monocular visual-inertial system（VINS）总结

问题

1. 注意各个变量是在IMU还是camera帧率保持不变的，理论上是连续的，但实际需要按IMU或camera帧率离散化
2. Imu偏差与其测量值大小的关系，计算其百分比
3. 机体静止太久，优化会挂掉吗？由于长时间积分？可否根据方差值决定是否优化cameraIMU约束项？马氏距离实现了这一点，是否需要加Huber核？
4. 比较CV库和ORBSLAM2中求解F、pnp等方法的速度
5. 图片可否在树莓派上处理完，传有用的信息到笔记本，保证传输的实时性，尽力运用分布式处理
6. IMU预积分初值有多准呢？需要经过多少次迭代才比较好？这些可否用图表打印出来，更好的选择参数和算法
7. 2种角点如何保持旋转不变性？

解决：计算描述子时，考虑方向

1. 人是不知道机器人到底是在空间哪个位置初始化的
2. 边缘化可不可以不要
3. imu和视觉约束之间的权重大小会有影响吗？打印两项方差观察，猜想应该是有影响的，因为他们是放在一起优化的。
4. 1

参考论文：

1. 总框架：VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator（19周周报有详细描述）
2. EuRoc相机模型及标定方法（包含自己推导公式）：Single View Point Omnidirectional Camera Calibration from Planar Grids
3. 初始化：Robust initialization of monocular visual-inertial estimation on aerial robots,
4. VIO：Monocular visual-inertial state estimation for mobile augmented reality
5. 重定位、闭环检测和地图融合：Relocalization, global optimization and map merging for monocular visual-inertial SLAM
6. 前端预处理（特征提取和匹配）：特征- Good features to track

匹配（KLT稀疏光流）-An iterative image registration technique

with an application to stereo vision

Kanade-lucas-tomasi (klt) feature tracker（PPT）

1. IMU预积分：

* 主要-Tightly-coupled monocular visualinertial fusion for autonomous flight of rotorcraft MAVs
* 偏差估计（zero-order离散积分方法）- On-manifold preintegration for real-time visual–inertial odometry；

IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation

1. 视觉SfM：

* 2D-2D三维重建（5点法）- An efficient solution to the five-point relative pose problem
* 3D-2D- EPnP: An accurate O(n) solution to the PnP problem
* BA-Bundle adjustment: A modern synthesis

1. 非线性优化

Ceres：<http://ceressolver.org>

Huber核：Robust estimation of a location parameter

1. 边缘化方法（schur）：Sliding window filter with application to planetary landing
2. 重定位

词袋-bag of binary words for fast place recognition in image sequences

描述子-Brief: Binary robust independent elementary features

1. 时间偏移标定：Online Temporal Calibration for Monocular Visual-Inertial Systems

参考链接：

<https://mp.weixin.qq.com/s/ZQGN8FGtskuYiTr24NEW3g>

<https://blog.csdn.net/gyhjlauy/article/details/103063045>

1. 安装

安装ORBSLAM2环境后，只需按代码官网安装ceres即可。注意opencv安装3.3.1版本，两个代码通用。

eigen需要安装3.3.3（官方）以上版本，更换后需，uninstall ceres，然后install ceres，因为都已经make，卸载重装即可，eigen3.2.0使用matrixXd相乘会出错，直接卡住，也不报错

1. 节点连接
2. Monocular+IMU，euroc数据集,见/experiment/vins\_node\_graph。
3. 代码总结（monocular+IMU）
4. 文件简介

功能包，包含cmakelists.txt文件,主函数文件在功能包下，其它源文件按文件夹打包

1. camera\_models

|  |  |  |  |
| --- | --- | --- | --- |
| 生成exe文件 | 源文件src/ | ros节点名 | 功能 |
| Calibrations | intrin\_calib.cc | 非节点 | 校正相机 |
| 生成lib文件 | 源文件src/ | ros节点名 |  |
| camera\_models | 较多，见源码 |  |  |

1. loop\_fusion:project-loop\_fusion，依赖camera\_models

|  |  |  |  |
| --- | --- | --- | --- |
| 生成exe文件 | 源文件src/ | ros节点名 | 功能 |
| loop\_fusion\_node | pose\_graph\_node.cpp | loop\_fusion | 共用的回环检测，配置文件不同 |

1. globle\_fusion：project-global\_fusion

|  |  |  |  |
| --- | --- | --- | --- |
| 生成exe文件 | 源文件src/ | ros节点名 |  |
| global\_fusion\_node | globalOptNode.cpp | globalEstimator |  |

1. vins\_estimator：project-vins，依赖camera\_models

|  |  |  |  |
| --- | --- | --- | --- |
| 生成exe文件 | 源文件(main)src/ | ros节点名 | 功能 |
| vins\_node | rosnodeTest.cpp | vins\_estimator | 根据yaml文件设置：  Euroc:Mono+imu、stereo+imu、stero、VINS-Fusion on car demo。  闭环都用loop\_fusion |
| kitti\_odom\_test | KITTIOdomTest.cpp | vins\_estimator | KITTI Odometry (Stereo)  闭环用loop\_fusion |
| kitti\_gps\_test | KITTIGPSTest.cpp | vins\_estimator | KITTIGPS Fusion (Stereo + GPS)  全局优化用globalEstimator |

其它

1. config：配置参数
2. docker：可以独立执行build，build后使用run.sh运行即可。Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。
3. 重要
4. 看源码步骤：
5. 看主要论文、博客和视频等了解基本原理
6. 下载源码运行一下，了解使用到的工具和库
7. 仔细分析主要论文相关论文的细节，弄懂7、8成，对于新的方法可能要结合源码、论文一点一点吃透
8. 看源码，先根据运行步骤，利清代码的主要函数，与论文对应起来，再逐步搞清各个细节。
9. 以IMU时间戳时间作为真实时间，当新图像帧来时，等待图像帧时间戳+td的IMU时间戳。即图像帧时间戳+td与IMU时间戳对应。
10. 在机身静止时启动程序会比较好，本文方法可以在动态、任意运动初始化，以便通过IMU重力在当前机身的读数得到机体（IMU）坐标系相对世界坐标系（z轴与重力向量方向平行）的位姿。水平放置重力向量读数为（0，0，1）。采用ypr的方式，见十四讲P50,因为yaw角不能确定，所以求出ypr旋转向量Rwi=RwyRypRpr后，用Rw-yRwi去掉yaw角，使世界坐标系x，y轴与初始化时机体坐标系重合。
11. 在归一化平面上，视差需要30/460的距离，极线约束上需要0.3/460的距离。？？
12. 若各状态变量的初始值不好（可以一些值好一些不好），可能陷入全局次优解。故初始值越好估计越准。
13. 光流法可以在减小模糊和低纹理的影响
14. 初始化时得到得参数除了重力向量，其他都不需要太精确，在vio非线性优化能容忍得范围内就好
15. 优化时，imu间隔超过10s认为不准确不进行优化，视觉残差共视大于等于4帧才优化
16. 判断异常点时，对于不同的焦距f相机，应该保持实际成像平面误差不变，故f越大，归一化平面上的误差越小
17. 各关键帧之间时间间隔是不同的，需要通过IMU的预积分不断叠加积分，在边缘化次新帧时要这么做
18. 加速度为0时，两帧之间自由落体位移和速度变化量都为0，那么尺度信息将不会被优化。向前作匀速运动时，只有一个轴有加速度，由论文中至少需要两个轴恢复尺度相矛盾
19. slam中2个重要组成：帧和路标点，局部窗口会不断更新这两个东西。
20. bug
21. estimator.cpp：line773-x向量与所有帧对应，并不单单是关键帧，将x所有kv改成kv\_i,与对应帧对应
22. vins\_estimator和loop\_fusion线程是分开工作的，可以单独运行vins\_estimator
23. vins\_estimator有三个线程
24. 订阅消息处理线程：订阅cam0（只是装入queue）和imu（在处理过程中直接调用imputIMU函数作预处理）话题，其它话题暂时没用
25. Sync\_thread(sync\_process):读取cam0的queue，读一帧删一帧，若queue较常则说明实时性不好。读取一帧后，调用inputImage处理图片。
26. processmeasurements：系统的主线程。
27. loop\_fusion有4个线程
28. 主函数跑一次完成配置后通过话题调用对应函数
29. process:主循环，主要函数addKeyFrame
30. command：键盘按s保存posegraph，按n新建轨迹
31. optimize4DoF:
32. 1
33. 重要源文件

* camera\_models
* camera\_models

1. camerafactory:在未知相机类型前，读取相机配置文件产生对应相机
2. camera:类，各种相机共用部分
3. catacamera:camera继承类，euroc中使用相机模型

* vins\_estimator：
* estimator

1. Estimator：系统类，包含各重要类的实例。主要函数如下：
2. inputIMU：处理新来的IMU帧
3. imputImage：处理新来的图片，进行跟踪，提取特征点
4. processmeasurements：系统的主线程。
5. processIMU：处理IMU数据，进行预积分
6. processImage：处理图片上跟踪到的特征点
7. 初始化过程，初始化共需要11帧，10个间隔，SFM
8. 第1、2帧无论如何设为关键帧
9. 之后帧按跟踪的一样的标准判断是不是关键帧。
10. 第11帧到来时进行第一次初始化计算，若初始化成功进行全局优化（无边缘信息优化，优化完后生成边缘信息）后，更新标志位，进行滑窗操作（边缘化最老帧或次新帧（边缘化时当前帧就是次新帧），窗口保留10帧，等待新帧的到来），下一帧到来进入跟踪B；失败则到d)步

初始化计算步骤：

1. 加速度的标准差小于0.25则表示IMU的刺激不足，平面小车可能需要考虑各个轴，
2. 前提时两帧匹配点大于20个，在11帧从左到右寻找与第11帧归一化平面上平均视差大于30/460（460怎么得来？）且能使用CV库通过基础矩阵求出相对位姿（内点数大于12），继续下一步，记该帧为初始帧L，否则返回false
3. 依次三角化L-10帧与第11帧3D点和计算其位姿：三角化L帧和第11帧的匹配点，然后循环利用3D-2D的pnp得到L+i帧的位姿，再三角化l+i帧与第11帧的匹配点
4. 依次三角化L与L+1-10帧间3D点
5. 依次计算1-L-1帧位姿和三角化1-L-1帧与第L帧3D点
6. 通过三角化尚未被三角化特征点共视帧大于等于2帧两端的两帧得到该特征点3D坐标
7. 进行ceres优化，收敛且最终损失小于0.005，继续下一步
8. 使用跟踪到的3D点求解除关键帧外所有帧的位姿，若能求解出所有帧位姿，则继续下一步

视觉和IMU对齐：窗口第1帧和第11帧间的所有帧，>=11

1. 求解角速度偏差，将视觉得到的相对位移作为测量值。先利用J矩阵，利用cholesky分解（LTL三角分解一种，两边乘转置得到）求出偏差，求解过程中只考虑虚部相等，最后传播得到更准确的预积分。
2. 视觉IMU对齐，g的模与9.8相差小于0.5且s为正继续下一步。
3. 优化g，先令g的模为9.8，方向不变，再再球面优化g，得到结果后，使g模为9.8，方向不变。整个过程相当于出重力向量以外又重新算了所有量。s为正继续下一步，注意得到的是Rclimui，Tclci，Vimuiimui
4. 重传播，利用重力向量得出当前世界坐标系到重力向量世界坐标系的旋转矩阵，更新Rswimui，Pswimui，Vswimui，以第1帧为起点，即Ps[0]=[0,0,0]
5. 清除深度后，再用现有位姿三角化一次，>=4帧观测帧的点使用SVD求解
6. 如果距离上次初始化时间大于0.1s，进行入c）步，否则只进行滑窗操作。滑窗操作会逐渐使滑窗内的帧全部变为关键帧。
7. 跟踪过程：注意是先优化再滑窗，滑窗前需要产生边缘话信息。
8. 通过IMU预积分得到当前帧位姿（已经比较准，可以打印两个数据观察），三角化当前帧与之前帧（因为光流法跟踪是连续的，之前帧就是次新帧）匹配到的非3D点得到更多3D点
9. 进行非线性优化,边缘化老帧时只关心第0帧看到的路标（会牵扯到速度偏差之外的所有变量）和第0帧和第1帧之间的IMU约束（会牵扯到帧相应的位姿、速度偏差），将第0帧位姿和第0帧看到的路标边缘化掉，从而优化其他的在边缘化项的变量，此时以通过边缘化加入了第0帧相关的约束。
10. 添加边缘项，刚初始化完成还没有边缘项，先建立边缘项再优化
11. 添加IMU项，时间间隔大于10s的不添加，当J元素的绝对值大于1e8时说明数据不稳定？
12. 添加视觉项，共视帧少于4帧的特征不添加
13. 优化会检测出异常点并清除，此时是有11帧
14. 进行滑窗操作，保留10帧，以备新帧到来
15. 以最新位姿更新到最新的IMU，imu话题通过数列缓存等待，两个线程通过互锁实现，只在imu话题中对外发布，这里会存在一定延迟。
16. slidewidow
17. 滑掉最老帧：将窗口内11帧左移，第十一帧删除预积分指针后，再新建指针，初始化时还需将all\_image\_frame类实例第1帧删掉。然后，改变start\_frame，不是第一帧只需减一操作即可，第一帧需要判断是不是需要删除该特征（因为观测到该特征的帧数量减1，观测到帧为0时删除（初始化），为1时删除（初始化完成后）），删除该特征观察帧包含第一帧则删除，初始化完成后，还需更新其深度值
18. 滑掉次新帧：删掉窗口中第10帧，将第11帧参数赋给第10帧，并将预积分参数和数据装到第10帧预积分参数和数据末尾，第十一帧删除预积分指针后，再新建指针，最后改变startframe即可，然后，改变start\_frame，删除特征观察帧包含次新帧，则删除，=0时删除该特征
19. 错误检测：是否应该将方差加入？
20. 跟踪到的特征小于2，代码只打印信息，注释掉return
21. 加速度偏差大于2.5
22. 角速度偏差大于1
23. 相机和imu相对位移大于1，相对位移变化不应太大，否则说明优化环节出现问题，代码注释掉
24. 两帧之间位移太大，代码注释掉
25. 两帧之间深度变化太大，代码注释掉
26. 两帧之间转化成轴角所得角度太大，代码中注释掉return，只打印信息
27. 1
28. parameter：常用的配置参数，从配置文件中读取参数的函数。
29. feature\_manager:类，包含featureperframe、featureperid、featuremanager三个类，featuremanager包含featureperid实例list，featureperid包含featureperframe的实例list，featureperframe中包含每帧的7个状态信息（归一化平面坐标3个，原始图像像素坐标2个，归一化平面速度2个）。处理跟踪到的特征点。每个特征点（根据id区分）创建一个实例，在该特征点中创建一个featureperframe的list保存该特征点，在每帧中的位置、速度等信息。特征点只会在连续的帧出现？所以只要知道开始帧就可以了？

判断是否为关键帧：需要保证帧之间视差（两帧之间位移）的同时保证跟踪到足够多的特征（超过30个），两个条件是矛盾的，注意检查的是次新帧与次次新帧之间的视差和特征数。代码没有考虑用IMU补偿

1. 先更新f\_manager
2. 前2帧或跟踪到的特征小于20或算上当前帧>=4个的特征小于40或当前帧新的特征大于0.5倍跟踪到的特征，返回true
3. 对开始于当前帧前2帧之前，止于当前帧或当前帧前一帧的特征点进行补偿，若不存在这种点返回true，否则继续下一步
4. 计算两帧（次新帧与次次新帧）像素距离差，若平均视差大于等于MIN\_PARALLAX

* Initial

1. initial\_alignment:imageframe类，用于初始化时对齐和新建image类，初始化求角速度和vi对齐函数，非类函数
2. initial\_sfm：SFMFeature、ReprojectionError3D结构体，GlobalSFM类。
3. solve\_5pts:motionestimator类。

* featuretracker

1. feature\_tracker:特征跟踪类。重要函数
2. trackImage(line 94):
3. 第一帧时，因为之前没有帧，只会提取MAX\_CNT角点
4. 有了第一帧后开始跟踪，通过正向和反向跟踪增加跟踪的准确性
5. setmask：将已跟踪到的特征点的周围MIN\_DIST距离灰度值设为0，其余为255，用于提取角点保证角点相隔MIN\_DIST
6. goodfeaturetotrack：若跟踪角点不足MAX\_CNT，提取新角点，以便下一次跟踪使用，保证角点相距MIN\_DIST
7. 返回当前帧归一化平面特征点x，y，z=1（去畸变后），原图中特征点坐标x，y，归一化平面速度vx，vy

* utility

1. utility：类，一些通用的矩阵操作。
2. visualization：非类，用于注册和发布话题的函数。主要用于显示，将话题发布给loop和rviz

* factor

1. integration\_base:类，作预积分
2. marginalization：residualblockinfo结构体、threadstruct结构体marginalizationinfo类、margnalizationfactor类。优化函数中边缘化项
3. IMUfactor：IMUfactor类，优化函数中IMU残差项
4. Projectiontwoframeonecamfactor：Projectiontwoframeonecamfactor类，优化函数中视觉残差项

* loop\_fusion

1. pose\_graph\_node:主函数
2. pose\_graph：posegraph类、anglelocalparameterization类。重要函数
3. addKeyFrame：闭环检测处理函数
4. keyframe：keyframe类，briefextractor类。主要用于新建关键帧。
5. parameter.h：重要参数头文件

* utility

1. utility：类，一些通用的矩阵操作。
2. cameraposevisualization：

* thirdparty

1. DBoW：词袋
2. Briefvocabulary：
3. Briefdatabase：
4. query：返回当前帧的询问结果queryresult类。可限制搜索范围，离太近的帧（50帧）不要。询问时只包括FAST特征？
5. add：将新关键帧加入briefdb
6. DUtils
7. Dvision：
8. BRIEF:从已知x，y坐标中提取BRIEF特征，importPairs-提取固定点对，compute-计算特征点特征
9. 1
10. Opencv函数：注意直接读函数注释
11. Filestorage：读取配置文件
12. circle：在图片以某一中心点画圆
13. arrowedline：画箭头
14. scalar：设置颜色
15. goodfeaturetotrack：提取好的角点
16. calcopticalflowpyrLK:使用Lucas-Kanade方法跟踪角点，可以通过增加金字塔层数，增加跟踪的特征点
17. findfundamentalmat：计算两帧之间基础矩阵F，vins中因为归一化平面的K矩阵为单位矩阵，所以得到的是F=E。mask中保存了异常点和正常点的标志。
18. recoverpose：从基础矩阵中计算出位姿T21，返回内点数，输入mask，经过深度为正检查，输出mask。
19. eigen2cv：函数，将eigen矩阵转换为CV矩阵
20. cv2eigen：与上面相反
21. Rodrigues：将旋转矩阵转换成旋转向量
22. solvepnp：函数，求解3D-2D姿态，返回Riw，本文使用迭代的方法，将上一帧位姿作为初始值。
23. solvepnpRansac：使用Ransac方法求解3D-2D，返回Tiw
24. mat.at<>:类型在注释中有
25. FAST：提取FAST角点
26. resize：使用插值方法放缩图片
27. CV\_MAJOR\_VERSION:位于core/version.hpp中的版本号
28. Eigen：C++开源线性代数库，它提供了快速的有关矩阵的线性代数运算，包括解线性方程等功能。没有欧拉角和旋转矩阵之间的转换。代码中使用的基本都是Eigen的矩阵，可以直接调用矩阵类函数。旋转矩阵可以直接赋值给四元数。

<https://blog.csdn.net/yang_q_x/article/details/52383289>

1. Eigen教程5 - 求解稀疏线性方程组：初始化角速度时（initial\_aligment-line36）使用cholesky分解（ldlt）求解系数矩阵为对称矩阵线性非齐次方程，链接中有详细介绍

<https://blog.csdn.net/xuezhisdc/article/details/54634080>

1. C++矩阵库 Eigen 简介

<https://www.cnblogs.com/rainbow70626/p/8819119.html>

1. Eigen踩坑1：Matrix的transpose（矩阵转置）计算之后不能赋值给自身 <https://blog.csdn.net/HW140701/article/details/93592741>
2. Eigen的使用总结2：常用变量

<https://blog.csdn.net/reasonyuanrobot/article/details/86614381>

1. Quaterniond::FromTwoVectors（n1，n2）：（普通）函数，从2个3维向量得到它们之间的旋转四元数q21，2所在坐标系为参考坐标系，用于n1，n2的坐标系
2. toRotationMatrix：类函数，将四元数转换成旋转矩阵。
3. 四元数乘法，直接用乘号。四元数和3维向量相乘也直接用乘号
4. matrix.jacobiSVD：求矩阵的SVD分解，见initial\_sfm-line16
5. q.vec()：四元数虚部
6. vector.normalized():变成模长为1的向量
7. vector.norm()：向量模长
8. a.cross(b)：返回垂直于a、b的向量
9. matrixXd：非方阵或方阵
10. LLT：对矩阵进行LTL分解
11. Eigen::Matrix与array数据转换：.data()将eigen matrix变成数组

<https://blog.csdn.net/weixin_33695450/article/details/94609540>

1. ceres：作非线性优化，slam十四讲11。只接受最小二乘优化，mahalanobis分解需要先进行LTL分解。求导只需求残差项，如10-x。求jacobian矩阵可以使用自动求导，数值方法求导，自己给定jacobian矩阵（使用已知结果理想数据加噪声测试矩阵得可行性）。

<http://ceres-solver.org/nnls_tutorial.html>

1. ROS
2. 话题
3. nav\_msgs::Path
4. nav\_msgs::Odometry
5. sensor\_msgs::PointCloud
6. sensor\_msgs::Image
7. visualization\_msgs::MarkerArray：保存历史轨迹使用，

<http://wiki.ros.org/visualization_msgs>

<https://blog.csdn.net/u013834525/article/details/80447931>

1. header.stamp=ros：：Time（double）
2. set\_logger\_level：一行一行设置对应的要显示的信息
3. cv\_bridge
4. <https://www.ncnynl.com/archives/201702/1299.html>
5. ToCvCopy(image\_msg):返回图片指针，指针->image.clone()复制CV图片
6. CvImage().toImageMsg():将CV格式的图片变成图片topic
7. 1
8. 1