

第一节课习题

高翔

2019 年 6 月 14 日

1 习题说明

- 第 i 节课习题所有材料打包在 $L_i.zip$ 中, $\forall i = 1 \dots 8$ 。
- 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。
- 每个习题会有一些的分值。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。

2 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/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. 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?
2. linux 的环境变量是什么? 我如何定义新的环境变量?
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。
4. 假设我要给 a.sh 加上可执行权限, 该输入什么命令?
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?

3 SLAM 综述文献阅读 (3 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。
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?
3. SLAM 发展历史如何? 我们可以将它划分成哪几个阶段?
4. 列举三篇在 SLAM 领域的经典文献。

4 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。
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。

5 理解 ORB-SLAM2 框架 (3 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](https://github.com/raulmur/ORB_SLAM2) 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. 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:
 - (a) ORB-SLAM2 将编译出什么结果? 有几个库文件和可执行文件?
 - (b) ORB-SLAM2 中的 include, src, Examples 三个文件夹中都含有什么内容?
 - (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?

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运行。下面的作业将指导你用你自己的笔记本摄像头来运行 ORB-SLAM2。

a) 库文件 : 19+5 (第三方库) 可执行 : 6

b) include: header files

src: .cc文件 (Unix系统常用c++源文件)

examples: 4个不同针对不同摄像头/场景的例子 (Monocular, RGB-D, ROS, Stereo)

c) 链接到了OpenCV_LIBS, EIGEN3_LIBS, Pangolin_LIBRARIES, libDBow2.so, libg2o.so

6 * 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，我将指导你用你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
libcxspase3.1.2 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。

2. 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。所以最方便的方法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。
3. 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。

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.