A Report of Final Project

KITTI 的双目视觉里程计设计

Author：Vance 吴方熠

School：福州大学

Date ：2018.50.20

深蓝学院视觉 SLAM 理论与实践（第二期）

目　录

1. 相关说明

1）本次大作业由本人独自完成。

2）项目工程以 slambook 中的 project 0.4 为原型框架，在其基础上修改补充完成，具体的工作量和工作内容见开发文档/code/NOTE.md.

3）由于本人最近一段时间忙于课题组的项目，从开始做大作业到结束只花了不到 10 天的时间，所以实现的效果有限，只完成了一个基础的 VO，在运行一段时间后由于累计误差过大程序会中止。

4）由于程序效果不太好，所以没有浪费很多精力去测试每一个序列，仅在序列 00、01、05、07、09 上测试过。

5)其中序列 01 上的跟踪和匹配效果相对较好，以下涉及的相关数据若无特别说明的话，都以序列 01 的双目灰度图输入作为数据来源。

2. 项目框架搭建

由于本工程以 slambook 中的 project 0.4 为原型框架搭建而来，故在此将引用《视觉 SLAM 十四讲》的部分内容。

2.1 文档内容

工程的目录如图 1 所示：

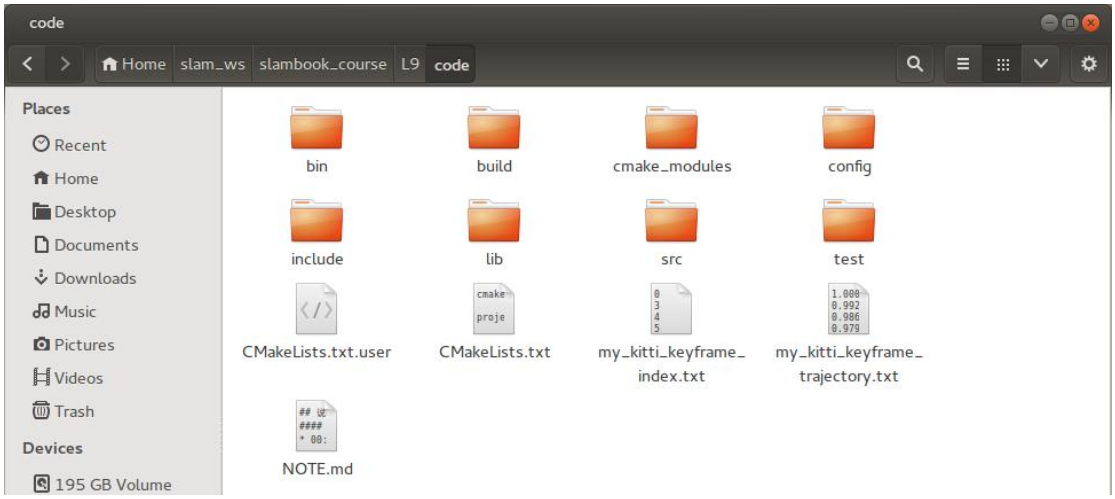

图 1 工程的目录结构

几个文件夹及文件的作用说明：

- bin：放置工程生成的二进制可执行文件；
- build：构建目录；
- cmake_modules：放置 CMake 模块文件，用于寻找第三方库；
- config：放置配置文件；
- include：放置工程的头文件；
- lib：放置工程生成的库文件；
- src：放置工程的源文件和头文件的代码实现；
- test：放置相关测试源文件；
- 两个轨迹文件：一个记录关键帧的位姿，一个记录关键帧的序号；
- NOTE.md：工程开发日志。

2.2 基本类的关系

VisualOdometry 类（以下简称 VO 类）是整个工程的核心部分，是工程的执行基础。它包含了两个 Frame 类（其中一个为当前帧，另一个为参考帧）和一个 Map 类。Map 类包含了 KeyFrame 类和 MapPoint 类，每个 Frame 类又包含了一个 Camera 类。Config 类作为工程中一个独立的辅助类，实现了参数的读取和设置等功能。几个类之间的包含关系如图 2 所示。

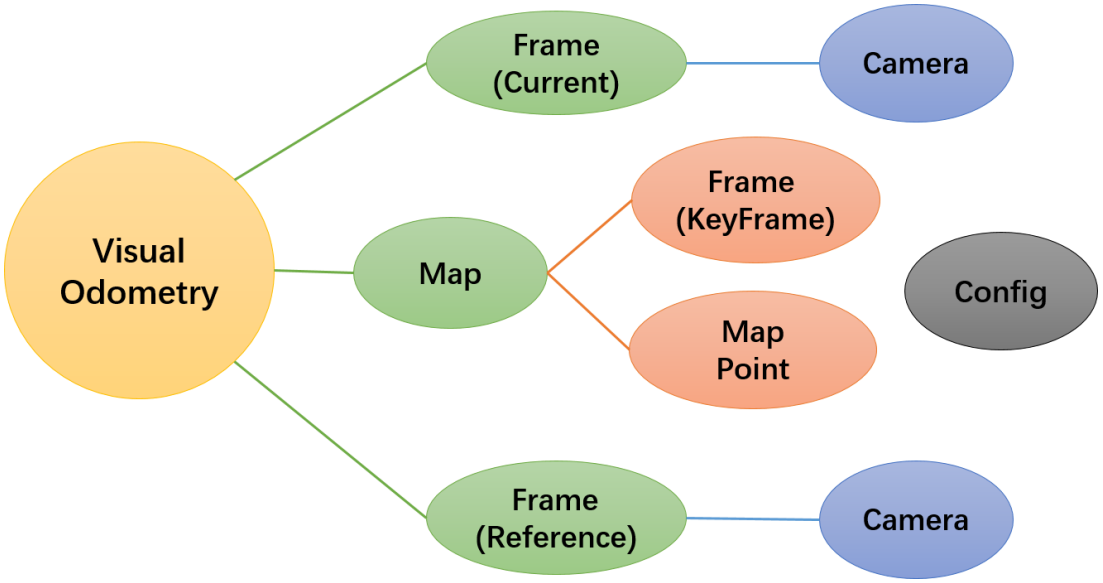

图 2 几个类直接的包含关系

2.3 各类的主要作用

从简单的说起。

- Config 类已经在 2.2 中介绍过了，这里就不在赘述。
- Camera 类存储着相机的内参、畸变参数、图像尺寸和基线距离等数据，此外还提供了各个坐标系之间的坐标转换功能。
- MapPoint 类携带了一个地图点世界坐标、描述子、该点属于的 Frame 帧数等信息，是局部地图 Map 中的点。
- Map 类管理了所有的地图点 MapPoint 以及所有的关键帧。
- Frame 类作为整个系统的基础单元，负责着数据的输入工作。每帧传入的图像在这里被保存，并计算出特征点和对应的描述子，这些预处理都是为了最后的 VisualOdometry 类服务。
- VisualOdometry 类是整个工程的核心，它对将当前帧传入的特征点与局部地图中的地图点进行匹配，并用 PnP 算法估计当前帧的相机位姿，然后对其进行优化，最后再对优化后的结果进行判断，如果结果合理，则将此结果记为当前帧的位姿，并将当前帧记为关键帧，根据匹配情况更新局部地图。如果结果不合理，则抛弃该帧继续计算下一帧，直到处理完所有数据或连续出现不合理的帧数达到了阈值。

3. 重点部分的实现介绍

3.1 Frame 类

Void Frame::computeStereoMatches()成员函数负责计算左右两幅图像的特征点和匹配情况，并筛选出匹配正确的特征点存储（因为只有正确的匹配才能保证此特征点通过视差计算出的深度值可信，没有匹配的特征点无法通过视差算出深度）。这里的难点在于保证匹配关系的正确与否。所以在第一次暴力匹配后，本人还加入了 RANSAC 算法，通过 OpenCV 的 findHomography 函数，完成了左右图像的精确匹配。

void Frame::computeDepth()成员函数负责依照左右两幅图正确的匹配关系，依照视差计算特征点的深度值，并且对深度值进行了简单的筛选，对视差为0或深度值太大的点进行剔除，最后再更新一下保留下的特征点以及其对应的描述子。

3.2 VisualOdometry类

bool VisualOdometry::addFrame(Frame::Ptr frame)成员函数的输入数据为预处理过的普通帧Frame。它的实质是一个状态机，每次输入的帧（即当前帧）由当前系统的状态（初始化INITIALIZING / 正常运行OK / 已丢失LOST）决定对其的处理方式。当系统处于INITIALIZING，即第一帧传入时，VO将其定义为关键帧，并将其特征点加入到局部地图中，再将状态改为OK；当系统处于OK时，VO将尝试对当前帧的特征点和局部地图中的地图点进行匹配，并估计当前帧的位姿，最后再对估计的位姿进行判断；当系统处于LOST状态时，停止程序。

void VisualOdometry::featureMatching()成员函数的功能是对当前帧的特征点和局部地图的地图点进行匹配。在匹配前，程序先对各地图点进行判断，判断其是否在当前帧的视野内，若在，则将此点标记为候选点。程序再通过FLANN算法对候选点和特征点进行匹配，筛选出匹配点，结束。

void poseEstimationPnP()成员函数根据上一步中的匹配情况，提取点对的像素坐标值，再用OpenCV的solvePnPRansas()估计当前帧的位姿和局内点数量。对估计出的位姿还要进一步通过和局内点的捆集调整（即Bundle Ajustment, BA）进行优化。

void optimizeMap()成员函数在当前帧的位姿计算合理时，对局部地图进行更新。它会移除到很久没有被观测到的地图点，会再匹配点数较少时添加新的地图点，会在局部地图规模过大时提高筛选比率以剔除更多的地图点，从而使局部地图的地图点保持在一个恒定的规模。

4. 运行结果

本工程在特征点匹配方面由于引入了 RANSAC 法降低了误匹配，所以其左右图像的特征点匹配效果很好，如图 3 所示，几乎没有误匹配。

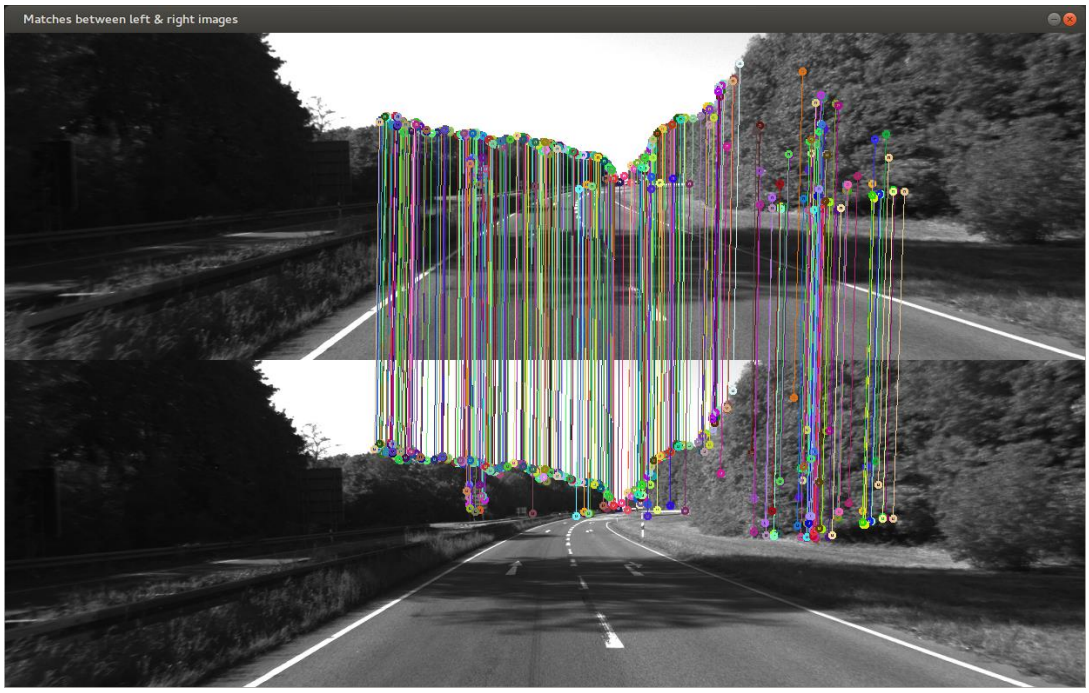

图 3 序列 01 的某帧左右图像特征点匹配效果

本工程在地图点的可视化方面，加入文字描述，可以从图像上看出当前的帧数和当前的地图点数量，如图 4 所示。

图 4 序列 01 的某帧地图点显示效果

本工程在序列 01 上运行效果较好，可以跟踪到 180 多帧。但是由于序列的场景在高速路上，速度较快，特征点集中在天空和树的交界处，累计误差有些大，故实际轨迹的匹配效果（见图 5）并不是很好。注：以下轨迹比对图绿色均代表真实值，红色均代表估计值。

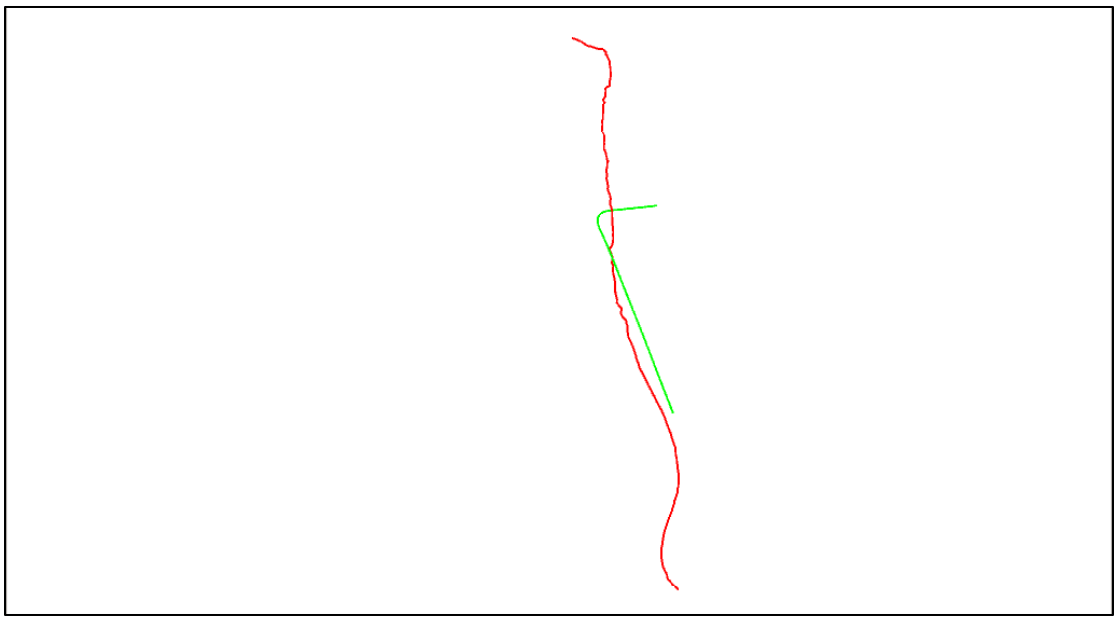

图 5 序列 01 的关键帧轨迹匹配效果

对于其他序列，跟踪效果一般，跟踪的帧数一般不超过 50 帧，时常在转弯时由于运动过大而丢失，或因为特征点筛选数量过少造成匹配不足而丢失。其他序列的跟踪效果如图 6-8 所示。

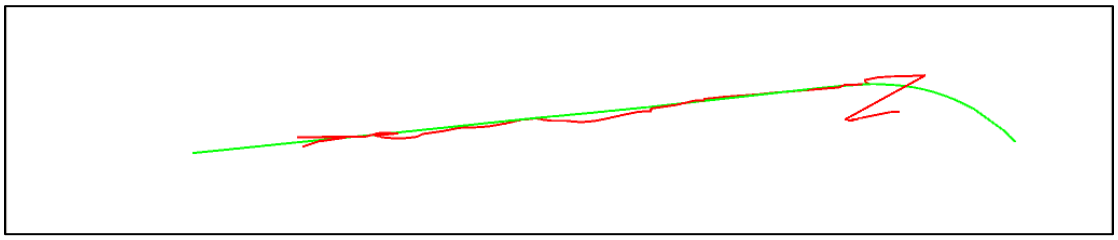

图 6 序列 05 的关键帧轨迹匹配效果

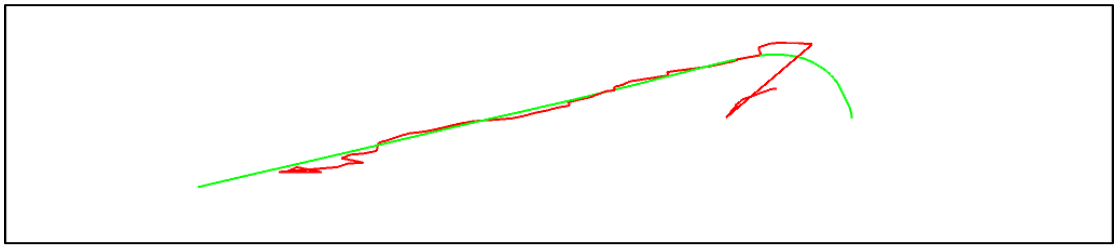

图 7 序列 07 的关键帧轨迹匹配效果

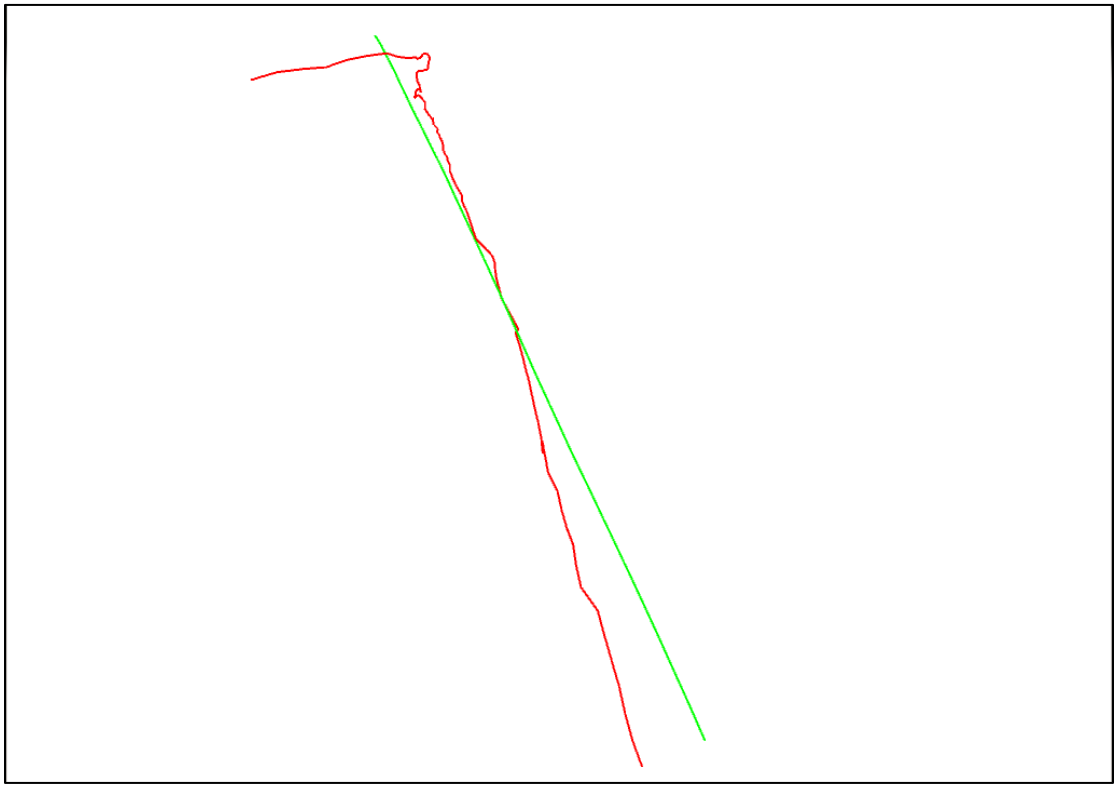

图 8 序列 00 的关键帧轨迹匹配效果

程序每帧的处理时间约在 0.095 至 0.13s，其中特征的提取与匹配占了 90%以上的时间。每帧有效的精确匹配点数在 250-450 个。

本工程仅在序列 00、01、05、07 和 09 上做个测试，由于效果普遍不太好，所以没有打算对其他的序列也做测试，也没有必要计算 ate 和 rpe.

本人录制了本工程在序列 01 上的运行情况视频，详见附件（vedio/ KITTI 序列 01 双目 VO 运行录屏.mp4）。

5. 总结

5.1 本工程的不足之处

1）特征点计算采用的是 OpenCV 自带的函数，特征点采集不够均匀，而且容易扎堆，使得特征点之间的匹配容易混乱（肉眼不容易看出来）；

2）特征点对动态物体的筛选效果不太好，序列 01 中因为将汽车的点作为 MapPoint 造成丢失；

3）程序无法消除累计误差，无法长时间运行；

4）没有回环检测和建图模块，不算一个完整的 SLAM。

5.2 本工程可以改进的地方

1）可以借鉴 ORB_SLAM2 的方式，特征提取时将图像分成若干个 cell 提取特征再统一到一起，这样可以使特征点分布更加均匀；

2）可以加入词袋加速匹配并且引入回环机制；

3）可以考虑用光流法来代替特征点法，不仅可以加速匹配，还可以获取更多的特征点。

5.3 实践心得

由于本人最近一段时间较忙，就没有打算组队坑别人了，所以这个工程是本人自己一个人完成的。另一方面从开始做大作业到结束所花的时间也不到 10 天，所以效果有限，只完成了一个基础的 VO。又因为本工程以书中的 project 0.4 为框架，在其基础上修改补充而成，所以看起来工作量也没有很大。

但是在这次的大作业完成过程中，依然碰到了很多难题。本次工程涉及的理论基础已经懂得差不多了，但在代码的实现上又总会跑出各种各样的 BUG。

- Unordered_map 没见过，遇到了指针溢出，折腾了半天，终于解决了；
- 程序跑一半突然 Segmentation Fault，搞了一个工具去查内存泄漏；
- 程序编译没问题，Debug 找不到错误，这样的情况最可怕，要把程序从头到尾理一遍，判断有可能出现坏数据的位置，然后再逐步测试……

也许在内行的人看来这些工作量的确是没什么，但我庆幸自己有这样的一个过程，让我

- 知道一个项目的落地离不开团队的合作、理论的严谨、代码能力和经验、善于调试和解决 BUG 的能力等等；
- 知道了要写一个项目要先出框架，分模块，再逐一实现，最后整体测试；
- 知道了写工程要有版本管理的意识，要学会记录开发节点，要有一个良好的代码风格。

感谢高翔博士的教学和群里各位友人的指点，经过这次课程和作业的摧残，算是入门了吧，感恩！