FAST-LIO English Version

Kihoon Shin @ ASL, INHA Univ

June 2023

1 Introduction

저자에게 허락을 받고 중국어 버전을 번역기를 이용하여 영어로 번역하였다.

2 Methodology

2.1 State Estimation

This section uses the error-state Kalman filter to estimate the state quantity x. We first clarify the definition in the text

- x indicates the truth value
- $\bar{x}$ Indicates the optimal estimate obtained after the filtering algorithm ends
- $\hat{x}$ Indicates the estimated value obtained during the filtering algorithm
- $\tilde{x}$ Indicates the error value

Suppose we are now ready to estimate the state of the k frame, then the estimation of the $k-1$ frame is completed. Therefore, we define the error state

$$\tilde{x}_{k-1} \doteq x_{k-1} \boxminus \bar{x}_{k-1} = \begin{bmatrix} \delta\theta^T & {}^G\tilde{p}_I^T & {}^G\tilde{v}_I^T & \tilde{b}_w^T & \tilde{b}_a^T & {}^G\tilde{g}_I^T \end{bmatrix} \tag{1}$$

Furthermore, the covariance matrix is defined as $\bar{P}_{k-1}$.

2.2 Forward Propagation

This section is about forward propagation. The forward propagation actually has two contents: one is to calculate a rough state quantity $\hat{x}_k$(see Figure 1) through the IMU integration, and this state quantity will be used in the subsequent backpropagation to compensate for motion distortion. This formula is as follows:

$$\hat{x}_{i+1} = \hat{x}_i \boxplus (\Delta t f(\hat{x}_i, u_i, 0)); \ \hat{x}_0 = \bar{x}_{k-1} \tag{2}$$

Where $\Delta t = \tau_{i+1} - \tau_i$, which represents the time difference between two IMU frames. The value of noise $w_i = 0$ here is because we don't know the actual size of the noise, so it is set to zero during propagation(the noise will actually be considered in the subsequent error state equation). We will perform the above calculation every time we receive an IMU until the last IMU frame is calculated.

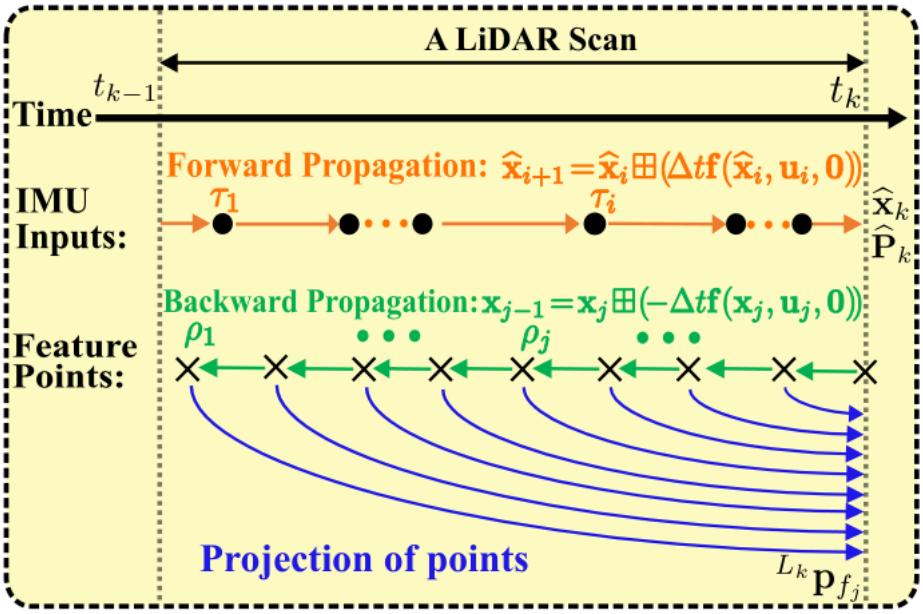

Figure 1: Forward and Backward Propagation

Another content is to propagate the error amount and calculate the corresponding covariance matrix. Readers may wonder here: We don't know the true value, then how can we calculate the error? In fact, the amount of error we calculate is also an approximate value, so it will have a corresponding covariance matrix to judge the confidence. The error-state Kalman filter(ESKF) used in this paper takes the error amount as the quantity to be estimated, that is,

$$\tilde{x}_k = \tilde{f}(\tilde{x}_{k-1}, u_k) + \tilde{\omega}_k \tag{3}$$

$$z_k = \tilde{h}(\tilde{x}_k) + \tilde{v}_k \tag{4}$$

what we want to estimate now is the error state $\tilde{x}$, instead of directly estimating the true state x. And with the estimation of the error amount, directly adding the estimation of the state quantity is the optimal estimate we obtained. This will bring the following benefits

- In the processing of rotation, the state variable of ESKF can be expressed with minimized parameters, that is, three-dimensional variables are used to express the increment of rotation. The traditional KF needs to use quaternions (4 dimensions) or higher-dimensional expressions (rotation matrix, 9 dimensions), or use singular expressions (Euler angles)
- ESKF is always near the origin, far away from the singular point, and will not cause insufficient linearization approximation due to being far away from the operating point
- The state quantity of ESKF is a small quantity, and its second-order variables can be ignored relatively. At the same time, most Jacobian matrices become very simple and can even be replaced by identity matrices

After understanding the benefits of ESKF, what we need to do now is to find the specific expressions of $\tilde{f}, \tilde{h}$ and linearize them. Back in the paper, for the equation of motion, the error model is as follows:

$$\tilde{x}_{i+1} = x_{i+1} \boxminus \hat{x}_{i+1} \tag{5}$$
$$= (x_i \boxplus \Delta t f(x_i, u_i, w_i)) \boxminus (\hat{x}_i \boxplus \Delta t f(\hat{x}_i, u_i, 0)) \tag{6}$$
$$\simeq F_{\tilde{x}} \tilde{x}_i + F_w w_i \tag{7}$$

Here $F_{\tilde{x}}$ and F_w are two large sparse matrices, see the paper for the specific form. These two troublesome guys we'll deduce it later. Assuming that the covariance matrix of white noise w is Q, the covariance matrix can be propagated as follows:

$$\hat{P}_{i+1} = F_{\tilde{x}} \hat{P}_i F_{\tilde{x}} + F_w Q F_w^T; \qquad \hat{P}_0 = \bar{P}_{k-1} \tag{8}$$

This formula is derived from the fundamental properties of the Gaussian distribution operation. So far we have completed the two contents of the forward propagation, that is, the state estimation $\hat{x}_k$ is obtained, the error state propagation is derived, and the covariance $\hat{P}_k$ of the error between the real state and the estimated state is obtained. Next we come to derive (7). In fact, there are two ways to derive the error state propagation of ESKF. One is a method similar to that in [5], which is based on the recurrence equation of the error with time. If we can derive the derivative relationship of state error with time, for example $\dot{\delta \boldsymbol{x}_k} = \boldsymbol{A}\delta\boldsymbol{x} + B\boldsymbol{w}$. Then the transfer equation of the error state is:

$$\delta \boldsymbol{x}_k = \delta \boldsymbol{x}_{k-1} + \dot{\delta \boldsymbol{x}}_{k-1} \Delta t \tag{9}$$

What we focus on is the error propagation method based on a Taylor expansion used in the paper. The linear recurrence relation of the state error of the nonlinear system $\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})$ is as follows:

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{10}$$
$$\rightarrow \hat{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k = f(\hat{\boldsymbol{x}}_{k-1} + \delta \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{11}$$
$$\xrightarrow{Taylor} \underline{\hat{\boldsymbol{x}}_k} + \delta \boldsymbol{x}_k = \underline{f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)} + \boldsymbol{F}\delta \boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{w}_{k-1} \tag{12}$$
$$\rightarrow \delta \boldsymbol{x}_k = \boldsymbol{F}\delta \boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{w}_{k-1} \tag{13}$$

Among them, $\boldsymbol{F}$ is the Jacobian matrix of the state quantity $\boldsymbol{x}_k$ to the state quantity $\boldsymbol{x}_{k-1}$ and $\boldsymbol{G}$ is the Jacobian matrix of the state quantity x_k to the noise quantity w_{k-1}. From now, $\mathbf{F}_{\tilde{\mathbf{x}}}$ in the paper corresponds to $\mathbf{F}$ in the above formula, and $\mathbf{F}_{\mathbf{w}}$ corresponds to $\mathbf{G}$ in the above formula.

- $\tilde{\boldsymbol{x}}_{i+1} = \boldsymbol{x}_{i+1} \boxminus \hat{\boldsymbol{x}}_{i+1} = (\boldsymbol{x}_i \boxplus \Delta t \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{w}_i)) \boxminus (\hat{\boldsymbol{x}}_i \boxplus \Delta t \boldsymbol{f}(\hat{\boldsymbol{x}}_i, \boldsymbol{u}_i, 0))$
- $\mathbf{x}_i = \hat{\mathbf{x}}_i \boxplus \tilde{\mathbf{x}}_i$
- Define $\mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i) = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)\Delta t = \mathbf{f}(\hat{\mathbf{x}}_\mathbf{i} \boxplus \tilde{\mathbf{x}}_\mathbf{i}, \mathbf{u}_\mathbf{i}, \mathbf{w}_\mathbf{i})\Delta t$

Then formula (6) $\tilde{x}_{i+1} = (x_i \boxplus \Delta t f(x_i, u_i, w_i)) \boxminus (\hat{x}_i \boxplus \Delta t f(\hat{x}_i, u_i, 0))$ can be written as:

$$\tilde{\mathbf{x}}_{i+1} = ((\hat{\mathbf{x}}_i \boxplus \tilde{\mathbf{x}}_i) \boxplus \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i)) \boxminus (\hat{\mathbf{x}}_i \boxplus \mathbf{g}(0, 0)) \stackrel{def}{=} \mathbf{G}(\tilde{\mathbf{x}}_i, \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i)) \tag{14}$$

For simplicity, we omit the annoying subscript i in the above formula in the subsequent derivation. What we want is the partial derivative of $\mathbf{G}(\tilde{\mathbf{x}}_i, \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i))$ w.r.t $\tilde{\mathbf{x}}$ and $\mathbf{w}$ (near $\tilde{\mathbf{x}} = \mathbf{w} = 0$ because

$\tilde{\mathbf{x}}$ is an error close to 0 and w is noise). This is a compound function, so there is:

$$\mathbf{F}_{\tilde{\mathbf{x}}} = \frac{\partial G(\tilde{x}, g(0,0))}{\partial \tilde{x}} + \frac{\partial G(0, g(\tilde{x},0))}{\partial g(\tilde{x},0)} \frac{\partial g(\tilde{x},0)}{\partial \tilde{x}}\bigg|_{\tilde{x}=0} \overset{simplify}{\Longrightarrow} \frac{\partial G}{\partial \tilde{x}} + \frac{\partial G}{\partial g}\frac{\partial g}{\partial \tilde{x}}\bigg|_{\tilde{x}=0} \tag{15}$$

$$\mathbf{F}_{w} = \frac{\partial G(0, g(0,w))}{\partial g(0,w)} \frac{\partial g(0,w)}{\partial w}\bigg|_{w=0} \overset{simplify}{\Longrightarrow} \frac{\partial G}{\partial g_w}\frac{\partial g}{\partial w}\bigg|_{w=0} \tag{16}$$

Next, we first consider the partial derivatives of such a manifold. For the manifold $\mathbf{E} = ((\mathbf{a} \boxplus \mathbf{b}) \boxplus \mathbf{c}) \boxminus \mathbf{d}$, have the following conclusions [3]:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{b}} = \frac{\partial \mathbf{E}}{\partial \mathbf{c}} = \mathbf{I}_{n\times n} \qquad a, b, c, d \in \mathbb{R}^n \tag{17}$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{b}} = \mathbf{A}(\mathbf{E})^{-\mathbf{T}} \mathbf{Exp}(-\mathbf{c}) \mathbf{A}(\mathbf{b})^{\mathbf{T}}, \frac{\partial \mathbf{E}}{\partial \mathbf{c}} = \mathbf{A}(\mathbf{E})^{-\mathbf{T}} \mathbf{A}(\mathbf{c})^{\mathbf{T}} \qquad a, d \in \mathcal{SO}(3), \quad b, c \in \mathbb{R}^3 \tag{18}$$

Where

$$\mathbf{A}(\mathbf{u})^{-1} = \mathbf{I} - \frac{1}{2}\lfloor \mathbf{u} \rfloor \wedge + (1 - \alpha(\|\mathbf{u}\|)) \frac{\lfloor \mathbf{u} \rfloor_{\wedge}^2}{\|\mathbf{u}\|^2} \tag{19}$$

$$\alpha(m) = \frac{m}{2} \cot\left(\frac{m}{2}\right) = \frac{m}{2} \frac{\cos(m/2)}{\sin(m/2)} \tag{20}$$

********** Here is my note **********

The $\mathbf{A}(\mathbf{u})^{-\mathbf{1}}$ is Left Inverse Jacobian $\mathbf{J_l}(\mathbf{u})^{-\mathbf{1}}$

Proof)

$$\mathbf{J_l^{-1}}(\theta) = \mathbf{I} - \frac{\mathbf{1}}{\mathbf{2}}\hat{\theta} + (\mathbf{I} - \alpha(\|\theta\|)) \frac{\hat{\theta}^{\mathbf{2}}}{\|\theta\|^{\mathbf{2}}}$$

$$\alpha(\mathbf{m}) = \frac{\mathbf{m}\cos(\frac{\mathbf{m}}{\mathbf{2}})}{\mathbf{2}\sin(\frac{\mathbf{m}}{\mathbf{2}})}$$

$$\therefore \mathbf{J_l^{-1}}(\theta) = \mathbf{I} - \frac{\mathbf{1}}{\mathbf{2}}\hat{\theta} + \hat{\theta}^{\mathbf{2}} \left(\frac{\mathbf{1}}{\|\theta\|^{\mathbf{2}}} - \frac{\cos(\frac{\|\theta\|}{\mathbf{2}})}{\mathbf{2}\|\theta\|\sin(\frac{\|\theta\|}{\mathbf{2}})} \right)$$

$$\textbf{where,} \qquad \sin x = 2\sin(\frac{x}{2})\cos(\frac{x}{2}), \quad \cos^2 x = \frac{1 + cos(2x)}{2}$$

$$= I - \frac{1}{2}\hat{\theta} + \hat{\theta}^2 \left(\frac{1}{\|\theta\|^2} - \frac{\cos\left(\frac{\|\theta\|}{2}\right)}{2\|\theta\| \cdot \frac{\sin(\|\theta\|)}{2\cos(\frac{\|\theta\|}{2})}} \right)$$

$$= I - \frac{1}{2}\hat{\theta} + \hat{\theta}^2 \left(\frac{1}{\|\theta\|^2} - \frac{\cos^2(\frac{\|\theta\|}{2})}{\|\theta\|\sin(\|\theta\|)} \right)$$

$$= I - \frac{1}{2}\hat{\theta} + \hat{\theta}^2 \left(\frac{1}{\|\theta\|^2} - \frac{1 + \cos(\|\theta\|)}{2\|\theta\|\sin(\|\theta\|)} \right)$$

This is same as shown in *Sensing & Estimation in Robotics Lecture 12: SO(3) and SE(3) Geometry and Kinematics'*

********** End of my note **********

That is, the Jacobian matrix in the BCH approximation. For formula (17), this is obvious, because $\boxplus, \boxminus$ in Euclidian space is the commonly used addition and subtraction. We next derive (18). The partial derivative w.r.t b is as follows:

$$E = ((a \boxplus b) \boxplus c) \boxminus d = Log(d^{-1} \cdot (a \boxplus b) \boxplus c)) \tag{21}$$

$$\rightarrow Exp(E) = d^{-1} \cdot a \cdot Exp(b) \cdot Exp(c) \tag{22}$$

$$\rightarrow Exp(E + \Delta E) = d^{-1} \cdot a \cdot Exp(b + \Delta b) \cdot Exp(c) \tag{23}$$

$$\xrightarrow[BCH]{} Exp(E)Exp(A(E)^T \Delta E) = d^{-1} \cdot a \cdot Exp(b)Exp(A(b)^T \Delta b) \cdot Exp(c) \tag{24}$$

$$\rightarrow Exp(A(E)^T \Delta E) = Exp(-c)Exp(A(b)^T \Delta b) \cdot Exp(c) \tag{25}$$

$$\rightarrow Exp(A(E)^T \Delta E) \overset{Adjoint}{=} Exp(Exp(-c)A(b)^T \Delta b) \tag{26}$$

$$\rightarrow \frac{\Delta E}{\Delta b} = A(E)^{-T} Exp(-c) A(b)^T \tag{27}$$

********** Here is my note **********

BCH Formula

$$\begin{aligned}
&Exp(\theta + \delta\theta) \\
&\simeq Exp(\theta)Exp(J_r(\theta)\delta\theta) \qquad \textbf{Right jacobian} \\
&\simeq Exp(J_l(\theta)\delta\theta)Exp(\theta) \qquad \textbf{Left jacobian} \\
&\text{using} \quad J_r(\theta) = J_l(-\theta) = J_l(\theta)^T \\
&Exp(\theta + \delta\theta) \simeq Exp(\theta)Exp(J_r(\theta)\delta\theta) = Exp(\theta)Exp(J_l(\theta)^T\delta\theta)
\end{aligned}$$

so(3), SO(3) Identities

$$R = exp(\hat{\theta}) = \sum_{n=0}^{\infty} \frac{1}{n!}\hat{\theta}^n = I + \left(\frac{\sin\|\theta\|}{\|\theta\|}\right)\hat{\theta} + \left(\frac{1 - \cos\|\theta\|}{\|\theta\|^2}\right)\hat{\theta}^2 \simeq I + \hat{\theta}$$

$$R^{-1} = R^T = exp(-\hat{\theta}) = \sum_{n=0}^{\infty} \frac{1}{n!}(-\hat{\theta})^n \simeq I - \hat{\theta}$$

$$\hat{\theta}^T = -\hat{\theta}$$

$$\exp((R\mathbf{m})^\wedge) = R\exp(\hat{\mathbf{m}})R^T, \quad \mathbf{m} \in \mathbb{R}^3 \qquad \text{used in eq(26)}$$

********** End of my note **********

The formula (24) uses the BCH approximation and formula (26) uses Adjoint properties of the matrix on SO(3). The same is true for partial derivatives of c:

$$\rightarrow Exp(E + \Delta E) = d^{-1} \cdot a \cdot Exp(b) \cdot Exp(c + \Delta c) \tag{28}$$

$$\xrightarrow[BCH]{} Exp(E)Exp(A(E)^T \Delta E) = d^{-1} \cdot a \cdot Exp(b) \cdot Exp(c)Exp(A(c)^T \Delta c) \tag{29}$$

$$\rightarrow Exp(A(E)^T \Delta E) = Exp(A(c)^T \Delta c) \tag{30}$$

$$\rightarrow \frac{\Delta E}{\Delta c} = A(E)^{-T} A(c)^T \tag{31}$$

So far we have completed the derivation of formula (18). Going back to formula (15), (16) $\mathbf{G}$ is in the form of $((\mathbf{a} \boxplus \mathbf{b}) \boxplus \mathbf{c}) \boxminus \mathbf{d}$. Because $\mathbf{G} = ((\hat{\mathbf{x}}_i \boxplus \tilde{\mathbf{x}}_i) \boxplus \mathbf{g}(\tilde{\mathbf{x}}_i, \mathbf{w}_i)) \boxminus (\hat{\mathbf{x}}_i \boxplus \mathbf{g}(0, 0))$. From

$\mathbf{F}_{\tilde{x}} = \frac{\partial G}{\partial \tilde{x}} + \frac{\partial G}{\partial g}\frac{\partial g}{\partial \tilde{x}}\Big|_{\tilde{x}=0}$, we derive:

$$\frac{\partial G}{\partial \tilde{x}}\Big|_{\tilde{x}=0} \overset{(18)}{=\!=} \begin{bmatrix} \mathbf{A(0)^{-1}Exp(-g(0,0))A(0)^T} & 0 \\ 0 & \mathbf{I_{15\times 15}} \end{bmatrix} \tag{32}$$

$$= \begin{bmatrix} \mathbf{Exp(-f(\hat{x}, u, 0)\Delta t)} & 0 \\ 0 & \mathbf{I_{15\times 15}} \end{bmatrix} \tag{33}$$

$$= \begin{bmatrix} \mathbf{Exp}(-(\omega_{\mathbf{m}} - \hat{\mathbf{b}}_{\mathbf{w}})\mathbf{\Delta t}) & 0 \\ 0 & \mathbf{I_{15\times 15}} \end{bmatrix} \tag{34}$$

When $\tilde{x} = 0$, $G = 0$ and $A(0) = I$(easy to verify). Since the first item of the state quantity is SO(3), the other items are all vectors, so only the first three lines of the matrix in the formula are substituted into formula (18), and the other lines are substituted into formula (17). Similarly, We can derive

$$\frac{\partial G}{\partial g}\Big|_{\tilde{x}=0} \overset{(18)}{=\!=} \begin{bmatrix} \mathbf{A(0)^{-1}A(g(0,0))^T} & 0 \\ 0 & \mathbf{I_{15\times 15}} \end{bmatrix} \tag{35}$$

$$= \begin{bmatrix} \mathbf{A}((\omega_m - \hat{\mathbf{b}}_\omega)\Delta t) & 0 \\ 0 & \mathbf{I_{15\times 15}} \end{bmatrix} \tag{36}$$

Finally, regarding the solution $\frac{\partial g}{\partial \tilde{x}}$ due to $\tilde{\mathbf{x}} \doteq \mathbf{x} \boxminus \hat{\mathbf{x}} = \begin{bmatrix} \delta\theta^T & {}^G\tilde{\mathbf{p}}_I^T & {}^G\tilde{\mathbf{v}}_I^T & \tilde{\mathbf{b}}_w^T & \tilde{\mathbf{b}}_a^T & {}^G\tilde{\mathbf{g}}^T \end{bmatrix}^T$, $\mathbf{w} \doteq \begin{bmatrix} \mathbf{n}_w^T & \mathbf{n}_a^T & \mathbf{n}_{bw}^T & \mathbf{n}_{ba}^T \end{bmatrix}^T$ as well as:

$$\mathbf{g}(\tilde{\mathbf{x}}, \mathbf{w}) = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w})\mathbf{\Delta t} = \begin{bmatrix} \omega_{\mathbf{m}} - \mathbf{b}_\omega - \mathbf{n}_\omega \\ {}^G\mathbf{v}_I \\ \mathbf{R_I}(\mathbf{a_m} - \mathbf{b_a} - \mathbf{n_a}) + {}^{\mathbf{G}}\mathbf{g} \\ \mathbf{n_{b\omega}} \\ \mathbf{n_{ba}} \\ \mathbf{0_{3\times 1}} \end{bmatrix} \Delta t \tag{37}$$

$$= \begin{bmatrix} \boldsymbol{\omega_m} - \hat{\mathbf{b}}_{\boldsymbol{\omega}} - \tilde{\mathbf{b}}_{\boldsymbol{\omega}} - \boldsymbol{n_\omega} \\ {}^G\hat{\mathbf{v}}_I + {}^G\tilde{\mathbf{v}}_I \\ \hat{\mathbf{R}}_{\mathbf{I}}\mathbf{Exp}(\delta\boldsymbol{\theta}^{\mathbf{T}})(\mathbf{a_m} - \hat{\mathbf{b}}_{\mathbf{a}} - \tilde{\mathbf{b}}_{\mathbf{a}} - \mathbf{n_a}) + {}^{\mathbf{G}}\mathbf{g} \\ \boldsymbol{n_{b\omega}} \\ \mathbf{n_{ba}} \\ \mathbf{0_{3\times 1}} \end{bmatrix} \Delta t \tag{38}$$

So,

$$\frac{\partial g(\tilde{\boldsymbol{x}}, 0)}{\partial \tilde{x}}\Big|_{\tilde{x}=0} = \begin{pmatrix} 0 & 0 & 0 & -\boldsymbol{I}_{3\times 3}\Delta t & 0 & 0 \\ 0 & 0 & \boldsymbol{I}_{3\times 3}\Delta t & 0 & 0 & 0 \\ -\hat{\boldsymbol{R}}_{\boldsymbol{I}}(\boldsymbol{a_m} - \hat{\boldsymbol{b}}_{\boldsymbol{a}})^\wedge \boldsymbol{\Delta t} & 0 & 0 & 0 & -\hat{\boldsymbol{R}}_{\boldsymbol{I}}\Delta t & \boldsymbol{I}_{3\times 3}\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{39}$$

Among them, the derivation of R in the third line uses the right multiplication perturbation model.

note) The first column in 3rd line derivations

$$\frac{\partial R\mathbf{s}}{\partial \theta} = -\mathbf{R}\mathbf{s}^\wedge \mathbf{J_l}(\theta) \simeq -\mathbf{R}\mathbf{s}^\wedge \qquad \text{when} \quad \theta \simeq \mathbf{0} \tag{40}$$

So far, according to formula (15), $\mathbf{F}_{\tilde{x}} = \frac{\partial G}{\partial \tilde{x}} + \frac{\partial G}{\partial g}\frac{\partial g}{\partial \tilde{x}}\Big|_{\tilde{x}=0}$, Adding the unobtrusive subscript i, we get:

$$\boldsymbol{F}_{\tilde{\boldsymbol{x}}} = \begin{bmatrix} \boldsymbol{Exp}(-(\boldsymbol{\omega_m} - \hat{\boldsymbol{b}}_{\boldsymbol{\omega}})\boldsymbol{\Delta t}) & 0 & 0 & -\boldsymbol{A}((\boldsymbol{\omega_m} - \hat{\boldsymbol{b}}_{\boldsymbol{w}})\boldsymbol{\Delta t})\boldsymbol{\Delta t} & 0 & 0 \\ 0 & \boldsymbol{I} & \boldsymbol{I}\Delta t & 0 & 0 & 0 \\ -{}^{G}\hat{\boldsymbol{R}}_{I_i}(\boldsymbol{a_m} - \hat{\boldsymbol{b}_{\boldsymbol{a}}})^{\wedge}\Delta t & 0 & \boldsymbol{I} & 0 & -{}^{G}\hat{\boldsymbol{R}}_{I_i}\Delta t & \boldsymbol{I}\Delta t \\ 0 & 0 & 0 & \boldsymbol{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{I} \end{bmatrix} \tag{41}$$

Similarly, according to $F_w = \frac{\partial G}{\partial g_w}\frac{\partial g}{\partial w}\Big|_{w=0}$, we derive:

$$\frac{\partial G}{\partial g_w}\Big|_{w=0} \overset{(18)}{=\!=\!=} \begin{bmatrix} \boldsymbol{A(0)^{-1}A(g(0,0))^T} & 0 \\ 0 & \boldsymbol{I_{15\times 15}} \end{bmatrix} \tag{42}$$

$$= \begin{bmatrix} \boldsymbol{A}((\boldsymbol{\omega_m} - \hat{\boldsymbol{b}}_{\boldsymbol{w}})\boldsymbol{\Delta t}) & 0 \\ 0 & \boldsymbol{I_{15\times 15}} \end{bmatrix} \tag{43}$$

as well as:

$$\frac{\partial g}{\partial \omega}\Big|_{\omega=0} = \begin{bmatrix} -\boldsymbol{I}_{3\times 3}\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\hat{\boldsymbol{R}}_I\Delta t & 0 & 0 \\ 0 & 0 & \boldsymbol{I_{3\times 3}} & 0 \\ 0 & 0 & 0 & \boldsymbol{I_{3\times 3}} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{44}$$

We get:

$$\boldsymbol{F}_{\boldsymbol{\omega}} = \begin{pmatrix} -\boldsymbol{A}((\boldsymbol{\omega_m} - \hat{\boldsymbol{b}}_{\boldsymbol{\omega}})\Delta t)\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -\hat{\boldsymbol{R}}_{\boldsymbol{I}}\Delta t & 0 & 0 \\ 0 & 0 & \boldsymbol{I}\Delta t & 0 \\ 0 & 0 & 0 & \boldsymbol{I}\Delta t \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{45}$$

So far, we have finally deduced this big guy! With this recursive relation, we have already completed the derivation of the linearization of the motion equation of ESKF. We will deduce the ESKF observation equation later. It is worth mentioning here that some readers may ask, is this not the same as the results of other papers or the matrix in the code? In fact, some quantities in the formula can be simplified, For example, $-\boldsymbol{A}((\boldsymbol{\omega_m} - \hat{\boldsymbol{b}}_{\boldsymbol{\omega}})\Delta t)\Delta t$ can be simplified to $\boldsymbol{I}$ because Δt is small enough. For example, the simplification is consistent with the matrix in [5].

2.3 Backward Propagation and Motion Compensation

In this part, we mainly perform backpropagation, that is, motion compensation. As we mentioned above, since the sampling time of each point of the lidar is different, and the lidar is moving, it will cause motion distortion. The data we hope to get is the sampling of all points at the same time, which is at time t_k in this article. Therefore, we forward each point to time t_k according to the pose estimated by the IMU integration, as shown in Figure 2

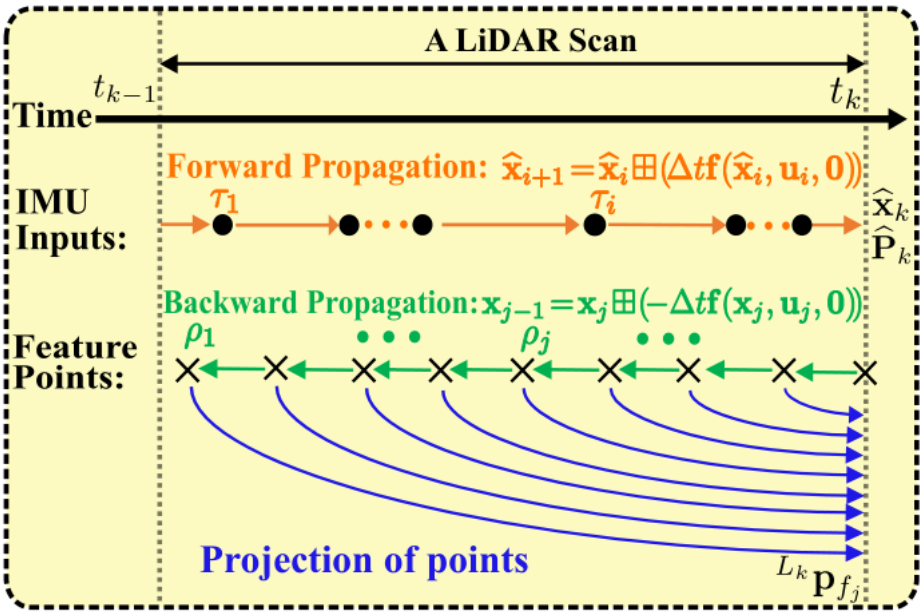

Figure 2: Forward and Backward Propagation

Starting at $\hat{\mathbf{x}}_k$(set to zero here since we are actually computing the pose relative to $\hat{\mathbf{x}}_k$, the relative pose is computed using $\check{\mathbf{x}}_{j-1} = \check{\mathbf{x}}_j \boxplus (-\Delta t \boldsymbol{f}(\check{\boldsymbol{x}}_{\boldsymbol{j}}, \boldsymbol{u}_{\boldsymbol{j}}, \mathbf{0}))$. Usually the frequency of points is greater than the IMU frequency, so for multiple feature points between two frame IMUs, we take the left IMU frame as the standard. Furthermore, we simplify backpropagation by considering that the last three lines (acceleration noise) are zero:

$$
\begin{aligned}
{}^{I_k}\check{\mathbf{p}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{p}}_{I_j} - {}^{I_k}\check{\mathbf{v}}_{I_j}\Delta t, \qquad s.f. {}^{I_k}\check{\mathbf{p}}_{I_m} = 0; && (46)\\
{}^{I_k}\check{\mathbf{v}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{v}}_{I_j} - {}^{I_k}\check{\mathbf{R}}_{I_j}(\mathbf{a}_{m_{i-1}} - \hat{\mathbf{b}}_{a_k})\Delta t - {}^{I_k}\hat{\mathbf{g}}_k\Delta t && (47)\\
s.f. \quad {}^{I_k}\hat{\mathbf{v}}_{I_m} &= {}^{G}\hat{\mathbf{R}}_{I_k}^T {}^{G}\hat{\mathbf{v}}_{I_k} \qquad {}^{I_k}\hat{\mathbf{g}}_k = {}^{G}\hat{\mathbf{R}}_{I_k}^T\hat{\mathbf{g}}_k; && (48)\\
{}^{I_k}\check{\mathbf{R}}_{I_{j-1}} &= {}^{I_k}\check{\mathbf{R}}_{I_j} Exp((\hat{\mathbf{b}}_{\omega_k} - \boldsymbol{\omega}_{m_{i-1}})\Delta t), \quad s.f, {}^{I_k}\mathbf{R}_{I_m} = \mathbf{I}. && (49)
\end{aligned}
$$

Where s.f. meanse "starting from". Define the relative pose of the points we get through backpropagation ${}^{I_k}\check{\boldsymbol{T}}_{I_j} = \left({}^{I_k}\check{\boldsymbol{R}}_{I_j}, {}^{I_k}\check{\mathbf{p}}_{I_j}\right)$, we project the point ${}^{L_j}\boldsymbol{p}_{f_j}$ to the scan end time t_k:

$$
{}^{L_k}\mathbf{p}_{f_j} = {}^{\boldsymbol{I}}\boldsymbol{T}_{\boldsymbol{L}}^{-1}{}^{\boldsymbol{I_k}}\check{\boldsymbol{T}}_{\boldsymbol{I_j}}{}^{\boldsymbol{I}}\boldsymbol{T}_{\boldsymbol{L}}{}^{\boldsymbol{L_j}}\boldsymbol{p}_{\boldsymbol{f_j}} \tag{50}
$$

${}^{I}T_L$ is extrinsic LiDAR IMU transformation. The above formula means that the feature point of L coordinate system is first transferred to I coordinate system, then multiplied by the compensation

matrix of backpropagation, then transferred back to L coordinate system. After all the projection points are turned like this, you can start calculating the residuals.

2.4 Residual computation

Let the current iteration of iterative Kalman Filter be κ, we can transform the compensated feature points into global coordinate systems:

$$^{G}\widehat{\mathbf{p}}_{f_j}^{\kappa} = {}^{G}\widehat{\mathbf{T}}_{I_k}^{\kappa}\,{}^{I}\mathbf{T}_{L}\,{}^{L_k}\mathbf{p}_{f_j}; \qquad j = 1, \cdots, m \tag{51}$$

Where $^{G}\widehat{\mathbf{T}}_{I_k}^{\kappa}$ is the pose transformation to be found. Residuals are calculated in the same way as in LOAM, using point-to-plane or point-to-edge distance as residuals. Defining $\mathbf{u}_j$ as a normal vector for a plane or edge and $^{G}\mathbf{q}_j$, is a point on a plane or edge. the residual $\mathbf{z}_j^{\kappa}$ is expressed as:

$$\mathbf{z}_j^{\kappa} = \boldsymbol{G}_j \left({}^{G}\widehat{\mathbf{p}}_{f_j}^{\kappa} - {}^{G}\mathbf{q}_j \right) \tag{52}$$

where $\boldsymbol{G}_j = \mathbf{u}_j^T$ if p is a plane point, $\boldsymbol{G}_j = \mathbf{u}_j^{\wedge}$ if p is an edge point. The meaning of the formula indicates the distance between point and planes or edges Like LOAM, it uses KD-tree for near-neighbor search.

2.5 Iterated State update

This section begins with an iterative update. We have already linearized the equations of motion for ESKF, and now we need to linearize the equations of observation. The noise of the observation equation comes from the error of LiDAR ranging. For a point $^{L_j}\boldsymbol{p}_{f_j}^{gt}$, we define the noise amount as follows:

$$^{L_j}\boldsymbol{p}_{f_j}^{gt} = {}^{L_j}\boldsymbol{p}_{f_j} + {}^{L_j}\boldsymbol{n}_{f_j} \tag{53}$$

Theoretically, if we use the true value $\mathbf{x}_k$ of the state and true value $^{L_j}\boldsymbol{p}_{f_j}$ of the LiDAR point, the residual should be zero.

$$0 = \mathbf{h}_j(\mathbf{x}_k, {}^{L_j}\mathbf{n}_{f_j}) = \mathbf{G}_j \left({}^{G}\boldsymbol{T}_{I_k}\,{}^{I_k}\tilde{\boldsymbol{T}}_{I_j}\,{}^{I}\boldsymbol{T}_{L}({}^{L_j}\boldsymbol{p}_{f_j} - {}^{L_j}\boldsymbol{n}_{f_j}) - {}^{G}\boldsymbol{q}_j \right) \tag{54}$$

where $h(x, n)$ represents the function of the observation equation. We need to linearize it, expanding around $\widehat{\mathbf{x}}_k^{\kappa}$

$$\begin{aligned} \mathbf{0} = \boldsymbol{h}_j(\mathbf{x}_k, {}^{L_j}\mathbf{n}_{f_j}) &= \boldsymbol{h}_j(\widehat{\mathbf{x}}_k^{\kappa} \boxplus \tilde{\mathbf{x}}_k^{\kappa}, {}^{L_j}\boldsymbol{n}_{f_j}) && (55)\\ &\simeq \boldsymbol{h}_j(\widehat{\mathbf{x}}_k^{\kappa}, \mathbf{0}) + \boldsymbol{H}_j^{\kappa}\tilde{\mathbf{x}}_k^{\kappa} + \mathbf{v}_j && (56)\\ &= \mathbf{z}_j^{\kappa} + \boldsymbol{H}_j^{\kappa}\tilde{\mathbf{x}}_k^{\kappa} + \mathbf{v}_j && (57) \end{aligned}$$

Where $\boldsymbol{H}_j^{\kappa}$ is the jacobian matrix of $h(x, n)$ about $\tilde{\mathbf{x}}_k^{\kappa}$. It can be calculated using chain rule:

$$\boldsymbol{H} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{p}} \frac{\partial \boldsymbol{p}}{\partial \tilde{\boldsymbol{x}}} \tag{58}$$

Where $\boldsymbol{p}$ presents the feature point transformed to $\{G\}$ system. the feature points set in coordinate system $\{L\}$ is ^{L}p, which has the following conversion relationships:

$$\boldsymbol{p} = {}^{G}\widehat{\boldsymbol{R}}_I({}^{I}\boldsymbol{R}_L\,{}^{L}\boldsymbol{p} + {}^{I}\boldsymbol{T}_L) + {}^{G}\widehat{\boldsymbol{T}}_I \tag{59}$$

Take the plane as an example, and if the unit normal vector of the plane formed by the nearest point of $\boldsymbol{p}$ is $\vec{n}$, the residual is defined as follows:

$$\boldsymbol{h} = (\boldsymbol{p} - \boldsymbol{p_a}) \cdot \vec{\boldsymbol{n}} \tag{60}$$

Where $\boldsymbol{p_a}$ is the point on that plane

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{p}} = \vec{\boldsymbol{n}} \tag{61}$$

And then we gonna solve $\frac{\partial \boldsymbol{p}}{\partial \delta \tilde{\mathbf{x}}}$. In fact, the matrix only relates to ${}^{G}\widehat{\boldsymbol{R}}_I$ and ${}^{G}\widehat{\boldsymbol{T}}_I$, meaning that the matrix only needs to find the first six columns and other rows are zero. That is:

$$\frac{\partial \boldsymbol{p}}{\partial \tilde{\mathbf{x}}} = \left[-{}^{G}\widehat{\boldsymbol{R}}_I ({}^{I}\boldsymbol{R}_L {}^{L}\boldsymbol{p} + {}^{I}\boldsymbol{T}_L)^{\wedge}, \boldsymbol{I}_{3\times 3}, 0, 0, 0, 0 \right] \tag{62}$$

By multiplying the above two equations:

$$H = \left[-\vec{\boldsymbol{n}}^{G}\widehat{\boldsymbol{R}}_I ({}^{I}\boldsymbol{R}_L {}^{L}\boldsymbol{p} + {}^{I}\boldsymbol{T}_L)^{\wedge}, 0, 0, 0, 0 \right] \tag{63}$$

So far, we have completed the derivation of the observation eqn. But it is not yet time to launch a general offensive. what we now get is an expression of EKF, and the IEKF is an improvment of EKF with the direct idea of iterating multiple times over $x_k \to x_{k+1}$ to eliminate non-linear effects. Interestingly, IEKF can prove equivalent to Gauss-Newtom method, so IEKF guarantee global convergence. In IEKF, the iterative process does not update the covariance matrix P. The following formula is used to update the covariance matrix:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k \boxminus \hat{\mathbf{x}}_k = (\hat{\mathbf{x}}_k^{\kappa} \boxplus \tilde{\mathbf{x}}_k^{\kappa}) \boxminus \hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{\kappa} \boxminus \hat{\mathbf{x}}_k + \boldsymbol{J}^{\kappa} \tilde{\mathbf{x}}_k^{\kappa} \tag{64}$$

Where

$$\boldsymbol{J}^{\kappa} = \begin{bmatrix} \boldsymbol{A}\left({}^{G}\widehat{\boldsymbol{R}}_{I_k}^{\kappa} \boxminus {}^{G}\widehat{\boldsymbol{R}}_{I_k} \right)^{-T} & \mathbf{0}_{3\times 15} \\ \mathbf{0}_{15\times 3} & \boldsymbol{I}_{15\times 15} \end{bmatrix} \tag{65}$$

For Jacobian matrices, the above formula is also derived from equation(18). Because $\tilde{\mathbf{x}}$ changes afte each iteration, its covariance matrix is theoretically not the origianl $\boldsymbol{P}_k$, so it can be updated throughout the iteration.So far, We have completed the pre-tasks of ESKF. Maximum a posteriori estimation:

$$\min_{\tilde{\boldsymbol{x}}_k^{\kappa}} \left(\|\mathbf{x}_k \boxminus \hat{\mathbf{x}}_k\|^2_{\hat{\boldsymbol{P}}_k^{-1}} + \sum_{j=1}^{m} \|\mathbf{z}_j^{\kappa} + \boldsymbol{H}_j^{\kappa} \tilde{\mathbf{x}}_k^{\kappa}\|^2_{\mathbf{R}_j^{-1}} \right) \tag{66}$$

Here $\boldsymbol{H} = \left[\boldsymbol{H}_1^{\kappa T}, \cdots, \boldsymbol{H}_m^{\kappa T} \right]^T$, $\boldsymbol{R} = diag(\boldsymbol{R_1}, \cdots, \boldsymbol{R}_m)$, $\boldsymbol{P} = (\boldsymbol{J}^{\kappa})^{-1} \hat{\boldsymbol{P}}_k (\boldsymbol{J}^{\kappa})^{-T}$ as well as $\mathbf{z}_k^{\kappa} = \left[\mathbf{z}_1^{\kappa T}, \cdots, \mathbf{z}_m^{\kappa} \right]^T$, Then the Kalman gain:

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{H}^{\boldsymbol{T}} (\boldsymbol{H}\boldsymbol{P}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-1} \tag{67}$$

Iteration formula:

$$\hat{\mathbf{x}}_k^{\kappa+1} = \hat{\mathbf{x}}_k^{\kappa} \boxplus \left(-\boldsymbol{K}\boldsymbol{z}_k^{\kappa} - (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}(\boldsymbol{J}^{\kappa})^{-1} (\hat{\mathbf{x}}_k^{\kappa} \boxminus \hat{\mathbf{x}}_k \right) \tag{68}$$

The total error of equation (62) can be expressed as the least squres of the following

$$E = \min_{\tilde{x}^\kappa} \frac{1}{2} \begin{pmatrix} z^\kappa + H\tilde{x}^\kappa \\ X + J^\kappa \tilde{x}^\kappa \end{pmatrix}^T \begin{pmatrix} R & \\ & \hat{P} \end{pmatrix}^{-1} \begin{pmatrix} z^\kappa + H\tilde{x}^\kappa \\ X + J^\kappa \tilde{x}^\kappa \end{pmatrix} = \min_{\tilde{x}^\kappa} \frac{1}{2} M^T M \tag{69}$$

where

$$M = S \begin{pmatrix} z^\kappa + H\tilde{x}^\kappa \\ X + J^\kappa \tilde{x}^\kappa \end{pmatrix}, \qquad S^T S = \begin{pmatrix} R & \\ & \hat{P} \end{pmatrix}^{-1} \tag{70}$$

Jacobian of M about $\tilde{x}^\kappa$ is:

$$\mathcal{J} = S \begin{pmatrix} H \\ J^\kappa \end{pmatrix} \tag{71}$$

According to Gauss-Newton method, the increment for each iteration is:

$$\Delta\tilde{x}^\kappa = (\mathcal{J}^T \mathcal{J})^{-1} \mathcal{J}^T M \tag{72}$$

It is deduced separately, among them:

$$\begin{aligned} (\mathcal{J}^T \mathcal{J})^{-1} = & \left[\begin{pmatrix} H^T & J^{\kappa T} \end{pmatrix} \begin{pmatrix} R^{-1} & \\ & \hat{P}^{-1} \end{pmatrix} \begin{pmatrix} H \\ J^\kappa \end{pmatrix} \right]^{-1} && (73) \\ & = (J^{\kappa T} \hat{P}^{-1} J^\kappa + H^T R^{-1} H)^{-1} && (74) \\ & \overset{SMW}{=} P - PH^T (R + HPH^T)^{-1} HP && (75) \\ & = (I - KH)P && (76) \\ & = KRH^{-T} && (77) \end{aligned}$$

Here $\boldsymbol{P = J^{\kappa-1} \hat{P} J^{\kappa-T}, K = PH^T (HPH^T + R)^{-1}}$. The third row utilizes the SMW formula:

$$(A^{-1} + BD^{-1}C)^{-1} = A - AB(D + CAB)^{-1}CA \tag{78}$$

The last line is derived as follows:

$$\begin{aligned} & K = PH^T (HPH^T + R)^{-1} && (79) \\ & K(HPH^T + R) = PH^T && (80) \\ & KR - (I - KH)PH^T = 0 && (81) \\ & (I - KH)P = KRH^{-T} && (82) \end{aligned}$$

Back to formual (68), the latter part is:

$$\begin{aligned} -\mathcal{J}^T M = & -\begin{pmatrix} H^T & J^{\kappa T} \end{pmatrix} \begin{pmatrix} R^{-1} & \\ & \hat{P}^{-1} \end{pmatrix} \begin{pmatrix} z^\kappa + H\tilde{x}^\kappa \\ X + J^\kappa \tilde{x}^\kappa \end{pmatrix} && (83) \\ & = -H^T R^{-1} (z^\kappa + H\tilde{x}^\kappa) - J^{\kappa T} \hat{P}^{-1} (X + J^\kappa \tilde{x}^\kappa) && (84) \end{aligned}$$

Multiplying formula (73) and formula (78), we get:

$$\begin{aligned} \Delta\tilde{x}^\kappa &= KRH^{-T}(-H^T R^{-1}(z^\kappa + H\tilde{x}^\kappa)) - (I - KH)P(J^{\kappa T} \hat{P}^{-1}(X + J^\kappa \tilde{x}^\kappa)) && (85) \\ &= -Kz^\kappa - KH\tilde{x}^\kappa - (I - KH)J^{\kappa-1} \hat{P} J^{\kappa-T} (J^{\kappa T} \hat{P}^{-1}(X + J^\kappa \tilde{x}^\kappa)) && (86) \\ &= -Kz^\kappa - KH\tilde{x}^\kappa - (I - KH)J^{\kappa-1}(X + J^\kappa \tilde{x}^\kappa) && (87) \\ &= -Kz^\kappa - \tilde{x}^\kappa - (I - KH)J^{\kappa-1} X && (88) \end{aligned}$$

Therefore,

$$\tilde{x}^{\kappa+1} = \tilde{x}^{\kappa} + \Delta\tilde{x}^{\kappa} = -Kz^{\kappa} - (I - KH)J^{\kappa-1}(\hat{\mathbf{x}}^{\kappa} \boxminus \hat{\mathbf{x}}) \tag{89}$$

The derivation is complete! It can be seen that the ESKF and Gauss-Newton methods are equivalent. Finally, an engineering problem is mentioned in the paper. When calculating the Kalman gain, the dimension of the matrix $\boldsymbol{HPH^T + R}$ is very big($m \times m$, because $H_{m\times 18}, P_{18\times 18}, R_{m\times m}$), The number of feature points m may be thousands, and the inversion of such a high-dimensional matrix is very time-consuming. Therefore, a new Kalman gain calculation method is given in this paper, which is equivalent to the original one:

$$\boldsymbol{K = (H^T R^{-1} H + P^{-1})^{-1} H^T R^{-1}} \tag{90}$$

The proof is easy to understand, see the appendix [9]. where $\boldsymbol{H^T R^{-1} H}$ is 18×18, which avoids the inversion of high-dimensional matrices.

2.6 The Algorithm

Algorithm 1: State Estimation

Input: Last optimal estimation $\bar{\mathbf{x}}_{k-1}$ and $\bar{\boldsymbol{P}}_{k-1}$
IMU inputs $(\mathbf{a}_m, \boldsymbol{\omega}_m)$ in current scan
LiDAR feature points ${}^{L_j}\mathbf{p}_{f_j}$ in current scan.

Forward propagation to obtain state prediction $\hat{\mathbf{x}}_k$ and covariance prediction $\hat{\boldsymbol{P}}_k$
Backward propagation to obtain ${}^{L_k}\mathbf{p}_{f_j}$ via (46);
$\kappa = -1, \widehat{\mathbf{x}}_k^{\kappa=0} = \widehat{\mathbf{x}}_k$;
while $\|\widehat{\mathbf{x}}_k^{\kappa+1} \boxminus \widehat{\mathbf{x}}_k^{\kappa} < \epsilon\|$ **do**
    $\kappa = \kappa + 1$;
    Compute $\boldsymbol{J}^{\kappa}$ via (61) and $\boldsymbol{P} = (\boldsymbol{J}^{\kappa})^{-1}\widehat{\boldsymbol{P}}_k(\boldsymbol{J}^{\kappa})^{-T}$;
    Compute residual $\mathbf{z}_j^{\kappa}$ (48) and Jacobian $\boldsymbol{H}_j^{\kappa}$(59);
    Compute the state update $\widehat{\mathbf{x}}_k^{\kappa+1}$ with the Kalman gain $\boldsymbol{K}$;
$\bar{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^{\kappa+1}; \bar{\boldsymbol{P}}_k = \boldsymbol{(I - KH)P}$.

Output: Current optimal estimation $\bar{\mathbf{x}}_k$ and $\bar{\boldsymbol{P}}_k$

2.7 Map Update

The update of the map is to project all the feature points to the global coordinate system according to the estimated optimal state variable $\bar{\mathbf{x}}_k$ (written as $SE(3) : {}^{G}\bar{\boldsymbol{T}}_{I_k} = ({}^{G}\bar{\boldsymbol{R}}_{I_k}, {}^{G}\bar{\boldsymbol{p}}_{I_k})$):

$${}^{G}\bar{\mathbf{p}}_{f_j} = {}^{G}\bar{\boldsymbol{T}}_{I_k}{}^{I}\boldsymbol{T}_{L}{}^{L_k}\boldsymbol{p}_{f_j}; j = 1, \cdots, m \tag{91}$$

2.8 Initialization

Initialization: Hold for 2s at the beginning, use the collected data to initialize the gravity vector, IMU bias, noise, etc.

3 Experiment Results

3.1 Computational Complexity Experiments

Table II Comparing the calculation efficiency of two kinds of Kalman gains, it is obvious that the new formula has greatly improved

3.2 UAV Flight Experiments

- drone experiment
- 70°FOV Livox Livox Avia LiDAR, DJI Manifold 2-C 1.8 GHz Intel i7-8550U CPU
- Maximum 50HZ odometer output + mapping
- Drift less than 0.3% (0.08m on 32m trace)

3.3 Outoddr Experiments

- Outdoor Handheld, Main Building, University of Hong Kong
- Drift less than 0.05% (0.07m on a 140m track), 10HZ, average processing time 25ms, average effective feature points 1497
- Compared on the LINS data set, the average computing time of FAST-LIO is 7.3ms, and the average of LINS is 34.5ms (the theoretical difference of personal feeling is mainly in the formula for calculating the gain K)