# okvis算法详解

参考博客：<https://blog.csdn.net/datase/article/details/78573674>

## 1、initPoseFromImu

已知imu的加速度测量值计算第一帧图像在地球坐标系下的姿态:

这里作者使用了几个假设：假设从系统开始运行到第一帧图像出现的时间段内，imu基本上处于静止状态，则imu测量得到的加速度就只有重力加速度。

我们知道重力加速度在地球坐标系下一定是。那么我们imu得到的重力加速度是在imu坐标系下的。我们想得到在第一帧图像出现时世界坐标系移动到imu坐标系下的姿态变换，即将imu坐标系的点变换到世界坐标系下的姿态矩阵。

而且我们已经知道一个结论：对于姿态矩阵TWI，拥有三个物理意义：a、将W坐标系移动到I坐标系的位姿变化；b、能够将I坐标系下的点变换到W坐标系下；c、对于空间中一个点，拥有W坐标系和I坐标系下的坐标，并且将这两个坐标放在同一个坐标系下那么我们能够求得W坐标系下的坐标变换到I坐标系下的坐标的变换，那么这个变换等于TWI的转置。

我们这里使用第三个物理意义，我们已经获得了重力在地球坐标下的坐标和在imu坐标系下的坐标，我们想要求得TWI，那么我们把两个坐标放在同一个坐标系下，然后计算从W坐标系下的坐标移动到I坐标系下需要的旋转矩阵，那么这个旋转矩阵的转置就是RWI。

单位化(叉乘imu加速度)= 旋转轴

点乘imu加速度 能够求得夹角

使用上述得到的旋转轴和夹角我们能够求得RIW。

## 3、bool Frontend::detectAndDescribe

在我们得到图像后我们需要对其进行特征点的提取和描述子的提取，作者使用的是现有的brisk方法。但是有一点需要特殊说明的，每一个特征点都有一个属性叫做angle(double类型)，用来描述这个特征点的主方向。那么这里的主方向是如何计算的呢？

首先我们根据IMU的测量值得到这一帧图像在世界坐标下的姿态逆RWC-1

然后我们构建一个BA方程，然后计算关于这个地图点的雅克比J2\*3

eg\_projected2\*1 = J2\*3\*RWC-1\*

angle = artan[eg\_projected(1)/eg\_projected(0)]

**注：**还有一点需要特殊说明的就是上述的BA方程是如何构造的：此时我们只有图像的像素坐标，没有这个像素坐标对应的三维地图点那么如何构造BA呢？

首先根据像素坐标得到畸变的归一化平面坐标，然后将这个畸变的归一化平面坐标去畸变，得到了没有畸变的归一化平面坐标。我们将这个坐标当成三维坐标。

projectHomogeneous函数内容：我们得到相机下的三维坐标后，然后这里的BA优化方程不是slam14讲中的理想针孔模型而是考虑了畸变函数，在求解雅克比时同时需要求解对这个畸变函数的雅克比，幸运的是opencv提供了一个函数能够直接计算得到。同时将这个投影的雅克比给计算了出来，计算雅克比的函数是project函数：



distortion的雅克比



(x,y,z)是三维点在相机下的坐标。

**注：计算得到的特征点方向只在绘图中使用到了。**

## 4、match

### 4.1 doSetup()

我们已经通过imu测量值得到了当前帧的预估姿态，并且已经对当前帧提取过特征点了。这个函数的作用是实现两张图像的3D-2D和2D-2D匹配初始化工作，判定哪些特征点需要参加匹配哪些不需要。筛选依据大致如下：

如果是3d-2d: 对于图像A，没有地图点对应的特征点不参与匹配

对于图像B, 特征点已经有地图点和其匹配的不参与匹配

如果是2d-2d: 对于图像A,有地图点和其对应则不参与匹配

对于图像B，有地图点和其对应则不参与匹配

1. 首先获取信息矩阵：这里的W是信息矩阵，

当前帧的速度和bias在优化窗口中并且要匹配的两个图像不是来自于一个双目帧则，v是当前帧的速度。

如果不是当前帧则，第一帧图像时进入这个条件。

1. resetFrames 更新矩阵ProbabilisticStereoTriangulator类中的H\_矩阵；

更新了ProbabilisticStereoTriangulator类中的 sigmaRay\_变量.

Tm=[tm,qm]为B图像相对于A图像的相对姿态

Ts=[ts,qs] 为待优化的B帧相对A帧的相对姿态(初值设置等于Tm)

误差 =，这里的右下标x,y,z表示括号中四元数的xyz维度

此误差的雅克比使用PoseError::EvaluateWithMinimalJacobians函数计算得到



Q-1和Q+查看QK论文公式15

然后作者使用更新矩阵ProbabilisticStereoTriangulator类中的变量H\_矩阵



更新了ProbabilisticStereoTriangulator类中的 sigmaRay\_变量



3、对图像A进行处理

**如果是3d-2d:**

遍历A图像的特征点：

该特征点没有地标点或者landmarksMap\_没有对应的地图点，直接跳过，continue。

这个地图点没有被加入到ceres的参数块中，跳过该点匹配，continue。

将世界坐标系下的地图点投影到B图像中，投影失败则直接跳过，continue。

如果这个地图点观测数量小于2个则跳过该点的匹配并设置标志位还没有将其加入到ceres的参数块中，continue。

这个特征点有对应的地图点则将其投影到B相机坐标系下得到B相机下的像素点pb,也可以得到投影的雅克比J,使用这两个参数更新VioKeyframeWindowMatchingAlgorithm类中的对象

projectionsIntoBUncertainties\_= J\*(Cov.左上角3\*3矩阵)\*JT.

projectionsIntoB\_= pb

raySigmasA\_=,d是这个特征点对应的直径大小，f表示相机的焦距

**如果匹配类型是2d-2d:**

遍历A图像的特征点：

更新VioKeyframeWindowMatchingAlgorithm类中的对象raySigmasA\_=。

如果图像A这个特征点看到了地图点，continue。

如果在世界地图中有地图点被这个帧观测到了并且这个地图点已经压入到了ceres的参数块中，匹配过程直接跳过该特征点；

4、对图像B进行处理

**如果是3d-2d:**

遍历B图像的特征点：

是否这个特征点已经有地图点和其对应且landmarksMap\_结构中有地图点，且这个地图点观测到了这个特征点，不对这个特征点进行匹配。其他情况都不能跳过对这个特征点的匹配。

更新VioKeyframeWindowMatchingAlgorithm类中的对象

raySigmasB\_=,d是这个特征点对应的直径大小，f表示相机的焦距.

**如果是2d-2d:**

遍历B图像的特征点：

更新VioKeyframeWindowMatchingAlgorithm类中的对象

raySigmasB\_=,d是这个特征点对应的直径大小，f表示相机的焦距.

特征点还没有对应的地标点，则认为这个特征点需要参加匹配，continue;

如果在在landmarksMap\_结构中有这个地图点并且这个地图点已经压入到了ceres的参数块中，则认为可以跳过。否则认为不能跳过。

### 4.2 matchBody

作者生成了四个线程做匹配，

#### a、doWorkLinearMatching

主要是完成匹配工作

#### b、setBestmatch

这个函数的主要作用是根据上面函数匹配得到的结果：向ceres添加残差函数，向ceres中添加地图点的参数块，更改ceres中参数块的读数。

**如果匹配类型是2d-2d:**

1.是否两个特征点都看到了地图点，如果是则直接返回。

2.**stereoTriangulate**：三角化得到特征点在A相机坐标系下的坐标,并得到是否这两个射线平行，如果平行直接返回。并通过A相机在世界坐标系下的位姿，我们能计算出这个地图点在世界坐标系下的坐标pw，在后面会用到这个变量。

3. **getUncertainty**：判断得到是否误差在规定的范围内

a. 将三角化的点重投影到A帧中计算得到雅克比J1min\_A(对应5.4中的J1min)和Jmin\_A(对应5.4中的Jmin，但是只包括了对地图点求雅克比的项)，其中在计算的过程中信息矩阵设置为=，d为对应的特征点的邻域直径。

b再将三角化的点重投影到B帧中计算得到雅克比Jmin\_B(对应5.4中的J1min)和Jmin\_B(对应5.4中的Jmin，只包括了对地图点和相机姿态求雅克比的项)，其中在计算的过程中信息矩阵设置为=，d为对应的特征点的邻域直径。这里需要注意的是我们这里的点是A相机坐标系下的，B帧的位姿也是相对于A帧的。这里作者使用了一个小技巧：当把三角化的点投影到B时，其坐标并不是之前三角化的结果pA而是，这里的是B帧坐标系移动到A帧的位移。

c. 如果投影到B帧时的残差平方小于4 则outCanBeInitialized=false，否则为true。

d. 使用上述得到的结果计算得到ProbabilisticStereoTriangulator 类中的H\_矩阵:



如果H\_矩阵不可逆则输出的outCanBeInitialized=false。

e.最终返回得到不确定度矩阵 = 矩阵的右下角3\*3矩阵。

4. 如果左右两个特征点都没有地图点和其对应，则新建一个地图点并设置这两帧图像看到了这地图点。准备向ceres中插入这个地图点的参数快。

如果A图像特征点有地图点对应并且landmarksMap\_结构中没有这个地图点，则准备向ceres中插入这个地图点的参数快

如果B图像特征点有地图点对应并且landmarksMap\_结构中中没有这个地图点，则准备向ceres中插入这个地图点的参数快

5. 如果准备向ceres中插入参数块，则向ceres中添加这个地图点参数块.更新landmarksMap\_结构中的信息即新加入一个地图点。

如果不准备向ceres中插入参数块则(如果A或B图像中的特征点有地图点对应并且这个地图点都已经存在于landmarksMap\_状态空间中了)，从mapPtr\_中提取出地图点对应的参数块指针，并修改其值为pw，修改landmarksMap\_中地图点的值。

6. 如果A帧的特征点没有看到地图点则，则设置A帧的特征点看到了这个地图点，向ceres中添加BA的残差方程，更新landmarksMap\_的信息,让地图点看到这个特征点。

如果B帧的特征点没有看到地图点则，则设置B帧的特征点看到了这个地图点，向ceres中添加BA的残差方程，更新landmarksMap\_的信息,让地图点看到这个特征点。

7、从mapPtr\_中提取出地图点对应的参数块指针，并修改其值pw，修改landmarksMap\_中地图点的值。

**如果匹配类型是3d-2d:**

1·、提取B图像中的特征点

2、error = 与A中特征点匹配的B图像中的坐标- B帧中特征点的观测值（真实值）

3、计算误差= ,这里的详见doSetup函数。d是B图像特征点的邻域大小。

4、如果误差>4直接返回

5、如果矩阵的模超过,，则直接返回。

6、设置B帧看到了这个地图点，

7.如果这个地图点之前没有被B观测到过则向ceres中添加BA的残差方程，更新landmarksMap\_的信息即让地图点看到这个特征点。

#### c、stereoTriangulate-setBestmatch调用的函数

注意了okvis使用的三角化方法和orb代码中的不同。

1.得到两个图像的坐标，分别得到不带畸变的归一化的平面坐标。

2.使用triangulateFast函数判断是否平行和有效，并得到三角化的在A相机坐标系下的三维坐标点。

3.如果有效则使用computeReprojectionError4分别将三维坐标点投影到图像中计算误差，如果任何一个图像中的误差大于4，则认为无效。

<https://blog.csdn.net/yg838457845/article/details/81293286>

<https://www.cnblogs.com/liufuqiang/p/6792763.html>

**triangulateFast**

输入参数是A相机到B相机的位置变化t，由两个特征点形成的两条在A相机坐标系下的射线。这个函数的作用是判断两个射线是否平行，三角化能否有效的进行。**computeReprojectionError4**

此时我们已经得到了三角化的点在A相机坐标系下的坐标，现在我们需要计算误差。

我们将这个三角化的点重新投影回相机A然后得到一个像素坐标p, 此时这个特征点有自己的二维像素坐标pm,则误差公式如下：

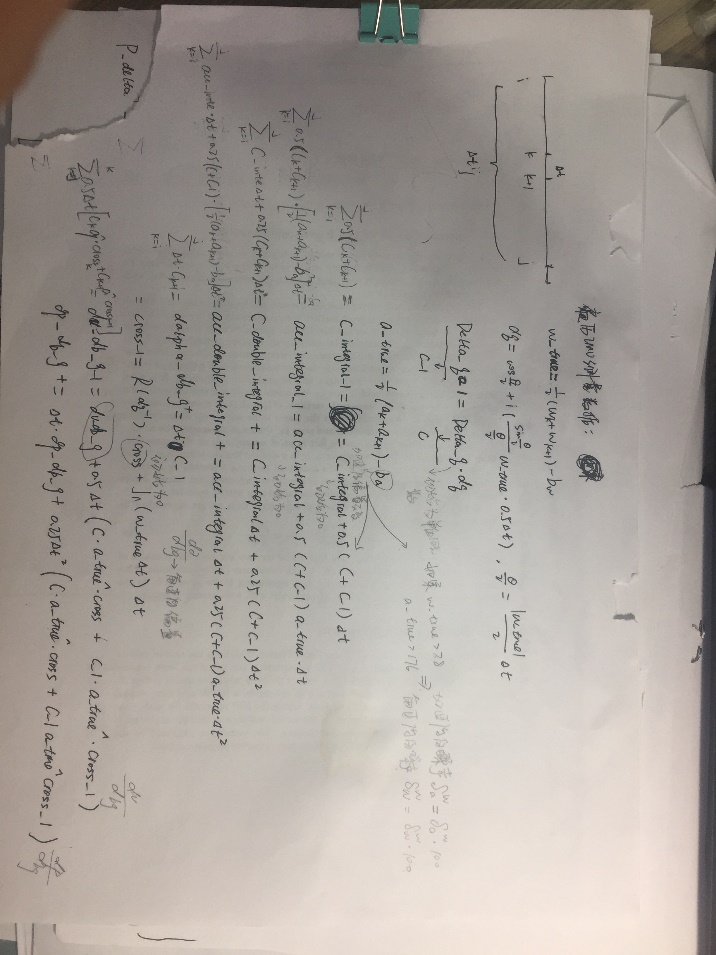


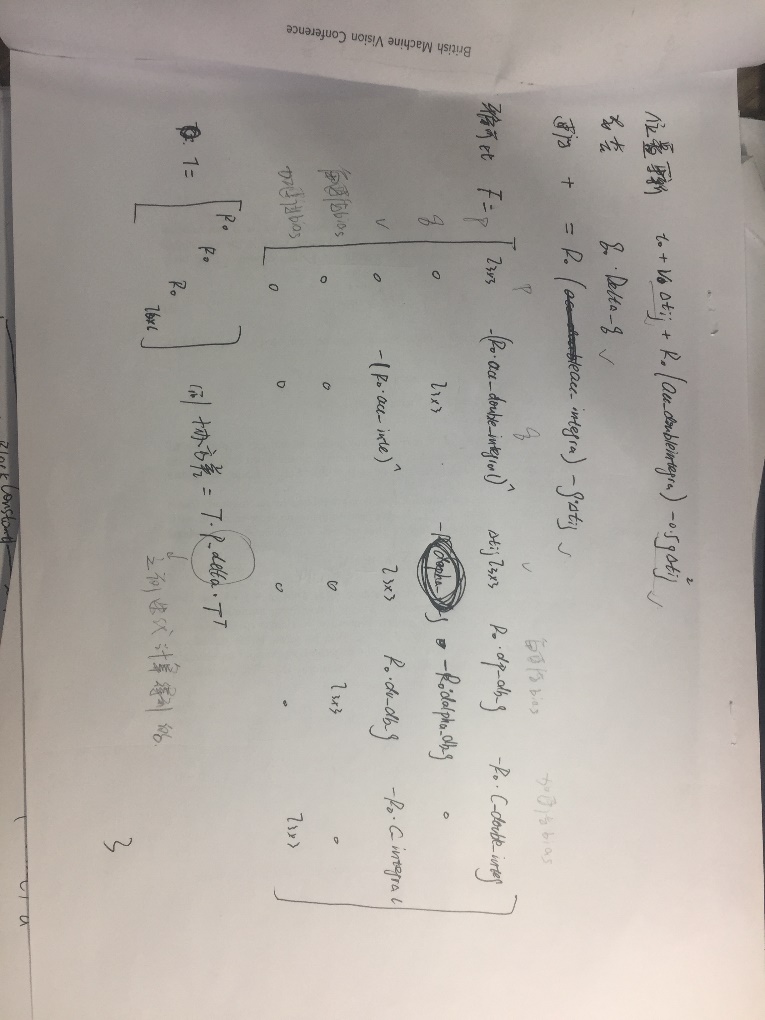
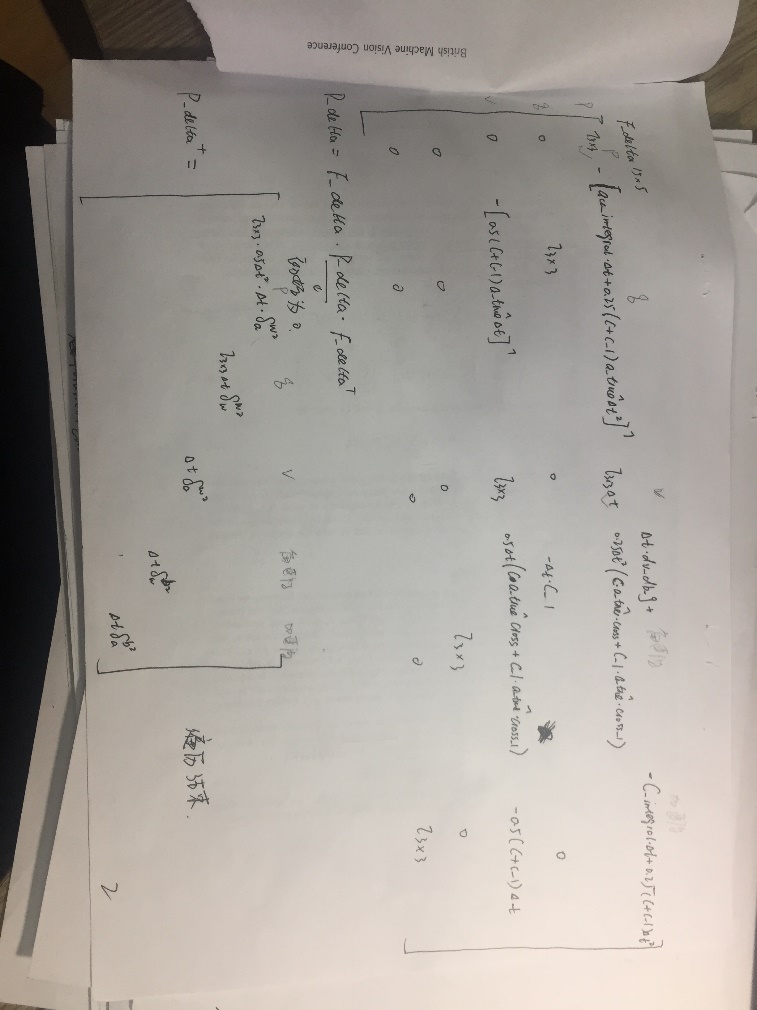
注意这里的乘法是向量的点乘，d是这个特征点对应的方差(=邻域直径)。

## 5. ImuError::propagation

输入的变量依次为 imu测量值，imu参数，上一时刻相机位姿T\_WS(同时也是输出值)，上一时刻的速度和bias，上一帧图像的时间戳，当前帧图像的时间戳。

协方差和雅克比是输出值，T\_WS和speedAndBiases也是输出参数 表示是这一时刻的机器人位姿。





## 6、ceres中的残差函数

### 6.1 ceres::PoseError

#### a.状态

状态=t+四元数，维度等于7

#### b.残差



#### c.雅克比





#### d.权重矩阵

第一帧的权重矩阵= 

如果在applyMarinalizationStrategy被调用则权重矩阵等于

### 6.2 ceres::SpeedAndBiasError

#### a.状态

状态 = 速度+角速度和加速度的bias

#### b.残差



#### c.雅克比

因为速度、加速度bias和角速度bias都是欧式空间的因此雅克比计算较简单：





#### d.权重矩阵

速度的信息矩阵=单位矩阵

角速度的信息矩阵=，是参数输入的角速度的白噪声标准差。

加速度的信息矩阵=，是参数输入的加速度的白噪声标准差。

### 6.3 ceres::ImuError

#### a.状态

状态=上一时刻的姿态+上一时刻的速度和bias+这一时刻的姿态+这一时刻的速度和bias。维度等于32.

#### b.残差









这里一定要注意如果需要重积分则，如果不需要重积分则。注意这里的和是imu参考bias，只在redoPreintegration函数中被更新。

#### c.雅克比



其中









其实代码中是先计算计算得到F0和F1然后再根据两个矩阵再计算得到J0 J1 J2 J3，为了方便对照代码中的结果，这里也给出F0 F1矩阵





#### d.权重矩阵

只在redoPreintegration函数中被更新了，信息矩阵 =P\_delta-1(此参数的定义详见5节),而redoPreintegration函数只在计算imuError的雅克比时才被调用过。当上一时刻的速度bias-速度参考bias的模乘以两个状态的相对时间超过0.0001，则认为需要重积分。而参考bias的值只在 redoPreintegration函数中被更新。

### 6.4 ceres::ReprojectionError

#### a.状态

世界坐标系到imu坐标系的变换(7个维度)

地图点在世界坐标系下的点(因为是齐次点坐标，所以维度等于4)

imu坐标系到相机坐标系的变换=通常认为是常量不进行优化(7个维度)

#### b.残差

\*(像素的测量值-投影值)

#### c.雅克比



其中：







Jh是投影得到的雅克比函数大小是2\*4。

注: 作者这里使用了一个小技巧，如果这个地图点距离相机小于20cm则将jacobians矩阵设置为0，也就是说不参加优化。









#### d.权重矩阵

，这里的s是特征点的size大小。

### 6.5 MarginalizationError

**b.残差：**

，详见9.3

**c.雅克比：**

没有loss函数

## 7、参数化方法

### 7.1 PoseLocalParameterization- Pose6d

#### a、Plus函数

t\_delta = t + △t；q\_delta=△q\*q； (注意是左乘更新)

#### b、Minus函数

△t=t\_delta-t；四元数的变化量=2\*( q\_delta\*q-1)x,yz，右下标表示只取四元数的虚部

#### c、ComputeJacobian

q是输入的姿态



#### d、liftJacobian

输入的参数是q



#### e、globalsize=7,localsize=6.

注：speedbias状态因为是在欧式空间的因此没有参数化方法。也就是对应代码中的trivial参数化方法。

### 7.2 HomogeneousPointLocalParameterization

#### a、Plus函数

,此处的p是齐次坐标

#### b、Minus函数

(delta\_p-p)1:3，这里的坐标都是齐次坐标，右下角的1:3表示只取前三维。

#### c、ComputeJacobian



#### d、liftJacobian



#### e、globalsize=4,localsize=3.

## 8、okvis中的state说明



## 9、applyMarginalizationStrategy

allLinearizedFrames=removeAllButPose集合=除了imu窗口帧(最新的连续的三个帧)之外的所有帧。

removeFrames集合=除去imu窗口帧和最近的五个关键帧剩下的所有帧。

经过我们测试发现 removeFrames的数量一直维持在1个，而removeAllButPose数量一直维持在6个。

a.如果imu窗口中载具姿态数量小于三则直接返回

b.从ceres中删除之前的边缘化误差残差块，并更新相关变量

c.获取要被剔除的状态帧

d.首先遍历removeAllButPose(除了imu窗口帧之外的所有帧)：

得到某帧的速度和bias的参数块序号(其实就是这个帧的速度和bias的id)；

找到与这个参数块相关的残差块；

遍历这些残差块，将这些残差块通过addResidualBlock函数加入到marginalizationErrorPtr\_结构中。

e.遍历removeFrames(除去imu窗口帧和最近的五个关键帧剩下的所有帧=普通帧)：

得到某帧的位姿的参数块序号(其实就是这个帧的global存储的序号)；

找到与这个参数块相关的残差块；

5.1遍历这些残差块：

如果这个残差块是poseerror残差块，则将这个残差块从ceres中直接删除，continue；

如果这个残差块不是重投影残差块(是imu误差的残差块)，则将这些残差块通过addResidualBlock函数加入到marginalizationErrorPtr\_结构中。

5.2遍历landmarksMap\_结构中的地图点：



a.所有的帧都没有看到这个地图点则从ceres中删除这个地图点对应的所有残差块，删除这个地图点的参数块；



b.如果普通帧没有看到则，直接跳过这个地图点不执行下面的条件判断了；



c.如果普通帧观测到了这个地图点并且imu窗口也观测到了这个地图点则需要从ceres中删除这个地图点对应的残差块。这个残差块指的是普通帧与地图点构成的残差块。



d.如果关键帧和普通帧观测到这个地图点数量小于2(表示只有普通帧观测到了这个地图点)个且imu窗口没有看到这地图点则需要从ceres中删除这个残差块。这里的残差块指的是地图点与普通帧构建的残差块。



e.如果关键帧和普通帧观测到这个地图点数量大于2个(表示关键帧观测到了这个地图点)且imu窗口没有看到这地图点则需要addResidualBlock将这个地图点的残差块添加到marginalizationErrorPtr\_中，这里的残差块指的是地图点与关键帧或者普通帧构建的残差块。

f.marginalizeOut-见9.2

g.updateErrorComputation-见9.3

h.向ceres中添加边缘化误差函数：得到未被边缘化的数据块，向ceres添加边缘化误差函数(就只执行了一次)。边缘化误差函数详见6.5

i. 获取imu窗口的距离当前帧最远的帧，向地图添加姿态误差函数，并且这个姿态误差函数的信息矩阵设置为。

### 9.1 MarginalizationError::addResidualBlock

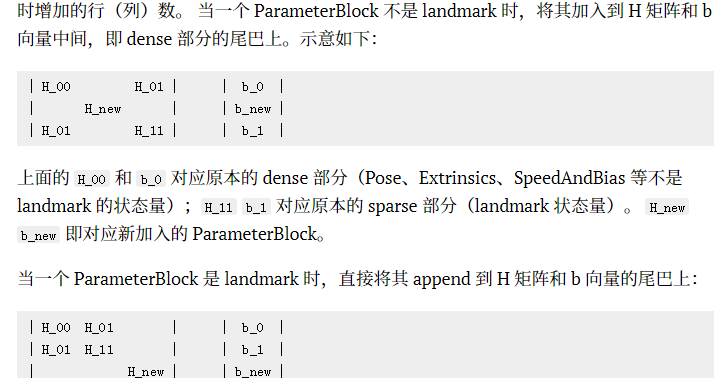
这个函数的作用是更新了边缘化的H矩阵并且从ceres中删除要被边缘化的残差块。

输入的参数是残差块：

1. 根据输入的残差块得到这个残差块对应的参数块
2. 遍历参数块：

如果parameterBlockId2parameterBlockInfoIdx\_结构中不存在这个参数块：

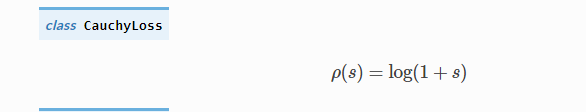
更新H矩阵和b向量(注意这里的更新只是为H矩阵和b向量腾出来空间但是还没有向里面赋值，也就是说现在的H\_new和b\_new还都是0)，并修改ceres代价函数的参数块的大小。更新方法如下：

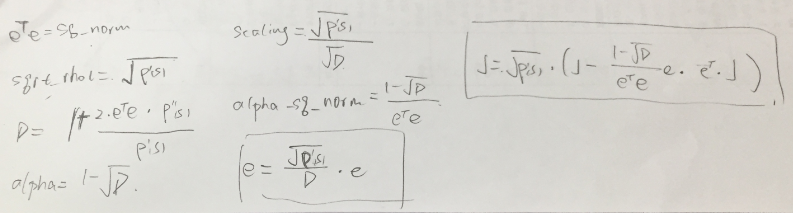


c．设置ceres代价函数的残差的维度

d．计算得到这个残差对于每个参数块的雅克比矩阵和残差

e．如果这个残差块使用了loss function那么更新雅克比和残差。经过我们验证okvis中只有BA残差中设置了loss function，使用的是cauchy函数，参数设置为1。



f．根据新得到的雅克比矩阵和残差更新H矩阵和b向量：

遍历参数块：

首先更新参数自己在H矩阵中的内容，然后更新和其他参数块相耦合的内容,

(为什么这么更新详见深蓝学院课程)

g. 从ceres中删除这个残差块。

### 9.2 marginalizeOut

输入的参数是要边缘化的参数块的id：

主要是完成了shur补的计算。设置ceres中代价函数残差的维度=原来残差维度-边缘化残差维度。删除ceres代价函 数的参数块。从ceres中删除与这个参数块相关的残差块，从ceres中删除这个参数块。

我们知道，如果J中的值特别小，那么会造成H矩阵不可逆。这里作者用了一个小技巧：让J矩阵右乘一个对角尺度矩阵Pinv，则公式变成，因为Pinv是对角矩阵则PinvT=Pinv,则上述公式变为。那么这个尺度矩阵是如何选取的呢？提取原始H矩阵的对角线上元素,如果元素大于1.0e-9,选择对角线上元素的平方根构建P向量,否则这个值赋值为10^-3，对P向量构造的对角矩阵求逆就是Pinv矩阵。

则我们在进行shur计算时实际上使用的H矩阵是, b向量是。我们另, ，schur补公式：





当我们计算完成schur后，需要去掉尺度的影响:





这里的P\_a矩阵是由P向量中与非边缘化状态相关的元素构成的对角矩阵。

### 9.3 updateErrorComputation

0.设置ceres代价函数的残差维度

a．当我们执行完schur后，我们得到了如下的方程：



b．我们仔细分析一下如上的方程：

等式左边：U是雅克比矩阵，根据一致性原则是不变量。W和V也都是雅克比矩阵因此在迭代的过程中也不发生改变。

等式右边：b\_a和b\_b与残差相关因此会发生改变，W和V都是雅克比因此不发生改变。

c．整体的迭代的流程是这样的：通过上述方程计算得到状态的变化量，用其更新状态。再将新的状态带入到残差中计算得到新的b\_a和b\_b。然后再将新的b\_a和b\_b带入上述方程进行下次迭代。

d．可以看到整体的迭代流程并不复杂，但是这种方式无法加入到ceres的代码中实现。主要原因是ceres需要用户定义每个残差块的残差计算方法。很显然schur的结果只给出了整体的残差计算方法但是并不符合JTf的形式。那么如何解决这个问题呢？

e．okvis中作者使用了对残差线性化的方法：



则,带入到schur补的结果中，可得：



我们另，，这两个变量都是常亮不会发生变化。上式变为：



上式右边进行化简：，此处。我们可以得到残差的定义为。

f．通过上述的线性化我们可以得到：



我们可以得到每一项的残差方程而且只和每一项的状态有关。仔细分析上述的等式，等式左右两边只有发生了变化，而其他项在边缘化时就已经是不变值了。

g．同样作者这里也使用了尺度化的小技巧来保证计算结果的稳定。H\*矩阵是。我们已知H\*矩阵，需要计算得到J\*矩阵。H\*=USUT，其中S矩阵是对角阵，元素是H\*矩阵的特征值。U矩阵由H矩阵的特征向量构成。对H\*矩阵进行分解：

H\*=USUT==J\*TJ\*

则J\*=()T, .

去掉尺度化的结果：J\*=()T，。

## 10. runRansac2d2d更新世界坐标系下的位姿

前提是我们已经知道了A,B帧所在的载体在世界坐标系下的坐标，并且我们知道了载体坐标系到相机坐标系的变换。Okvis中载体坐标系就是imu坐标系。

如果发生了位移：

首先根据ransac的结果得到相机A坐标系到相机B坐标系的位姿变换Transac。

然后根据上述的已知条件计算得到相机A坐标系到相机B坐标系的位姿变换TC1C2。

使用ransac得到的位移去更新TC1C2中的位移，tC1C2=transac\*max(0,transacT\*tC1C2)。

如果仅仅发生了旋转：

则RC1C2=Rransac

使用计算得到的TC1C2更新B帧相机在世界坐标系下的坐标。