说明：重装unbuntu后从零开始复现orb\_slam2的总结：

吴博注释代码：<http://git.oschina.net/paopaoslam/ORB-SLAM2>

PPT网站：<https://wenku.baidu.com/view/8cf70ca7bdeb19e8b8f67c1cfad6195f312be813.html>

其他相关网站：<http://paopaorobot.org/420.html>

解压缩：<https://www.linuxidc.com/Linux/2012-08/68122.htm>

tgz：<https://blog.csdn.net/weixin_40533355/article/details/80473223>

数据库云盘：<https://www.sohu.com/a/219232053_715754>

使用自己总结文件：xzbots和廖雪峰Git summary；Linux and Ubuntu；树莓派：linux常用命令以及vim编辑器（命令图）；视觉SLAM十四讲-高翔

相关文献，总结看周报：

* 初步版本-小改版本-slam2：ORB-SLAM Tracking and Mapping Recognizable；ORB-SLAM A Versatile and Accurate Monocular SLAM System；ORBSLAM2\_An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras
* Fast角点：(OFAST1\_CORR)Machine\_Learning\_for\_High-Speed\_Corner\_Detection；(OFAST2)Faster and Better\_A Machine Learning approach to corner detection
* 描述子：Brief\_Binary robust independent elementary features
* ORB特征：ORB\_an efficient alternative to SIFT or SURF
* 词袋：Bags of Binary Words for Fast Place Recognition in Image Sequences
* 词袋相似度计算（需采集数据集大量特征，十四讲P309）：Video google: A text retrieval approach to object matching in videos；Understanding inverse document frequency: on theoretical arguments for idf。
* 图优化：g2o\_A General Framework for Graph Optimization
* 闭环检测和重定位方法：Fast Relocalisation and Loop Closing in Keyframe-Based SLAM
* 3D-2D：重定位时获得当前帧的初始位姿：EPnP An accurate On solution to the PnP problem
* 闭环检测？由3D-3D点（重定位时用到）求解R、T: Least-Squares Fitting of Two 3-D Point Sets
* 在跟踪线程中局部地图优化时用到相似于double window的方法：double window optimisation for constant time visual SLAM
* 闭环检测计算相似变换、pose graph optimization：Scale Drift-Aware Large Scale Monocular SLAM；closed-form solution of absolute orientation using unit quaternions
* H分解出t、R：Motion and structure from motion in a piecewise planar environment
* E的SVD分解：Multiple View Geometry in Computer Vision.
* RANSAC：Random sample consensus:a paradigm for model fitting with applications to image analysis and automated cartography

待通过实验解决：

* 不同层会不会提取到相同的特征点？提取到的特征点会不会匹配上？若匹配上是不是意味着不确定在哪看到的点了？
* Searchbybow函数中未考虑多次的匹配的异常点，可以设计函数放于continue之前计算分数
* 跑同样的轨迹有时会丢失有时不会

问题

* 总结各种已知条件求解方法，如已知3D点投影到当前帧，若已有3D点如何替换
* 静止时时也会初始化：窗的边框和墙边缘会发生匹配导致初始化
* 自己按当前环境建的词袋和作者建的词袋会有什么区别？

客观来看：词数是2^256=1.157921\*10^77,作者建的词袋是：；因为是二进制，不同环境还是会有很多相同的，且可以通过距离近似；但词的权重会不一样。

* 注意相似变换的是哪到哪的变换？应该是闭环帧到闭环参考帧的变换
* 在lost处作标记
* 期望特征数与金字塔尺度关系

解决：见代码

* 用滑窗提取特征点的作用

解决：更加均匀，特征之间相距更远，从线性方程来说各方程更不相关，没那么病态

* 索引值为何经常初始化为-1，

解决：若后面将其赋值，那么正常索引大于0，若其小于零则说明未进行任何操作

* 相邻两帧图像检查依据，

解决：两帧图像所有特征点角度变化相似，具体看代码

* SVD分解？F、H分解为什么有几种可能的解？初始化时重建原理？

解决

* 跟踪时位姿得计算过程，直接使用g2o计算？

解决：2种方法，固定motion+g2o；参考帧3D-2D+g2o

* VSLAM基础（六）————超定线性方程组的解法，搜索这个系列看看

解决：potxxx的博客\_CSDN博客-c++,slam,计算机视觉领域博主 <https://blog.csdn.net/potxxx/article/list/2>

* Household变换

解决：数值分析求特征值章，为QR分解作准备

<https://blog.csdn.net/flyfish5/article/details/52315062>

第二遍待搞清问题，不留疑惑的全看懂，如果是自己该怎么写

* 各线程之间的协调，怎么通过操作内存相互协调
* 局部地图、闭环检测、重定位等理论及具体实现方式

解决：看论文

* 理清各类的作用，搞清public、protect等的作用

解决：protect可以被子类继承而private不能

* 如何评价结果？将结果以可视化的方式表现出来
* 注意观察照片缩放前后的区别，缩放使用什么方法比较好
* 金字塔各层比较和存储的方式，只存储匹配的一帧吗？注意factor的使用

解决：代码中按1.2比例缩小图像，对于同一个物体，相当于距离变远。金字塔的缩放相当于摄像头沿光轴平移，之后改变平移量就好。代码的思路是：通过双线性内插缩放图片，这里图片的间隔是一样的，分辨率不一样。然后使用相同的方法（如相同的patch），提取角点和描述子，最后将不同层（缩小）的特征点的坐标放大（乘对应层数即可）。特征点中还保存着对应层数。还需结合计算过程综合考虑

* 跟踪没用过的特征继续保存在关键帧中吗？

解决：对

* 估计初值怎么得来

解决：第一次计算之后都有初值了

* 优化时用的是特征点世界坐标系坐标到相机坐标系之间的位姿变换，故各帧之间不会相互关联？

解决：优化有很多种，有pose-point、pose-pose

* 相机参数校正过程

解决：提取完ORB特征后在使用校正参数校正特征点的坐标，包括四个顶点的坐标

* KD树词的格式，观察各特征之间1,0差别，一位的偏差是否会带来较大影响
* 地图点的描述子取哪个特征帧的描述子

解决：离其他描述子距离最近的描述子

* 参考帧怎么选取？从covisibility graph（只包括关键帧）吗？还是直接用上一帧作为参考帧，丢失之后才从地图上查找？
* 上次插入关键帧与当前决定插入关键帧之间帧怎么处理？论文中track是指匹配点还是map point？包括恢复的map point吗？
* 建立局部地图，怎么计算期望看到某个点的帧数？有个判断条件用到
* 局部地图停止后是否插入关键帧到local队列？
* dmin、dmax怎么算

解决：在mappoint中有一个参考帧专门供其计算，会随着运动不断更新，dmax=1.2\*该帧光心到该路标点距离\*观测所在层尺度（1.2的n次方），dmin=0.8\*dmax/最底层尺度（1.2的7次方）

* 视差的判断
* 在关键帧中出现但词袋中没有的词怎么处理？不考虑他们会带来什么影响？考虑会不会改善效果？
* 最近邻ratio？
* 通过旁边三个点恢复一些3D点的方法，3D融合过程中先匹配在映射吗？（两帧图都是用loop帧的路标点）？
* 尺寸的初始化？

解决：先使t计算位姿时使t的模为1，新建地图时将相机中间深度的距离设为1

* 倒数深度的作用？
* Local队列是在闭环检测的时候用的吗？注意各段时间
* 为什么会有四个解？
* 提取fast角点圆是怎样的？，，
* 只运行tracking线程跑一个循环会不会是运行三个进程时间的3倍？
* 从E分解出R得大于0
* 对于每张图片的特征是离谁最近挂在哪个页节点下吗？id一样？
* EPnP求解控制点坐标：待看代码具体计算过程，22点有参考网站。ICP问题？
* 正交相机模型和投影相机模型？
* 程序中4个和个beta的求法
* EPNP使用高斯牛顿方法优化beta过程，知道2个坐标系下多个3D点，求解Rt的过程，为什么可以用SVD分解

解决：<https://www.cnblogs.com/wall-e2/p/8057448.html>（重要，找出整个系列看看）

qr分解为作者自己写的，有空看

<https://blog.csdn.net/EyeToTheWorld/article/details/100113266>

* 求解时正负的选取问题
* 把slam2中求解方法总结一下，可尝试推理，以后面试用
* 非关键帧是怎么处理的，非关键帧数量和关键帧数量数字

1. 安装及设置git

安装：sudo apt-get install git -y

设置：<https://www.cnblogs.com/superGG1990/p/6844952.html>

（生成密钥对那就是邮箱，未设置前没有权限下载github上code）

警告：搜索问题，在某个文件后面加上ip即可

1. 下载安装工具及第三方库

参照源码网站：<https://github.com/raulmur/ORB_SLAM2>

包的两种常用方法：

* 直接使用sudo apt-get install 包名 安装
* 到网站下载包含cmakelist.txt文件的压缩包，解压后按cmake的方法安装：在解压目录mkdir build；cd build；cmake ..（根据cmakelist里分析里面文件、相关链接库，生成自动编译脚本Makefile等中间文件）；make –jn //n线程（编译）；sudo make install（安装）

卸载安装包方法：

<https://www.cnblogs.com/blackay03/p/9062503.html>

1. 安装cmake：编译用

sudo apt-get install cmake

1. 安装kdevelop

sudo apt-get install kdevelop

1. 安装vim

sudo apt-get install vim

1. 下载源码

注意按网站上clone语句：git clone https://github.com/raulmur/ORB\_SLAM2.git ORB\_SLAM2，没有后面ORB\_SLAM2会下不了，这与一般的下载不一样

对于DBow2和g2o已经包装在thirdparty文件夹中了，下载好程序后按照github网站上三步运行安装即可：cd ORB\_SLAM2 chmod +x build.sh ./build.sh

安装完上面几项后，用kdevelop打开文件，根据报错信息安装相应包，可参考如下网站

<https://blog.csdn.net/radiantjeral/article/details/82193370>

问题：pangolin使用catkin\_make\_isolated替代catkin\_make

1. 安装opencv（opencv 3.2.0，自己安装，记得先安装依赖项）

<https://blog.csdn.net/ksws0292756/article/details/79511170>

后面发现其与官网步骤类似，最好参考官网步骤：

https://docs.opencv.org/master/d7/d9f/tutorial\_linux\_install.html

注意先解压，解压步骤见最上方网站，然后按网站步骤弄

不先安装依赖项带来的麻烦，及解决步骤

* 报未安装lihgtk2.0-dev 和pkg-config依赖项错

解决：先安装两个依赖项包：sudo apt-get install 依赖项包，使用下网站第一条解决，第一句只是查看当前两个包版本。make比较慢，记得使用多线程。后安装相当于要把依赖项和cmake连接起来，会麻烦一点

<https://stackoverflow.com/questions/28776053/opencv-gtk2-x-error>

* cmake句时报未安装qt4错

解决：sudo apt-get install qt-sdk，然后cmake成功，继续按照上面网站操作就好

1. 安装eigen3（按上面网站），安装包无uninstall文件，卸载链接如下

<https://blog.csdn.net/j_____j/article/details/80622570>

1. 安装pangolin（按上面网站）

其中src文件夹相当于自己package放文件包的文件夹，不过这里得用src才能执行下面命令，故重新在package目录新建src文件，将pangolin下载此文件夹下。然后按照提示安装catkin，运行catkin\_make出问题后运行提示目录isolate。然后继续走在src上一级会多出build build\_isolated devel devel\_isolated多个目录，下次安装时注意多加一级，这样太乱

1. 1 sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ $DISTRIB\_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
2. 源码理解

* Tracking：重定位gauss\_netown优化及QR分解待看

<https://www.cnblogs.com/wall-e2/p/8057448.html>

* Local mapping

<https://www.cnblogs.com/wall-e2/p/8678839.html>

* Loop closure

<https://www.cnblogs.com/wall-e2/p/8686859.html>

1. 自己总结
2. tracking：线程只优化当前帧pose，先跟踪帧与帧之间的3D点（包括重定位），完成跟踪后再跟踪局部地图上的3D点，pose优化时会改变当前帧连接到3D点的outlier标志位，不影响3D点好坏。决定是否插入关键帧给localmapping
3. 只要相机中心有位移（baseline>0）就能进行三角化，若是前进后退，轴线上点不能三角化
4. 较远的特征对位姿估计更好
5. localmapping：三角化生成3D点，判断3D点、KF的好坏及是否删除，重复3D点融合，改变她们的isbad（）对应标志位，
6. 各线程切换重要标志位和函数（操作标志位）：
7. tracking：；
8. localmapping：

* mbacceptkeyframes：
* mbstoprequested：loop要求local暂停，准备correct\_loop()
* mbstopped：localmapping是否暂停的标志
* mbnotstop:track插入
* mbabortBA:是否放弃local中BA的标志。mbabortBA=true可以快点结束当前BA，如Track插入新帧时、track想要插入帧欲打断localBA、loop要求暂停local时
* setnotstop（）：track创建新帧时用
* stop()：local判断是否停止
* release():loop闭环校正完用

1. loopclosing：correcloop时间应该还是比较短的？GBA时间比较长？

* mbrunningGBA：loop开的全局优化线程是否运行标志位
* mbstopGBA：是否暂停正在进行的GBA的标志位

1. GBA：loopclosing更新完essential graph后进行GBA，先释放localmapping，再单独更新，最后暂停localmapping进行融合。
2. 多线程互锁：对于一个几个线程都会调用的类实例的变量使用函数进行打包，使用时调用该函数，重要的是如何把经常一起出现的变量进行分类。
3. 先在畸变的原始图中提取特征，再去畸变（改变特征点xy值）
4. 在跟踪特征时，motion模式时寻找3D点在金字塔的与lastframe相邻3帧之间寻找，如果寻找太多层反而容易出现误匹配。
5. 用mnid赋给某个变量防止重复访问，如keyframedatabase.ccline222
6. 重建路标点时，像素点应该离极点较远才好
7. 第2幅图上的极线：l2=x1TF12，F12=K1-Tt12^R12K2-1。x1TF12x2=0。极线[a,b,c]具有尺度自由，需用a\*a+b\*b来限定尺度。点在直线l上，xTl=0。缩小越大，可能误差越大。
8. 注意重建路标点时，检查尺度是否一致方法。用距离和缩小因子比例来判断。Localmapping的line431
9. Init：2D-2D
10. 第二帧特征点大于100
11. 第二帧与第一帧匹配点数大于100
12. 根据阈值RH(可取0.4-0.46)选择合适的方法重建，3D点只要求50个

* 8点法得到F矩阵，F矩阵分解得到R、t（可随意给定），三角化得到3D坐标。2处判断，需要足够点和足够明显优于其它3种方案的位姿，第二处视差大于1°。
* 8点法得到H矩阵，H矩阵分解得到R、t（可随意给定），三角化得到3D坐标。2处判断，5个条件。

1. 2帧经过全局优化后，若3D点少于100或深度中值为负，初始化失败
2. 三角化：基线长度已知，求某3D点在两帧中的深度，若基线/深度的值越小（取0.01），那么基线的误差对深度值影响比较大。
3. 小车上的摄像头不装在转轴上不会出现纯旋转，那装在什么位置会比较好呢，纯旋转相当于基线值为0，那么基线/深度=0，不能进行三角化，那么要装个好位置
4. Frame和KF的mnid贯穿整个代码
5. 在求sim3时，使用RANSAC每次迭代5次寻找大于20个点的相似变换（可能小于5次），当大于累计大于最大迭代次数时nomore=true。
6. 特征点匹配方法（数据关联）：具体细节看源码，每层最大特征点数：设置数（初始化2000，平常1000）\*0.217（0.181、0.151、0.126、0.105、0.087、0.073），相机内参都已知
7. searchforinitalization（1F,2F,第一帧特征点，匹配点（F1.size），windowsize=10：初始化时用，windowsize=100，nnratio=0.9，checkori=true。为了使精度更好只要2帧金字塔第一层特征点（434-434）。
8. 目的：寻找1F、2F之间匹配点
9. 已知条件：没有任何先验信息
10. 搜索潜在匹配点：以第一帧特征点为外循环，在第二帧中以第一帧特征点坐标windowsize\*2正方形像素窗口找特征点。第二次匹配上，取最好的。
11. 描述子（2D-2D）距离判断：满足<TH\_LOW(50),best<0.9\*best2。
12. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。

重建时

FindF、H

1. 使用8点法的RANSAC利用F（x1Fx2=0、x2Fx1=0）和H（p1-Hp2=0，p2-Hp1=0）约束去掉一些异常点

CheckRT()

1. 利用匹配点三角化后3d点在相机前方（z>0）去掉一些异常点，注意平行（cos>=0.00002）时，可以不大于0，即该点为无穷远点
2. 三角化后3d点投影到两帧图的重投影误差都小于4个像素

重建成功后一般还剩100个点左右，要求>50个。

1. Searchbyprojection（cF,lF，边长th，bool mono）：motion跟踪时currentF-lastF，现在正方形边长th为15找，match小于20，th改为30，pose优化后还需保留10个点以上才算成功，checkori=true；
2. 目的：在cF的3D点中寻找与cF特征点的匹配点
3. 已知条件：lF对应的路标点和相机外参都已知，cF的相机外参已知（motion-假设为lF\*v；）
4. 搜索潜在匹配点：将lF的所有3D点转换到cF坐标系，再投影到cF像素坐标系
5. 非null和非outlier的3D点
6. 深度z为正
7. 投影后像素坐标不超过图片校正后的边界
8. 根据投影后像素坐标搜索（双目有所不同）：r=th\*lF该点所在层缩小倍数（不是原图层特征可能没那么准），先取mgrid（64\*48）正方形内对应特征，然后取3D点所在lF层和前后层，共三层的点。第二次匹配上直接跳过。
9. 描述子（3D-2D）距离判断：满足<TH\_HIGH(100)。
10. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。
11. Searchbybow（rKF，cF，cF对应3d点）：trackreferenceKF时，特征匹配>=15,pose优化后>=10，nnratio=0.7，checkori=true；重定位，从多个候选帧（太多影响速度）找与cF匹配点>=15，nnratio=0.75，checkori=true；
12. 目的：在rKF的2d特征点暴力与cF特征点匹配
13. 已知条件：rKF对应的路标点（匹配时不用，pose优化才用）和相机外参都已知，rKF和cF（使用前算得）的mfeatvec（node（第4层），特征容器）已知
14. 搜索潜在匹配点：2帧相同node特征进行暴力匹配，外层为rKF的特征点，选取非null和非isbad（localmapping会更改）3D点，内层cF相同node所有点，内层若被第二次匹配直接跳过。
15. 描述子（2D-2D）距离判断：满足<TH\_LOW(50),best< nnratio \*best2。
16. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。
17. Searchbyprojection(cF,KF,已经匹配到3D点，边长th，ORBdist)：重定位，根据判断条件，checkori=true，可能会使用2次，1-th=10，ORBdist=100，2（位姿又一次优化）-th=3，ORBdist=64；
18. 目的：在KF的3D点中寻找除cF已经匹配到的3D点以外的点，即新增点
19. 已知条件：KF对应的路标点和相机外参都已知，cF已经匹配的3D点，cF的pose已知，且已经过优化（reloc），较准确
20. 搜索潜在匹配点：将KF的所有3D点转换到cF坐标系，再投影到cF像素坐标系
21. 非null和非isbad、非已经匹配
22. 深度z为正（未加）
23. 投影后像素坐标不超过图片校正后的边界
24. 3D点与cF光心距离未超出3D点最大、小距离
25. 根据投影后像素坐标搜索（双目有所不同）：r=th\*该点在cF中预测到所在层缩小倍数（不是原图层特征可能没那么准），先取mgrid（64\*48）正方形内对应特征，然后取3D点所在cF预测层和前后层，共三层的点。第二次匹配上直接跳过。
26. 描述子（3D-2D）距离判断：满足<= ORBdist。
27. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。
28. Searchbyprojection（cF，localmappoints，th）：tracklocalmap时，正常th=1，刚重定位完（<2）th=5,viewingcoslimit=0.5, nnratio=0.8；
29. 目的：跟踪localmappoints中在frustm内的点
30. 已知条件：经过isinfrustm（viewingcoslimit）判断的localmappoints，cF已经跟踪到的3D点(trackinview=false)不再重复投影，cF的pose已知
31. Isinfrustm（viewingcoslimit）：判断某3D点是否在当前帧的射影锥内，满足trackinview=true，否则trackinview=false
32. 将3D点世界坐标转换到当前帧坐标系，深度z>0
33. 重投影不超过图片校正后的边界
34. 3D点与cF光心距离未超出3D点最大、小距离
35. 射线与3D点方向夹角的余弦viewingcos< viewingcoslimit
36. 从localmappoints搜索潜在匹配点：外循环3D点，
37. trackinview=true
38. 非isbad（）
39. 根据viewingcos选择r（通常是4，大于0.998取2.5），在重投影坐标周围r\*th\*predictscale的缩放倍数，cF第predictscale-1和predictscale两层找潜在匹配点。第二次且观察到帧大于0的cF特征点直接跳过。
40. 描述子（3D-2D）距离判断：满足<TH\_HIGH(100)，best< nnratio \*best2||bestlevel!=best2level
41. Searchfortriangulation（pKF1,pKF2,matched<idx1,idx2>，onlystereo）:localmaping三角化新增3D点时，首先判断当前帧（pKF1）和共视帧pKF2baseline（相机中心的距离）是否足够（baseline/共视帧深度中值>0.01），最后匹配点是否能三角化还得两帧看同一个点射线夹角余弦值大于0小于0.9998，即大于1.15°小于90°。
42. 目的：寻找pKF1、pKF2除以及匹配好的3D点外特征点的匹配，并将匹配结果存在matched<idx1,idx2>
43. 已知条件：pKF1,pKF2已经匹配好的3D点和比较准确的位姿（那么可以求出F12）
44. 搜索潜在匹配点：外层pKF1的相同节点的除3D点外所有特征点，内层pKF2的相同节点的除3D点外所有特征点，第二次匹配直接跳过
45. 描述子（2D-2D）距离判断：满足<TH\_LOW(50)，dist<bestdist
46. 得到pKF2的特征点与极点（pKF1的光心在pKF2像素平面的投影）差的二范数平方>100\*该特征点所在层的缩放因子
47. 极线约束（理论上x1TF12x2=0）：x1TF12x2/（极线l=x1TF12前两个系数的平方和，抵消尺度）<3.84\* pKF2特征点所在层的缩放因子平方
48. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。
49. fuse(pKF，vpmappoints，th=3)：localmapping3D点融合时，先假设pKF所有特征点没有3D点，th=3，匹配后再融合，已有3D点的用观察多的替换观察少的，没有的关联起来即可。
50. 目的：先假设pKF所有特征点没有3D点，匹配后再融合，已有3D点的用观察多的替换观察少的，没有的关联起来即可。
51. 已知条件：3D点世界坐标，pKF的pose以及已经匹配的3D点
52. 搜索潜在匹配点：将所有3D点转换到pKF坐标系，再投影到pKF像素坐标系
53. 非null、非isbad和pKF尚未观察到该点的3D点
54. 深度z为正
55. 投影后像素坐标不超过图片校正后的边界
56. 3D点与cF光心距离未超出3D点最大、小距离
57. 3D点与cF光心射线与3D点normal方向的余弦值>0.5(即<60°)
58. 根据投影后像素坐标搜索：r=th\*3D点预测层缩放倍数周围，3D点预测层及其前一层（-1）2层找潜在匹配点
59. 3D点重投影误差小于5.99\*特征点所在层数缩放倍数平方
60. 描述子（3D-2D）距离判断：满足<TH\_LOW(50)
61. 融合：已有3D点的用观察多的替换观察少的，没有的关联起来即可。
62. Searchbybow（pKF1，pKF2，pKF1对应索引数内容为pKF2的3D点）：闭环检测查找当前帧pKF1和闭环候选帧pKF2匹配3D点时，nnratio=0.75，checkori=true，匹配点需要>=20；
63. 目的：利用featvec暴力匹配pKF1，pKF2的3D点，使用是2D点描述子
64. 已知条件：pKF1，pKF2中的3D点，但pKF2的尺度、pose和3D坐标有漂移。
65. 搜索潜在匹配点：2帧相同node特征进行暴力匹配，外层为pKF1的特征点，选取非null和非isbad（localmapping会更改）3D点，内层pKF2相同node所有点，选取非null和非isbad（localmapping会更改）3D点，内层若被第二次匹配直接跳过。
66. 描述子（2D-2D）距离判断：满足<TH\_LOW(50),best< nnratio \*best2。
67. 整体方向判断：最后验证特征点方向（帧与帧之间可以验证方向）。
68. Searchbysim3（pKF1，pKF2，pKF1对应索引数内容为pKF2的3D点，s12，R12，t12，th）：闭环检测时求解出相似变换后，寻找当前帧pKF1和闭环候选帧pKF2更多的匹配3D点
69. 目的：利用pKF1投影到pKF2，寻找pKF1、pKF2已经匹配到3D点之外的点
70. 已知条件：pKF1，pKF2中的3D点和已经匹配的3D点，以及它们之间的相似变换
71. 搜索潜在匹配点：外循环pKF1非null、未与pKF2匹配和非isbad的3D点，内循环pKF2非null、未与pKF1匹配和非isbad的3D点
72. 将pKF1的3D点通过相似变换转换的pKF2坐标系
73. 深度z为正
74. 投影后像素坐标不超过图片校正后的边界
75. 3D点与cF光心距离未超出3D点最大、小距离
76. 根据投影后像素坐标搜索：r=th\*3D点预测层缩放倍数周围，3D点预测层及其前一层（-1）2层找潜在匹配点
77. 描述子（3D（pKF1）-2D（pKF2））距离判断：满足<=TH\_HIGH(100)
78. 搜索潜在匹配点：外循环pKF2非null、未与pKF2匹配和非isbad的3D点，内循环pKF1非null、未与pKF1匹配和非isbad的3D点
79. 将pKF2的3D点通过相似变换转换的pKF1坐标系
80. 深度z为正
81. 投影后像素坐标不超过图片校正后的边界
82. 3D点与cF光心距离未超出3D点最大、小距离
83. 根据投影后像素坐标搜索：r=th\*3D点预测层缩放倍数周围，3D点预测层及其前一层（-1）2层找潜在匹配点
84. 描述子（3D（pKF2）-2D（pKF1））距离判断：满足<=TH\_HIGH(100)
85. 检查两次匹配结果是否一样，选择2次匹配一致的3D点，一致才加入第三个参数
86. Searchbyprojection（pKF，scw，vppoints，vpmatched（索引与pKF对应），th）：闭环检测时，用于由闭环帧及其共视帧看到的vppoints建立与pKF除已匹配到的3D点vpmatched以外特征点的匹配
87. 目的：从vppoints建立与pKF除已匹配到的3D点vpmatched以外特征点的匹配
88. 已知条件：3D点到pKF坐标系的相似变换
89. 搜索潜在匹配点：外循环vppoints未与pKF2匹配和非isbad的3D点，内循环pKF未匹配到3D点的特征点
90. 将vppoints的3D点通过相似变换转换的pKF坐标系
91. 深度z为正
92. 投影后像素坐标不超过图片校正后的边界
93. 3D点与pKF光心距离未超出3D点最大、小距离
94. 3D点与pKF光心射线与3D点normal方向的余弦值>0.5(即<60°)
95. 根据投影后像素坐标搜索：r=th\*3D点预测层缩放倍数周围，3D点预测层及其前一层（-1）2层找潜在匹配点
96. 描述子（3D-2D）距离判断：满足<=TH\_LOW(50)
97. fuse（pKF，scw，vppoints，th,vpreplace（索引与vppoints对应,内容为pKF对应的3D点）：闭环检测时，用于由闭环帧及其共视帧看到的vppoints建立与当前帧及其共视帧pKF除已替换（已换成了vppoints）的3D点以外特征点的匹配，th=4
98. 目的：从vppoints建立与pKF除已替换（已换成了vppoints）的3D点以外特征点的匹配
99. 已知条件：3D点到pKF坐标系的相似变换
100. 搜索潜在匹配点：外循环vppoints未与pKF匹配和非isbad的3D点，内循环pKF未匹配到3D点的特征点
101. 将vppoints的3D点通过相似变换转换的pKF坐标系
102. 深度z为正
103. 投影后像素坐标不超过图片校正后的边界
104. 3D点与pKF光心距离未超出3D点最大、小距离
105. 3D点与pKF光心射线与3D点normal方向的余弦值>0.5(即<60°)
106. 根据投影后像素坐标搜索：r=th\*3D点预测层缩放倍数周围，3D点预测层及其前一层（-1）2层找潜在匹配点
107. 描述子（3D-2D）距离判断：满足<=TH\_LOW(50)
108. 若匹配点已有3D点，将结果存在vpreplace，待之后替换，没有3D点则直接建立与pKF特征点关联。
109. 丢失时，重定位必须进行，闭环不一定会进行。
110. Bow的用处：重定位和闭环检测时，通过当前帧包含的词，使用bow的invertfeat由词找帧，另外通过帧向量计算帧的相似度；帧与帧之间匹配时，使用（node（bow某一层）-index（特征在帧的索引））暴力匹配相同node下的特征。
111. Tracking：都是3D-2D

跟踪

1. Trackmotion（Frame-Frame）：根据匀速速度作为帧pose初值，再使用poseoptimization优化
2. TrackreferenceKF（KF-Frame）：将KF的pose作为初值，再使用poseoptimization优化

若Frame通过上面两种方法都丢失，在下一帧Frame来时，进行重定位

1. Relocalization：挑选候选帧后，使用EPnP(通过控制点求出当前帧3D相机坐标，再通过3D-3Dset求出相机pose)方法得到当前帧pose作为初值，再使用poseoptimization优化
2. Localmapping
3. Loopclosing
4. 重复点判定方法：
5. 线性方程求解及SVD求解
6. 使用SVD计算H、F线性方程，还有H、F分解，F还需利用SVD分解将奇异值变为[r,s,0]形式，能否通过最小奇异值判断好坏？
7. 三角化2点使用SVD求解3D点(初始化和localmapping新增3D点)
8. 重定位3D-2D（EPnP）
9. 计算控制点（PCA）
10. 计算M对应奇异向量v
11. 从3D-3D点计算R、t
12. 自己改变过或可能存在bug的地方
13. 重定位（keyframedatabase.ccline288）：分数不是和当前帧的分数，不改也能用，在264更改
14. 初始化（initializer.cc-line52）：mvmatches12.reserve（）预留内存出错，已改
15. ORBmatcher.cc-line786：vbmatched[id2]未赋值。
16. Optimizer.cc-line836:忘设setfixed了？默认为false吗？第0帧应该设为true
17. 别人总结

* orb前端中的特征选择策略大致是对存在多个特征的grid四等分（且优先等分特征数目较多的），当等分出的grid达到需要的特征数目时就停止等分，然后从每个grid中挑出最强特征作为特征选择的输出。这样做的好处是（相对于opencv中最原始的grid策略，即等分全图，每个grid取一个点），强纹理区域会提取更多特征，弱纹理相对较小，不过局部区域都相对比较均匀，更加合理。maplab中也提出了一种类似的策略，大致是先用原始grid策略找出一定数量强特征，如果特征数目不够，再用原始grid策略找出一些相对较弱的特征，也在一定程度上弥补了原始grid策略不能自适应的缺陷。另外自动驾驶的前视相机根据场景特性会发现如果将图片从左往右等分成若干列，保证每列特征数目尽量相等也能获取不错的特征分布。
* <https://mp.weixin.qq.com/s/t3T7g5sFvFXhgVNn0vlNtA>
* <http://www.shenlanxueyuan.com/open/course/27>
* ORB特征均匀提取策略对性能的提升有多大：<https://mp.weixin.qq.com/s/MgJizauaLqigkV-UlOg9uQ>
* GPU版：<https://github.com/yunchih/ORB-SLAM2-GPU2016-final>
* PNP和ICP ：<http://mp.weixin.qq.com/s?__biz=MzA4NTEwNTI0Nw==&mid=2247483918&idx=1&sn=8c04e7d360f9ac7c9c9c8f34c657526e&chksm=9fdc4d3ea8abc428da8229c3ec24fc8d91a998fd8b2d27633217bbfe4beeebe02673814998a6&mpshare=1&scene=23&srcid=&sharer_sharetime=1573794045740&sharer_shareid=49a11175da4d4edb605c7f35b37db012#rd>

1. 各类的作用及重要类函数和变量，首先构建一个system类实例，然后在这个类里面包含所有类的实例指针，有些类可能又在实例指针中再建立实例指针。相机坐标系的中心在光心。自己正面看图片时。代码中横向为x轴-u-cols-width，纵向为y轴-v-rows-height，圆心在左上角。故在用at定位像素是应是<y,x>。new的地方创建一个指针实例。

注意有些类不建立整个周期的实例，用一次就释放了，更多关于类的细节见类定义

* Mono\_tum：主函数，在TUM数据集进行单目运算
* System：全局普通变量。整个系统相关操作。
* Initializer：初始化时候用，计算H、F矩阵，并分解得出位姿。在tracking中新建该类的指针实例。
* ORBextractor（是tracking内局部指针类）、Extractornode（不新建指针，只作类型用）：提取特征时用，两个在ORBextractor.h头文件中定义。
* 提取角点：首先，使用30\*30的网格提取金字塔（金字塔保存的是用101填充后的图像）每层图像角点，尽量保证在每个网格提取到角点，若FAST阈值从20变为7还提取不到就没办法了；然后使用四叉树不断分裂的方法使特征点均匀分布（可以防止扎堆的角点），分裂到直到方框数大于要求的特征数或者不能再分裂位置，选取每个方框响应强度最大的角点。（如果提取角点小于要求的角点，分裂也没意义了，代码是否可以优化？），最后保存的角点的x、y坐标是按填充19个像素后算的，再对金字塔每层坐标进行放大。（这样会不会有问题？）
* Frame：局部普通类。帧，新来的帧，需要在它上面提取特征。包括创建序号mnid和timestamp两个id。对当前帧进行处理，提取特征，供track使用。操作函数在ORBextractor，将提取特征的关键点存于其mvkeys（keypoint）-，描述子存于mDescriptors（mat：cols-特征点个数，rows-描述子32）。用于创建当前帧mcurrentframe。提取特征点后校正特征点x、y坐标，将其存于mvkeysUn（keypoint）。将特征点与mvpmappoints对应起来。

其包含复制构造函数，在初始化时会用于复制类实例。

getfeatureinarea（）函数有层数限制。

* Keyframe：一帧关键帧创建一个指针实例，新来的帧满足条件可变为关键帧。初始化时创建初始化地图就有用了，包括创建序号mframeid、mnid和timestamp（与frame相同）3个id。getfeatureinarea（）函数没有层数限制。重要变量：
* 特征点
* 描述子（矩阵形式，行号对应特征点索引）
* Mappoint:大小和索引与特征点相同，没有3D坐标特征点为NULL
* mconectedKeyFrameWeights：所有与其相连的KF和连接权重（边数）
* mvporderedconectedkeyframes：连接数（权重）大于等于15的关键帧
* mvorderedweights：与mvporderedconectedkeyframes对应的权重
* mfeatvec<词袋第4层的节点，该节点包含的当前帧的特征的索引>:用于两帧的暴力匹配
* mbowvec<bow词，权重>：图片向量,计算图片相似度用
* Keyframedataset（是system内指针类）：保存所有关键帧，主要用于重定位和闭环检测回环查找候选帧。重要变量：
* Mvinvertedfile<word,<image>>:由当前帧的特征（单词）通过DBOW2公共单词（特征）找出现过这些特征的关键帧
* Mappoint：一个3D路标点创建一个指针实例。重要变量
* 路标点最大、最小距离，以参考帧（通过三角化新建该路标点的当前KF帧，若该帧被删则取观察到它的下一帧）作为参考，最大距离=光心与路标点距离\*其所在层数尺度因子，最小距离=最大距离/最后一层尺度因子。金字塔缩小（描述子实际面积变大），尺度因子变大。对于同一个角点来说，若参考帧是第一层，距离再远一些，实际图像缩小，描述子范围变小，没有层描述子与它匹配，但当距离近一些时，实际图像放大，描述子范围变大，缩小的层可能与它匹配。描述子取与其他描述子中位距离最小的（为什么不用参考帧的呢？重定位或闭环检测要用？）。
* 描述子：取所有与路标点相连关键帧对应特征点描述子与其它帧具有最小距离中值的帧的特征点描述子。
* normal方向为各方向和的平均值。
* 路标点在某帧层数预测predictscale：scale=log（最大距离/路标点与光心距离）/logscalefactor，推导：最大距离是金字塔第0层的观测，缩放倍数=1，由几何知识知：，两边取对数，可求得scale。的值和深度z的比值相同。结果向上舍入，那么真实可能出现在预测层及其前一层。
* Observation：map<KF,i>-保存观察到该点的关键帧和特征索引。
* Map（是system内指针类）：z关键帧（mspkeyframes）和3D路标点(mspmappoints)，使用指针，便于节省空间。
* ORBmatcher：临时用，特征匹配。在跟踪时新建一个局部类
* Tracking（是system内指针类）：跟踪用，主要在主线程跟踪线程用。包含currentframe

和lastframe用于跟踪。

* Localmapping（是system内指针类）：局部地图构建线程
* Loopclosing（是system内指针类）：闭环检测线程
* Converter：g2o、Eigen、cv之间的矩阵转换
* Optimizer：优化器，BA、pose、sim3、essential graph等优化方法，不见实例，像函数一样调用
* Pnpsolver：3D-2D求解
* Sim3solver：求相似变换

画图用

* Framedrawer：画出窗口
* Mapdrawer：在窗口中画出关键帧和路标点
* Viewer：画图线程主函数
* Plane
* viewerVR

第三方库

* DBoW2：词袋（ORBVocabulary），具体可查看DBoW2源码库，ORBSLAM2对其进行了一些修改

ORBvoc.txt格式：父结点id 是否是叶节点（>0表示是） 32为无符号8位整数（描述子32\*8=256） 描述子权值（非叶结点权值为0）

<https://github.com/dorian3d/DBoW2>

* Dutils：产生随机数
* g2o：图优化，具体可查看g2o源码git库，ORBSLAM2对其进行了一些修改

<https://github.com/RainerKuemmerle/g2o>

1. 重要阈值或参数

* 初始化时：
* 提取比跟踪两倍的特征2000
* 初始化成功要求：当前帧特征点大于100；匹配点大于100；
* 特征匹配：只在金字塔第一层做匹配；在参考帧像素上下左右100个像素点找合适匹配点；汉明距离小于50且最好距离小于次好距离0.9倍；同一个点不能被两次匹配；方向检测词袋数量为30，选择其中三个数量最多词袋，两个次好词袋个数应大于最好词袋的0.1倍，否则丢弃。
* 开辟两个线程计算H和F：从匹配点中随机选出200组（迭代次数）8个点对，特征点可以重复使用，保证每组不重复即可。计算分数阈值看代码和吴博ppt-p24，
* 使用F恢复三维坐标

checkRT：两个在cosparallax<0.99998(即保证方向不相同)时判断深度是否为负；重投影误差（差的平方和）小于4；保留的点cosparallax<0.99998(即保证不平行)。满足前两个条件就认为是好的点ngood++；满足三个条件才将对应标志置true

选出最好的一组：最好的RT中正确的点要多于局内点的0.9且总数在50以上，其他非正确RT要少于最好的RT点数的0.7倍。

另外parallax（两条射线的角度）要大于1度，当满足要求点多于50个时，要求前50个大于1度，否则要求全部大于1度

* 使用H恢复三维坐标：checkRT和F一样

选明显优于其他组最好的一组（条件为且关系）：最优解点数的0.75倍比次优多；大于局内点的0.9倍且大于50个；视差大于1度

* covisibility graph：共视个数-；
* 场景识别：最好分数的百分数-；
* DBOW2：类数K=10（0-20）；深度（不包括根节点）L-6（1-10）；scoringtype（两个特征间6种相似性打分方法）-L1\_NORM(L2\_NORM、CHI\_SQUARE、KL、BHATTACHARYYA、DOT\_PRODUCT)；weightingtype（特征权重4种计算方法）-TF\_IDF(TF、IDF、BINARY)
* 初始化：匹配的特征值最小值-；
* Tracking：金字塔层数-；尺寸因子；欲提取的特征数-；检测FAST的阈值；是否保留某关键点（角点）的规则；局部地图优化K1与当前帧共视帧的个数阈值；及K2帧数阈值；；自适应detector阈值；自适应cell阈值；判断跟踪丢失条件；pnp中inlier数；匹配时方向一致性检测；最近邻ratio-；
* Trackwithmotionmodel（）：位姿初值为mverlocity（之前两帧的运动速度）\*mlastframe，使用lastframe的重投影（重载函数）来找
* Trackreferencekeyframe（）：位姿初值lastframe

跟踪到路标点超过15个。

* 两个都使用pose优化计算pose，同时探测一些异常的路标点，结果>=10返回true
* Tracklocalmap：正常需30个点，重定位阶段需50个点才算成功
* Map：利用旁边N帧进行重建地图；三角化极线约束-；视差；删除帧的百分数-；dmin、dmax；视角差别阈值-60；
* Loop：smin计算边权重；一致性检测；
* EDGE\_THRESHOLD=19：代码中采用101反射方便提取边缘上角点，FAST提取中的6，+-3是为了保证提取的都是图片内的点，而不提取填充上的点。19是为了计算描述子用的，斜框时15\*根号2=21。

1. 用到opencv类和函数

* FileStorage: 数据持久化, FileStorage类的数据存取操作与示例，常用于打开yaml参数文件

<https://blog.csdn.net/iracer/article/details/51339377>

* mat: 属性step，size，step1，elemSize，elemSize1。create-直接创建一个矩阵

<https://blog.csdn.net/Felaim/article/details/80702787>

矩阵操作的一些方法

<https://blog.csdn.net/b1342504010/article/details/17717137>

* outputArray：特征提取返回描述子引用时

<https://blog.csdn.net/yang_xian521/article/details/7755101>

* copyMakeBorder：(在程序中等于没用)扩充边缘（扩充能不能增加像素的利用率，不扩充最后一层相对原图来说边缘有1.2^7\*19=68的像素没用）：101-以最边缘像素为轴，关于BORDER\_ISOLATED用法在函数声明中有详细描述。注意rect(),sz的使用

<https://www.colabug.com/1691132.html>

* resize：对图像进行放缩，也称图像内插，书P37，内插过程理解：改变放缩后图片像素之间间隔，使其与原图像重合，注意两者间隔不同，放大，间隔变大，缩小间隔变小，在此基础上进行插值。常用双线性插值。

<https://blog.csdn.net/guyuealian/article/details/85097633>

<https://blog.csdn.net/qq_37577735/article/details/80041586>

* 给单通道赋值，注意不同通道的类型

<https://zhidao.baidu.com/question/1050998807787878299.html>

* FAST：threshold为中心点与圆上点边缘差值（本文最大20，最小7），其为FASTx从半径为3（原图边缘3个像素提取不到，需要填充）的16个像素选取9个的情况。得到kepoint，得到pt、response可直接用

<https://blog.csdn.net/zhaocj/article/details/40301561>

* Keypoint：pt(0,0)坐标, size(0)代码中PATCH\_SIZE\*缩小比例（>1），缩小n倍后提取特征，相当于在原图n\*patch上提取特征。angle(-1)方向以x轴到y轴为正方向, response(0)响应强度，和中心点与圆上像素差值有关, octave(0)所在金字塔层, class\_id(-1)

<https://blog.csdn.net/u010821666/article/details/52883580>

* fastAtan2：计算反正切函数，返回结果为0-360，误差0.3
* getMat 、 InputArray 、OutputArray、 CV\_Assert 、 create、\_tchar、wchar\_t、char、typename、类的静态数据成员、Mat

<https://blog.csdn.net/weixin_41484240/article/details/80595507>

* mat.reshape：第一个参数通道数，所有通道看成矩阵一个元素，第二个参数行数.肯定是2\*2

<https://www.cnblogs.com/denny402/p/5035535.html>

* undistortPoints（）：提取完特征点后进行校正，输入参数为2通道的

<https://blog.csdn.net/yong_qi2015/article/details/52946821>

<https://www.cnblogs.com/riddick/p/6711263.html>

* SVDcomp：返回值已经按从大到小排列

<https://blog.csdn.net/weixin_42587961/article/details/96865137>

<https://www.cnblogs.com/mikewolf2002/p/3454760.html>

* norm：求范数
* cvmultransposed：求AT\*A
* determinant：求行列式
* cvSVD: <http://www.mamicode.com/info-detail-57033.html>
* cvRealeaseMat: 该函数参数为指向cvmat的图像指针的引用，运行完该函数后使参数指针置空（NULL），该函数与cvReleaseImage(&pImage)的用法类似
* cvMat：将数组与cv矩阵绑定起来，这样方便操作矩阵一些
* cvsolve：解线性方程组
* reduce：将一个矩阵变为向量，如求3个3D坐标得中心坐标
* eigen：计算矩阵特征值和特征向量
* 【杂谈opencv】OpenCV中的cvRound()、cvFloor()、 cvCeil()函数讲解\_拜小白的成长之路，告别小白-CSDN博客

<https://blog.csdn.net/sinat_36264666/article/details/78849125>

* Cvtcolor：颜色空间转换
* Gaussianblur：高斯滤波
* Cvset：设置矩阵的某个元素为浮点数
* Rodrigues：轴角转旋转矩阵或旋转矩阵转轴角，自动识别。
* Pow：对个各元素求指数
* 1

1. std常用函数

* norm：求向量的2范数

<https://blog.csdn.net/gxiaob/article/details/8471158>

* 1

1. g2o：

* g2o矩阵使用的是eigen矩阵

1. eigen:代数计算，g2o要用

* quaternion：将矩阵转化成四元数

1. pangolin：可视化
2. TUM数据集：包括室内室外范围比较小的场景

* 数据集的文件名就是时间

1. timestamp记录图片的先后顺序，covisibility graph和essential graph 、span tree用于优化
2. 各线程通信标志：tracking用covisibility graph；local mapping会用也会改graph，且操作较频繁；loop会用也会改graph，且改得较少，检测到闭环的时候停止local

* Loop启停local标志位：

1. 文件加下有build.sh，执行了一系列命令行操作，建立好库函数和可执行程序，examples里有各中情形的可执行程序。可以通过cmakelists.txt文件查找生成对应库和可执行程序使用的源文件
2. yaml文件常用于写配置文件，简介如下

<https://www.cnblogs.com/wxmdevelop/p/7341292.html>

1. 使用kedevelop打开吴博注释代码及源码两个工程时，源码system.h的头文件system的构造函数与吴博代码链接上了。原因不详，中间有过挺多乱七八糟操作。如改文件名，build多次
2. 程序的线程之间通过标志位来控制

<https://blog.csdn.net/hzwwpgmwy/article/details/80493247>

1. C++ 互锁mutex的使用，待进一步理解，b站有收藏，lock是保护在其之下的函数及变量？
2. copyMakeBorder：<https://blog.csdn.net/qq_22764813/article/details/52787553>
3. 取整函数：<https://www.cnblogs.com/zjutlitao/p/3558218.html>
4. 立方体可以用八叉树切分，图片可以用四叉树切分
5. 多线程

* 线程名.join:等待直到线程结束

1. 提取fast角点时，在每个cell设定最小数量，从大到小（20-7）设置fast的阈值，直到获得合适数量角点，阈值越大越好
2. 类之间相互包含彼此的指针类，既可以增强彼此的联系，又可以节省空间，使用指针操作相当于在不同类型操作同一片内存，但是多线程操作时可能会出现冲突
3. 计算汉明距离在代码中有快速的计算方法，代码中有相应链接
4. EPnP求解控制点坐标：（实验所得）一个控制点取4个路标点质心坐标，然后去质心后，其余三个控制点取四个路标点PCA主方向上三个点，PCA能更好的表达这四个点的信息

<https://blog.csdn.net/jessecw79/article/details/82945918>

<https://www.jianshu.com/p/312cc524b8d1>

1. 1

