VINS-Mono 代码解读

2017年09月28日 20:58:36 Rain-XIA 阅读数：10170

VINS-Mono SLAM源码解读

标签： SLAM VIO IMU

系统启动命令

```
$ roslaunch vins_estimator euroc.launch
$ roslaunch vins_estimator vins_rviz.launch
$ rosbag play YOUR_PATH_TO_DATASET/MH_03_medium.bag
$ roslaunch benchmark_publisher publish.launch  sequence_name:=MH_03_medium
```

文件目录概述

- ar_demo
- benchmark_publisher 发布数据集中参考值
- config 硬件、系统配置文件
- camera_model
  - calib
  - camera_models 各类相机模型
  - chessboard 用于检测棋盘格特征点
  - gpl
  - sparse_graph
  - intrinsic_calib.cc 相机矫正模块主函数
- feature_trackers
  - feature_tracker_node(main()函数,ROS接受图像的回调函数)
  - feature_tracker.c 特征点跟踪的具体实现
- support_files
- vins_estimator
  - src
    - factor 实现IMU、camera等残差模型，涉及了ceres优化库，Jacobian矩阵。
    - initial 系统初始化，外参标定，SFM
    - loop_closure 闭环检测，这里主要是利用DBOW2作者的一个demo程序
    - utility 相机显示，四元数等数据转换
    - estimator_node(main()函数)
      - 多线程 measurement_process、loop_detection、 pose_graph
    - feature_manager.cpp 特征点管理，三角化，关键帧操作
    - parameters.cpp 外部系统设置参数输入

重要变量说明

Ps：世界坐标系下机体的平移量$ \mathbf p_{w}^{b}, Rs：世界坐标系下机体的旋转量 \mathbf q_{w}^{b}, Vs：世界坐标系机体的速度量 \mathbf v_{w}^{b} ;$para_Pose$跟Ps和Rs是等价的，只是数据类型有区别ric、tic：IMU与$camera$之间的外参 \mathbf p_{b}^{c}, \mathbf q_{b}^{c}$
estimator.f_manager.feature: 全局特征点
备注:这里的符号跟VINS原作者是一致的,即T_{from}^{to}.

TF关系

\world -> \body -> \camera
\body坐标系为IMU坐标系

系统结构

![Screenshot from 2017-10-17 10:40:41.png-111.2kB][1]

特征点跟踪过程

feature_tracker_node.cpp main()

- readParameters() 通过ROS来读取参数
- FeatureTracker::readIntrinsicParameter() 读取相机内参
- 读取mask图片（鱼眼相机）
- img_callback() ROS回调函数
  - FeatureTracker::readImage()
    - cv::createCLAHE() 直方图均衡化（可选）
    - cv::calcOpticalFlowPyrLK() LK金字塔光流法(同时会去除那些无法追踪到的特征点)

- FeatureTracker::rejectWithF() 通过F矩阵去除外点
- FeatureTracker::setMask() 设置遮挡部分（鱼眼相机）
  - 对检测到的特征点按追踪到的次数排序
  - 在mask图像中将追踪到点的地方设置为0，否则为255，目的是为了下面做特征点检测的时候可以选择没有特征点的区域进行检测。在同一区域内，追踪到次数最多的点会被留，其他的点会被删除
- cv::goodFeaturesToTrack() 添加角点(第一帧初始化特征点检测也是通过这里完成的)
- FeatureTracker::addPoints() 添加新检测到的特征点
- FeatureTracker::undistortedPoints() 将所有特征点转换到一个归一化平面并且进行畸变
- 发送图像特征点

estimator_node

- readParameters() 通过ROS来读取参数
- Estimator::setParameter()
- registerPub() 用于RVIZ显示的Topic (visualization.cpp)
- 订阅IMU、特征点和camera原始图像Topic消息
  - imu_callback()
    - 将新得到的IMU数据放入到imu_buf中
    - pubLatestOdometry() 将最新的里程计数据发送的Rviz中显示tmp_P、tmp_Q和tmp_V
  - feature_callback() 将最新的特征点数据放入缓冲区feature_buf
  - raw_image_callback() 如果进行闭环检测的话，才会有后续对原始图像处理，将原始图像放入到缓冲区中image_buf
- 多线程处理 process()、loop_detection()、 pose_graph()
- 闭环处理（可选）

process线程

- getMeasurements() 获得IMU测量数据与camera特征点对齐数据队列
- send_imu()
  - Estimator::processIMU() IMU预积分
    - IntegrationBase::push_back()每一帧都对应一个预积分
      - IntegrationBase::propagate()
        - IntegrationBase::midPointIntegration()计算状态转移矩阵F，雅克比矩阵 jacobian = (F+I)*jacobian，和协方差矩阵
  - 系统位置Ps，速度Vs，旋转Rs通过IMU测量值进行更新
- Estimator::processImage() 处理图像
  - FeatureManager::addFeatureCheckParallax() 通过检测两帧之间的视差决定是否作为关键帧，同时添加之前检测到的特征点到feature（list< FeaturePerId >）这个容器中，计算每一个点跟踪的次数，以及它的视差
    - FeatureManager::compensatedParallax2() 对相近的特征点进行视差计算
  - 是否初始化camera与IMU之间的外参
    - FeatureManager::getCorresponding() 得到两帧之间特征点关系
    - InitialEXRotation::CalibrationExRotation() 得到camera与IMU之间的旋转偏移常量
    - Estimator::initialStructure() 视觉结构初始化
      - 通过重力协方差检测IMU的可观测性
      - Estimator::relativePose() 在滑动窗口中选择两帧中有足够多特征点和视差的帧l，利用五点法恢复相对旋转和平移量
        - MotionEstimator::solveRelativeRT() 利用五点法求解相机初始位姿
          - cv::findFundamentalMat() 利用opencv函数求解F矩阵
          - cv::recoverPose() 利用opencv函数分解F矩阵，得到相机旋转量、位移量
      - GlobalSFM::construct() 全局SFM初始化全部初始帧中的相机位置和特征点空间3D位置（l是F矩阵初始化得到的，下面做的SFM，就是将l帧作为系统初始化原点，通过PnP和三角化得到滑动窗口中l帧之后和l帧之前的相机位置和3D特征点）
        - 1). 三角化l帧与frame_num-1帧 GlobalSFM::triangulatePoint()
        - 2). 对l+1, l+2, l+3, ... frame_num-2 帧与sfm_f特征点队列进行PnP求解GlobalSFM::solveFrameByPnP()，得到这些帧的相机在空间的位置，三角化 l, l+1， l+2, ... frame_num-2 帧与frame_num-1帧，得到这些特征点在空间的3D位置
        - 3). 三角化l+1, l+2 ... frame_num-2帧与l帧
        - 4). 对l-1, l-2, l-3, ..., 0帧与sfm_f特征点队列进行PnP求解，得到这些帧的相机在空间的位置,三角化l-1, l-2, l-3, ..., 0帧与l帧
        - 5). 三角化其他所有的点
        - 6). SFM全局BA优化
      - 将相机坐标系转换到IMU坐标系中，然后再一次进行PnP求解，3D特征点还是使用之前SFM中求解出来的，后续也没有进行优化
      - Estimator::visualInitialAlign() cameara与IMU对齐
        - VisualIMUAlignment() 对齐camera与IMU (initial_aligmentc.cpp)
          - VisualIMUAlignment() IMU陀螺仪零偏初始化；主要是通过滑动窗口中，每两帧之间通过SFM求解出来的旋转与IMU预积分的旋转量组成一个最小二乘法形式的等式，求解出来陀螺仪的零偏。
          - LinearAlignment() 速度，重力向量，尺度初始化；由于在视觉初始化SFM的过程中，将其中位姿变化较大的两帧中使用E矩阵求解出来旋转和位移，后续的PnP和都是在这个尺度下完成的。所以当前的尺度与IMU测量出来的真实世界尺度肯定不是一致的，所以需要这里进行对齐。这里对齐的方法主要是通过在滑动窗口中每帧间的位置和速度与IMU预积分出来的位置和速度组成一个最小二乘法的形式，然后求解出来
            - RefineGravity() 进一步细化重力加速度，提高估计值的精度，形式与LinearAlignment()是一致的，只是将g改为$g \cdot \mathbf{\bar{ \hat{g}}} + w_1 \mb_1 + w_2 \mathbf b_2 $
        - FeatureManager::triangulate() 三角化

- 更新相机速度，位置和旋转量(通过精确求解的尺度，重力向量)
- Estimator::solveOdometry()
  - FeatureManager::triangulate() 三角化
  - Estimator::optimization()
    - 添加ceres的参数块，在滑动窗口中机体在世界坐标系的坐标，速度，旋转，IMU的陀螺仪零偏，加速度零偏。
    - Estimator::vector2double() 系统数据类型转换，由Ps,Rs转换为para_Pose，Vs,Bas,Bgs转换为para_SpeedBias，tic,ric转换为para_Ex_Pose。
    - 添加先验残差，IMU测量残差，camera测量残差。注意这里IMU项和camera项之间是有一个系数，这个系数就是他们各自的协方差矩阵：IMU的协方差是预积分的协方差(IMUFactor::Evaluate，中添加IMU协方差，求解jacobian矩阵)，而camera的测量残差则是一个固定的系数（$f/1.5$）
    - 如果进行闭环优化的话，还要添加闭环检测优化残差，计算滑动窗口中与每一个闭环关键帧的相对位姿，这个相对位置是为后面的图优化准备
    - 边缘化处理
      - 如果第二最新帧是关键帧
        - 向ResidualBlockInfo容器中(factors)添加先验残差，最新IMU测量残差，camera所有特征点测量残差
        - MarginalizationInfo::preMarginalize()
          - 根据各个测量模型Evaluate() 计算残差；各个参数块拷贝到统一的内存（parameter_block_data）中
        - MarginalizationInfo::marginalize()
          - 利用多线程构造稀疏矩阵H
          - 利用舒尔补消元简化稀疏矩阵求解过程
          - 滑动窗口中参数块存储地址调整
      - 如果第二最新帧不是关键帧的话，则把这帧的视觉测量舍弃掉（边缘化）而保留IMU测量值在滑动窗口中。（其他步骤和上一步骤相同）
    - Estimator::double2vector() 系统数据类型转换，与vector2double()过程相反。通过闭环检测得到最新一个关键帧之前跟踪得到的位姿与闭环检测得到的位姿得到矫正量
    - Estimator::failureDetection() 检测SLAM系统是否失败
- Estimator::slideWindow() 滑动窗口all_image_frame，维持滑动窗口的大小，保证SLAM运行计算的复杂度。如果第二最新帧是关键帧的话，那么这个关键帧就会留在滑动窗口中，时间最长的一帧和其测量值就会被边缘化掉如果第二最新帧不是关键帧的话，则把这帧的视觉测量舍弃掉而保留IMU测量值在滑动窗口中这样的策略会保证系统的稀疏性。(0, 1, ..., N)关键帧，0是时间最长的关键帧，N是最新关键帧。
  - Estimator::slideWindowOld() marginalization_flag == MARGIN_OLD
    - FeatureManager::removeBack() 将特征点进行处理，将每一个特征点开始出现的帧号减一，如果这个关键点的帧号为0，则直接删除
  - Estimator::slideWindowNew() marginalization_flag == MARGIN_SECOND_NEW
    - FeatureManager::removeFront()

- 是否进行闭环(可系统设置)
  - 清除estimator.retrive_data_vector前一个闭环，添加最新retrive feature（retrive_data_buf）
  - 如果经过前面检测，最新第二帧是关键帧，则将这帧添加到关键帧队列中
  - KeyFrame::setExtrinsic() 设置IMU与camera的外参
  - KeyFrame::buildKeyFrameFeatures() 将空间的3D点构建当前关键帧的特征点
  - 添加到关键帧队列中（keyframe_buf）
  - 检查闭环是否出错
    - 两个匹配帧之间yaw角度过大或者是平移量过大，则认为是匹配错误，移除此次闭环匹配
    - 将此次闭环检测添加到位图优化缓冲区optimize_posegraph_buf
- 给RVIZ发送里程计信息、关键位置、相机位置、点云和TF关系
- update() 更新IMU系统参数
  - predict() 通过IMU的测量值进行tmp_Q，tmp_P，tmp_V预测更新

loop_detection线程

- process_loop_detection()
  - LoopClosure::initCameraModel() 初始化相机模型
  - KeyFrame::extractBrief() 在前端视觉追踪过程中提取的特征点对于闭环检测是不够的，所以需要提取更加多的特征点以及相应的描述子
  - LoopClosure::startLoopClosure() 开始闭环检测
    - demoDetector<TVocabulary, TDetector, TDescriptor>::run()
    - TemplatedLoopDetector<TDescriptor, F>::detectLoop()
  - 是否闭环成功
    - KeyFrame::findConnectionWithOldFrame() 利用描述子匹配关键帧库中与当前关键帧的特征点，并且用PnP算法求解这两帧之间的关系
      - KeyFrame::searchByDes() 关键帧库中匹配上的关键帧特征点与当前帧上的特征点进行匹配
        - KeyFrame::searchInAera() 通过一个描述子对一个描述子向量进行匹配，得到最高得分的特征点
    - KeyFrame::FundmantalMatrixRANSAC() 利用opencv中的findFundamentalMat函数和RANSAC方法进行基础矩阵求解，排除外点
    - KeyFrame::PnPRANSAC() 利用RANSAC算法求解PnP问题，得到当前关键帧在世界坐标系下的位姿，并且删除外点
    - 向retrived feature缓冲区（retrive_data_buf）中添加新的数据
    - KeyFrameDatabase::addLoop() 向关键帧数据库中添加闭环序号,更新闭环显示
      - CameraPoseVisualization::add_loopedge() 更新关键帧库中闭环显示
    - PoseGraph显示更新
  - 如果关键帧库中帧数过大，则减低采样，删除那些位置和角度关键帧密集的关键帧，保留位置和角度有一定间隔的关键帧

pose_graph线程

- process_pose_graph

- KeyFrameDatabase::optimize4DoFLoopPoseGraph()
  - FourDOFError结构体 遍历所有关键帧，添加该关键帧与之前五帧的残差
  - FourDOFWeightError结构体 如果是闭环关键帧，添加闭环残差
  - ceres进行图优化
  - 将优化的结果用于更新所有关键帧位置
  - 遍历所有关键帧，利用当前优化和优化前的Yaw轴的值，进行所有关键帧的Yaw矫正

非线性优化

非线性优化是VINS中非常创新的一部分，也是整个代码中最为复杂的部分。
$$\min\limits_{ \mathcal{X}, \mathbf q_{v}^{w}, \mathbf p_{v}^{w}}{ | \mathbf r_p - \mathbf H_p \mathcal{X} |^{2}

- \sum_{k \in \mathcal{B}} | r_{\mathcal{B}} (\mathbf {\hat z}*{b}{k+1}}^{b_{k}}, \mathcal{X}) |*{\|\mathbf p*{b_{k+1}}^{b_{k}}}{2} + \
  \sum_{(l,j) \in \mathcal{C}} | r_{\mathcal{C}} (\mathbf {\hat z}*{l}^{c}{j}}, \mathcal{X}) |*{\|\mathbf p*{l}^{c_{j}}}{2} +
  \sum_{l \in \mathcal{L}} | r_{\mathcal{C}} (\mathbf {\hat z}*{l}^{c}{j}}, \mathcal{X}, \mathbf q_{v}^{w},\mathbf p_{v}^{w}) |*{\|\mathbf p*{l}^{c_{v}}}{2}
  } \tag{3}

  \mathcal{X} = \begin{bmatrix} \mathbf x_0,\mathbf x_1, \dots, \mathbf x_n, \mathbf x_c^b, \lambda *0, \|lambda 1, \|dots, \|lambda m
  \|end{bmatrix} \
  \|mathbf x_k = \|begin{bmatrix} \|mathbf p*{b{k}}^{w}, \|mathbf v*{b_{k}}^{w}, \mathbf q_{b_{k}}^{w}, \mathbf b_{a}, \mathbf b_{g}
  \end{bmatrix}, k \in [0,n] \
  \mathbf x_{c}^{b} =

$$
\begin{bmatrix} \mathbf{p}_c^b & \mathbf{q}_c^b \end{bmatrix}
$$

  \tag{4}
  $$
  在全局非线性优化中，第一项为先验信息优化，第二项为IMU测量残差优化，第三项为camera测量残差优化，第四项为全局闭环残差优化

- 符号说明
  $\mathcal{X}$为滑动窗口中的所有状态
  $\{\mathbf{r}_p, \mathbf{H}_p\}$是通过边缘化得到先验信息
  $\mathcal{C}$为在当前滑动窗口中最少被观察到两次的特征点集
  $\mathcal{B}$为所有的IMU测量数据集
  $\mathcal{L}$为所有的retrieved feature特征点集，通过retrieved feature特征点可以连接关键帧库中闭环关键帧和滑动窗口中新的关键帧。$\mathbf{q}_w^v, \mathbf{p}_w^v$，其中下标$v$表示的是关键帧库中闭环关键帧，$m$表示的是当前关键帧，整体表示在闭环检测中通过关键帧库中闭环关键帧求解出来世界坐标系下的旋转量和位移量。

Vision Model

$$
\begin{eqnarray*}
&& r_{\mathcal{C}} (\mathbf {\hat z}*{l}^{c}{j}},\mathcal{X})
=

$$
\begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}
$$

^{T} \cdot
(\mathcal{\bar P_{l}^{c_j}} - \frac{P_{l}^{c_j}}{|P_{l}^{c_j}|}) \
&& \mathcal{\bar P_{l}^{c_j}} = \pi *{c}^{-1}*
(\begin{bmatrix}
\|hat u*{l}^{c_j} \
\hat v_{l}^{c_j}
\end{bmatrix}) \
&& \mathcal P_{l}^{c_j} = \mathbf q_{b}^{c} (\mathbf q_{w}^{b_j}(\mathbf q_{b_i}^{w}(\mathbf q_{c}^{b} \frac{1}{\lambda_l} \pi *{c}^{-1}*
(\begin{bmatrix}
\|hat u*{l}^{c_i} \
\hat v_{l}^{c_i}
\end{bmatrix})

- \mathbf p_{c}^{b}) + \mathbf p_{b_i}^{w} - \mathbf p_{b_j}^{w})

- \mathbf p_{b}^{c})
  \end{eqnarray*} \tag{1}
  $$
  空间上的一个3D特征点会被camera多次观测到，j^{th}图像帧的camera的残差值被定义为，考虑l^{th}特征点第一次在i^{th}图像帧被观察到恢复的深度信息，投影到第j图像帧的重投影误差。

- 符号说明
  $\mathcal{P}_l^{c_j}$是l^{th}图像特征点的射线矢量被j^{th}图像帧观察到,$\mathcal{P}_l^{c_j}$是在l^{th}帧中变换的射线矢量。
  $\mathcal{P}_l^{c_j}$的实际含义：将第i帧图像中观测到的2D特征点转换为3D特征点，通过camera->IMU坐标系变换转换为body坐标系下的坐标，通过body->world坐标系变换转换为世界坐标系下的坐标。接着通过第j帧图像的body->world坐标系变换矩阵转换到body坐标系下的坐标，最后通过IMU->camera外参转换为camera坐标系下的3D坐标。

- 注意 这里的相机模型为MEI相机模型主要是针对大视角的鱼眼相机，将2D特征点反投影到单位球面上。

```
problem.AddResidualBlock(f, loss_function, para_Pose[imu_i], para_Pose[imu_j], para_Ex_Pose[0], para_Feature[feature_index]);
```

其中para_Pose[i]分别包含了系统位移和旋转；para_SpeedBias[i]分别包含了系统的速度，加速度的零偏，陀螺仪的零偏。

非线性优化求解，需要求解出camera model的Jacobian矩阵：

- 第一部分。分别对$\mathbf{p}_{b_i}^w, \mathbf{q}_{b_i}^w$求导可得到：

$$
\begin{bmatrix} \mathbf{q}_b^c \mathbf{q}_w^{b_j} & -\mathbf{q}_b^c \mathbf{q}_w^{b_j} [\mathbf{q}_{b_i}^w \begin{bmatrix} \hat{x}_l^{b_i} & \hat{y}_l^{b_i} & \hat{z}_l^{b_i} \end{bmatrix}^T]_\times & \mathbf{0} \end{bmatrix}
$$

- 第二部分。分别对$\mathbf{p}_{bj}^{w}$, $\mathbf{q}_{bj}^{w}$求导可得到:

$$\begin{bmatrix} -\mathbf{q}_b^c\mathbf{q}_w^{b_j} & \mathbf{q}_b^c\left[\hat{x}_l^{b_j} \quad \hat{y}_l^{b_j} \quad \hat{z}_l^{b_j}\right]_\times^T & \mathbf{0} \end{bmatrix}$$

- 第三部分。分别对$\mathbf{p}_b^c$, $\mathbf{q}_b^c$求导可得到:

$$\begin{bmatrix} \mathbf{q}_b^c(\mathbf{q}_w^{b_j}\mathbf{q}_{b_i}^w - \mathbf{I}) & \begin{matrix} -\mathbf{q}_b^c\mathbf{q}_w^{b_j}\mathbf{q}_{b_i}^w[\mathbf{q}_c^b\left[\hat{x}_l^{b_i} \quad \hat{y}_l^{b_i} \quad \hat{z}_l^{b_i}\right]_\times^T]_\times + \\ [\mathbf{q}_b^c\mathbf{q}_w^{b_j}\mathbf{q}_{b_i}^w\mathbf{q}_c^b\left[\hat{x}_l^{b_i} \quad \hat{y}_l^{b_i} \quad \hat{z}_l^{b_i}\right]_\times^T]_\times + \\ [\mathbf{q}_b^c(\mathbf{q}_w^{b_j}(\mathbf{q}_{b_i}^w\mathbf{p}_c^b + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b)]_\times \end{matrix} & \mathbf{0} \end{bmatrix}$$

- 第四部分，分别对$\mathbf{\lambda }$求导可得到:

$$\begin{bmatrix} \mathbf{q}_b^c\mathbf{q}_w^{b_j}\mathbf{q}_{b_i}^w\mathbf{q}_c^b[\pi_c^{-1}(\begin{bmatrix}\hat{u}_l^{c_i}\\ \hat{v}_l^{c_i}\end{bmatrix})]_\times\lambda^{-2} \end{bmatrix}$$

- 注意：对Jacobian矩阵中第三部分，Jacobian矩阵的变量是$ \mathbf q_{b}^{c}$，先对观测方程进行如下分解，然后下式进行求导，利用李代数扰动模型求导性质可以得到上式方程的结
$$
\begin{aligned}
\mathcal P_{l}^{c_j}
& = \mathbf q_{b}^{c} (\mathbf q_{w}^{b_j}(\mathbf q_{b_i}^{w}(\mathbf q_{c}^{b} \frac{1}{\lambda_l} \pi ^{c}^{-1}
(\begin{bmatrix}
\hat u_{l}^{c_i} \\
\hat v_{l}^{c_i}
\end{bmatrix})
- \mathbf p_{c}^{b}) + \mathbf p_{b_i}^{w}) - \mathbf p_{b_j}^{w}) + \mathbf p_{c}^{b})
\end{aligned}
$$

- \mathbf p_{c}^{b}) + \mathbf p_{b_i}^{w} - \mathbf p_{b_j}^{w})

- \mathbf p_{b}^{c}) \\
& =
\mathbf q_{b}^{c} \mathbf q_{w}^{b_j} \mathbf q_{b_i}^{w} \mathbf q_{c}^{b} \frac{1}{\lambda_l} \pi ^{c}^{-1}
(\begin{bmatrix}
\hat u_{l}^{c_i} \\
\hat v_{l}^{c_i}
\end{bmatrix}) +
\mathbf q_{b}^{c} (\mathbf q_{w}^{b_{j}}(\mathbf q_{b_{i}}^{w} \mathbf p_{c}^{b} + \mathbf p_{b_i}^{w} - \mathbf p_{b_j}^{w}) -\mathbf p_{c}^{b})
\end{aligned} \tag{3}
$$

IMU

预积分

- 将频率较高的多个IMU测量值转换为单个测量项。新得到测量项可以非线性迭代中重新进行线性化
- 可以不用知道全局旋转的情况下进行坐标变化

$$\begin{aligned}
\mathbf{p}_{b_{k+1}}^{b_0} &= \mathbf{p}_{b_k}^{b_0} + \mathbf{R}_{b_k}^{b_0}\mathbf{v}_{b_k}^{b_k} - \mathbf{g}^{b_0}\Delta t^2/2 + \mathbf{R}_{b_k}^{b_0}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\
\mathbf{v}_{b_{k+1}}^{b_{k+1}} &= \mathbf{R}_{b_k}^{b_{k+1}}\mathbf{v}_{b_k}^{b_k} - \mathbf{g}^{b_0}\Delta t + \mathbf{R}_{b_k}^{b_{k+1}}\boldsymbol{\beta}_{b_{k+1}}^{b_k} \\
\mathbf{R}_{b_{k+1}}^{b_0} &= \mathbf{R}_{b_k}^{b_0}\mathbf{R}_{b_{k+1}}^{b_k}
\end{aligned}$$

IMU measurement model
Technical Report VINS_Mono 公式(17)

$$\begin{aligned}
\hat{\alpha}_{b_{k+1}}^{b_k} &= \mathbf{q}_{c0}^{b_k}(s(\bar{\mathbf{p}}_{b_{k+1}}^{c0} - \bar{\mathbf{p}}_{b_k}^{c0}) + \frac{1}{2}\mathbf{g}^{c0}\Delta t_k^2 - \mathbf{v}_{b_{k+1}}^{c0}\Delta t_k) \\
\hat{\beta}_{b_{k+1}}^{b_k} &= \mathbf{q}_{c0}^{b_k}(\mathbf{v}_{b_{k+1}}^{c0} + \mathbf{g}^{c0}\Delta t_k - \mathbf{v}_{b_k}^{c0})
\end{aligned}$$

$$\begin{aligned}
r_{\mathcal{B}}(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}_{b_{k+1}}^{b} \\ \delta\boldsymbol{\beta}_{b_{k+1}}^{b} \\ \delta\boldsymbol{\theta}_{b_{k+1}}^{b} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_g \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{q}_w^{b_k}(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k) - \mathbf{q}_w^{b_k}\mathbf{v}_w^{b_k}\Delta t_k - \hat{\boldsymbol{\alpha}}_{b_{k+1}}^b \\ \mathbf{q}_w^{b_k}(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k) - \mathbf{q}_w^{b_k}\mathbf{v}_w^{b_k} - \hat{\boldsymbol{\beta}}_{b_{k+1}}^b \\ 2[\mathbf{q}_{b_{k+1}}^{w^{-1}} \otimes \mathbf{q}_{b_k}^w \otimes \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k}]_{xyz} \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k} \end{bmatrix}
\end{aligned} \tag{4}$$

在IMU measurements residual中，其中优化的是误差状态，其中前面部分是与camera融合且经过优化然后推导出来的期望值，后面部分是IMU实际测量值得到的。

采用中值数值积分方式，由牛顿定理可到的状态转移方程：
KaTeX parse error: No such environment: eqnarray* at position 8: \begin{eqnarray*} && \mathbf p_{...
然后通过上式推到误差状态转移方程，同时由误差状态状态转移方程（from ESKF）可知速度误差状态方程：

$$\delta\mathbf{v} \leftarrow \delta\mathbf{v} + (-\mathbf{R}[\mathbf{a}_w - \mathbf{a}_b]_\times\delta\boldsymbol{\theta} - \mathbf{R}\delta\mathbf{a}_b + \delta\mathbf{g})\Delta t - \mathbf{R}\mathbf{a}_n\Delta t \tag{6}$$

则位移状态误差方程：

$$\delta\mathbf{p} \leftarrow \delta\mathbf{p} + \delta\mathbf{v}\Delta t + \frac{1}{2}(-\mathbf{R}[\mathbf{a}_w - \mathbf{a}_b]_\times\delta\boldsymbol{\theta} - \mathbf{R}\delta\mathbf{a}_b)\Delta t^2 + \frac{1}{2}\mathbf{R}\mathbf{a}_n\Delta t^2$$

通过连续域上的旋转误差状态得到离散化的旋转误差状态：

$$\delta\dot{\boldsymbol{\theta}} = -[\boldsymbol{\omega}_k - \delta\boldsymbol{\omega}_{b_k}]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_{b_k} - \boldsymbol{\omega}_n$$
$$\delta\boldsymbol{\theta}_{k+1} = (\mathbf{I} - [\frac{\boldsymbol{\omega}_k + \boldsymbol{\omega}_{k+1}}{2} - \delta\boldsymbol{\omega}_{b_k}]_\times \delta t)\delta\boldsymbol{\theta}_k - \delta\boldsymbol{\omega}_{b_k}\delta t \tag{7}$$

因为VINS中代码实现的方法是中值积分方法，所以在ESKF中的位移误差状态转移中使用的欧拉法需要改为中值积分方法。

$$\delta\mathbf{p}_{k+1} = \delta\mathbf{p}_k + \delta\mathbf{v}\Delta t + \frac{1}{2}(\frac{-\mathbf{q}_k[\mathbf{a}_k - \mathbf{a}_{b_k}]_\times \delta\boldsymbol{\theta}_k}{2} + \frac{-\mathbf{q}_{k+1}[\mathbf{a}_{k+1} - \mathbf{a}_{b_{k+1}}]_\times \delta\boldsymbol{\theta}_{k+1}}{2})\delta t^2 + -\frac{1}{4}(\mathbf{q}_k + \mathbf{q}_{k+1})\delta\boldsymbol{a}_{b_k}\delta t^2 + \frac{1}{4}\mathbf{q}_k\mathbf{n}_{a_k}\delta t^2 + +\frac{1}{4}\mathbf{q}_{k+1}\mathbf{n}_{a_{k+1}}\delta t^2$$

作者在Github issue中给出的关于IMU中值积分中旋转误差是δq，但是感觉这个应该是$\delta\theta$，因为论文中也是这样表示旋转误差的，所以下面的公式符号中没有严格区分两者的区别。

$$
\delta \mathbf p_{k+1} = \delta \mathbf p_k + \delta \mathbf v_k \Delta t + \frac{1}{2}(\frac{- \mathbf q_k [\mathbf a_k - \mathbf a_{b_k}] \times \delta \mathbf \theta_k}{2} + \frac{- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_k}] \times (\mathbf I - [\omega_k - \delta \mathbf \omega_{b_k}] \times \Delta t) \delta \mathbf \theta_k - \delta \mathbf \omega_{b_k} \Delta t}{2}) \Delta t^2 + \frac{1}{4}(\mathbf q_k + \mathbf q_{k+1}) \delta \mathbf a_{b_k} \Delta t^2 + \frac{1}{4} \mathbf q_k \mathbf n_{a_k} \Delta t^2 + \frac{1}{4} \mathbf q_{k+1} \mathbf n_{a_{k+1}} \Delta t^2
$$

- $$\delta \mathbf v \Delta t - \frac{1}{4}(\mathbf q_k + \mathbf q_{k+1}) \delta \boldsymbol a_{b_k} \delta t^2 \tag{9}$$
  KaTeX parse error: Can't use function '$' in math mode at position 96: …记出来，也就是参考公式（5）中$\mathbf a_k + \…
  $$
  \begin{bmatrix}
  \delta \mathbf p_{k+1} \\
  \delta \mathbf q_{k+1} \\
  \delta \mathbf v_{k+1} \\
  \delta \boldsymbol a_{b_{k+1}} \\
  \delta \boldsymbol \omega_{b_{k+1}}
  \end{bmatrix} =
  \begin{bmatrix}
  \mathbf I & f_{01} & \delta t & -\frac{1}{4}(\mathbf q_k + \mathbf q_{k+1}) \delta t^2 & f_{05} \\
  \boldsymbol 0 & \mathbf I - [\frac{\boldsymbol \omega_{k+1} + \boldsymbol \omega_{b_k}}{2} \delta t]_\times & \boldsymbol 0 & f_{35} \\
  \boldsymbol 0 & \boldsymbol 0 & \mathbf I & \boldsymbol 0 \\
  \boldsymbol 0 & \boldsymbol 0 & \boldsymbol 0 & \mathbf I
  \end{bmatrix}
  \begin{bmatrix}
  \delta \mathbf p_k \\
  \delta \mathbf q_k \\
  \delta \mathbf v_k \\
  \delta \boldsymbol a_{b_k} \\
  \delta \boldsymbol \omega_{b_k}
  \end{bmatrix} +
  $$
- $$\begin{bmatrix} \frac{1}{4}\mathbf{q}_k\delta t^2 & v_{01} & \frac{1}{4}\mathbf{q}_{k+1}\delta t^2 & v_{03} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{1}{2}\delta t & \mathbf{0} & \frac{1}{2}\delta t & \mathbf{0} & \mathbf{0} & \frac{1}{2}\mathbf{q}_k\delta t^2 & v_{31} & \frac{1}{4}\mathbf{q}_{k+1}\delta t & v_{33} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \delta t & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \delta t \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{n}_{a_k} & \boldsymbol{n}_{w_k} & \boldsymbol{n}_{a_{k+1}} & \boldsymbol{n}_{w_{k+1}} & \boldsymbol{n}_{a_{b_k}} & \boldsymbol{n}_{w_{b_k}} \end{bmatrix}$$

= \boldsymbol F_t \delta \boldsymbol z_{t}^{b_k} + \boldsymbol G_t \boldsymbol n_t\

$$
\begin{eqnarray*}
&& f_{01} = \frac{- \mathbf q_{k} [\mathbf a_k - \mathbf a_{b_k}] \times \delta t^{2}}{4} + \frac{- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times (\mathbf I - [\frac{\boldsymbol \omega_{k+1} + \boldsymbol \omega_{b_k}}{2} \delta t]_\times) \delta t^{2}}{4} \\
&& f_{31} = \frac{- \mathbf q_{k} [\mathbf a_k - \mathbf a_{b_k}] \times \delta t}{2} + \frac{- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times (\mathbf I - [\frac{\boldsymbol \omega_{k+1} + \boldsymbol \omega_{b_k}}{2} \delta t]_\times) \delta t}{2} \\
&& f_{05} = \frac{1}{4}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t^{2})(- \delta t) \\
&& f_{35} = \frac{1}{2}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t)(- \delta t) \\
&& v_{01} = \frac{1}{4}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t^{2}) \frac{1}{2} \delta t \\
&& v_{03} = \frac{1}{4}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t^{2}) \frac{1}{2} \delta t \\
&& v_{31} = \frac{1}{2}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t) \frac{1}{2} \delta t \\
&& v_{33} = \frac{1}{2}(- \mathbf q_{k+1} [\mathbf a_{k+1} - \mathbf a_{b_{k+1}}] \times \delta t) \frac{1}{2} \delta t \\
\end{eqnarray*} \tag{10}
$$

更新IMU协方差矩阵：

$$\boldsymbol P_{t + \delta t}^{b_k} = (\boldsymbol I + \boldsymbol F_t \delta t) P_{t}^{b_k} (\boldsymbol I + \boldsymbol F_t \delta t)^T + \boldsymbol G_t \delta t \boldsymbol Q (\boldsymbol G_t \delta t)^T, t \in [k, k+1]$$

更新一阶雅克比矩阵, 利用下面的矫正预积分的近似来代替由IMU直接测量得到的预积分：

$$\boldsymbol J_{t + \delta t} = (\mathbf I + \mathbf F_t \delta t) \mathbf J_t$$

KaTeX parse error: Can't use function '$' in math mode at position 10: 通过一阶近似得到$\alpha _{b_{k+…

$$
\alpha_{b_{k+1}}^{b_k} \approx \hat \alpha_{b_{k+1}}^{b_k} + \mathbf J_{b_a}^{\alpha} \delta \mathbf b_{a_k} + \mathbf J_{b_w}^{\alpha} \delta \mathbf b_{w_k} \\
\beta_{b_{k+1}}^{b_k} \approx \hat \beta_{b_{k+1}}^{b_k} + \mathbf J_{b_a}^{\beta} \delta \mathbf b_{a_k} + \mathbf J_{b_w}^{\beta} \delta \mathbf b_{w_k} \\
\gamma_{b_{k+1}}^{b_k} \approx \hat \gamma_{b_{k+1}}^{b_k} \otimes
$$

$$\begin{bmatrix} 1 & \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\mathbf{b}_{w_k} \end{bmatrix}$$

求解雅克比矩阵

$$
r_{\mathcal{B}} (\mathbf {\hat z}_{b_{k+1}}^{b_k}, \mathcal{X}) =
\begin{bmatrix}
\delta \boldsymbol \alpha_{b_{k+1}}^{b} \\

$$
\delta \boldsymbol \beta {b{k+1}^{b} \\
\delta \boldsymbol \theta {b{k+1}^{b} \\
\delta \mathbf b {a} \|
\|delta \|mathbf b {g} \|
\|end{bmatrix} \| =
\|begin{bmatrix}
\|mathbf q{w}^{b_k}(\|mathbf p{b_{k+1}^{w} - \mathbf p_{b_{k}}^{w} + \frac{1}{2} \mathbf g^{w}\Delta t_k) - \mathbf q_{w}^{b_k} \mathbf v_{w}^{b_k} \Delta t_k -
(\hat \alpha {b{k+1}^{b_{k}} + \mathbf J_{b_a}^{\alpha} \mathbf b {ab{k}} + \mathbf J_{b_w}^{\alpha} \mathbf b {wb{k}}) \\
\mathbf q_{w}^{b_k} (\mathbf v_{b_{k+1}}^{w} + \mathbf g^{w} \Delta t_k) - \mathbf q_{w}^{b_k} \mathbf v_{w}^{b_k} -
(\hat \beta {b{k+1}^{b_{k}} + \mathbf J_{b_a}^{\beta} \mathbf b {ab{k}} + \mathbf J_{b_w}^{\beta} \mathbf b {wb{k}}) \\
2 \Bigg[\mathbf q_{b_{k+1}}^{w{-1} \otimes \mathbf q_{b_{k}}^{w} \otimes
(\hat \gamma {b{k+1}^{b_{k}} \otimes \begin{bmatrix} 1 \ \frac{1}{2} \mathbf J_{b_w}^{\gamma} \mathbf b {wb{k}} \end{bmatrix}) \Bigg]_{xyz} \\
\mathbf b {ab{k+1}} - \mathbf b {ab{k}} \\
\mathbf b {wb{k+1}} - \mathbf b {wb{k}} \\
\end{bmatrix}
$$

```
problem.AddResidualBlock(imu_factor, NULL, para_Pose[i], para_SpeedBias[i], para_Pose[j], para_SpeedBias[j]);
```

其中para_Pose[i]分别包含了系统位移和旋转；para_SpeedBias[i]分别包含了系统的速度，加速度的零偏，陀螺仪的零偏。

因为VINS中进行非线性优化中的IMU残差,进行优化的时候,优化的参数是这四个,所以可以求解出来下面四个部分Jacobian矩阵.

对上述公式求雅克比矩阵：

- 分别$\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{p}_{b_k}^{w}}$, $\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{q}_{w}^{b_k}}$, $\frac{\partial \delta \boldsymbol{\theta}_{b_{k+1}}^{b}}{\partial \mathbf{q}_{w}^{b_k}}$, $\frac{\partial \delta \boldsymbol{\beta}_{b_{k+1}}^{b}}{\partial \mathbf{q}_{w}^{b_k}}$ 求导可得Jacobian第一部分:

$$
\begin{bmatrix}

- \mathbf q_{w}^{b_k} & [\mathbf q_{w}^{b_k}(\mathbf p_{b_{k+1}}^{w} - \mathbf p_{b_{k}}^{w} + \frac{1}{2} \mathbf g^{w}\Delta t_k - \mathbf v_{w}^{b_k} \Delta t_k)] |times & \|mathbf 0 \|
\|mathbf 0 & \|begin{bmatrix} [\|mathbf q{b_{k+1}}^{w{-1}} \otimes \mathbf q_{b_{k}}^{w}]_L
\|Bigg [\hat \|gamma {b{k+1}}^{b{k}} \otimes \begin{bmatrix} 1 \ \frac{1}{2} \mathbf J_{b_w}^{\gamma} \mathbf b {wb{k}} \end{bmatrix} \Bigg]_R \|end{bmatrix} {xyz} & \|mathbf 0 \|
\|mathbf 0 & [\|mathbf q{w}^{b_k}(\|mathbf g^{w}\|Delta t_k + \|mathbf v{b_{k+1}}^{w} - \mathbf v_{b_k}^{w} \Delta t_k)]_\times & \mathbf 0
\end{bmatrix}
$$

- 分别$\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{v}_{w}^{a_{k+1}}}$, $\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{b}_{wb_k}}$, $\frac{\partial \delta \boldsymbol{\theta}_{b_{k+1}}^{b}}{\partial \mathbf{b}_{wb_k}}$, $\frac{\partial \delta \boldsymbol{\gamma}_{b_{k+1}}^{b}}{\partial \mathbf{b}_{wb_k}}$, $\frac{\partial \delta \boldsymbol{\beta}_{b_{k+1}}^{b}}{\partial \mathbf{v}_{w}^{a_{k+1}}}$, $\frac{\partial \delta \boldsymbol{\beta}_{b_{k+1}}^{b}}{\partial \mathbf{b}_{wb_k}}$, $\frac{\partial \delta \boldsymbol{\beta}_{b_{k+1}}^{b}}{\partial \mathbf{b}_{wb_k}}$, $\frac{\partial \delta \mathbf{b}_{g}}{\partial \mathbf{b}_{wa_k}}$, $\frac{\partial \delta \mathbf{b}_{g}}{\partial \mathbf{b}_{wb_k}}$求导可得Jacobian第二部分：

$$
\begin{bmatrix}

- \mathbf q_{w}^{b_k} \Delta t_k& -\mathbf J_{b_a}^{\alpha} & -\mathbf J_{b_a}^{\alpha} \
\mathbf 0 & \begin{bmatrix} \Bigg [\mathbf q_{b_{k+1}}^{w{-1}} \otimes \mathbf q_{b_{k}}^{w} \otimes
\hat \gamma {b{k+1}}^{b_{k}} \otimes \begin{bmatrix} 1 \ \frac{1}{2} \mathbf J_{b_w}^{\gamma} \mathbf b {wb{k}} \end{bmatrix} \Bigg]_L \end{bmatrix} {xyz} & \mathbf 0 \

- \mathbf q_{w}^{b_k} & -\mathbf J_{b_a}^{\beta} & -\mathbf J_{b_w}^{\beta} \
\mathbf 0 & -\mathbf I & \mathbf 0 \
\mathbf 0 & \mathbf 0 & -\mathbf I \
\end{bmatrix}
$$

- 分别是$\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{p}_{b_{k+1}}^{w}}$, $\frac{\partial \delta \boldsymbol{\theta}_{b_{k+1}}^{b}}{\partial \mathbf{q}_{b_{k+1}}^{w}}$求导可得Jacobian第三部分:

$$
\begin{bmatrix}
\mathbf{q}_{w}^{b_k} & \mathbf{0} \\
\mathbf{0} & \left[\left[(\hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\mathbf{b}_{wb_k} \end{bmatrix})^{-1} \otimes \mathbf{q}_{b_k}^{w^{-1}} \otimes \mathbf{q}_{b_{k+1}}^{w} \right]_L \right]_{xyz} & \mathbf{0} \\
\end{bmatrix}
$$

- 分别是$\frac{\partial \delta \boldsymbol{\alpha}_{b_{k+1}}^{b}}{\partial \mathbf{p}_{b_{k+1}}^{w}}$, $\frac{\partial \delta \mathbf{b}_{g}}{\partial \mathbf{b}_{wa_{k+1}}}$, $\frac{\partial \delta \mathbf{b}_{g}}{\partial \mathbf{b}_{wb_{k+1}}}$求导可得Jacobian第四部分:

$$
\begin{bmatrix}
\mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{q}_{w}^{b_k} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{I} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{I}
\end{bmatrix}
$$

边缘化

$$
\Lambda _{\mathbf p}^{\mathbf+} = \Lambda _{\mathbf p}

- \sum {k \in \mathcal{D^{-}}} \boldsymbol H{B_{k+1}}^{B_{k}}{T} \boldsymbol P_{B_{k+1}}^{B_{k}}{-1} \boldsymbol H_{B_{k+1}}^{B_{k}}
- \sum {(l,j) \in \mathcal{C^{-}}} \boldsymbol H_{l}^{B_{j}}{T} \boldsymbol P_{l}^{B_{j}}{-1} \boldsymbol H_{l}^{B_{j}} \tag{11}

在图优化中对误差函数进行优化时，可以利用舒尔消元进行降低运算复杂程度，下式是优化算法中常见的增量线性方程：

\mathbf H \Delta \mathbf x = \mathbf g

$$
\begin{bmatrix} \mathbf{B} & \mathbf{E} \\ \mathbf{E}^T & \mathbf{C} \end{bmatrix}
$$

$$[\Delta \mathbf{x}_c \ \Delta \mathbf{x}_p \ ]$$

=

$$[\boldsymbol{v} \ \boldsymbol{w} \ ]$$

\tag{12}

KaTeX parse error: Can't use function '$' in math mode at position 4: 其中B是对角块矩阵，每个对角块的...

$$[\mathbf{I} \quad -\mathbf{E}\mathbf{C}^{-1} \mathbf{0} \quad \mathbf{I} \ ]$$

$$[\mathbf{B} \quad \mathbf{E} \ \mathbf{E}^T \quad \mathbf{C} \ ]$$

$$[\Delta \mathbf{x}_c \ \Delta \mathbf{x}_p \ ]$$

=

$$[\mathbf{I} \quad -\mathbf{E}\mathbf{C}^{-1} \mathbf{0} \quad \mathbf{I} \ ]$$

$$[\boldsymbol{v} \ \boldsymbol{w} \ ]$$

\tag{13}

整理可得到下式：

$$[\mathbf{B} - \mathbf{E}\mathbf{C}^{-1}\mathbf{E}^T \quad \mathbf{0} \ \mathbf{E}^T \quad \mathbf{C} \ ]$$

$$[\Delta \mathbf{x}_c \ \Delta \mathbf{x}_p \ ]$$

=

$$[\boldsymbol{v} - \mathbf{E}\mathbf{C}^{-1}\boldsymbol{w} \ \boldsymbol{w} \ ]$$

S

$$