VIO from scratch 3-3

Sunday, March 22, 2020 5:21 PM

VIO sliding window:

$$\min_{\chi} \rho\left(\|r_p - J_p\chi\|^2_{\Sigma_p}\right) + \sum_{i\in B} \rho\left(\|r_b(z_{b_ib_{i+1}},\chi)\|^2_{\Sigma_{b_ib_{i+1}}}\right) + \sum_{(i,j)\in F} \rho\left(\|r_f(z^{c_i}_{f_j},\chi)\|^2_{\Sigma^{c_i}_{f_j}}\right)$$

prior IMU error

image error

State in BA at time i: inverse depth.

$X = [X_n, X_{n+1} \cdots X_{n+N}, \lambda_m, \lambda_{m+1}, \cdots, \lambda_{m+M}]$

$X_i = [P_{wb_i}, q_{wb_i}, V^w_i, b^{b_i}_a, b^{b_i}_g]^T, \ i\in[n, n+N]$,

N: Key frame number,

M: features number,

reprojection error:

$$r_c = \begin{bmatrix} \frac{x}{z} - u \\ \frac{y}{z} - v \end{bmatrix}, \quad (u,v)^T \text{ is normalized coordinate.}$$

inverse depth:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{\lambda}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \lambda = \frac{1}{z} \text{ is inverse depth.}$$

$$\begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \\ 1 \end{bmatrix} = \underset{\text{to } c_j}{T^{-1}_{bc}} \ \underset{\text{to } b_j}{T^{-1}_{wb_j}} \ \underbrace{T_{wb_i} T_{bc} \begin{bmatrix} \frac{1}{\lambda}u_{c_i} \\ \frac{1}{\lambda}v_{c_i} \\ \frac{1}{\lambda} \\ 1 \end{bmatrix}}_{\text{body frame}}$$

bias white noise

$\tilde{\omega}^b = \omega^b + b^g + n^g$

$\tilde{a}^b = q_{bw}(a^w + g^w) + b^a + n^a$.

PVQ derivative:

$\dot{P}_{wb_t} = V^w_t$

$\dot{V}^w_t = a^w_t$

$\dot{q}_{wb_t} = q_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix}$

from time i, we can integrate to get PVQ in time j.

$P_{wb_j} = P_{wb_i} + V^w_i \Delta t + \iint_{t\in[i,j]} (q_{wb_t}a^{b_t} - g^w)\delta t^2$

$V^w_j = V^w_i + \int_{t\in[i,j]} (q_{wb_t}a^{b_t} - g^w)\delta t$

$q_{wb_j} = \int_{t\in[i,j]} q_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix} \delta t$

Every time q_{wb_i} changes, we need to re-integrate.

pre-integration:

from global frame to local frame

$q_{wb_t} = q_{wb_i} \otimes q_{b_ib_t}$

$P_{wb_j} = P_{wb_i} + V^w_i \Delta t - \frac{1}{2}g^w\Delta t^2 + q_{wb_i}\underline{\iint_{t\in[i,j]} (q_{b_ib_t}a^{b_t})\delta t^2}$

$V^w_j = V^w_i - g^w\Delta t + q_{wb_i}\underline{\int_{t\in[i,j]} (q_{b_ib_t}a^{b_t})\delta t}$

$q_{wb_j} = q_{wb_i}\underline{\int_{t\in[i,j]} q_{b_ib_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix} \delta t}$

these terms are independent of global terms, and are the pre-integration terms:

$\alpha_{b_ib_j} = \iint_{t\in[i,j]} (q_{b_ib_t}a^{b_t})\delta t^2$ relative position of i,j

$\beta_{b_ib_j} = \int_{t\in[i,j]} (q_{b_ib_t}a^{b_t})\delta t$ relative velocity of i,j

$q_{b_ib_j} = \int_{t\in[i,j]} q_{b_ib_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix} \delta t$ relative rotation of i,j

Hence PVQ integration becomes:

$$\begin{bmatrix} P_{wb_j} \\ V^w_j \\ q_{wb_j} \\ b^a_j \\ b^g_j \end{bmatrix} = \begin{bmatrix} P_{wb_i} + V^w_i\Delta t - \frac{1}{2}g^w\Delta t^2 + q_{wb_i}\alpha_{b_ib_j} \\ V^w_i - g^w\Delta t + q_{wb_i}\beta_{b_ib_j} \\ q_{wb_i}q_{b_ib_j} \\ b^a_i \\ b^g_i \end{bmatrix}$$

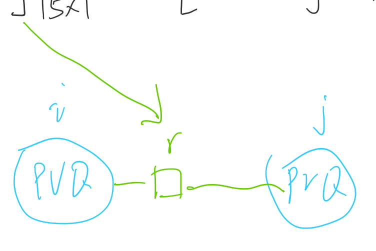
We can constrain the poses as

$$\begin{bmatrix} r_p \\ r_q \\ r_v \\ r_{ba} \\ r_{bg} \end{bmatrix}_{15\times1} = \begin{bmatrix} q_{b_iw}(P_{wb_j} - P_{wb_i} - V^w_i\delta t + \frac{1}{2}g^w\Delta t^2) - \alpha_{b_ib_j} \\ 2[q_{b_jb_i} \otimes (q_{b_iw} \otimes q_{wb_j})]_{xyz} \\ q_{b_iw}(V^w_j - V^w_i + g^w\Delta t) - \beta_{b_ib_j} \\ b^a_j - b^a_i \\ b^g_j - b^g_i \end{bmatrix}$$

we only take the imaginary part.

blue terms are pre-integration terms

we assume the biases are constant between i,j.



We use mid-point method to compute the discrete form:

$\omega = \frac{1}{2}((\omega^{b_k} - b^g_k) + (\omega^{b_{k+1}} - b^g_k))$

$q_{b_ib_{k+1}} = q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}$

$a = \frac{1}{2}(q_{b_ib_k}(a^{b_k} - b^a_k) + q_{b_ib_{k+1}}(a^{b_{k+1}} - b^a_k))$

$\alpha_{b_ib_{k+1}} = \alpha_{b_ib_k} + \beta_{b_ib_k}\delta t + \frac{1}{2}a\delta t^2$

$\beta_{b_ib_{k+1}} = \beta_{b_ib_k} + a\delta t$

$b^a_{k+1} = b^a_k + n_{b^a_k}\delta t$

$b^g_{k+1} = b^g_k + n_{b^g_k}\delta t$

Covariance propagation:

$y = Ax, \ X \in N(0, \Sigma_x), \ \Sigma_y = A\Sigma_xA^T$.

$\Sigma_y = E((Ax)(Ax)^T)$

$= E(Axx^TA^T)$

$= A\Sigma_xA^T$

For the pre-integration terms:

$\eta_{ik} = F_{k-1}\eta_{ik-1} + G_{k-1}n_{k-1}$

$\eta_{ik} = [\delta\theta_{ik}, \delta v_{ik}, \delta p_{ik}]. \ n_k = [n^g_k, n^a_k]$

$\therefore \Sigma_{ik} = F_{k-1}\Sigma_{ik-1}F^T_{k-1} + G_{k-1}\Sigma_nG^T_{k-1}$

Linerization: $X_k = f(X_{k-1}, u_{k-1})$,

$\begin{cases} \text{taylor series} & ① \\ \text{error change wrt time}, \ \dot{X}_k = A. \ X_k = X_{k-1} + \dot{X}\Delta t. & ② \end{cases}$

①: $X = \hat{X} + \delta X$, u noise is n. X is estimate, $\hat{X}$ is true value, δx is error

$X_k = f(X_{k-1}, u_{k-1}), \ \delta X_k = F\delta X_{k-1} + Gn_{k-1}$.

Proof: $X_k = f(X_{k-1}, u_{k-1})$ taylor expansion.

$\hat{X}_k + \delta X_k = f(\hat{X}_{k-1} + \delta X_{k-1}, \hat{u}_{k-1} + n_{k-1})$

$\hat{X}_k + \delta X_k = f(\hat{X}_{k-1}, \hat{u}_{k-1}) + F\delta X_{k-1} + Gn_{k-1}$

these terms are equal since they are the true values.

②. suppose $\delta\dot{x} = A\delta x + Bn$

$\delta x_k = \delta x_{k-1} + \delta\dot{x}_{k-1}\Delta t \rightarrow \delta x_k = (I + A\Delta t)\delta x_{k-1} + B\Delta t n_{k-1}$

①, ② $\Rightarrow F = I + A\Delta t, \ G = B\Delta t$.

papers usually use ②, since $\dot{v} = Ra^b + g$ is known.

$\dot{v} + \delta\dot{v} = R(I + [\delta\theta]_\times)(a^b + \delta a^b) + g + \delta g$ these are error terms

$\delta\dot{v} = R\delta a^b + R[\delta\theta]_\times(a^b + \delta a^b) + \delta g$

$\delta\dot{v} = R\delta a^b - R[a^b]_\times\delta\theta + \delta g$

We can write out in $\delta\dot{x} = A\delta x + Bn$ form

but ② is not general, sometimes we may not know this form.

If we use Taylor expansion, we want to get this form:

$$\begin{bmatrix} \delta\alpha_{b_{k+1}b'_{k+1}} \\ \delta\theta_{b_{k+1}b'_{k+1}} \\ \delta\beta_{b_{k+1}b'_{k+1}} \\ \delta b^a_{k+1} \\ \delta b^g_{k+1} \end{bmatrix} = F\begin{bmatrix} \delta\alpha_{b_kb'_k} \\ \delta\theta_{b_kb'_k} \\ \delta\beta_{b_kb'_k} \\ \delta b^a_k \\ \delta b^g_k \end{bmatrix} + G\begin{bmatrix} n^a_k \\ n^g_k \\ n^a_{k+1} \\ n^g_{k+1} \\ n_{b^a_k} \\ n_{b^g_k} \end{bmatrix}$$

To derive some terms in F, G as examples:

for simplicity: $\frac{\partial x_a}{\partial\delta\theta} = \lim_{\delta\theta\to0}\frac{R_{ab}\exp([\delta\theta]_\times)x_b - R_{ab}x_b}{\delta\theta}$

$\Rightarrow \frac{\partial x_a}{\partial\delta\theta} = \frac{\partial R_{ab}\exp([\delta\theta]_\times)x_b}{\partial\delta\theta}$

3rd row of F:

$\beta_{b_i,b_{k+1}} = \beta_{b_i,b_k} + a\delta t$

$= \beta_{b_ib_k} + \frac{1}{2}(q_{b_ib_k}(a^{b_k} - b^a_k) + q_{b_ib_{k+1}}(a^{b_{k+1}} - b^a_k))\delta\theta$

$f_{33} = \frac{\partial\beta_{b_ib_{k+1}}}{\partial\delta\beta_{b_kb'_k}} = I_{3\times3}$

for $\frac{\partial\beta_{b_ib_{k+1}}}{\partial\theta_{b_{k+1}b'_{k+1}}}$, first rewrite $\beta_{b_ib_{k+1}}$ as

$\beta_{b_ib_{k+1}} = \beta_{b_ib_k} + \frac{1}{2}(q_{b_ib_k}(a^{b_k} - b^a_k) + q_{b_ib_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b^a_k))\delta t$

$\therefore \frac{\partial\beta_{b_ib_{k+1}}}{\partial\delta\theta_{b_kb'_k}} = \frac{\partial a\delta t}{\partial\delta\theta_{b_kb'_k}}$ add perturbation

where $a\delta t = \frac{1}{2}q_{b_ib_k} \otimes \boxed{\begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{b_kb'_k} \end{bmatrix}}(a^{b_k} - b^a_k)\delta t + \frac{1}{2}q_{b_ib_k} \otimes \boxed{\begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{b_kb'_k} \end{bmatrix}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}(a^{b_{k+1}} - b^a_k)\delta\theta$

We can re-write in SO3.

$= \frac{1}{2}R_{b_ib_k}\exp([\delta\theta_{b_kb'_k}]_\times)(a^{b_k} - b^a_k)\delta t + \frac{1}{2}R_{b_ib_k}\exp([\delta\theta_{b_kb'_k}]_\times)\exp([\omega\delta t]_\times)(a^{b_{k+1}} - b^a_k)\delta t$

where $\frac{\partial R_{b_ib_k}\exp([\delta\theta_{b_kb'_k}]_\times)(a^{b_k} - b^a_k)\delta t}{\partial\delta\theta_{b_kb'_k}} = \frac{\partial R_{b_ib_k}(I + [\delta\theta_{b_kb'_k}]_\times)(a^{b_k} - b^a_k)\delta t}{\partial\delta\theta_{b_kb'_k}}$

move $\delta\theta$ out of skew

$= \frac{\partial - R_{b_ib_k}[(a^{b_k} - b^a_k)\delta t]_\times\delta\theta_{b_kb'_k}}{\partial\delta\theta_{b_kb'_k}}$

$= -R_{b_ib_k}[(a^{b_k} - b^a_k)\delta t]_\times$

similarly we can get

$\frac{\partial R_{b_ib_k}\exp([\delta\theta_{b_kb'_k}]_\times)\exp([\omega\delta\theta]_\times)(a^{b_{k+1}} - b^a_k)\delta t}{\partial\delta\theta_{b_kb'_k}} \approx -R_{b_ib_{k+1}}[(a^{b_k} - b^a_k)\delta t]_\times(I - [\omega\delta t]_\times)$

$\therefore f_{32} = \frac{\partial\beta_{b_ib_{k+1}}}{\partial\delta\theta_{b_kb'_k}} = -\frac{1}{2}(R_{b_ib_k}[a^{b_k} - b^a_k]_\times\delta t + R_{b_ib_{k+1}}[(a^{b_k} - b^a_k)]_\times(I - [\omega]_\times\delta t)\delta t)$

f_{32} means how the error in $\theta_{b_kb'_k}$ will affect the error in $\beta_{b_ib_{k+1}}$

Jacobian of reprojection:

$$r_c = \begin{bmatrix} \frac{x_{c_j}}{z_{c_j}} - u_{c_j} \\ \frac{y_{c_j}}{z_{c_j}} - v_{c_j} \end{bmatrix}$$

for features in frame i, we can project it to frame j

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

use R, P instead of T

$$f_{c_j} = \begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \end{bmatrix} = R^T_{bc}R^T_{wb_j}R_{wb_i}R_{bc}\frac{1}{\lambda}\begin{bmatrix} u_{c_i} \\ v_{c_i} \\ 1 \end{bmatrix} + R^T_{bc}(R^T_{wb_j}((R_{wb_i}P_{bc} + P_{wb_i}) - P_{wb_j}) - P_{bc})$$

define $f_{b_i} = R_{bc}f_{c_i} + P_{bc}$

$f_w = R_{wb_i}f_{b_i} + P_{wb_i}$

$f_{b_j} = R^T_{wb_j}(f_{c_i} - P_{bc})$

$$J = \begin{bmatrix} \frac{\partial r_c}{\partial\begin{bmatrix} \delta P_{b_ib'_i} \\ \delta\theta_{b_ib'_i} \end{bmatrix}} & \frac{\partial r_c}{\partial\begin{bmatrix} \delta P_{b_jb'_j} \\ \delta\theta_{b_jb'_j} \end{bmatrix}} & \frac{\partial r_c}{\partial\begin{bmatrix} \delta P_{cc'} \\ \delta\theta_{cc'} \end{bmatrix}} & \frac{\partial r_c}{\partial\delta\lambda} \end{bmatrix}$$

Step 1:

$$\frac{\partial r_c}{\partial f_{c_j}} = \begin{bmatrix} \frac{1}{z_{c_j}} & 0 & -\frac{x_{c_j}}{z^2_{c_j}} \\ 0 & \frac{1}{z_{c_j}} & -\frac{y_{c_j}}{z^2_{c_j}} \end{bmatrix}$$

Step 2:

$$\frac{\partial f_{c_j}}{\partial\delta P_{b_ib'_i}} = R^T_{bc}R^T_{wb_j}$$

Jacobian of IMU residuals:

$$e_B(X_i, X_j) = \begin{bmatrix} r_p \\ r_q \\ r_v \\ r_{ba} \\ r_{bg} \end{bmatrix} = \begin{bmatrix} q_{b_iw}(P_{wb_j} - P_{wb_i} - V^w_i\Delta t + \frac{1}{2}g^w\Delta t^2) - \alpha_{b_ib_j} \\ 2[q_{b_jb_i} \otimes (q_{b_iw} \otimes q_{wb_j})]_{xyz} \\ q_{b_iw}(V^w_j - V^w_i + g^w\Delta t) - \beta_{b_ib_j} \\ b^a_j - b^a_i \\ b^g_j - b^g_i \end{bmatrix}_{15\times1}$$

Homework: LM for curve fitting

Construct LS problem:

```
Problem problem(Problem::ProblemType::Generic_Problem);
shared_ptr<CurveFittingVertex> vertex(new CurveFittingVertex());
vertex->setParameters(Eigen::Vector3d(0,0,0));
problem.AddVertex(vertex);
```

at this step, we know we want to estimate X, but we don't know how, and we need to construct the residuals

These residuals are the edges.