TARGETS libhello LIBRARY DESTINATION /usr/local/lib)

# 添加一个可执行程序

# 语法：add\_executable( 程序名 源代码文件 ）

add\_executable(sayhello useHello.c)

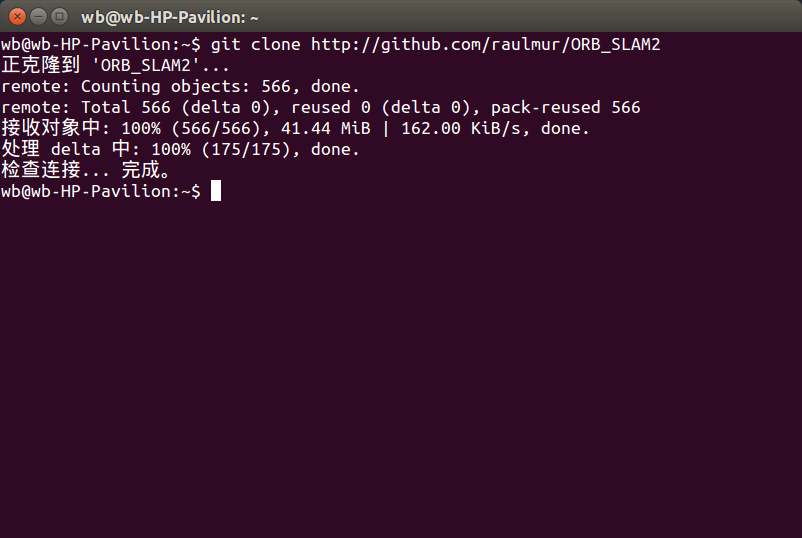
# 将库文件链接到可执行程序上

target\_link\_libraries(sayhello libhello)

5 理解ORB-SLAM2 框架

（1）从github.com 下载ORB-SLAM2 的代码。地址在：https://github.com/raulmur/ORB\_SLAM2.

提⽰：在安装git 之后，可以⽤git clone https://github.com/raulmur/ORB\_SLAM2 命令下载ORB-SLAM2。下载完成后，请给出终端截图。



（2）ORB-SLAM2 是⼀个cmake ⼯程，所以可以从CMakeLists.txt 上⾯来了解它的组织⽅式。阅读ORB-SLAM2 代码⽬录下的CMakeLists.txt，回答问题：

(a) ORB-SLAM2 将编译出什么结果？有⼏个库⽂件和可执⾏⽂件？

编译出共享库和可执行文件。

生成了3个库文件，分别为：libORB\_SLAM2.so，libBoW2.so，libg2o.so。

生成了6个可执行文件，分别为：mono\_tum，mono\_euroc，mono\_kitti，rgbd\_tum，stereo\_euroc，stereo\_kitti。

(b) ORB-SLAM2 中的include, src, Examples 三个⽂件夹中都含有什么内容？

include文件夹中为头文件，src文件夹中为源文件，Examples文件夹中为Monocular，Stereo，RGB-D，ROS。前三种为不同的传感器，最后一个为基于ROS平台。

(c) ORB-SLAM2 中的可执⾏⽂件链接到了哪些库？它们的名字是什么？

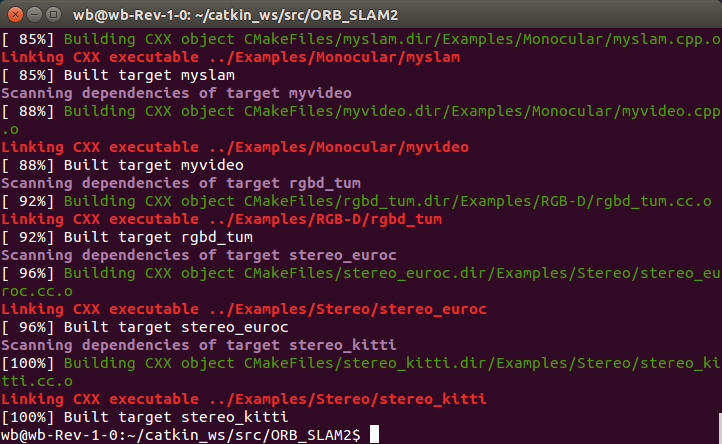
|  |
| --- |
| add\_library(${PROJECT\_NAME} SHARED |
|  | src/System.cc |
|  | src/Tracking.cc |
|  | src/LocalMapping.cc |
|  | src/LoopClosing.cc |
|  | src/ORBextractor.cc |
|  | src/ORBmatcher.cc |
|  | src/FrameDrawer.cc |
|  | src/Converter.cc |
|  | src/MapPoint.cc |
|  | src/KeyFrame.cc |
|  | src/Map.cc |
|  | src/MapDrawer.cc |
|  | src/Optimizer.cc |
|  | src/PnPsolver.cc |
|  | src/Frame.cc |
|  | src/KeyFrameDatabase.cc |
|  | src/Sim3Solver.cc |
|  | src/Initializer.cc |
|  | src/Viewer.cc |
|  | ) |

|  |
| --- |
| target\_link\_libraries(${PROJECT\_NAME} |
|  | ${OpenCV\_LIBS} |
|  | ${EIGEN3\_LIBS} |
|  | ${Pangolin\_LIBRARIES} |
|  | ${PROJECT\_SOURCE\_DIR}/Thirdparty/DBoW2/lib/libDBoW2.so |
|  | ${PROJECT\_SOURCE\_DIR}/Thirdparty/g2o/lib/libg2o.so |
|  | ) |

从CMakeLists.txt中的以上代码可以看出，可执行文件链接了源文件对应的共享库，以及OpenCV、EIGEN3、Pangolin、libDBoW2.so、libg2o.so。

6 \* 使用摄像头或视频运行ORB-SLAM2

1. 编译ORB-SLAM完成的截图

（2）CMakeLists.txt 修改⽅案

在文件最后添加：

add\_executable(myslam Examples/Monocular/myslam.cpp)

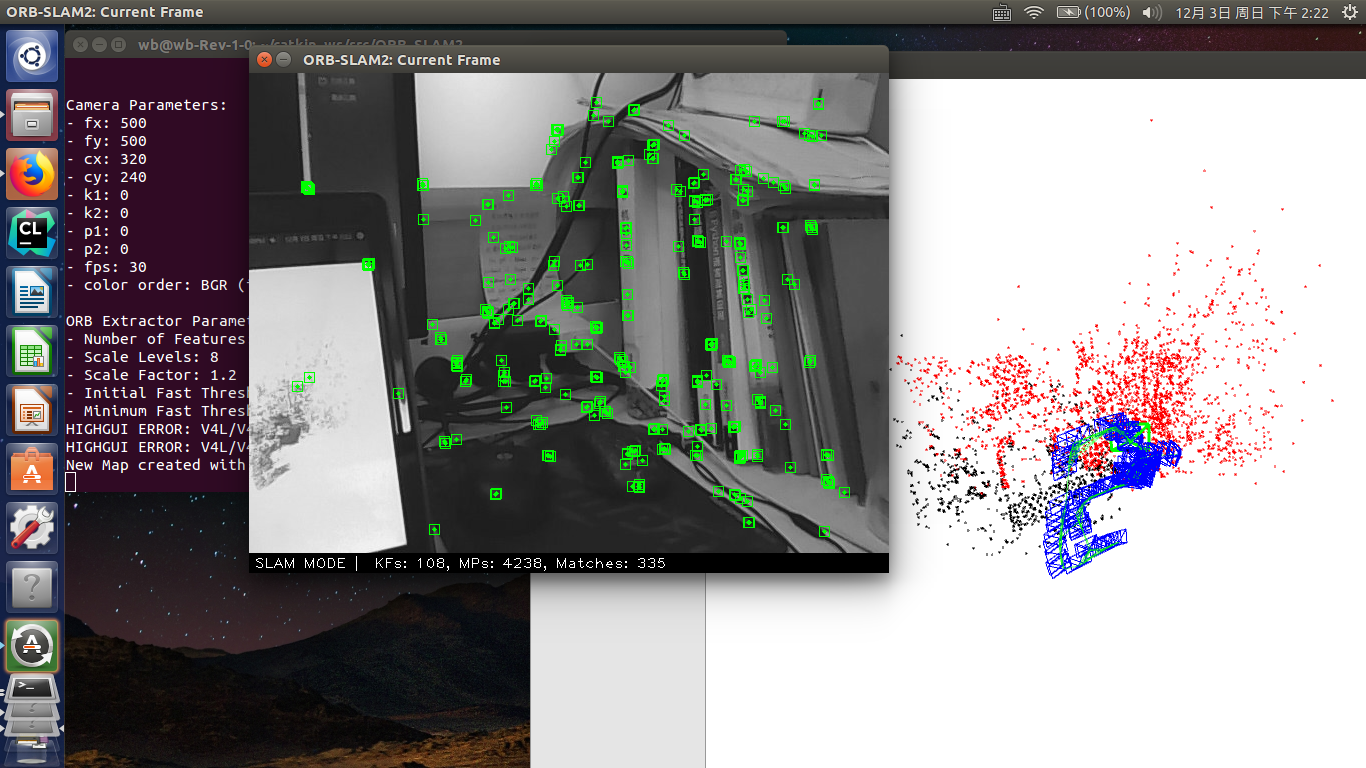
target\_link\_libraries(myslam ${PROJECT\_NAME})

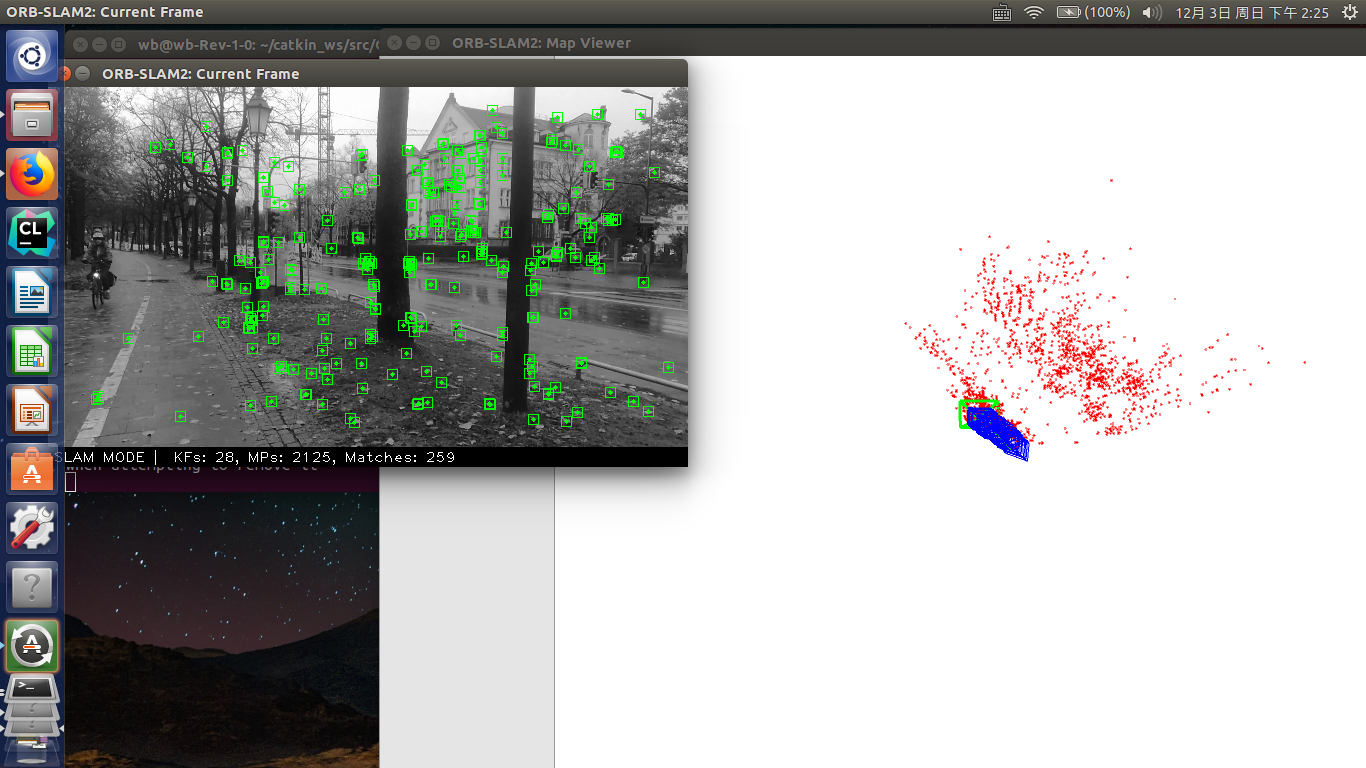
add\_executable(myvideo Examples/Monocular/myvideo.cpp)

target\_link\_libraries(myvideo ${PROJECT\_NAME})

（3）运行截图

外接USB摄像头





体会

1. 为了能够顺利地进行初始化，必须先选择纹理丰富的场景，并使相机有一定的平移。
2. 不能快速移动，最好保持匀速运动，保持当前帧与上一帧有足够数量的匹配的特征点，否则会跟踪失败。
3. 跟踪失败后，应回到之前经过的场景，进行重定位。
4. 电脑性能不能太低，保持电量充足。