

原

VINS-Mono 论文公式推导与代码解析

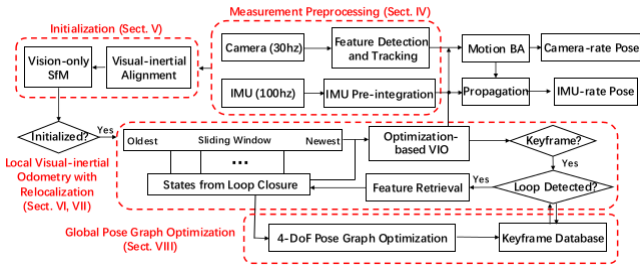
置顶 2019年03月23日 21:59:39 cggos 阅读数: 455 更多

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/u011178262/article/details/88769414>

文章目录

- 概述
1. 测量预处理
- 1.1 前端视觉处理
- 1.2 IMU 预积分
- IMU 测量方程
- 预积分方程
- 误差状态方程
2. 初始化(松耦合)
- 2.1 相机与IMU之间的相对旋转
- 2.2 检测IMU可观性
- 2.3 相机初始化 (Vision-Only SFM)
- 2.4 视觉与IMU对齐
- 陀螺仪Bias标定
- 初始化速度、重力向量和尺度因子
- 优化重力
3. 后端优化(紧耦合)
- 3.1 IMU 测量残差
- 3.2 视觉(td) 测量残差
- 3.3 Temporal Calibration
- Timestamps
- Time Synchronization
- Temporal Calibration
- 3.4 边缘化(Marginalization)
- Marginalization
- Schur Complement
- First Estimate Jacobin
4. 重定位
- 4.1 Loop Detection
- 4.2 Feature Retrieval
- 4.3 Tightly-Coupled Relocalization
5. 全局位姿图优化
6. Remarks on Monocular Visual-Inertial SLAM
- 参考文献

概述



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

- 代码（注释版）：cggos/vins_mono_cg

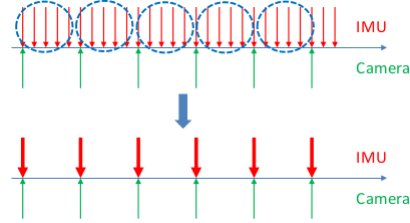
1. 测量预处理

1.1 前端视觉处理

- Simple feature processing pipeline
 - 自适应直方图均衡化（cv::CLAHE）
 - 掩模处理，特征点均匀分布（setMask）
 - 提取图像Harris角点（cv::goodFeaturesToTrack）
 - KLT金字塔光流跟踪（cv::calcOpticalFlowPyrLK）

- 连续帧跟踪
- 本质矩阵(RANSAC)去除外点 (`rejectWithF`)
- 发布`feature_points(id_of_point, un_pts, cur_pts, pts_velocity)`
- Keyframe selection
 - Case 1: Rotation-compensated average feature parallax is larger than a threshold
 - Case 2: Number of tracked features in the current frame is less than a threshold
 - All frames are used for optimization, but non-keyframes are removed first

1.2 IMU 预积分



IMU 测量方程

忽略地球旋转，IMU 测量方程为

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

预积分方程

(1) IMU integration in world frame

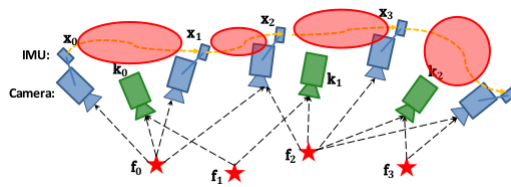
由上面的IMU测量方程积分就可以计算出下一时刻的p、v和q:

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

where

$$\Omega(\omega) = \begin{bmatrix} -[\omega]_{\times} & \omega \\ \omega^T & 0 \end{bmatrix}, [\omega]_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

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



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

$$\begin{aligned}\mathbf{R}_{b_k}^{b_k} \mathbf{p}_{b_{k+1}}^w &= \mathbf{R}_{b_k}^{b_k} (\mathbf{p}_{b_k}^w + \mathbf{v}_{b_k}^w \Delta t_k - \frac{1}{2} \mathbf{g}^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k} \\ \mathbf{R}_{b_k}^{b_k} \mathbf{v}_{b_{k+1}}^w &= \mathbf{R}_{b_k}^{b_k} (\mathbf{v}_{b_k}^w - \mathbf{g}^w \Delta t_k) + \beta_{b_{k+1}}^{b_k} \\ \mathbf{q}_{b_k}^{b_k} \otimes \mathbf{q}_{b_{k+1}}^w &= \gamma_{b_{k+1}}^{b_k},\end{aligned}$$

则 预积分IMU测量模型 (估计值) 为

$$\begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} \\ \hat{\gamma}_{b_{k+1}}^{b_k} \\ \hat{\beta}_{b_{k+1}}^{b_k} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_{b_k}^{b_k} (p_{b_{k+1}}^w - p_{b_k}^w + \frac{1}{2} g^w \Delta t_k^2 - v_{b_k}^w \Delta t_k) \\ q_{b_k}^{b_k} \otimes q_{b_{k+1}}^w \\ R_{b_k}^{b_k} (v_{b_{k+1}}^w + g^w \Delta t_k - v_{b_k}^w) \\ b_{ab_{k+1}} - b_{ab_k} \\ b_{wb_{k+1}} - b_{wb_k} \end{bmatrix}$$

离散状态下采用 **中值法积分** 的预积分方程 (预积分 测量值) 为

$$\begin{aligned}\delta q_{i+1} &= \delta q_i \otimes \begin{bmatrix} 1 \\ \frac{1}{2} w'_i \delta t \end{bmatrix} \\ \delta \alpha_{i+1} &= \delta \alpha_i + \delta \beta_i \delta t + 0.5 a'_i \delta t^2 \\ \delta \beta_{i+1} &= \delta \beta_i + a'_i \delta t \\ b_{a_{i+1}} &= b_{a_i} \\ b_{g_{i+1}} &= b_{g_i}\end{aligned}$$

其中

$$\begin{aligned}w'_i &= \frac{w_{i+1} + w_i}{2} - b_i \\ a'_i &= \frac{\delta q_i(a_i + n_{a0} - b_{a_i}) + \delta q_{i+1}(a_{i+1} + n_{a1} - b_{a_i})}{2}\end{aligned}$$

midPointIntegration 中的相关代码:

```
1 Vector3d un_gyr = 0.5 * (_gyr_0 + _gyr_1) - linearized_bg;
2 result_delta_q = delta_q * Quaterniond(1, un_gyr(0) * _dt / 2, un_gyr(1) * _dt / 2, un_gyr(2) * _dt / 2);
3
4 Vector3d un_acc_0 = delta_q * (_acc_0 - linearized_ba);
5 Vector3d un_acc_1 = result_delta_q * (_acc_1 - linearized_ba);
6 Vector3d un_acc = 0.5 * (un_acc_0 + un_acc_1);
7
8 result_delta_p = delta_p + delta_v * _dt + 0.5 * un_acc * _dt * _dt;
9 result_delta_v = delta_v + un_acc * _dt;
10
11 // 预积分的过程中Bias没有发生改变
12 result_linearized_ba = linearized_ba;
13 result_linearized_bg = linearized_bg;
```

误差状态方程

IMU误差状态向量

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

根据ESKF中 5.3.3 The error-state kinematics 小节公式

$$\begin{aligned}\dot{\delta \mathbf{p}} &= \delta \mathbf{v} & (237a) \\ \dot{\delta \mathbf{v}} &= -\mathbf{R}[\mathbf{a}_m - \mathbf{a}_b]_{\times} \delta \boldsymbol{\theta} - \mathbf{R} \delta \mathbf{a}_b + \delta \mathbf{g} - \mathbf{R} \mathbf{a}_n & (237b) \\ \dot{\delta \boldsymbol{\theta}} &= -[\boldsymbol{\omega}_m - \boldsymbol{\omega}_b]_{\times} \delta \boldsymbol{\theta} - \delta \boldsymbol{\omega}_b - \boldsymbol{\omega}_n & (237c) \\ \delta \dot{\mathbf{a}}_b &= \mathbf{a}_w & (237d) \\ \delta \dot{\boldsymbol{\omega}}_b &= \boldsymbol{\omega}_w & (237e) \\ \delta \dot{\mathbf{g}} &= 0. & (237f)\end{aligned}$$

对于 中值积分 下的 误差状态方程 为

$$\delta \dot{X}_k = \begin{cases} \delta \dot{\theta}_k = -[\frac{w_{k+1} + w_k}{2} - b_{gk}]_{\times} \delta \theta_k - \delta b_{gk} + \frac{n_{a0} + n_{w1}}{2} \\ \delta \dot{\beta}_k = -\frac{1}{2} q_k [a_k - b_{ak}]_{\times} \delta \theta \\ \quad - \frac{1}{2} q_{k+1} [a_{k+1} - b_{ak}]_{\times} ((I - [\frac{w_{k+1} + w_k}{2} - b_{gk}]_{\times} \delta t) \delta \theta_k - \delta b_{gk} \delta t + \frac{n_{a0} + n_{w1}}{2} \delta t) \\ \quad - \frac{1}{2} q_k \delta b_{ak} - \frac{1}{2} q_{k+1} \delta b_{ak} - \frac{1}{2} q_k n_{a0} - \frac{1}{2} q_k n_{a1} \\ \delta \dot{\alpha}_k = -\frac{1}{4} q_k [a_k - b_{ak}]_{\times} \delta \theta \delta t \\ \quad - \frac{1}{4} q_{k+1} [a_{k+1} - b_{ak}]_{\times} ((I - [\frac{w_{k+1} + w_k}{2} - b_{gk}]_{\times} \delta t) \delta \theta_k - \delta b_{gk} \delta t + \frac{n_{a0} + n_{w1}}{2} \delta t) \delta t \\ \quad - \frac{1}{4} q_k \delta b_{ak} \delta t - \frac{1}{4} q_{k+1} \delta b_{ak} \delta t - \frac{1}{4} q_k n_{a0} \delta t - \frac{1}{4} q_k n_{a1} \delta t \\ \delta \dot{b}_{ak} = n_{ba} \\ \delta \dot{b}_{gk} = n_{bg} \end{cases}$$

简写为

$$\delta \dot{X}_k = F \delta X_k + G n$$

所以

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

展开得

$$\begin{aligned} \begin{bmatrix} \delta \alpha_{k+1} \\ \delta \theta_{k+1} \\ \delta \beta_{k+1} \\ \delta b_{ak+1} \\ \delta b_{gk+1} \end{bmatrix} &= \begin{bmatrix} I & f_{01} & \delta t & -\frac{1}{4}(q_k + q_{k+1})\delta t^2 & f_{04} \\ 0 & I - [\frac{w_{k+1} + w_k}{2} - b_{wk}]_{\times} \delta t & 0 & 0 & -\delta t \\ 0 & f_{21} & I & -\frac{1}{2}(q_k + q_{k+1})\delta t & f_{24} \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \delta \alpha_k \\ \delta \theta_k \\ \delta \beta_k \\ \delta b_{ak} \\ \delta b_{gk} \end{bmatrix} \\ &+ \begin{bmatrix} \frac{1}{4} q_k \delta t^2 & v_{01} & \frac{1}{4} q_{k+1} \delta t^2 & v_{03} & 0 & 0 \\ 0 & \frac{1}{2} \delta t & 0 & \frac{1}{2} \delta t & 0 & 0 \\ \frac{1}{2} q_k \delta t & v_{21} & \frac{1}{2} q_{k+1} \delta t & v_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \delta t \end{bmatrix} \begin{bmatrix} n_{a0} \\ n_{w0} \\ n_{a1} \\ n_{w1} \\ n_{ba} \\ n_{bg} \end{bmatrix} \end{aligned}$$

其中



$$\begin{aligned} f_{01} &= -\frac{1}{4}q_k[a_k - b_{ak}] \times \delta t^2 - \frac{1}{4}q_{k+1}[a_{k+1} - b_{ak}] \times (I - [\frac{w_{k+1} + w_k}{2} - b_{gk}] \times \delta t) \delta t^2 \\ f_{21} &= -\frac{1}{2}q_k[a_k - b_{ak}] \times \delta t - \frac{1}{2}q_{k+1}[a_{k+1} - b_{ak}] \times (I - [\frac{w_{k+1} + w_k}{2} - b_{gk}] \times \delta t) \delta t \\ f_{04} &= \frac{1}{4}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t^2)(-\delta t) \\ f_{24} &= \frac{1}{2}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t)(-\delta t) \\ v_{01} &= \frac{1}{4}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t^2) \frac{1}{2} \delta t \\ v_{03} &= \frac{1}{4}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t^2) \frac{1}{2} \delta t \\ v_{21} &= \frac{1}{2}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t^2) \frac{1}{2} \delta t \\ v_{23} &= \frac{1}{2}(-q_{k+1}[a_{k+1} - b_{ak}] \times \delta t^2) \frac{1}{2} \delta t \end{aligned}$$

令

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

则简写为

$$\delta X_{k+1} = F' \delta X_k + V n$$

此处 F' 即代码中 `F`，相关代码见 `midPointIntegration`。

最后得到 IMU 预积分测量关于 IMU Bias 的 雅克比矩阵 J_{k+1} 、IMU 预积分测量的 协方差矩阵 P_{k+1} 和 噪声的 协方差矩阵 Q ，初始状态下的雅克比矩阵和协方差矩阵为 单位阵 和 零矩阵

$$\begin{aligned} J_{b_k} &= I \\ P_{b_k}^{b_k} &= 0 \\ J_{t+\delta t} &= F' J_t = (I + F_t \delta t) J_t, \quad t \in [k, k+1] \\ P_{t+\delta t}^{b_k} &= F' P_t^{b_k} F'^T + V Q V^T \\ &= (I + F_t \delta t) P_t^{b_k} (I + F_t \delta t) + (G_t \delta t) Q (G_t \delta t) \\ Q &= \text{diag}(\sigma_{a_0}^2 \quad \sigma_{\omega_0}^2 \quad \sigma_{a_1}^2 \quad \sigma_{\omega_1}^2 \quad \sigma_{b_a}^2 \quad \sigma_{b_g}^2) \in \mathbb{R}^{18 \times 18} \end{aligned}$$

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

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

此时，可以与 卡尔曼滤波 对比一下：



2. 初始化(松耦合)

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

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

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

2.1 相机与IMU之间的相对旋转

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

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

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

将旋转矩阵写为四元数，则上式可以写为

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

将其写为左乘和右乘的形式

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

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

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

那么对于 n 对测量值，则有

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

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

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

从而权重为

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

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

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

对应代码在 `InitialExRotation::CalibrationExRotation` 中。

```
1 // 相机与IMU之间的相对旋转
2 if(ESTIMATE_EXTRINSIC == 2)
3 {
4     ROS_INFO("calibrating extrinsic param, rotation movement is needed");
5     if (frame_count != 0)
6     {
7         // 选取两帧之间共有的Features
8         vector<pair<Vector3d, Vector3d>> corres = f_manager.getCorresponding(frame_count - 1, frame_count);
9
10        // 校准相机与IMU之间的旋转
11        Matrix3d calib_ric;
12        if (initial_ex_rotation.CalibrationExRotation(corres, pre_integrations[frame_count]->delta_q, calib_ric))
13        {
14            ROS_WARN("initial extrinsic rotation calib success");
15            ROS_WARN_STREAM("initial extrinsic rotation: " << endl << calib_ric);
16            ric[0] = calib_ric;
17            RIC[0] = calib_ric;
18            ESTIMATE_EXTRINSIC = 1;
19        }
20    }
21 }
```

2.2 检测IMU可观性

```
1 // 计算均值
2 map<double, ImageFrame>::iterator frame_it;
3 Vector3d sum_g;
4 for (frame_it = all_image_frame.begin(), frame_it++; frame_it != all_image_frame.end(); frame_it++)
5 {
6     double sum_dt = frame_it->second.pre_integration->sum_dt;
7     Vector3d tmp_g = frame_it->second.pre_integration->delta_v / sum_dt;
8     sum_g += tmp_g;
9 }
10 Vector3d aver_g = sum_g * 1.0 / ((int)all_image_frame.size() - 1);
11
12 // 计算方差
13 double var = 0;
14 for (frame_it = all_image_frame.begin(), frame_it++; frame_it != all_image_frame.end(); frame_it++)
15 {
16     double sum_dt = frame_it->second.pre_integration->sum_dt;
17     Vector3d tmp_g = frame_it->second.pre_integration->delta_v / sum_dt;
18     var += (tmp_g - aver_g).transpose() * (tmp_g - aver_g);
19 }
20
21 // 计算标准差
22 var = sqrt(var / ((int)all_image_frame.size() - 1));
```

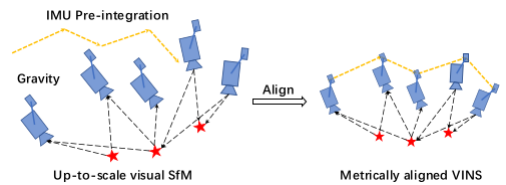
```
23 //ROS_WARN("IMU variation %f!", var);
24 if(var < 0.25) //! 以标准差判断可观性
25 {
26     ROS_INFO("IMU excitation not enough!");
27     //return false;
28 }
```

2.3 相机初始化 (Vision-Only SFM)

- 求取本质矩阵求解位姿 (relativePose)
- 三角化特征点 (sfm.construct)
- PnP求解位姿 (cv::solvePnP)
- 转换到IMU坐标系下
- c0 坐标系作为参考系
- 不断重复直到恢复出滑窗内的Features和相机位姿

2.4 视觉与IMU对齐

- Gyroscope Bias Calibration
- Velocity, Gravity Vector and Metric Scale Initialization
- Gravity Refinement
- Completing Initialization



对应代码: VisualIMUAlignment

陀螺仪Bias标定

标定陀螺仪Bias使用如下代价函数

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

因为四元数最小值为单位四元数 $[1, 0_v]^T$ ，所以令

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

其中

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

所以

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

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

只取上式虚部，再进行最小二乘求解

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

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

对应代码: solveGyroscopeBias

```
1 void solveGyroscopeBias(map<double, ImageFrame> &all_image_frame, Vector3d* Bgs) {
2     Matrix3d A;
3     Vector3d b;
4     A.setZero();
5     b.setZero();
6
7     map<double, ImageFrame>::iterator frame_i;
8     map<double, ImageFrame>::iterator frame_j;
9     for (frame_i = all_image_frame.begin(); next(frame_i) != all_image_frame.end(); frame_i++) {
10         frame_j = next(frame_i);
11
12         MatrixXd tmp_A(3, 3);
13         VectorXd tmp_b(3);
14         tmp_A.setZero();
15         tmp_b.setZero();
16
17         Eigen::Quaterniond q_ij(frame_i->second.R.transpose() * frame_j->second.R);
18         tmp_A = frame_j->second.pre_integration->jacobian.template block<3, 3>(0_R, 0_BG);
19         tmp_b = 2 * (frame_j->second.pre_integration->delta_q.inverse() * q_ij).vec();
20
21         A += tmp_A.transpose() * tmp_A;
```

```

22     b += tmp_A.transpose() * tmp_b;
23 }
24
25 Vector3d delta_bg = A.ldlt().solve(b);
26 ROS_WARN_STREAM("gyroscope bias initial calibration " << delta_bg.transpose());
27
28 // 因为求解出的Bias是变化量，所以要累加
29 for (int i = 0; i <= WINDOW_SIZE; i++)
30     Bgs[i] += delta_bg;
31
32 // 利用新的Bias重新repropagate
33 for (frame_i = all_image_frame.begin(); next(frame_i) != all_image_frame.end(); frame_i++) {
34     frame_j = next(frame_i);
35     frame_j->second.pre_integration->repropagate(Vector3d::Zero(), Bgs[0]);
36 }
37 }

```

初始化速度、重力向量和尺度因子

要估计的状态量

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

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

根据IMU测量模型可知

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

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

$$\begin{aligned}q_{b_k}^{c_0} &= q_{c_k}^{c_0} \otimes (q_b^c)^{-1} \\ s\bar{p}_{b_k}^{c_0} &= s\bar{p}_{c_k}^{c_0} - R_{b_k}^{c_0} p_b^c\end{aligned}$$

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

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

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

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

通过Cholosky分解求解 X_I

$$H^T H X_I = H^T b$$

对应代码: `LinearAlignment`

优化重力

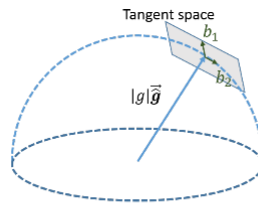


Fig. 5. Illustration of 2 DOF parameterization of gravity. Since the magnitude of gravity is known, \mathbf{g} lies on a sphere with radius $g \approx 9.81m/s^2$. The gravity is parameterized around current estimate as $\mathbf{g} \cdot \hat{\mathbf{g}} + w_1 \mathbf{b}_1 + w_2 \mathbf{b}_2$, where \mathbf{b}_1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

重力矢量的模长固定 (9.8), 其为2个自由度, 在切空间上对其参数化

$$\begin{aligned}\hat{\mathbf{g}} &= \|\mathbf{g}\| \cdot \hat{\mathbf{g}} + w_1 \vec{b}_1 + w_2 \vec{b}_2 \\ &= \|\mathbf{g}\| \cdot \hat{\mathbf{g}} + B\vec{w}, \quad B \in \mathbb{R}^{3 \times 2}, \vec{w} \in \mathbb{R}^{2 \times 1}\end{aligned}$$

令 $\hat{\mathbf{g}} = g^{c_0}$, 将其代入上一小节公式得

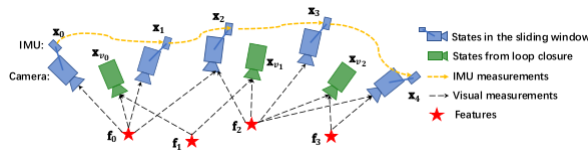
$$\begin{aligned}& \begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 B & R_{c_0}^{b_k} (\bar{p}_{c_{k+1}}^{c_0} - \bar{p}_{c_k}^{c_0}) \\ -I & R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k} \Delta t_k B & 0 \end{bmatrix} \begin{bmatrix} v_{b_{k+1}}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}}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

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

最后，通过将 g^w 旋转至惯性坐标系（世界系）中的 z 轴方向[0,0,1]，可以计算第一帧相机系到惯性系的旋转矩阵 q_{c0}^w ，这样就可以将所有变量调整至惯性世界系（**水平坐标系**，z轴与重力齐）中。

对应代码：RefineGravity

3. 后端优化(紧耦合)



VIO 紧耦合方案的主要思路就是通过将基于视觉构造的残差项和基于IMU构造的残差项放在一起构造一个联合优化的问题，整个优化问题的最优解即可认为是比较准确的状态估计。

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

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

$$x_k = [p_{b_k}^w, v_{b_k}^w, q_{b_k}^w, b_a, b_g], \quad k \in [0, n]$$

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

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

优化过程中的 **误差状态量**

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

$$\delta x_k = [\delta p_{b_k}^w, \delta v_{b_k}^w, \delta \theta_{b_k}^w, \delta b_a, \delta b_g], \quad k \in [0, n]$$

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

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

$$\min_X \left\{ \|r_p - H_p X\|^2 + \sum_{k \in B} \left\| r_B(\hat{z}_{b_{k+1}}^k, X) \right\|_{P_{b_{k+1}}^k}^2 + \sum_{(i,j) \in C} \left\| r_C(\hat{z}_i^{c_j}, X) \right\|_{P_i^{c_j}}^2 \right\}$$

IMU measurement residual Vision measurement residual
Prior from marginalization Covariance from IMU pre-integration Pixel reprojection covariance

其中三个残差项依次是

- 边缘化的先验信息
- IMU测量残差
- 视觉的观测残差

三种残差都是用 **马氏距离**（与量纲无关）来表示的。

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

3.1 IMU 测量残差

(1) IMU 测量残差

上面的IMU预积分（估计值 - 测量值），得到IMU测量残差

$$r_B(\hat{z}_{b_{k+1}}^k, X) = \begin{bmatrix} \delta \alpha_{b_{k+1}}^{b_k} \\ \delta \theta_{b_{k+1}}^{b_k} \\ \delta \beta_{b_{k+1}}^{b_k} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_w^{b_k} (p_{b_{k+1}}^w - p_{b_k}^w + \frac{1}{2} g^w \Delta t^2 - v_{b_k}^w \Delta t) - \hat{\alpha}_{b_{k+1}}^{b_k} \\ 2[q_{b_k}^{w-1} \otimes q_{b_{k+1}}^w \otimes (\hat{\gamma}_{b_{k+1}}^{b_k})^{-1}]_{xyz} \\ R_w^{b_k} (v_{b_{k+1}}^w + g^w \Delta t - v_{b_k}^w) - \hat{\beta}_{b_{k+1}}^{b_k} \\ b_{ab_{k+1}} - b_{ab_k} \\ b_{gb_{k+1}} - b_{gb_k} \end{bmatrix}$$

其中 $[\hat{\alpha}_{b_{k+1}}^{b_k}, \hat{\gamma}_{b_{k+1}}^{b_k}, \hat{\beta}_{b_{k+1}}^{b_k}]$ 为 **IMU预积分Bias修正值**。

```

1 /**
2  * [evaluate 计算IMU测量模型的残差]
3  * @param Pi, Qi, Vi, Bai, Bgi  [前一次预积分结果]
4  * @param Pj, Qj, Vj, Baj, Bgj  [后一次预积分结果]
5  */
6 Eigen::Matrix<double, 15, 1> evaluate(
7     const Eigen::Vector3d &Pi, const Eigen::Quaterniond &Qi, const Eigen::Vector3d &Vi, const Eigen::Vector3d &Bai, const Eigen::Vector3d &Bgi,
8     const Eigen::Vector3d &Pj, const Eigen::Quaterniond &Qj, const Eigen::Vector3d &Vj, const Eigen::Vector3d &Baj, const Eigen::Vector3d &Bgj)
9 {
10     Eigen::Matrix<double, 15, 1> residuals;
11
12     Eigen::Matrix3d dp_dba = jacobian.block<3, 3>(0_P, 0_BA);
13     Eigen::Matrix3d dp_dbg = jacobian.block<3, 3>(0_P, 0_BG);
14
15     Eigen::Matrix3d dq_dbg = jacobian.block<3, 3>(0_R, 0_BG);
16
17     Eigen::Matrix3d dv_dba = jacobian.block<3, 3>(0_V, 0_BA);
18     Eigen::Matrix3d dv_dbg = jacobian.block<3, 3>(0_V, 0_BG);
19

```



```
20 Eigen::Vector3d dba = Bai - linearized_ba;
21 Eigen::Vector3d dbg = Bgi - linearized_bg;
22
23 // IMU预积分的结果,消除掉acc bias和gyro bias的影响, 对应IMU_model中的\hat{\alpha},\hat{\beta},\hat{\gamma}
24 Eigen::Quaterniond corrected_delta_q = delta_q * Utility::deltaQ(dq_dbg * dbg);
25 Eigen::Vector3d corrected_delta_v = delta_v + dv_dba * dba + dv_dbg * dbg;
26 Eigen::Vector3d corrected_delta_p = delta_p + dp_dba * dba + dp_dbg * dbg;
27
28 // IMU项residual计算,输入参数是状态的估计值, 上面correct_delta_*是预积分值, 二者求'diff'得到residual
29 residuals.block<3, 1>(0_P, 0) = Qi.inverse() * (0.5 * G * sum_dt * sum_dt + Pj - Pi - Vi * sum_dt) - corrected_delta_p;
30 residuals.block<3, 1>(0_R, 0) = 2 * (corrected_delta_q.inverse() * (Qi.inverse() * Qj)).vec();
31 residuals.block<3, 1>(0_V, 0) = Qi.inverse() * (G * sum_dt + Vj - Vi) - corrected_delta_v;
32 residuals.block<3, 1>(0_BA, 0) = Baj - Bai;
33 residuals.block<3, 1>(0_BG, 0) = Bgj - Bgi;
34
35 return residuals;
36 }
```

(2) 协方差矩阵

此处用到的协方差矩阵为前面IMU预积分计算出的协方差矩阵。

残差的后处理对应代码：

```
1 // 在优化迭代的过程中, 预积分值是不变的, 输入的状态值会被不断的更新, 然后不断的调用evaluate()计算更新后的IMU残差
2 Eigen::Map<Eigen::Matrix<double, 15, 1>> residual(residuals);
3 residual = pre_integration->evaluate(Pi, Qi, Vi, Bai, Bgi,
4                                   Pj, Qj, Vj, Baj, Bgj);
5
6 Eigen::Matrix<double, 15, 15> sqrt_info =
7     Eigen::LLT<Eigen::Matrix<double, 15, 15>>(pre_integration->covariance.inverse()).matrixL().transpose();
8 //sqrt_info.setIdentity();
9
10 residual = sqrt_info * residual; // 为了保证 IMU 和 视觉参差项在尺度上保持一致, 一般会采用与量纲无关的马氏距离
```

这里残差 residual 乘以 sqrt_info, 这是因为真正的优化项其实是 Mahalanobis 距离: $d = r^T P^{-1} r$, 其中 P 是协方差。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。

而 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 $d = r^T (LL^T) r = (L^T r)^T (L^T r)$, 令 $r' = (L^T r)$, 作为新的优化误差, 所以 sqrt_info 等于 L^T 。

(3) 雅克比矩阵

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

对 $[\delta p_{b_k}^w, \delta \theta_{b_k}^w]$ 求偏导得

$$J[0] = \begin{bmatrix} -q_w^{b_k} & R_w^{b_k} [(p_{b_{k+1}}^w - p_{b_k}^w + \frac{1}{2} g^w \Delta t^2 - v_{b_k}^w \Delta t)]_{\times} \\ 0 & [q_{b_{k+1}}^{w-1} q_{b_k}^w]_L [\hat{\gamma}_{b_{k+1}}^{b_k}]_R J_{b_w}^{\gamma} \\ 0 & R_w^{b_k} [(v_{b_{k+1}}^w + g^w \Delta t - v_{b_k}^w)]_{\times} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{15 \times 7}$$

对 $[\delta v_{b_k}^w, \delta b_{ab_k}, \delta b_{wb_k}]$ 求偏导得

$$J[1] = \begin{bmatrix} -q_w^{b_k} \Delta t & -J_{ba}^{\alpha} & -J_{ba}^{\alpha} \\ 0 & 0 & -[q_{b_{k+1}}^{w-1} \otimes q_{b_k}^w \otimes \hat{\gamma}_{b_{k+1}}^{b_k}]_L J_{b_w}^{\gamma} \\ -q_w^{b_k} & -J_{ba}^{\beta} & -J_{ba}^{\beta} \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix} \in \mathbb{R}^{15 \times 9}$$

对 $[\delta p_{b_{k+1}}^w, \delta \theta_{b_{k+1}}^w]$ 求偏导得

$$J[2] = \begin{bmatrix} -q_w^{b_k} & 0 \\ 0 & [\hat{\gamma}_{b_{k+1}}^{b_k-1} \otimes q_{b_k}^{b_k} \otimes q_{b_{k+1}}^w]_L \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{15 \times 7}$$

对 $[\delta v_{b_k}^w, \delta b_{ab_k}, \delta b_{wb_k}]$ 求偏导得

$$J[3] = \begin{bmatrix} -q_w^{b_k} & 0 & 0 \\ 0 & 0 & 0 \\ q_w^{b_k} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \in \mathbb{R}^{15 \times 9}$$

雅克比矩阵计算的对应代码在 class IMUFactor : public ceres::SizedCostFunction<15, 7, 9, 7, 9> 中的 Evaluate() 函数中。

3.2 视觉(td) 测量残差

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

(1) 切平面重投影误差 (Spherical camera model)

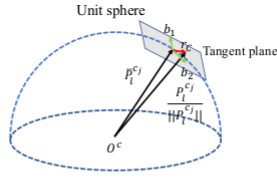


Fig. 6. An illustration of the visual residual on a unit sphere. $\hat{\mathcal{P}}_l^{c_j}$ is the unit vector for the observation of the l^{th} feature in the j^{th} frame. $\mathcal{P}_l^{c_j}$ is predicted feature measurement on the unit sphere by transforming its first observation in the i^{th} frame to the j^{th} frame. The residual is defined on the tangent plane of $\hat{\mathcal{P}}_l^{c_j}$.

$$r_C = (z_l^{c_j}, X) = [b_1, b_2]^T \cdot (\bar{P}_l^{c_j} - \frac{P_l^{c_j}}{\|P_l^{c_j}\|})$$

其中,

$$P_l^{c_j} = q_b^c(q_w^{bj}(q_c^w(q_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b) + p_{b_i}^w - p_{b_j}^w) - p_c^b)$$

```
1 // 将第i frame下的3D点转到第j frame坐标系下
2 Eigen::Vector3d pts_camera_i = pts_i / inv_dep_i; // pt in ith camera frame, 归一化平面
3 Eigen::Vector3d pts_imu_i = qic * pts_camera_i + tic; // pt in ith body frame
4 Eigen::Vector3d pts_w = Qi * pts_imu_i + Pi; // pt in world frame
5 Eigen::Vector3d pts_imu_j = Qj.inverse() * (pts_w - Pj); // pt in jth body frame
6 Eigen::Vector3d pts_camera_j = qic.inverse() * (pts_imu_j - tic); // pt in jth camera frame
```

(2) 像素重投影误差 (Pinhole camera model)

$$r_C = (z_l^{c_j}, X) = (\frac{f}{1.5} \cdot I_{2 \times 2}) \cdot (\frac{\bar{P}_l^{c_j}}{Z} - \frac{P_l^{c_j}}{Z_j})_2$$

```
1 Eigen::Map<Eigen::Vector2d> residual(residuals);
2 #ifdef UNIT_SPHERE_ERROR
3 // 把归一化平面上的重投影误差投影到unit sphere上的好处就是可以支持所有类型的相机 why
4 // 求取切平面上的误差
5 residual = tangent_base * (pts_camera_j.normalized() - pts_j.normalized());
6 #else
7 // 求取归一化平面上的误差
8 double dep_j = pts_camera_j.z();
9 residual = (pts_camera_j / dep_j).head<2>() - pts_j.head<2>();
10 #endif
11 residual = sqrt_info * residual; // 转成 与量纲无关的马氏距离
```

(3) 协方差矩阵

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

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

(4) 雅克比矩阵

下面关于误差状态量对相机测量残差求偏导, 得到高斯迭代优化过程中的雅克比矩阵。

对 $[\delta p_{b_i}^w, \delta \theta_{b_i}^w]$ 求偏导

$$J[0] = \begin{bmatrix} q_b^c q_w^{bj} & -q_b^c q_w^{bj} q_{b_i}^w [q_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b]_{\times} \end{bmatrix} \in \mathbb{R}^{3 \times 6}$$

对 $[\delta p_{b_j}^w, \delta \theta_{b_j}^w]$ 求偏导

$$J[1] = \begin{bmatrix} -q_b^c q_w^{bj} & q_b^c q_w^{bj} [q_{b_i}^w (q_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l} + p_c^b) + p_{b_i}^w - p_{b_j}^w]_{\times} \end{bmatrix} \in \mathbb{R}^{3 \times 6}$$

对 $[\delta p_c^b, \delta \theta_c^b]$ 求偏导

$$J[2] = \begin{bmatrix} q_b^c (q_w^{bj} q_{b_i}^w - I_{3 \times 3}) & -q_b^c q_w^{bj} q_{b_i}^w [q_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l}]_{\times} + [q_b^c (q_w^{bj} (q_{b_i}^w p_c^b + p_{b_i}^w - p_{b_j}^w) - p_c^b)] \end{bmatrix} \in \mathbb{R}^{3 \times 6}$$

对 $\delta \lambda_l$ 求偏导

$$J[3] = -q_b^c q_w^{bj} q_{b_i}^w q_c^b \frac{\bar{P}_l^{c_i}}{\lambda_l^2} \in \mathbb{R}^{3 \times 1}$$

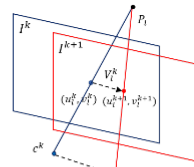
(5) Vision measurement residual for temporal calibration

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

Feature velocity on image plane

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

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



Visual measurement residual with time offset

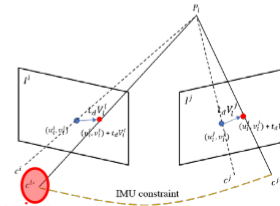
- New state variable t_d , and estimate states $(c^{i'}, c^{j'})$ at time stamping

$$\mathbf{r}_l^{c_j}(\mathcal{X}) = [\mathbf{b}_1 \ \mathbf{b}_2]^T \cdot (\mathcal{P}_l^{c_j} - \frac{\mathcal{P}_l^{c_j}}{\|\mathcal{P}_l^{c_j}\|})$$

$$\bar{\mathcal{P}}_l^{c_j} = \pi_c^{-1}(\begin{bmatrix} u_l^{c_j} \\ v_l^{c_j} \end{bmatrix} + t_d \bar{\mathbf{V}}_l^{c_j})$$

$$\mathcal{P}_l^{c_j} = \mathbf{R}_b^c(\mathbf{R}_{b_i}^b(\mathbf{R}_{b_j}^w(\mathbf{R}_c^b \frac{1}{\lambda_l} \pi_c^{-1}(\begin{bmatrix} u_l^{c_i} \\ v_l^{c_i} \end{bmatrix} + t_d \bar{\mathbf{V}}_l^{c_i}) + \mathbf{p}_c^b) + \mathbf{p}_{b_i}^w - \mathbf{p}_{b_j}^w) - \mathbf{p}_c^b)$$

“Virtual image” at time stamping



```

1 // TR / ROW * row_i 是相机 rolling 到这一行时所用的时间
2 Eigen::Vector3d pts_i_td, pts_j_td;
3 pts_i_td = pts_i - (td - td_i + TR / ROW * row_i) * velocity_i;
4 pts_j_td = pts_j - (td - td_j + TR / ROW * row_j) * velocity_j;
5
6 Eigen::Vector3d pts_camera_i = pts_i_td / inv_dep_i;
7 Eigen::Vector3d pts_imu_i = qic * pts_camera_i + tic;
8 Eigen::Vector3d pts_w = Qi * pts_imu_i + Pi;
9 Eigen::Vector3d pts_imu_j = Qj.inverse() * (pts_w - Pj);
10 Eigen::Vector3d pts_camera_j = qic.inverse() * (pts_imu_j - tic);
11
12 Eigen::Map<Eigen::Vector2d> residual(residuals);
13 #ifdef UNIT_SPHERE_ERROR
14 residual = tangent_base * (pts_camera_j.normalized() - pts_j_td.normalized());
15 #else
16 double dep_j = pts_camera_j.z();
17 residual = (pts_camera_j / dep_j).head<2>() - pts_j_td.head<2>();
18 #endif
19 residual = sqrt_info * residual;

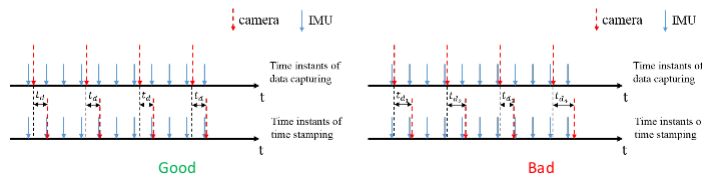
```

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

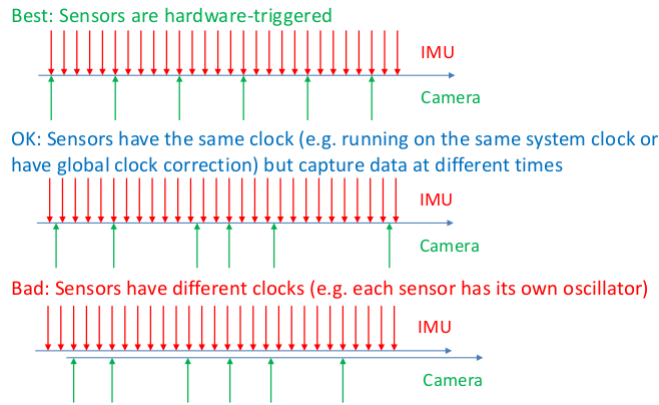
3.3 Temporal Calibration

Timestamps

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



Time Synchronization

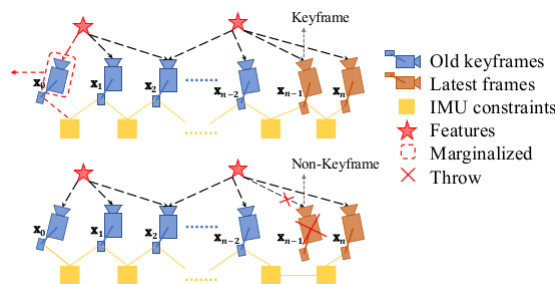


Temporal Calibration

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

3.4 边缘化(Marginalization)

SLAM is tracking a normal distribution through a large state space



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

Marginalization

然而，将pose移出windows时，有些约束会被丢弃掉，这样势必会导致求解的精度下降，而且当MAV进行一些退化运动(如：匀速运动)时，没有历史信息做约束的话是无法求解的。所以，在移出位姿或特征的时候，需要将相关联的约束转变为一个约束项作为prior放到优化问题中，这就是marginalization要做的事情。

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

直接进行边缘化而不加入先验条件的后果：

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

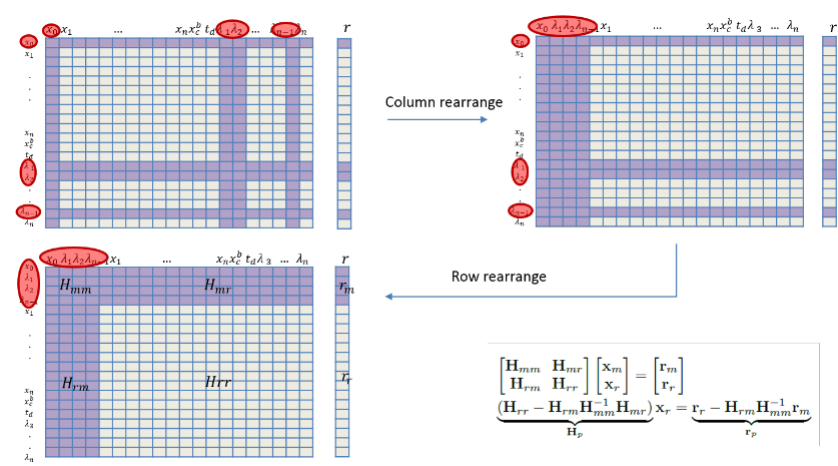
VINS-Mono中为了处理一些悬停的case，引入了一个two-way marginalization：

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

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

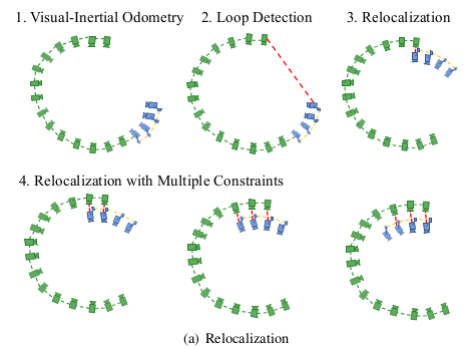
Schur Complement

- Marginalization via Schur complement on information matrix



First Estimate Jacobin

4. 重定位



4.1 Loop Detection

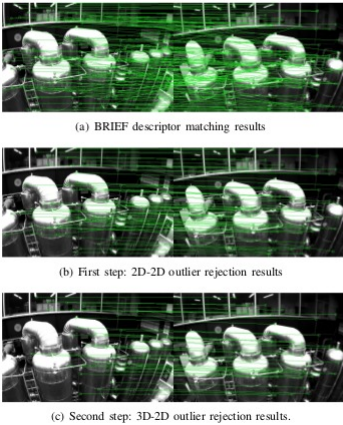
Vins-Mono利用 词袋 DBow2 做Keyframe Database的构建和查询。在建立闭环检测的数据库时，关键帧的Features包括两部分：VIO部分的200个强角点和 500个Fast角点，然后描述子使用 BRIEF (因为旋转可观，匹配过程中对旋转有一定的适应性，所以不用使用ORB)。

- Describe features by BRIEF
 - Features that we use in the VIO (200, not enough for loop detection)
 - Extract new FAST features (500, only use for loop detection)
- Query Bag-of-Word (DBow2)
 - Return loop candidates

4.2 Feature Retrieval

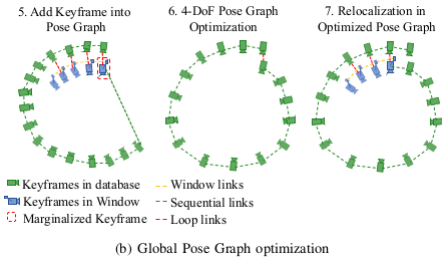
在闭环检测成功之后，会得到回环候选帧，所以要在已知位姿的回环候选帧和滑窗内的匹配帧通过 BRIEF描述子匹配，然后把回环帧加入到滑窗的优化当中，这时整个滑窗的状态量的维度是不发生变化的，因为回环帧的位姿是固定的。

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



4.3 Tightly-Coupled Relocalization

5. 全局位姿图优化



因为之前做的非线性优化本质只是在一个滑窗之内求解出了相机的位姿，而且在回环检测部分，利用固定位姿的回环帧只是纠正了滑窗内的相机位姿，并没有修正其他位姿(或者说没有将回环发现的误差分配到整个相机的轨迹上)，缺少全局的一致性，所以要进行一次全局的Pose Graph。全局的Pose Graph较之滑窗有一定的迟滞性，只有相机的Pose滑出滑窗的时候，Pose才会被加到全局的Pose Graph当中。

(1) Adding Keyframes into the Pose Graph

- Sequential edges from VIO
 - Connected with 4 previous keyframes
- Loop closure edges
 - Only added when a keyframe is marginalized out from the sliding window VIO
 - Multi-constraint relocalization helps eliminating false loop closures
 - Huber norm for rejection of wrong loops

(2) 4-DOF Pose Graph Optimization

- Roll and pitch are observable from VIO

(3) Pose Graph Management

(4) Map Reuse

- Save map at any time
- Load map and re-localize with respect to it
- Pose graph merging

6. Remarks on Monocular Visual-Inertial SLAM

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

参考文献

- [1] VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator
- [2] Shaojie Shen, Monocular Visual-Inertial SLAM slides, 2018
- [3] Quaternion kinematics for the error-state Kalman filter
- [4] Xiaobuyi, [VINS-Mono代码分析总结](#)

👍
2

💬

📖

🔖

📄

<

惊艳!!! 这家的空气电加湿器 质量保证 错过今天再等一年

鸿鑫机械 · 顶新



想对作者说点什么

VINS-Mono算法梳理与代码介绍

07-20

对于VINS-Mono算法进行了梳理，结合算法对代码进行了简单介绍

下载

港科VINS-Mono系统学习

阅读数 1190

视频地址: <https://www.youtube.com/watch?v=l4txdvGhT6I>github: <https://github.com/HKUST-Aerial-Robo...> 博文 来自: [snnily的博客](#)

VINS-Mono 代码解读

阅读数 1万+

VINS-Mono代码解读标签 (空格分隔) : SLAMVIO系统启动命令\$roslaunchvins_estimatoruroc.launch\$roslau... 博文 来自: [u012871872的博客](#)



知网论文查重入口

VINS-Mono详解 (一)

阅读数 521

文章目录VINS-Mono前端概述入口函数main()回调函数img_callback()发布频率控制特征点提取与光流跟踪单目处...

博文 来自: [伟钢的碎碎念](#)

VINS-Mono源码解析 (二) 前端: 特征跟踪

阅读数 4306

VINS-Mono源码解析 (二) 前端: 特征跟踪 VINS-Mono中的前端处理在ROS节点feature_tracker_node中, ... 博文 来自: [roboto](#)

VINS理论与代码详解4——初始化

阅读数 5450

VINS理论与代码详解4——初始化1. 基于滑动窗口的纯视觉单目初始化 在介绍纯视觉初始化前我们首先讲一讲...

博文 来自: [wangshuailpp的博...](#)

Forster论文预积分难理解部分推导 (不定期更新)

阅读数 444

预积分: 横线上面是今早泡泡机器人推送的推导过程, 完全摘抄了下来, 但这个式4-1到下一步却很晦涩, 下面是详...

博文 来自: [l741299292的博客](#)

VINS技术路线与代码详解

阅读数 2万+

VINS技术路线 写在前面: 本文整和自己的思路, 希望对学习VINS或者VIO的同学有所帮助, 如果你觉得文章写的对...

博文 来自: [wangshuailpp的博...](#)

股市彻底变天了, 不看你就亏大了!

指南针 · 顶新

VINS-Mono 代码解读 - u012871872的博客 - CSDN博客

11-2

Loading [MathJax]/jax/output/HTML-CSS/jax.js...

VINS-Mono源码解析(一)系统框架 - roboto - CSDN博客

3-15

<https://blog.csdn.net/q597967420/article/details...HKUST-Aerial-Robotics/VINS-Mono>, 论文里面也有...博文 来自: wiked的专栏 VINS 代码学习(...

VINS-详细解读 (论文+代码)

阅读数 1712

VINS-初始化BVINS-非线性优化VINS-非线性优化-先验残差

博文 来自: [JH_2333的博客](#)

VINS-Mono源码解析 (一) 系统框架

阅读数 1万+

VINS-Mono源码解析 (一) 系统框架1.VINS-Mono简介VINS-Mono是HKUST的ShenShaojie团队开源的一套Visu... 博文 来自: [roboto](#)

VINS-Mono论文要点总结(四)——重定位 - blowballs的专..._CSDN博客

2-9

<https://blog.csdn.net/u014527548/article/details...>博文 来自: ZCC的专栏 VINS-Mono 论文要点总结(一...VINS-Mono论文学习与代码解读——目录与...

VINS-Mono源码解析(四)后端: Initialization - roboto - CSDN博客

11-4

个人分类: VINS-Mono源码解析 版权声明:本文为博主...https://blog.csdn.net/q597967420/article/details...变化” 斜率” 就是对的Jacobian, 对应论文...



HFUT_qiangyang
225篇文章
排名:3000+

关注




snnily
21篇文章
排名:千里之外

关注



Rain-XIA
3篇文章
排名:千里之外

关注



谁特么是人造革
2篇文章
排名:千里之外

关注

VINS-Mono源码解析 (四) 后端: Initialization

阅读数 3925

VINS-Mono源码解析 (四) 后端:Initialization整个Initialization部分对应Estimator::initialStructure()函数,包括Visi... 博文 来自: [roboto](#)

VINS-mono详细解读与实现 - Darlingqiang的博客 - CSDN博客

Vins-mono是香港科技大学开源的一个VIO算法,https://github.com/HKUST-Aerial-Robotics/VINS-Mono,是用紧耦...

VINS-Mono源码解析(二)前端:特征跟踪 - roboto - CSDN博客

<https://blog.csdn.net/q597967420/article/details...>- 论文中的说法是: State Estimation其实是想估计...VINS(-)简介与代码结构 VINS-Mono和VINS-