**vins\_estimator**

# rosNodeTest.cpp

## 读取配置文件的参数readParameters(config\_file);

### readParameters()将配置文件的参数读取到parameters.h中

## estimator.setParameter()

### 设置estimator的外参、td、g

### 设置FeatureManager的旋转外参

### 设置ProjectionFactor的协方差（此处是协方差的开方）

### 设置FeatureTracker的内参

### 如果是多线程，processMeasurements()会一直工作

## 让发布者注册话题registerPub(ros::NodeHandle &n)

## 订阅者订阅话题

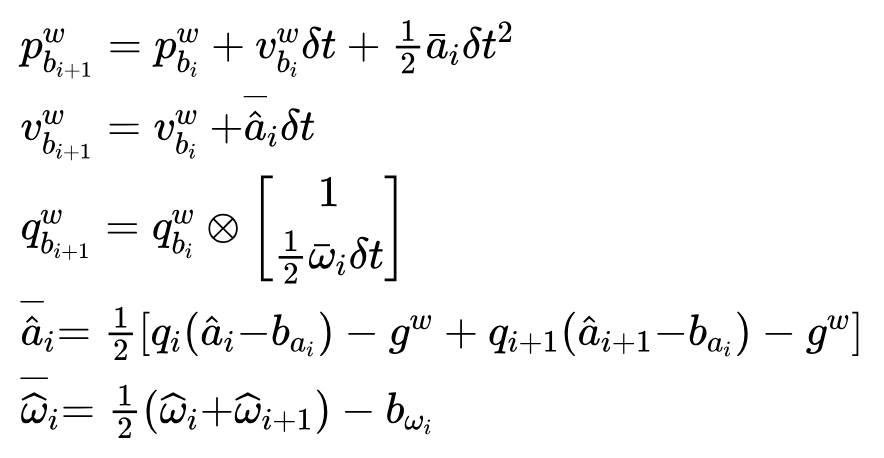
### imu\_callback

* 从消息中获取时间戳、线加速度和角加速度
* 将imu信息输入到estimator中estimator.inputIMU(t, acc, gyr)

• 将值加到accBuf和gyrBuf中

• 根据上一帧的pvq和imu的输入来更新此时的pvqfastPredictIMU(t, linearAcceleration, angularVelocity)

•



• 发布最新的pvq信息pubLatestOdometry(latest\_P, latest\_Q, latest\_V, t)

• 在初始化完成后执行

### feature\_callback

* 获得点云的id、cameraid、3d坐标、像素坐标、像素速度、时间戳信息
* 将特征点信息输入到estimator中estimator.inputFeature(t, featureFrame)

• 将值加入到featureBuf中

• processMeasurements()

• !MULTIPLE\_THREAD

### img0\_callback

* 获得左目的图片消息，存到rosNodeTest的img0\_buf中

### img1\_callback

* 获得右目的图片消息，存到rosNodeTest的img1\_buf中

### restart\_callback重启estimator，重新设置参数

* estimator.clearState()把buf清空，参数设置成初始值
* estimator.setParameter()

### imu\_switch\_callback

* 更改estimator中是否使用imu选项estimator.changeSensorType()

• 如果现在使用了imu，要重启estimator

### cam\_switch\_callback

* 更改estimator中是否使用imu选项estimator.changeSensorType()

## 将图像送给estimatorstd::thread sync\_thread{sync\_process}不断执行

### 双目

* 从img0\_buf和img1\_buf中判断两帧的时间差不超过0.003s
* 通过getImageFromMsg()获得左右目的cv::Mat图像
* estimator.inputImage(time, image0, image1)

### 单目

* 从img0\_buf中取第一帧，getImageFromMsg()获得图像
* estimator.inputImage(time, image)

• featureFrame = featureTracker.trackImage(t, \_img)

• cv::calcOpticalFlowPyrLK()

• 反向追踪

• reduceVector()把没追踪到的点除去

• 使特征点分布均匀setMask()

• 设置mask图

• 将当前追踪的点按追踪次数降序排

• 清空cur\_pts、ids、track\_cnt

• 通过画圈的方式使特征点均匀，重新填充cur\_pts、ids、track\_cnt

• cv::goodFeaturesToTrack()提取shi-tomas角点

• 把新角点增加进cur\_pts等

• undistortedPts(cur\_pts, m\_camera[0])将像素坐标恢复成归一化坐标放到cur\_un\_pts中

• ptsVelocity(ids, cur\_un\_pts, cur\_un\_pts\_map, prev\_un\_pts\_map)求像素速度（实际上是在归一化平面求）把结果放入pts\_velocity

• 把id和当前帧像素坐标对应

• 通过前后两帧像素差除以时间求速度

• 左右目光流追踪

• 反向左右目光流追踪

• 和单目相似，更新右目信息

• drawTrack()

• prev=curhasPrediction = false

• 将归一化坐标、真正的像素坐标、归一化平面xy的速度封装成featureFrame，如果有右目，再来一次，返回featureFrame

• 不是第0帧

• 双目相机

• 获得左目图像，pubTrackImage(imgTrack, t)发布消息用于可视化

• featureBuf.push(make\_pair(t, featureFrame))

• processMeasurements()

• feature为当前帧的特征点，加上时间戳curTime是当前时间

• 循环等待当前时刻的imu数据到来

• 把前一帧和当前帧之间的buf数据取出放Vector中 getIMUInterval(prevTime, curTime, accVector, gyrVector)

• initFirstIMUPose(accVector)

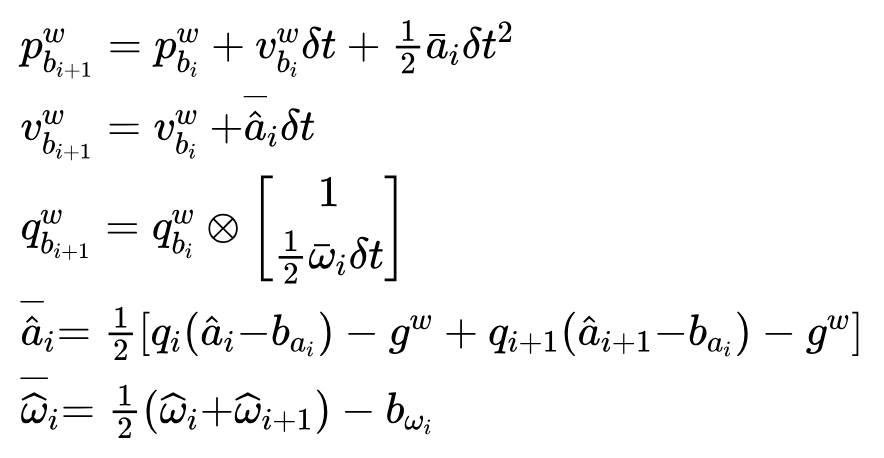
• 取这段时间的平均加速度作为重力

• 让加速度和重力对其，修正Rs[0]

• 根据上一图像帧的位姿和之间的imu数据进行粗略的预积分得到现在图像帧的pvqprocessIMU(accVector[i].first, dt, accVector[i].second, gyrVector[i].second)

• 把dt，加速度、角速度加到当前帧的pre\_integrations中，再加入到当前帧的对应的buf中

•



• processImage(feature.second, feature.first)

• addFeatureCheckParallax(frame\_count, image, td)判断边缘化最老帧还是次新帧

• 更新feature数组

• compensatedParallax2(it\_per\_id, frame\_count)算视差

• 在归一化平面计算点的距离

• 通过视差判断是否为关键帧

• 将当前帧封装成ImageFrame，装进all\_image\_frame

• 获得特征点在两帧下的归一化坐标getCorresponding(frame\_count - 1, frame\_count)

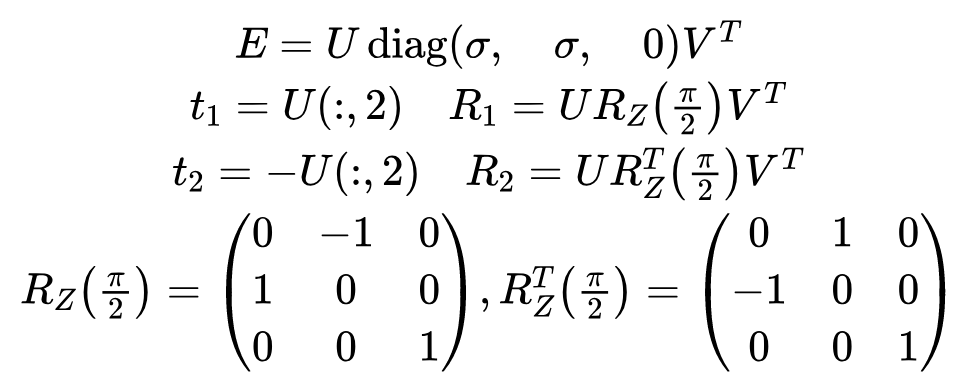
• CalibrationExRotation(corres, pre\_integrations[frame\_count]->delta\_q, calib\_ric)

• 根据对极约束求位姿，8点法solveRelativeR(corres)

• 通过两帧的像素坐标(归一化坐标XY)求本质矩阵

• 将E（-E）分解成R和tdecomposeE()

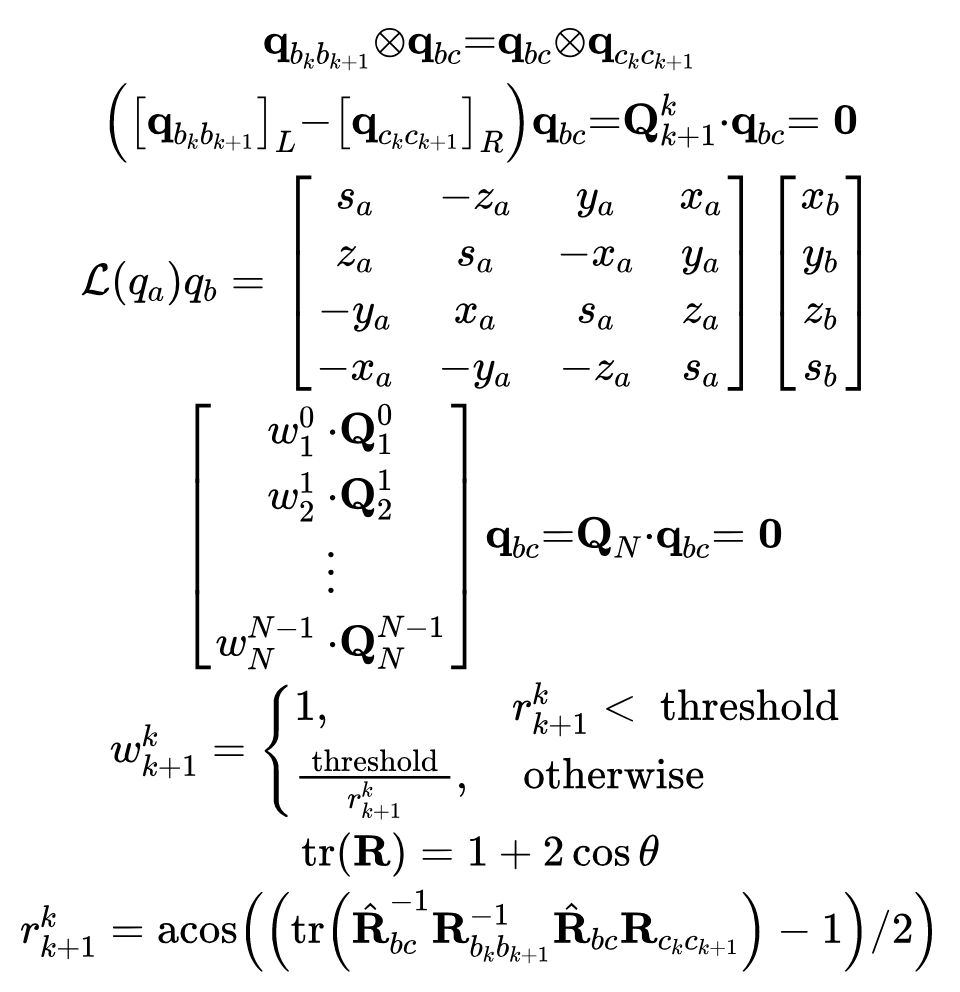
•



• 检验四组解testTriangulation(ll, rr, R1, t1)获得真正的R

• 通过cv::triangulatePoints()检验深度为正的点的比例

• 求qbc，详细看《手写VIO》第七讲第10页



• 如果成功把结果给calib\_ric\_result

• 初始化

• 单目+imu

• initialStructure()

• imu激励是否足够

• 计算所有帧的加速度的标准差

• 创建Q、T、sfm\_f、sfm\_tracked\_points

• 把feature信息填充到sfm\_f中

• 确定参考帧l，求最新帧到它的位姿relativePose(relative\_R, relative\_T, l)

• 遍历滑窗，获得第i帧和最新帧的共视点

• 计算共视点的平均视差并判断

• 对极约束，五点法求位姿solveRelativeRT(corres, relative\_R, relative\_T)

• RANSAC求解本质矩阵E

• 对极约束恢复位姿cv::recoverPose(E, ll, rr, cameraMatrix, rot, trans, mask)

• SFMsfm.construct(frame\_count + 1, Q, T, l, relative\_R, relative\_T,sfm\_f, sfm\_tracked\_points)

• 创建c\_XXX数组表示第l帧相机在别的帧相机系下的表示

• 三角化l和最新帧triangulateTwoFrames()

• 遍历特征点，如果是这两个帧的共视点triangulatePoint()，求出在l相机系的空间坐标，更新sfm\_f

• pnp求l+1帧的位姿solveFrameByPnP()

• 获得特征点的3d坐标和2d坐标

• 将cv::Eigen转化为cv::Mat

• cv::solvePnP()求旋转向量和平移向量

• 三角化l+1帧和最新帧triangulateTwoFrames()

• 三角化l帧和l+1帧

• pnp求l-1帧

• 三角化l-1帧和l帧

• triangulatePoint()三角化剩余点，更新sfm\_f

• 全局ba

• 用ceres求解，添加参数块，将先验设为恒定

• ReprojectionError3D定义残差

• 重投影-光流

• 添加残差块，ceres求解

• 更新滑窗每一帧的q和T，填充sfm\_tracked\_points

• 遍历l+1到次新帧重复执行

• 遍历l+1到次新帧重复执行

• 遍历l-1到最老帧重复执行

• pnp求所有帧

• 如果当前帧在滑窗内，更新ImageFrame的R和T

• 如果当前帧不在滑窗内，找这个帧被三角化的特征点，获得3d、2d坐标

• cv::solvePnP()求解该帧的位姿

• 更新该帧ImageFrame的R和T

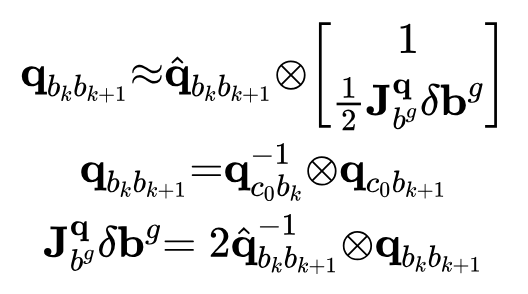
• 遍历所有帧重复执行

• 视觉imu对齐visualInitialAlign()

• VisualIMUAlignment(all\_image\_frame, Bgs, g, x)

• solveGyroscopeBias(all\_image\_frame, Bgs)

•

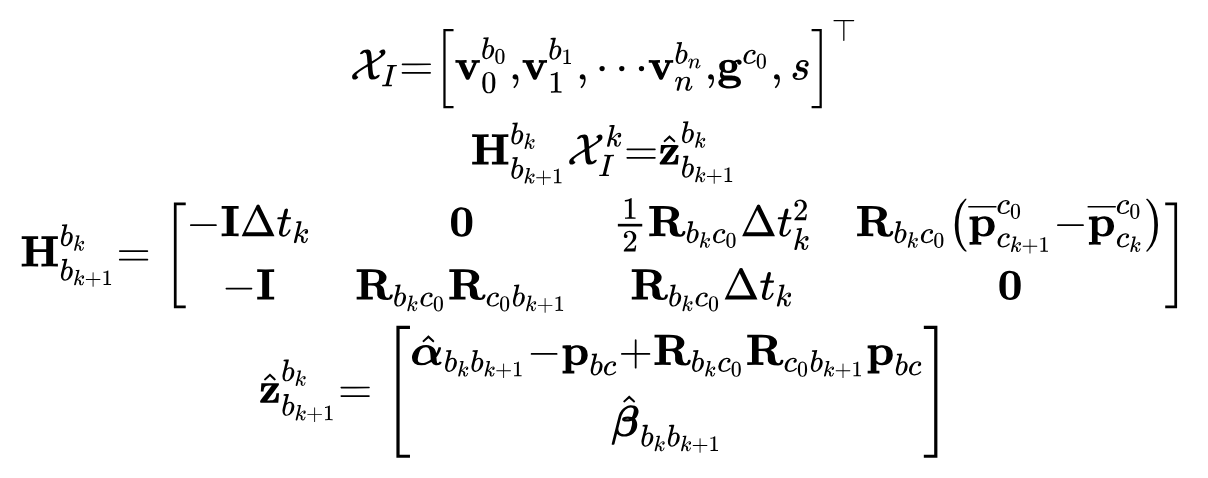


• ldlt求解更新滑窗内的Bgs数组

• 所有帧重新预积分repropagate(Vector3d::Zero(), Bgs[0])

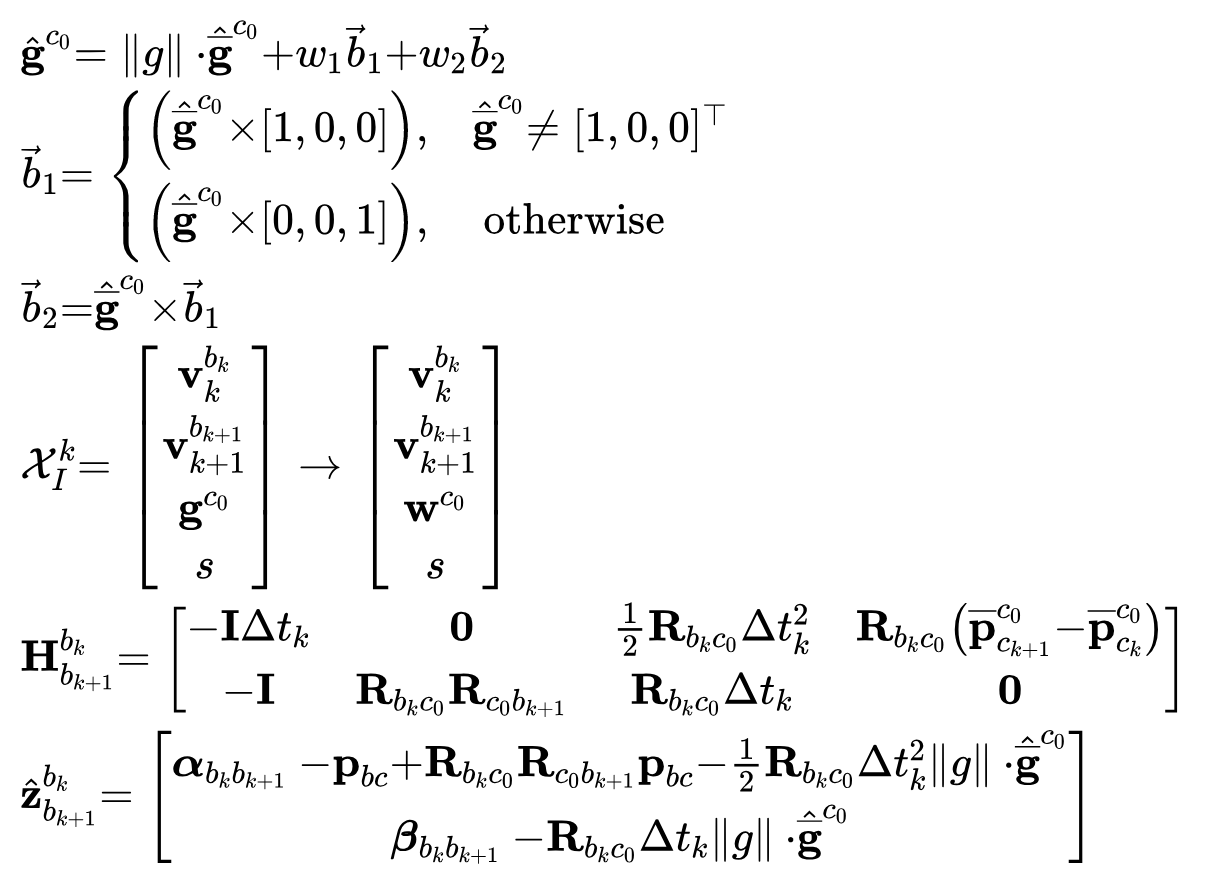
• LinearAlignment(all\_image\_frame, g, x)

•



• RefineGravity(all\_image\_frame, g, x)

•



• 迭代4次

• 用尺度、优化后的速度更新滑窗内的Ps、Rs、Vs，并把坐标系从第l帧相机系转到世界系

• 对滑窗内的帧重新预积分repropagate(Vector3d::Zero(), Bgs[i])

• 求将第l帧g变换到世界系的g的旋转，把参考系从第l帧变到第0帧body系（世界系），现在的Rs、Ps、Vs是第i帧imu在第0帧body系

• clearDepth()清楚feature数组FeaturePerId的深度

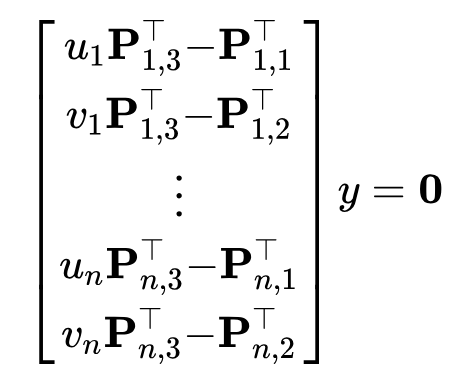
• 重新求深度（相对于世界系）triangulate(frame\_count, Ps, Rs, tic, ric)

• 获得首次观测到这个特征点的帧获得左目右目的像素坐标triangulatePoint()三角化求世界系的空间坐标

• feature数组的FeaturePerId的深度是在首次观测到该特征点的帧的相机系的z

• 三角化的两帧时首次观测到该特征点的帧和它的下一帧

• 详细看手写VIO第6讲25页



• SVD求解

• 双目

• 单目

• 遍历滑窗内的特征点

• optimization()

• updateLatestStates()

• slideWindow()

• slideWindow()

• 初始化成功

• 初始化失败

• 双目+imu

• initFramePoseByPnP(frame\_count, Ps, Rs, tic, ric)

• 遍历滑窗特征点，获得世界系3d坐标和最新帧的像素坐标

• solvePoseByPnP()求位姿

• 主要是格式转换旋转向量和旋转矩阵转换cv::solvePnP()

• 更新最新帧的Ps、Rs

• triangulate(frame\_count, Ps, Rs, tic, ric)

• 更新all\_image\_frame的RT（all\_image\_frame都在滑窗内）

• solveGyroscopeBias(all\_image\_frame, Bgs)

• repropagate(Vector3d::Zero(), Bgs[i])重新预积分

• optimization()

• updateLatestStates()

• slideWindow()

• 仅双目

• initFramePoseByPnP(frame\_count, Ps, Rs, tic, ric)

• triangulate(frame\_count, Ps, Rs, tic, ric)

• optimization()

• updateLatestStates()

• slideWindow()

• 把当前帧信息pvqb给下一帧

• initFramePoseByPnP(frame\_count, Ps, Rs, tic, ric)

• 估计滑窗特征点的深度三角化滑窗中的特征点，把结果给feature数组featureperid的estimated\_depth，这个深度是对于首次观测到它的帧的相机系triangulate(frame\_count, Ps, Rs, tic, ric)，把

• optimization()

• vector2double()

• 填充para\_Pose、para\_SpeedBias、para\_Ex\_Pose、para\_Feature、para\_Td其中para\_Feature是逆深度，取自于featurePerId.estimated\_depth的倒数，且只有4帧及以上追踪到这个特征点才考虑

• 添加参数块

• 添加残差块（imu、视觉、先验）

• ceres::Solve()

• double2vector()

• 把double数组填进Ps、Rs等

• setDepth(dep)

• featurePerId的estimated\_depth，通过深度正负设置featurePerId的solve\_flag为1或2，为之后的removeFailures()使用

• 边缘化

• 边缘化最老帧

• vector2double()

• 把最老帧有关的因子 （先验因子、imu因子、视觉因子和drop\_set增加进marginalization\_info中

• 求雅克比和残差marginalization\_info->preMarginalize()

• 构建H和b，反解出J和bmarginalization\_info->marginalize()

• 填充addr\_shift

• 新值赋给旧值

• 边缘化次新帧

• vector2double()

• 边缘化先验因子中和次新帧有关的变量，通过drop\_set的形式添加进marginalization\_info中

• marginalization\_info->preMarginalize()

• marginalization\_info->marginalize()

• 填充addr\_shift

• 新值赋给旧值

• 外点检测outliersRejection(removeIndex)

• 获得这个特征点首次观测的帧和共视帧

• 求重投影误差reprojectionError

• 如果是双目，要增加和共视帧右目的重投影误差

• 平均误差大于阈值，认为是外点

• 遍历滑窗每个特征点，只考虑被追踪4帧及以上的点

• 去除外点removeOutlier(removeIndex)

• 根据removeIndex把feature中对应的特征点去除

• removeOutliers(removeIndex)

• 在featureTracker中把prev\_pts、ids、track\_cnt对应的点去除

• 预测下一帧特征点出现的位置predictPtsInNextFrame()

• 获得的当前帧和上一帧的位姿getPoseInWorldFrame()

• 假设下一帧的位姿和当前帧到前一帧的位姿相同，进而求下一帧位姿

• 通过位姿重投影，形成预测点

• setPrediction(predictPts)

• 填充featuretracker的predict\_pts，存的是真正的像素坐标，用于下一帧光流追踪

• slideWindow()

• 最老帧

• 信息前移Ps、Rs、Headers、pre\_integrations、dt\_buf、linear\_acceleration\_buf、angular\_velocity\_buf、Vs、Bas、Bgs

• 第10帧信息赋值给第11帧，重新创建第11帧的预积分，清空第11帧的buf

• 从all\_image\_frame中删除最老帧之前的帧

• slideWindowOld()

• removeBackShiftDepth(R0, P0, R1, P1)

• start\_frame--

• 更新featurePerId和featurePerFrame

• 重新计算featurePerId的estimated\_depth

• start\_frame!=0

• start\_frame=0

• removeBack()

• start\_frame--

• 更新featurePerId和featurePerFrame

• start\_frame!=0

• start\_frame=0

• 已经初始化了

• 还没初始化

• 次新帧

• 用最新帧信息覆盖次新帧

• 对于imu，最新帧的buf直接拼接到次新帧上（此时的次新帧是最新帧，信息已被覆盖）

• 对第11帧重新创建预积分，清空第11帧的buf

• slideWindowNew()

• removeFront(frame\_count)

• 更新featurePerId和featurePerFrame

• 把滑窗中深度为负的点移除removeFailures()

• 将滑窗中的Ps填充进key\_poses数组中，用于可视化

• 更新last\_RP、last\_R0P0

• updateLatestStates()

• 更新latest\_xxx变量，用于fastpredictimu()

• 用最新的数据不断fastpredictimu()

• 没有标定外参

• 仅一次

• 没有imu

• ! MULTIPLE\_THREAD

• printStatistics()输出数据

• 向rziv发布

• 只执行一次

• 在图像帧之间多次调用

# IntegrationBase

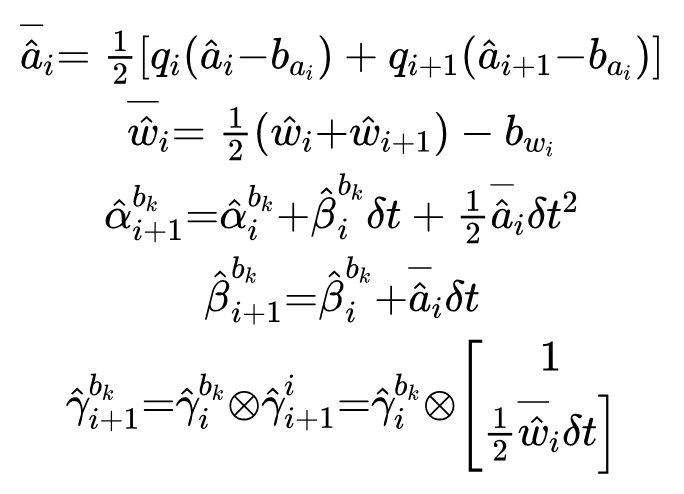
## repropagate()

### 把预积分相关的变量设为初始值

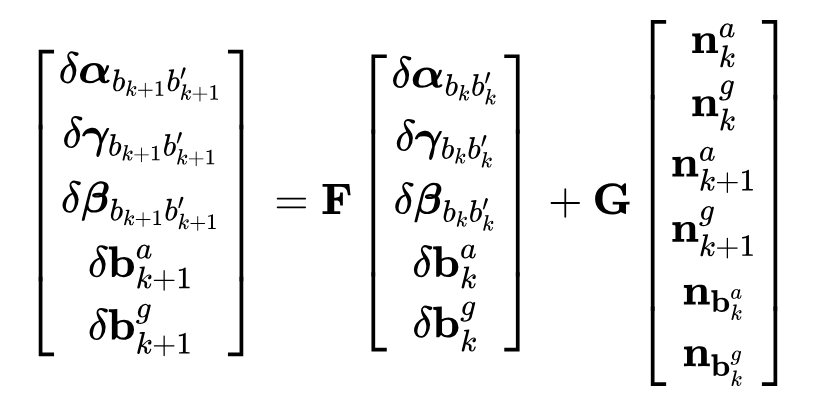
### 遍历所有imu帧，预积分propagate(dt\_buf[i], acc\_buf[i], gyr\_buf[i])

* midPointIntegration(\_dt, acc\_0, gyr\_0, \_acc\_1, \_gyr\_1, delta\_p, delta\_q, delta\_v, linearized\_ba, linearized\_bg, result\_delta\_p, result\_delta\_q, result\_delta\_v, result\_linearized\_ba, result\_linearized\_bg, 1)

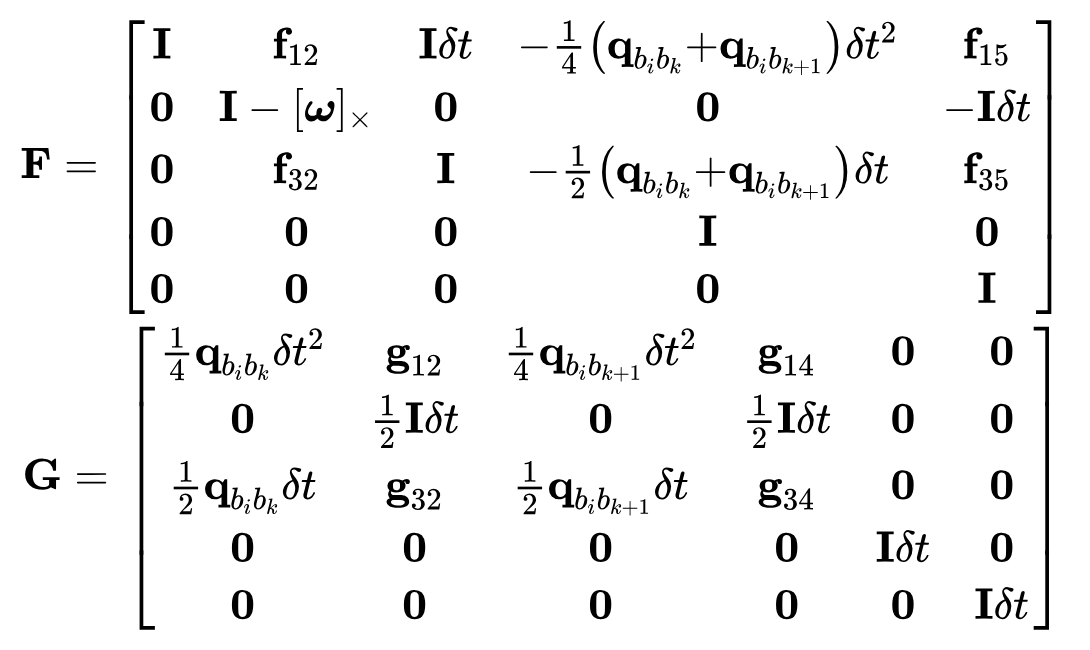
•



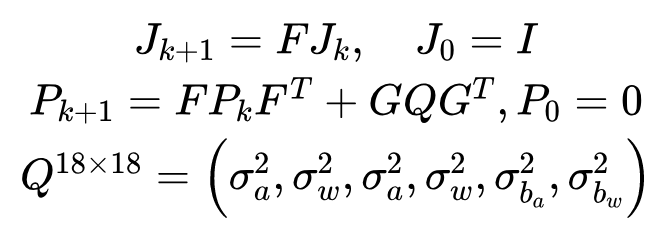
•



•



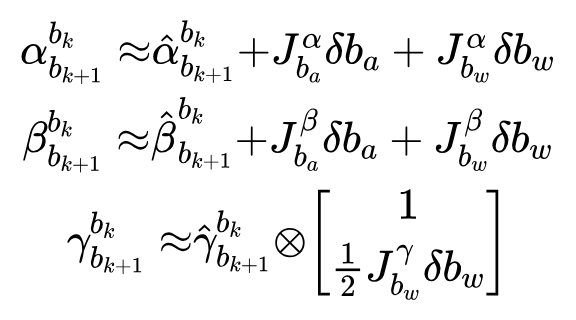
•



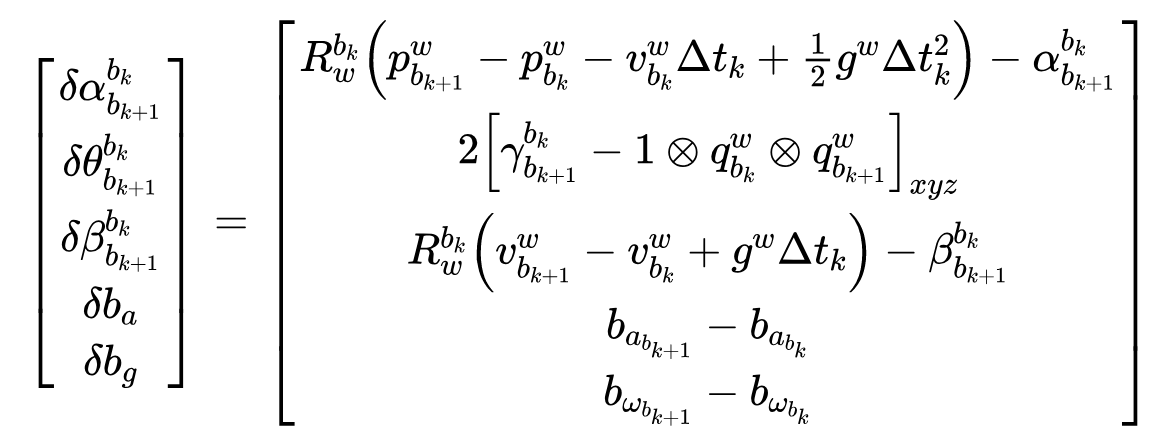
* 新值赋给旧值

## evaluate()

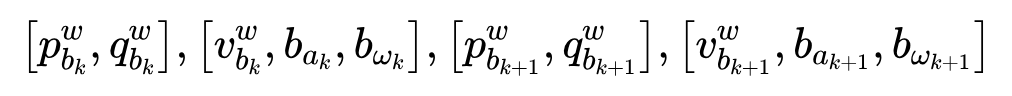
### 



### 



### 

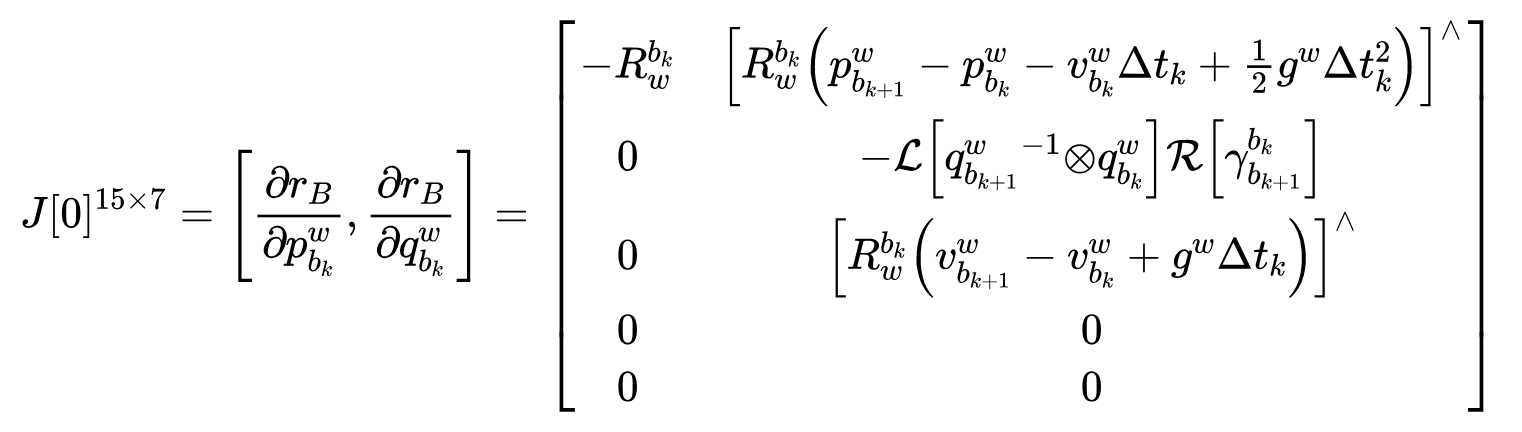


# IMUFactor

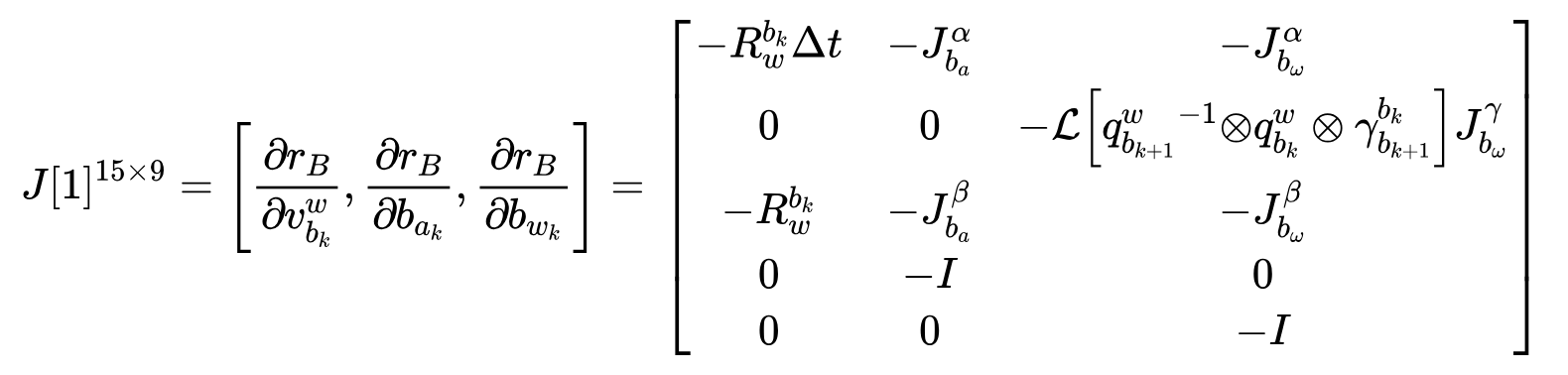
## evaluate()

### 计算残差pre\_integration->evaluate(Pi, Qi, Vi, Bai, Bgi, Pj, Qj, Vj, Baj, Bgj)

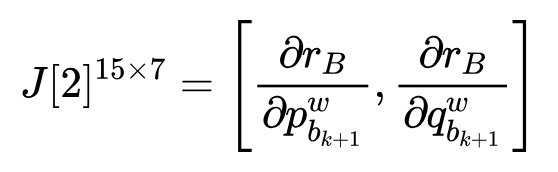
### 



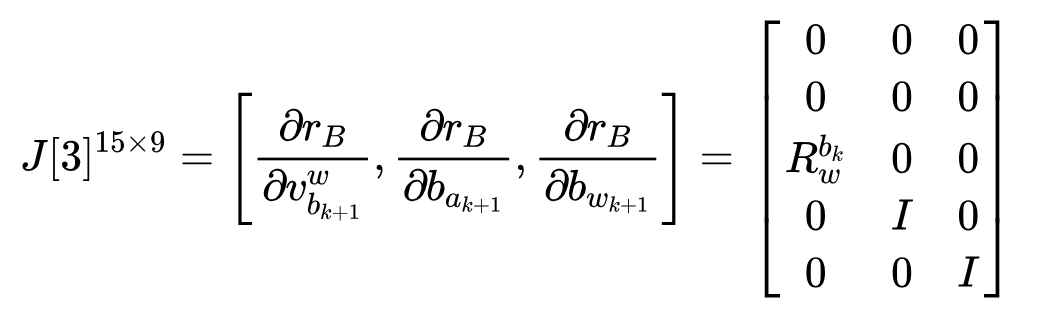
### 



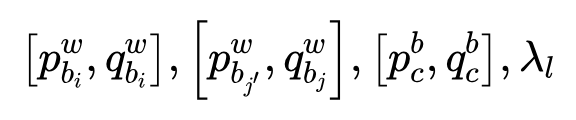
### 

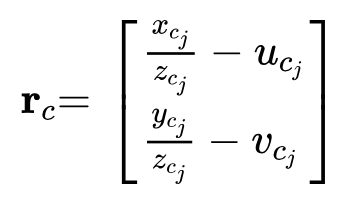


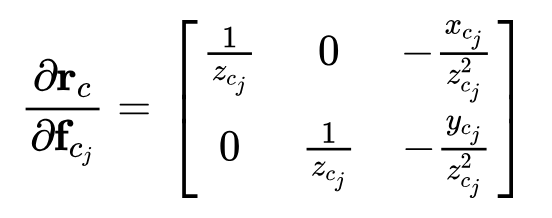
### 

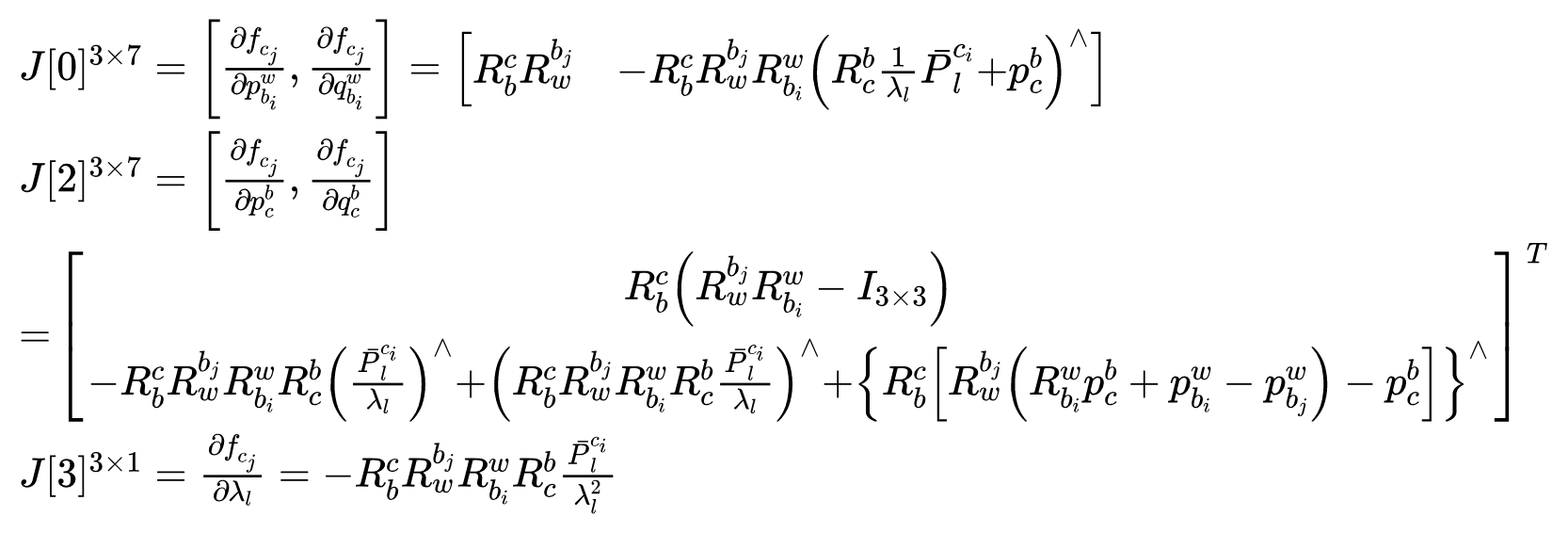


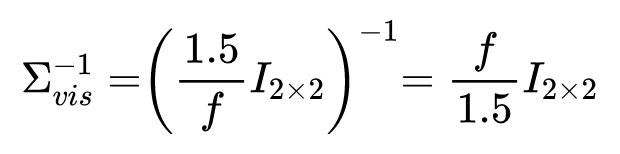
# ProjectionFactor











# MarginalizationFactor

## ResidualBlockInfo

### Evaluate()

* 调用对应残差的Evalua()函数求残差和雅克比cost\_function->Evaluate(parameter\_blocks.data(), residuals.data(), raw\_jacobians)
* 如果有核函数，残差和雅克比需要缩放

## MarginalizationInfo

### addResidualBlockInfo()

* 填充parameter\_block\_size和parameter\_block\_idx，先记为0

### getParameterBlocks()

* 将边缘化保留的变量存入到keep\_block\_xxx中

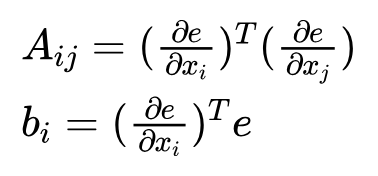
### preMarginalize()

* 调用对应的Evaluate()求残差和雅克比
* 填充parameter\_block\_data
* 遍历参与边缘化的因子

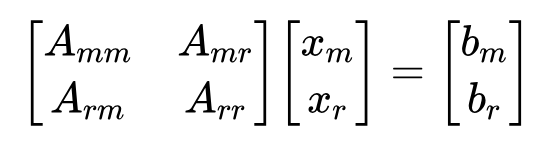
### marginalize()

* 把边缘化掉的变量排在前面，在parameter\_block\_idx中体现
* 构造H和b，舒尔补求新的H和b

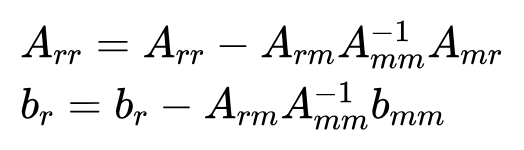
•



•



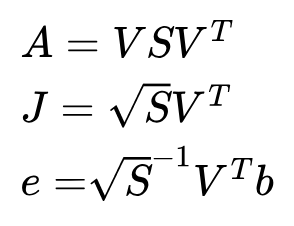
•



• 遍历每个因子

* 将H和b分解出J和e

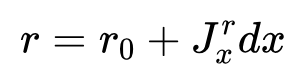
•



## MarginalizationFactor

### Evaluate()

* 计算残差



* 雅克比就是marginalization\_info的雅克比