

VINS 论文推导及代码解析

崔华坤 2019.3.17

一、总体框架	3
1.1 图像和 IMU 预处理	3
1.2 初始化	3
1.3 后端滑窗优化	3
1.4 闭环检测和优化	4
二、IMU 预积分	4
2.1 当前时刻 PVQ 的连续形式	4
2.2 当前时刻 PVQ 的中值法离散形式	5
2.3 两帧之间 PVQ 增量的连续形式	5
2.4 两帧之间 PVQ 增量的欧拉法离散形式	6
2.5 两帧之间 PVQ 增量的中值法离散形式	6
2.6 连续形式下 PVQ 增量的误差、协方差及 Jacobian	7
2.7 离散形式的 PVQ 增量误差分析	8
2.8 离散形式的 PVQ 增量误差的 Jacobian 和协方差	9
三、后端非线性优化	9
3.1 状态向量	9
3.2 目标函数	9
3.3 IMU 约束	10
3.4 视觉约束	12
四、前端视觉处理	13
4.1 特征点检测	13
4.2 特征点跟踪	14
五、初始化	14
5.1 relativePose	16
5.2 GlobalSFM.construct	16

5.3 solvePnP	17
5.4 visualInitialAlign	18
六、边缘化 Marginalization 和 FEJ	20
6.1 边缘化和 Schur 补公式	20
6.3 一个边缘化的例子	21
6.3 VINS 两种边缘化策略	24
6.4 First Estimate Jacobian(FEJ)	28
七、闭环检测和优化	29
7.1 闭环检测	30
7.2 快速重定位	31
7.3 闭环关键帧数据库	31
7.4 闭环优化	32
7.5 程序逻辑	32
八、其他	34
8.1 选 KF 策略	34
8.2 后端优化后的变量更新	34
8.3 丢失后多地图融合	34
8.4 小滑窗 PnP 优化	35
九、参考文献	36
十、附录	37
10.1 IMU 状态积分公式推导	37
10.2 连续形式 IMU 误差动力学方程推导	38
10.3 IMU 离散误差动力学方程推导	39
10.4 IMU 角度误差对 k 时刻角度误差的 Jacobian 推导	41
10.5 IMU 角度误差对 k 时刻 bias 的 Jacobian 推导	42
10.6 IMU 角度误差对 $k+1$ 时刻的角度误差 Jacobian 推导	42
10.7 视觉误差项推导	42
10.8 视觉误差项的 Jacobian 推导	43

一、总体框架

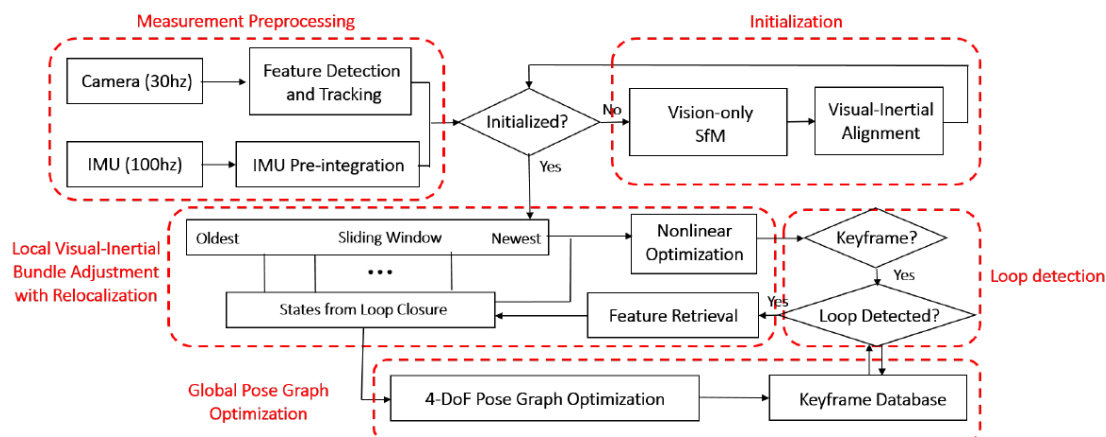


图 1 VINS 框架

VINS^[1]的功能模块可包括五个部分：数据预处理、初始化、后端非线性优化、闭环检测及闭环优化。代码中主要开启了四个线程，分别是：前端图像跟踪、后端非线性优化（其中初始化和 IMU 预积分在这个线程中）、闭环检测、闭环优化。

各个功能模块的作用主要有：

1.1 图像和 IMU 预处理

- 图像：提取图像 Harris 角点，利用金字塔光流跟踪相邻帧，通过 RANSAC 去除异常点，最后将跟踪到的特征点 push 到图像队列中，并通知后端进行处理。
- IMU：将 IMU 数据进行积分，得到当前时刻的位置、速度和旋转（PVQ），同时计算在后端优化中将用到的相邻帧的预积分增量，及预积分误差的 Jacobian 矩阵和协方差项。

1.2 初始化

首先，利用 SFM 进行纯视觉估计滑窗内所有帧的位姿及 3D 点逆深度，最后与 IMU 预积分进行对齐求解初始化参数。

1.3 后端滑窗优化

将视觉约束、IMU 约束和闭环约束放在一个大的目标函数中进行非线性优化，求解滑窗内所有帧的 PVQ、bias 等。

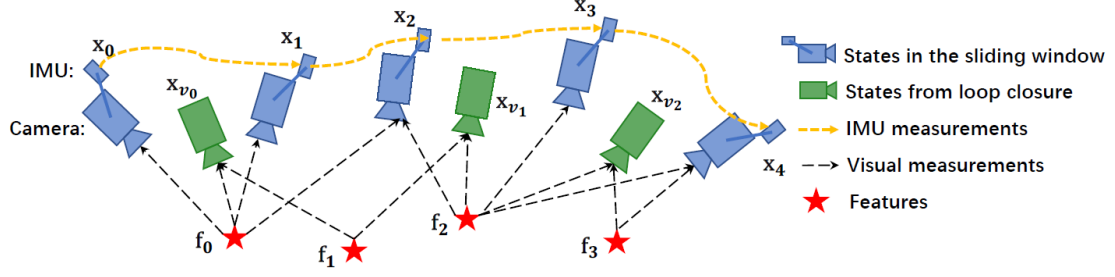


图 2 滑窗优化示意图

1.4 闭环检测和优化

利用 DBoW 进行闭环检测，当检测成功后进行重定位，最后对整个相机轨迹进行闭环优化。

二、IMU 预积分

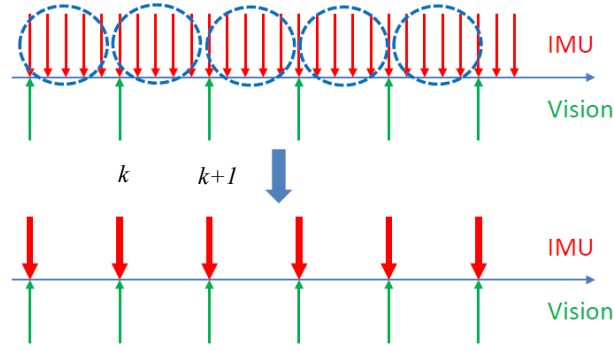


图 3 IMU 预积分示意图

2.1 当前时刻 PVQ 的连续形式

将第 k 帧和第 $k+1$ 帧之间的所有 IMU 进行积分，可得第 $k+1$ 帧的位置、速度和旋转 (PVQ)，作为视觉估计的初始值，这里的旋转采用的四元数。

$$\begin{aligned}
 p_{b_{k+1}}^w &= p_{b_k}^w + v_{b_k}^w \Delta t_k + \iint_{t \in [k, k+1]} [R_t^w (\hat{a}_t - b_{a_t}) - g^w] dt^2 \\
 v_{b_{k+1}}^w &= v_{b_k}^w + \int_{t \in [k, k+1]} [R_t^w (\hat{a}_t - b_{a_t}) - g^w] dt \\
 q_{b_{k+1}}^w &= q_{b_k}^w \otimes \int_{t \in [k, k+1]} \frac{1}{2} \Omega(\hat{\omega}_t - b_{\omega_t}) q_t^{b_k} dt
 \end{aligned} \tag{1}$$

其中， \hat{a}_t 和 $\hat{\omega}_t$ 为 IMU 测量的加速度和角速度，是在 Body 自身坐标系，world 坐标系是 IMU 所在的惯导系，上式的旋转公式推导可参考附录 10.1。

2.2 当前时刻 PVQ 的中值法离散形式

公式(1)给出的是连续时刻的相机当前 PVQ 的迭代公式，为了跟代码一致，下面给出基于中值法的公式，即从第 i 个 IMU 时刻到第 $i+1$ 个 IMU 时刻的积分过程，这与 Estimator::processIMU() 函数中的 $Ps[j]$ 、 $Rs[j]$ 和 $Vs[j]$ 是一致的（代码中的 j 时刻即为此处的 $i+1$ ），IMU 积分出来的第 j 时刻的物理量可以作为第 j 帧图像的初始值。

$$\begin{aligned} p_{b_{i+1}}^w &= p_{b_i}^w + v_{b_i}^w \delta t + \frac{1}{2} \bar{a}_i \delta t^2 \\ v_{b_{i+1}}^w &= v_{b_i}^w + \bar{a}_i \delta t \\ q_{b_{i+1}}^w &= q_{b_i}^w \otimes \left[\frac{1}{2} \bar{\omega}_i \delta t \right] \end{aligned} \quad (2)$$

其中：

$$\begin{aligned} \bar{a}_i &= \frac{1}{2} [q_i(\hat{a}_i - b_{a_i}) - g^w + q_{i+1}(\hat{a}_{i+1} - b_{a_i}) - g^w] \\ \bar{\omega}_i &= \frac{1}{2} (\hat{\omega}_i + \hat{\omega}_{i+1}) - b_{\omega_i} \end{aligned} \quad (3)$$

2.3 两帧之间 PVQ 增量的连续形式

通过观察公式(1)可知，IMU 的预积分需要依赖与第 k 帧的 v 和 R ，当我们在后端进行非线性优化时，需要迭代更新第 k 帧的 v 和 R ，这将导致我们需要根据每次迭代后值重新进行积分，这将非常耗时。因此，我们考虑将优化变量从第 k 帧到第 $k+1$ 帧的 IMU 预积分项中分离开来，通过对公式（1）左右两侧各乘 $R_w^{b_k}$ ，可化简为：

$$\begin{aligned} R_w^{b_k} p_{b_{k+1}}^w &= R_w^{b_k} \left(p_{b_k}^w + v_{b_k}^w \Delta t_k - \frac{1}{2} g^w \Delta t_k^2 \right) + \alpha_{b_{k+1}}^{b_k} \\ R_w^{b_k} v_{b_{k+1}}^w &= R_w^{b_k} (v_{b_k}^w - g^w \Delta t_k) + \beta_{b_{k+1}}^{b_k} \\ q_w^{b_k} \otimes q_{b_{k+1}}^w &= \gamma_{b_{k+1}}^{b_k} \end{aligned} \quad (4)$$

其中：

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \iint_{t \in [k, k+1]} [R_t^{b_k} (\hat{a}_t - b_{a_t})] dt^2 \\ \beta_{b_{k+1}}^{b_k} &= \int_{t \in [k, k+1]} [R_t^{b_k} (\hat{a}_t - b_{a_t})] dt \\ \gamma_{b_{k+1}}^{b_k} &= \int_{t \in [k, k+1]} \frac{1}{2} \Omega(\hat{\omega}_t - b_{\omega_t}) \gamma_t^{b_k} dt \end{aligned} \quad (5)$$

这样我们就得到了连续时刻的 IMU 预积分公式，可以发现，上式得到的 IMU 预积分的值只与不同时刻的 \hat{a}_t 和 $\hat{\omega}_t$ 相关。

这里我们需要重新讨论下公式(5)的预积分公式, 以 $\hat{\alpha}_{b_{k+1}}^{b_k}$ 为例, 我们发现它是与 IMU 的 bias 相关的, 而 bias 也是我们需要优化的变量, 这将导致的问题是, 当每次迭代时, 我们得到一个新的 bias, 又得根据公式(5)重新对第 k 帧和第 k+1 帧之间的 IMU 预积分, 非常耗时。这里假设预积分的变化量与 bias 是线性关系, 可以写成:

$$\begin{aligned}\alpha_{b_{k+1}}^{b_k} &\approx \hat{\alpha}_{b_{k+1}}^{b_k} + J_{b_a}^\alpha \delta b_a + J_{b_\omega}^\alpha \delta b_\omega \\ \beta_{b_{k+1}}^{b_k} &\approx \hat{\beta}_{b_{k+1}}^{b_k} + J_{b_a}^\beta \delta b_a + J_{b_\omega}^\beta \delta b_\omega \\ \gamma_{b_{k+1}}^{b_k} &\approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \left[\frac{1}{2} J_{b_\omega}^\gamma \delta b_\omega \right]\end{aligned}\tag{6}$$

2.4 两帧之间 PVQ 增量的欧拉法离散形式

下面给出离散时刻的 IMU 预积分公式, 首先按照论文中采用的欧拉法, 给出第 i 个 IMU 时刻与第 i+1 个 IMU 时刻的变量关系为:

$$\begin{aligned}\hat{\alpha}_{i+1}^{b_k} &= \hat{\alpha}_i^{b_k} + \hat{\beta}_i^{b_k} \delta t + \frac{1}{2} R(\hat{\gamma}_i^{b_k})(\hat{a}_i - b_{a_i}) \delta t^2 \\ \hat{\beta}_{i+1}^{b_k} &= \hat{\beta}_i^{b_k} + R(\hat{\gamma}_i^{b_k})(\hat{a}_i - b_{a_i}) \delta t \\ \hat{\gamma}_{i+1}^{b_k} &= \hat{\gamma}_i^{b_k} \otimes \hat{\gamma}_{i+1}^i = \hat{\gamma}_i^{b_k} \otimes \left[\frac{1}{2} (\hat{\omega}_i - b_{\omega_i}) \delta t \right]\end{aligned}\tag{7}$$

2.5 两帧之间 PVQ 增量的中值法离散形式

下面给出代码中采用的基于中值法的 IMU 预积分公式, 这与 Estimator::processIMU() 函数中的 IntegrationBase::push_back() 上是一致的。注意这里跟公式(2)是不一样的, 这里积分出来的是前后两帧之间的 IMU 增量信息, 而公式(2)给出的当前帧时刻的物理量信息。

$$\begin{aligned}\hat{\alpha}_{i+1}^{b_k} &= \hat{\alpha}_i^{b_k} + \hat{\beta}_i^{b_k} \delta t + \frac{1}{2} \bar{\tilde{a}}_i' \delta t^2 \\ \hat{\beta}_{i+1}^{b_k} &= \hat{\beta}_i^{b_k} + \bar{\tilde{a}}_i' \delta t \\ \hat{\gamma}_{i+1}^{b_k} &= \hat{\gamma}_i^{b_k} \otimes \hat{\gamma}_{i+1}^i = \hat{\gamma}_i^{b_k} \otimes \left[\frac{1}{2} \bar{\tilde{\omega}}_i' \delta t \right]\end{aligned}\tag{8}$$

其中,

$$\bar{\tilde{a}}_i' = \frac{1}{2} [q_i(\hat{a}_i - b_{a_i}) + q_{i+1}(\hat{a}_{i+1} - b_{a_i})]$$

$$\bar{\omega}_i' = \frac{1}{2}(\bar{\omega}_i + \bar{\omega}_{i+1}) - b_{\omega_i} \quad (9)$$

2.6 连续形式下 PVQ 增量的误差、协方差及 Jacobian

IMU 在每一个时刻积分出来的值是有误差的，下面我们对误差进行分析。首先我们直接给出在 t 时刻误差项的导数为：

$$\begin{bmatrix} \delta \dot{\alpha}_t^{b_k} \\ \delta \dot{\beta}_t^{b_k} \\ \delta \dot{\theta}_t^{b_k} \\ \delta \dot{b}_{a_t} \\ \delta \dot{b}_{w_t} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -R_t^{b_k}(\hat{a}_t - b_{a_t})^\wedge - R_t^{b_k} & 0 & 0 \\ 0 & 0 & -(\hat{\omega}_t - b_{\omega_t})^\wedge & 0 & -I \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta \alpha_t^{b_k} \\ \delta \beta_t^{b_k} \\ \delta \theta_t^{b_k} \\ \delta b_{a_t} \\ \delta b_{w_t} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -R_t^{b_k} & 0 & 0 & 0 \\ 0 & -I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} n_a \\ n_w \\ n_{b_a} \\ n_{b_w} \end{bmatrix} \quad (10)$$

$$= F_t \delta z_t^{b_k} + G_t n_t$$

其中： $F_t^{15 \times 15}$ ， $G_t^{15 \times 12}$ ， $\delta z_t^{b_k^{15 \times 1}}$ ， $n_t^{12 \times 1}$ ，上式推导可参考附录 10.2。下面我们讨论它的作用，将其可以简写为：

$$\delta \dot{z}_t^{b_k} = F_t \delta z_t^{b_k} + G_t n_t$$

根据导数定义可知： $\delta z_t^{b_k} = \lim_{\delta t \rightarrow 0} \frac{\delta z_{t+\delta t}^{b_k} - \delta z_t^{b_k}}{\delta t}$

$$\begin{aligned} \delta z_{t+\delta t}^{b_k} &= \delta z_t^{b_k} + \delta \dot{z}_t^{b_k} \delta t = (I + F_t \delta t) \delta z_t^{b_k} + (G_t \delta t) n_t \\ &= F \delta z_t^{b_k} + V n_t \end{aligned} \quad (11)$$

为了简化下一节中对离散形式的分析，上式中我们令： $F = I + F_t \delta t$ ， $V = G_t \delta t$ 。

这里我们对公式(11)的 IMU 误差运动方程再说明，将上式和 EKF 对比可知，上式恰好给出了如 EKF 一般对非线性系统线性化的过程，这里的意义是表示下一个时刻的 IMU 测量误差与上一个时刻的成线性关系，这样我们根据当前时刻的值，可以预测出下一个时刻的均值和协方差，而公式(11)给出的是均值预测，协方差预测公式如下：

$$P_{t+\delta t}^{b_k} = (I + F_t \delta t) P_t^{b_k} (I + F_t \delta t)^T + (G_t \delta t) Q (G_t \delta t)^T \quad (12)$$

上式给出了协方差的迭代公式，初始值 $P_{b_k}^{b_k} = 0$ 。其中， Q 为表示噪声项的对角协方差矩阵：

$$Q^{12 \times 12} = \begin{bmatrix} \sigma_a^2 & 0 & 0 & 0 \\ 0 & \sigma_w^2 & 0 & 0 \\ 0 & 0 & \sigma_{b_a}^2 & 0 \\ 0 & 0 & 0 & \sigma_{b_w}^2 \end{bmatrix} \quad (13)$$

另外根据(11)式可获得误差项的 Jacobian 的迭代公式：

$$J_{t+\delta t} = (I + F_t \delta t) J_t \quad (14)$$

其中 Jacobian 的初始值为 $J_{b_k} = I$ 。

2.7 离散形式的 PVQ 增量误差分析

我们首先直接给出 PVQ 增量误差在离散形式下的矩阵形式，为了与代码一致，我们修改下变量顺序，这和代码中 `midPointIntegration()` 函数是一致的。（但不知为何计算的 V 中与前四个噪声项相关的差个负号？）

$$\begin{aligned} \begin{bmatrix} \delta\alpha_{k+1} \\ \delta\theta_{k+1} \\ \delta\beta_{k+1} \\ \delta b_{a_{k+1}} \\ \delta b_{w_{k+1}} \end{bmatrix} &= \begin{bmatrix} I & f_{01} & \delta t & f_{03} & f_{04} \\ 0 & f_{11} & 0 & 0 & -\delta t \\ 0 & f_{21} & I & f_{23} & f_{24} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \delta\alpha_k \\ \delta\theta_k \\ \delta\beta_k \\ \delta b_{a_k} \\ \delta b_{w_k} \end{bmatrix} \\ &+ \begin{bmatrix} v_{00} & v_{01} & v_{02} & v_{03} & 0 & 0 \\ 0 & \frac{-\delta t}{2} & 0 & \frac{-\delta t}{2} & 0 & 0 \\ -\frac{R_k \delta t}{2} & v_{21} & -\frac{R_{k+1} \delta t}{2} & v_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta t \end{bmatrix} \begin{bmatrix} n_{a_k} \\ n_{w_k} \\ n_{a_{k+1}} \\ n_{w_{k+1}} \\ n_{b_a} \\ n_{b_w} \end{bmatrix} \end{aligned} \quad (15)$$

其中，推导可参考附录 10.3:

$$f_{01} = \frac{\delta t}{2} f_{21} = -\frac{1}{4} R_k (\hat{a}_k - b_{a_k})^\wedge \delta t^2 - \frac{1}{4} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \delta t^2$$

$$f_{03} = -\frac{1}{4} (R_k + R_{k+1}) \delta t^2$$

$$f_{04} = \frac{\delta t}{2} f_{24} = \frac{1}{4} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^3$$

$$f_{11} = I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t$$

$$f_{21} = -\frac{1}{2} R_k (\hat{a}_k - b_{a_k})^\wedge \delta t - \frac{1}{2} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \delta t$$

$$f_{23} = -\frac{1}{2} (R_k + R_{k+1}) \delta t$$

$$f_{24} = \frac{1}{2} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^2$$

$$v_{00} = -\frac{1}{4} R_k \delta t^2$$

$$v_{01} = v_{03} = \frac{\delta t}{2} v_{21} = \frac{1}{4} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^2 \frac{\delta t}{2}$$

$$v_{02} = -\frac{1}{4} R_{k+1} \delta t^2$$

$$v_{21} = v_{23} = \frac{1}{4} R_{k+1} (\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^2$$

2.8 离散形式的 PVQ 增量误差的 Jacobian 和协方差

将公式(15)简写为:

$$\delta Z_{k+1}^{15 \times 1} = F^{15 \times 15} \delta Z_k^{15 \times 1} + V^{15 \times 18} Q^{18 \times 1}$$

则 Jacobian 的迭代公式为:

$$J_{k+1}^{15 \times 15} = F J_k \quad (16)$$

其中, Jacobian 的初始值为 $J_k = I$ 。这里计算出来的 J_{k+1} 只是为了给后面提供对 bias 的 Jacobian。

协方差的迭代公式为:

$$P_{k+1}^{15 \times 15} = F P_k F^T + V Q V^T \quad (17)$$

其中, 初始值 $P_k = 0$ 。 Q 为表示噪声项的对角协方差矩阵:

$$Q^{18 \times 18} = \begin{bmatrix} \sigma_a^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_w^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_a^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_w^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{b_a}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{b_w}^2 \end{bmatrix} \quad (18)$$

三、后端非线性优化

3.1 状态向量

状态向量共包括滑动窗口内的 $n+1$ 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)、Camera 到 IMU 的外参、 $m+1$ 个 3D 点的逆深度:

$$X = [x_0, x_1, \dots, x_n, x_c^b, \lambda_0, \lambda_1, \dots, \lambda_m]$$

$$x_k = [p_{b_k}^w, v_{b_k}^w, q_{b_k}^w, b_a, b_g]$$

$$x_c^b = [p_c^b, q_c^b]$$

3.2 目标函数

$$\min_X \left\{ \|r_p - J_p X\|^2 + \sum_{k \in B} \left\| r_B \left(\hat{z}_{b_{k+1}}^{b_k}, X \right) \right\|_{P_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j) \in C} \left\| r_C \left(\hat{z}_l^{c_j}, X \right) \right\|_{P_l^{c_j}}^2 \right\} \quad (19)$$

其中三个残差项即误差项分别为边缘化的先验信息、IMU 测量残差、视觉的重投影残差。三种残差都是用马氏距离表示。

根据《十四讲》中高斯-牛顿法, 若要计算目标函数的最小值, 可以理解为, 当优化变量有一个增量后, 目标函数值最小, 以 IMU 残差为例, 可写成如下所示:

$$\min_{\delta X} \left\| r_B \left(\hat{z}_{b_{k+1}}^{b_k}, X + \Delta X \right) \right\|_{P_{b_{k+1}}^{b_k}}^2 = \min_{\delta X} \left\| r_B \left(\hat{z}_{b_{k+1}}^{b_k}, X \right) + J_{b_{k+1}}^{b_k} \Delta X \right\|_{P_{b_{k+1}}^{b_k}}^2$$

其中 $J_{b_{k+1}}^{b_k}$ 为误差项 r_B 关于所有状态向量（即优化变量） X 的 Jacobian，将上式展开并令关于 ΔX 的导数为 0，可得增量 ΔX 的计算公式：

$$J_{b_{k+1}}^{b_k T} P_{b_{k+1}}^{b_k -1} J_{b_{k+1}}^{b_k} \Delta X = -J_{b_{k+1}}^{b_k T} P_{b_{k+1}}^{b_k -1} r_B$$

那么，对于公式(19)的整体目标函数的整体增量方程可写成：

$$\begin{aligned} & \left(H_p + \sum J_{b_{k+1}}^{b_k T} P_{b_{k+1}}^{b_k -1} J_{b_{k+1}}^{b_k} + \sum J_l^{c_j T} P_l^{c_j -1} J_l^{c_j} \right) \Delta X \\ & = b_p + \sum J_{b_{k+1}}^{b_k T} P_{b_{k+1}}^{b_k -1} r_B + \sum J_l^{c_j T} P_l^{c_j -1} r_c \end{aligned}$$

上式中， $P_{b_{k+1}}^{b_k}$ 为 IMU 预积分噪声项的协方差， $P_l^{c_j}$ 为 visual 观测的噪声协方差。当 IMU 的噪声协方差 $P_{b_{k+1}}^{b_k}$ 越大时，其信息矩阵 $P_{b_{k+1}}^{b_k -1}$ 将越小，意味着该 IMU 观测越不可信，换句话说，因 IMU 噪声较大，越不可信 IMU 预积分数据，而更加相信 visual 观测。注意，这里的 IMU 和 visual 协方差的绝对值没有意义，因为考虑得是两者的相对性。

可将上式继续简化为：

$$(\Lambda_p + \Lambda_B + \Lambda_C) \Delta X = b_p + b_B + b_C$$

其中， Λ_p 、 Λ_B 和 Λ_C 为 Hessian 矩阵，上述方程称之为增量方程。

值得说明的是，上面中的 Jacobian 虽然是误差项 r 关于状态向量 X 的一阶导，但在具体求解时，通常采用扰动的方式进行计算，如下所示，注意其中的 δX 是状态向量的微小扰动，并非上面求解的增量 ΔX ：

$$J = \frac{\partial \gamma}{\partial X} = \lim_{\delta X \rightarrow 0} \frac{\gamma(X \oplus \delta X) - \gamma(X)}{\delta X}$$

3.3 IMU 约束

1) 残差：两帧之间的 PVQ 和 bias 的变化量的差

$$r_B^{15 \times 1} \left(\hat{z}_{b_{k+1}}^{b_k}, X \right) = \begin{bmatrix} \delta \alpha_{b_{k+1}}^{b_k} \\ \delta \theta_{b_{k+1}}^{b_k} \\ \delta \beta_{b_{k+1}}^{b_k} \\ \delta b_a \\ \delta b_g \end{bmatrix} = \begin{bmatrix} R_w^{b_k} \left(p_{b_{k+1}}^w - p_{b_k}^w - v_{b_k}^w \Delta t_k + \frac{1}{2} g^w \Delta t_k^2 \right) - a_{b_{k+1}}^{b_k} \\ 2 \left[\gamma_{b_{k+1}}^{b_k -1} \otimes q_{b_k}^{w -1} \otimes q_{b_{k+1}}^w \right]_{xyz} \\ R_w^{b_k} \left(v_{b_{k+1}}^w - v_{b_k}^w + g^w \Delta t_k \right) - \beta_{b_{k+1}}^{b_k} \\ b_{a_{b_{k+1}}} - b_{a_{b_k}} \\ b_{\omega_{b_{k+1}}} - b_{\omega_{b_k}} \end{bmatrix} \quad (20)$$

其中各增量关于 bias 的 Jacobian 可从公式(16)的大 Jacobian 中的相应位置获得。上面与

代码中 `IntegrationBase::evaluate()` 对应。

2) 优化变量:

$$[p_{b_k}^w, q_{b_k}^w], [v_{b_k}^w, b_{a_k}, b_{\omega_k}], [p_{b_{k+1}}^w, q_{b_{k+1}}^w], [v_{b_{k+1}}^w, b_{a_{k+1}}, b_{\omega_{k+1}}]$$

3) Jacobian:

计算 Jacobian 时，残差对应的求偏导对象为上面的优化变量，但是计算时采用扰动方式计算，即扰动为 $[\delta p_{b_k}^w, \delta \theta_{b_k}^w], [\delta v_{b_k}^w, \delta b_{a_k}, \delta b_{\omega_k}], [\delta p_{b_{k+1}}^w, \delta \theta_{b_{k+1}}^w], [\delta v_{b_{k+1}}^w, \delta b_{a_{k+1}}, \delta b_{\omega_{k+1}}]$ 。

$$J[0]^{15 \times 7} = \left[\frac{\partial r_B}{\partial p_{b_k}^w}, \frac{\partial r_B}{\partial q_{b_k}^w} \right] = \begin{bmatrix} -R_w^{b_k} & \left[R_w^{b_k} (p_{b_{k+1}}^w - p_{b_k}^w - v_{b_k}^w \Delta t_k + \frac{1}{2} g^w \Delta t_k^2) \right]^\wedge \\ 0 & -\mathcal{L}[q_{b_{k+1}}^w]^{-1} \otimes q_{b_k}^w \mathcal{R}[\gamma_{b_{k+1}}^{b_k}] \\ 0 & \left[R_w^{b_k} (v_{b_{k+1}}^w - v_{b_k}^w + g^w \Delta t_k) \right]^\wedge \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (21)$$

推导可参考附录 10.4。

$$J[1]^{15 \times 9} = \left[\frac{\partial r_B}{\partial v_{b_k}^w}, \frac{\partial r_B}{\partial b_{a_k}}, \frac{\partial r_B}{\partial b_{\omega_k}} \right] = \begin{bmatrix} -R_w^{b_k} \Delta t & -J_{b_a}^\alpha & -J_{b_\omega}^\alpha \\ 0 & 0 & -\mathcal{L}[q_{b_{k+1}}^w]^{-1} \otimes q_{b_k}^w \otimes \gamma_{b_{k+1}}^{b_k} J_{b_\omega}^\gamma \\ -R_w^{b_k} & -J_{b_a}^\beta & -J_{b_\omega}^\beta \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix} \quad (22)$$

推导可参考附录 10.5。

$$J[2]^{15 \times 7} = \left[\frac{\partial r_B}{\partial p_{b_{k+1}}^w}, \frac{\partial r_B}{\partial q_{b_{k+1}}^w} \right] = \begin{bmatrix} R_w^{b_k} & 0 \\ 0 & \mathcal{L}[\gamma_{b_{k+1}}^{b_k}]^{-1} \otimes q_{b_k}^w]^{-1} \otimes q_{b_{k+1}}^w \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (23)$$

推导可参考附录 10.6

$$J[3]^{15 \times 9} = \left[\frac{\partial r_B}{\partial v_{b_{k+1}}^w}, \frac{\partial r_B}{\partial b_{a_{k+1}}}, \frac{\partial r_B}{\partial b_{\omega_{k+1}}} \right] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ R_w^{b_k} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \quad (24)$$

上面公式在代码中对应: `class IMUFactor : public ceres::SizedCostFunction<15, 7, 9, 7, 9>`

对于 Evaluate 输入 `double const *const *parameters, parameters[0], parameters[1], parameters[2], parameters[3]` 分别对应 4 个输入参数，它们的长度依次是 7,9,7,9，分别对应 4 个优化变量的参数块。

代码 `IMUFactor::Evaluate()` 中 `residual` 还乘以一个信息矩阵 `sqrt_info`，这是因为真正的优化项其实是 Mahalanobis 距离: $d = r^T P^{-1} r$ ， P 是协方差，又因为 Ceres 只接受最小二乘优化，也就是 $\min(e^T e)$ ，所以把 P^{-1} 做 LLT 分解，即 $LL^T = P^{-1}$ ，则有：

$$d = r^T L L^T r = (L^T r)^T (L^T r)$$

令 $r' = L^T r$ 作为新的优化误差, 这样就能用 Ceres 求解了。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。若写成“`sqrt_info.setIdentity()`”相当于不加权。

4) 协方差

上面提到的 IMU 协方差 P 为公式 (17) 推导的 IMU 预积分中迭代出来的 IMU 增量误差的协方差。

3.4 视觉约束

1) 残差

视觉残差是重投影误差, 对于第 l 个路标点 P , 将 P 从第一次观看到它的第 i 个相机坐标系, 转换到当前的第 j 个相机坐标系下的像素坐标, 可定义视觉误差项为:

$$r_c(\hat{z}_l^{c_j}, X) = \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \end{bmatrix} \left(\frac{P_l^{c_j}}{\|P_l^{c_j}\|} - \bar{P}_l^{c_j} \right) \quad (25)$$

其中, $\bar{P}_l^{c_j}$ 为第 l 个路标点在第 j 个相机归一化相机坐标系中的观察到的坐标:

$$\bar{P}_l^{c_j} = \pi_c^{-1} \left(\begin{bmatrix} \hat{u}_l^{c_j} \\ \hat{v}_l^{c_j} \end{bmatrix} \right) \quad (26)$$

另外, $P_l^{c_j}$ 是估计第 l 个路标点在第 j 个相机归一化相机坐标系中的可能坐标, 推导可参附录 10.7:

$$P_l^{c_j} = R_b^c \left\{ R_w^{b_j} \left[R_c^{b_j} \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \quad (27)$$

因为视觉残差的自由度为 2, 因此将视觉残差投影到正切平面上, b_1, b_2 为正切平面上的任意两个正交基。

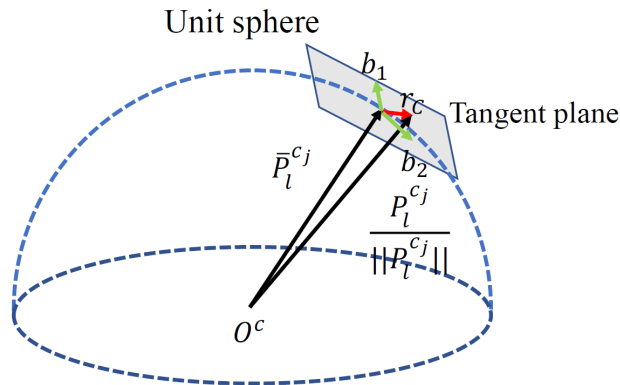


图 4 在单位球面上的视觉残差

2) 优化变量

$$[p_{b_i}^w, q_{b_i}^w], [p_{b_j}^w, q_{b_j}^w], [p_c^b, q_c^b], \lambda_l.$$

3) Jacobian

根据视觉残差公式，我们可以得到相对于各优化变量的 Jacobian，推导可参考附录 10.7:

$$\begin{aligned} J[0]^{3 \times 7} &= \left[\frac{\partial r_c}{\partial p_{b_i}^w}, \frac{\partial r_c}{\partial q_{b_i}^w} \right] = \left[R_b^c R_w^{b_j} \quad -R_b^c R_w^{b_j} R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right)^\wedge \right] \\ J[1]^{3 \times 7} &= \left[\frac{\partial r_c}{\partial p_{b_j}^w}, \frac{\partial r_c}{\partial q_{b_j}^w} \right] = \left[-R_b^c R_w^{b_j} \quad R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] \right\}^\wedge \right] \\ J[2]^{3 \times 7} &= \left[\frac{\partial r_c}{\partial p_c^b}, \frac{\partial r_c}{\partial q_c^b} \right] \\ &= \left[\begin{array}{c} R_b^c (R_w^{b_j} R_{b_i}^w - I_{3 \times 3}) \\ -R_b^c R_w^{b_j} R_{b_i}^w R_c^b \left(\frac{\bar{P}_l^{c_i}}{\lambda_l} \right)^\wedge + \left(R_b^c R_w^{b_j} R_{b_i}^w R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} \right)^\wedge + \left\{ R_b^c \left[R_w^{b_j} (R_{b_i}^w p_c^b + p_{b_i}^w - p_{b_j}^w) - p_c^b \right] \right\}^\wedge \end{array} \right]^T \\ J[3]^{3 \times 1} &= \frac{\partial r_c}{\partial \lambda_l} = -R_b^c R_w^{b_j} R_{b_i}^w R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l^2} \end{aligned} \quad (28)$$

4) 协方差

视觉约束的噪声协方差与标定相机内参时的重投影误差，也就是偏离几个像素有关，代码对应为 `ProjectionTdfactor::sqrt_info`，这里取的 1.5 个像素，对应到归一化相机平面上的协方差矩阵需除以焦距 f ，则信息矩阵等于协方差矩阵的逆，为：

$$\Omega_{vis} = \Sigma_{vis}^{-1} = \left(\frac{1.5}{f} I_{2 \times 2} \right)^{-1} = \frac{f}{1.5} I_{2 \times 2}$$

5) 如果考虑 Cam 和 IMU 的时间戳偏移量 td ，则需要增加优化变量，后续再补。

四、前端视觉处理

4.1 特征点检测

Frame 1: `goodFeaturesToTrack` 检测 MAX_CNT 个特征点，设置 `forw_pts` 如下：

表 1 第一帧的特征点

ids	forw_pts	track_cnt
4	[600,400]	1
3	[500,300]	1
2	[400,200]	1
1	[300,100]	1

0	[100,50]	1
---	----------	---

4.2 特征点跟踪

Frame 2: calcOpticalFlowPyrLK 跟踪，将跟踪失败的点删除，跟踪成功的点点跟踪计数 +1，并调用 goodFeaturesToTrack 检测出 MAX_CNT - forw_pts.size()个特征点，并添加到 forw_pts 中，并调用 updateID 更新 ids，最后得到的 forw_pts 如下：

表 2 第二帧的特征点

ids	forw_pts	track_cnt
6	[200,150]	1
5	[100,100]	1
4	[580,400]	2
1	[280,100]	2
0	[700,50]	2

代码 FeatureTracker::undistortedPoints()中 cur_un_pts 为归一化相机坐标系下的坐标，pts_velocity 为当前帧相对前一帧特征点沿 x,y 方向的像素移动速度。

五、初始化

初始化采用视觉和 IMU 的松耦合方案，首先用 SFM 求解滑窗内所有帧的位姿，和所有路标点的 3D 位置，然后跟 IMU 预积分的值对齐，求解重力方向、尺度因子、陀螺仪 bias 及每一帧对应的速度。初始化的流程图如下所示：

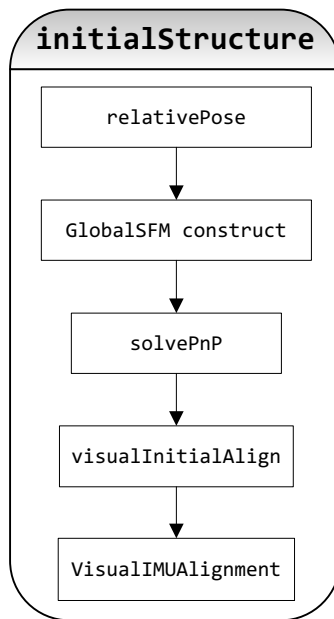
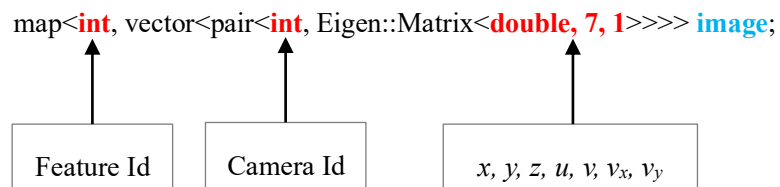


图 5 初始化流程图

首先，我们对代码中几个易混变量进行说明：

- 1) 传到 processImage 的 image 表示 当前帧跟踪到上一帧中的特征点集合，也就是当前帧观测到的所有的路标点（不包括在当前帧新提取的点），如表 2 中 track_cnt ≥ 2 的 feature，image 类型为：



- 2) FeatureManager 中几个变量：

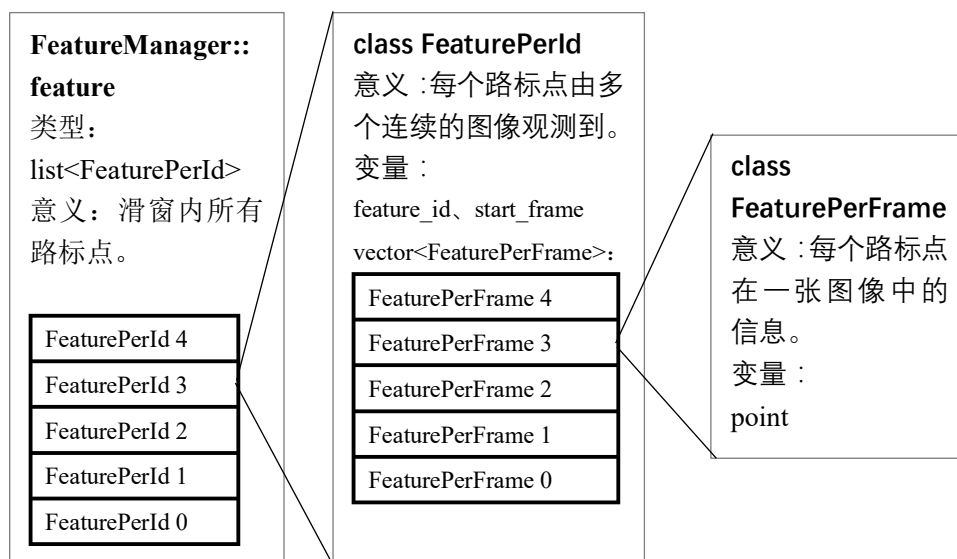


图 6 FeatureManager 中的几个易混变量

5.1 relativePose

在滑窗内寻找与当前帧的匹配特征点数较多的关键帧作为参考帧，并通过求基础矩阵 `cv::findFundamentalMat` 计算出当前帧到参考帧的 `T`；

5.2 GlobalSFM.construct

首先，将图 4 中 feature 队列，放到 `vector<SFMFeature> sfm_f` 中，其中 `SFMFeature` 中的 `observation` 存放的是观测到该路标点的 `FrameId` 及特征点坐标，如下所示：

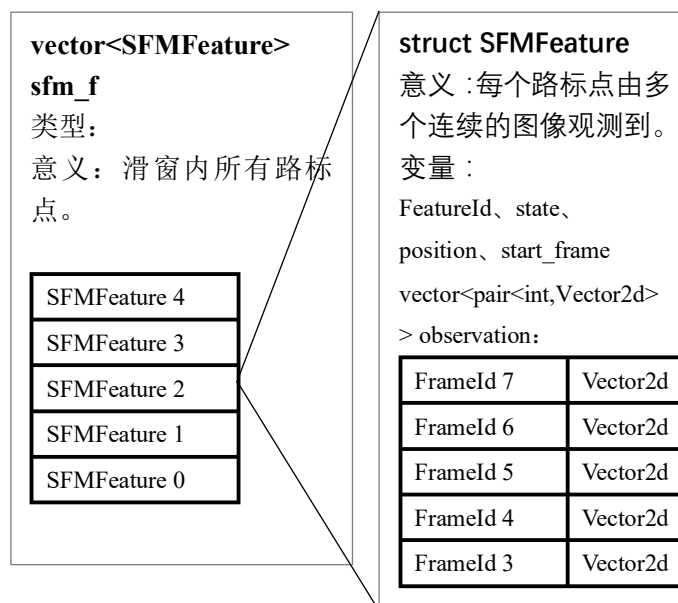


图 7 SFM 中的几个易混变量

GlobalSFM 的流程如下，其中，参考帧为上一步选出来的最共视帧，最后得到的 `sfm_tracked_points` 为三角化出来的 3D 路标点。

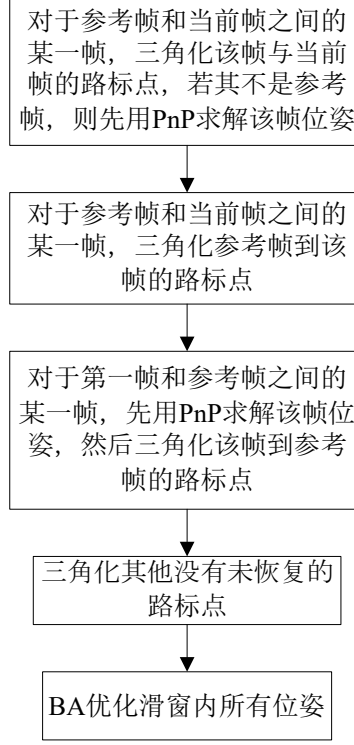


图 8 SFM 流程图

5.3 solvePnP

对滑窗内每一帧图像，都跟上一帧 SFM 得到的所有 3D 路标点，进行 `cv::solvePnP` 求解位姿。

纯视觉初始化时，我们采用第一帧 c_0 作为基准坐标系，若要转化为从 `body` 坐标系到 c_0 坐标系，可以进行如下变换：

$$q_{b_k}^{c_0} = q_{c_k}^{c_0} \otimes (q_c^b)^{-1} \quad (29)$$

$$s\bar{p}_{b_k}^{c_0} = s\bar{p}_{c_k}^{c_0} - R_{b_k}^{c_0} p_c^b \quad (30)$$

上式推导如下，这里采用个人较熟悉的写法：

$$\begin{aligned}
 T_{c_0 \leftarrow b_k} &= T_{c_0 \leftarrow c_k} T_{b \leftarrow c}^{-1} \\
 &\Leftrightarrow T_{c_0 \leftarrow b} \left(T_{b \leftarrow c} \right) = T_{c_0 \leftarrow c_k} \\
 &\Leftrightarrow R_{c_0 \leftarrow b_k} (R_{b \leftarrow c} P_{c_k} + t_{b \leftarrow c}) + t_{c_0 \leftarrow b_k} = R_{c_0 \leftarrow c_k} P_{c_k} + t_{c_0 \leftarrow c_k}
 \end{aligned}$$

$$\Rightarrow \begin{cases} R_{c_0 \leftarrow b_k} = R_{c_0 \leftarrow c_k} R_{b \leftarrow c}^{-1} \\ t_{c_0 \leftarrow b_k} = t_{c_0 \leftarrow c_k} - R_{c_0 \leftarrow b_k} t_{b \leftarrow c} \end{cases}$$

5.4 visualInitialAlign

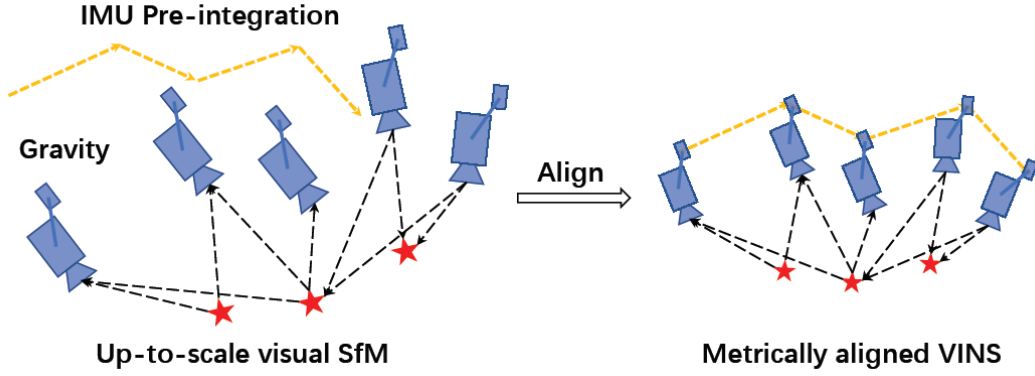


图 9 IMU 预积分与视觉结构对齐

1) 陀螺仪 **bias** 校正：对应代码中 solveGyroscopeBias()函数

目标函数为：visual 给出的相邻帧间的旋转应等于 IMU 预积分的旋转值

$$\min_{\delta b_\omega} \sum_{k \in B} \left\| q_{b_{k+1}}^{c_0}{}^{-1} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} \right\|^2 \quad (31)$$

其中：

$$\gamma_{b_{k+1}}^{b_k} \approx \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \left[\frac{1}{2} J_{b_\omega}^\gamma \delta b_\omega \right] \quad (32)$$

上述目标函数的最小值为单位四元数，所以可以将目标函数进一步写为：

$$\begin{aligned} q_{b_{k+1}}^{c_0}{}^{-1} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \left[\frac{1}{2} J_{b_\omega}^\gamma \delta b_\omega \right] &= q_{b_k}^{c_0}{}^{-1} \otimes q_{b_{k+1}}^{c_0} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \left[\frac{1}{2} J_{b_\omega}^\gamma \delta b_\omega \right] &= \hat{\gamma}_{b_{k+1}}^{b_k}{}^{-1} \otimes q_{b_k}^{c_0}{}^{-1} \otimes q_{b_{k+1}}^{c_0} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned}$$

我们只考虑虚部，则有：

$$J_{b_\omega}^\gamma \delta b_\omega = 2 \left(\hat{\gamma}_{b_{k+1}}^{b_k}{}^{-1} \otimes q_{b_k}^{c_0}{}^{-1} \otimes q_{b_{k+1}}^{c_0} \right)_{vec}$$

将左边转为正定阵，这样就可以直接用 Cholesky 进行分解了：

$$J_{b_\omega}^{\gamma T} J_{b_\omega}^\gamma \delta b_\omega = 2 J_{b_\omega}^{\gamma T} \left(\hat{\gamma}_{b_{k+1}}^{b_k}{}^{-1} \otimes q_{b_k}^{c_0}{}^{-1} \otimes q_{b_{k+1}}^{c_0} \right)_{vec} \quad (33)$$

求解出陀螺仪的 bias 后，需要对 IMU 预积分值进行重新计算。

2) 初始化速度、重力和尺度因子：对应代码中 LinearAlignment ()函数

优化变量为：

$$X_I^{3(n+1)+3+1} = [v_{b_0}^{b_0}, v_{b_1}^{b_1}, \dots, v_{b_n}^{b_n}, g^{c_0}, s] \quad (34)$$

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

残差可定义为相邻两帧之间 IMU 预积分出的增量 $\alpha_{b_{k+1}}^{b_k}$ ， $\beta_{b_{k+1}}^{b_k}$ ，与预测值之间的误差，可写成：

$$r(\hat{z}_{b_{k+1}}^{b_k}, X_I) = \begin{bmatrix} \delta\alpha_{b_{k+1}}^{b_k} \\ \delta\beta_{b_{k+1}}^{b_k} \end{bmatrix} = \begin{bmatrix} \alpha_{b_{k+1}}^{b_k} - R_{c_0}^{b_k} \left(s(\bar{p}_{b_{k+1}}^{c_0} - \bar{p}_{b_k}^{c_0}) - R_{b_k}^{c_0} v_{b_k}^{b_k} \Delta t_k + \frac{1}{2} g^{c_0} \Delta t_k^2 \right) \\ \beta_{b_{k+1}}^{b_k} - R_{c_0}^{b_k} \left(R_{b_{k+1}}^{c_0} v_{b_{k+1}}^{b_k} - R_{b_k}^{c_0} v_{b_k}^{b_k} + g^{c_0} \Delta t_k \right) \end{bmatrix} = \begin{bmatrix} 0_{3 \times 1} \\ 0_{3 \times 1} \end{bmatrix} \quad (35)$$

将公式(30)代入上式中的 $\delta\alpha_{b_{k+1}}^{b_k}$ 可得：

$$\begin{aligned} \delta\alpha_{b_{k+1}}^{b_k} &= \alpha_{b_{k+1}}^{b_k} - R_{c_0}^{b_k} \left(s(\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) - R_{b_k}^{c_0} p_c^b - (s\bar{p}_{c_k}^{c_0} - R_{b_k}^{c_0} p_c^b) - R_{b_k}^{c_0} v_{b_k}^{b_k} \Delta t_k + \frac{1}{2} g^{c_0} \Delta t_k^2 \right) \\ &= \alpha_{b_{k+1}}^{b_k} - R_{c_0}^{b_k} s(\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b - p_c^b + v_{b_k}^{b_k} \Delta t_k - \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 g^{c_0} \end{aligned}$$

我们想办法将上式转为 $Hx=b$ 的形式，这样便于直接利用 Cholesky 进行求解：

$$\begin{aligned} \delta\alpha_{b_{k+1}}^{b_k} &= \alpha_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b - R_{c_0}^{b_k} (\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) s + \Delta t_k v_{b_k}^{b_k} - \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 g^{c_0} = 0_{3 \times 1} \\ R_{c_0}^{b_k} (\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) s - \Delta t_k v_{b_k}^{b_k} + \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 g^{c_0} &= \alpha_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b \end{aligned}$$

转成矩阵形式可写成：

$$\begin{bmatrix} -I \Delta t_k & 0 & \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 & R_{c_0}^{b_k} (\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_k} \\ g^{c_0} \\ s \end{bmatrix} = \alpha_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b$$

同样的也将 $\delta\beta_{b_{k+1}}^{b_k}$ 转为矩阵形式，综合上式可写成：

$$\begin{bmatrix} -I \Delta t_k & 0 & \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 & R_{c_0}^{b_k} (\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) \\ -I & R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k} \Delta t_k & 0 \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_k} \\ g^{c_0} \\ s \end{bmatrix} = \begin{bmatrix} \alpha_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b \\ \beta_{b_{k+1}}^{b_k} \end{bmatrix} \quad (36)$$

即： $H^{6 \times 10} X_I^{10 \times 1} = b^{6 \times 1}$

这样，可以用 Cholosky 分解下面方程求解 X_I ：

$$H^T H X_I^{10 \times 1} = H^T b$$

3) 修正重力矢量：对应代码中 RefineGravity()函数

因此重力矢量的模固定，因此将为 2 个自由度，可写成：

$$\hat{\mathbf{g}}^{3 \times 1} = \|\mathbf{g}\| \cdot \bar{\mathbf{g}}^{3 \times 1} + \omega_1 \bar{\mathbf{b}}_1^{3 \times 1} + \omega_2 \bar{\mathbf{b}}_2^{3 \times 1} = \|\mathbf{g}\| \cdot \bar{\mathbf{g}}^{3 \times 1} + \bar{\mathbf{b}}^{3 \times 2} \omega^{2 \times 1} \quad (37)$$

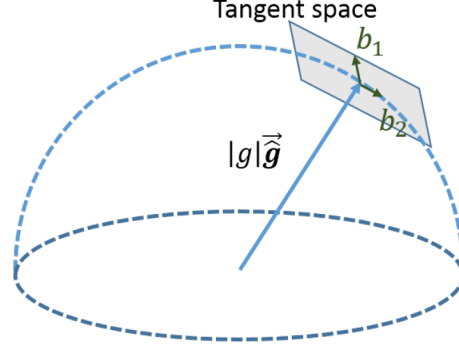


图 10 两自由度的重力矢量参数化

将上式代入(36)，重新整理可得：

$$\begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2}R_{c_0}^{b_k}\Delta t_k^2\bar{\mathbf{b}} & R_{c_0}^{b_k}(\bar{\mathbf{p}}_{c_{k+1}}^{c_0} - \bar{\mathbf{p}}_{c_k}^{c_0}) \\ -I & R_{c_0}^{b_k}R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k}\Delta t_k\bar{\mathbf{b}} & 0 \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}} \\ \omega \\ s \end{bmatrix} = \begin{bmatrix} \alpha_{b_{k+1}}^{b_k} - p_c^b + R_{c_0}^{b_k}R_{b_{k+1}}^{c_0}p_c^b - \frac{1}{2}R_{c_0}^{b_k}\Delta t_k^2\|\mathbf{g}\| \cdot \bar{\mathbf{g}} \\ \beta_{b_{k+1}}^{b_k} - R_{c_0}^{b_k}\Delta t_k\|\mathbf{g}\| \cdot \bar{\mathbf{g}} \end{bmatrix} \quad (38)$$

即： $H^{6 \times 9}X_l^{9 \times 1} = b^{6 \times 1}$, $\omega^{2 \times 1} = [\omega_1, \omega_2]^T$

这样，可以用 Cholosky 分解下面方程求解 X_l ：

$$H^T H X_l = H^T b \quad (39)$$

这样我们就得到了在 C_0 系下的重力向量 g^{c_0} ，通过将 g^{c_0} 旋转至惯性坐标系中的 z 轴方向，可以计算相机系到惯性系的旋转矩阵 $q_{c_0}^w$ ，这样就可以将所有变量调整至惯性世界系中。

六、边缘化 Marginalization 和 FEJ

6.1 边缘化和 Schur 补公式

根据前面讨论的基于高斯牛顿的非线性优化理论可知， $H\delta x = b$ 可写成如下形式：

$$\begin{bmatrix} \Lambda_a & \Lambda_b \\ \Lambda_b^T & \Lambda_c \end{bmatrix} \begin{bmatrix} \delta x_a \\ \delta x_b \end{bmatrix} = \begin{bmatrix} g_a \\ g_b \end{bmatrix} \quad (40)$$

值得说明的是，上式中 δx_a 、 δx_b 并非一定为相机位姿部分和路标部分，而是希望 marg 的部分和希望保留的部分。另外，VINS 中的边缘化与 G2O 计算过程中的边缘化意义不大相

同（虽然处理方法一致）：G2O 中对路标点设置边缘化(pPoint->setMarginalized(true))是为了在计算求解过程中，先消去路标点变量，实现先求解相机位姿，然后再利用求解出来的相机位姿反过来计算路标点的过程，目的是为了加速求解，并非真的将路标点给边缘化掉；而 VINS 中则真正需要边缘化掉滑动窗口中的最老帧或者次新帧，目的是希望不再计算这一帧的位姿或者与其相关的路标点，但是希望保留该帧对窗口内其他帧的约束关系。

假设上式中的 x_a 是我们要 marg 的变量，比如一个相机的 pose，因此我们更关心如何只去求解我们希望保留的变量 δx_b ，而不再求解 δx_a ，同时我们又不能直接将 δx_a 和与其相关的路标点等信息直接删除掉，因为这样会减少约束，丢失信息。因此，采用如下 Schur 进行消元：

$$\begin{aligned} \begin{bmatrix} I & 0 \\ -\Lambda_b^T \Lambda_a^{-1} & I \end{bmatrix} \begin{bmatrix} \Lambda_a & \Lambda_b \\ \Lambda_b^T & \Lambda_c \end{bmatrix} \begin{bmatrix} \delta x_a \\ \delta x_b \end{bmatrix} &= \begin{bmatrix} I & 0 \\ -\Lambda_b^T \Lambda_a^{-1} & I \end{bmatrix} \begin{bmatrix} g_a \\ g_b \end{bmatrix} \\ \Leftrightarrow \begin{bmatrix} \Lambda_a & \Lambda_b \\ 0 & \Lambda_c - \Lambda_b^T \Lambda_a^{-1} \Lambda_b \end{bmatrix} \begin{bmatrix} \delta x_a \\ \delta x_b \end{bmatrix} &= \begin{bmatrix} g_a \\ g_b - \Lambda_b^T \Lambda_a^{-1} g_a \end{bmatrix} \end{aligned} \quad (41)$$

其中， $\Lambda_b^T \Lambda_a^{-1} \Lambda_b$ 就称为 Λ_a 在 Λ_b 中的 Schur 项，那么有了上面式子，我们就可以直接计算 δx_b 了：

$$(\Lambda_c - \Lambda_b^T \Lambda_a^{-1} \Lambda_b) \delta x_b = g_b - \Lambda_b^T \Lambda_a^{-1} g_a \quad (42)$$

注意上式是从(40)式转化而来，我们并未丢失任何约束，因此不会丢失信息。值得说明的，上面的公式即为要保留变量 δx_b 的先验信息。

6.3 一个边缘化的例子

下面我们用一个具体例子来形象说明边缘化过程及其导致的矩阵稠密现象(fill-in)。假设有四个相机位姿 x_{pi} ，以及 6 个路标点 x_{mk} （路标点用 xyz 的参数化），相机与路标点的边表示一次观测，相邻相机之间的边表示 IMU 约束，相互关系如下：

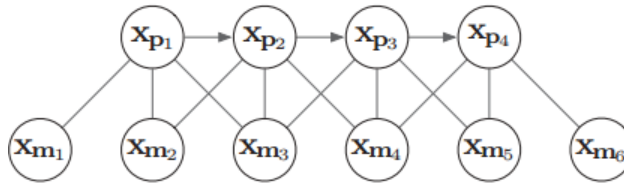


图 11 4 个相机观察到 6 个路标点的图关系

下面试图将 x_{p1} 给 marg 掉，然后再将 x_{m1} 给 marg 掉，看看 H 矩阵会如何变化。

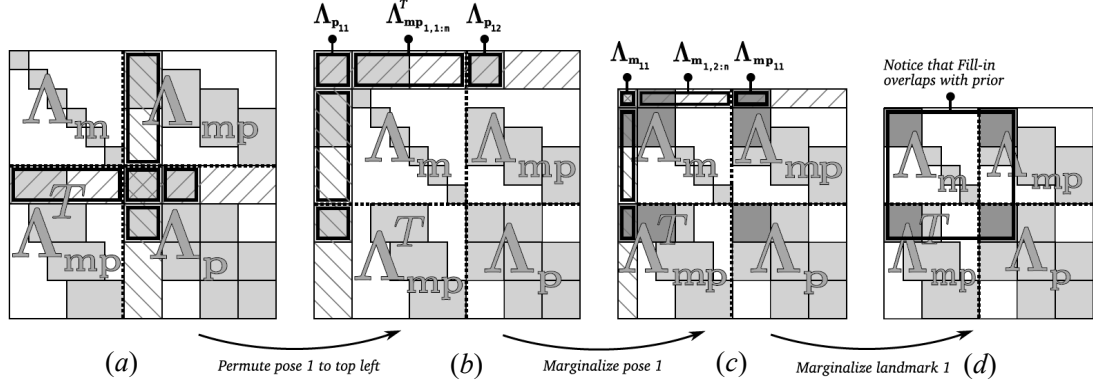


图 12 边缘化掉位姿和路标点时 H 矩阵的变化

下面我们对上图进行详细说明，图(12-a)表示原始的 H 矩阵，注意这里的左上角为路标点相关部分，而右下角是 pose 相关部分。

	m1	m2	m3	m4	m5	m6	p1	p2	p3	p4
m1	1						1			
m2		1					1	1		
m3			1				1	1	1	
m4				1				1	1	1
m5					1				1	1
m6						1				1
p1	1	1	1				1	1		
p2		1	1	1			1	1	1	
p3			1	1	1			1	1	1
p4				1	1	1			1	1

图 13 图(12-a)的详细表示

图(12-b)是把 H 矩阵中跟 x_{p1} 相关的部分移动到 H 矩阵左上角。

	p1	m1	m2	m3	m4	m5	m6	p2	p3	p4
p1	1	1	1	1				1		
m1	1	1								
m2	1		1					1		
m3	1			1				1	1	
m4					1			1	1	1
m5						1			1	1
m6							1			1
p2	1		1	1	1			1	1	
p3				1	1	1		1	1	1
p4					1	1	1		1	1

图 14 图(12-b)的详细表示

图(12-c)是把 x_{p1} 边缘掉后的 H 矩阵，实际上就是(42)式中对应的 $(\Lambda_c - \Lambda_b^T \Lambda_a^{-1} \Lambda_b)$ 。这里

详细说明下，我们从上面(40)式进行说明，这里的 $\delta x_a = \delta x_{p1}^{6 \times 1}$,

$$\begin{bmatrix} \Lambda_a^{6 \times 6} & \Lambda_b^{6 \times (3 \times 6 + 6 \times 3) = 6 \times 36} \\ \Lambda_b^{T 36 \times 6} & \Lambda_c^{36 \times 36} \end{bmatrix} \begin{bmatrix} \delta x_a^{6 \times 1} \\ \delta x_b^{36 \times 1} \end{bmatrix} = \begin{bmatrix} g_a^{6 \times 1} \\ g_b^{36 \times 1} \end{bmatrix} \quad (43)$$

其中各矩阵的维度已在上式标明，也就是说图(12-c)的矩阵大小为 36×36 。我们可以看到，图(c)相对于图(b)变得更稠密了，即 marg 掉一个 pose，会使得剩余的 H 矩阵有 3 个地方被 fill-in，如图(c)中颜色加重区域。

	m1	m2	m3	m4	m5	m6	p2	p3	p4
m1	1	1	1				1		
m2	1	1	1				1		
m3	1	1	1				1	1	
m4				1			1	1	1
m5					1			1	1
m6						1			1
p2	1	1	1	1			1	1	
p3			1	1	1		1	1	1
p4				1	1	1		1	1

图 15 图(12-c)的详细表示

这时图关系则变为：

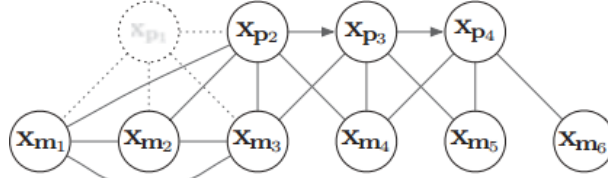


图 16 边缘化掉 x_{p1} 后的图关系

注意，观察图 16 可发现这时的 x_{m1} 、 x_{m2} 和 x_{m3} 彼此之间已经产生了新的约束关系，且 x_{p2} 也与 x_{m1} 产生了新的约束关系。

图(12-d)是 marg 掉 x_{m1} 后的 H 矩阵，详细如下所示：

	m2	m3	m4	m5	m6	p2	p3	p4
m2	1	1				1		
m3	1	1				1	1	
m4			1			1	1	1
m5				1			1	1
m6					1			1
p2	1	1	1			1	1	
p3		1	1	1		1	1	1
p4			1	1	1		1	1

图 17 图(13-d)的详细表示

对应的图关系如下：

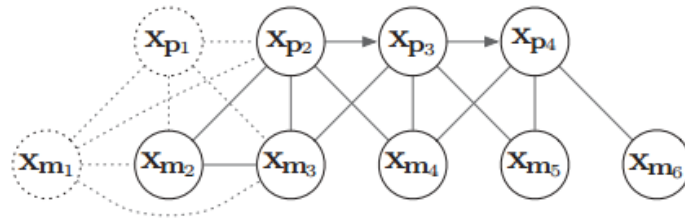


图 18 边缘化掉 x_{m1} 后的图关系

我们发现，marg 掉 x_{m1} 后，并没有使 H 矩阵更稠密，这是因为 x_{m1} 之前并未与其他 pose 有约束关系，即并未被观察到，因此如果 marg 掉那些不被其他帧观测到的路标点，不会显著地使 H 矩阵变得稠密。而要 marg 掉的路标点中，对于那些被其他帧观测到路标点，要么就别设置为 marg，要么就宁愿丢弃，这就是 OKVIS 中用到的策略。

6.3 VINS 两种边缘化策略

marginalization_factor.cpp 代码中有几个变量需要提前说明：

```

struct ResidualBlockInfo{
    CostFunction *cost_function;
    (其中parameter_block_sizes每个优化变量块的变量大小, 以IMU残差为例, 为[7,9,7,9])
    LossFunction *loss_function;
    //优化变量数据
    <double *> parameter_blocks;
    //待marg的优化变量id
    <int> drop_set;
    //Jacobian
    double **raw_jacobians;
    <MatrixXd> jacobians;
    //残差, IMU : 15x1, 视觉 : 2x1
    VectorXd residuals;
}

```



```

class MarginalizationInfo{
    //所有观测项
    <ResidualBlockInfo *> factors;
    //m为要marg掉的变量个数，也就是parameter_block_idx的总
    localSize，以double为单位，VBias为9，PQ为6，
    //n为要保留下的优化变量的变量个数，
    n=localSize(parameter_block_size) - m
    int m,n;
    //<优化变量内存地址， localSize>
    <long, int> parameter_block_size;
    int sum_block_size;
    //<待marg的优化变量内存地址，在
    //parameter_block_size中的id,以double为单位>
    <long, int> parameter_block_idx;
    //<优化变量内存地址，数据>
    <long, double*> parameter_block_data;
    <int> keep_block_size;
    <int> keep_block_idx;
    <double*> keep_block_data;
    MatrixXd linearied_jacobians;
    VectorXd lineared_residuals;
    //加残差块相关信息(优化变量、待marg的变量)
    void addResidualBlockInfo(ResidualBlockInfo *);
    //计算每个残差对应的Jacobian，并更新parameter_block_data
    void preMarginalize();
    //pos为所有变量维度，m为需要marg掉的变量，n为需要保留的变量
    void marginalize();
}

```

举例说明，当第一次 marg 掉最老帧时，parameter_block_size 为所有变量及其对应的 localSize，parameter_block_data 为对应的数据（double*类型）。下面举例说明，假设最老帧即第 0 帧看到了 $k=68$ 个路标点，那么代码中的 parameter_block_size 或 parameter_block_data 两个 vector 的大小均为：

$$2(V_{0:1}) + 11(P_{0:10}) + 1(T_{bc}) + 1(t_b) + 68(\lambda_{l:k}) = 83$$

pos=所有变量总 localSize 为： $9 \times 2 + 6 \times 11 + 6 + 1 + 68 = 159$

0x...	1	λ_k
0x...	1	...
0x...	1	λ_1
0x...	1	t_b
0x...	6	T_{bc}
0x...	6	P_{10}
0x...	6	...
0x...	6	P_2
0x...	9	V_1
0x...	6	P_1
0x...	9	V_0
0x...	6	P_0

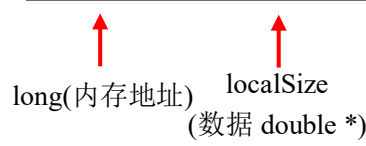


图 19 parameter_block_size(或 parameter_block_data)的一个例子

parameter_block_idx 为需要 marg 掉的变量，我们仍然假设第 0 帧看到 $k=68$ 个路标点，则 parameter_block_idx 的数组大小为：

$$\text{parameter_block_idx.size()} = 1(P_0) + 1(V_0) + 68(\lambda_{1:k}) = 70$$

MarginalizationInfo::m 表示需 marg 变量的总 localSize，即为：

$$m = 6 + 9 + 68 = 83$$

那么，MarginalizationInfo::n 表示需保留变量的总 localSize 为：

$$n = \text{pos} - m = 159 - 83 = 76$$

0x...	82	λ_n
0x...
0x...	16	λ_2
0x...	15	λ_1
0x...	6	V_0
0x...	0	P_0




图 20 parameter_block_idx 的一个例子

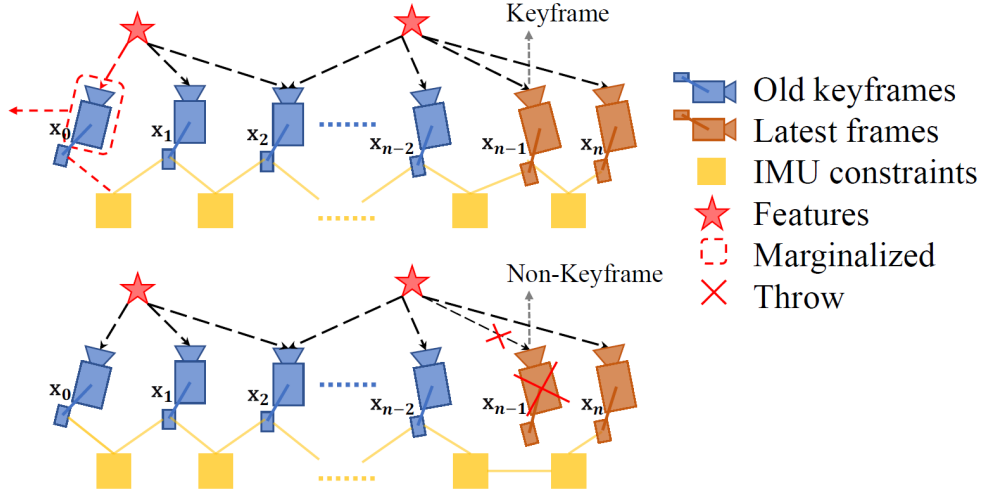


图 21 VINS 边缘化策略

VINS 根据次新帧是否为关键帧，分为两种边缘化策略：通过对比次新帧和次次新帧的视差量，来决定 marg 掉次新帧或者最老帧，对应到 Estimator::optimization 代码中详细分析：

1. 当次新帧为关键帧时，MARGIN_OLD，将 marg 掉最老帧，及其看到的路标点和相关联的 IMU 数据，将其转化为先验信息加到整体的目标函数中：

1) 把上一次先验项中的残差项（尺寸为 n ）传递给当前先验项，并从中去除需要丢弃的状态量；

2) 将滑窗内第 0 帧和第 1 帧间的 IMU 预积分因子（pre_integrations[1]）放到 marginalization_info 中，即图 11 中上半部分中 x_0 和 x_1 之间的表示 IMU 约束的黄色块；

3) 挑选出第一次观测帧为第 0 帧的路标点，将对应的多组视觉观测放到 marginalization_info 中，即图 11 中上半部分中 x_0 所看到的红色五角星的路标点；

4) marginalization_info->preMarginalize(): 得到每次 IMU 和视觉观测(cost_function)对应的参数块(parameter_blocks)，雅可比矩阵(jacobians)，残差值(residuals)；

5) marginalization_info->marginalize(): 多线程计整个先验项的参数块，雅可比矩阵和残差值，即计算公式(42)，其中与代码对应关系为：

```
Eigen::VectorXd bmm = b.segment(0, m);
Eigen::MatrixXd Amr = A.block(0, m, m, n);
Eigen::MatrixXd Arm = A.block(m, 0, n, m);
Eigen::MatrixXd Arr = A.block(m, m, n, n);
Eigen::VectorXd brr = b.segment(m, n);
A = Arr - Arm * Amm_inv * Amr;
b = brr - Arm * Amm_inv * bmm;
```

$$\begin{array}{cc}
& \begin{matrix} m & n \end{matrix} \\
\begin{matrix} m \\ n \end{matrix} & \begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{mr} \\ \mathbf{A}_{rm} & \mathbf{A}_{rr} \end{bmatrix}
\end{array}
\times
\begin{array}{c}
\begin{bmatrix} \delta x_m \\ \delta x_r \end{bmatrix}
\end{array}
=
\begin{array}{cc}
& \begin{matrix} m \\ n \end{matrix} \\
\begin{matrix} m \\ n \end{matrix} & \begin{bmatrix} \mathbf{b}_{mm} \\ \mathbf{b}_{rr} \end{bmatrix}
\end{array}$$

图 22 marginalize()中涉及的几个矩阵

2. 当次新帧不是关键帧时，MARGIN_SECOND_NEW，我们将直接扔掉次新帧及它的视觉观测边，而不对次新帧进行 marg，因为我们认为当前帧和次新帧很相似，也就是说当前帧跟路标点之间的约束和次新帧与路标点的约束很接近，直接丢弃并不会造成整个约束关系丢失过多信息。但是值得注意的是，我们要保留次新帧的 IMU 数据，从而保证 IMU 预积分的连贯性。

通过以上过程先验项就构造完成了，在对滑动窗口内的状态量进行优化时，把它与 IMU 残差项和视觉残差项放在一起优化，从而得到不丢失历史信息的最新状态估计的结果。

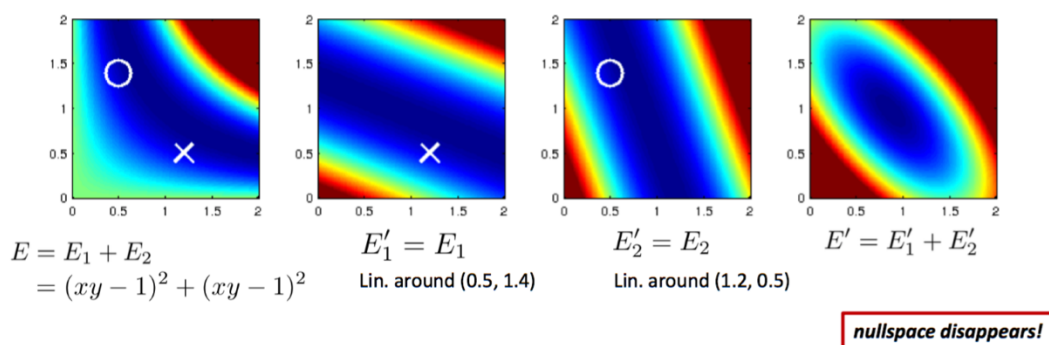
6.4 First Estimate Jacobian(FEJ)

因为迭代过程中，状态变量会被不断更新，计算雅克比时我们要 fix the linearization point。所谓 linearization point 就是线性化时的状态变量，即求雅克比时的变量，因为计算雅克比是为了泰勒展开对方程进行线性化。我们要固定在点 \mathbf{x}_0 (marg 时的状态变量)附近去计算雅克比，而不是每次迭代更新以后的 \mathbf{x} 。

但是，VINS 中并未使用 FEJ 的策略，这里我们进行简要说明：对于滑窗内剩余的优化变量，如倒数第二帧位姿 \mathbf{T}_1 ，当边缘化掉最老帧 \mathbf{T}_0 后，会给 \mathbf{T}_1 加上新的约束。值得注意的是，这个新约束的线性化点是在 marg 掉 \mathbf{T}_0 时刻的 \mathbf{T}_1 的值，而当之后 \mathbf{T}_1 迭代更新后，该 marg 产生的约束并不会调整线性化点，即不会在新的 \mathbf{T}_1 处重新展开，这样会导致两次的线性化点不一致。但据作者描述因未发现明显的 yaw 不可观性变化导致的轨迹漂移，因此并未采用 FEJ 策略，反而加入 FEJ 后导致结果不佳。可以结合 OC-EKF 和 MSCKF 2.0 等一系列论文对 FEJ 进行深入研究，这里仅将 FEJ 的相关知识罗列在此。

Windowed, real-time optimization: Consistency.

(for now, let's assume we have initializations, and know which points to use and where they are visible.)



never combine linearizations around different linearization points,
especially in the presence of non-linear nullspaces!
It will render unobservable dimensions observable, and corrupt the system.

图 20 FEJ 说明

在刘毅(稀疏毅), 王京, 晓佳等人讨论下, 对这张图作出了如下解释: 四张能量图中, 第一张是说明能量函数 E 由两个同样的非线性函数 E_1 和 E_2 组成, 我们令函数 $E=0$, 这时方程的解为 $xy=1$, 对应图中深蓝色的一条曲线。第二张能量函数图中的 E'_1 对应函数 E_1 在点 (0.5,1.4) 处的二阶泰勒展开, 第三张能量函数图中的 E'_2 对应函数 E_2 在点 (1.2,0.5) 处的二阶泰勒展开。注意这两个近似的能量函数 E'_1 和 E'_2 是在不同的线性点附近对原函数展开得到的。最后一张图就是把这个近似得到的能量函数合并起来, 对整个系统 E 的二阶近似。

从第四个能量函数图中, 我们发现一个大问题, 能量函数为 0 的解由以前的一条曲线变成了一个点, 不确定性的东西变得确定了, 专业的术语叫不可观的状态变量变得可观了, 说明我们人为的引入了错误的信息。这个实验的实质在于, 在不同的点线性化后, 强行加起来, 实际上引入了一些人为的约束, 或者说引入了人为的“错误观测”, 导致整个系统的崩溃。对应到 marg 的问题上, 本来我们是在最初(initial)那个点附近进行线性化, 但是在 marg 的过程, initial 那个点变了, 它一开始是有未 marg 的点的, marg 之后, 把那些点的信息给了留下的那些点, 这就使得剩下那些点进行了一些偏移, 他们和之前的状态不同了, 这个时候再线性化, 就会导致在不同的地方进行了线性化, 这样就会像上面那个例子一样, 引入了错误的信息, 导致整个优化过程的崩溃。因此, marg 时, 被 marg 的那些变量的雅克比已经不更新了, 而此时留在滑动窗口里的其他变量的雅克比要用和 marg 时一样的线性点, 就是 FEJ 去算, 不要用新的线性点了。

七、闭环检测和优化

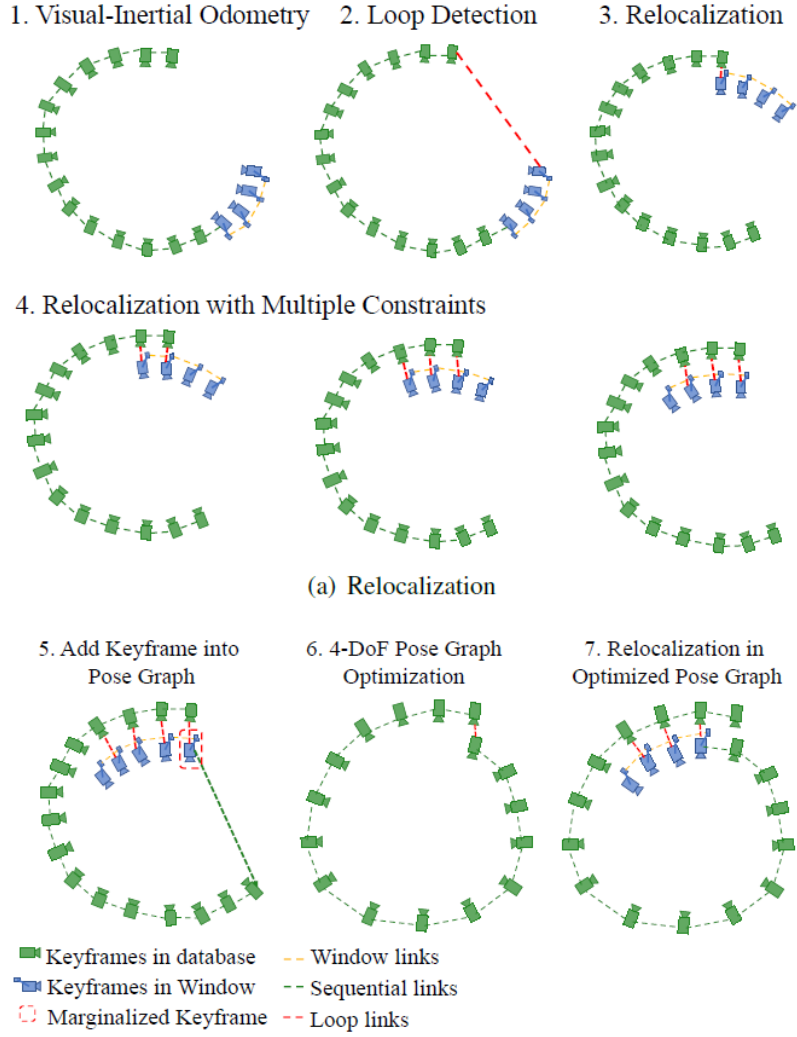


图 21 闭环检测和优化的示意图

7.1 闭环检测

VINS 是采用 BRIEF 描述子的 DBoW2 词袋进行闭环检测，因为前端识别的 Harris 角点数量通常只有 70 个 (VINS-Mobile)，对于闭环检测远远不够，因此会对新来的 KeyFrame 即后端非线性优化刚处理完的关键帧，再重新检测出 500 个 FAST 角点进行闭环检测用，同时对所有新老角点进行 BRIEF 描述 (70 个点的描述在 `computeWindowBRIEFPoint` 中，新的 500 个点在 `computeBRIEFPoint` 中，具体描述子代码在 `BriefExtractor::operator`)。然后，将 KF 添加到数据中时 (`addKeyFrame`)，计算当前帧与词袋的相似度分数，并与关键帧数据库中所有帧进行对比，并进行闭环一致性检测，获得闭环的候选帧 (`detectLoop`)。

当检测到闭环后，我们在 `findConnection` 函数中进行 PnP 求解相对位姿，首先利用 BRIEF 描述子对闭环对老帧的 500 个 FAST 角点，和当前帧中的 70 个 Harris 角点进行基于描述

子的邻域匹配 (searchByBRIEF); 当匹配点数大于阈值后, 将匹配点对利用 PnPRANSAC 求基础矩阵进行 RANSAC 异常点剔除, 并求解相对位姿存储在 loop_info 变量中用于后续的闭环优化使用 (为何求 PnP 时要用新帧的 3D 点, 老帧的 2D 点呢? !), 当剔除后的匹配点仍超过阈值时, 我们最终认为该候选帧是一个正确的闭环帧, 并将闭环帧 push 到 optimize_buf 中。

7.2 快速重定位

当检测到当前帧与之前帧 (记为第 v 帧) 有闭环时, 若开启快速重定位, 即 FAST_RELOCALIZATION 为 true 时, 会将第 v 帧的位姿和相关特征点作为视觉约束项, 加到后端非线性优化的整体目标函数中, 但是并未固定第 v 帧的位姿, 只是用滑窗优化来更精确计算第 v 帧的位姿, 从而计算出闭环帧之间的相对位姿关系。这样的目标函数可写为:

$$\min_X \left\{ \|r_p - H_p X\|^2 + \sum_{k \in B} \left\| r_B \left(\hat{z}_{b_{k+1}}^{b_k}, X \right) \right\|_{P_{b_{k+1}}^{b_k}}^2 + \sum_{(l,j) \in C} \left\| r_C \left(\hat{z}_l^{C_j}, X \right) \right\|_{P_l^{C_j}}^2 + \sum_{(l,v) \in L} \left\| r_C \left(\hat{z}_l^v, X, \hat{q}_v^w, \hat{p}_v^w \right) \right\|_{P_l^{C_v}}^2 \right\} \quad (44)$$

这样, 我们根据优化得到的相对位姿关系, 对滑窗内的所有帧进行调整, 完成快速重定位。

7.3 闭环关键帧数据库

上面说的关键帧数据库, 是当滑窗内的关键帧被移出滑窗时才会添加进数据库中 (注意这种策略貌似在最新的 VINS-Mono 已放弃), 这样可以保证添加进来的关键帧位姿基本正确, 因为已经经过前面的多次滑窗优化。当然, 即使不正确也无所谓, 因为会在闭环优化时进行修正。数据库中的帧对于下一步的闭环优化会提供两种边:

- a. 序列边 (Sequential edge): 是指通过 VIO 计算的两帧之间的相对位姿:

$$\hat{p}_{ij}^i = (\hat{R}_i^w)^{-1} (\hat{p}_j^w - \hat{p}_i^w) \quad (45)$$

$$\hat{\psi}_{ij} = \hat{\psi}_j - \hat{\psi}_i \quad (46)$$

- b. 闭环边 (Loop edge): 是指检测到闭环的两帧。

当运行时间越来越长, 数据库将变得越来越大, 导致闭环检测和闭环优化的耗时也将越来越长。虽然前面已经仅保留每帧图像的位姿和特征点描述子, 已扔掉原始图像, 但是当运行几小时后仍将无法实时。因此, 我们将根据分布密度对数据库进行下采样, 保留那些与周围帧的位置和角度不同的, 而丢弃集中在一起的帧 (代码对应

KeyFrameDatabase::downsample)。

7.4 闭环优化

当从滑窗内滑出的帧与数据库中的帧为闭环帧时，则对数据库的所有帧进行闭环优化。因为前面已经跟重力对齐，因此根据重力方向可以观测出俯仰 θ 和翻滚 ϕ 角度，即 pitch 和 roll 可观。因此闭环优化时，我们仅优化位置 x,y,z 和偏航角 yaw 这四个自由度。

那么，第 i 帧和第 j 帧的残差可写成：

$$r_{i,j}(p_i^w, \psi_i, p_j^w, \psi_j) = \begin{bmatrix} R(\hat{\phi}_i, \hat{\theta}_i, \psi_i)^{-1}(p_j^w - p_i^w) - \hat{p}_{ij}^i \\ \psi_j - \psi_i - \hat{\psi}_{ij} \end{bmatrix} \quad (47)$$

所有序列边和回环边的整体目标函数可写为：

$$\min_{p, \psi} \left\{ \sum_{(i,j) \in \mathcal{S}} \|r_{i,j}\|^2 + \sum_{(i,j) \in \mathcal{L}} h(\|r_{i,j}\|^2) \right\} \quad (48)$$

其中， \mathcal{S} 为序列边， \mathcal{L} 为回环边， $h(\cdot)$ 为 huber 核函数。对应代码为 KeyFrameDatabase::optimize4DoFLoopPoseGraph，注意代码中采用的 Ceres 自动求导方式。

到目前为止，我们就完成了对整个轨迹的闭环优化。

7.5 程序逻辑

下面我们对闭环与后端优化的关系再进行详细说明。

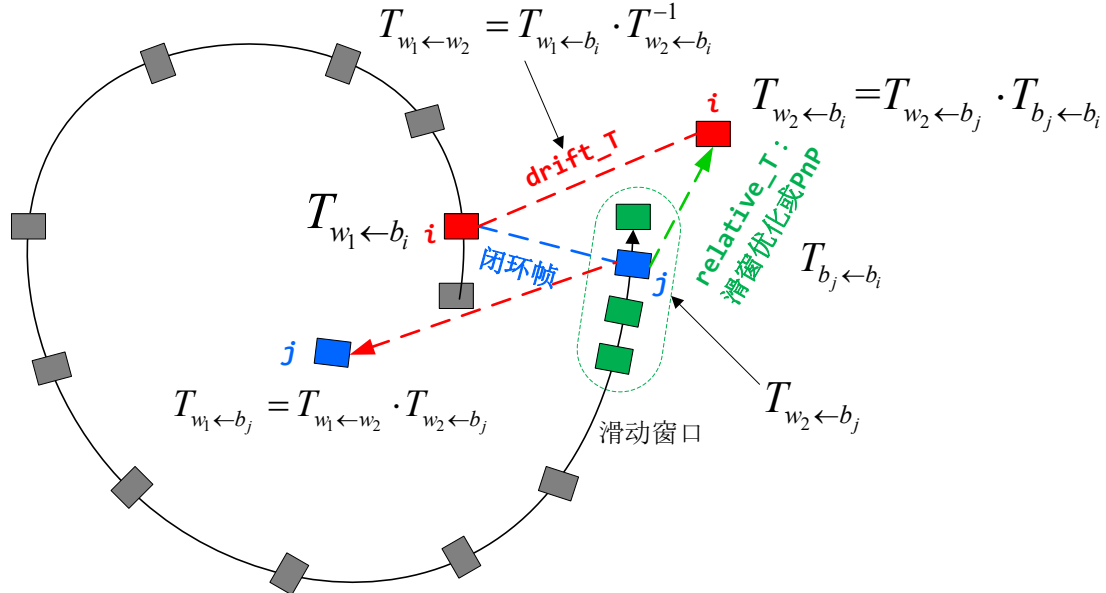


图 22 闭环关系图

假设发生闭环的两帧为滑窗内的第 j 帧（蓝色帧），和数据库中的第 i 帧（红色帧）。令第 i 帧的位姿为 $T_{w_1 \leftarrow b_i}$ ，第 j 帧的位姿为 $T_{w_2 \leftarrow b_j}$ ，值得注意的是，两帧的世界系并不相同，滑

窗所在的世界系 w_2 为原世界系 w_1 产生累积误差后的结果，重定位需要做的就是将 w_2 改到 w_1 。那么，根据 PnP 可得到两帧之间的相对位姿 $T_{b_j \leftarrow b_i}$ ，注意这里的 PnP 是用的第 i 帧的 2D 点和第 j 帧的 3D 点。但是 PnP 计算的相对位姿通常不准确，因此，可将 PnP 的结果作为初始值传到滑窗中，优化得到第 i 帧在世界系 w_2 中的位姿 $T_{w_2 \leftarrow b_i}$ 。为了将 w_2 改到 w_1 ，我们计算累积偏移位姿： $T_{w_1 \leftarrow w_2} = T_{w_1 \leftarrow b_i} \cdot T_{w_2 \leftarrow b_i}^{-1}$ ，这样的话，就可以将当前的第 j 帧重定位到原世界系中： $T_{w_1 \leftarrow b_j} = T_{w_1 \leftarrow w_2} \cdot T_{w_2 \leftarrow b_j}$ 。如下图所示：

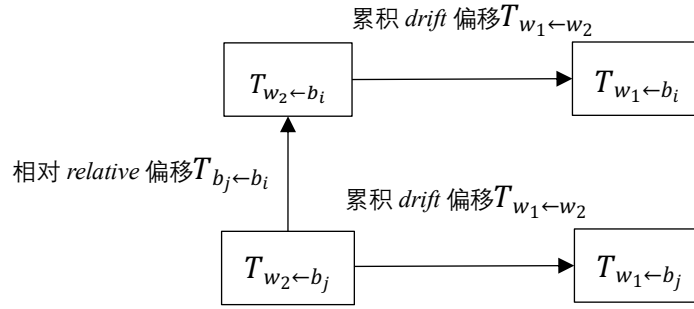


图 23 闭环帧之间的两种关系

下图给出了闭环检测和优化的程序流程图：

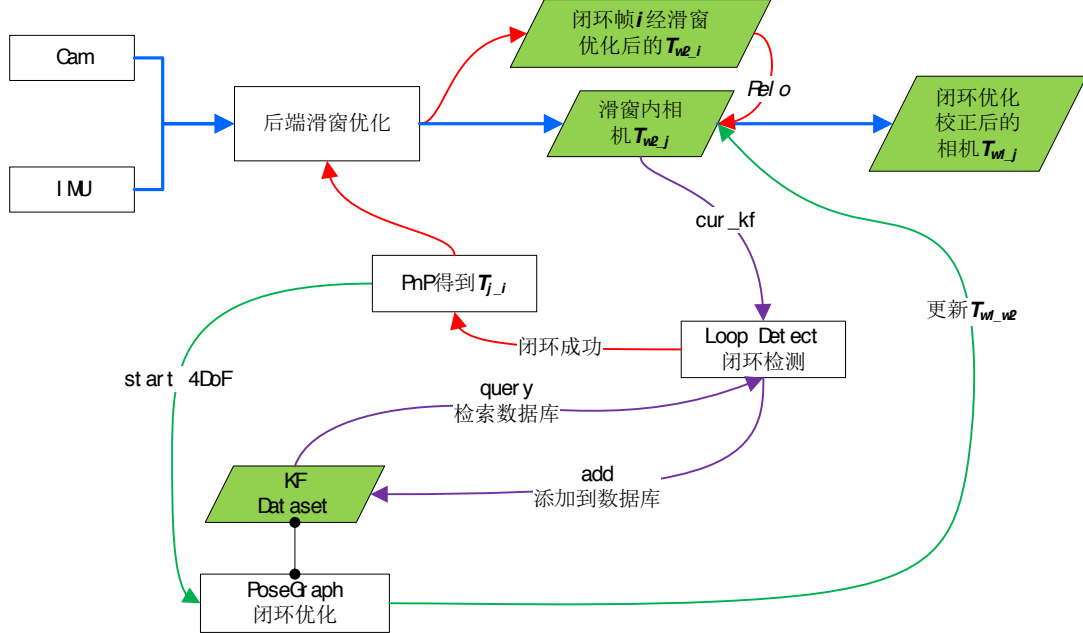


图 24 闭环优化流程图

上图中，蓝线为正常的闭环优化流程，即通过后端的非线性优化来更新滑窗内所有相机的位姿。紫线为闭环检测模块，当后端优化完成后，会将滑窗内的次新帧进行闭环检测，即首先提取新角点并进行描述，然后与数据库进行检索，寻找闭环帧，并将该帧添加到数据库

中。红线为快速重定位模块，当检测到闭环帧后，会将闭环约束添加到后端的整体目标函数中进行非线性优化，得到第 i 帧（注意这里的帧为闭环帧中的老帧）经过滑窗优化后的位姿，从而计算出累积的偏移误差，进而对滑窗内的位姿进行修正。

这里，也可以在四自由度优化后 `PoseGraph::optimize4DoF()`，将优化后的 $T_{w_1 \leftarrow w_2}$ ，即对应代码中的 `drift_correct_r`、`drift_correct_t`，传回给后端优化线程，来更新滑窗内的相机位姿 `Estimator::update_loop_correction()`。

八、其他

8.1 选 KF 策略

代码对应在 `addFeatureCheckParallax`。

首先，迭代 `image` 即当前帧检测到的每个路标点 `id`，看看图 4 中的 `feature` 队列中是否包含，若不含，则添加到 `feature` 队列中；若包含，则添加到对应 `id` 的 `FeaturePerFrame` 队列。

然后，`compensatedParallax2` 计算每个路标点在两幅图中的视差量，若视差量大于某个阈值或者当前帧跟踪的路标点小于某个阈值，则边缘化滑窗内最老的帧；否则，边缘化掉次新帧，即倒数第二帧。

8.2 后端优化后的变量更新

代码对应：`Estimator::double2vector()`，其中 `rot_diff` 是根据滑窗中第一帧在优化前后的 `yaw` 偏差计算得到的旋转矩阵，之后对滑窗内所有帧都进行 `rot_diff` 的校正。这是因为在后端滑动窗口的非线性优化时，我们并没有固定住第一帧的位姿不变，而是将其作为优化变量进行调整。但是，因为相机的偏航角 `yaw` 是不可观测的，也就是说对于任意的 `yaw` 都满足优化目标函数，因此优化之后我们将偏航角旋转至优化前的初始状态。

8.3 丢失后多地图融合

当丢失后，会从丢失位置重新初始化，并将相机位姿 `Pose` 修正为上次修正完 `drift` 后的 `Pose`，并将 `posegraph` 中的 `drift` 清零，就修改 $R_{b_j}^w$ 为 $R_{b_j-drift}^w$ ，并在此基础上进行闭环优化。

同时，每次跟丢后的地图记为新地图，并跟老地图一块放到 `datebase` 中进行闭环检测，只是 `sequence++`，当闭环检测成功后，将新老地图进行对齐融合。

8.4 小滑窗 PnP 优化

为了控制后端滑窗优化的计算量，通常是每隔几帧（比如 3 帧）选一帧送到后端处理。为了计算被舍弃的帧的位姿，在 VINS-Mobile 版本中，加入了一种轻量级的基于小滑窗的 PnP+IMU 联合优化方案，对应在代码 vins_pnp.cpp 中。实现流程如下图所示：

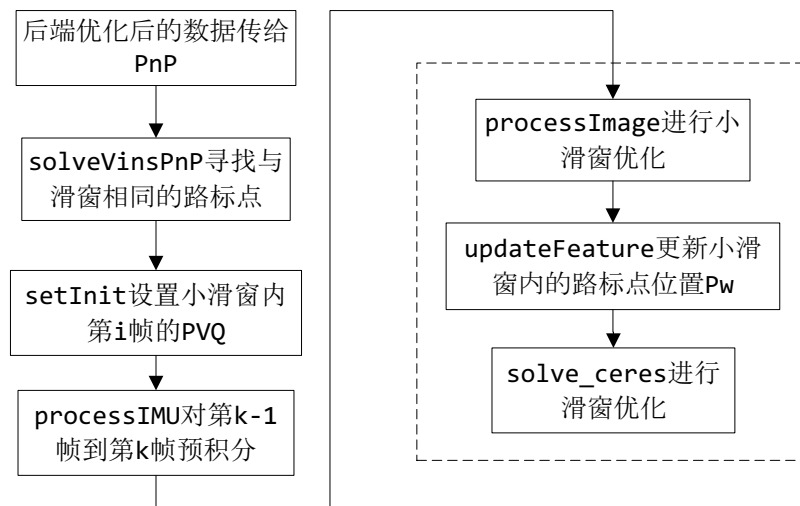


图 25 vins_pnp 流程

1. 当后端优化完成后，将滑窗内的最新帧（假设第 i 帧）的信息（时间戳、PVQ、bias）给到 FeatureTracker 的 solved_vins，并将滑窗中所有的路标点信息（id、 P_w 、被观测到的次数）给到 FeatureTracker 的 solved_features。另外，通过 GetImuMeasurements()函数计算两帧之间的 IMU 数据，送给 FeatureTracker 的 imu_msgs，注意这里的两帧是前端获得的最新帧（假设第 k 帧，注意 $k > i$ ）和上一个连续帧之间。如下图所示：

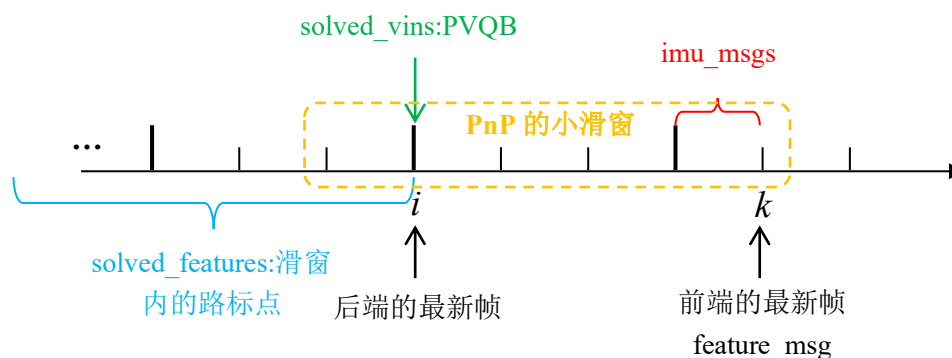


图 26 传给 FeatureTracker 的数据

2. FeatureTracker::solveVinsPnP()中寻找第 k 帧中与滑窗中相同的路标点，计算该路标点在归一化相机系中的坐标，传给 feature_msg；

3. vinsPnP::setInit()对 PnP 的小滑窗（比如 6 个相邻帧）内的 PVQB 进行赋值，当小滑窗内有第 i 帧时，则用从后端传过来的第 i 帧 PVQ 来更新；

4. processImu 是进行第 $k-1$ 帧和第 k 帧的预积分；

5. processImage 中首先通过 updateFeatures 更新小滑窗中与第 k 帧有共视关系的帧的 3D 点坐标 P_w ，最后调用 solve_ceres 来优化小滑窗内的 PVQ，不优化路标点，涉及的两个约束为：IMU 的帧间约束，和每一帧的 PnP（即每一帧的 2D-3D 观测）的视觉重投影误差约束，因子图如下所示：

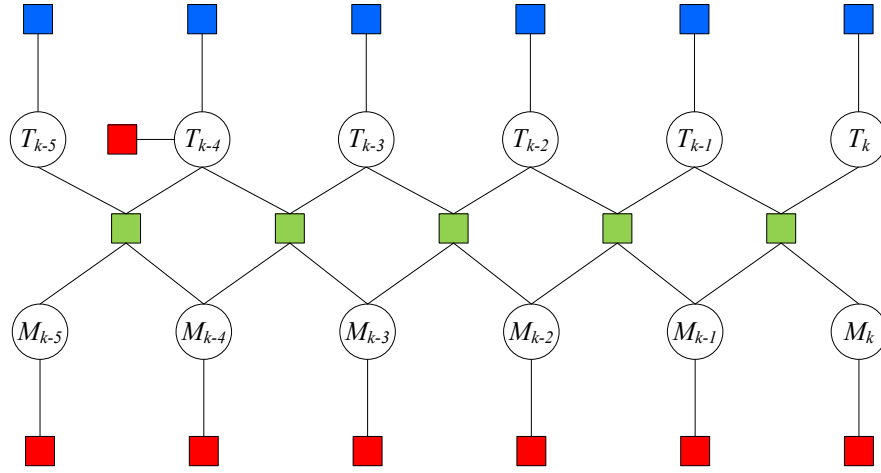


图 27 vins_pnp 的小滑窗因子图

上图中，小方块为约束因子，圆圈为优化变量（这里路标点不是优化变量）： $T = [R|t]$ ， $M = [v, b_a, b_w]$ ，绿色小块为 IMU 约束，蓝色小块为 PnP 视觉约束，红色小块为先验约束，比如第 $k-4$ 帧对应后端大滑窗的第 i 帧，因此信赖并固定住第 $k-4$ 帧的 T_{k-4} 和 M_{k-4} ；另外，固定住所有帧的 bias。

九、参考文献

- [1] T. Qin. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. arXiv preprint arXiv: 1708.03852, 2017.
- [2] N. Trawny. Indirect Kalman Filter for 3D Attitude Estimation. 2005.
- [3] Sola. Quaternion kinematics for error-state kalman filter. 2017.
- [4] K. Eickenhoff. Decoupled, Consistent Node Removal and Edge sparsification for graph-based SLAM. 2016.
- [5] J. Engel. Direct Sparse Odometry. 2016.

[6] Sliding Window Filter with Application to Planetary Landing. 2010.

[7] S. Leutenegger. Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization. 2015.

[8] “VINS-Mono 代码分析总结”, <https://www.zybuluo.com/Xiaobuyi/note/866099>.

[9] 贺一家, “SLAM 中的 marginalization 和 Schur complement”, <https://blog.csdn.net/heyijia0327/article/details/52822104>.

十、附录

10.1 IMU 状态积分公式推导

公式(1)的 IMU 连续形式下的旋转状态推导如下, 首先可写成:

$$q_{b_{k+1}}^w = q_{b_k}^w \otimes \int_{t \in [k, k+1]} \dot{q}_t dt \quad (A1)$$

根据《视觉 SLAM 十四讲》3.4.2 的四元数乘法, 我们引入左乘和右乘符号如下:

$$\begin{aligned} q_a \otimes q_b &= \mathcal{R}(q_b)q_a = \begin{bmatrix} s_b & z_b & -y_b & x_b \\ -z_b & s_b & x_b & y_b \\ y_b & -x_b & s_b & z_b \\ -x_b & -y_b & -z_b & s_b \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ z_a \\ s_a \end{bmatrix} \\ &= \mathcal{L}(q_a)q_b = \begin{bmatrix} s_a & -z_a & y_a & x_a \\ z_a & s_a & -x_a & y_a \\ -y_a & x_a & s_a & z_a \\ -x_a & -y_a & -z_a & s_a \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \\ s_b \end{bmatrix} \end{aligned} \quad (A2)$$

注意上式中的形式与文献[2]的式(10)是不相同的, 这是因为两者的左右手系不同。

为了简化, 令 $q = [x \ y \ z \ s] = [\omega \ s]$, 则有:

$$\begin{aligned} \mathcal{R}(q) &= \Omega(\omega) + sI_{4 \times 4} = \begin{bmatrix} -\omega^\wedge & \omega \\ -\omega^T & 0 \end{bmatrix} + sI_{4 \times 4} \\ \mathcal{L}(q) &= \Psi(\omega) + sI_{4 \times 4} = \begin{bmatrix} \omega^\wedge & \omega \\ -\omega^T & 0 \end{bmatrix} + sI_{4 \times 4} \end{aligned} \quad (A3)$$

其中根据论文[2]公式(86)可推导四元数的导数如下:

$$\begin{aligned} \dot{q}_t &= \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} (q_{t+\delta t} - q_t) \\ &= \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} (q_t \otimes q_{t+\delta t}^t - q_t \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}) \\ &= \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} \left(q_t \otimes \begin{bmatrix} \hat{\mathbf{k}} \sin \frac{\theta}{2} \\ \cos \frac{\theta}{2} \end{bmatrix} - q_t \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \\ &\approx \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} \left(q_t \otimes \begin{bmatrix} \hat{\mathbf{k}} \frac{\theta}{2} \\ 1 \end{bmatrix} - q_t \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \end{aligned}$$

$$\begin{aligned}
&= \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} \left[\mathcal{R} \left(\begin{bmatrix} \hat{\mathbf{k}} \frac{\theta}{2} \\ 1 \end{bmatrix} \right) - \mathcal{R} \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \right] q_t \\
&= \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} \begin{bmatrix} -\frac{\theta^\wedge}{2} & \frac{\theta}{2} \\ \frac{\theta^T}{2} & 0 \end{bmatrix} q_t
\end{aligned}$$

又有角速度为: $\omega = \lim_{\delta t \rightarrow 0} \frac{\theta}{\delta t}$, 则上式可化为:

$$\dot{q}_t = \frac{1}{2} \begin{bmatrix} -\omega^\wedge & \omega \\ -\omega^T & 0 \end{bmatrix} q_t = \frac{1}{2} \Omega(\omega) q_t = \frac{1}{2} \mathcal{R} \left(\begin{bmatrix} \omega \\ 0 \end{bmatrix} \right) q_t = \frac{1}{2} q_t \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} \quad (\text{A4})$$

10.2 连续形式 IMU 误差动力学方程推导

根据文献[3]对公式 (10) IMU 误差动力学方程中的 $\delta\dot{\beta}_t^{b_k}$ 和 $\delta\dot{\theta}_t^{b_k}$ 进行推导, 首先我们引入两个概念: **true** 和 **nominal**, 其中:

true: 真实测量值, 包含了噪声

nominal: 无噪声的理论值

首先推导 $\delta\dot{\beta}_t^{b_k}$, 下面为了书写方便, 简写成 $\delta\dot{\beta}$, 有 $\delta\dot{\beta} = \dot{\beta}_{true} - \dot{\beta}_{nominal}$, 其中:

$$\begin{aligned}
\dot{\beta}_{true} &= R_{t_{true}}^{b_k} (\hat{a}_{t_{true}} - b_{a_{t_{true}}}) = R_t^{b_k} \exp(\delta\theta^\wedge) (\hat{a}_t - n_a - b_{a_t} - \delta b_{a_t}) \\
&= R_t^{b_k} (I + \delta\theta^\wedge) (\hat{a}_t - n_a - b_{a_t} - \delta b_{a_t}) \\
&= R_t^{b_k} [\hat{a}_t - n_a - b_{a_t} - \delta b_{a_t} + \delta\theta^\wedge (\hat{a}_t - b_{a_t})] \\
&= R_t^{b_k} [\hat{a}_t - n_a - b_{a_t} - \delta b_{a_t} - (\hat{a}_t - b_{a_t})^\wedge \delta\theta]
\end{aligned}$$

$$\dot{\beta}_{nominal} = R_t^{b_k} (\hat{a}_t - b_{a_t})$$

则:

$$\delta\dot{\beta} = \dot{\beta}_{true} - \dot{\beta}_{nominal} = -R_t^{b_k} (\hat{a}_t - b_{a_t})^\wedge \delta\theta - R_t^{b_k} \delta b_{a_t} - R_t^{b_k} n_a \quad (\text{A5})$$

下面推导 $\delta\dot{\theta}_t^{b_k}$, 下面还是简写成 $\delta\dot{\theta}$, 根据(A4)式可得:

$$\begin{aligned}
\dot{q}_{t_{true}} &= \frac{1}{2} q_{t_{true}} \otimes \begin{bmatrix} \omega_{true} \\ 0 \end{bmatrix} = \frac{1}{2} q_t \otimes \delta q \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} - n_\omega - \delta b_{\omega_t} \\ 0 \end{bmatrix} \\
\dot{q}_{t_{nominal}} &= \dot{q}_t = \frac{1}{2} q_t \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix}
\end{aligned}$$

根据导数性质, 又有:

$$\dot{q}_{t_{true}} = (q_t \otimes \delta q) = \dot{q}_t \otimes \delta q + q_t \otimes \dot{\delta q}$$

$$= \frac{1}{2} q_t \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix} \otimes \delta q + q_t \otimes \delta \dot{q}$$

综合上式，可得等式如下：

$$\begin{aligned} \frac{1}{2} q_t \otimes \delta q \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} - n_{\omega} - \delta b_{\omega_t} \\ 0 \end{bmatrix} &= \frac{1}{2} q_t \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix} \otimes \delta q + q_t \otimes \delta \dot{q} \\ \Leftrightarrow \frac{1}{2} \delta q \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} - n_{\omega} - \delta b_{\omega_t} \\ 0 \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix} \otimes \delta q + \delta \dot{q} \\ \Leftrightarrow 2\delta \dot{q} &= \delta q \otimes \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} - n_{\omega} - \delta b_{\omega_t} \\ 0 \end{bmatrix} - \begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix} \otimes \delta q \\ 2\delta \dot{q} &= \mathcal{R} \left(\begin{bmatrix} \hat{\omega}_t - b_{\omega_t} - n_{\omega} - \delta b_{\omega_t} \\ 0 \end{bmatrix} \right) \delta q - \mathcal{L} \left(\begin{bmatrix} \hat{\omega}_t - b_{\omega_t} \\ 0 \end{bmatrix} \right) \delta q \\ 2\delta \dot{q} &= \begin{bmatrix} -(2\hat{\omega}_t - 2b_{\omega_t} - n_{\omega} - \delta b_{\omega_t})^\wedge & -n_{\omega} - \delta b_{\omega_t} \\ (n_{\omega} + \delta b_{\omega_t})^T & 0 \end{bmatrix} \begin{bmatrix} \delta \theta \\ \frac{2}{1} \end{bmatrix} \end{aligned}$$

上式左侧也可以写成：

$$\begin{aligned} 2\delta \dot{q} = \begin{bmatrix} \delta \dot{\theta} \\ 0 \end{bmatrix} &= \begin{bmatrix} -(2\hat{\omega}_t - 2b_{\omega_t} - n_{\omega} - \delta b_{\omega_t})^\wedge & -n_{\omega} - \delta b_{\omega_t} \\ (n_{\omega} + \delta b_{\omega_t})^T & 0 \end{bmatrix} \begin{bmatrix} \delta \theta \\ \frac{2}{1} \end{bmatrix} \\ \delta \dot{\theta} &= -(2\hat{\omega}_t - 2b_{\omega_t} - n_{\omega} - \delta b_{\omega_t})^\wedge \frac{\delta \theta}{2} - n_{\omega} - \delta b_{\omega_t} \\ &\approx -(\hat{\omega}_t - b_{\omega_t})^\wedge \delta \theta - n_{\omega} - \delta b_{\omega_t} \end{aligned} \tag{A6}$$

10.3 IMU 离散误差动力学方程推导

下面我们对公式（15）的 IMU 离散的误差动力学方程进行详细推导：

1) $\delta \theta_{k+1}$ ：

由公式(A6)可知，角度误差导数的连续形式为：

$$\delta \dot{\theta} = -(\hat{\omega}_t - b_{\omega_t})^\wedge \delta \theta - n_{\omega} - \delta b_{\omega_t}$$

则离散形式为：

$$\delta \dot{\theta}_k = - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta \theta_k - \frac{n_{\omega_k} + n_{\omega_{k+1}}}{2} - \delta b_{\omega_k}$$

根据导数定义可得下一个离散时刻的误差为：

$$\delta \theta_{k+1} = \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \delta \theta_k - \frac{n_{\omega_k} + n_{\omega_{k+1}}}{2} \delta t - \delta t \delta b_{\omega_k} \tag{A7}$$

2) $\delta \beta_{k+1}$ ：

由公式(A5)可得，速度误差导数的连续形式为：

$$\delta \dot{\beta} = -R_t^{b_k} (\hat{a}_t - b_{a_t})^\wedge \delta \theta - R_t^{b_k} \delta b_{a_t} - R_t^{b_k} n_a$$

则离散形式为：

$$\begin{aligned}\delta\dot{\beta}_k = & -\frac{1}{2}R_k(\hat{a}_k - b_{a_k})^\wedge \delta\theta_k - \frac{1}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \delta\theta_{k+1} - \frac{1}{2}(R_k + R_{k+1})\delta b_{a_k} \\ & - \frac{1}{2}R_k n_{a_k} - \frac{1}{2}R_{k+1} n_{a_{k+1}}\end{aligned}$$

将公式(A7)代入上式可得：

$$\begin{aligned}\delta\dot{\beta}_k = & -\frac{1}{2}R_k(\hat{a}_k - b_{a_k})^\wedge \delta\theta_k - \frac{1}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \\ & \left\{ \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \delta\theta_k - \frac{n_{\omega_k} + n_{\omega_{k+1}}}{2} \delta t - \delta t \delta b_{\omega_k} \right\} \\ & - \frac{1}{2}(R_k + R_{k+1})\delta b_{a_k} - \frac{1}{2}R_k n_{a_k} - \frac{1}{2}R_{k+1} n_{a_{k+1}} \\ = & \left\{ -\frac{1}{2}R_k(\hat{a}_k - b_{a_k})^\wedge - \frac{1}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \right\} \delta\theta_k \\ & + \frac{\delta t}{4}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge n_{\omega_k} + \frac{\delta t}{4}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge n_{\omega_{k+1}} + \frac{\delta t}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \delta b_{\omega_k} \\ & - \frac{1}{2}(R_k + R_{k+1})\delta b_{a_k} - \frac{1}{2}R_k n_{a_k} - \frac{1}{2}R_{k+1} n_{a_{k+1}}\end{aligned}$$

则下一个时刻的速度误差为：

$$\begin{aligned}\delta\beta_{k+1} = & f_{21}\delta\theta_k + \delta\beta_k - \frac{1}{2}(R_k + R_{k+1})\delta t \delta b_{a_k} + f_{24}\delta b_{\omega_k} - \frac{1}{2}R_k \delta t n_{a_k} \\ & + v_{21}n_{\omega_k} - \frac{1}{2}R_{k+1}\delta t n_{a_{k+1}} + v_{23}n_{\omega_{k+1}}\end{aligned}\quad (A8)$$

其中：

$$\begin{aligned}f_{21} = & -\frac{1}{2}R_k(\hat{a}_k - b_{a_k})^\wedge \delta t - \frac{1}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \left[I - \left(\frac{\hat{\omega}_k + \hat{\omega}_{k+1}}{2} - b_{\omega_k} \right)^\wedge \delta t \right] \delta t \\ f_{24} = & \frac{1}{2}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^2 \\ v_{21} = v_{23} = & \frac{1}{4}R_{k+1}(\hat{a}_{k+1} - b_{a_k})^\wedge \delta t^2\end{aligned}$$

3) $\delta\alpha_{k+1}$ ：

由公式(5)可得，位置误差导数的连续形式为：

$$\delta\dot{\alpha} = \delta\beta$$

则离散形式为：

$$\begin{aligned}\delta\dot{\alpha}_k = & \frac{1}{2}\delta\beta_k + \frac{1}{2}\delta\beta_{k+1} \\ = & \delta\beta_k + \frac{1}{2}f_{21}\delta\theta_k - \frac{1}{4}(R_k + R_{k+1})\delta t \delta b_{a_k} + \frac{1}{2}f_{24}\delta b_{\omega_k} - \frac{1}{4}R_k \delta t n_{a_k} \\ & + \frac{1}{2}v_{21}n_{\omega_k} - \frac{1}{4}R_{k+1}\delta t n_{a_{k+1}} + \frac{1}{2}v_{23}n_{\omega_{k+1}}\end{aligned}$$

则下一个时刻的位置误差为：

$$\begin{aligned}\delta\alpha_{k+1} = & \delta\alpha_k + \frac{\delta t}{2}f_{21}\delta\theta_k + \delta t\delta\beta_k - \frac{1}{4}(R_k + R_{k+1})\delta t^2\delta b_{a_k} + \frac{\delta t}{2}f_{24}\delta b_{\omega_k} - \frac{1}{4}R_k\delta t^2n_{a_k} \\ & + \frac{\delta t}{2}v_{21}n_{\omega_k} - \frac{1}{4}R_{k+1}\delta t^2n_{a_{k+1}} + \frac{\delta t}{2}v_{23}n_{\omega_{k+1}}\end{aligned}\quad (A9)$$

10.4 IMU 角度误差对 k 时刻角度误差的 Jacobian 推导

下面我们推导公式 (21) 中 IMU 预积分出的相邻帧角度误差项，关于第 k 时刻的角度误差

的 Jacobian $\frac{\partial\delta\theta_{b_{k+1}}^{b_k}}{\partial q_{b_k}^w}$ ：

$$\begin{aligned}\frac{\partial\delta\theta_{b_{k+1}}^{b_k}}{\partial q_{b_k}^w} &= 2 \lim_{\delta\theta_{b_k}^w \rightarrow 0} \frac{\gamma_{b_{k+1}}^{b_k}{}^{-1} \otimes \left(q_{b_k}^w \otimes \left[\frac{1}{\delta\theta_{b_k}^w} \right] \right)^{-1} \otimes q_{b_{k+1}}^w - \gamma_{b_{k+1}}^{b_k}{}^{-1} \otimes \left(q_{b_k}^w \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^{-1} \otimes q_{b_{k+1}}^w}{\delta\theta_{b_k}^w} \\ &= 2 \lim_{\delta\theta_{b_k}^w \rightarrow 0} \frac{\gamma_{b_{k+1}}^{b_k}{}^{-1} \otimes \left[-\frac{1}{\delta\theta_{b_k}^w} \right] \otimes q_{b_k}^w{}^{-1} \otimes q_{b_{k+1}}^w - \gamma_{b_{k+1}}^{b_k}{}^{-1} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes q_{b_k}^w{}^{-1} \otimes q_{b_{k+1}}^w}{\delta\theta_{b_k}^w} \\ &= 2 \lim_{\delta\theta_{b_k}^w \rightarrow 0} \frac{\mathcal{R}[q_{b_k}^w{}^{-1} \otimes q_{b_{k+1}}^w] \mathcal{L}[\gamma_{b_{k+1}}^{b_k}{}^{-1}] \left(\begin{bmatrix} 1 \\ -\frac{1}{\delta\theta_{b_k}^w} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)}{\delta\theta_{b_k}^w}\end{aligned}\quad (A10)$$

下面我们讨论一个性质，令 $q = [x \ y \ z \ s] = [\omega \ s]$ ，我们把公式(4)重写到这里：

$$\begin{aligned}\mathcal{R}(q) &= \Omega(\omega) + sI_{4 \times 4} = \begin{bmatrix} -\omega^\wedge & \omega \\ -\omega^T & 0 \end{bmatrix} + sI_{4 \times 4} \\ \mathcal{L}(q) &= \Psi(\omega) + sI_{4 \times 4} = \begin{bmatrix} \omega^\wedge & \omega \\ -\omega^T & 0 \end{bmatrix} + sI_{4 \times 4}\end{aligned}$$

则有：

$$\mathcal{R}(q^{-1}) = \Omega(-\omega) + sI_{4 \times 4} = \begin{bmatrix} \omega^\wedge & -\omega \\ \omega^T & 0 \end{bmatrix} + sI_{4 \times 4}$$

如果我们只取左上角的 3×3 的虚部部分，则有：

$$\mathcal{R}(q^{-1})_{3 \times 3} = sI_{3 \times 3} + \omega^\wedge = \mathcal{L}(q)_{3 \times 3} \quad (A11)$$

将上式代入(20)式可得：

$$\begin{aligned}\frac{\partial\delta\theta_{b_{k+1}}^{b_k}}{\partial q_{b_k}^w} &= \lim_{\delta\theta_{b_k}^w \rightarrow 0} \frac{\left\{ \mathcal{L}[q_{b_{k+1}}^w{}^{-1} \otimes q_{b_k}^w] \mathcal{R}[\gamma_{b_{k+1}}^{b_k}] \begin{bmatrix} 0 \\ -\delta\theta_{b_k}^w \end{bmatrix} \right\}_{3 \times 3}}{\delta\theta_{b_k}^w} \\ &= -\mathcal{L}[q_{b_{k+1}}^w{}^{-1} \otimes q_{b_k}^w] \mathcal{R}[\gamma_{b_{k+1}}^{b_k}]\end{aligned}$$

证毕！

10.5 IMU 角度误差对 k 时刻 bias 的 Jacobian 推导

下面推导公式 (22) 中的 $\frac{\partial \delta \theta_{b_{k+1}}^{b_k}}{\partial b_{\omega_k}}$

$$\begin{aligned}
 \frac{\partial \delta \theta_{b_{k+1}}^{b_k}}{\partial b_{\omega_k}} &= 2 \lim_{\rightarrow 0} \frac{\left[\gamma_{b_{k+1}}^{b_k} \otimes \left[\frac{1}{2} J_{b_{\omega}}^{\gamma} \delta b_{\omega} \right] \right]^{-1} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w - \left[\gamma_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right]^{-1} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w}{\delta b_{\omega_k}} \\
 &= 2 \lim_{\rightarrow 0} \frac{\begin{bmatrix} 0 \\ -\frac{1}{2} J_{b_{\omega}}^{\gamma} \delta b_{\omega} \end{bmatrix} \otimes \gamma_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w}{\delta b_{\omega_k}} \\
 &= -\mathcal{R} \left[\gamma_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w \right] \begin{bmatrix} 0 \\ J_{b_{\omega}}^{\gamma} \end{bmatrix} \\
 &= -\mathcal{L} \left[q_{b_{k+1}}^{w^{-1}} \otimes q_{b_k}^w \otimes \gamma_{b_{k+1}}^{b_k} \right]_{3 \times 3} J_{b_{\omega}}^{\gamma}
 \end{aligned}$$

10.6 IMU 角度误差对 k+1 时刻的角度误差 Jacobian 推导

下面我们重点推导 $\frac{\partial \delta \theta_{b_{k+1}}^{b_k}}{\partial q_{b_{k+1}}^w}$:

$$\begin{aligned}
 \frac{\partial \delta \theta_{b_{k+1}}^{b_k}}{\partial q_{b_{k+1}}^w} &= 2 \lim_{\delta \theta_{b_k}^w \rightarrow 0} \frac{\gamma_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w \otimes \left[\frac{1}{2} \frac{\delta \theta_{b_{k+1}}^w}{\delta \theta_{b_k}^w} \right] - \gamma_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{\delta \theta_{b_{k+1}}^w} \\
 &= \mathcal{L} \left[\gamma_{b_{k+1}}^{b_k^{-1}} \otimes q_{b_k}^{w^{-1}} \otimes q_{b_{k+1}}^w \right]
 \end{aligned}$$

10.7 视觉误差项推导

公式 (27) 中 P 在第 i 个相机的像素坐标系下坐标为:

$$\begin{aligned}
 P_{uv_i} &= \lambda_l \pi_c (T_{b \leftarrow c}^{-1} T_{w \leftarrow b_i}^{-1} P_{w_l}) \\
 \Leftrightarrow P_{w_l} &= T_{w \leftarrow b_i} T_{b \leftarrow c} \frac{1}{\lambda_l} \pi_c^{-1} (P_{uv_i}) \\
 P_{w_l} &= R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \pi_c^{-1} \left(\begin{bmatrix} u_i^{c_i} \\ v_i^{c_i} \end{bmatrix} \right) + p_c^b \right) + p_{b_i}^w \quad (A12)
 \end{aligned}$$

P 在第 j 个相机的相机坐标系下坐标为:

$$\begin{aligned}
 P_l^{c_j} &= T_{b \leftarrow c}^{-1} T_{w \leftarrow b_j}^{-1} P_{w_l} \\
 \Leftrightarrow P_{w_l} &= T_{w \leftarrow b_j} T_{b \leftarrow c} P_l^{c_j} \\
 P_{w_l} &= R_{b_j}^w (R_c^b P_l^{c_j} + p_c^b) + p_{b_j}^w \quad (A13)
 \end{aligned}$$

将(A12)代入(A13)可得:

$$\begin{aligned}
R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \pi_c^{-1} \left(\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix} \right) + p_c^b \right) + p_{b_i}^w &= R_{b_j}^w (R_c^b P_l^{c_j} + p_c^b) + p_{b_j}^w \\
\Rightarrow P_l^{c_j} &= R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \pi_c^{-1} \left(\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix} \right) + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \\
&= R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\}
\end{aligned} \tag{A14}$$

10.8 视觉误差项的 Jacobian 推导

公式 (28) 部分推导如下：

- $\bullet \quad \frac{\partial r_c}{\partial q_{b_i}^w}:$

$$\begin{aligned}
\frac{\partial r_c}{\partial q_{b_i}^w} &= \lim_{\delta \theta_{b_i}^w \rightarrow 0} \frac{R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \exp(\delta \theta_{b_i}^{w\wedge}) \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} - P_l^{c_j}}{\delta \theta_{b_i}^w} \\
&= \lim_{\delta \theta_{b_i}^w \rightarrow 0} \frac{R_b^c \left\{ R_w^{b_j} \left[-R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right)^\wedge \delta \theta_{b_i}^w \right] \right\}}{\delta \theta_{b_i}^w} \\
&= -R_b^c R_w^{b_j} R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right)^\wedge
\end{aligned}$$
- $\bullet \quad \frac{\partial r_c}{\partial q_{b_j}^w}:$

$$\begin{aligned}
\frac{\partial r_c}{\partial q_{b_j}^w} &= \lim_{\delta \theta_{b_j}^w \rightarrow 0} \frac{R_b^c \left\{ \left[R_{b_j}^w \exp(\delta \theta_{b_j}^{w\wedge}) \right]^{-1} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} - P_l^{c_j}}{\delta \theta_{b_j}^w} \\
&= \lim_{\delta \theta_{b_j}^w \rightarrow 0} \frac{R_b^c \left\{ -\delta \theta_{b_j}^{w\wedge} R_{b_j}^{w-1} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] \right\}}{\delta \theta_{b_j}^w} \\
&= \lim_{\delta \theta_{b_i}^w \rightarrow 0} \frac{R_b^c \left\{ \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] \right\}^\wedge \delta \theta_{b_j}^w \right\}}{\delta \theta_{b_i}^w} \\
&= R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] \right\}^\wedge
\end{aligned}$$
- $\bullet \quad \frac{\partial r_c}{\partial q_c^b}:$

$$\begin{aligned}
\frac{\partial r_c}{\partial q_c^b} &= \lim_{\delta \theta_c^b \rightarrow 0} \frac{\left[R_c^b \exp(\delta \theta_c^{b\wedge}) \right]^{-1} \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \exp(\delta \theta_c^{b\wedge}) \frac{1}{\lambda_l} \bar{P}_l^{c_i} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} - P_l^{c_j}}{\delta \theta_c^b}
\end{aligned}$$

上式分子可分解为：

$$\begin{aligned}
& R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \delta \theta_c^{b \wedge} \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \\
& \quad - \delta \theta_c^{b \wedge} R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b (I + \delta \theta_c^{b \wedge}) \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \\
& \approx R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \delta \theta_c^{b \wedge} \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \\
& \quad - \delta \theta_c^{b \wedge} R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\}
\end{aligned}$$

代回到原式中可得：

$$\begin{aligned}
\frac{\partial r_c}{\partial q_c^b} &= -R_b^c R_w^{b_j} R_{b_i}^w R_c^b \left(\frac{\bar{P}_l^{c_i}}{\lambda_l} \right)^\wedge + \left\{ R_b^c \left\{ R_w^{b_j} \left[R_{b_i}^w \left(R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b \right) + p_{b_i}^w - p_{b_j}^w \right] - p_c^b \right\} \right\}^\wedge \\
&= -R_b^c R_w^{b_j} R_{b_i}^w R_c^b \left(\frac{\bar{P}_l^{c_i}}{\lambda_l} \right)^\wedge + \left(R_b^c R_w^{b_j} R_{b_i}^w R_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} \right)^\wedge + \left\{ R_b^c \left[R_w^{b_j} \left(R_{b_i}^w p_c^b + p_{b_i}^w - p_{b_j}^w \right) - p_c^b \right] \right\}^\wedge
\end{aligned}$$