第一节课习题

高翔

2021年9月8日

* **习题说明**
  + 第 *i* 节课习题所有材料打包在 L*i*.zip 中， *i* = 1 *. . .* 8。
  + 习题分为若干种：**计算类**习题，需要读者编程计算一个实际问题，我们会附有参考答案以供自测。**操作类**习题，会指导读者做一个具体的实验，给出中间步骤截图或结果。**简述类**习题则提供阅读材料，需要读者阅读材料后，回答若干问题。
  + 每个习题会有一定的分值。每次习题分值加和为 10 分。你需要获得 8 分以上才能得到“通过”的评价。带 的习题为附加题，会在总分之外再提供一定的分值，所以总和可能超过 10 分。换句话说，你也可以选择一道附加题，跳过一道正常题。
  + 每道习题的给分由助教评判，简述类习题可能存在一定开放性，所以评分也存在主观因素。
  + 请利用深蓝学院系统提交习题。每次习题我们会记通过与否。提交形式为 word 或 pdf 格式报告，如有编程习题请提交可编译的源码。
  + 为方便读者，我通常会准备一些阅读材料，放在 books/或 papers/目录下。请读者按个人需求使用这些材料。它们多数是从网络下载的，如果侵犯到你的权利，请及时告诉我。
  + 每个习题会标注大致用时，但视同学个人水平可能会有出入。
  + 习题的完成情况会影响你对本课程内容的掌握程度，请认真、独立完成。**习题总得分较高的同学将获得推荐资格**。

备注：

* 本习题内容更新于 2021 年 9 月。

1

* **熟悉 Linux (2 分，约 2 小时)**

计算机领域的绝大多数科研人员都在 Linux 下工作，不掌握 Linux 会使你在研究道路上寸步难行。 Linux 系统的基本知识亦是学习本课程的先决条件。如果你还未接触过 Linux，请阅读本次习题中提供的材料（见 books/目录下）。我建议阅读《鸟哥的 Linux 私房菜》第 1、2 章了解 Linux 历史，第 5-8 章了解基础操作。如果你在用 Ubuntu，也可以参考 Ubuntu 维基上自带的指南：[http://wiki.ubuntu.org.cn/](http://wiki.ubuntu.org.cn/Ubuntuæ¡é¢å¥é¨æå)

Ubuntu .

不要把 Linux 想得太困难。现代的 Linux 系统多数具有方便的图形界面，十分容易上手。最好的学习方式可能是马上安装一个 Linux 然后熟悉它的操作界面，多数时候和 Windows/mac 差别不大。我们在本书中使用 Ubuntu 16.04，读者也可以按个人口味选择任意适合你的发行版，不过最好使用 Ubuntu 系列，这样我和你的操作方式会比较相似。

等你熟悉 Linux 后，请回答以下问题（如果你已经很熟悉，就跳过上面的阅读内容，直接回答即可）：

1. 请描述 apt-get 安装软件的整体步骤，说明 Ubuntu 是如何管理软件依赖关系和软件版本的。

答：apt-get 可以从认证软件源下载软件包及相关信息，以便安装和升级软件包，

或者用于移除软件包。在这些过程中，软件包依赖会被妥善处理。我们用apt-get install的时候，实际上是从/var/lib/apt/lists中扫描的得到该软件的最新版本地址，并通过该地址下载该软件包，软件包将会被保存在临时文件夹\*\*/var/cache/apt/archives\*\*下，最后开始安装该软件。/etc/apt/sources.list这个文件指定apt查找或安装某个软件包时要搜索的仓库，里面每一行定义一个仓库。仓库就是存放着一组软件包及其相关信息，包括用来描述每个软件包的头部信息、包之间的依赖关系。Ubuntu仓库里面的软件包可以划分为以下几个类别：

   1）main                #Ubuntu支持的开源软件

   2）universe           #由社区维护的开源软件

   3）multiverse       #存在版权限制或法律问题的软件

   4）restricted         #专有设备驱动程序

   5）backports        #较新的Ubuntu发行中有的但较早发行中不存在的软件包

apt在选择软件包时，根据sources.list文件中指定的类别来搜索各个仓库。

1. 什么是软件源？如何更换系统自带的软件源？如何安装来自第三方软件源中的软件？

答：软件源就是一个应用程序安装库，很多很多的应用软件都在这个库里面。他可以是[网络服务器](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E6%9C%8D%E5%8A%A1%E5%99%A8/99096" \t "/home/wang/Documents\\x/_blank)，是光盘，甚至是硬盘上的一个目录。

打开Ubuntu的显示应用程序，在里面选中软件与更新，在Ubuntusoftware里更换软件源即可。

更换第三方软件源后，在终端输入sudo apt-get update命令，再安装第三方软件源中的软件。

1. 除了 apt-get 以外，还有什么方式在系统中安装所需软件？除了 Ubuntu 以外，其他发行版使用什么软件管理工具？请至少各列举两种。

答：dpkg可以安装下载的安装包；Ubuntu软件商店安装；使用snap命令安装；使用dpkg命令安装deb包；使用源码编译安装。

1.DPKG（适用于Debian）

DPKG,Debian Package Management System, 中文为Debian包管理系统。DPKG是DebianLinux家族的基础包管理系统，它用于安装、删除、存储和提供.deb包的信息。

DPKG前端命令行工具有：  
（1）APT

APT, Advanced Packaging Tool, 中文为高级打包工具， APT是一个 dpkg 包管理系统的前端工具。

（2）Aptitude

Aptitude是Debian Linux家族一个非常出名的命令行前端包管理工具，它工作方式类似APT ，它们之间有很多可以比较的地方。Aptitude最初为 Debian及其衍生版设计的，但是现在它的功能延伸到RHEL家族。

（3）Synaptic

Synaptic是一个基于GTK+的APT的可视化包管理器，对于一些不想使用命令行的用户，它非常好用。

2.RPM（适用于RedHat）

RPM, Red Hat Package Manager, 中文为红帽包管理器。RPM是红帽创建的Linux 基本标准（LSB）打包格式和基础包管理系统。基于这个底层系统，有多个前端包管理工具可供你使用，如下：

（1）YUM

YUM, Yellowdog Updater, Modified, 中文卫黄狗更新器，修改版。YUM是一个开源、流行的命令行包管理器，它是用户使用RPM的界面之一。

（2）DNF

DNF,Dandified Yum, 中文为优美的YUM。DNF是基于RPM的发行版的包管理器，Fedora 18引入了它，它是下一代 YUM。

3.Pacman包管理器（适用于Arch Linux）

Pacman包管理器是一个流行的、强大而易用的包管理器，它用于Arch Linux和其他的一些小众发行版。它提供了一些其他包管理器提供的基本功能，包括安装、自动解决依赖关系、升级、卸载和降级软件。

4.Zypper包管理器（适用于OpenSUSE）

Zypper包管理器是一个使用libzypp库制作的用于OpenSUSE系统上的命令行包管理器，它的常用功能包括访问仓库、安装包、解决依赖问题和其他功能。

5.Portage包管理器（适用于Gentoo）

Portage包管理器是Gentoo的包管理器，当下不怎么流行的一个发行版，但是这并不阻止它成为Linux下最好的软件包管理器之一。

1. 环境变量 PATH 是什么？有什么用途？LD\_LIBRARY\_PATH 是什么？指令 ldconfig 有什么用途？

答：PATH变量就是用于保存可以搜索的目录路径，如果待运行的程序不在当前目录，操作系统便可以去依次搜索PATH变量变量中记录的目录，如果在这些目录中找到待运行的程序，操作系统便可以运行。环境变量是包含关于系统及当前登录用户的环境信息的字符串,一些软件程序使用此信息确定在何 处放置文件(如临时文件). 环境变量说白了就是指定一个软件的路径，比如说配置TomcatJdk等软件时就必须设置环境变量。PATH说简单点就是一个字符串变量，当输入命令的时候LINUX会去查找PATH里面记录的路径。比如在根目录/下可以输入命令ls,在/usr目录下也可以输入ls,但其实ls这个命令根本不在这个两个目录下，事实上当你输入命令的时候LINUX会去/bin,/usr/bin,/sbin等目录下面去找你此时输入的命令，而PATH的值恰恰就是/bin:/sbin:/usr/bin:……。其中的冒号使目录与目录之间隔开。

LD\_LIBRARY\_PATH是Linux环境变量名，该环境变量主要用于指定查找共享库（动态链接库）时除了默认路径之外的其他路径。

ldconfig 命令的用途,主要是在默认搜寻目录(/lib和/usr/lib)以及动态库配置文件/etc/ld.so.conf内所列的目录下,搜索出可共享的动态链接库(格式如前介绍,lib\*.so\*),进而创建出动态装入程序(ld.so)所需的连接和缓存文件

1. Linux 文件权限有哪几种？如何修改一个文件的权限？

答：Linux 文件的基本权限就有九个，分别是owner/group/others(拥有者/组/其他) 三种身份各有自己的read/write/execute 权限。为了保护系统的安全性，Linux 系统对不同的用户访问同一文件（包括目录文件）的权限做了不同的规定。 在Linux 中我们通常使用以下两个命令来修改文件或目录的所属用户与权限： chown (change ownerp) ： 修改所属用户与组。 chmod (change mode) ： 修改用户的权限。

1. Linux 用户和用户组是什么概念？用户组的权限是什么意思？有哪些常见的用户组？

答：

1、用户（user）的概念；  
  
通过前面对Linux 多用户的理解，我们明白Linux 是真正意义上的多用户操作系统，所以我们能在Linux系统中建若干用户（user）。比如我们的同事想用我的计算机，但我不想让他用我的用户名登录，因为我的用户名下有不想让别人看到的资料和信息（也就是隐私内容）这时我就可以给他建一个新的用户名，让他用我所开的用户名去折腾，这从计算机安全角度来说是符合操作规则的；

当然用户（user）的概念理解还不仅仅于此，在Linux系统中还有一些用户是用来完成特定任务的，比如nobody和ftp 等，我们访问LinuxSir.Org 的网页程序，就是nobody用户；我们匿名访问ftp 时，会用到用户ftp或nobody ；如果您想了解Linux系统的一些帐号，请查看 /etc/passwd ；

2、用户组（group）的概念；  
  
用户组（group）就是具有相同特征的用户（user）的集合体；比如有时我们要让多个用户具有相同的权限，比如查看、修改某一文件或执行某个命令，这时我们需要用户组，我们把用户都定义到同一用户组，我们通过修改文件或目录的权限，让用户组具有一定的操作权限，这样用户组下的用户对该文件或目录都具有相同的权限，这是我们通过定义组和修改文件的权限来实现的；

举例：我们为了让一些用户有权限查看某一文档，比如是一个时间表，而编写时间表的人要具有读写执行的权限，我们想让一些用户知道这个时间表的内容，而不让他们修改，所以我们可以把这些用户都划到一个组，然后来修改这个文件的权限，让用户组可读，这样用户组下面的每个用户都是可读的；

由于系统文件的用户太多为了便于管理 我们便把系统的用户 进行分类 赋予权限 合理地分配资源 所以今天介绍的是 系统用户 用户组 的一些权限的 分配 修改 和文件的 权限修改 和赋权 以及一些特殊权限的用法。

常见的用户组，其中包括administrators、backup operators、guests、power users、replicator还有users用户，默认新建立的用户属于users组、也就是everyone。其中administrators组的用 户具有与administrator相同的权限。

1. 常见的 Linux 下 C++ 编译器有哪几种？在你的机器上，默认用的是哪一种？它能够支持 C++ 的哪个标准？

答：gcc，g++，cmake，我的机器上默认的是g++，它能支持c++17标准。

* **SLAM 综述文献阅读 (2 分，约 3 小时)**

当你对某个研究领域不了解时，最好是从综述文献开始了解这个领域的整体面貌。SLAM 作为一个近 30 年的研究领域，至今也存在着大量的综述、总结类的文章。请阅读本次作业 paper/目录下的文章 [1–3] （其中 [3] 是中文文献），了解这个领域的大致情况。如果你的时间有限，可以仅阅读每篇文章的第一章（也就是引言一章），然后回答下列问题：

1. SLAM 会在哪些场合中用到？至少列举三个方向。

答：移动机器人，无人机，自动驾驶

1. 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？

答：完美的定位需要用到一个无偏差的地图；但这样的地图又需要精确的位置估测来描绘.定位是需要周围环境的信息的，是借助周围地图信息来定位的； 建图又需要准确知道目前摄像机随着机器人的自身位置； 准确的定位需要准确的地图； 精准的地图构建又需要精准的位置信息；  
是一个很耦合的问题。 所以，定位的同时需要建图。

1. SLAM 发展历史如何？我们可以将它划分成哪几个阶段？

答：（1）第一阶段：定位和建图分开研究的阶段；

（2）第二阶段：定位和建图在一起进行研究的阶段；

（3）第三阶段：开始运用摄像头的阶段（这里需要注意的是，之前由于计算机视觉的发展还不太到位，因此之前基本都是基于激光测距这种方法的）

（4）第四阶段：多元化的阶段（我之所以把这个阶段叫做多元化的阶段，是因为SLAM和深度学习的融合，视觉和惯性传感器的融合等解决方案不断的涌现，对于SLAM的发展可以说是起了极大的加速作用，也使得SLAM这个已经好几十年的老话题在今天又被大家重新认识，且重新投入努力和热情去对待。

1. 从什么时候开始 SLAM 区分为前端和后端？为什么我们要把 SLAM 区分为前端和后端？

答：前端是进行特征提取，获取相邻帧的位姿变换，而后端则是进行地图的拼接和位姿优化。

在特定的机器人应用中，或许很难直接写出传感器的测量结果，如状态的分析函数，正如MAP估计中需要使用的。举个例子，如果未处理的传感器数据是一个图像，可能很难将每个像素的强度表达为SLAM的状态函数；对于更简单的传感器也会出现同样的困难（例如，一个单光束的激光）。在这两个情况中，该问题与我们无法设计出一个充分一般的表达式来表述环境的事实有关。甚至是一个十分一般的表达式，都很难写出该表达式的与测量有关的参数。

出于这个原因，在SLAM后端之前，通常需要有一个模块，称之为前端，可以从传感器数据中提取出相关的特征。例如，在视觉SLAM中，前端从环境中提取出几个可区分点的像素位置；观测到这些点的像素现在就可以很容易地在后端中建模。前端还负责将每个测量结果与环境中的特定地标（如三维点）相关联：因此也称之为数据关联。

1. 列举三篇在 SLAM 领域的经典文献

答：《LSD-SLAM: Large-Scale Direct Monocular SLAM》

《ORB-SLAM:ORB-SLAM: a Versatile and Accurate Monocular SLAM System》

《VINS-Mono: A Robust and Versatile Monocular Visual Inertial State Estimator

* **CMake 练习 （2 分，约 1.5 小时）**

cmake 是一种常用、方便的，用于组织 Linux 下 C++ 程序的工具。有许多库，例如 OpenCV、g2o、 Ceres 等，都用 cmake 组织它们的工程。所以，不管是使用别人的库，还是编写自己的库，都需要掌握一

些 cmake 的基本知识。也许你之前没有听过这个工具，但不要紧，我们准备了阅读材料“books/Cmake Practice.pdf”（cmake 实践，由一位北大同学撰写）。请阅读此文的第 1 至 6 章，并完成以下工作：

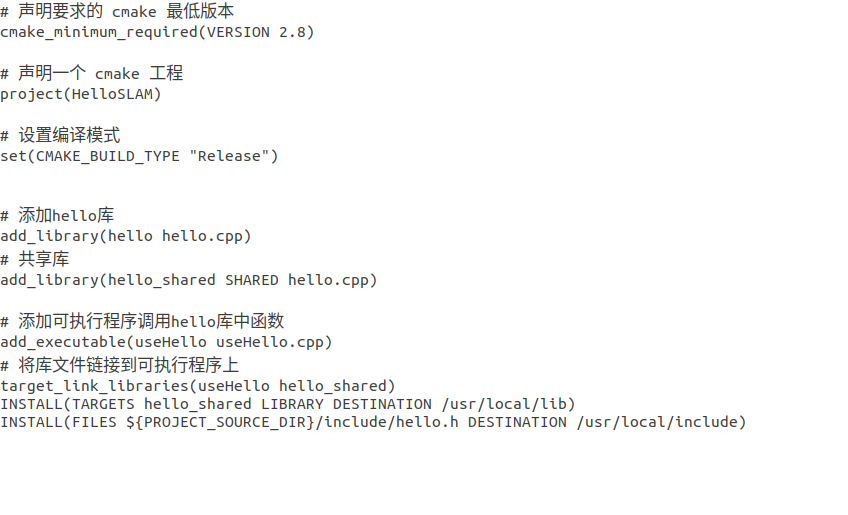
书写一个由 cmake 组织的 C++ 工程，要求如下：

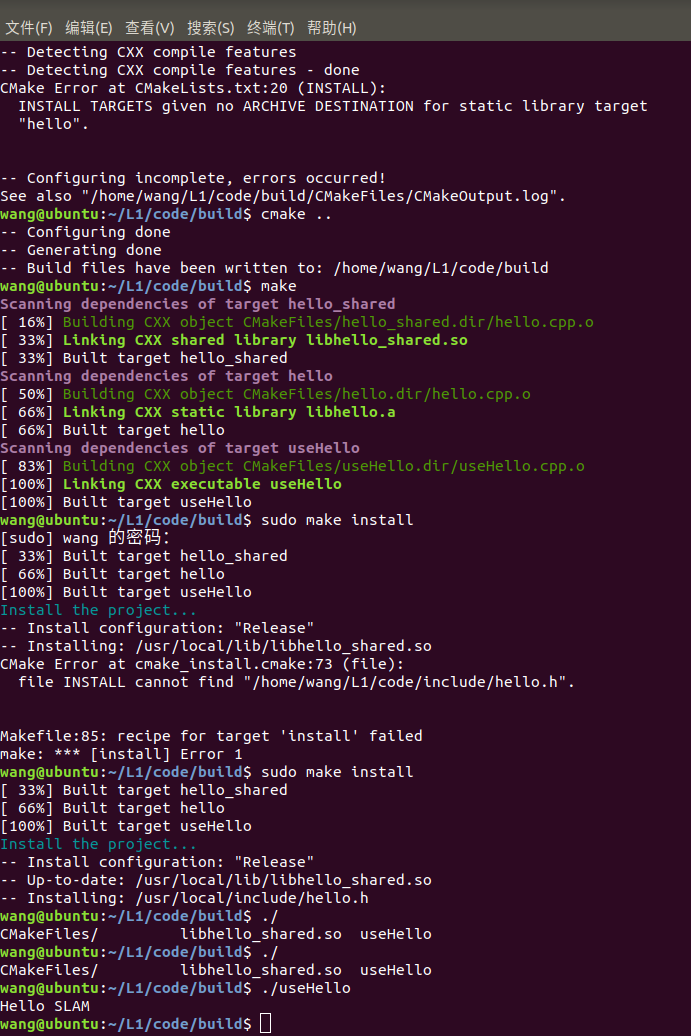
* 1. include/hello.h 和 src/hello.c 构成了 libhello.so 库。hello.c 中提供一个函数 sayHello()，调用此

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。

1. 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”。
2. 默认用 Release 模式编译这个工程。
3. 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。
4. 为你的库提供 FindHello.cmake 文件，让其他用户可以通过 find\_package 命令找到你的库，并实际测试你的程序确实可以这样做。

请按照上述要求组织源代码文件，并书写 CMakeLists.txt。

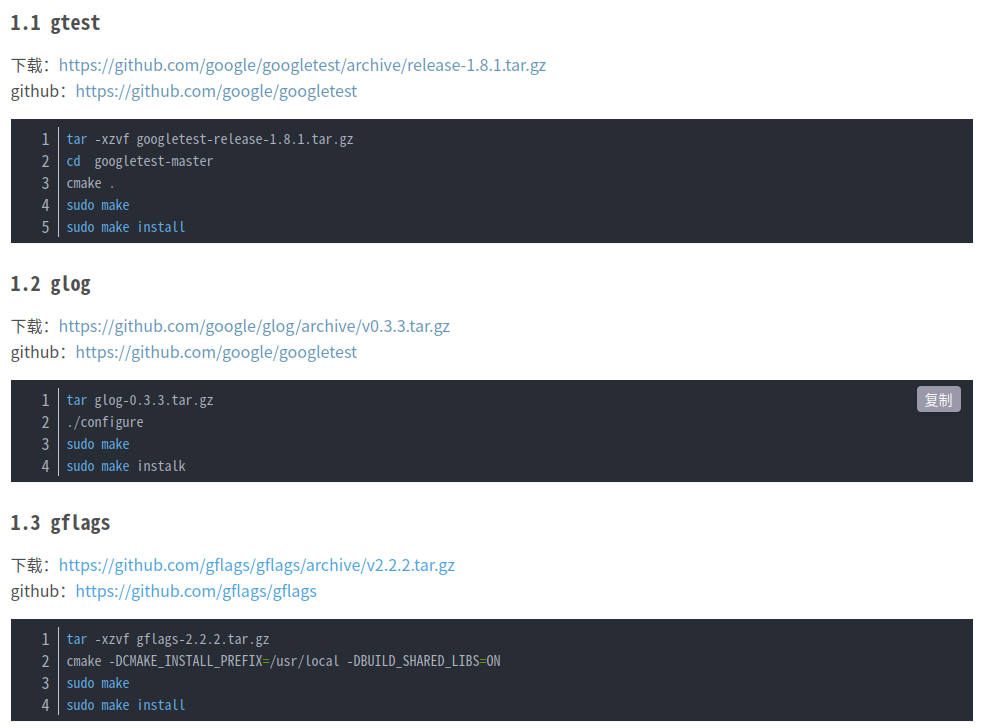


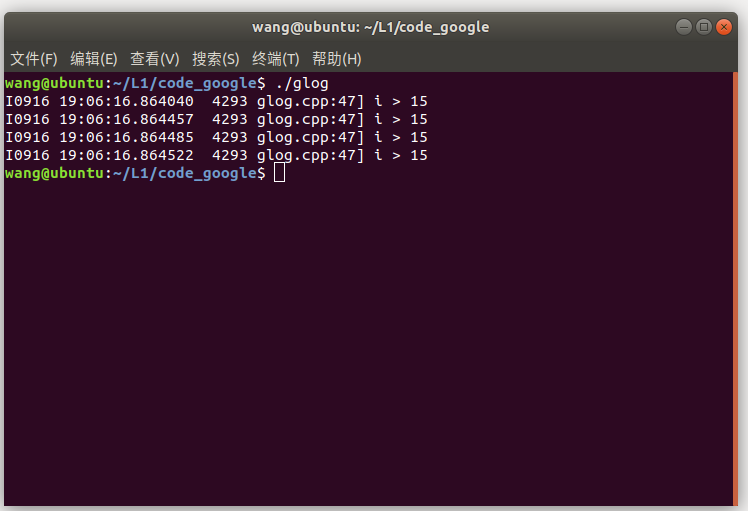


* **gflags, glog, gtest 的使用 （2 分，约 2 小时）**

Google 提供了一系列非常好用的 C++ 工具，例如 gflags, glog, gtest 这三件套。这些程序在很多工程应用中都会用到。我们趁这个机会来熟悉它们。glog 是日志打印工具，gflags 是参数管理工具，而 gtest 是单元测试工具。当我们在开发自己的应用时，可以灵活地使用这些三方库，加快算法的开发效率。

1. 请自行寻找这三个库的说明文档，并在系统中安装它们。请说明你是如何安装的。

答：



1. 将上一题中的打印改为使用 glog 的打印方式，以替代 std::cout 的输出方式。
2. 在 useHello.c 中增加一个 gflags 以指明打印的次数 print\_times，默认为 1。当用户传递该参数时，即打印多少遍 Hello SLAM。
3. 书写一个 gtest 单元测试程序来测试你的工程能够正常运行。修改你的 CMakeLists.txt 来增加这个单元测试。

请提交你的代码和运行结果。

* **理解 ORB-SLAM2 框架 （2 分，约 2 小时）**

ORB-SLAM2[4] 是一个非常经典的视觉 SLAM 开源方案，它可以作为你学习 SLAM 的范本。但是现在我们还没有讲解很多关于视觉 SLAM 的知识，所以仅从代码工程角度上来了解 ORB-SLAM2。请按照提示完成以下工作。

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。

1. 此时我们不着急直接运行 ORB-SLAM2，让我们首先来看它的代码结构。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：
   1. ORB-SLAM2 将编译出什么结果？有几个库文件和可执行文件？

答：将编译出可执行文件和库文件：可执行文件6个和1个库文件

库文件：通过/src文件夹中的.cc文件生成动态库文件ORB\_SLAM2  
可执行文件：rgbd\_tum、stereo\_kitti、stereo\_euroc、mono\_tum、mono\_kitti、mono\_euroc

* 1. ORB-SLAM2 中的 include, src, Examples 三个文件夹中都含有什么内容？

答：include文件夹包含：对应src中程序的代码函数头文件  
src文件夹包含：相应程序的代码函数的c++文件  
Examples文件夹包含：对应不同类别的相机的主程序

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

答：（1）OPENCV\_LIBS

（2）EIGEN3\_LIBS

（3）Pangolin\_LIBRARIES

（4）/ORB-SLAM2/Thirdparty/DBoW2/lib/libDBoW2.so

（5）/ORB-SLAM2/Thirdparty/g2o/lib/libg2o.so

你会发现 ORB-SLAM2 从代码组织方式来看并不复杂。实际上大部分中小型库都不会很复杂，而更大的库可能在 CMakeLists.txt 中有各种各样的兼容性检查，确保它们在各个平台上都能顺利运行。

现在你已经了解了 ORB-SLAM2 的代码结构了。抛开代码内容来说，至少你已经知道如何编译，使用这个库了。ORB-SLAM2 可以在数据集上运行，也可以在实际的摄像头上运行。下面的作业将指导你用自己的笔记本摄像头来运行 ORB-SLAM2。

* **\* 使用摄像头或视频运行 ORB-SLAM2（3 分，约 1 小时）**

请注意本题为附加题。

了解一样东西最快的方式是自己上手使用它，不要担心弄坏你的笔记本，大部分时候它都是你可靠的伙伴。这个作业中，我将指导你用自己的笔记本摄像头读取到的图像，来运行 ORB-SLAM2，看看它能不能实际工作。你也可以外接一个 usb 摄像头，这会让你的手更加灵活一些（不用费力端着笔记本到处跑）。或者，如果你的电脑碰巧没有摄像头/摄像头故障了/你正在用虚拟机，那我们也可以在事先录制好的一段视频中运行 ORB-SLAM2（见 code/myvideo.mp4，这是我在特蕾西亚草坪散步的时候用手机拍摄的小视频）。

由于我们还没有讲过任何关于 OpenCV 或者图像方面的问题，所以本节我给你写好了一个 myslam.cpp 文件（如果你使用录制视频，请用 myvideo.cpp）。这个文件会打开你自带的摄像头（或视频），读取图像，并交给 ORB-SLAM2 处理。由于你现在已经了解 cmake 原理了，所以我要请你自己来思考如何将这个文件与 ORB-SLAM2 结合起来。相信我，这件事并不难。myslam.cpp 和 myvideo.cpp 文件见本次作业的code/文件夹下。

下面是本题的提示：

1. 为了实际运行 ORB-SLAM2，你需要安装它的依赖项，并通过它本身的编译。它的依赖项见它自

己的 github 主页，请按照主页上的提示安装好 ORB-SLAM2 的依赖项。具体来说，对于 pangolin （一个 GUI 库），你需要下载并安装它，它同样是个 cmake 工程，所以我不必谈怎么编译安装的细节了。对于 opencv 和 eigen3，你可以简单的用一行命令来解决：

1

sudo apt-get install libopencv-dev libeigen3-dev libqt4-dev qt4-qmake libqglviewer-dev libsuitesparse-dev libcxsparse3.1.2 libcholmod-dev

其中一部分是 g2o 的依赖项，现阶段不用太在意它的具体内容。至此，你应该可以顺利编译 ORB-SLAM2 了，请给出它编译完成的截图。

1. 注意到，ORB-SLAM2 提供了若干数据集中的运行示例，这可以作为我们运行自己摄像头程序的参考，因为它们很相似。对于数据集上的示例，ORB-SLAM2 会首先读取数据集中的图像，再放到 SLAM 中处理。那么对于我们自己的摄像头，同样可以这样处理。所以最方便的方案是直接将我们的程序作为一个新的可执行程序，加入到 ORB-SLAM2 工程中。那么请问，如何将 myslam.cpp

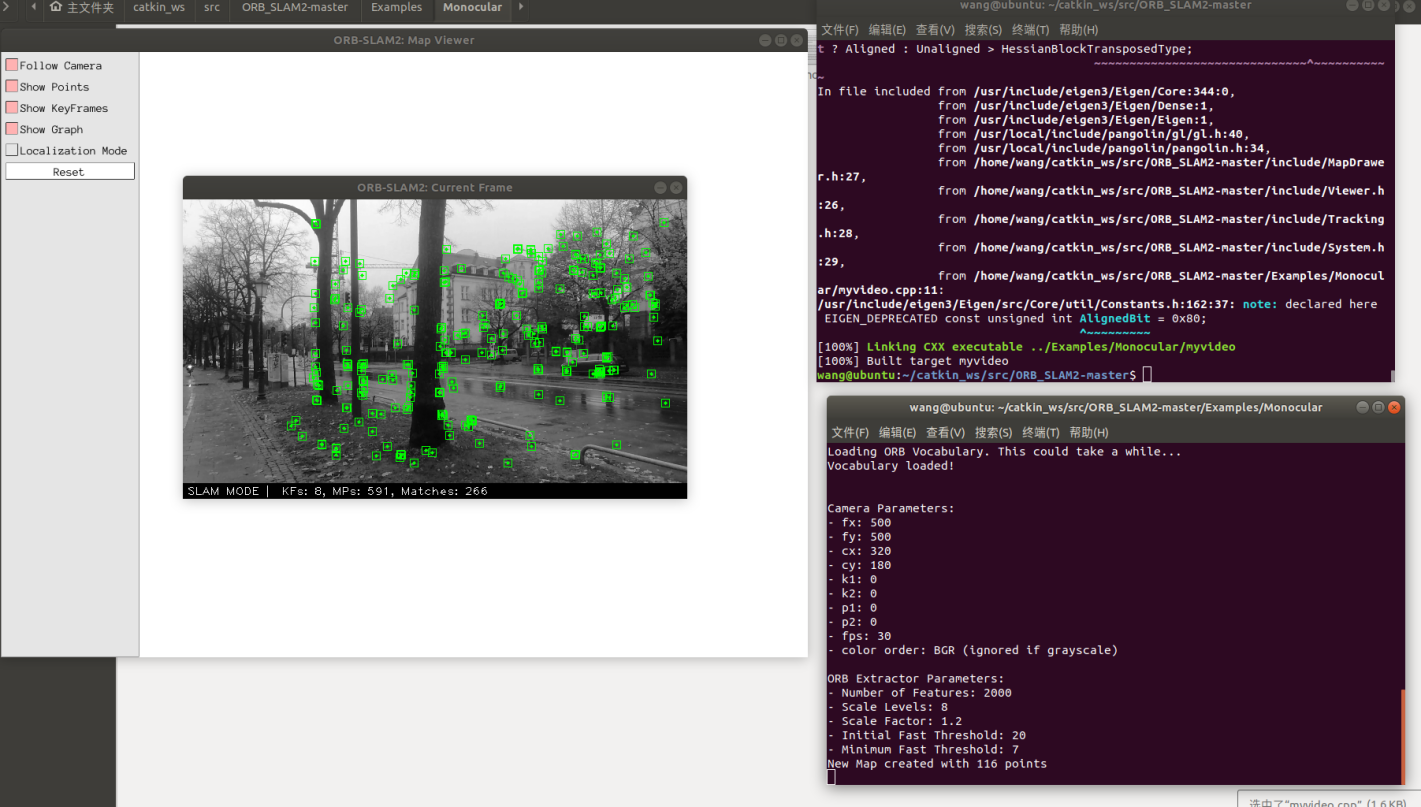
或 myvideo.cpp 加入到 ORB-SLAM2 工程中？请给出你的 CMakeLists.txt 修改方案。

1. 现在你的程序应该可以编译出结果了。但是我们现在还没有谈相机标定，所以你还没办法标定你的摄像头。但没有关系，我们也可以用一个不那么好的标定参数，先来试一试效果（所幸 ORB-SLAM2 对标定参数不太敏感）。我为你提供了一个 myslam.yaml（myvideo.yaml），这个文件是我们假想

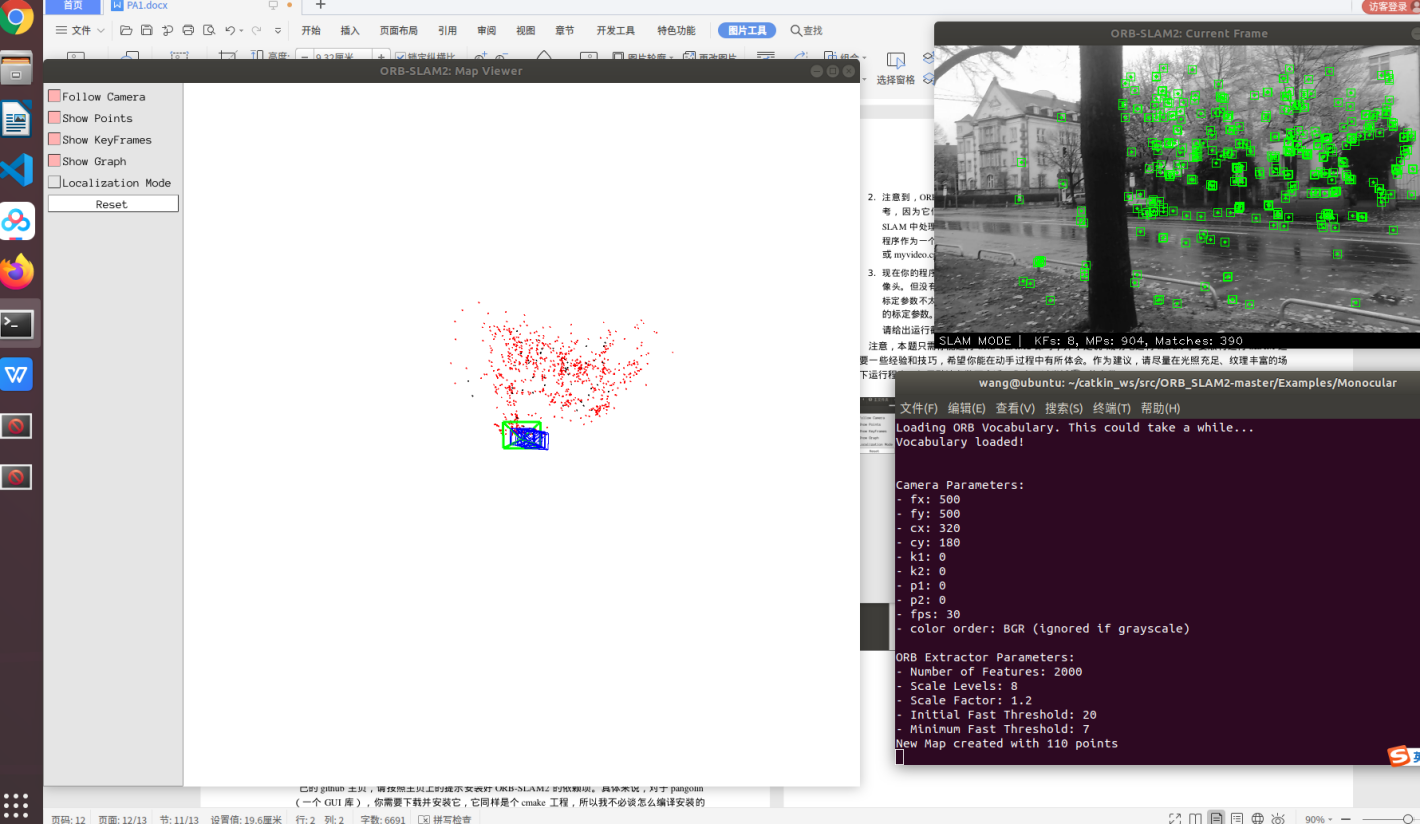
的标定参数。现在，用这个文件让 ORB-SLAM2 运行起来，看看 ORB-SLAM2 的实际效果吧。

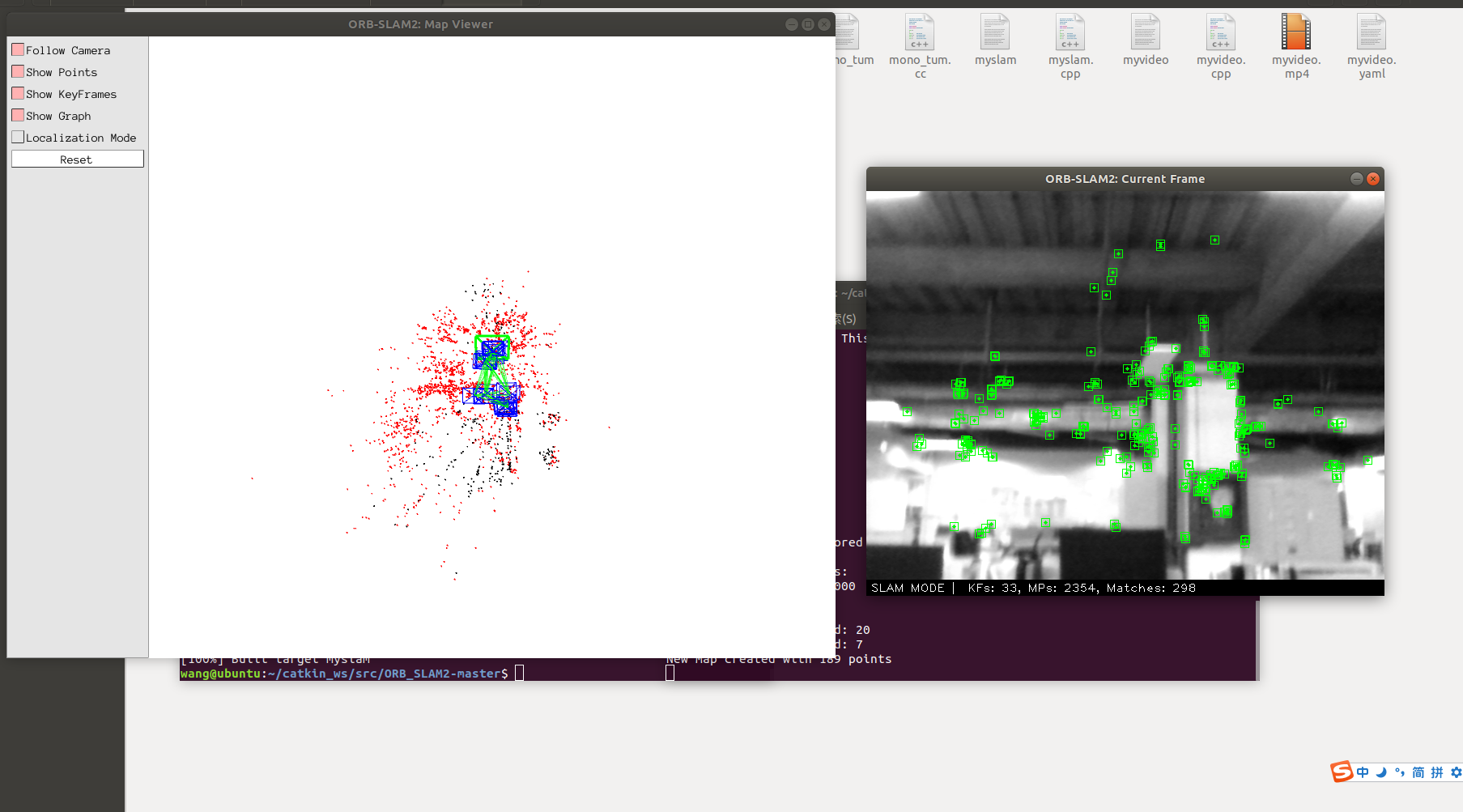
请给出运行截图，并谈谈你在运行过程中的体会。

注意，本题只需你能运行 ORB-SLAM2 即可，并不是说“成功地运行 SLAM”。要顺利运行 SLAM 还需要一些经验和技巧，希望你能在动手过程中有所体会。作为建议，请尽量在光照充足、纹理丰富的场合下运行程序。如果默认参数不合适，你也可以尝试换一换参数。



7





**Bibliography**

1. C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
2. J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.
3. L. Haomin, Z. Guofeng, and B. Hujun, “A survey of monocular simultaneous localization and mapping,” *Journal of Computer-Aided Design and Compute Graphics*, vol. 28, no. 6, pp. 855–868, 2016. in Chinese.
4. R. Mur-Artal, J. Montiel, and J. D. Tardós, “Orb-slam: a versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

8