深蓝学院
shenlanxueyuan.com

# 基于图优化的建图方法

主讲人 游如光

# 纲要

- 第一部分：雅可比推导
- 第二部分：代码实现及效果分析
- 第三部分：编码器融合

# 雅可比推导



- 残差
- 状态量
  - 对位姿、速度、 bias 的求导以扰动的方式进行，所以此处状态量实际是扰动

$$
\begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_\theta \\ \mathbf{r}_v \\ \mathbf{r}_{b^a} \\ \mathbf{r}_{b^g} \end{bmatrix}
=
\begin{bmatrix}
\mathbf{q}_{wb_i}^{*}\left(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2 - \mathbf{q}_{wb_i}\alpha_{b_i b_j}\right) \\
2\left[\mathbf{q}_{b_i b_j}^{*} \otimes \left(\mathbf{q}_{wb_i}^{*} \otimes \mathbf{q}_{wb_j}\right)\right]_{xyz} \\
\mathbf{q}_{wb_i}^{*}\left(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t - \mathbf{q}_{wb_i}\beta_{b_i b_j}\right) \\
\mathbf{b}_j^a - \mathbf{b}_i^a \\
\mathbf{b}_j^g - \mathbf{b}_i^g
\end{bmatrix}
$$

$$
=
\begin{bmatrix}
\mathbf{q}_{wb_i}^{*}\left(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2\right) - \alpha_{b_i b_j} \\
2\left[\mathbf{q}_{b_i b_j}^{*} \otimes \left(\mathbf{q}_{wb_i}^{*} \otimes \mathbf{q}_{wb_j}\right)\right]_{xyz} \\
\mathbf{q}_{wb_i}^{*}\left(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t\right) - \beta_{b_i b_j} \\
\mathbf{b}_j^a - \mathbf{b}_i^a \\
\mathbf{b}_j^g - \mathbf{b}_i^g
\end{bmatrix}
$$

$$
\begin{bmatrix} \delta\mathbf{p}_{wb_i} & \delta\theta_{wb_i} & \delta\mathbf{v}_i^w & \delta\mathbf{b}_i^a & \delta\mathbf{b}_i^g & \delta\mathbf{p}_{wb_j} & \delta\theta_{wb_j} & \delta\mathbf{v}_j^w & \delta\mathbf{b}_j^a & \delta\mathbf{b}_j^g \end{bmatrix}
$$

# 雅可比推导

深蓝学院
shenlanxueyuan.com

- 对 p 残差求导

$$
\begin{aligned}
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{p}_{wb_i}} &= \frac{\partial -\mathbf{q}_{wb_i}^*(\mathbf{p}_{wb_i}+\delta \mathbf{p}_{wb_i})}{\partial \delta \mathbf{p}_{wb_i}} \\
&= -\mathbf{R}_{b_i w} \\
\frac{\partial \mathbf{r}_p}{\partial \delta \theta_{wb_i}} &= \frac{\partial (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta \theta_{wb_i} \end{bmatrix})^*(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2)}{\partial \delta \theta_{wb_i}} \\
&= \frac{\partial (\mathbf{R}_{wb_i} \exp(\delta \theta_{wb_i}^\wedge))^{-1}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2)}{\partial \delta \theta_{wb_i}} \\
&= \frac{\partial \exp((-\delta \theta_{wb_i})^\wedge)\mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2)}{\partial \delta \theta_{wb_i}} \\
&\approx \frac{\partial (\mathbf{I} - \delta \theta_{wb_i}^\wedge)\mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2)}{\partial \delta \theta_{wb_i}} \\
&= (\mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2))^\wedge \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{v}_i^w} &= -\mathbf{R}_{b_i w} \Delta t \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{b}_i^a} &= \frac{\partial -(\bar{\alpha}_{b_i b_j} + \mathbf{J}_{b_i^a}^\alpha \delta \mathbf{b}_i^a + \mathbf{J}_{b_i^g}^\alpha \delta \mathbf{b}_i^g)}{\partial \delta \mathbf{b}_i^a} \\
&= -\mathbf{J}_{b_i^a}^\alpha \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{b}_i^g} &= -\mathbf{J}_{b_i^g}^\alpha
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{b}_i^g} &= -\mathbf{J}_{b_i^g}^\alpha \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{p}_{wb_j}} &= \mathbf{R}_{b_i w} \\
\frac{\partial \mathbf{r}_p}{\partial \delta \theta_{wb_j}} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{v}_j^w} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{b}_j^a} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_p}{\partial \delta \mathbf{b}_j^g} &= \mathbf{0}
\end{aligned}
$$

# 雅可比推导

- 对 theta 残差求导

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{p}_{wb_i}} = \mathbf{0}$$

$$\begin{aligned}
\frac{\partial \mathbf{r}_\theta}{\partial \delta \theta_{wb_i}} &= \frac{\partial 2[\mathbf{q}_{b_ib_j}^* \otimes (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_i} \end{bmatrix})^* \otimes \mathbf{q}_{wb_j}]_{xyz}}{\partial \delta \theta_{wb_i}} \\
&= \frac{\partial - 2[(\mathbf{q}_{b_ib_j}^* \otimes (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_i} \end{bmatrix})^* \otimes \mathbf{q}_{wb_j})^*]_{xyz}}{\partial \delta \theta_{wb_i}} \\
&= -2\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \frac{\partial \mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_i} \end{bmatrix} \otimes \mathbf{q}_{b_ib_j}}{\partial \delta \theta_{wb_i}} \\
&= -2\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} [\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i}]_L [\mathbf{q}_{b_ib_j}]_R \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{I} \end{bmatrix}
\end{aligned}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{v}_i^w} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{b}_i^a} = \mathbf{0}$$

$$\begin{aligned}
\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{b}_i^g} &= \frac{\partial 2[(\mathbf{q}_{b_ib_j} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_i^g}^q \delta \mathbf{b}_i^g \end{bmatrix})^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j}]_{xyz}}{\partial \delta \mathbf{b}_i^g} \\
&= -2\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} [\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i} \mathbf{q}_{b_ib_j}]_L \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{b_i^g}^q \end{bmatrix}
\end{aligned}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{p}_{wb_j}} = \mathbf{0}$$

$$\begin{aligned}
\frac{\partial \mathbf{r}_\theta}{\partial \delta \theta_{wb_j}} &= \frac{\partial 2[\mathbf{q}_{b_ib_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_j} \end{bmatrix}]_{xyz}}{\partial \delta \theta_{wb_j}} \\
&= 2\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} [\mathbf{q}_{b_ib_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j}]_L \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{I} \end{bmatrix}
\end{aligned}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{v}_j^w} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{b}_j^a} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{b}_j^g} = \mathbf{0}$$

# 雅可比推导

深蓝学院 shenlanxueyuan.com

- 对 v 残差求导

$$
\begin{aligned}
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{p}_{wb_i}} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \theta_{wb_i}} &= \left(\mathbf{R}_{b_i w}\left(\mathbf{v}_{wb_j} - \mathbf{v}_{wb_i} + \mathbf{g}^w \Delta t\right)\right)^{\wedge} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{v}_i^w} &= -\mathbf{R}_{b_i w} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{b}_i^a} &= \frac{\partial - \left(\bar{\beta}_{b_i b_j} + \mathbf{J}_{b_i^a}^{\beta} \delta \mathbf{b}_i^a + \mathbf{J}_{b_i^g}^{\beta} \delta \mathbf{b}_i^g\right)}{\partial \delta \mathbf{b}_i^a} = -\mathbf{J}_{b_i^a}^{\beta} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{b}_i^g} &= -\mathbf{J}_{b_i^g}^{\beta}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{p}_{wb_j}} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \theta_{wb_j}} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{v}_j^w} &= \mathbf{R}_{b_i w} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{b}_j^a} &= \mathbf{0} \\
\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{b}_j^g} &= \mathbf{0}
\end{aligned}
$$

# 雅可比推导

- 对 ba, bg 残差求导

$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{p}_{wb_i}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \theta_{wb_i}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{v}_i^w} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{b}_i^a} = \frac{\partial (\mathbf{b}_j^a - (\mathbf{b}_i^a + \delta \mathbf{b}_i^a))}{\partial \delta \mathbf{b}_i^a} = -\mathbf{I}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{b}_i^g} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{p}_{wb_j}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \theta_{wb_j}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{v}_j^w} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{b}_j^a} = \mathbf{I}$$
$$\frac{\partial \mathbf{r}_{b^a}}{\partial \delta \mathbf{b}_j^g} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{p}_{wb_i}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \theta_{wb_i}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{v}_i^w} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{b}_i^a} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{b}_i^g} = \frac{\partial (\mathbf{b}_j^g - (\mathbf{b}_i^g + \delta \mathbf{b}_i^g))}{\partial \delta \mathbf{b}_i^g} = -\mathbf{I}$$

$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{p}_{wb_j}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \theta_{wb_j}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{v}_j^w} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{b}_j^a} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta \mathbf{b}_j^g} = \mathbf{I}$$

# 纲要



- 第一部分：雅可比推导
- 第二部分：代码实现及效果分析
- 第三部分：编码器融合

# 代码实现

深蓝学院 shenlanxueyuan.com

- 预积分
  - 中值： w_mid, a_mid
  - 积分项： theta_ij, alpha_ij, beta_ij

```
// 1. get w_mid:
w_mid = 0.5 * (prev_w + curr_w);
// 2. update relative orientation, so3:
prev_theta_ij = state.theta_ij_;
d_theta_ij = Sophus::SO3d::exp(w_mid * T);
state.theta_ij_ *= d_theta_ij;
curr_theta_ij = state.theta_ij_;
// 3. get a_mid:
a_mid = 0.5 * (prev_theta_ij * prev_a + curr_theta_ij * curr_a);
// 4. update relative translation:
state.alpha_ij_ += state.beta_ij_ * T + 0.5 * a_mid * T * T;
// 5. update relative velocity:
state.beta_ij_ += a_mid * T;
```

# 代码实现

深蓝学院 shenlanxueyuan.com

- 方差

```cpp
// TODO: 4. update P_:
const MatrixF F = MatrixF::Identity() + T * F_;
const MatrixB B = T * B_;
P_ = F * P_ * F.transpose() + B * Q_ * B.transpose();
```

- 雅可比

```cpp
J_ = F * J_;
```

```cpp
// 1. intermediate results:
dR_inv = d_theta_ij.inverse().matrix();
prev_R = prev_theta_ij.matrix();
curr_R = curr_theta_ij.matrix();
prev_R_a_hat = prev_R * Sophus::SO3d::hat(prev_a);
curr_R_a_hat = curr_R * Sophus::SO3d::hat(curr_a);
```

```cpp
// F12 & F32:
F_.block<3, 3>(INDEX_ALPHA, INDEX_THETA) = -0.25 * T * (prev_R_a_hat + curr_R_a_hat * dR_inv);
F_.block<3, 3>(INDEX_BETA, INDEX_THETA) = -0.5 * (prev_R_a_hat + curr_R_a_hat * dR_inv);
// F14 & F34:
F_.block<3, 3>(INDEX_ALPHA, INDEX_B_A) = -0.25 * T * (prev_R + curr_R);
F_.block<3, 3>(INDEX_BETA, INDEX_B_A) = -0.5 * (prev_R + curr_R);
// F15 & F35:
F_.block<3, 3>(INDEX_ALPHA, INDEX_B_G) = 0.25 * T * T * curr_R_a_hat;
F_.block<3, 3>(INDEX_BETA, INDEX_B_G) = 0.5 * T * curr_R_a_hat;
// F22:
F_.block<3, 3>(INDEX_THETA, INDEX_THETA) = -Sophus::SO3d::hat(w_mid);
```

```cpp
// B11 & B31:
B_.block<3, 3>(INDEX_ALPHA, INDEX_M_ACC_PREV) = 0.25 * T * prev_R;
B_.block<3, 3>(INDEX_BETA, INDEX_M_ACC_PREV) = 0.5 * prev_R;
// B12 & B32:
B_.block<3, 3>(INDEX_ALPHA, INDEX_M_GYR_PREV) = -0.125 * T * T * curr_R_a_hat;
B_.block<3, 3>(INDEX_BETA, INDEX_M_GYR_PREV) = -0.25 * T * curr_R_a_hat;
// B13 & B33:
B_.block<3, 3>(INDEX_ALPHA, INDEX_M_ACC_CURR) = 0.25 * T * curr_R;
B_.block<3, 3>(INDEX_BETA, INDEX_M_ACC_CURR) = 0.5 * curr_R;
// B14 & B34:
B_.block<3, 3>(INDEX_ALPHA, INDEX_M_GYR_CURR) = -0.125 * T * T * curr_R_a_hat;
B_.block<3, 3>(INDEX_BETA, INDEX_M_GYR_CURR) = -0.25 * T * curr_R_a_hat;
```

# 代码实现

- 残差计算

```
_error.block<3, 1>(INDEX_P, 0) =
    ori_i.inverse() * (pos_j - pos_i - vel_i * T_ + 0.5 * g_ * T_ * T_) - alpha_ij;
_error.block<3, 1>(INDEX_R, 0) =
    2 * (Sophus::SO3d::exp(theta_ij).inverse() * ori_i.inverse() * ori_j).log();
_error.block<3, 1>(INDEX_V, 0) =
    ori_i.inverse() * (vel_j - vel_i + g_ * T_) - beta_ij;
_error.block<3, 1>(INDEX_A, 0) = b_a_j - b_a_i;
_error.block<3, 1>(INDEX_G, 0) = b_g_j - b_g_i;
```

- 使用 g2o 的自动求导，上一部分推导的雅可比并未在此实现

# 代码实现

- Vertex plus operation

```cpp
virtual void oplusImpl(const double *update) override {
    //
    // TODO: do update
    //
    const Eigen::Vector3d delta_pos = Eigen::Vector3d(
        update[PRVAG::INDEX_POS + 0], update[PRVAG::INDEX_POS + 1], update[PRVAG::INDEX_POS + 2]
    );
    const Sophus::SO3d delta_ori = Sophus::SO3d::exp(Eigen::Vector3d(
        update[PRVAG::INDEX_ORI + 0], update[PRVAG::INDEX_ORI + 1], update[PRVAG::INDEX_ORI + 2]
    ));
    const Eigen::Vector3d delta_vel = Eigen::Vector3d(
        update[PRVAG::INDEX_VEL + 0], update[PRVAG::INDEX_VEL + 1], update[PRVAG::INDEX_VEL + 2]
    );
    const Eigen::Vector3d delta_b_a = Eigen::Vector3d(
        update[PRVAG::INDEX_B_A + 0], update[PRVAG::INDEX_B_A + 1], update[PRVAG::INDEX_B_A + 2]
    );
    const Eigen::Vector3d delta_b_g = Eigen::Vector3d(
        update[PRVAG::INDEX_B_G + 0], update[PRVAG::INDEX_B_G + 1], update[PRVAG::INDEX_B_G + 2]
    );

    _estimate.pos += delta_pos;
    _estimate.ori *= delta_ori;
    _estimate.vel += delta_vel;
    _estimate.b_a += delta_b_a;
    _estimate.b_g += delta_b_g;
    updateDeltaBiases(delta_b_a, delta_b_g);
}
```

# 效果分析



- cost 变化
  - 不加 imu： 191599 → 1204.9
  - 加 imu： 9.10146e8 → 307585

```
I0325 18:19:40.204689 30762 g2o_graph_optimizer.cpp:72]
------ Finish Iteration 44 of Backend Optimization -------
        Num. Vertices: 1913
        Num. Edges: 5794
        Num. Iterations: 25/512
        Time Consumption: 3.40843
        Cost Change: 191599--->1204.19
```

```
I0325 17:41:26.744853 29087 g2o_graph_optimizer.cpp:72]
------ Finish Iteration 44 of Backend Optimization -------
        Num. Vertices: 1913
        Num. Edges: 7706
        Num. Iterations: 17/512
        Time Consumption: 3.1373
        Cost Change: 9.10147e+08--->307585
```

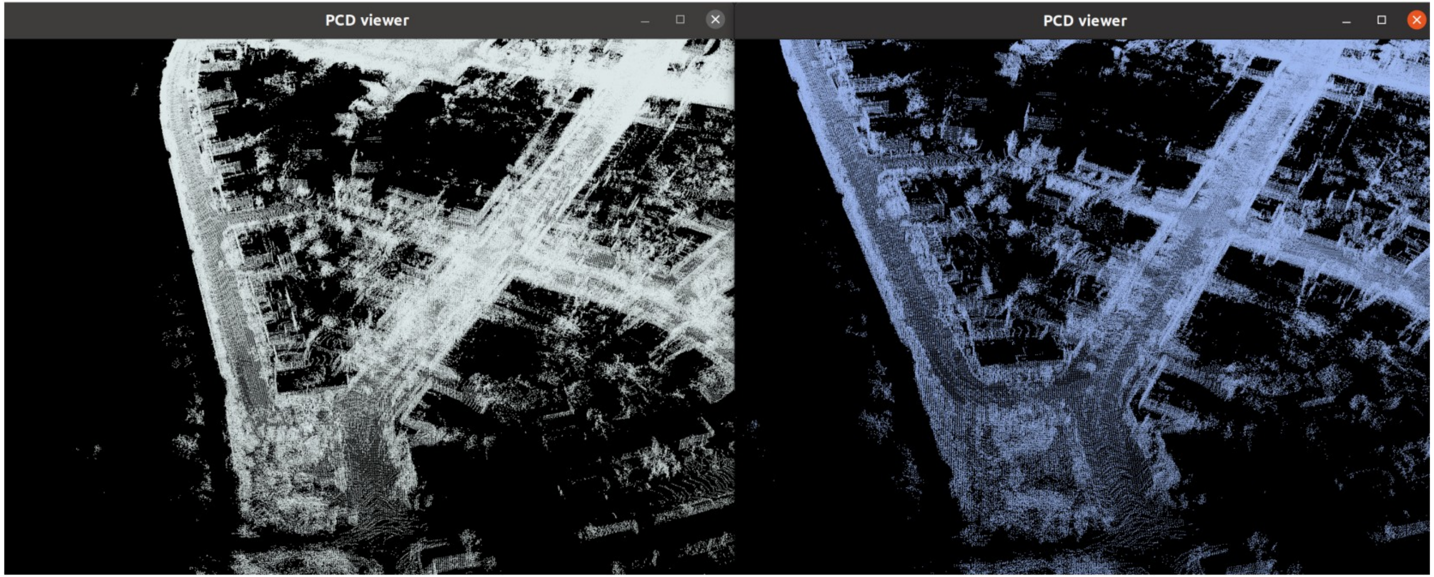
- 轨迹精度
  - 加与不加 imu 轨迹的精度相差不大，都与 laser_odom 相似

| | EVO APE |
|---|---|
| laser_odom | APE w.r.t. full transformation (unit-less) (not aligned)<br>max 29.330464<br>mean 11.783149<br>median 11.009685<br>min 0.000001<br>rmse 13.870033<br>sse 368018.763763<br>std 7.316777 |
| optimized (with imu pre-integration) | APE w.r.t. full transformation (unit-less) (not aligned)<br>max 29.052853<br>mean 11.908738<br>median 11.903517<br>min 0.271909<br>rmse 13.917529<br>sse 370543.549087<br>std 7.202748 |
| optimized (without imu pre-integration) | APE w.r.t. full transformation (unit-less) (not aligned)<br>max 29.371823<br>mean 11.795292<br>median 11.359344<br>min 0.256469<br>rmse 13.872213<br>sse 368134.469642<br>std 7.301328 |

# 效果分析

● 建图效果（左、右图分别是加与不加 imu 所建的地图）

- 左图在垂直方向上有重影，由此推测部分 imu 数据在垂直方向或俯仰角上有问题



# 纲要

- 第一部分：雅可比推导
- 第二部分：代码实现及效果分析
- 第三部分：编码器融合

# 编码器融合公式推导



In the case when IMU provides angular velocity, wheel encoder provides linear velicoty:

$$\omega_k^b = \begin{bmatrix} \omega_{xk} \\ \omega_{yk} \\ \omega_{zk} \end{bmatrix}, \mathbf{v}_k^b = \begin{bmatrix} v_{xk} \\ 0 \\ 0 \end{bmatrix}$$

Integrating from time `i` to time `j`:

$$\begin{aligned}
\mathbf{p}_{wb_j} &= \mathbf{p}_{wb_i} + \int_{t\in[i,j]} \mathbf{q}_{wb_t} \mathbf{v}_t^b \delta t \\
&= \mathbf{p}_{wb_i} + \mathbf{q}_{wb_i} \int_{t\in[i,j]} (\mathbf{q}_{b_i b_t} \mathbf{v}_t^b) \delta t \\
\mathbf{q}_{wb_j} &= \int_{t\in[i,j]} \mathbf{q}_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega_t^b \end{bmatrix} \delta t \\
&= \mathbf{q}_{wb_i} \int_{t\in[i,j]} \mathbf{q}_{b_i b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega_t^b \end{bmatrix} \delta t
\end{aligned}$$

Define the pre-integration terms:

$$\begin{aligned}
\alpha_{b_i b_j} &= \int_{t\in[i,j]} (\mathbf{q}_{b_i b_t} \mathbf{v}_t^b) \delta t \\
\mathbf{q}_{b_i b_j} &= \int_{t\in[i,j]} \mathbf{q}_{b_i b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega_t^b \end{bmatrix} \delta t
\end{aligned}$$

# 编码器融合公式推导

深蓝学院
shenlanxueyuan.com

With mid-value method:

$$\omega^b = \frac{1}{2}[(\omega^{b_k} - \mathbf{b}_k^g) + (\omega^{b_{k+1}} - \mathbf{b}_k^g)]$$
$$\mathbf{v}^b = \frac{1}{2}(\mathbf{q}_{b_i b_k}\mathbf{v}^{b_k} + \mathbf{q}_{b_i b_{k+1}}\mathbf{v}^{b_{k+1}})$$

The discrete and iterative form:

$$\alpha_{b_i b_{k+1}} = \alpha_{b_i b_k} + \mathbf{v}^b \delta t$$
$$\mathbf{q}_{b_i b_{k+1}} = \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^b \delta t \end{bmatrix}$$

The residual is:

$$\begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_\theta \\ \mathbf{r}_{b^g} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{wb_i}^*(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i}) - \alpha_{b_i b_j} \\ 2[\mathbf{q}_{b_i b_j}^* \otimes (\mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j})]_{xyz} \\ \mathbf{b}_j^g - \mathbf{b}_i^g \end{bmatrix}$$

State:

$$\begin{bmatrix} \delta\mathbf{p}_{wb_i} & \delta\theta_{wb_i} & \delta\mathbf{p}_{wb_j} & \delta\theta_{wb_j} \end{bmatrix}$$

Jacobians:

$$\frac{\partial \mathbf{r}_p}{\partial \delta\mathbf{p}_{wb_i}} = \frac{\partial - \mathbf{q}_{wb_i}^*(\mathbf{p}_{wb_i} + \delta\mathbf{p}_{wb_i})}{\partial \delta\mathbf{p}_{wb_i}} = -\mathbf{R}_{b_i w}$$

$$\frac{\partial \mathbf{r}_p}{\partial \delta\theta_{wb_i}} = \frac{\partial(\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_i} \end{bmatrix})^*(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i})}{\partial \delta\theta_{wb_i}} = (\mathbf{R}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i}))^\wedge$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta\mathbf{p}_{wb_i}} = \mathbf{0}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta\theta_{wb_i}} = \frac{\partial 2[\mathbf{q}_{b_i b_j}^* \otimes (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta_{wb_i} \end{bmatrix})^* \otimes \mathbf{q}_{wb_j}]_{xyz}}{\partial \delta\theta_{wb_i}} = -2\begin{bmatrix} 0 & \mathbf{I} \end{bmatrix}[\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i}]_L [\mathbf{q}_{b_i b_j}]_R \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{I} \end{bmatrix}$$

$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta\mathbf{p}_{wb_i}} = \mathbf{0}$$
$$\frac{\partial \mathbf{r}_{b^g}}{\partial \delta\theta_{wb_i}} = \mathbf{0}$$

# 编码器融合公式推导

深蓝学院 shenlanxueyuan.com

For covariance propagation, need to find out the state transition equation:

$$\dot{\mathbf{x}} = \mathbf{F}_t \mathbf{x} + \mathbf{B}_t \mathbf{w}$$
$$\mathbf{P}_{i,k+1} = \mathbf{F}_k \mathbf{P}_{i,k} \mathbf{F}_k^T + \mathbf{B}_k \mathbf{Q} \mathbf{B}_k$$

From the differential equations:

$$\begin{aligned}
\delta\dot{\alpha}_t^{b_k} &= -\mathbf{R}_t^{wb} v_t^{\wedge} \delta\theta_t^{b_k} + \mathbf{R}_t^{wb} \mathbf{n}_v \\
\delta\dot{\theta}_t^{b_k} &= -(\omega_t - \mathbf{b}_{\omega_t})^{\wedge} \delta\theta_t^{b_k} + \mathbf{n}_\omega
\end{aligned}$$

Discrete form:

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{w}_k$$

$$\mathbf{x}_{k+1} = \begin{bmatrix} \delta\alpha_{k+1} \\ \delta\theta_{k+1} \end{bmatrix} \quad \mathbf{x}_k = \begin{bmatrix} \delta\alpha_k \\ \delta\theta_k \end{bmatrix} \quad \mathbf{w}_k = \begin{bmatrix} \delta\mathbf{n}_{v_k} \\ \delta\mathbf{n}_{\omega_k} \\ \delta\mathbf{n}_{v_{k+1}} \\ \delta\mathbf{n}_{\omega_{k+1}} \end{bmatrix}$$

$$\begin{aligned}
\delta\dot{\theta}_k &= -\left(\frac{\omega_k + \omega_{k+1}}{2} - \mathbf{b}_{\omega_t}\right)^{\wedge} \delta\theta_k + \frac{\mathbf{n}_{\omega_k} + \mathbf{n}_{\omega_{k+1}}}{2} \\
\delta\theta_{k+1} &= (\mathbf{I} - \bar{\omega}^{\wedge}\delta t)\delta\theta_k + \frac{\delta t}{2}\mathbf{n}_{\omega_k} + \frac{\delta t}{2}\mathbf{n}_{\omega_{k+1}}
\end{aligned}$$

$$\begin{aligned}
\delta\dot{\alpha}_k &= -\frac{1}{2}\mathbf{R}_k \mathbf{v}_k^{\wedge}\delta\theta_k - \frac{1}{2}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\delta\theta_{k+1} + \frac{1}{2}\mathbf{R}_k \mathbf{n}_{v_k} + \frac{1}{2}\mathbf{R}_{k+1}\mathbf{n}_{v_{k+1}} \\
&= -\frac{1}{2}\mathbf{R}_k \mathbf{v}_k^{\wedge}\delta\theta_k - \frac{1}{2}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\left((\mathbf{I} - \bar{\omega}^{\wedge}\delta t)\delta\theta_k + \frac{\delta t}{2}\mathbf{n}_{\omega_k} + \frac{\delta t}{2}\mathbf{n}_{\omega_{k+1}}\right) \\
&\quad + \frac{1}{2}\mathbf{R}_k \mathbf{n}_{v_k} + \frac{1}{2}\mathbf{R}_{k+1}\mathbf{n}_{v_{k+1}} \\
&= -\frac{1}{2}[\mathbf{R}_k \mathbf{v}_k^{\wedge} + \mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}(\mathbf{I} - \bar{\omega}^{\wedge}\delta t)]\delta\theta_k \\
&\quad - \frac{\delta t}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\mathbf{n}_{\omega_k} \\
&\quad - \frac{\delta t}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\mathbf{n}_{\omega_{k+1}} \\
&\quad + \frac{1}{2}\mathbf{R}_k \mathbf{n}_{v_k} \\
&\quad + \frac{1}{2}\mathbf{R}_{k+1}\mathbf{n}_{v_{k+1}} \\
\delta\alpha_{k+1} &= \delta\alpha_k \\
&\quad - \frac{\delta t}{2}[\mathbf{R}_k \mathbf{v}_k^{\wedge} + \mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}(\mathbf{I} - \bar{\omega}^{\wedge}\delta t)]\delta\theta_k \\
&\quad - \frac{\delta t^2}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\mathbf{n}_{\omega_k} \\
&\quad - \frac{\delta t^2}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^{\wedge}\mathbf{n}_{\omega_{k+1}} \\
&\quad + \frac{\delta t}{2}\mathbf{R}_k \mathbf{n}_{v_k} \\
&\quad + \frac{\delta t}{2}\mathbf{R}_{k+1}\mathbf{n}_{v_{k+1}}
\end{aligned}$$

# 编码器融合公式推导



So `F` and `B` matrix are:

$$\mathbf{F}_k = \mathbf{I}_6 + \delta t \begin{bmatrix} 0 & -\frac{1}{2}[\mathbf{R}_k \mathbf{v}_k^\wedge + \mathbf{R}_{k+1}\mathbf{v}_{k+1}^\wedge(\mathbf{I} - \bar{\omega}^\wedge \delta t)] \\ 0 & -\bar{\omega}^\wedge \end{bmatrix}$$

$$\mathbf{B}_k = \delta t \begin{bmatrix} \frac{1}{2}\mathbf{R}_k & -\frac{\delta t}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^\wedge & \frac{1}{2}\mathbf{R}_{k+1} & -\frac{\delta t}{4}\mathbf{R}_{k+1}\mathbf{v}_{k+1}^\wedge \\ 0 & \frac{1}{2}\mathbf{I} & 0 & \frac{1}{2}\mathbf{I} \end{bmatrix}$$

Jacobian update:

$$\mathbf{J}_{k+1} = \mathbf{F}_k \mathbf{J}_k$$

# 在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

# 感谢各位聆听！

Thanks for Listening