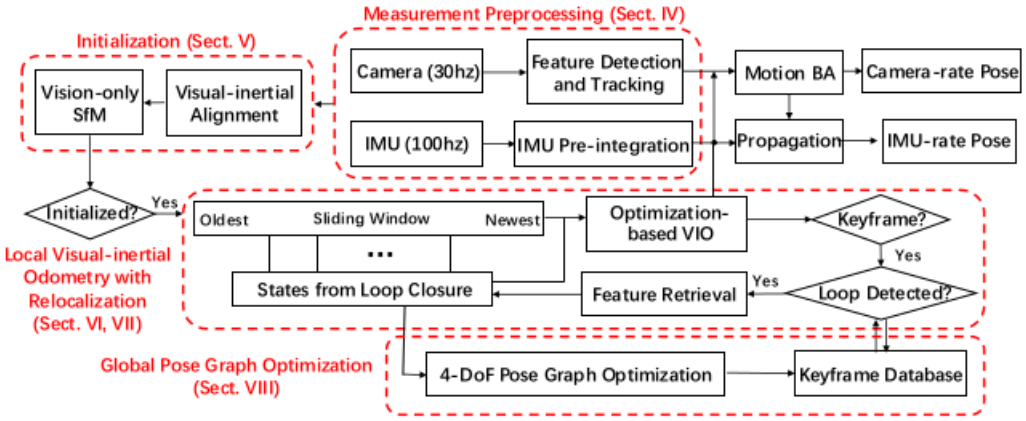
VINS-Mono 论文公式推导与代码解析

高洪臣

2019 年 3 月 16 日

目录

1 概述



Monocular visual-inertial odometry with relocalization achieved via nonlinear graph optimization-based, tightly-coupled, sliding window, visual-inertial bundle adjustment. [1]

代码（注释版）：cggos/vins_mono_cg

2 测量预处理

2.1 前端视觉处理

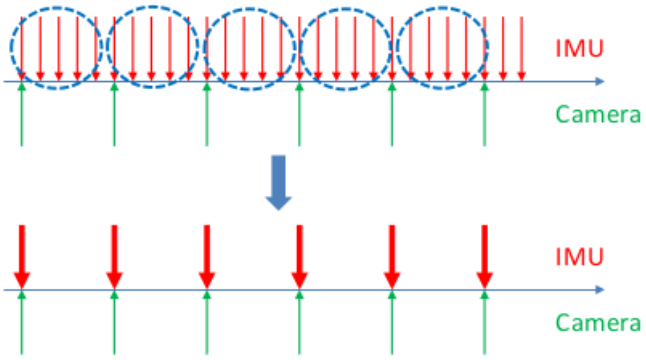
1. Simple feature processing pipeline
2. 自适应直方图均衡化（`cv::CLAHE`）
3. 掩模处理，特征点均匀分布（`setMask`）
4. 提取图像 Harris 角点（`cv::goodFeaturesToTrack`）
5. KLT 金字塔光流跟踪（`cv::calcOpticalFlowPyrLK`）
6. 连续帧跟踪
7. 本质矩阵 (RANSAC) 去除外点（`rejectWithF`）
8. 发布 `feature_points(id_of_point, un_pts, cur_pts, pts_velocity)`
9. Keyframe selection
   (a) Case 1: Rotation-compensated average feature parallax is larger than a threshold
   (b) Case 2: Number of tracked features in the current frame is less than a threshold
   (c) All frames are used for optimization, but non-keyframes are removed first

2.2 IMU 预积分

2.2.1 IMU 测量方程

忽略地球旋转，IMU 测量方程为

$$\begin{aligned} \hat{a}_t &= a_t + b_{at} + R_w^t g^w + n_a \\ \hat{\omega}_t &= \omega_t + b_{\omega t} + n_\omega \end{aligned} \tag{1}$$



2.2.2 预积分方程

(1) **IMU integration in world frame**

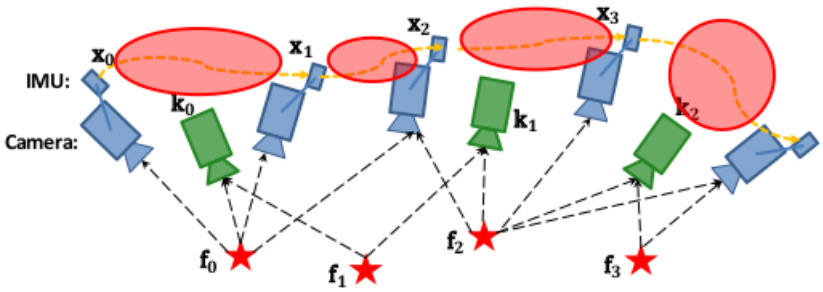
由上面的 IMU 测量方程积分就可以计算出下一时刻的 p、v 和 q:

$$\begin{aligned}
\mathbf{p}_{b_{k+1}}^{w} &= \mathbf{p}_{b_k}^{w} + \mathbf{v}_{b_k}^{w}\Delta t_k + \iint_{t\in[t_k,t_{k+1}]} \left(\mathbf{R}_t^w\left(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a\right) - \mathbf{g}^w\right) dt^2 \\
\mathbf{v}_{b_{k+1}}^{w} &= \mathbf{v}_{b_k}^{w} + \int_{t\in[t_k,t_{k+1}]} \left(\mathbf{R}_t^w\left(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t} - \mathbf{n}_a\right) - \mathbf{g}^w\right) dt \\
\mathbf{q}_{b_{k+1}}^{w} &= \mathbf{q}_{b_k}^{w} \otimes \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\mathbf{\Omega}\left(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t} - \mathbf{n}_w\right)\mathbf{q}_t^{b_k} dt
\end{aligned} \tag{2}$$

其中,

$$\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}, \lfloor\boldsymbol{\omega}\rfloor_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3}$$

(2) **IMU integration in the body frame of first pose of interests**



为避免重新传播 IMU 观测值，选用 IMU 预积分模型，从世界坐标系转为本体坐标系

$$\begin{aligned}
\mathbf{R}_w^{b_k}\mathbf{p}_{b_{k+1}}^{w} &= \mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_k}^{w} + \mathbf{v}_{b_k}^{w}\Delta t_k - \frac{1}{2}\mathbf{g}^w\Delta t_k^2\right) + \boldsymbol{\alpha}_{b_{k+1}}^{b_k} \\
\mathbf{R}_w^{b_k}\mathbf{v}_{b_{k+1}}^{w} &= \mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_k}^{w} - \mathbf{g}^w\Delta t_k\right) + \boldsymbol{\beta}_{b_{k+1}}^{b_k} \\
\mathbf{q}_w^{b_k} \otimes \mathbf{q}_{b_{k+1}}^{w} &= \gamma_{b_{k+1}}^{b_k}
\end{aligned} \tag{4}$$

则 IMU 预积分模型（**预积分估计值**）为

$$\begin{bmatrix} \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} \\ \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k+1} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_{k+1}}^w - \mathbf{p}_{b_k}^w + \frac{1}{2}\mathbf{g}^w\Delta t_k^2 - \mathbf{v}_{b_k}^w\Delta t_k\right) \\ \mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_{k+1}}^w + \mathbf{g}^w\Delta t_k - \mathbf{v}_{b_k}^w\right) \\ \mathbf{q}_{b_k}^{w_k} \otimes \mathbf{q}_{b_{k+1}} \\ \mathbf{b}_{ab_{k+1}} - \mathbf{b}_{ab_k} \\ \mathbf{b}_{wb_{k+1}} - \mathbf{b}_{wb_k} \end{bmatrix} \tag{5}$$

离散状态下采用 **中值积分 (mid-point)** 的预积分方程（**预积分测量值**）为

$$\begin{cases} \delta q_{i+1} = \delta q_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2}w_i'\delta t \end{bmatrix} \\ \delta\alpha_{i+1} = \delta\alpha_i + \delta\beta_i t + \frac{1}{2}a_i'\delta t^2 \\ \delta\beta_{i+1} = \delta\beta_i + a_i'\delta t \\ b_{ai+1} = b_{ai} + n_{ba}\delta t \\ b_{g_{i+1}} = b_{g_i} + n_{bg}\delta t \end{cases} \tag{6}$$

其中

$$\begin{aligned} w_i' &= \frac{(w_i - b_{g_i} + n_{w0}) + (w_{i+1} - b_{g_i} + n_{w1})}{2} \\ &= \frac{w_{i+1} + w_i + n_{w0} + n_{w1}}{2} - b_{g_i} \\ a_i' &= \frac{\delta q_i(a_i - b_{ai} + n_{a0}) + \delta q_{i+1}(a_{i+1} - b_{ai} + n_{a1})}{2} \end{aligned} \tag{7}$$

`midPointIntegration` 中的相关代码（没有考虑噪声）：

```
Vector3d un_gyr = 0.5 * (_gyr_0 + _gyr_1) - linearized_bg;
result_delta_q =
  delta_q * Quaterniond(1, un_gyr(0)*_dt/2, un_gyr(1)*_dt/2, un_gyr(2)*_dt/2);

Vector3d un_acc_0 =        delta_q * (_acc_0 - linearized_ba);
Vector3d un_acc_1 = result_delta_q * (_acc_1 - linearized_ba);
Vector3d un_acc   = 0.5 * (un_acc_0 + un_acc_1);

result_delta_p = delta_p + delta_v * _dt + 0.5 * un_acc * _dt * _dt;
result_delta_v = delta_v + un_acc * _dt;

// 预积分的过程中Bias没有发生改变
result_linearized_ba = linearized_ba;
result_linearized_bg = linearized_bg;
```

2.2.3 误差状态方程

IMU 误差状态向量

$$\delta X = [\delta P \quad \delta\theta \quad \delta v \quad \delta b_a \quad \delta b_g]^T \in \mathbb{R}^{15\times 1} \tag{8}$$

根据 [2] ESKF 中 ***5.3.3 The error-state kinematics*** 小节公式

$$
\begin{aligned}
\dot{\delta \mathbf{p}} &= \delta \mathbf{v} \\
\dot{\delta \mathbf{v}} &= -\mathbf{R}\left[\mathbf{a}_m - \mathbf{a}_b\right]_\times \delta\boldsymbol{\theta} - \mathbf{R}\delta\mathbf{a}_b + \delta\mathbf{g} - \mathbf{R}\mathbf{a}_n \\
\dot{\delta \boldsymbol{\theta}} &= -\left[\boldsymbol{\omega}_m - \boldsymbol{\omega}_b\right]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n \\
\delta\dot{\mathbf{a}}_b &= \mathbf{a}_w \\
\delta\dot{\boldsymbol{\omega}}_b &= \boldsymbol{\omega}_w \\
\dot{\delta \mathbf{g}} &= 0
\end{aligned} \tag{9}
$$

对于 **中值积分 (mid-point)** 下的 **误差状态方程的导数**为

$$
\delta\dot{X}_k = \begin{cases}
\delta\dot{\theta}_k = & -[\frac{w_{k+1}+w_k}{2} - b_{g_k}]_\times \delta\theta_k - \delta b_{g_k} + \frac{n_{w0}+n_{w1}}{2} \\
\delta\dot{\beta}_k = & -\frac{1}{2}q_k[a_k - b_{a_k}]_\times \delta\theta \\
& -\frac{1}{2}q_{k+1}[a_{k+1} - b_{a_k}]_\times((I - [\frac{w_{k+1}+w_k}{2} - b_{g_k}]_\times \delta t)\delta\theta_k - \delta b_{g_k}\delta t + \frac{n_{w0}+n_{w1}}{2}\delta t) \\
& -\frac{1}{2}q_k\delta b_{a_k} - \frac{1}{2}q_{k+1}\delta b_{a_k} - \frac{1}{2}q_k n_{a0} - \frac{1}{2}q_k n_{a1} \\
\delta\dot{\alpha}_k = & -\frac{1}{4}q_k[a_k - b_{a_k}]_\times \delta\theta\delta t \\
& -\frac{1}{4}q_{k+1}[a_{k+1} - b_{a_k}]_\times((I - [\frac{w_{k+1}+w_k}{2} - b_{g_k}]_\times \delta t)\delta\theta_k - \delta b_{g_k}\delta t + \frac{n_{w0}+n_{w1}}{2}\delta t)\delta t \\
& -\frac{1}{4}q_k\delta b_{a_k}\delta t - \frac{1}{4}q_{k+1}\delta b_{a_k}\delta t - \frac{1}{4}q_k n_{a0}\delta t - \frac{1}{4}q_k n_{a1}\delta t \\
\delta\dot{b}_{a_k} = & n_{b_a} \\
\delta\dot{b}_{g_k} = & n_{b_g}
\end{cases} \tag{10}
$$

则 **IMU 预积分（误差）状态方程**为

$$
\delta\dot{X}_k = F\delta X_k + Gn \tag{11}
$$

则 IMU 预积分状态传递方程为

$$
\begin{aligned}
\delta X_{k+1} &= \delta X_k + \delta\dot{X}_k\delta t \\
&= \delta X_k + (F\delta X_k + Gn)\delta t \\
&= (I + F\delta t)\delta X_k + (G\delta t)n
\end{aligned} \tag{12}
$$

令

$$
\begin{aligned}
F' &= I + F\delta t \quad \in \mathbb{R}^{15\times 15} \\
G' &= G\delta t \quad\quad\;\; \in \mathbb{R}^{15\times 18}
\end{aligned} \tag{13}
$$

最终，**IMU 预积分递推方程**为

$$
\delta X_{k+1} = F'\delta X_k + G'n \tag{14}
$$

式(14) 展开得

$$
\begin{bmatrix} \delta\alpha_{k+1} \\ \delta\theta_{k+1} \\ \delta\beta_{k+1} \\ \delta b_{ak+1} \\ \delta b_{gk+1} \end{bmatrix} = \begin{bmatrix} I & f_{12} & I\delta t & -\frac{1}{4}(q_k+q_{k+1})\delta t^2 & f_{15} \\ 0 & I-[\frac{w_{k+1}+w_k}{2}-b_{wk}]_\times\delta t & 0 & 0 & -I\delta t \\ 0 & f_{32} & I & -\frac{1}{2}(q_k+q_{k+1})\delta t & f_{35} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \delta\alpha_k \\ \delta\theta_k \\ \delta\beta_k \\ \delta b_{ak} \\ \delta b_{gk} \end{bmatrix}
$$

$$
+ \begin{bmatrix} \frac{1}{4}q_k\delta t^2 & g_{12} & \frac{1}{4}q_{k+1}\delta t^2 & g_{14} & 0 & 0 \\ 0 & \frac{1}{2}I\delta t & 0 & \frac{1}{2}I\delta t & 0 & 0 \\ \frac{1}{2}q_k\delta t & g_{32} & \frac{1}{2}q_{k+1}\delta t & g_{34} & 0 & 0 \\ 0 & 0 & 0 & 0 & I\delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & I\delta t \end{bmatrix} \begin{bmatrix} n_{a0} \\ n_{w0} \\ n_{a1} \\ n_{w1} \\ n_{ba} \\ n_{bg} \end{bmatrix} \tag{15}
$$

其中，

$$
\begin{cases}
f_{12} = \dfrac{\partial\alpha_{k+1}}{\partial\delta\theta_k} = -\dfrac{1}{4}q_k[a_k-b_{a_k}]_\times\delta t^2 - \dfrac{1}{4}q_{k+1}[a_{k+1}-b_{a_k}]_\times(I-[\dfrac{w_{k+1}+w_k}{2}-b_{g_k}]_\times\delta t)\delta t^2 \\
f_{32} = \dfrac{\partial\beta_{k+1}}{\partial\delta\theta_k} = -\dfrac{1}{2}q_k[a_k-b_{a_k}]_\times\delta t - \dfrac{1}{2}q_{k+1}[a_{k+1}-b_{a_k}]_\times(I-[\dfrac{w_{k+1}+w_k}{2}-b_{g_k}]_\times\delta t)\delta t \\
f_{15} = \dfrac{\partial\alpha_{k+1}}{\partial\delta b_{g_k}} = \dfrac{1}{4}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t^2)(-\delta t) \\
f_{35} = \dfrac{\partial\beta_{k+1}}{\partial\delta b_{g_k}} = \dfrac{1}{2}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t)(-\delta t) \\
g_{12} = \dfrac{\partial\alpha_{k+1}}{\partial n_{g_0}} = \dfrac{1}{4}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t^2)\dfrac{1}{2}\delta t \\
g_{14} = \dfrac{\partial\alpha_{k+1}}{\partial n_{g_1}} = \dfrac{1}{4}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t^2)\dfrac{1}{2}\delta t \\
g_{32} = \dfrac{\partial\beta_{k+1}}{\partial n_{g_0}} = \dfrac{1}{2}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t^2)\dfrac{1}{2}\delta t \\
g_{34} = \dfrac{\partial\beta_{k+1}}{\partial n_{g_1}} = \dfrac{1}{2}(-q_{k+1}[a_{k+1}-b_{a_k}]_\times\delta t^2)\dfrac{1}{2}\delta t
\end{cases} \tag{16}
$$

式(13)(14)中的 F' 即代码中 `F`，G' 即代码中 `V`，相关代码见 `midPointIntegration`。

另外，状态误差传递的线性递推关系可通过两种方法实现：

- **基于误差随时间变化的递推方程**
- **基于一阶泰勒展开的误差递推方程**

上面式(14) F' 和 G' 的计算是用的第一种方法；我们还可以通过第二种方法：已知 IMU 预积分（中值积分）测量方程(6)，将其表示成式(14)的形式（其中，F' 是状态量 δX_{k+1} 对状态量 δX_k 的雅克比矩阵，G' 是状态量 δX_{k+1} 对输入量 n_{k-1} 的雅克比矩阵），通过求导计算出 F' 和 G'。

最后，计算

- IMU 预积分测量关于 IMU Bias 的 **雅克比矩阵** $\mathbf{J}_{k+1}$
- IMU 预积分测量的 **协方差矩阵** $\mathbf{P}_{k+1}$
- 噪声的 **协方差矩阵** $\mathbf{Q}$

雅克比矩阵的传递方程为

$$\mathbf{J}_{b_k} = \mathbf{I}_{15\times 15} \tag{17}$$

$$\begin{aligned}\mathbf{J}_{t+\delta t} &= \mathbf{F}'\mathbf{J}_t \\ &= (\mathbf{I} + \mathbf{F}_t\delta t)\mathbf{J}_t, \quad t \in [k, k+1]\end{aligned} \tag{18}$$

协方差矩阵的传递方程为

$$\mathbf{P}_{b_k}^{b_k} = \mathbf{0}_{15\times 15} \tag{19}$$

$$\begin{aligned}\mathbf{P}_{t+\delta t}^{b_k} &= F' P_t^{b_k} F'^T + G' Q G'^T \\ &= (I + F_t\delta t)P_t^{b_k}(I + F_t\delta t)^T + (G_t\delta t)Q(G_t\delta t)^T\end{aligned} \tag{20}$$

其中，噪声协方差矩阵为

$$\mathbf{Q} = \mathrm{diag}(\sigma_{a_0}^2 \quad \sigma_{\omega_0}^2 \quad \sigma_{a_1}^2 \quad \sigma_{\omega_1}^2 \quad \sigma_{b_a}^2 \quad \sigma_{b_g}^2) \in \mathbb{R}^{18\times 18} \tag{21}$$

当 bias 估计轻微改变时，我们可以使用如下的一阶近似 **对中值积分得到的预积分测量值进行矫正**，而**不重传播**，从而得到 **更加精确的预积分测量值**（bias 修正的线性模型）

$$\begin{aligned}\boldsymbol{\alpha}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\alpha}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\alpha}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\alpha}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\beta}}_{b_{k+1}}^{b_k} + \mathbf{J}_{b_a}^{\beta}\delta\mathbf{b}_{a_k} + \mathbf{J}_{b_w}^{\beta}\delta\mathbf{b}_{w_k} \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} &\approx \hat{\boldsymbol{\gamma}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_w}^{\gamma}\delta\mathbf{b}_{w_k} \end{bmatrix}\end{aligned} \tag{22}$$

其中，式(23)(24)(25)(26)(27) 均为 $\mathbf{J}_{b_{k+1}}$ 的一部分。

$$\mathbf{J}_{b_a}^{\alpha} = \frac{\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{a_k}} \tag{23}$$

$$\mathbf{J}_{b_w}^{\alpha} = \frac{\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{w_k}} \tag{24}$$

$$\mathbf{J}_{b_a}^{\beta} = \frac{\delta\boldsymbol{\beta}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{a_k}} \tag{25}$$

$$\mathbf{J}_{b_w}^{\beta} = \frac{\delta\boldsymbol{\beta}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{w_k}} \tag{26}$$

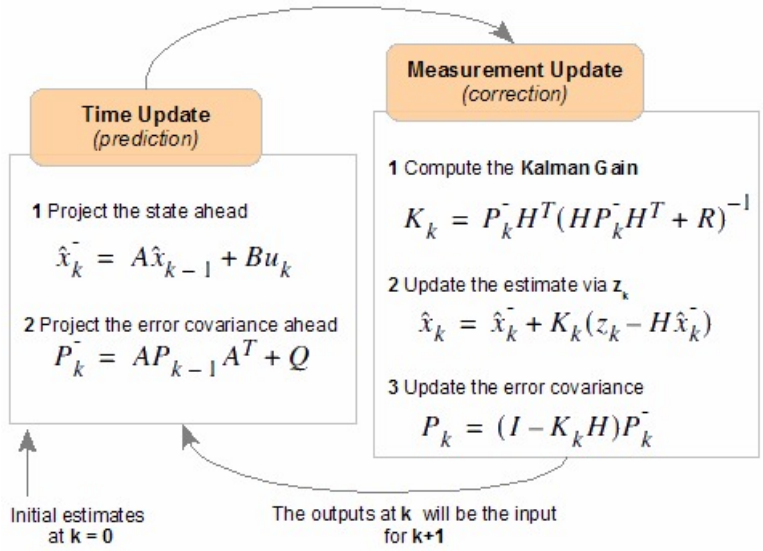
$$\mathbf{J}_{b_w}^{\gamma} = \frac{\delta\boldsymbol{\gamma}_{b_{k+1}}^{b_k}}{\delta\mathbf{b}_{w_k}} \tag{27}$$

```
// O_P = 0, O_R = 3, O_V = 6, O_BA = 9, O_BG = 12
Eigen::Matrix3d dp_dba = jacobian.block<3, 3>(O_P, O_BA);
Eigen::Matrix3d dp_dbg = jacobian.block<3, 3>(O_P, O_BG);

Eigen::Matrix3d dq_dbg = jacobian.block<3, 3>(O_R, O_BG);

Eigen::Matrix3d dv_dba = jacobian.block<3, 3>(O_V, O_BA);
Eigen::Matrix3d dv_dbg = jacobian.block<3, 3>(O_V, O_BG);
```

此时，可以与 **卡尔曼滤波**对比一下：



3 初始化（松耦合）

在提取的图像的 Features 和做完 IMU 的预积分之后，进入了系统的初始化环节，主要的目的有以下两个：

1. 系统使用单目相机，如果没有一个良好的尺度估计，就无法对两个传感器做进一步的融合，这个时候需要恢复出尺度；
2. 要对 IMU 进行初始化，IMU 会受到 bias 的影响，所以要得到 IMU 的 bias。

所以我们要从初始化中恢复出尺度、重力、速度以及 IMU 的 bias，因为视觉 (SFM) 在初始化的过程中有着较好的表现，所以在初始化的过程中主要以 SFM 为主，然后将 IMU 的预积分结果与其对齐，即可得到较好的初始化结果。

3.1 相机与 IMU 之间的相对旋转

相机与 IMU 之间的旋转标定非常重要，**偏差 1-2° 系统的精度就会变的极低** [3]。

设相机利用对极关系得到的旋转矩阵为 $R_{c_{k+1}}^{c_k}$，IMU 经过预积分得到的旋转矩阵为 $R_{b_{k+1}}^{b_k}$，相机与 IMU 之间的相对旋转为 R_c^b，则对于任一帧满足，

$$R_{b_{k+1}}^{b_k} R_c^b = R_c^b R_{c_{k+1}}^{c_k} \tag{28}$$

将旋转矩阵写为四元数，则上式可以写为

$$q_{b_{k+1}}^{b_k} \otimes q_c^b = q_c^b \otimes q_{c_{k+1}}^{c_k} \tag{29}$$

将其写为左乘和右乘的形式

$$([q_{b_{k+1}}^{b_k}]_L - [q_{c_{k+1}}^{c_k}]_R) q_c^b = Q_{k+1}^k q_c^b = 0 \tag{30}$$

$[q]_L$ 与 $[q]_R$ 分别表示 **四元数左乘矩阵**和 **四元数右乘矩阵**，其定义为（四元数实部在后）

$$\begin{aligned} {}_L &= \begin{bmatrix} q_w I_3 + [q_{xyz}]_\times & q_{xyz} \\ -q_{xyz} & q_w \end{bmatrix} \\ [q]_R &= \begin{bmatrix} q_w I_3 - [q_{xyz}]_\times & q_{xyz} \\ -q_{xyz} & q_w \end{bmatrix} \end{aligned} \tag{31}$$

那么对于 n 对测量值，则有

$$\begin{bmatrix} w_1^0 Q_1^0 \\ w_2^1 Q_2^1 \\ \vdots \\ w_N^{N-1} Q_N^{N-1} \end{bmatrix} q_c^b = Q_N q_c^b = 0 \tag{32}$$

其中 w_N^{N-1} 为外点剔除权重，其与相对旋转求得的角度残差有关，N 为计算相对旋转需要的测量对数，其由最终的终止条件决定。角度残差可以写为，

$$\theta_{k+1}^k = arccos\left(\frac{tr(\hat{R}_c^{b^{-1}} R_{b_{k+1}}^{b_k^{-1}} \hat{R}_c^b R_{c_{k+1}}^{c_k}) - 1}{2}\right) \tag{33}$$

从而权重为

$$w_{k+1}^k = \begin{cases} 1, & \theta_{k+1}^k < threshold(\text{一般 } 5°) \\ \frac{threshold}{\theta_{k+1}^k}, & otherwise \end{cases} \tag{34}$$

至此，就可以通过求解方程 $Q_N q_c^b = 0$ 得到相对旋转，解为 Q_N 的左奇异向量中最小奇异值对应的特征向量。

但是，在这里还要注意求解的终止条件 (校准完成的终止条件) 。在足够多的旋转运动中，我们可以很好的估计出相对旋转 R_c^b，这时 Q_N 对应一个准确解，且其零空间的秩为 1。但是在校准的过程中，某些轴向上可能存在退化运动 (如匀速运动)，这时 Q_N 的零空间的秩会大于 1。判断条件就是 Q_N 的第二小的奇异值是否大于某个阈值，若大于则其零空间的秩为 1，反之秩大于 1，相对旋转 R_c^b 的精度不够，校准不成功。

对应代码在 'InitialEXRotation::CalibrationExRotation' 中。

```
// 相机与IMU之间的相对旋转
if(ESTIMATE_EXTRINSIC == 2)
{
    ROS_INFO("calibrating extrinsic param, rotation movement is needed");
    if (frame_count != 0)
    {
        // 选取两帧之间共有的Features
        vector<pair<Vector3d, Vector3d>> corres = f_manager.getCorresponding(
            frame_count - 1, frame_count);

        // 校准相机与IMU之间的旋转
        Matrix3d calib_ric;
        if (initial_ex_rotation.CalibrationExRotation(corres, pre_integrations[
            frame_count]->delta_q, calib_ric))
        {
            ROS_WARN("initial extrinsic rotation calib success");
            ROS_WARN_STREAM("initial extrinsic rotation: " << endl << calib_ric);
            ric[0] = calib_ric;
            RIC[0] = calib_ric;
            ESTIMATE_EXTRINSIC = 1;
        }
    }
}
```

3.2 检测 IMU 可观性

```
// 计算均值
map<double, ImageFrame>::iterator frame_it;
Vector3d sum_g;
for (frame_it = all_image_frame.begin(), frame_it++; frame_it != all_image_frame.end
    (); frame_it++)
{
    double sum_dt  = frame_it->second.pre_integration->sum_dt;
    Vector3d tmp_g = frame_it->second.pre_integration->delta_v / sum_dt;
    sum_g += tmp_g;
}
Vector3d aver_g = sum_g * 1.0 / ((int)all_image_frame.size() - 1);

// 计算方差
double var = 0;
for (frame_it = all_image_frame.begin(), frame_it++; frame_it != all_image_frame.end
    (); frame_it++)
{
    double sum_dt  = frame_it->second.pre_integration->sum_dt;
    Vector3d tmp_g = frame_it->second.pre_integration->delta_v / sum_dt;
    var += (tmp_g - aver_g).transpose() * (tmp_g - aver_g);
}

// 计算标准差
var = sqrt(var / ((int)all_image_frame.size() - 1));
//ROS_WARN("IMU variation %f!", var);
if(var < 0.25) //! 以标准差判断可观性
{
    ROS_INFO("IMU excitation not enouth!");
    //return false;
}
```

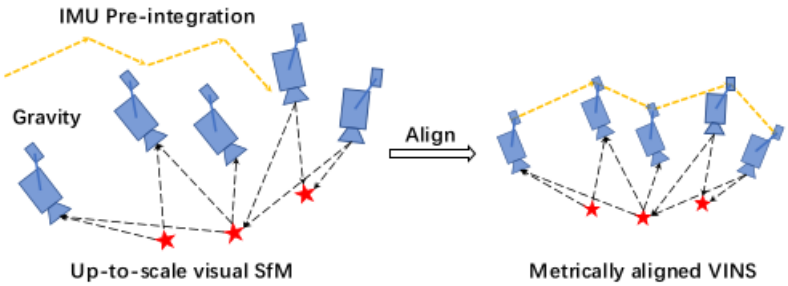
3.3 相机初始化（Vision-Only SFM）

1. 求取本质矩阵求解位姿（`relativePose`）
2. 三角化特征点（`sfm.construct`）
3. PnP 求解位姿（`cv::solvePnP`）
4. 转换到 IMU 坐标系下
5. c_0 坐标系作为参考系
6. 不断重复直到恢复出滑窗内的 Features 和相机位姿

3.4 视觉与 IMU 对齐

对应代码：`VisualIMUAlignment`

1. Gyroscope Bias Calibration
2. Velocity, Gravity Vector and Metric Scale Initialization
3. Gravity Refinement
4. Completing Initialization



3.4.1 陀螺仪 Bias 标定

标定陀螺仪 Bias 使用如下代价函数

$$\min_{\delta b_w} \sum_{k\in B} \left\| q_{b_{k+1}}^{c_0^{-1}} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} \right\|^2 \tag{35}$$

因为四元数最小值为单位四元数 $[1, 0_v]^T$，所以令

$$q_{b_{k+1}}^{c_0^{-1}} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{36}$$

其中

$$\gamma_{b_{k+1}}^{b_k} \approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix} \tag{37}$$

所以

$$\hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix} = q_{b_k}^{c_0^{-1}} \otimes q_{b_{k+1}}^{c_0} \tag{38}$$

$$\begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix} = \hat{\gamma}_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{c_0^{-1}} \otimes q_{b_{k+1}}^{c_0} \tag{39}$$

只取上式虚部，再进行最小二乘求解

$$J_{b_w}^{\gamma^T} J_{b_w}^{\gamma} \delta b_w = 2 \cdot J_{b_w}^{\gamma^T} (\hat{\gamma}_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{c_0^{-1}} \otimes q_{b_{k+1}}^{c_0})_{vec} \tag{40}$$

求解上式的最小二乘解，即可得到 δb_w，注意这个地方得到的只是 Bias 的变化量，需要在滑窗内累加得到 Bias 的准确值。

对应代码：`solveGyroscopeBias`

```
void solveGyroscopeBias(map<double, ImageFrame> &all_image_frame, Vector3d* Bgs) {
    Matrix3d A;
    Vector3d b;
    A.setZero();
    b.setZero();

    map<double, ImageFrame>::iterator frame_i;
    map<double, ImageFrame>::iterator frame_j;
    for (frame_i = all_image_frame.begin(); next(frame_i) != all_image_frame.end();
        frame_i++) {
        frame_j = next(frame_i);

        MatrixXd tmp_A(3, 3);
        VectorXd tmp_b(3);
```

```
        tmp_A.setZero();
        tmp_b.setZero();

        Eigen::Quaterniond q_ij(frame_i->second.R.transpose() * frame_j->second.R);
        tmp_A = frame_j->second.pre_integration->jacobian.template block<3, 3>(O_R,
            O_BG);
        tmp_b = 2 * (frame_j->second.pre_integration->delta_q.inverse() * q_ij).vec()
            ;

        A += tmp_A.transpose() * tmp_A;
        b += tmp_A.transpose() * tmp_b;
    }

    Vector3d delta_bg = A.ldlt().solve(b);
    ROS_WARN_STREAM("gyroscope bias initial calibration " << delta_bg.transpose());

    // 因为求解出的Bias是变化量，所以要累加
    for (int i = 0; i <= WINDOW_SIZE; i++)
        Bgs[i] += delta_bg;

    // 利用新的Bias重新repropagate
    for (frame_i = all_image_frame.begin(); next(frame_i) != all_image_frame.end();
        frame_i++) {
        frame_j = next(frame_i);
        frame_j->second.pre_integration->repropagate(Vector3d::Zero(), Bgs[0]);
    }
}
```

3.4.2 初始化速度、重力向量和尺度因子

要估计的状态量

$$X_I = [v_{b_0}^{b_0}, v_{b_1}^{b_0}, \cdots, v_{b_n}^{b_n}, g^{c_0}, s] \in \mathbb{R}^{3(n+1)+3+1} \tag{41}$$

其中，g^{c_0} 为在第 0 帧 Camera 相机坐标系下的重力向量。

根据 IMU 测量模型可知

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= R_{c_0}^{b_k}(s(\bar{p}_{b_{k+1}}^{c_0} - \bar{p}_{b_k}^{c_0}) + \frac{1}{2}g^{c_0}\Delta t_k^2 - R_{b_k}^{c_0}v_{b_k}^{b_k}\Delta t_k) \\ \beta_{b_{k+1}}^{b_k} &= R_{c_0}^{b_k}(R_{b_{k+1}}^{c_0}v_{b_{k+1}}^{b_{k+1}} + g^{c_0}\Delta t_k - R_{b_k}^{c_0}v_{b_k}^{b_k}) \end{aligned} \tag{42}$$

我们已经得到了 IMU 相对于相机的旋转 q_b^c，假设 IMU 到相机的平移量 p_b^c，那么可以很容易地将相机坐标系下的位姿转换到 IMU 坐标系下

$$\begin{aligned} q_{b_k}^{c_0} &= q_{c_k}^{c_0} \otimes (q_c^b)^{-1} \\ s\bar{p}_{b_k}^{c_0} &= s\bar{p}_{c_k}^{c_0} - R_{b_k}^{c_0}p_c^b \end{aligned} \tag{43}$$

所以，定义相邻两帧之间的 IMU 预积分出的增量（$\hat{\alpha}_{b_{k+1}}^{b_k}$，$\hat{\beta}_{b_{k+1}}^{b_k}$）与预测值之间的残差，即

$$\begin{aligned} r(\hat{z}_{b_{k+1}}^{b_k}, X_I) &= \begin{bmatrix} \delta\alpha_{b_{k+1}}^{b_k} \\ \delta\beta_{b_{k+1}}^{b_k} \end{bmatrix} \\ &= \begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} - & R_{c_0}^{b_k}(s(\bar{p}_{b_{k+1}}^{c_0} - \bar{p}_{b_k}^{c_0}) + \frac{1}{2}g^{c_0}\Delta t_k^2 - R_{b_k}^{c_0}v_{b_k}^{b_k}\Delta t_k) \\ \hat{\beta}_{b_{k+1}}^{b_k} - & R_{c_0}^{b_k}(R_{b_{k+1}}^{c_0}v_{b_{k+1}}^{b_{k+1}} + g^{c_0}\Delta t_k - R_{b_k}^{c_0}v_{b_k}^{b_k}) \end{bmatrix} \end{aligned} \tag{44}$$

令 $r(\hat{z}^{b_k}_{b_{k+1}}, X_I) = \mathbf{0}$，转换成 $Hx = b$ 的形式

$$\begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2}R^{b_k}_{c_0}\Delta t_k^2 & R^{b_k}_{c_0}(\bar{p}^{c_0}_{c_{k+1}} - \bar{p}^{c_0}_{c_k}) \\ -I & R^{b_k}_{c_0}R^{c_0}_{b_{k+1}} & R^{b_k}_{c_0}\Delta t_k & 0 \end{bmatrix} \begin{bmatrix} v^{b_k}_{b_k} \\ v^{b_{k+1}}_{b_{k+1}} \\ g^{c_0} \\ s \end{bmatrix} = \begin{bmatrix} \alpha^{b_k}_{b_{k+1}} - p^b_c + R^{b_k}_{c_0}R^{c_0}_{b_{k+1}}p^b_c \\ \beta^{b_k}_{b_{k+1}} \end{bmatrix} \tag{45}$$

通过 Cholosky 分解求解 X_I

$$H^T H X_I = H^T b \tag{46}$$

对应代码：‘LinearAlignment‘

3.4.3 优化重力

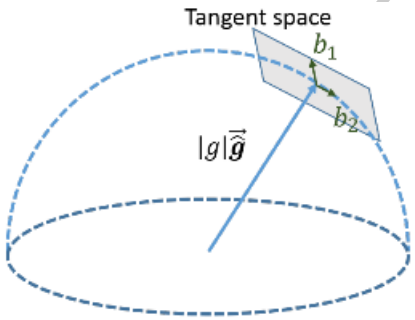


Fig. 5. Illustration of 2 DOF parameterization of gravity. Since the magnitude of gravity is known, $\mathbf{g}$ lies on a sphere with radius $g \approx 9.81m/s^2$. The gravity is parameterized around current estimate as $g \cdot \hat{\bar{\mathbf{g}}} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2$, where $\mathbf{b}_1$ and $\mathbf{b}_2$ are two orthogonal basis spanning the tangent space.

重力矢量的模长固定（9.8），其为 2 个自由度，在切空间上对其参数化

$$\begin{aligned} \hat{g} &= \|g\| \cdot \hat{\bar{g}} + \omega_1\vec{b_1} + \omega_2\vec{b_2} \\ &= \|g\| \cdot \hat{\bar{g}} + B\vec{\omega} \end{aligned}, \quad B \in \mathbb{R}^{3\times 2}, \vec{\omega} \in \mathbb{R}^{2\times 1} \tag{47}$$

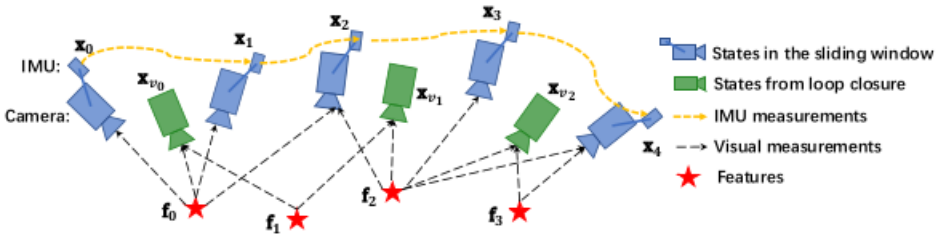
令 $\hat{g} = g^{c_0}$，将其代入上一小节公式得

$$\begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2}R^{b_k}_{c_0}\Delta t_k^2 B & R^{b_k}_{c_0}(\bar{p}^{c_0}_{c_{k+1}} - \bar{p}^{c_0}_{c_k}) \\ -I & R^{b_k}_{c_0}R^{c_0}_{b_{k+1}} & R^{b_k}_{c_0}\Delta t_k B & 0 \end{bmatrix} \begin{bmatrix} v^{b_k}_{b_k} \\ v^{b_{k+1}}_{b_{k+1}} \\ \vec{\omega} \\ s \end{bmatrix} = \begin{bmatrix} \alpha^{b_k}_{b_{k+1}} - p^b_c + R^{b_k}_{c_0}R^{c_0}_{b_{k+1}}p^b_c - \frac{1}{2}R^{b_k}_{c_0}\Delta t_k^2\|g\| \cdot \hat{\bar{g}} \\ \beta^{b_k}_{b_{k+1}} - R^{b_k}_{c_0}\Delta t_k\|g\| \cdot \hat{\bar{g}} \end{bmatrix} \tag{48}$$

同样，通过 Cholosky 分解求得 g^{c_0}，即相机 C_0 系下的重力向量。

最后，通过将 g^{c_0} 旋转至惯性坐标系（世界系）中的 z 轴方向 [0,0,1]，可以计算第一帧相机系到惯性系的旋转矩阵 $q^w_{c_0}$，这样就可以将所有变量调整至惯性世界系（**水平坐标系**，z 轴与重力方向对齐）中。

对应代码：`RefineGravity`



4 后端优化（紧耦合）

VIO 紧耦合方案的主要思路就是通过将基于视觉构造的残差项和基于 IMU 构造的残差项放在一起构造成一个联合优化的问题，整个优化问题的最优解即可认为是比较准确的状态估计。

为了限制优化变量的数目，VINS-Mono 采用了滑动窗口的形式，**滑动窗口中的全状态量**：

$$
\begin{aligned}
X &= [x_0, x_1, \cdots, x_n, x_c^b, \lambda_0, \lambda_1, \cdots, \lambda_m] \\
x_k &= [p_{b_k}^w, v_{b_k}^w, q_{b_k}^w, b_a, b_g], \quad k \in [0, n] \\
x_c^b &= [p_c^b, q_c^b]
\end{aligned} \tag{49}
$$

1. 滑动窗口内 n+1 个所有相机的状态 (包括位置、朝向、速度、加速度计 bias 和陀螺仪 bias)
2. Camera 到 IMU 的外参
3. m+1 个 3D 点的逆深度

优化过程中的 **误差状态量**

$$
\begin{aligned}
\delta X &= [\delta x_0, \delta x_1, \cdots, \delta x_n, \delta x_c^b, \lambda_0, \delta\lambda_1, \cdots, \delta\lambda_m] \\
\delta x_k &= [\delta p_{b_k}^w, \delta v_{b_k}^w, \delta\theta_{b_k}^w, \delta b_a, \delta b_g], \quad k \in [0, n] \\
\delta x_c^b &= [\delta p_c^b, \delta q_c^b]
\end{aligned} \tag{50}
$$

进而得到系统优化的代价函数（Minimize residuals from all sensors）

$$
\min_{\mathcal{X}} \left\{ \|\mathbf{r}_p - \mathbf{H}_p \mathcal{X}\|^2 + \sum_{k \in \mathcal{B}} \left\| \mathbf{r}_{\mathcal{B}} \left(\hat{\mathbf{z}}_{b_{k+1}}^{b_k}, \mathcal{X} \right) \right\|_{\mathbf{P}_{b_k+1}^2}^2 + \sum_{(l,j) \in \mathcal{C}} \rho \left(\left\| \mathbf{r}_{\mathcal{C}} \left(\hat{\mathbf{z}}_l^{c_j}, \mathcal{X} \right) \right\|_{\mathbf{P}_l^{c_j}}^2 \right) \right\} \tag{51}
$$

其中三个残差项依次是边缘化的先验信息、IMU 测量残差、视觉的观测残差，都是用 **马氏距离**（与量纲无关）来表示的。

Motion-only visual-inertial bundle adjustment: Optimize **position, velocity, rotation** in a smaller windows, assuming all other quantities are fixed

4.1 IMU 测量残差

1. IMU 测量残差

   根据上面的 IMU 预积分，得到 **IMU 预积分残差（估计值 - 测量值）**

$$\mathbf{r}_{\mathcal{B}}\left(\hat{\mathbf{z}}_{b_{k+1}}^{b_k},\mathcal{X}\right)=\begin{bmatrix}\delta\boldsymbol{\alpha}_{b_{k+1}}^{b_k}\\ \delta\boldsymbol{\theta}_{b_{k+1}}^{b_k}\\ \delta\boldsymbol{\beta}_{b_{k+1}}^{b_k}\\ \delta\mathbf{b}_{b_k}\\ \delta\mathbf{b}_{g_k}\end{bmatrix}_{15\times1}=\begin{bmatrix}\mathbf{R}_w^{b_k}\left(\mathbf{p}_{b_{k+1}}^w-\mathbf{p}_{b_k}^w+\frac{1}{2}\mathbf{g}^w\Delta t_k^2-\mathbf{v}_{b_k}^w\Delta t_k\right)-\boldsymbol{\alpha}_{b_{k+1}}^{b_k}\\ 2\left[\mathbf{q}_{b_k}^{w^{-1}}\otimes\mathbf{q}_{b_{k+1}}^w\otimes\left(\gamma_{b_{k+1}}^{b_k}\right)^{-1}\right]_{xyz}\\ \mathbf{R}_w^{b_k}\left(\mathbf{v}_{b_{k+1}}^w+\mathbf{g}^w\Delta t_k-\mathbf{v}_{b_k}^w\right)-\boldsymbol{\beta}_{b_{k+1}}^{b_k}\\ \mathbf{b}_{ab_{k+1}}-\mathbf{b}_{ab_k}\\ \mathbf{b}_{wb_{k+1}}-\mathbf{b}_{wb_k}\end{bmatrix}\tag{52}$$

其中 $[\alpha_{b_{k+1}}^{b_k},\gamma_{b_{k+1}}^{b_k},\beta_{b_{k+1}}^{b_k}]$ 为 **IMU 预积分测量 Bias 修正值**。

将式(52) 写成四元数形式

$$\mathbf{r}_{\mathcal{B}}=\begin{bmatrix}\mathbf{r}_p\\ \mathbf{r}_q\\ \mathbf{r}_v\\ \mathbf{r}_{ba}\\ \mathbf{r}_{bg}\end{bmatrix}=\begin{bmatrix}\mathbf{q}_{b_iw}\left(\mathbf{p}_{wb_j}-\mathbf{p}_{wb_i}-\mathbf{v}_i^w\Delta t+\frac{1}{2}\mathbf{g}^w\Delta t^2\right)-\boldsymbol{\alpha}_{b_ib_j}\\ 2\left[\mathbf{q}_{b_jb_i}\otimes\left(\mathbf{q}_{b_iw}\otimes\mathbf{q}_{wb_j}\right)\right]_{xyz}\\ \mathbf{q}_{b_iw}\left(\mathbf{v}_j^w-\mathbf{v}_i^w+\mathbf{g}^w\Delta t\right)-\boldsymbol{\beta}_{b_ib_j}\\ \mathbf{b}_j^a-\mathbf{b}_i^a\\ \mathbf{b}_j^g-\mathbf{b}_i^g\end{bmatrix}_{15\times1}\tag{53}$$

```
Eigen::Quaterniond dq_rected = delta_q * Utility::deltaQ(dq_dbg * dbg);
Eigen::Vector3d    dv_rected = delta_v + dv_dba * dba + dv_dbg * dbg;
Eigen::Vector3d    dp_rected = delta_p + dp_dba * dba + dp_dbg * dbg;

residuals.block<3, 1>(O_P, 0)  =
    Qi.inverse() * (0.5*G*sum_dt*sum_dt + Pj - Pi - Vi*sum_dt) - dp_rected;
residuals.block<3, 1>(O_R, 0)  =
    2 * (dq_rected.inverse() * (Qi.inverse() * Qj)).vec();
residuals.block<3, 1>(O_V, 0)  =
    Qi.inverse() * (G * sum_dt + Vj - Vi) - dv_rected;
residuals.block<3, 1>(O_BA, 0) = Baj - Bai;
residuals.block<3, 1>(O_BG, 0) = Bgj - Bgi;
```

2. 协方差矩阵

此处用到的协方差矩阵为前面 IMU 预积分计算出的协方差矩阵 $\mathbf{P}$。

残差的后处理对应代码：

```
Eigen::Map<Eigen::Matrix<double, 15, 1>> residual(residuals);
residual=pre_integration->evaluate(Pi, Qi, Vi, Bai, Bgi, Pj, Qj, Vj, Baj, Bgj);

Eigen::Matrix<double, 15, 15> cov_inv = pre_integration->covariance.inverse();
Eigen::Matrix<double, 15, 15> sqrt_info =
    Eigen::LLT<Eigen::Matrix<double, 15, 15>>(cov_inv).matrixL().transpose();

residual = sqrt_info * residual;
```

这里残差 `residual` 乘以 `sqrt_info`，是因为真正的优化项其实是 Mahalanobis 距离

$$d=\mathbf{r}^T\mathbf{P}^{-1}\mathbf{r}\tag{54}$$

Mahalanobis 距离其实相当于一个残差加权，协方差大的加权小，协方差小的加权大，着重优化那些比较确定的残差。

因为 ceres 只接受最小二乘优化，也就是 $\min e^Te$，所以把 P^{-1} 做 LLT 分解

$$\mathbf{L}\mathbf{L}^T=\mathbf{P}^{-1}\tag{55}$$

则

$$d = \mathbf{r}^T(\mathbf{L}\mathbf{L}^T)\mathbf{r} = (\mathbf{L}^T\mathbf{r})^T(\mathbf{L}^T\mathbf{r}) \tag{56}$$

令 $\mathbf{r}' = (\mathbf{L}^T\mathbf{r})$ 作为新的优化误差，所以

$$\text{sqrt\_info} = \mathbf{L}^T \tag{57}$$

3. 雅克比矩阵

高斯迭代优化过程中会用到 IMU 测量残差对状态量的雅克比矩阵，但此处我们是 **对误差状态量求偏导**，下面根据式(53) 对四部分误差状态量求取雅克比矩阵。

(a) 对i 时刻 $[\delta p_{b_i}^w, \delta\theta_{b_i}^w]$ 求偏导

$$\mathbf{J}[0] = \begin{bmatrix} -\mathbf{R}_{b_iw} & \left[\mathbf{R}_{b_iw}\left(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w\Delta t + \frac{1}{2}\mathbf{g}^w\Delta t^2\right)\right]_\times \\ \mathbf{0} & -2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\left[\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i}\right]_L\left[\mathbf{q}_{b_ib_j}\right]_R\begin{bmatrix}\mathbf{0} \\ \frac{1}{2}\mathbf{I}\end{bmatrix} \\ \mathbf{0} & \left[\mathbf{R}_{b_iw}\left(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w\Delta t\right)\right]_\times \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{15\times 7} \tag{58}$$

(b) 对i 时刻 $[\delta v_i^w, \delta b_i^a, \delta b_i^g]$ 求偏导

$$\mathbf{J}[1] = \begin{bmatrix} -\mathbf{R}_{b_iw}\Delta t & -\mathbf{J}_{b_i^a}^{\alpha} & -\mathbf{J}_{b_i^g}^{\alpha} \\ \mathbf{0} & \mathbf{0} & -2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\left[\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_ib_j}\right]_L\begin{bmatrix}\mathbf{0} \\ \frac{1}{2}\mathbf{J}_{b_i^g}^{q}\end{bmatrix} \\ -\mathbf{R}_{b_iw} & -\mathbf{J}_{b_i^a}^{\beta} & -\mathbf{J}_{b_i^g}^{\beta} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} \end{bmatrix} \in \mathbb{R}^{15\times 9} \tag{59}$$

(c) 对j 时刻 $[\delta p_{b_j}^w, \delta\theta_{b_j}^w]$ 求偏导

$$\mathbf{J}[2] = \begin{bmatrix} \mathbf{R}_{b_iw} & \mathbf{0} \\ \mathbf{0} & 2\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\left[\mathbf{q}_{b_ib_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j}\right]_L\begin{bmatrix}\mathbf{0} \\ \frac{1}{2}\mathbf{I}\end{bmatrix} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{15\times 7} \tag{60}$$

(d) 对j 时刻 $[\delta v_j^w, \delta b_j^a, \delta b_j^g]$ 求偏导

$$\mathbf{J}[3] = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_{b_iw} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{15\times 9} \tag{61}$$

雅克比矩阵计算的对应代码在`class IMUFactor : public ceres::SizedCostFunction` 中的 `Evaluate()` 函数中。

4.2 视觉测量残差

视觉测量残差即 **特征点的重投影误差**，视觉残差和雅克比矩阵计算的对应代码在`ProjectionFactor::E`函数中。

1. 重投影误差（视觉测量残差）

   对于第 i 帧中的特征点，它投影到第 j 帧相机坐标系下的值为

$$\begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \\ 1 \end{bmatrix} = \mathbf{T}_{bc}^{-1}\mathbf{T}_{wb_j}^{-1}\mathbf{T}_{wb_i}\mathbf{T}_{bc}\begin{bmatrix} \frac{1}{\lambda}u_{c_i} \\ \frac{1}{\lambda}v_{c_i} \\ \frac{1}{\lambda} \\ 1 \end{bmatrix} \tag{62}$$

   拆成三维坐标形式为

$$\begin{aligned}\mathbf{P}_{c_j} = \begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \end{bmatrix} &= \mathbf{R}_{bc}^{\top}\left(\mathbf{R}_{wb_j}^{\top}\left(\mathbf{R}_{wb_i}\left(\mathbf{R}_{bc}\mathbf{P}_{c_i}+\mathbf{p}_{bc}\right)+\mathbf{p}_{wb_i}-\mathbf{p}_{wb_j}\right)-\mathbf{p}_{bc}\right) \\ &= \mathbf{R}_{bc}^{\top}\mathbf{R}_{wb_j}^{\top}\mathbf{R}_{wb_i}\mathbf{R}_{bc}\mathbf{P}_{c_i}+\mathbf{R}_{bc}^{\top}\left(\mathbf{R}_{wb_j}^{\top}\left(\left(\mathbf{R}_{wb_i}\mathbf{p}_{bc}+\mathbf{p}_{wb_i}\right)-\mathbf{p}_{wb_j}\right)-\mathbf{p}_{bc}\right)\end{aligned} \tag{63}$$

   其中，

$$\mathbf{P}_{c_i} = \frac{\bar{\mathbf{P}}_{c_i}}{\lambda} = \frac{1}{\lambda}\begin{bmatrix} u_{c_i} \\ v_{c_i} \\ 1 \end{bmatrix} \tag{64}$$

   为了后面方便求 Jacobian，对式(63)拆解，定义如下变量

$$\begin{aligned}\mathbf{f}_{b_i} &= \mathbf{R}_{bc}\mathbf{P}_{c_i}+\mathbf{p}_{bc} \\ \mathbf{f}_{w} &= \mathbf{R}_{wb_i}\mathbf{f}_{b_i}+\mathbf{p}_{wb_i} \\ \mathbf{f}_{b_j} &= \mathbf{R}_{wb_j}^{\top}\left(\mathbf{f}_{w}-\mathbf{p}_{wb_j}\right) \\ \mathbf{P}_{c_j} &= \mathbf{R}_{bc}^{\top}\left(\mathbf{f}_{b_j}-\mathbf{p}_{bc}\right)\end{aligned} \tag{65}$$

```
// 将第i frame下的3D点转到第j frame坐标系下
Eigen::Vector3d pts_camera_i = pts_i / inv_dep_i;
Eigen::Vector3d pts_imu_i    = qic * pts_camera_i + tic;
Eigen::Vector3d pts_w        = Qi * pts_imu_i + Pi;
Eigen::Vector3d pts_imu_j    = Qj.inverse() * (pts_w - Pj);
Eigen::Vector3d pts_camera_j = qic.inverse() * (pts_imu_j - tic);
```

   视觉测量残差为

$$\mathbf{r}_c = \Sigma\cdot\begin{bmatrix} \frac{x_{c_j}}{z_{c_j}}-u_{c_j} \\ \frac{y_{c_j}}{z_{c_j}}-v_{c_j} \end{bmatrix} = \Sigma\cdot\left(\frac{P_{c_j}}{Z_j}-\hat{\bar{P}}_{c_j}\right)_2 \in \mathbb{R}^{2\times 1} \tag{66}$$

```
Eigen::Map<Eigen::Vector2d> residual(residuals);
#ifdef UNIT_SPHERE_ERROR
// why
// 把归一化平面上的重投影误差投影到Unit sphere上的好处就是可以支持所有类型的相机
// 求取切平面上的误差
residual =  tangent_base * (pts_camera_j.normalized() - pts_j.normalized());
#else
// 求取归一化平面上的误差
```

```
double dep_j = pts_camera_j.z();
residual = (pts_camera_j / dep_j).head<2>() - pts_j.head<2>();
#endif
residual = sqrt_info * residual; // 转成与量纲无关的马氏距离
```

2. 协方差矩阵

固定的协方差矩阵，归一化平面的标准差为 $\frac{1.5}{f}$，即像素标准差为 1.5

$$\Sigma = \begin{bmatrix} \frac{f}{1.5} & 0 \\ 0 & \frac{f}{1.5} \end{bmatrix} \tag{67}$$

```
ProjectionFactor::sqrt_info = FOCAL_LENGTH / 1.5 * Matrix2d::Identity();
```

3. 雅克比矩阵

Jacobian 为视觉误差对两个时刻的状态量、外参以及逆深度求导

$$\mathbf{J} = \frac{\partial \mathbf{r}_c}{\partial \mathbf{X}_c} = \begin{bmatrix} \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_i} \\ \delta \boldsymbol{\theta}_{wb_i} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_j} \\ \delta \boldsymbol{\theta}_{wb_j} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{bc} \\ \delta \boldsymbol{\theta}_{bc} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \lambda} \end{bmatrix} \tag{68}$$

根据链式法则

$$\begin{aligned} \mathbf{J} &= \frac{\partial \mathbf{r}_c}{\partial \mathbf{P}_{c_j}} \frac{\partial \mathbf{P}_{c_j}}{\partial \mathbf{X}_c} = \mathbf{J}_1 \mathbf{J}_2 \\ &= \frac{\partial \mathbf{r}_c}{\partial \mathbf{P}_{c_j}} \begin{bmatrix} \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_i} \\ \delta \boldsymbol{\theta}_{wb_i} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_j} \\ \delta \boldsymbol{\theta}_{wb_j} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{bc} \\ \delta \boldsymbol{\theta}_{bc} \end{bmatrix}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \lambda} \end{bmatrix} \end{aligned} \tag{69}$$

先求 $\mathbf{J}$ 的第一部分 $\mathbf{J}_1$，根据式(66)

$$\mathbf{J}_1 = \frac{\partial \mathbf{r}_c}{\partial \mathbf{P}_{c_j}} = \Sigma \cdot \begin{bmatrix} \frac{1}{z_{c_j}} & 0 & -\frac{x_{c_j}}{z_{c_j}^2} \\ 0 & \frac{1}{z_{c_j}} & -\frac{y_{c_j}^2}{z_{c_j}^2} \end{bmatrix} \in \mathbb{R}^{2\times 3} \tag{70}$$

```
Eigen::Matrix<double, 2, 3> reduce(2, 3);
reduce << 1. / dep_j, 0, -pts_camera_j(0) / (dep_j * dep_j),
          0, 1. / dep_j, -pts_camera_j(1) / (dep_j * dep_j);
reduce = sqrt_info * reduce;
```

再求 $\mathbf{J}$ 的第二部分 $\mathbf{J}_2$，根据式(63)(64)(65)，对各状态量求偏导。

(a) 对i 时刻的状态量 $[\delta p_{b_i}^w, \delta\theta_{b_i}^w]$ 求偏导

$$\begin{aligned} \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \mathbf{p}_{wb_i}} &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \\ \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \boldsymbol{\theta}_{wb_i}} &= \frac{\partial \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \left(\mathbf{I} + [\delta \boldsymbol{\theta}_{wb_i}]_\times\right) \mathbf{f}_{b_i}}{\partial \delta \boldsymbol{\theta}_{wb_i}} \quad \text{(丢弃了不相关部分)} \\ &= -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} [\mathbf{f}_{b_i}]_\times \end{aligned} \tag{71}$$

$$\mathbf{J}_2[0] = \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_i} \\ \delta \boldsymbol{\theta}_{wb_i} \end{bmatrix}} = \begin{bmatrix} \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top & -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} [\mathbf{f}_{b_i}]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{72}$$

(b) 对j 时刻的状态量 $[\delta p_{b_j}^w, \delta\theta_{b_j}^w]$ 求偏导

$$\begin{aligned}
\frac{\partial \mathbf{P}_{c_j}}{\partial \delta \mathbf{p}_{wb_j}} &= -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \\
\frac{\partial \mathbf{P}_{c_j}}{\partial \delta \boldsymbol{\theta}_{wb_j}} &= \frac{\partial \mathbf{R}_{bc}^\top \left(\mathbf{I} - \left[\delta\boldsymbol{\theta}_{wb_j}\right]_\times\right) \mathbf{R}_{wb_j}^\top \left(\mathbf{f}_w - \mathbf{p}_{wb_j}\right)}{\partial \delta \boldsymbol{\theta}_{wb_j}} \\
&= \frac{\partial \mathbf{R}_{bc}^\top \left(\mathbf{I} - \left[\delta\boldsymbol{\theta}_{wb_j}\right]_\times\right) \mathbf{f}_{b_j}}{\partial \delta \boldsymbol{\theta}_{wb_j}} \\
&= \mathbf{R}_{bc}^\top \left[\mathbf{f}_{b_j}\right]_\times
\end{aligned} \tag{73}$$

$$\mathbf{J}_2[1] = \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{wb_j} \\ \delta \boldsymbol{\theta}_{wb_j} \end{bmatrix}} = \begin{bmatrix} -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top & \mathbf{R}_{bc}^\top \left[\mathbf{f}_{b_j}\right]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{74}$$

(c) 对imu 和相机之间的外参 $[\delta p_c^b, \delta\theta_c^b]$ 求偏导

$$\begin{aligned}
\frac{\partial \mathbf{P}_{c_j}}{\partial \delta \mathbf{p}_{bc}} &= \mathbf{R}_{bc}^\top \left(\mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} - \mathbf{I}_{3\times 3}\right) \\
\frac{\partial \mathbf{P}_{c_j}}{\partial \delta \boldsymbol{\theta}_{bc}} &= \frac{\partial \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{P}_{c_i}}{\partial \delta \boldsymbol{\theta}_{bc}} + \frac{\partial \mathbf{R}_{bc}^\top \left(\mathbf{R}_{wb_j}^\top \left(\left(\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}\right) - \mathbf{p}_{wb_j}\right) - \mathbf{p}_{bc}\right)}{\partial \delta \boldsymbol{\theta}_{bc}} \\
&= \frac{\partial \left(\mathbf{I} - \left[\delta\boldsymbol{\theta}_{bc}\right]_\times\right) \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \left(\mathbf{I} + \left[\delta\boldsymbol{\theta}_{bc}\right]_\times\right) \mathbf{P}_{c_i}}{\partial \delta \boldsymbol{\theta}_{bc}} \\
&\quad + \frac{\partial \left(\mathbf{I} - \left[\delta\boldsymbol{\theta}_{bc}\right]_\times\right) \mathbf{R}_{bc}^\top \left(\mathbf{R}_{wb_j}^\top \left(\left(\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}\right) - \mathbf{p}_{wb_j}\right) - \mathbf{p}_{bc}\right)}{\partial \delta \boldsymbol{\theta}_{bc}} \\
&= \left(-\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \left[\mathbf{P}_{c_i}\right]_\times + \left[\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{P}_{c_i}\right]_\times\right) \\
&\quad + \left[\mathbf{R}_{bc}^\top \left(\mathbf{R}_{wb_j}^\top \left(\left(\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}\right) - \mathbf{p}_{wb_j}\right) - \mathbf{p}_{bc}\right)\right]_\times
\end{aligned} \tag{75}$$

$$\mathbf{J}_2[2] = \frac{\partial \mathbf{P}_{c_j}}{\partial \begin{bmatrix} \delta \mathbf{p}_{bc} \\ \delta \boldsymbol{\theta}_{bc} \end{bmatrix}} = \begin{bmatrix} \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \mathbf{p}_{bc}} & \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \boldsymbol{\theta}_{bc}} \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{76}$$

(d) 对特征逆深度 $\delta\lambda$ 求偏导

$$\begin{aligned}
\frac{\partial \mathbf{P}_{c_j}}{\partial \delta \lambda} &= \frac{\partial \mathbf{P}_{c_j}}{\partial \mathbf{P}_{c_i}} \frac{\partial \mathbf{P}_{c_i}}{\partial \delta \lambda} \\
&= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \left(-\frac{1}{\lambda^2} \begin{bmatrix} u_{c_i} \\ v_{c_i} \\ 1 \end{bmatrix}\right) \\
&= -\frac{1}{\lambda} \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{P}_{c_i}
\end{aligned} \tag{77}$$

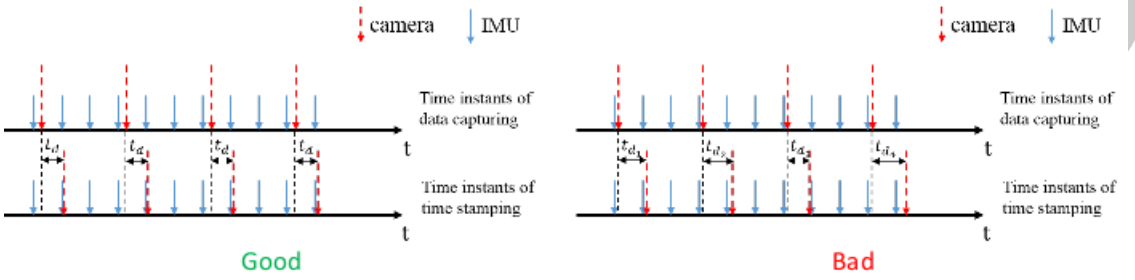
$$\mathbf{J}_2[3] = \frac{\partial \mathbf{P}_{c_j}}{\partial \delta \lambda} \in \mathbb{R}^{3\times 1} \tag{78}$$

4.3 Temporal Calibration

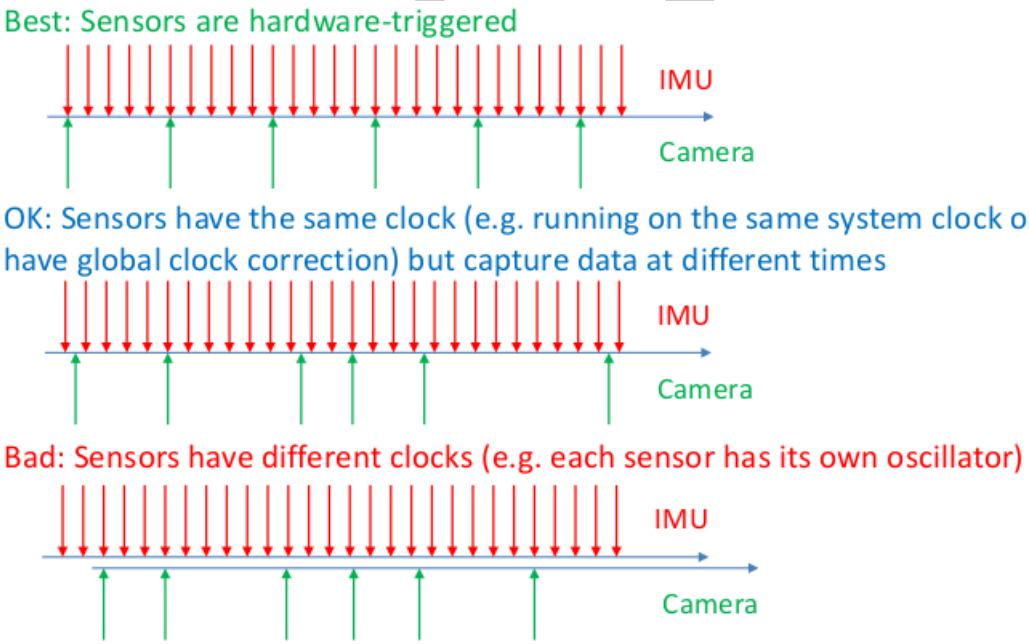
根据港科大沈老师 PPT [4]，可以知道：

1. Timestamps

- Timestamp: how the time for each sensor measurement is tagged
- Best: timestamping is done at data capture
- OK: fixed latency for time stamping
  - e.g. time is tagged on low-level hardware after some fixed-duration data processing, and will not be affected by any dynamic OS scheduling tasks
- Bad: variable latency in time stamping
  - e.g. plug two sensors into USB ports and time stamp according to the PC time. Time stamping is affected by data transmission latency from the sensor to PC



2. Time Synchronization



3. Temporal Calibration

- calibrate the fixed latency t_d occurred during time stamping
- change the IMU pre-integration interval to the interval between two image timestamps
  - linear incorporation of IMU measurements to obtain the IMU reading at image time stamping
  - estimates states(position, orientation, etc.) at image time stamping

4. Vision measurement residual for temporal calibration [5]

   - Feature velocity on image plane
     feature l moves at speed V_l^k from image k to $k+1$ in short time period $[t_k, t_{k+1}]$

$$\mathbf{V}_l^k = \left(\begin{bmatrix} u_l^{k+1} \\ v_l^{k+1} \end{bmatrix} - \begin{bmatrix} u_l^k \\ v_l^k \end{bmatrix} \right) / \left(t_{k+1} - t_k \right) \tag{79}$$

   - Visual measurement residual with time offset
     New state variable t_d, and estimate states $(c^{i'}, c^{j'})$ at time stamping

$$\begin{aligned} \mathbf{e}_l^j &= \mathbf{z}_l^j \left(t_d \right) - \pi \left(\mathbf{R}_{c_i}^{w^T} \left(\mathbf{R}_{c_j}^w \lambda_i \begin{bmatrix} \mathbf{z}_l^i \left(t_d \right) \\ 1 \end{bmatrix} + \mathbf{p}_{c_i}^w - \mathbf{p}_{c_j}^w \right) \right) \\ \mathbf{z}_l^i &= \left[u_l^i v_l^i \right]^T + t_d \mathbf{V}_l^i, \quad \mathbf{z}_l^j = \left[u_l^j v_l^j \right]^T + t_d \mathbf{V}_l^j \end{aligned} \tag{80}$$

视觉残差和雅克比矩阵计算的对应代码在 `ProjectionTdFactor::Evaluate` 函数中。

```
// TR / ROW * row_i 是相机 rolling 到这一行时所用的时间
Eigen::Vector3d pts_i_td, pts_j_td;
pts_i_td = pts_i - (td - td_i + TR / ROW * row_i) * velocity_i;
pts_j_td = pts_j - (td - td_j + TR / ROW * row_j) * velocity_j;

Eigen::Vector3d pts_camera_i = pts_i_td / inv_dep_i;
Eigen::Vector3d pts_imu_i    = qic * pts_camera_i + tic;
Eigen::Vector3d pts_w        = Qi * pts_imu_i + Pi;
Eigen::Vector3d pts_imu_j    = Qj.inverse() * (pts_w - Pj);
Eigen::Vector3d pts_camera_j = qic.inverse() * (pts_imu_j - tic);

Eigen::Map<Eigen::Vector2d> residual(residuals);
#ifdef UNIT_SPHERE_ERROR
residual =  tangent_base * (pts_camera_j.normalized() - pts_j_td.normalized());
#else
double dep_j = pts_camera_j.z();
residual = (pts_camera_j / dep_j).head<2>() - pts_j_td.head<2>();
#endif
residual = sqrt_info * residual;
```

添加对 imu-camera 时间戳不完全同步和 Rolling Shutter 相机的支持：通过前端光流计算得到每个角点在归一化的速度，根据 imu-camera 时间戳的时间同步误差和 Rolling Shutter 相机做一次 rolling 的时间，对角点的归一化坐标进行调整

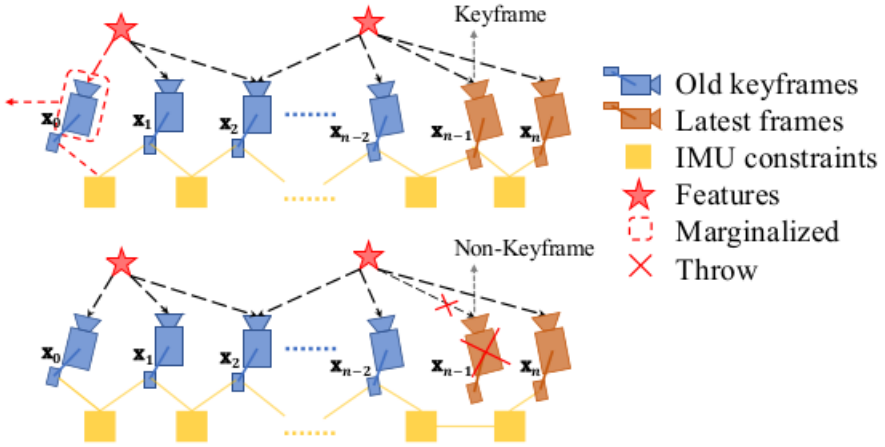
4.4 边缘化 (Marginalization)

SLAM is tracking a noraml distribution through a large state space.

滑窗 (Sliding Window) 限制了关键帧的数量，防止 pose 和 feature 的个数不会随时间不断增加，使得优化问题始终在一个有限的复杂度内，不会随时间不断增长。

4.4.1 Marginalization

然而，将 pose 移出 windows 时，有些约束会被丢弃掉，这样势必会导致求解的精度下降，而且当 MAV 进行一些退化运动 (如: 匀速运动) 时，没有历史信息做约束的话是无法求解的。所以，在移出位姿或特征的时候，需要将相关联的约束转变成一个约束项作为 prior 放到优化问题中，这就是 marginalization 要做的事情。



边缘化的过程就是将滑窗内的某些较旧或者不满足要求的视觉帧剔除的过程，所以边缘化也被描述为**将联合概率分布分解为边缘概率分布和条件概率分布的过程** (就是**利用 shur 补减少优化参数**的过程)。

直接进行边缘化而不加入先验条件的后果：

1. 无故地移除这些 pose 和 feature 会丢弃帧间约束，会降低了优化器的精度，所以在移除 pose 和 feature 的时候需要将相关联的约束转变为一个先验的约束条件作为 prior 放到优化问题中
2. 在边缘化的过程中，不加先验的边缘化会导致系统尺度的缺失 (参考 [6])，尤其是系统在进行退化运动时 (如无人机的悬停和恒速运动)。一般来说 **只有两个轴向的加速度不为 0 的时候，才能保证尺度可观**，而退化运动对于无人机或者机器人来说是不可避免的。所以在系统处于退化运动的时候，要加入先验信息保证尺度的可观性

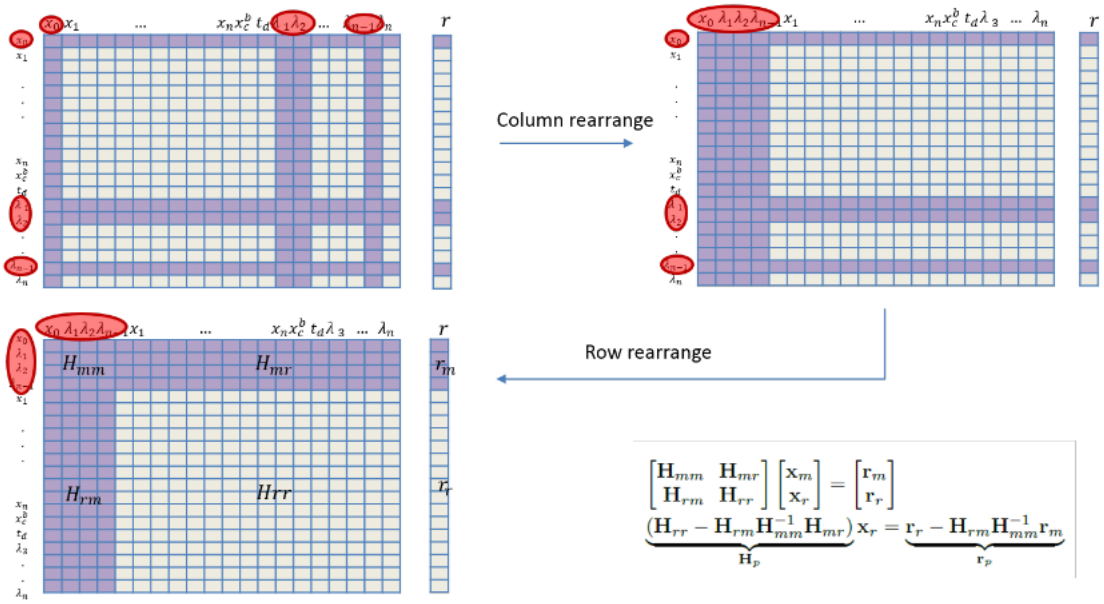
VINS-Mono 中为了处理一些悬停的 case，引入了一个 two-way marginalization：

1. `MARGIN_OLD`：如果次新帧是关键帧，则丢弃滑动窗口内最老的图像帧，同时对与该图像帧关联的约束项进行边缘化处理。这里需要注意的是，如果该关键帧是观察到某个地图点的第一帧，则需要把该地图点的深度转移到后面的图像帧中去。
2. `MARGIN_NEW`：如果次新帧不是关键帧，则丢弃当前帧的前一帧。因为判定当前帧不是关键帧的条件就是当前帧与前一帧视差很小，也就是说当前帧和前一帧很相似，这种情况下直接丢弃前一帧，然后用当前帧代替前一帧。为什么这里可以不对前一帧进行边缘化，而是直接丢弃，原因就是当前帧和前一帧很相似，因此当前帧与地图点之间的约束和前一帧与地图点之间的约束是很接近的，直接丢弃并不会造成整个约束关系丢失信息。这里需要注意的是，要把当前帧和前一帧之间的 IMU 预积分转换为当前帧和前二帧之间的 IMU 预积分。

在悬停等运动较小的情况下，会频繁的 `MARGIN_NEW`，这样也就保留了那些比较旧但是视差比较大的 pose。这种情况如果一直 `MARGIN_OLD` 的话，视觉约束不够强，状态估计会受 IMU 积分误差影响，具有较大的累积误差。

4.4.2 Schur Complement

Marginalization via Schur complement on information matrix



4.4.3 First Estimate Jacobin

5 重定位

5.1 Loop Detection

Vins-Mono 利用词袋 DBoW2 做 Keyframe Database 的构建和查询。在建立闭环检测的数据库时，关键帧的 Features 包括两部分：VIO 部分的 200 个强角点和 500 个 Fast 角点，然后描述子使用 BRIEF (因为旋转可观，匹配过程中对旋转有一定的适应性，所以不用使用 ORB)。

1. Describe features by BRIEF
   (a) Features that we use in the VIO (200, not enough for loop detection)
   (b) Extract new FAST features (500, only use for loop detection)
2. Query Bag-of-Word (DBoW2)
   (a) Return loop candidates

5.2 Feature Retrieval

在闭环检测成功之后，会得到回环候选帧，所以要在已知位姿的回环候选帧和滑窗内的匹配帧通过 **BRIEF 描述子匹配**，然后把回环帧加入到滑窗的优化当中，这时整个滑窗的状态量的维度是不发生变化的，因为回环帧的位姿是固定的。

1. Try to retrieve matches for features (200) that are used in the VIO
2. BRIEF descriptor match
3. Geometric check
   (a) 2D-2D: fundamental matrix test with RANSAC
   (b) 3D-3D: PnP test with RANSAC
   (c) At least 30 inliers

5.3 Tightly-Coupled Relocalization

6 全局位姿图优化

因为之前做的非线性优化本质只是在一个滑窗之内求解出了相机的位姿，而且在回环检测部分，利用固定位姿的回环帧只是纠正了滑窗内的相机位姿，并没有修正其他位姿 (或者说没有将回环发现的误差分配到整个相机的轨迹上)，缺少全局的一致性，所以要做一次全局的 Pose Graph。**全局的 Pose Graph 较之滑窗有一定的迟滞性，只有相机的 Pose 滑出滑窗的时候，Pose 才会被加到全局的 Pose Graph 当中**。

1. Adding Keyframes into the Pose Graph
   (a) Sequential edges from VIO
      i. Connected with 4 previous keyframes
   (b) Loop closure edges
      i. Only added when a keyframe is marginalized out from the sliding window VIO
      ii. Multi-constraint relocalization helps eliminating false loop closures
      iii. Huber norm for rejection of wrong loops
2. 4-DOF Pose Graph Optimization
   (a) Roll and pitch are observable from VIO
3. Pose Graph Management
4. Map Reuse
   (a) Save map at any time
   (b) Load map and re-localize with respect to it
   (c) Pose graph merging

7 Remarks on Monocular Visual-Inertial SLAM

1. Important factors
   (a) Access to raw camera data (especially for rolling shutter cameras)
   (b) Sensor synchronization and timestamps
   (c) Camera-IMU rotation
   (d) Estimator initialization
2. Not-so-important factors
   (a) Camera-IMU translation
   (b) Types of features (we use the simplest corner+KLT)
   (c) Quality of feature tracking (outlier is acceptable)
3. Failures –need more engineering treatment
   (a) Long range scenes (aerial vehicles)
   (b) Constant velocity (ground vehicle)
   (c) Pure rotation (augmented reality)
4. Be aware of computational power requirement

参考文献

[1] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[2] Joan Solà. Quaternion kinematics for the error-state kalman filter. *CoRR*, abs/1711.02508, 2017.

[3] Xiaobuyi. Vins-mono 代码分析总结. https://www.zybuluo.com/Xiaobuyi/note/866099, 2018.

[4] Shaojie Shen. Monocular visual-inertial slam slides. University Lecture, 2018.

[5] Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669. IEEE, 2018.