Vehicle State :-

$$x_k = \begin{bmatrix} p_k \\ v_k \\ q_k \end{bmatrix} \in R^{10\times 1}$$

Every thing is calculated wrt. Navigation frame (because state is measured w.r.t Navigation frame)

Σ_k → Uncertainity of the state Covariance Matrix ↓ 9×9 [variance for 3 entries across 3 dimensions]

$p_k \in R^3$ [Across 3 dimensions] [Position]

$v_k \in R^3$ [Across 3 dimension] [Velocity]

$Q_k \in R^4$ [Unit Quaternion] [Orientation]

Prediction Step of Error State Kalman Filter :-

We use IMU readings as input to the motion model to predict current state

IMU model output acts as input to motion model.

IMU → output → Motion Model → Current state Prediction

IMU output $u_k = \begin{bmatrix} f_k \\ \omega_k \end{bmatrix} \in R^{6\times 1}$

$f_k \in R^3$ [Across 3 dimensions] [Forces Acting]

$\omega_k \in R^3$ [Across 3 dimensions] [Rotational Rates]

Motion Model :- It is non-linear Model because of C_{ns} [rotation]

Position : $P_k = P_{k-1} + v_{k-1} \times \Delta t + \frac{\Delta t^2}{2}(C_{ns} f_{k-1} - g)$

C_{ns} → Rotation Matrix from Sensor frame to Navigation frame

$C_{ns} f_{k-1}$ → Tells How forces captured in Sensor frame look in Navigation frame.

$C_{ns} f_{k-1} - g$ → f_{k-1} Specific force measured by IHU. IHU measures forces in free fall - hence as the car is on the road as per newton's 3rd law an equal amount of gravity acts on the car upwards, which is excluded by IHU - it should be included in Navigation frame.

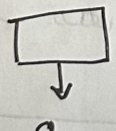
$P_k = P_{k-1} + \text{displacement}$

$s = ut + \frac{1}{2}at^2$

$t = \Delta t$

$u = v_{k-1}$

$a = C_{ns} f_{k-1} - g$



g

only force acting on the body is gravity.

Vector form

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}_k = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}_{k-1} + \Delta t \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{k-1} + \frac{\Delta t^2}{2}\left(C_{ns} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right)$$

3×1 $\quad$ 3×1 $\quad$ 3×1 $\quad$ 3×3 $\quad$ 3×1 $\quad$ 3×1

Velocities :-

$$v_k = v_{k-1} + \Delta t \left(C_{ns} f_{k-1} - g \right)$$

$$\boxed{v = u + at}$$

Vector form

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_k = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{k-1} + \Delta t \left[C_{ns} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}_{k-1} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right]$$

Orientation :-

$$q_k = \Omega \left(q(w_{k-1} \Delta t) \right) q_{k-1}$$

$q(\theta)$

Quaternion from Axis-Angle :-

$$q(\theta) = \begin{bmatrix} \sin \frac{\theta}{2} \\ \frac{\theta}{|\theta|} \cos \frac{\theta}{2} \end{bmatrix}$$

↳ Based on Angle axis representation.

where θ → rotation vector.

Quaternion Multiplication Matrix :- (Ω)

↳ function of 1-Quaternion.

↳ Takes 1-Quaternions as input and generate 4×4 quaternion Multiplication matrix.

$$q \in R^4 = \begin{bmatrix} q_s \\ q_{\vec{v}} \end{bmatrix} \in R^{4\times 1}$$

$q_s \rightarrow$ Scalar part $\in R^1$

$q_{\vec{v}} \rightarrow$ Vector part $\in R^3$

$$q = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \underbrace{a}_{\downarrow q_s} + \underbrace{bi + cj + dk}_{\downarrow q_{\vec{v}}}$$

$$\Omega(q) = \Omega\left(\begin{bmatrix} q_s \\ q_{\vec{v}} \end{bmatrix}\right)$$

$$= q_s I + \begin{bmatrix} 0 & -q_v^T \\ q_v & -\{q_v\}_x \end{bmatrix} \quad \text{where } q_v = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

$-\{q_v\}_x \rightarrow$ Skew Symmetric Matrix of q_v

$$\Rightarrow \begin{bmatrix} 0 & -d & c \\ d & 0 & -b \\ -c & b & 0 \end{bmatrix}$$

$$a \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & -b & -c & -d \\ b & 0 & -d & c \\ c & d & 0 & -b \\ d & -c & b & 0 \end{bmatrix}$$

$$\therefore \Omega(q) = \Omega\left(\begin{bmatrix} q_s \\ q_{\vec{v}} \end{bmatrix}\right) = \Omega\left(\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}\right)$$

$$= \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix}$$

Quaternion from axis-Angle :-

$$\theta = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \quad |\theta| = \sqrt{\theta_x^2 + \theta_y^2 + \theta_z^2}$$

Determine unit axis for rotation.

$$\hat{u} = \frac{\theta}{|\theta|} = \begin{bmatrix} \frac{\theta_x}{|\theta|} \\ \frac{\theta_y}{|\theta|} \\ \frac{\theta_z}{|\theta|} \end{bmatrix}$$

Calculate scalar & vector components of Quaternion;.

Scalar part $q_s = \cos\left(\frac{|\theta|}{2}\right)$

vector part $q_v = \hat{u} \sin\left(\frac{|\theta|}{2}\right) = \begin{bmatrix} \frac{\theta_x}{|\theta|} \sin\left(\frac{|\theta|}{2}\right) \\ \frac{\theta_y}{|\theta|} \sin\left(\frac{|\theta|}{2}\right) \\ \frac{\theta_z}{\theta} \sin\left(\frac{|\theta|}{2}\right) \end{bmatrix}$

If Quaternion Internal representation is $\begin{bmatrix} q_{\vec{v}} \\ q_s \end{bmatrix}$

then $q(\theta) = \begin{bmatrix} \sin\frac{\theta}{2} \\ \frac{\theta}{|\theta|}\cos\frac{\theta}{2} \end{bmatrix}$

$\therefore q_k = \Omega\big(\underbrace{q(\omega_{k-1}\Delta t)}_{\text{From angles to Quaternion}}\big)\, q_{k-1}$

Construct rotation Matrix using Quaternion

Rotate the Quaternion using rotation Matrix.

vector form

$$\begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix}_k = \Omega\left(q\left(\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}\Delta t\right)\right)\begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix}_{k-1}$$

4×1 $\quad 4\times4$ $\quad 4\times1$

Rotation Matrix from Quaternion

$$C_{ns} = \phi(q_{k-1})$$

ϕ is a function.

Linearized Motion Model

Error State

$$\delta x_k = \begin{bmatrix} \delta p_k \\ \delta v_k \\ \delta \theta_k \end{bmatrix} \in \mathbb{R}^9$$

$\delta\theta_k$: axis angles, not Quaternions, hence 3x1

Error Dynamics

$$\delta x_k = F_{k-1}\, \delta x_{k-1} + L_{k-1}\, n_{k-1}$$

IMU noise

$$n_{k-1} \sim \mathcal{N}(0, Q_k)$$

$$Q_k = \Delta t^2 \begin{bmatrix} \sigma_{acc}^2 & & & & & \\ & \sigma_{acc}^2 & & & & \\ & & \sigma_{acc}^2 & & & \\ & & & \sigma_{gyro}^2 & & \\ & & & & \sigma_{gyro}^2 & \\ & & & & & \sigma_{gyro}^2 \end{bmatrix}$$

6x6

$$\begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}$$

9x9 — Jacobians

$$F_{k-1} = \begin{bmatrix} I & I\Delta t & 0 \\ 0 & I & -[C_{ns} f_{k-1}]_\times \Delta t \\ 0 & 0 & I \end{bmatrix}$$

Skew Symmetric (3x3)

$$= \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & -[C_{ns} f_{k-1}]_\times \Delta t \\ 0_{3\times3} & 0_{3\times3} & I_{9\times9} \end{bmatrix}$$

for δp ← for δv for $\delta\theta$

Jacobian

$$L_{k-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow 9\times6$$

9x6

$$\delta x_k = F_{k-1}\,\delta x_{k-1} + L_{k-1}\,n_{k-1}$$

$9\times1 \quad 9\times9 \quad 9\times1 \quad 9\times6 \quad 6\times1$

$$\begin{bmatrix} \delta p_x \\ \delta p_y \\ \delta p_z \\ \delta v_x \\ \delta v_y \\ \delta v_z \\ \delta \theta_x \\ \delta \theta_y \\ \delta \theta_z \end{bmatrix}_{9\times1}$$

Measurement Model :-

$$y_k = h(x_k) + v_k \quad \text{Measurement Noise}$$

$$\sim (0,\ R_{GNSS/LiDAR})$$

3×9

$$H x_k \qquad \sigma^2_{GNSS/Lidar}\ I_{3\times3}$$

$$h(x_k) = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} x_k$$

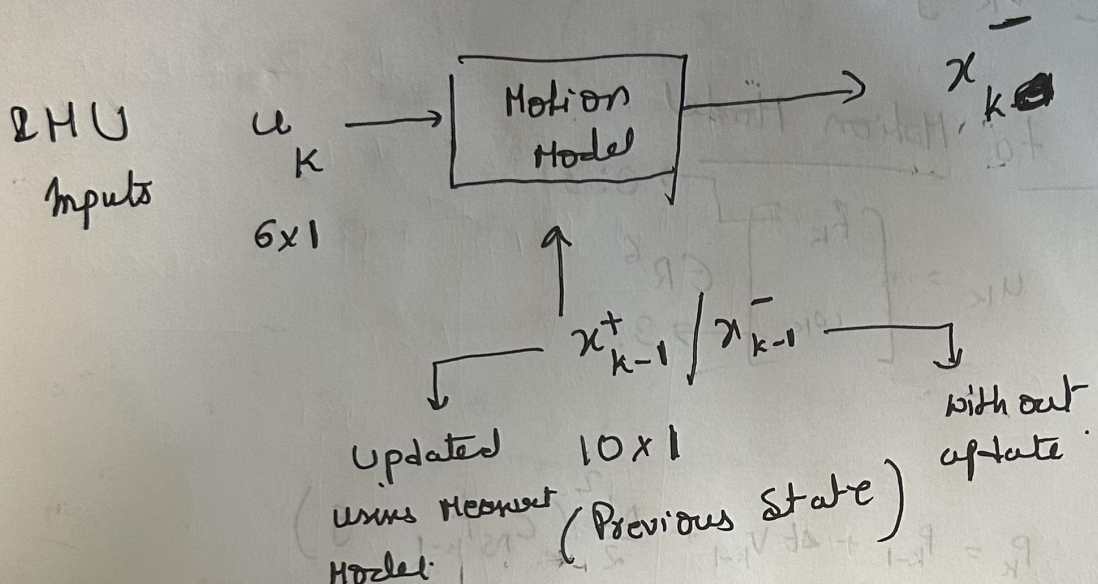
$3\times9 \qquad 9\times1$

3×1

ES-EKF

LOOP:-

1) Update state with IMU inputs



2) Propogate Uncertainity

$$\bar{\Sigma}_k = F_{k-1} \Sigma^{+/-}_{k-1} F^T_{k-1} + L_{k-1} Q_{k-1} L^T_{k-1}$$

$\bar{\Sigma}_k$: 9x9

F_{k-1}: 9x9, $\Sigma^{+/-}_{k-1}$: 9x9, F^T_{k-1}: 9x9 → 9x9

L_{k-1}: 9x6, Q_{k-1}: 6x6, L^T_{k-1}: 6x9 → 9x9

2b GNSS / LiDAR position (Data) Available :-

① Compute KALMAN gain :-

$$K_k = \Sigma_k^- H_k^T \left(H_k \Sigma_k^- H_k^T + R \right)^{-1}$$

Σ_k^- : 9×9, H_k^T : 9×3, H_k : 3×9, Σ_k^- : 9×9, H_k^T : 9×3, R : 3×3

9×9 9×3 | 3×9 9×9 9×3 | 3×3

9×3 | 3×3

9×3

② Compute Error State :-

$$\delta x_k = K_k \left(y_k - P_k^- \right)$$

δx_k : 9×1, K_k : 9×3, y_k : 3×1, P_k^- : 3×1

9×3 3×1 | 9×1

③ Correction of prediction State :-

$$\delta_k = \begin{bmatrix} \delta P_k \\ \delta V_k \\ \delta \Theta_k \end{bmatrix} \begin{matrix} \rightarrow 3\times1 \\ \rightarrow 3\times1 \\ \rightarrow 3\times1 \end{matrix}$$

$$P_k^+ = P_k^- + \delta P_k$$

$$V_k^+ = V_k^- + \delta V_k$$

$$q_k^+ = \Omega\left(q(\delta \Theta_k) \right) q_k^-$$

Measures how much correction in rotation.

④ Compute Corrected Covariance:-

$$\Sigma_k^+ = (I_{9\times9} - K_k H_k)\,\Sigma_k^-$$

K_k: 9×3, H_k: 3×9, $K_k H_k$: 9×9

Σ_k^+: 9×9, $(I - K_k H_k)$: 9×9, Σ_k^-: 9×9

Note:- "When will consider that GNSS / LiDAR to be available"

Here we have Asynchronous updates

If from GNSS, LIDAR we get information of time Stamp t at time Step $t+1$ [Because it takes some time for GNSS/LIDAR to process and propagate the measurements]

∴ trigger update/correction @ t

when

Data - time Stamp $[t-1]$ = gnss.time stamp $[gnss\_ptr]$

Data - time stamp $[t-1]$ = lidar.time Stamp $[lidar\_ptr]$