Visual-Inertial Mapping with Non-Linear Factor Recovery [2] 的论文阅读

王鹏，陈勇南

August 2019

1 IMU 预积分

IMU 的预积分是 Visual-Inertial Odometry 或者 Lidar-Inertial Odometry 的重要组成部分，其主要的原理就是将 IMU 预积分得到的相对位姿与视觉或者激光估计的帧间相对位姿构造 IMU 测量残差，并联合视觉构造的视觉残差（重投影误差或者光度误差）或者激光构造的 ICP 残差来估计 IMU 的 PQV(Position、Orientation、Velocity) 和偏置 (b_a 和 b_g) 和特征点的参数（有的只优化特征点在 host frame 中的 inverse depth 如 VINS-mono 或者优化特征点在 host frame 的 direction 和 inverse depth）。

IMU 预积分的方法 IMU 预积分的方法采用的数学公式大致分为两类 $SO3+T(3)$ 或者 $Quaterion+T(3)$, VI-Mapping 采用的是前者，而 VINS-mono 采用的是后者，因此下面将主要介绍采用 $SO3+T(3)$ 的预积分方法，对于 Quaterion+T(3) 不进行过多的讨论。下面我将介绍 IMU 的测量模型、IMU 的预积分的均值和协方差的的数学推导。

1.1 IMU 的测量模型

公式 (1)、(2) 就是 IMU 的加速度和角速度的测量模型，测量值受到了偏置 (b_a 和 b_g) 和测量噪声 (n_a 和 n_g) 的影响。

$$a_m = R_G^I(a-g) + b_a + n_a \tag{1}$$

$$w_m = w + b_w + n_w \tag{2}$$

1.2 IMU 预积分的均值、状态转移矩阵和控制矩阵的计算

为了对相邻帧 t 和 j 的相对位姿态进行约束, 我们通过预积分的方法将对应区间的高频的 IMU 的数据进行积分得到 $\Delta s = (\Delta R, \Delta v, \Delta p)$ 作为对其的一个伪观测。为了积分更加准确, 我们使用去除偏置的减速度 $a_t = a_t^m - b_t^a$ 和 $w_t = w_t^m - b_t^g$ 数据进行积分。为了计算方便，我们假设整个积分区间 IMU 的偏置不变，即 $b_t^a = b_t^a$ 和 $w_t = b_t^g = b_t^g$，同时为了避免每次迭代优化完后，由于 IMU 的偏置发生改变导致需要重新积分，我们通过求解预积分对 IMU 偏置的雅克比矩阵来一阶近似 IMU 偏置改变对预积分的影响。

对于第 t 帧对应的时间戳 t_i, 将对应的初始状态 $\Delta s = (I_{3\times3}, 0, 0)$, 然后对于每一个 IMU 的时间戳 t 满足 $t_i < t <= t_j$, 采用如下公式进行更新。

$$\Delta R_{t+1} = \Delta R_t Exp(w_{t+1}\Delta t) \tag{3}$$

$$\Delta R_{\frac{2t+1}{2}} = \Delta R_t Exp(0.5 w_{t+1}\Delta t) \tag{4}$$

$$\Delta v_{t+1} = \Delta v_t + \Delta R_{\frac{2t+1}{2}} a_{t+1}\Delta t \tag{5}$$

$$\Delta p_{t+1} = \Delta p_t + \Delta v_t \Delta t + 0.5\Delta R_{\frac{2t+1}{2}} a_{t+1}\Delta t^2 \tag{6}$$

首先我们求解状态转移矩阵 $\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}$

$$\frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta p_t} = I_{3\times3} \tag{7}$$

$$\frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta v_t} = \Delta t \tag{8}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta R_t} &= \frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}} a_{t+1}} \frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}} a_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}}} \frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}}}{\mathrm{d}\Delta R_t} \\ &= 0.5\Delta t^2 * \left[-\Delta R_{\frac{2t+1}{2}} a_{t+1}\right]^{\times} I_{3\times3}\end{aligned} \tag{9}$$

$$\frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}\Delta v_t} = I_{3\times3} \tag{10}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}\Delta R_t} &= \frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}} a_{t+1}} \frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}} a_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}}} \frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}}}{\mathrm{d}\Delta R_t} \\ &= \Delta t \left[-\Delta R_{\frac{2t+1}{2}} a_{t+1}\right]^{\times} I_{3\times3}\end{aligned} \tag{11}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta R_{t+1}}{\mathrm{d}\Delta R_t} &= \frac{\ln((exp(\delta\theta^{\times})\Delta R_t Exp(w_{t+1}\Delta t))(\Delta R_t Exp(w_{t+1}\Delta t))^{-1})}{\delta\theta} \\ &= I_{3\times3}\end{aligned} \tag{12}$$

注意对公式 (12)$\frac{\mathrm{d}\Delta R_{t+1}}{\mathrm{d}\Delta R_t}$ 的求解过程中是分别对 ΔR_{t+1}、ΔR_t 施加左扰动 $Exp(\delta\theta_1)$ 和 $Exp(\delta\theta_2)$。接下来将推导如何求解控制矩阵 $\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta w_{t+1}}$ 和 $\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta a_{t+1}}$。注意下面推到过程中对 $Exp(w_{t+1}\Delta t)$, $Exp(0.5w_{t+1}\Delta t)$ 施加的是右扰动。

$$\frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta a_{t+1}} = 0.5 * R_{\frac{2t+1}{2}}\Delta t^2 \tag{13}$$

$$\frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}\Delta a_{t+1}} = R_{\frac{2t+1}{2}}\Delta t \tag{14}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta w_{t+1}} &= \frac{\mathrm{d}\Delta p_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}}a_{t+1}}\frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}}a_{t+1}}{\mathrm{d}\Delta R_{\frac{2t+1}{2}}}\frac{\mathrm{d}\Delta R_{\frac{2t+1}{2}}}{\mathrm{d}Exp(0.5w_{t+1}\Delta t)} \\ &* \frac{\mathrm{d}Exp(0.5w_{t+1}\Delta t)}{\mathrm{d}(0.5w_{t+1}\Delta t)}\frac{\mathrm{d}(0.5w_{t+1}\Delta t)}{\mathrm{d}w_{t+1}} \\ &= 0.5\Delta t^2\left[-\Delta R_{\frac{2t+1}{2}}a_{t+1}\right]^{\times} * \\ &\frac{\ln((\Delta R_t Exp(0.5w_{t+1}\Delta t)exp(\delta\theta^{\times}))(\Delta R_t Exp(0.5w_{t+1}\Delta t))^{-1})}{\delta\theta} * \\ &J_r(0.5w_{t+1}\Delta t) * 0.5\Delta t \\ &= 0.5\Delta t^2\left[-\Delta R_{\frac{2t+1}{2}}a_{t+1}\right]^{\times}\Delta R_{\frac{2t+1}{2}}J_r(0.5w_{t+1}\Delta t) * 0.5\Delta t\end{aligned} \tag{15}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}\Delta w_{t+1}} &= \frac{\mathrm{d}\Delta v_{t+1}}{\mathrm{d}R_{\frac{2t+1}{2}}a_{t+1}}\frac{\mathrm{d}R_{\frac{2t+1}{2}}a_{t+1}}{\mathrm{d}w_{t+1}} \\ &= \Delta t\left[-\Delta R_{\frac{2t+1}{2}}a_{t+1}\right]^{\times}\Delta R_{\frac{2t+1}{2}}J_r(0.5w_{t+1}\Delta t) * 0.5\Delta t\end{aligned} \tag{16}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta R_{t+1}}{\mathrm{d}\Delta w_{t+1}} &= \frac{\mathrm{d}\Delta R_{t+1}}{\mathrm{d}Exp(w_{t+1}\Delta t)}\frac{\mathrm{d}Exp(w_{t+1}\Delta t)}{\mathrm{d}(w_{t+1}\Delta t)}\frac{\mathrm{d}(w_{t+1}\Delta t)}{\mathrm{d}(w_{t+1})} \\ &= \Delta R_{t+1}J_r(w_{t+1}\Delta t)w_{t+1}\end{aligned} \tag{17}$$

1.3 IMU 的预积分的协方差和对偏置 b_i^a、b_i^g 雅克比矩阵的计算

首先我们将推导 IMU 的预积分的协方差矩阵的计算

$$\begin{aligned}cov(\Delta s_{t+1}) = &\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}cov(\Delta s_t)\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}^T + \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta w_{t+1}}cov(w_{t+1})\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta w_{t+1}}^T + \\ &\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta a_{t+1}}cov(a_{t+1})\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta a_{t+1}}^T\end{aligned} \tag{18}$$

然后我们推导 IMU 预积分对 b_i^a、b_i^g 的雅克比矩阵

$$\begin{aligned}\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}b_i^a} &= \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}\frac{\mathrm{d}\Delta s_t}{\mathrm{d}b_i^a} + \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}a_{t+1}}\frac{\mathrm{d}a_{t+1}}{\mathrm{d}b_i^a} \\ &= \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}\frac{\mathrm{d}\Delta s_t}{\mathrm{d}b_i^a} - \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}a_{t+1}}\end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}b_i^g} &= \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}\frac{\mathrm{d}\Delta s_t}{\mathrm{d}b_i^g} + \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}w_{t+1}}\frac{\mathrm{d}w_{t+1}}{\mathrm{d}b_i^g} \\ &= \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}\Delta s_t}\frac{\mathrm{d}\Delta s_t}{\mathrm{d}b_i^g} - \frac{\mathrm{d}\Delta s_{t+1}}{\mathrm{d}w_{t+1}}\end{aligned} \tag{20}$$

1.4 IMU 预积分残差的构建

当 IMU 的偏置 b_i^a、b_i^g 发生改变时，IMU 新的预积分 $\Delta\tilde{s}(\tilde{b}_i^a, \tilde{b}_i^g) = \Delta s(b_i^a, b_i^g) \oplus (J^a\epsilon^a + J^g\epsilon^g))$, 其中 $\tilde{b}_i^a = b_i^a + \epsilon^a$、$\tilde{b}_g^a = b_g^a + \epsilon^g$,

$$\Delta\tilde{p} = \Delta p + J_g^p\epsilon^g + J_a^p\epsilon^a \tag{21}$$

$$\Delta\tilde{v} = \Delta v + J_g^v\epsilon^g + J_a^v\epsilon^a \tag{22}$$

$$\Delta\tilde{R} = Exp(J_g^R\epsilon^g)\Delta R \tag{23}$$

IMU 的预积分与其对应的积分区间对应的视觉位姿构造的残差如下，其中 Δt 表示的第 i 帧到第 j 帧之间的时间差，还有重力 g 是在世界坐标系下表示的:

$$r_{\Delta R} = \ln(\Delta\tilde{R}R_j^T R_i) \tag{24}$$

$$r_{\Delta v} = R_i^T(v_j - v_i - g\Delta t) - \Delta\tilde{v} \tag{25}$$

$$r_{\Delta p} = R_i^T(p_j - p_i - v_i\Delta t - 0.5g\Delta t^2) - \Delta\tilde{p} \tag{26}$$

针对 IMU 预积分时，IMU 的偏置在积分区间不变的假设，再增加两个约束 (27)、(28)，注意这两个残差中的 $\frac{1}{\Delta t}$ 是误差的权重项，表示 IMU 预积分区间越长，偏置相等的假设可以放松。

$$r_{b^a} = \frac{b_j^a - b_i^a}{\Delta t} \tag{27}$$

$$r_{b^g} = \frac{b_j^g - b_i^g}{\Delta t} \tag{28}$$

接下来我们将推导 IMU 误差对 IMU 状态 ($P\ R\ V\ b_a\ b_g$) 的导数。

$$\frac{\mathrm{d}r_{\Delta p}}{\mathrm{d}p_i} = -R_i^T \tag{29}$$

$$\begin{aligned}\frac{\mathrm{d}r_{\Delta p}}{\mathrm{d}R_i} &= \frac{\mathrm{d}R_i^T(p_j - p_i - v_i\Delta t - 0.5g\Delta t^2)}{\mathrm{d}R_i} \\ &= \frac{(exp(\delta\theta^\times)R_i)^T(p_j - p_i - v_i\Delta t - 0.5g\Delta t^2) - R_i^T(p_j - p_i - v_i\Delta t - 0.5g\Delta t^2)}{\delta\theta} \\ &= R_i^T\left[p_j - p_i - v_i\Delta t - 0.5g\Delta t^2\right]^\times \\ &= \left[R_i^T(p_j - p_i - v_i\Delta t - 0.5g\Delta t^2)\right]^\times R_i^T\end{aligned} \tag{30, 31}$$

$$\frac{\mathrm{d}r_{\Delta p}}{\mathrm{d}v_i} = -R_i^T\Delta t \tag{32}$$

$$\begin{aligned}\frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}R_i} &= \frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}\Delta\tilde{R}R_j^TR_i}\frac{\mathrm{d}\Delta\tilde{R}R_j^TR_i}{\mathrm{d}R_i} \\ &= J_r^{-1}(r_{\Delta R})\frac{\ln((\Delta\tilde{R}R_j^TR_i)^{-1}\Delta\tilde{R}R_j^Texp(\delta\theta^\times)R_i)}{\delta\theta} \\ &= J_r^{-1}(r_{\Delta R})R_i^T\end{aligned} \tag{33}$$

$$\begin{aligned}\frac{\mathrm{d}r_{\Delta v}}{\mathrm{d}R_i} &= \frac{\mathrm{d}R_i^T(v_j - v_i - g\Delta t)}{\mathrm{d}R_i} \\ &= \frac{(exp(\delta\theta^\times)R_i)^T(v_j - v_i - g\Delta t) - R_i^T(v_j - v_i - g\Delta t)}{\delta\theta} \\ &= R_i^T\left[v_j - v_i - g\Delta t\right]^\times \\ &= \left[R_i^T(v_j - v_i - g\Delta t)\right]^\times R_i^T\end{aligned} \tag{34}$$

$$\frac{\mathrm{d}r_{\Delta v}}{\mathrm{d}v_i} = -R_i^T \tag{35}$$

$$\frac{\mathrm{d}r_{\Delta p}}{\mathrm{d}p_j} = R_i^T \tag{36}$$

$$\frac{\mathrm{d}r_{\Delta v}}{\mathrm{d}v_j} = R_i^T \tag{37}$$

$$\begin{aligned}\frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}R_j} &= \frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}\Delta\tilde{R}R_j^TR_i}\frac{\mathrm{d}\Delta\tilde{R}R_j^TR_i}{\mathrm{d}R_j} \\ &= J_r^{-1}(r_{\Delta R})\frac{\ln((\Delta\tilde{R}R_j^TR_i)^{-1}\Delta\tilde{R}(exp(\delta\theta^\times)R_j)^TR_i)}{\delta\theta} \\ &= -J_r^{-1}(r_{\Delta R})R_i^T\end{aligned} \tag{38}$$

$$\frac{\mathrm{d}\left[r_{\Delta p}, r_{\Delta R}, r_{\Delta v}\right]}{b_i^a} = \frac{\mathrm{d}\left[r_{\Delta p}, r_{\Delta R}, r_{\Delta v}\right]}{\mathrm{d}\Delta\tilde{s}}\frac{\mathrm{d}\Delta\tilde{s}}{\mathrm{d}b_i^a} = -\frac{\mathrm{d}\Delta\tilde{s}}{\mathrm{d}b_i^a} \tag{39}$$

$$\frac{\mathrm{d}r_{\Delta p}}{b_i^w} = \frac{\mathrm{d}r_{\Delta p}}{\mathrm{d}\Delta\tilde{p}} * \frac{\mathrm{d}\Delta\tilde{p}}{\mathrm{d}b_i^g} = -\frac{\mathrm{d}\Delta\tilde{p}}{\mathrm{d}b_i^g} \tag{40}$$

$$\begin{aligned}\frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}b_i^g} &= \frac{\mathrm{d}r_{\Delta R}}{\mathrm{d}\Delta\tilde{R}R_j^T R_i}\frac{\mathrm{d}\Delta\tilde{R}R_j^T R_i}{\mathrm{d}\Delta\tilde{R}}\frac{\mathrm{d}\Delta\tilde{R}}{\mathrm{d}b_i^g} \\ &= J_l^{-1}(r_{\Delta R})\frac{\ln(exp(\delta\theta^{\times})\Delta\tilde{R}R_j^T R_i(\Delta\tilde{R}R_j^T R_i)^{-1})}{\delta\theta}\frac{\mathrm{d}\Delta\tilde{R}}{\mathrm{d}b_i^g} \\ &= J_l^{-1}(r_{\Delta R})\frac{\mathrm{d}\Delta\tilde{R}}{\mathrm{d}b_i^g}\end{aligned} \tag{41}$$

2 视觉残差的构建

2.1 3d 点的参数化表达

VIO 常用的 3d 点的参数化表达常用的是: bearing vector 加 inverse depth。有的只优化 3D 点的 inverse depth(如 VINS-mono), 有的还优化 bearing vector(如 Visual Inertial mapping), 其中 3d 点的 bearing vector 大概可以分为 3 种 (我所了解的):

1.(VINS-Mono, MSCKF)

$$(x, y, z) = (\frac{u}{\sqrt{u^2+v^2+1}}, \frac{v}{\sqrt{u^2+v^2+1}}, \frac{1}{\sqrt{u^2+v^2+1}}) \tag{42}$$

2.(R-VIO)

$$(x, y, z) = (\cos\phi\sin\psi, \sin\phi, \cos\phi\cos\psi) \tag{43}$$

3.(VI Mapping)

$$(x, y, z) = (\eta u, \eta v, \eta - 1) \tag{44}$$

$$\eta = \frac{2}{1+u^2+v^2} \tag{45}$$

第三种的参数表达方式相对于第一种参数表达的优点是，没有开根号运算，求导更加方便。但是不能表达通过像素位置 $(0,0)$ 的负方向 (0,0，-1)，然而在实际应用中这个缺点可以忽略，因为大多数的相机的视野都是有限的，无法看到相机后面的景物。

2.2 视觉残差

Visual-Inertial Mapping 使用的是视觉残差是重投影误差，当 host frame $h(i)$ 拥有的一个特征点 $p_i(u, v, d)$ 在 target frame 观测到时，观测的像素

位置为 z_{it}, 那么对应的重投影误差的定义如下所示: 其中 $c(t,i)$ 表示的是 target frame 观察到特征点 p_i 的相机索引 (支持单目和双目相机)。$c(h,i)$ 表达的意思类似。

$$r_{it} = z_{it} - \pi_{c(t,i)}((\mathrm{T}_{tI}^{W}\mathrm{T}_{c(t,i)}^{tI})^{-1} * \mathrm{T}_{hI}^{W}\mathrm{T}_{c(h,i)}^{hI}p_i(u,v,d)) \tag{46}$$

$$T_{c(h,i)}^{c(t,i)} = (\mathrm{T}_{tI}^{W}\mathrm{T}_{c(t,i)}^{tI})^{-1}\mathrm{T}_{hI}^{W}\mathrm{T}_{c(h,i)}^{hI} \tag{47}$$

$$p_i(u,v,d) = (x(u,v), y(u,v), z(u,v), d) \tag{48}$$

用 $P_i^{c(t)}$ 表示 p_i 在相机 $c(t,i)$ 的其次坐标系下的 3D 点，下面将推导相机下的 3D 点对相对位姿的雅克比矩阵 $\frac{\mathrm{d}P_i^{c(t)}}{\mathrm{d}T_{c(t,i)}^{c(t,i)}}$ 和相对位姿对绝对位姿的雅克比矩阵 $\frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}^{SO3+T(3)}\mathrm{T}_t}$、$\frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}^{SO3+T(3)}\mathrm{T}_h}$ 。在具体推导之前，我将先介绍 $SE(3)$ 和 $SO3(3)+T(3)$ 扰动之间的变换关系, 在推导变换关系时，等式中 SE(3) 上的扰动用 $\delta\xi_1$ 表示，$SO(3)+T(3)$ 上的扰动 $\delta\xi_2$ 表示

$$\delta\xi_1 = \begin{bmatrix} \delta\rho_1 \\ \delta\phi_1 \end{bmatrix} \tag{49}$$

$$\delta\xi_2 = \begin{bmatrix} \delta t_2 \\ \delta\phi_2 \end{bmatrix} \tag{50}$$

$$\begin{aligned} Exp(\delta\xi_1)T &= \begin{bmatrix} (I + [\delta\phi_1]^{\times}) & \delta\rho_1 \\ 0 & I \end{bmatrix} \begin{bmatrix} R & t \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} (I + [\delta\phi_1]^{\times})R & (I + [\delta\phi_1]^{\times})t + \delta\rho_1 \\ 0 & I \end{bmatrix} \end{aligned} \tag{51}$$

$$Exp(\delta\xi_2)T = \begin{bmatrix} (I + [\delta\phi_2]^{\times})R & t + \delta t_2 \\ 0 & I \end{bmatrix} \tag{52}$$

$$Exp(\delta\xi_1)T = Exp(\delta\xi_2)T \tag{53}$$

联合 (51)、(52) 和 (53), 可以得出

$$\delta\phi_1 = \delta\phi_2 \tag{54}$$

$$[\delta\phi_1]^{\times} t + \delta\rho_1 = \delta t_2 \tag{55}$$

由 (55) 和 (54) 可以得到

$$\delta\rho_1 = \delta t_2 + [t]^{\times}\delta\phi_2 \tag{56}$$

所以

$$\begin{bmatrix}\delta\rho_1\\ \delta\phi_1\end{bmatrix} = \begin{bmatrix}I_{3\times3} & [t]^\times\\ 0 & I_{3\times3}\end{bmatrix}\begin{bmatrix}\delta t_2\\ \delta\phi_2\end{bmatrix} \tag{57}$$

$$\frac{\mathrm{d}P_i^{c(t)}}{\mathrm{d}T_{c(h,i)}^{c(t,i)}} = \begin{bmatrix}I_{3\times3}P_i[3] & -[P_i^{c(t)}[0:2]]^\times\\ 0^T & 0\end{bmatrix} \tag{58}$$

注意为了简化 (59) (60) 雅克比矩阵的求解，式子中对 $\frac{\mathrm{d}((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}{\mathrm{d}((\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}$ 求解时分别使用的是左扰动和右扰动模型。

$$\begin{aligned}\frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}^{SO3+T(3)}\mathrm{T}_t} &= \frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}\frac{\mathrm{d}((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}{\mathrm{d}((\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}\\ &* \frac{\mathrm{d}((\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}{\mathrm{d}\mathrm{T}_{tI}^{W}}\frac{\mathrm{d}\mathrm{T}_{tI}^{W}}{\mathrm{d}^{SO(3)+T(3)}\mathrm{T}_{tI}^{W}}\\ &= I_{6\times6}Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}) * -Ad((T_{tI}^{W})^{-1}) * \begin{bmatrix}I_{3\times3} & [t_{tI}^{W}]^\times\\ 0 & I_{3\times3}\end{bmatrix}\\ &= -Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}) * \begin{bmatrix}R_W^{tI} & [t_W^{tI}]^\times R_W^{tI}\\ 0 & R_W^{tI}\end{bmatrix}\begin{bmatrix}I_{3\times3} & [t_{tI}^{W}]^\times\\ 0 & I_{3\times3}\end{bmatrix}\\ &= -Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}) * \begin{bmatrix}R_W^{tI} & -[R_W^{tI}t_{tI}^{W}]^\times R_W^{tI}\\ 0 & R_W^{tI}\end{bmatrix}\begin{bmatrix}I_{3\times3} & [t_{tI}^{W}]^\times\\ 0 & I_{3\times3}\end{bmatrix}\\ &= -Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}) * \begin{bmatrix}R_W^{tI} & -R_W^{tI}[t_{tI}^{W}]^\times\\ 0 & R_W^{tI}\end{bmatrix}\begin{bmatrix}I_{3\times3} & [t_{tI}^{W}]^\times\\ 0 & I_{3\times3}\end{bmatrix}\\ &= -Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}) * \begin{bmatrix}R_W^{tI} & 0\\ 0 & R_W^{tI}\end{bmatrix}\end{aligned} \tag{59}$$

$$\frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}^{SO3+T(3)}\mathrm{T}_h} = \frac{\mathrm{d}T_{c(h,i)}^{c(t,i)}}{\mathrm{d}((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})} \frac{\mathrm{d}((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}{\mathrm{d}((\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})} \tag{60}$$

$$* \frac{\mathrm{d}((\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W})}{\mathrm{d}\mathrm{T}_{hI}^{W}} \frac{\mathrm{d}\mathrm{T}_{hI}^{W}}{\mathrm{d}^{SO(3)+T(3)}\mathrm{T}_{hI}^{W}}$$

$$= Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W}) * I_{6\times6} * Ad((T_{h_i}^{W})^{-1}) \begin{bmatrix} I_{3\times3} & [t_{hI}^{W}]^{\times} \\ 0 & I_{3\times3} \end{bmatrix}$$

$$= Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W}) * \begin{bmatrix} R_W^{hI} & [t_W^{hI}]^{\times} R_W^{hI} \\ 0 & R_W^{hI} \end{bmatrix} \begin{bmatrix} I_{3\times3} & [t_{hI}^{W}]^{\times} \\ 0 & I_{3\times3} \end{bmatrix}$$

$$= Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W}) * \begin{bmatrix} R_W^{hI} & -[R_W^{hI} t_{hI}^{W}]^{\times} R_W^{hI} \\ 0 & R_W^{hI} \end{bmatrix} \begin{bmatrix} I_{3\times3} & [t_{hI}^{W}]^{\times} \\ 0 & I_{3\times3} \end{bmatrix}$$

$$= Ad((\mathrm{T}_{c(t,i)}^{tI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W}) * \begin{bmatrix} R_W^{hI} & -R_W^{hI}[t_{hI}^{W}]^{\times} \\ 0 & R_W^{hI} \end{bmatrix} \begin{bmatrix} I_{3\times3} & [t_{hI}^{W}]^{\times} \\ 0 & I_{3\times3} \end{bmatrix}$$

$$= Ad((\mathrm{T}_{c(t,i)}^{hI})^{-1}(\mathrm{T}_{tI}^{W})^{-1}\mathrm{T}_{hI}^{W}) * \begin{bmatrix} R_W^{hI} & 0 \\ 0 & R_W^{hI} \end{bmatrix}$$

2.3 程序的解读

Visual-Inertial Mapping 的 VIO 部分主要的两个线程: 特征点追踪线程和后端优化线程，这里我们主要讲的是后端优化线程。为了叙述更加方便，我简要介绍一下 Visual-Inertial Mapping 的主要的数据结构。

2.3.1 前端特征追踪的结果的数据类型

```
struct OpticalFlowResult {
using Ptr = std::shared_ptr<OpticalFlowResult>;
 //时间戳
int64_t t_ns;
// observations 数据类型是数组，支持单目和双目，存储的元素是map(其中key
    是特诊点的编号， value是观察到特征点的像素位置)
std::vector<Eigen::map<KeypointId, Eigen::AffineCompact2f>>
    observations;
```

```
OpticalFlowInput::Ptr input_images;
};
```

2.3.2 IMU 状态

Visual Inertial Mapping 对优化变量的 IMU 的状态采用了继承的方法构造了关于 IMU 状态变量的多种基础的数据结构，每种数据类型都实现了对状态变量更新的函数 applyInc() 和误差计算。

```
namespace basalt {


constexpr size_t POSE_SIZE = 6;
constexpr size_t POSE_VEL_SIZE = 9;
constexpr size_t POSE_VEL_BIAS_SIZE = 15;
// PoseState 是滑窗中比较老的KeyFrame的数据类型，
    只需要优化pose，无需优化velocity和bias
struct PoseState {
using VecN = Eigen::Matrix<double, POSE_SIZE, 1>;

PoseState() { t_ns = 0; }

PoseState(int64_t t_ns, const Sophus::SE3d& T_w_i)
: t_ns(t_ns), T_w_i(T_w_i) {}

void applyInc(const VecN& inc) { incPose(inc, T_w_i); }

inline static void incPose(const Sophus::Vector6d& inc, Sophus::SE3d&
    T) {
T.translation() += inc.head<3>();
T.so3() = Sophus::SO3d::exp(inc.tail<3>()) * T.so3();
}

EIGEN_MAKE_ALIGNED_OPERATOR_NEW

int64_t t_ns;
Sophus::SE3d T_w_i;
};
```

```
struct PoseVelState : public PoseState {
using VecN = Eigen::Matrix<double, POSE_VEL_SIZE, 1>;

PoseVelState() { vel_w_i.setZero(); };

PoseVelState(int64_t t_ns, const Sophus::SE3d& T_w_i,
const Eigen::Vector3d& vel_w_i)
: PoseState(t_ns, T_w_i), vel_w_i(vel_w_i) {}

void applyInc(const VecN& inc) {
// 调用父类的方法对继承自父类的变量进行更新
PoseState::applyInc(inc.head<6>());
// 实现自己特有变量的更新算法
vel_w_i += inc.tail<3>();
}
// 误差计算方法
VecN diff(const PoseVelState& other) const {
VecN res;
res.segment<3>(0) = other.T_w_i.translation() - T_w_i.translation();
res.segment<3>(3) = (other.T_w_i.so3() * T_w_i.so3().inverse()).log();
res.tail<3>() = other.vel_w_i - vel_w_i;
return res;
}

EIGEN_MAKE_ALIGNED_OPERATOR_NEW

Eigen::Vector3d vel_w_i;
};

struct PoseVelBiasState : public PoseVelState {
using Ptr = std::shared_ptr<PoseVelBiasState>;
using VecN = Eigen::Matrix<double, POSE_VEL_BIAS_SIZE, 1>;

PoseVelBiasState() {
bias_gyro.setZero();
bias_accel.setZero();
};
// PoseState 是滑窗中比较新的Frame的数据类型，需要优化velocity 和 bias
```

```
PoseVelBiasState(int64_t t_ns, const Sophus::SE3d& T_w_i,
const Eigen::Vector3d& vel_w_i,
const Eigen::Vector3d& bias_gyro,
const Eigen::Vector3d& bias_accel)
: PoseVelState(t_ns, T_w_i, vel_w_i),
bias_gyro(bias_gyro),
bias_accel(bias_accel) {}

void applyInc(const VecN& inc) {
PoseVelState::applyInc(inc.head<9>());
bias_gyro += inc.segment<3>(9);
bias_accel += inc.segment<3>(12);
}

VecN diff(const PoseVelBiasState& other) const {
VecN res;
res.segment<9>(0) = PoseVelState::diff(other);
res.segment<3>(9) = other.bias_gyro - bias_gyro;
res.segment<3>(12) = other.bias_accel - bias_accel;
return res;
}

EIGEN_MAKE_ALIGNED_OPERATOR_NEW

Eigen::Vector3d bias_gyro;
Eigen::Vector3d bias_accel;
};
// IMU 原始数据存储的类型
struct ImuData {
using Ptr = std::shared_ptr<ImuData>;

int64_t t_ns;
Eigen::Vector3d accel;
Eigen::Vector3d gyro;

Eigen::Vector3d accel_cov;
Eigen::Vector3d gyro_cov;

ImuData() {
```

```
accel.setZero();
gyro.setZero();

accel_cov.setZero();
gyro_cov.setZero();
}

EIGEN_MAKE_ALIGNED_OPERATOR_NEW
};

} // namespace basalt
\
```

再通过私有继承 PoseVelBiasState 和 PoseState 分别构造了两种数据类型，PoseVelBiasStateWithLin 和 PoseStateWithLin，并对其增加了一些新的属性和方法。具体包含一下几点。

- 分别对父类增加的 linearized 属性 (即变量的线性点是否固定，用于边缘化操作)；
- 分别增加了当前估计的状态变量 state_current 和 T_w_i_current；
- 分别增加了当前估计的状态到线性化点的偏差属性 delta；
- 分别增加了获取变量线性化点的函数 getStateLin；
- 分别增加了获取估计的当前状态的函数 getState；
- 分别增加了获取当前状态偏离线性化点的增量函数 getDelta；
- 分别增加了可用于边缘化和为边缘化变量更行的函数 applyInc (如果变量被边缘化，线性点固定，偏差累加，估计的当前状态为偏差加到线性化点上)，这个操作和 DSO 的操作类似；

2.3.3 后端优化的流程

Algorithm 1 backward optimazation

1. 根据 IMU 的预积分的结果预测 next_frame $[P, R, V]$ 的状态, 并保存视觉的测量 (prev_opt_flow_res : Eigen::map<int64_t, OpticalFlowResult::Ptr>) 和 IMU 的预积分测量 (imu_meas : Eigen::map<int64_t, IntegratedImuMeasurement>)

2. 检索当前帧追踪上的特征点是否是滑动窗口中的已经恢复深度的某个 3D 点的观测值,如果是,那么将对应的 3D 点 (存储在 kpts : Eigen::unordered_map<int, KeypointPosition>) 的主导帧 (host frame) 和当前帧 (target frame) 建立联系 (视觉残差构建需要,对应的联系存储在 obs : Eigen::map<TimeCamId, Eigen::map<TimeCamId, Eigen::vector<KeypointObservation> > >),并通过判断当前 cam0 能观测到 3d 的特征点的数量和未能观测到 3D 的特征点数量的比值来判断是否将当前帧作为关键帧，如果比值小于一定的阈值，那么当前帧需要作为关键帧

3. 如果当前帧是关键帧，那么通过三角化恢复一些 3D 点，并将恢复的 3D 添加到滑窗中的 3d 点集合中 kpts，所有恢复的 3D 点的 host frame 的都是当前帧，其他观测到 3D 点的帧作为 target frame，最后将 host frame 和 target frame 的新的视觉测量联系添加到 obs 中

4. 进行滑窗优化 (KeypointVioEstimator::optimize())

   1 将滑动窗口中的老的关键帧 ($frame\_poses$) 和最近的帧 ($frame\_states$) 添加到优化变量的状态变量的管理器中 $aom : AbsOrderMap$(其数据结构见图 3)。

   2 迭代优化
   在进行迭代优化之前，我们需要证明 3D 点分别在视觉残差的 hessian 矩阵构建完成后和在完整的残差 (视觉残差、IMU 残差和先验残差加起来) 的 hessian 矩阵构建完成后进行边缘化是等价的，证明见图 (4)

      A. 计算视觉残差的边缘化 3d 点的 ${}^{v}H_{pp}^{m}$ 和 ${}^{v}b^{m}$,

      B. 计算 IMU 误差的 ${}^{v}H_{pp}^{m}$ 和 ${}^{v}b^{m}$

5. 边缘化操作当 KFs 的数量大于 max_kf 或者 recent frames 的数量大于 max_states 时需要边缘化

   A. 统计必须边缘化的某些 KFs 和 recent frames，其中 KFs 的边缘化的条件是: 没有标记为关键帧的 recent frames; recent frames: 边缘的顺序是从 F_0 到 F_n，如果被标记为关键帧，则边缘化其速度和偏置，如果没有被标记为关键帧，则边缘化整个状态。
   如果被边缘化的 KFs 和 recent frames 是重投影误差的因子的 target

frame，这个因子将会被扔掉。此外要将边缘化的帧到 $F2$ 的观测去掉，使 F2 位姿态可以重新线性化 (使其能够进行多次迭代优化)

B. 如果 kf_ids 大于 max_kfs，那么只能剔除 KF 中的某些帧，剔除方法采用了循环逐个剔除，直到 kf_ids 等于 max_kfs

   a. 如果一个 KF 的 3D 点在最新帧的被观测到的比例小于 0.05，那么这个关键帧将会被剔除否则跳转到 b

   b. 选择一个离最新关键帧最近的 KF 将其剔除

C. 构建视觉、IMU 和先验的 hessian 和 b 使用 Schur complement 边缘化选出的 KF 和 recent frame 等

D. 更新边缘化变量的参差 $b_\alpha^m = b_\alpha^m + H_{\alpha\alpha}^m \delta_\alpha$

E. 零空间检测，参考 [1] 零空间的求解, 见图 (2)，其表达的意思是对所有的 IMU 状态加施平移或 yaw 角扰动，误差函数的值不变，而施加 roll pitch 扰动，误差将会有很大的变化，由此可以检查系统的一致性 (consistent) 是否被破坏。

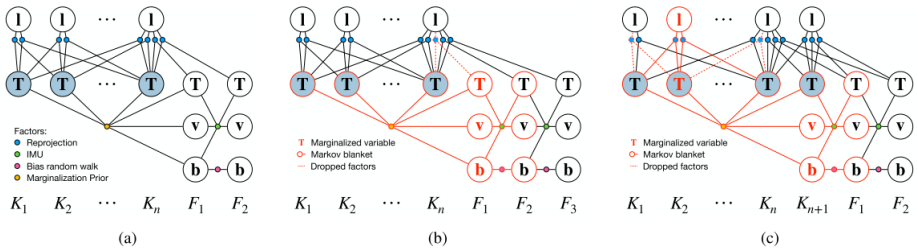

图 1: 边缘化策略

2.4 建图

Visual-Inertial Mapping 环节采用了非线性因子恢复 (NRF) 恢复技术, 保留了边缘 KF 之间的相关性, 但是由于边缘化关键帧状态变量之间的 hessian 矩阵过于稠密, 降低了求解效率, 为此它使用 NRF 提取了其中主要的几个因子 roll-pitch fatcor 和 relative pose factor 因子, 因为 VIO 中绝对

$$\mathcal{N}_{vins} = \begin{bmatrix} {}^{I_0}_{G}\mathbf{R}\,{}^{G}\mathbf{g} & \mathbf{0}_{3\times3} \\ -\lfloor {}^{G}\mathbf{p}_{I_0}\times\rfloor\,{}^{G}\mathbf{g} & \mathbf{I}_{3\times3} \\ -\lfloor {}^{G}\mathbf{v}_{I_0}\times\rfloor\,{}^{G}\mathbf{g} & \mathbf{0}_{3\times3} \\ -\lfloor {}^{G}\mathbf{p}_{f}\times\rfloor\,{}^{G}\mathbf{g} & \mathbf{I}_{3\times3} \end{bmatrix}$$

图 2: VINS 的零空间

	索引: 时间戳(ns)	变量的起始地址	变量的维度
KF_1		0	6
KF_2		6	12
KF_3		12	18
KF_r		6(r-1)	6r
F_1		6r	15
F_2		6r+15	15
F_3		6r+15*2	15
F_4		6r+15*3	15
F_n		6r+15(n-1)	6r+15n

图 3: KeyFrame pose 和 Frame pose 的管理器: AbsOrderMap

位置和 yaw 角不可观, 所以没有恢复 absolute pose factor 和 yaw factor, 然后在所有边缘化的关键帧上提取 orb 特征点和帧间匹配来构造视觉残差因子，最后通过使用非线性最小二乘算法最小化前面提取的视觉残差因子和 NRF 恢复的因子来优化相机的位姿和 3d 的位置, 完成建图。

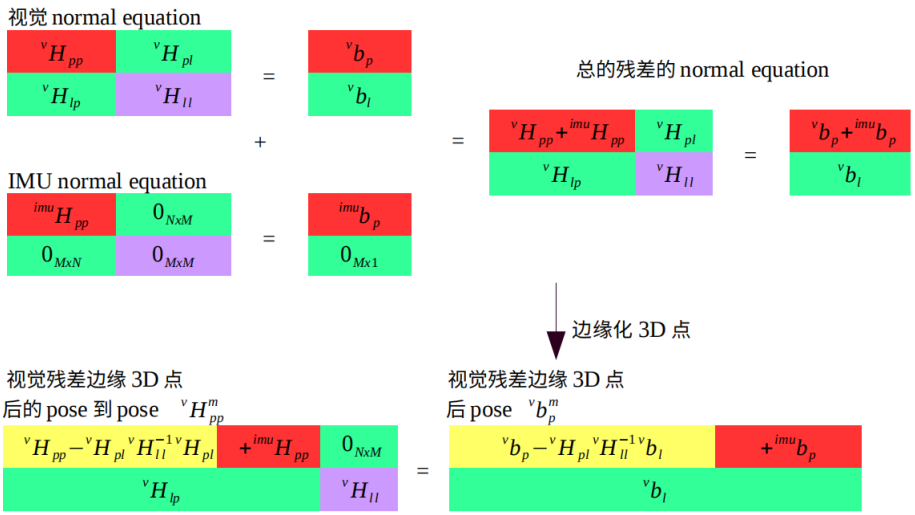

图 4: 3D 点分别在视觉残差的 normal equation 和在总的残差的 normal equation 边缘化是等价的

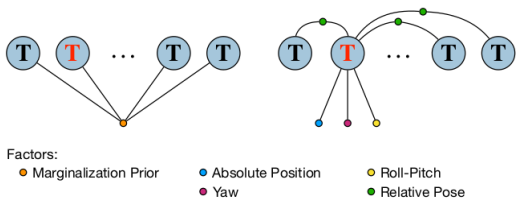

图 5: 非线性因子恢复

参考文献

[1] Stergios I Roumeliotis, Dimitrios G Kottas, Chao Guo, and Joel Hesch. Observability-constrained vision-aided inertial navigation, January 26 2016. US Patent 9,243,916.

[2] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *arXiv preprint arXiv:1904.06504*, 2019.