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

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
2. 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。

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. 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。
5. 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。
6. 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。？？
7. 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。
8. 光流法可以在减小模糊和低纹理的影响
9. 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
10. vins\_estimator和loop\_fusion线程是分开工作的，可以单独运行vins\_estimator
11. vins\_estimator有三个线程
12. 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
13. 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。
14. processmeasurements：系统的主线程。
15. 1
16. 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. parameter：常用的配置参数，从配置文件中读取参数的函数。
8. 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，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？

* 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. 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
4. goodfeaturetotrack：若跟踪角点不足MAX\_CNT，提取新角点，保证角点相距MIN\_DIST

* 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。

* factor

1. integration\_base:类，作预积分

* loop\_fusion
* utility

1. utility：类，一些通用的矩阵操作。
2. 1
3. Opencv函数
4. ROS中的cv\_bridge
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. circle：在图片以某一中心点画圆
8. goodfeaturetotrack：提取角点
9. calcopticalflowpyrLK:使用Lucas-Kanade方法跟踪角点
10. 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。
11. recoverpose：从基础矩阵中计算出位姿，返回内点数，输入mask，经过深度为正检查，输出mask。
12. eigen2cv：函数，将eigen矩阵转换为CV矩阵
13. cv2eigen：与上面相反
14. Rodrigues：将旋转矩阵转换成旋转向量
15. 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。
16. 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。
17. 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. 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. 1