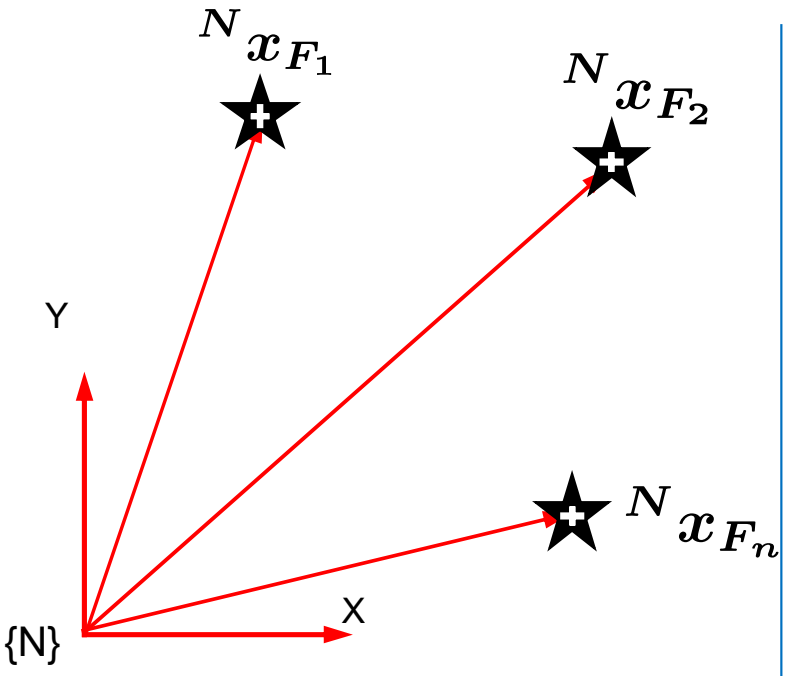
EKF Map Based Localization



Spatial Relationships

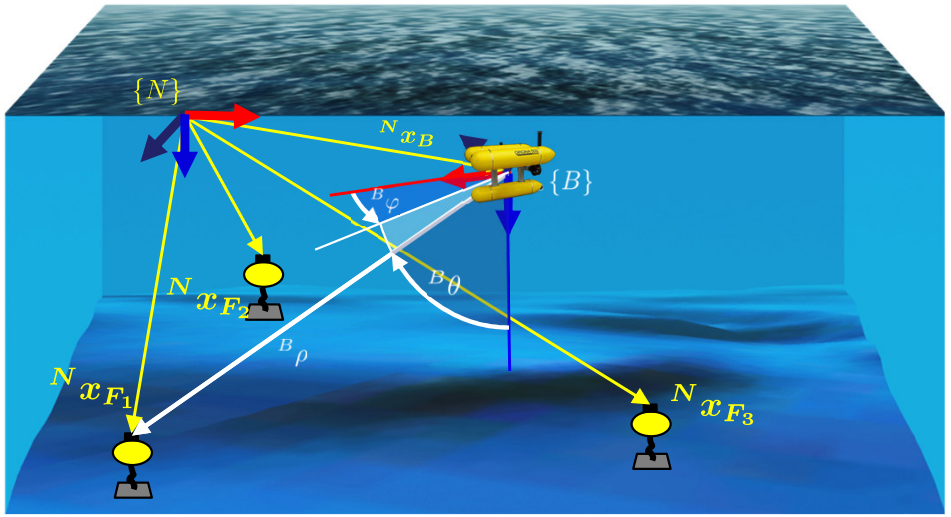
> A map is represented by:

$$\mathcal{M} = [{}^N \boldsymbol{x}_{\boldsymbol{F_1}}^{\boldsymbol{T}}, {}^N \boldsymbol{x}_{\boldsymbol{F_2}}^{\boldsymbol{T}}, \cdots, {}^N \boldsymbol{x}_{\boldsymbol{F_n}}^{\boldsymbol{T}}]^T$$

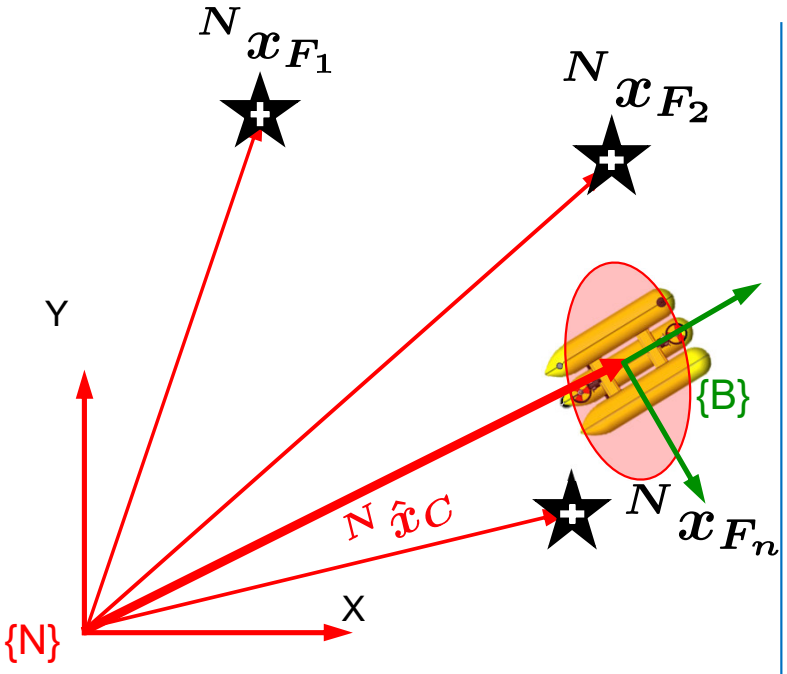
and it is known with total certainty.

> **Example:** Map of **Cartesian Features Observed in Spherical** coordinates (USBL Transponders)

$$\mathcal{M} = [{}^N \boldsymbol{x}_{\boldsymbol{F_1}}^{\boldsymbol{T}}, {}^N \boldsymbol{x}_{\boldsymbol{F_2}}^{\boldsymbol{T}}, {}^N \boldsymbol{x}_{\boldsymbol{F_3}}^{\boldsymbol{T}}]^T \; ; \; {}^N \boldsymbol{x}_{\boldsymbol{F_i}} = \begin{bmatrix} {}^N x_i \\ {}^N y_i \\ {}^N z_i \end{bmatrix} \in \mathbb{R}^3$$



EKF Map Based Localization



Spatial Relationships

> A map is represented by:

$$\mathcal{M} = [{}^N \boldsymbol{x}_{\boldsymbol{F_1}}^{\boldsymbol{T}}, {}^N \boldsymbol{x}_{\boldsymbol{F_2}}^{\boldsymbol{T}}, \cdots, {}^N \boldsymbol{x}_{\boldsymbol{F_n}}^{\boldsymbol{T}}]^T$$

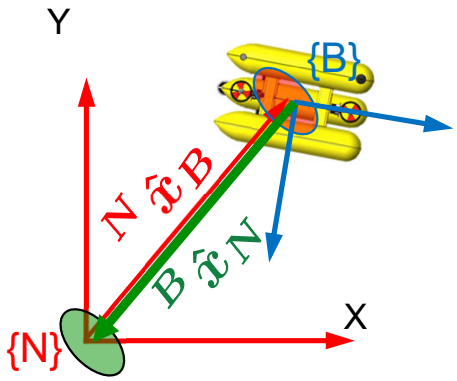
and it is known with total certainty: ${}^N \boldsymbol{x}_{\boldsymbol{F_i}} \in \mathbb{R}^n$

> The Robot pose has to be estimated

$${}^N \boldsymbol{x}_{\boldsymbol{R}} = {}^N \boldsymbol{x}_{\boldsymbol{B}} \sim \mathcal{N}({}^N \hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^N \boldsymbol{P}_{\boldsymbol{B}})$$

$${}^N \hat{\boldsymbol{x}}_{\boldsymbol{B}} = \begin{bmatrix} x_k \\ y_k \\ z_k \\ \psi_k \end{bmatrix} \quad {}^N \boldsymbol{P}_{\boldsymbol{B}} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & \sigma_{x\psi} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} & \sigma_{y\psi} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 & \sigma_{z\psi} \\ \sigma_{\psi x} & \sigma_{\psi y} & \sigma_{\psi z} & \sigma_\psi^2 \end{bmatrix}$$

EKF Map Based Localization



We need to define the **Pose Inversion:**

> Given a transformation

$$^{N}\boldsymbol{x}_{B} = [^{N}x_B \quad ^{N}y_B \quad ^{N}z_B \quad ^{N}\psi_B]^T$$

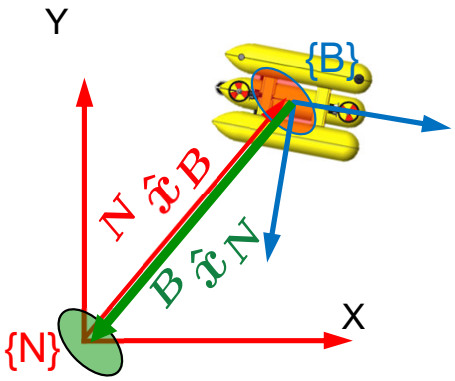
> The inverse transformation is given by:

$$^{B}\boldsymbol{x}_{N} = \ominus\, ^{N}\boldsymbol{x}_{B} = \begin{bmatrix} -\begin{bmatrix} \cos(^{N}\psi_B) & -\sin(^{N}\psi_B) & 0 \\ \sin(^{N}\psi_B) & \cos(^{N}\psi_B) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} ^{N}x_B \\ ^{N}y_B \\ ^{N}z_B \end{bmatrix} \\ -^{N}\psi_B \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} \cos(^{N}\psi_B) & \sin(^{N}\psi_B) & 0 \\ -\sin(^{N}\psi_B) & \cos(^{N}\psi_B) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} ^{N}x_B \\ ^{N}y_B \\ ^{N}z_B \end{bmatrix} \\ -^{N}\psi_B \end{bmatrix}$$

$$= \begin{bmatrix} -^{N}x_B \cos{}^{N}\psi_B - ^{N}y_B \sin{}^{N}\psi_B \\ {}^{N}x_B \sin{}^{N}\psi_B - ^{N}y_B \cos{}^{N}\psi_B \\ -^{N}z_B \\ -^{N}\psi_B \end{bmatrix}$$

EKF Map Based Localization



Pose Inversion

> If the pose is a GRVs

$$\ominus {}^{N}\boldsymbol{x}_{\boldsymbol{B}} \approx \mathcal{N}({}^{\boldsymbol{B}}\hat{\boldsymbol{x}}_{\boldsymbol{N}}, {}^{\boldsymbol{B}}\boldsymbol{P}_{\boldsymbol{N}})$$

$${}^{\boldsymbol{B}}\hat{\boldsymbol{x}}_{\boldsymbol{N}} = \ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}$$

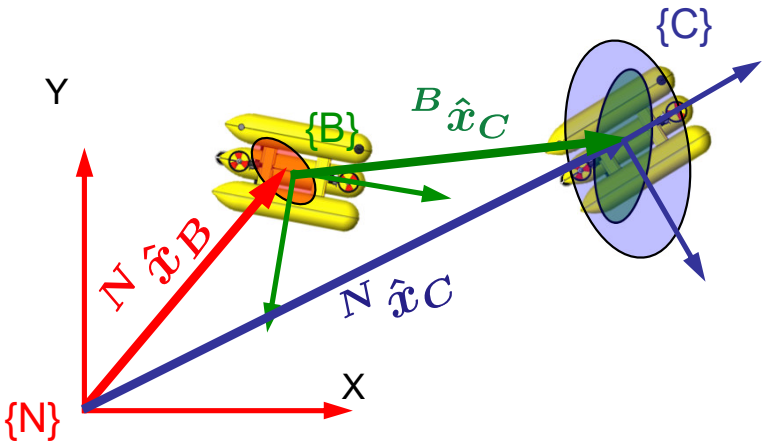
$${}^{\boldsymbol{B}}\boldsymbol{P}_{\boldsymbol{N}} = \boldsymbol{J}_{\ominus}\, {}^{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{B}} \boldsymbol{J}_{\ominus}^{T}$$

> The Jacobian is given by:

$$\boldsymbol{J}_{\ominus} = \left.\frac{\partial \ominus {}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B}}}{\partial {}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B}}}\right|_{{}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}} = \left.\frac{\partial \begin{bmatrix} -{}^{N}x_B \cos{}^{N}\psi_B - {}^{N}y_B \sin{}^{N}\psi_B \\ {}^{N}x_B \sin{}^{N}\psi_B - {}^{N}y_B \cos{}^{N}\psi_B \\ -{}^{N}z_B \\ -{}^{N}\psi_B \end{bmatrix}}{\partial [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]}\right|_{{}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}}$$

$$= \begin{bmatrix} -\cos{}^{N}\hat{\psi}_B & -\sin{}^{N}\hat{\psi}_B & 0 & {}^{N}\hat{x}_B \sin{}^{N}\hat{\psi}_B - {}^{N}\hat{y}_B \cos{}^{N}\hat{\psi}_B \\ \sin{}^{N}\hat{\psi}_B & -\cos{}^{N}\hat{\psi}_B & 0 & {}^{N}\hat{x}_B \cos{}^{N}\hat{\psi}_B \; {}^{N}\hat{y}_B \sin{}^{N}\hat{\psi}_B \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

EKF Map Based Localization



$$B \equiv B \ at \ k-1$$
$$C \equiv B \ at \ k$$

Pose Compounding

> Given 2 poses

$$^{N}\boldsymbol{x_B} = [^{N}x_B \quad ^{N}y_B \quad ^{N}z_B \quad ^{N}\psi_B]^T$$
$$^{B}\boldsymbol{x_C} = [^{B}x_C \quad ^{B}y_C \quad ^{B}z_C \quad ^{B}\psi_C]^T$$

> The compounded pose is given by:

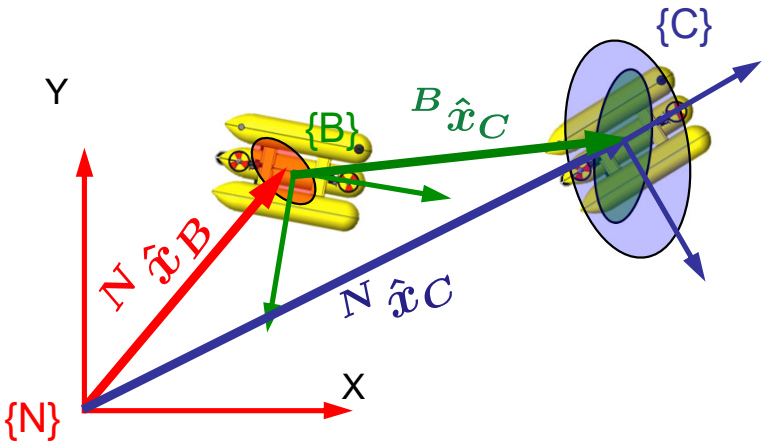
$$^{N}\boldsymbol{x_C} = ^{N}\boldsymbol{x_B} \oplus ^{B}\boldsymbol{x_C}$$

$$= \begin{bmatrix} ^{N}x_B \\ ^{N}y_B \\ ^{N}z_B \\ ^{N}\psi_B \end{bmatrix} \oplus \begin{bmatrix} ^{B}x_C \\ ^{B}y_C \\ ^{B}z_C \\ ^{B}\psi_C \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} ^{N}x_B \\ ^{N}y_B \\ ^{N}z_B \end{bmatrix} + \begin{bmatrix} \cos ^{N}\psi_B & -\sin ^{N}\psi_B & 0 \\ \sin ^{N}\psi_B & \cos ^{N}\psi_B & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ^{B}x_C \\ ^{B}y_C \\ ^{B}z_C \end{bmatrix} \\ ^{N}\psi_B + ^{B}\psi_C \end{bmatrix}$$

$$= \begin{bmatrix} ^{N}x_B + ^{B}x_C \cos ^{N}\psi_B - ^{B}y_C \sin ^{N}\psi_B \\ ^{N}y_B + ^{B}x_C \sin ^{N}\psi_B + ^{B}y_C \cos ^{N}\psi_B \\ ^{N}z_B + ^{B}z_C \\ ^{N}\psi_B + ^{B}\psi_C \end{bmatrix}$$

EKF Map Based Localization



$$B \equiv B \ at \ k-1$$
$$C \equiv B \ at \ k$$

Remind **Pose Compounding**

> Given 2 poses

$$^{N}\boldsymbol{x_B} = [^{N}x_B \quad ^{N}y_B \quad ^{N}z_B \quad ^{N}\psi_B]^T$$
$$^{B}\boldsymbol{x_C} = [^{B}x_C \quad ^{B}y_C \quad ^{B}z_C \quad ^{B}\psi_C]^T$$

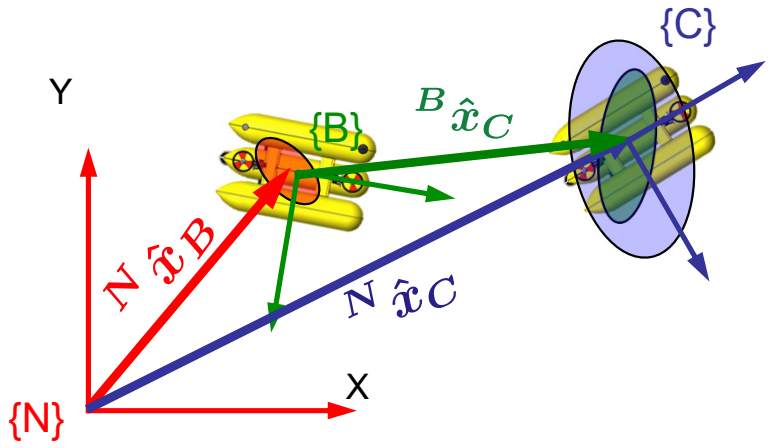
> When the poses are independent GRVs

$$^{N}\boldsymbol{x_B} \sim \mathcal{N}(^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, ^{N}\boldsymbol{P_B})$$
$$^{B}\boldsymbol{x_C} \sim \mathcal{N}(^{B}\hat{\boldsymbol{x}}_{\boldsymbol{C}}, ^{B}\boldsymbol{P_C})$$

> The linear approximation of the outcome is a GRVs:

$$^{N}\boldsymbol{x_B} \oplus ^{B}\boldsymbol{x_C} \approx \mathcal{N}(^{N}\hat{\boldsymbol{x}}_{\boldsymbol{C}}, ^{N}\boldsymbol{P_C})$$
$$^{N}\hat{\boldsymbol{x}}_{\boldsymbol{C}} = ^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \oplus ^{B}\hat{\boldsymbol{x}}_{\boldsymbol{C}}$$
$$^{N}\boldsymbol{P_C} = \boldsymbol{J_{1\oplus}} \, ^{N}\boldsymbol{P_B} \boldsymbol{J_{1\oplus}^T} + \boldsymbol{J_{2\oplus}} \, ^{B}\boldsymbol{P_C} \boldsymbol{J_{2\oplus}^T}$$

EKF Map Based Localization



$$B \equiv B \ at \ k-1$$
$$C \equiv B \ at \ k$$

$$^{N}\boldsymbol{x_C} = \begin{bmatrix} ^{N}x_B +^{B} x_C \cos{}^{N}\psi_B -^{B} y_C \sin{}^{N}\psi_B \\ ^{N}y_B +^{B} x_C \sin{}^{N}\psi_B +^{B} y_C \cos{}^{N}\psi_B \\ ^{N}z_B +^{B} z_C \\ ^{N}\psi_B +^{B} \psi_C \end{bmatrix}$$

> The Jacobians are given by:

$$\boldsymbol{J_{1\oplus}} = \left.\frac{\partial {}^{N}\boldsymbol{x_B} \oplus^{B} \boldsymbol{x_C}}{\partial {}^{N}\boldsymbol{x_B}}\right|_{{}^{N}\boldsymbol{x_B}={}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}},{}^{B}\boldsymbol{x_C}={}^{B}\hat{\boldsymbol{x}}_{\boldsymbol{C}}}$$

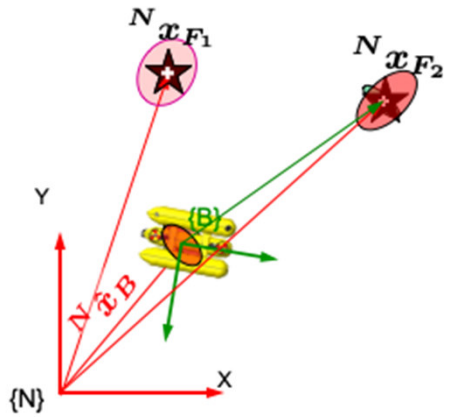
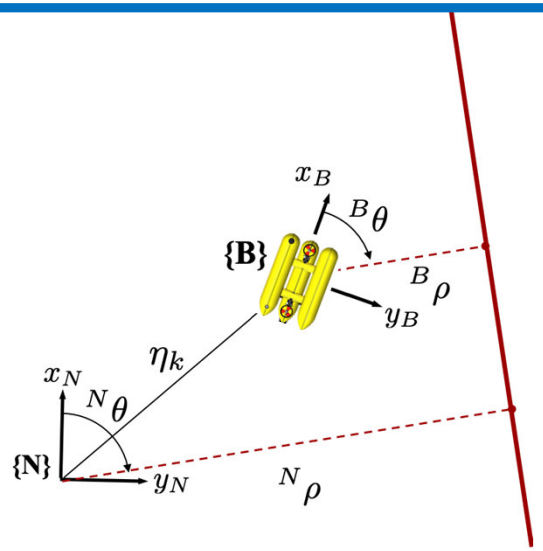
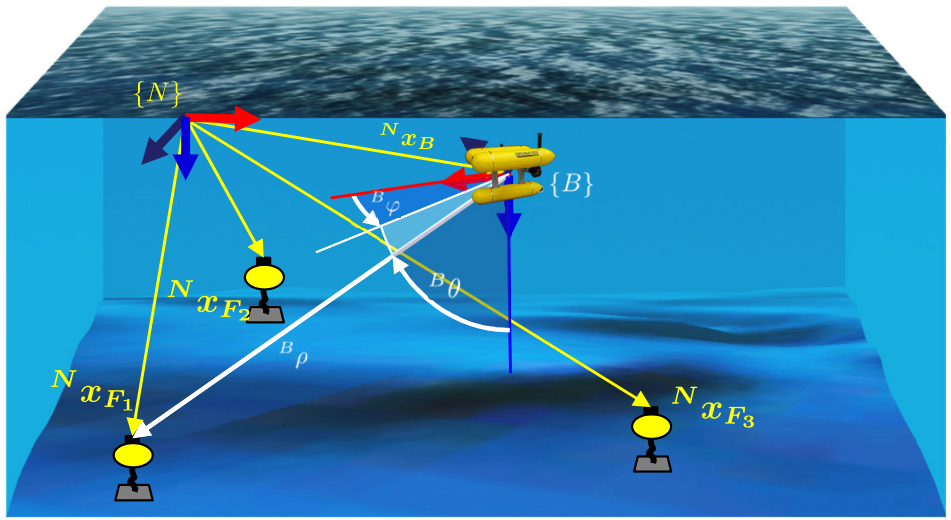
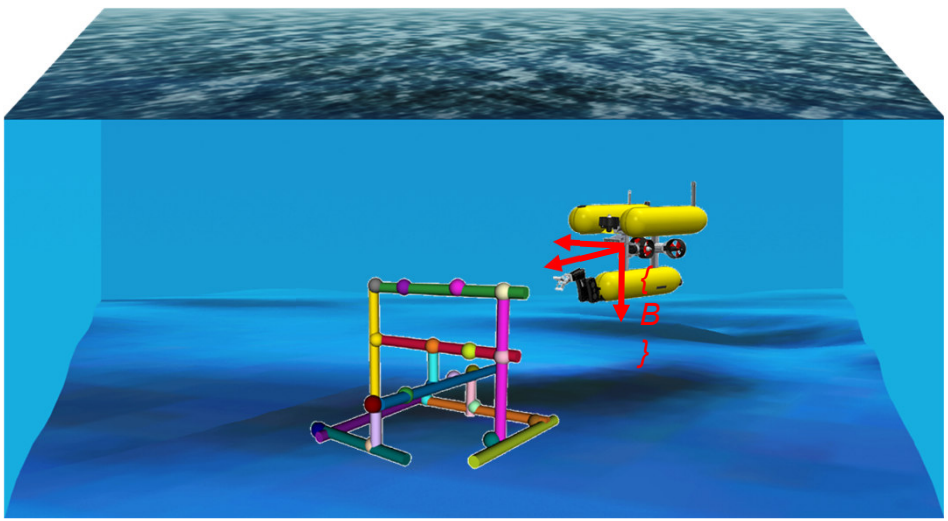
$$= \begin{bmatrix} 1 & 0 & 0 & -^{B}\hat{x}_C \cdot \sin{}^{N}\hat{\psi}_B -^{B}\hat{y}_C \cdot \cos{}^{N}\hat{\psi}_B \\ 0 & 1 & 0 & {}^{B}\hat{x}_C \cdot \cos{}^{N}\hat{\psi}_B -^{B}\hat{y}_C \cdot \sin{}^{N}\hat{\psi}_B \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{J_{2\oplus}} = \left.\frac{\partial {}^{N}\boldsymbol{x_B} \oplus^{B} \boldsymbol{x_C}}{\partial {}^{B}\boldsymbol{x_C}}\right|_{{}^{N}\boldsymbol{x_B}={}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}},{}^{B}\boldsymbol{x_C}={}^{B}\hat{\boldsymbol{x}}_{\boldsymbol{C}}}$$

$$= \begin{bmatrix} \cos{}^{N}\hat{\psi}_B & -\sin{}^{N}\hat{\psi}_B & 0 & 0 \\ \sin{}^{N}\hat{\psi}_B & \cos{}^{N}\hat{\psi}_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

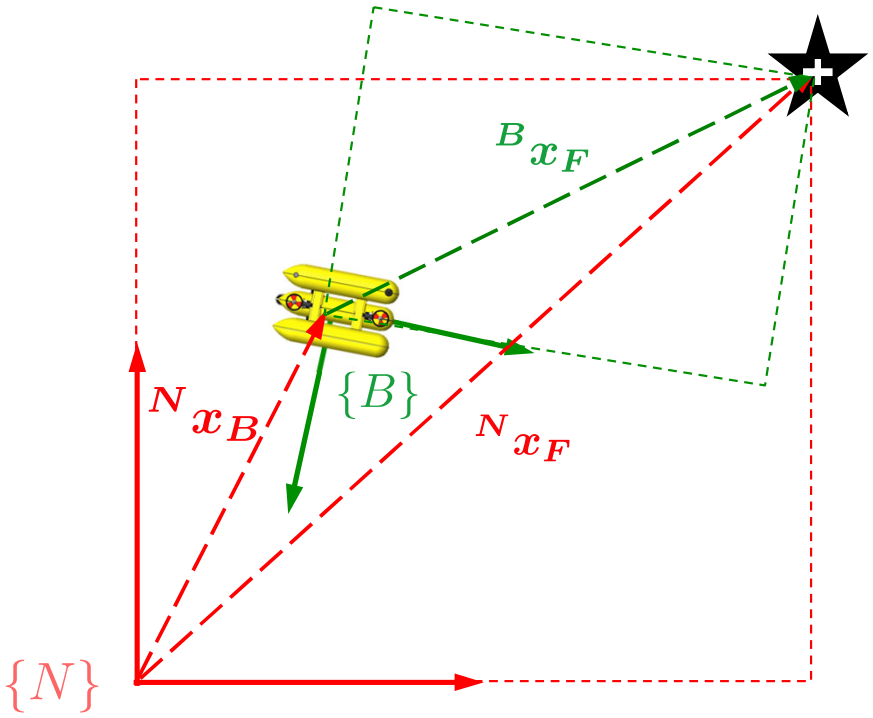
EKF Map Based Localization

Type of Features

	POINTS	LINES
2D		
3D		

EKF Map Based Localization

3D Point Feature Observed in Cartesian Coordinates

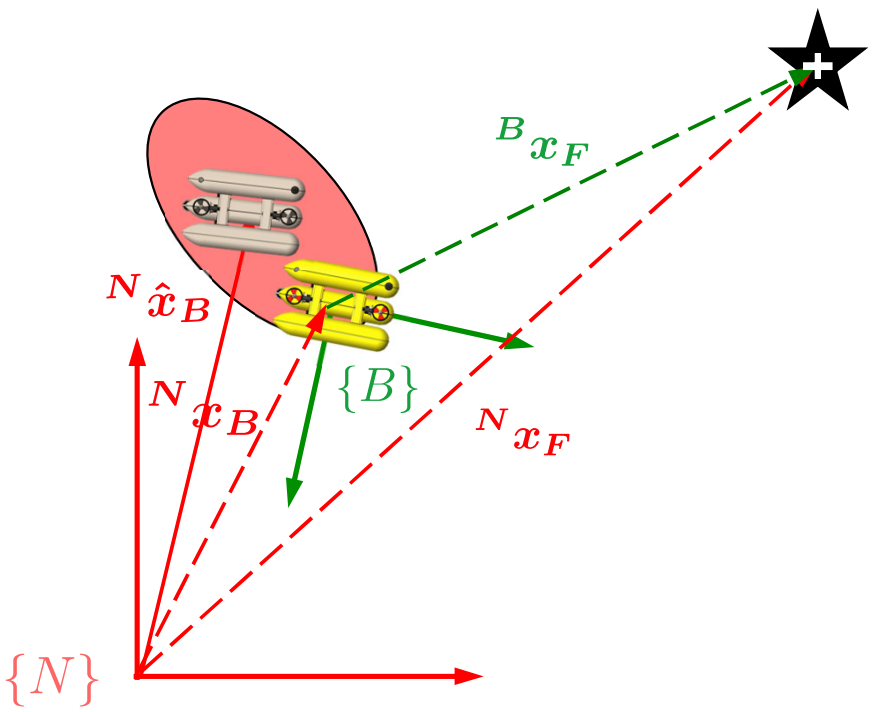


Ground Truth

> **Ground Truth:** The robot is at $^{N}\mathbf{x}_{B}$ and the feature is at $^{B}\mathbf{x}_{F}$.

EKF Map Based Localization

3D Point Feature Observed in Cartesian Coordinates



Ground Truth

Estimate

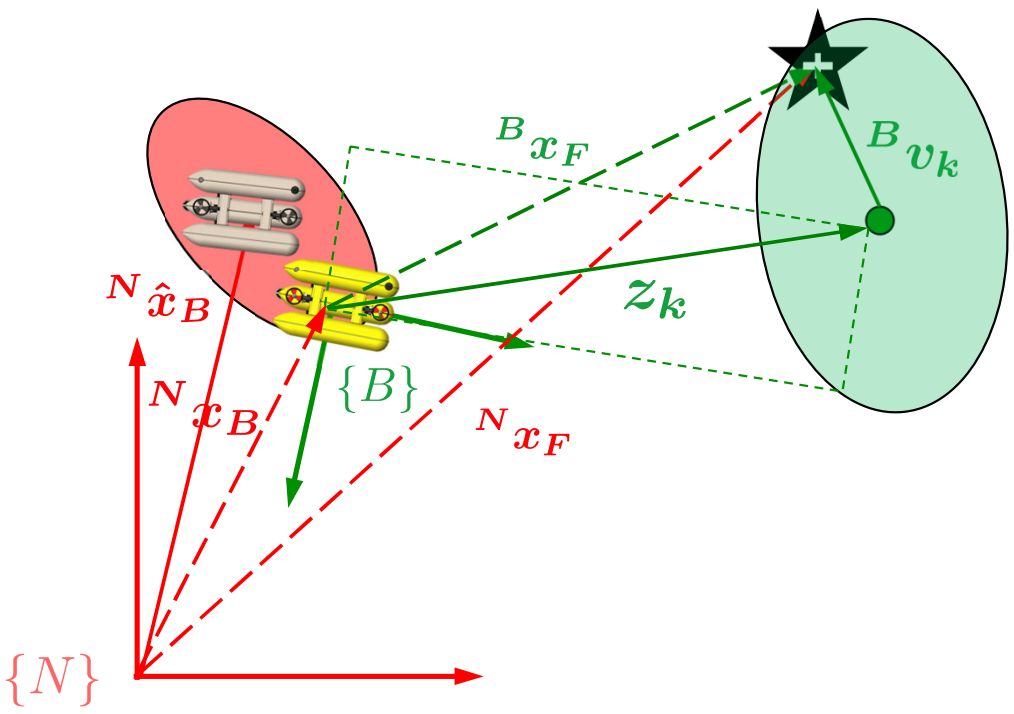
> **Ground Truth:** The robot is at $^{N}\mathbf{x}_{B}$ and the feature is at $^{B}\mathbf{x}_{F}$.

> The **robot pose estimate** is:

$$[^{N}\hat{\boldsymbol{x}}_{B}, ^{N}\boldsymbol{P}_{B}]$$

EKF Map Based Localization

3D Point Feature Observed in Cartesian Coordinates



Ground Truth

Estimate

> **Ground Truth:** The robot is at $^{N}\mathbf{x}_{B}$ and the feature is at $^{B}\mathbf{x}_{F}$.

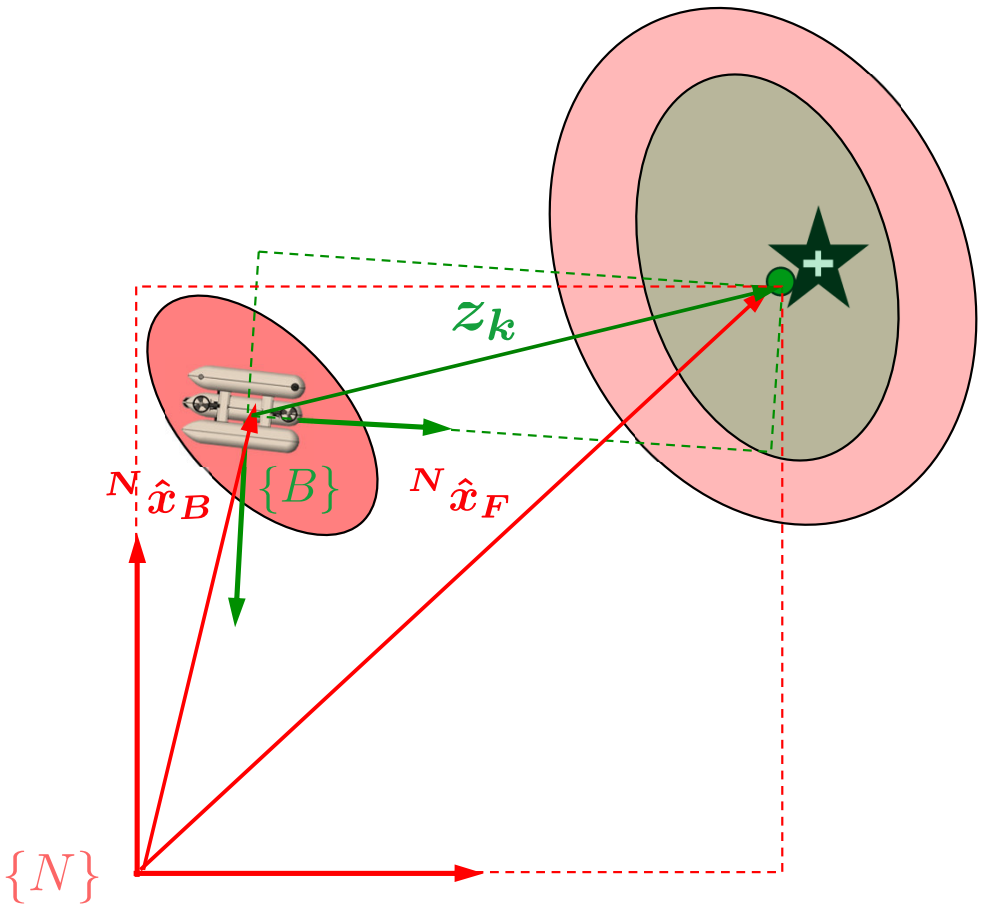
> The **robot pose estimate** is:

$$[^{N}\hat{x}_B, ^{N}P_B]$$

> The robot **makes an observation** $\mathbf{z}_k$ perturbed by a noise $^{B}\mathbf{v}_k$

EKF Map Based Localization

3D Point Feature Observed in Cartesian Coordinates



Ground Truth

Estimate

> **Ground Truth:** The robot is at $^N\mathbf{x}_B$ and the feature si at $^B\mathbf{x}_F$.

> The **robot pose estimate** is:

$$[^N\hat{\boldsymbol{x}}_B, ^N\boldsymbol{P}_B]$$

> The robot **makes an observation** $\mathbf{z}_k$ perturbed by a noise $^B\mathbf{v}_k$

> **The feature estimate** is:

$$^N\boldsymbol{x}_F = ^N\boldsymbol{x}_B \boxplus (\underbrace{^B\boldsymbol{x}_F + ^B\boldsymbol{v}_k}_{z_k})$$

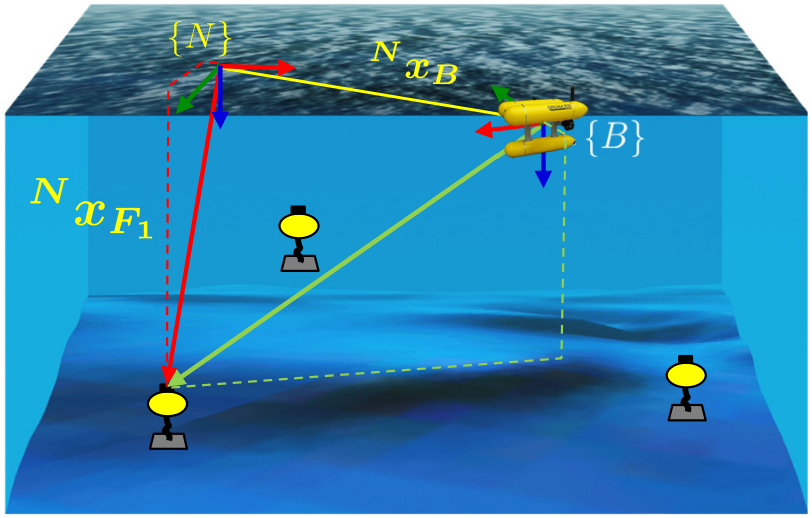
$$^N\hat{\boldsymbol{x}}_F = ^N\hat{\boldsymbol{x}}_B \boxplus \boldsymbol{z}_k$$

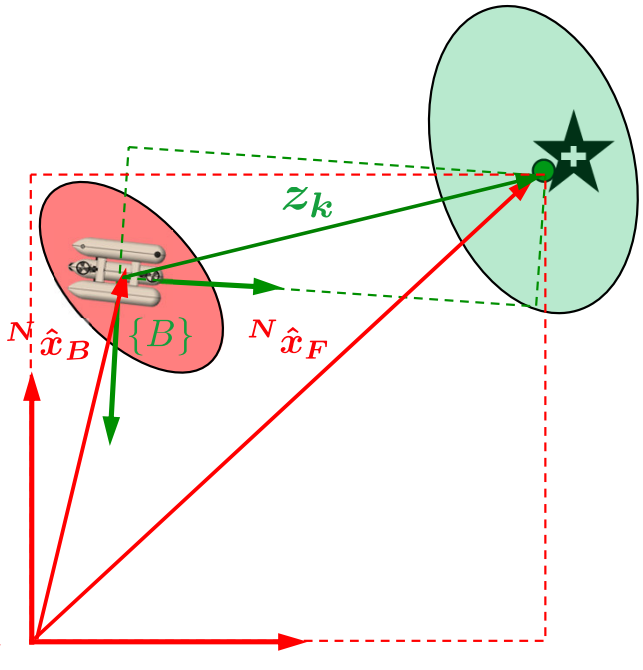
$$^N\boldsymbol{P}_F = \boldsymbol{J}_{1\boxplus} \cdot ^N\boldsymbol{P}_B \cdot \boldsymbol{J}_{1\boxplus}^T + \boldsymbol{J}_{2\boxplus} \cdot ^B\boldsymbol{P}_F \cdot \boldsymbol{J}_{2\boxplus}^T$$

where $\boxplus$ is the pose-Feature compounding operation

EKF Map Based Localization

4DPose-to-3Dpoint compunding





$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T \;;\; {}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T \;;\; {}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k} \;;\; {}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

$$\begin{aligned} {}^{N}\boldsymbol{x_F} &= {}^{N}\boldsymbol{x_B} \boxplus \underbrace{\left({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}\right)}_{\boldsymbol{z_k}} \\ &= \boldsymbol{F}\left({}^{N}\boldsymbol{x_B} \oplus \boldsymbol{F}^{\boldsymbol{T}} \cdot \left({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}\right)\right) \end{aligned}$$

Where: $\boldsymbol{F} = \left[\boldsymbol{I_{3\times 3}} \quad \boldsymbol{0_{3\times 1}}\right]$

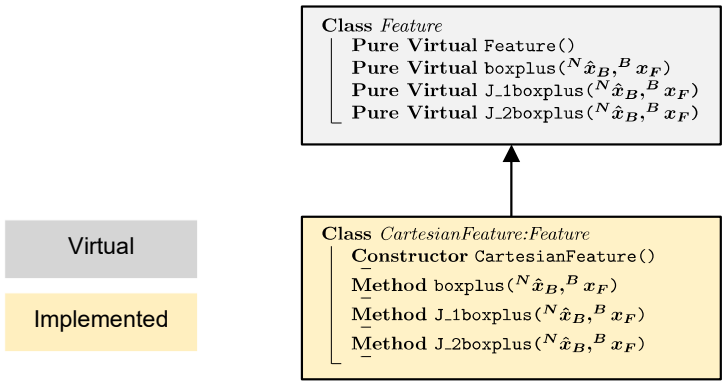
Whose Jacobian are given by:

$$\begin{aligned} \boldsymbol{J_{1\boxplus}} &= \frac{\partial {}^{N}\boldsymbol{x_B} \boxplus \left({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}\right)}{\partial {}^{N}\boldsymbol{x_B}} \\ &= \left.\frac{\partial \boldsymbol{F}\left({}^{N}\boldsymbol{x_B} \oplus \boldsymbol{F}^{\boldsymbol{T}} \cdot {}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}\right)}{\partial {}^{N}\boldsymbol{x_B}}\right|_{{}^{N}\boldsymbol{x_{B_k}} = {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{B}\boldsymbol{v_k} = \boldsymbol{0}} = \boldsymbol{F} \cdot \boldsymbol{J_{1\oplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{B}\boldsymbol{x_F}) \end{aligned}$$

$$\begin{aligned} \boldsymbol{J_{2\boxplus}} &= \frac{\partial {}^{N}\boldsymbol{x_B} \boxplus \left({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}\right)}{\partial {}^{B}\boldsymbol{x_F}} \\ &= \left.\frac{\partial \boldsymbol{F}\left({}^{N}\boldsymbol{x_B} \oplus \boldsymbol{F}^{\boldsymbol{T}} \cdot {}^{B}\boldsymbol{x_F}\right)}{\partial {}^{N}\boldsymbol{x_B}}\right|_{{}^{N}\boldsymbol{x_{B_k}} = {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{B}\boldsymbol{v_k} = \boldsymbol{0}} = \boldsymbol{F} \cdot \boldsymbol{J_{2\oplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{F}^{\boldsymbol{T}} \end{aligned}$$

EKF Map Based Localization

3D Point Feature: Implementation



Class *Feature*

- **Pure Virtual** `boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F) \rightarrow {}^N\boldsymbol{x}_F$;
- **Pure Virtual** `J_1boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F) \rightarrow \boldsymbol{J_{1\boxplus}}$;
- **Pure Virtual** `J_2boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F) \rightarrow \boldsymbol{J_{2\boxplus}}$;

Class *CartesianFeature:Feature*

- **Method** `Feature`(x, y, z)
  - $feature = \begin{bmatrix} x & y & z \end{bmatrix}^T$;
  - $\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$; // 3D to 4D
  - **return**
- **Method** `boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F)$
  - ${}^N\boldsymbol{x}_F = \boldsymbol{F}({}^N\hat{\boldsymbol{x}}_B \oplus \boldsymbol{F}^T \cdot {}^B\boldsymbol{x}_F)$
  - **return** ${}^N\boldsymbol{x}_F$
- **Method** `J_1boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F)$
  - **return** $\boldsymbol{F}\boldsymbol{J}_{1\oplus}({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F)$
- **Method** `J_2boxplus`$({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F)$
  - **return** $\boldsymbol{F}\boldsymbol{J}_{2\oplus}({}^N\hat{\boldsymbol{x}}_B, {}^B\boldsymbol{x}_F)\boldsymbol{F}^T$

Pose: ${}^N\boldsymbol{x}_B = \begin{bmatrix} {}^N x_B & {}^N y_B & {}^N z_B & {}^N\psi_B \end{bmatrix}^T$; ${}^N\boldsymbol{x}_B \equiv \mathcal{N}({}^N\hat{\boldsymbol{x}}_B, {}^N\boldsymbol{P}_B)$

Storage: ${}^B\boldsymbol{x}_F = \begin{bmatrix} {}^B x_F & {}^B y_F & {}^B z_F \end{bmatrix}^T$; ${}^B\boldsymbol{x}_F \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = {}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k$; ${}^B\boldsymbol{v}_k \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^B\boldsymbol{P}_F}_{\boldsymbol{R_k}})$

$$
\begin{aligned}
{}^N\boldsymbol{x}_F &= {}^N\boldsymbol{x}_B \boxplus \underbrace{\left({}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k\right)}_{\boldsymbol{z_k}} \\
&= \boldsymbol{F}\left({}^N\boldsymbol{x}_B \oplus \boldsymbol{F}^T \cdot \left({}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k\right)\right)
\end{aligned}
$$

Where: $\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 1} \end{bmatrix}$

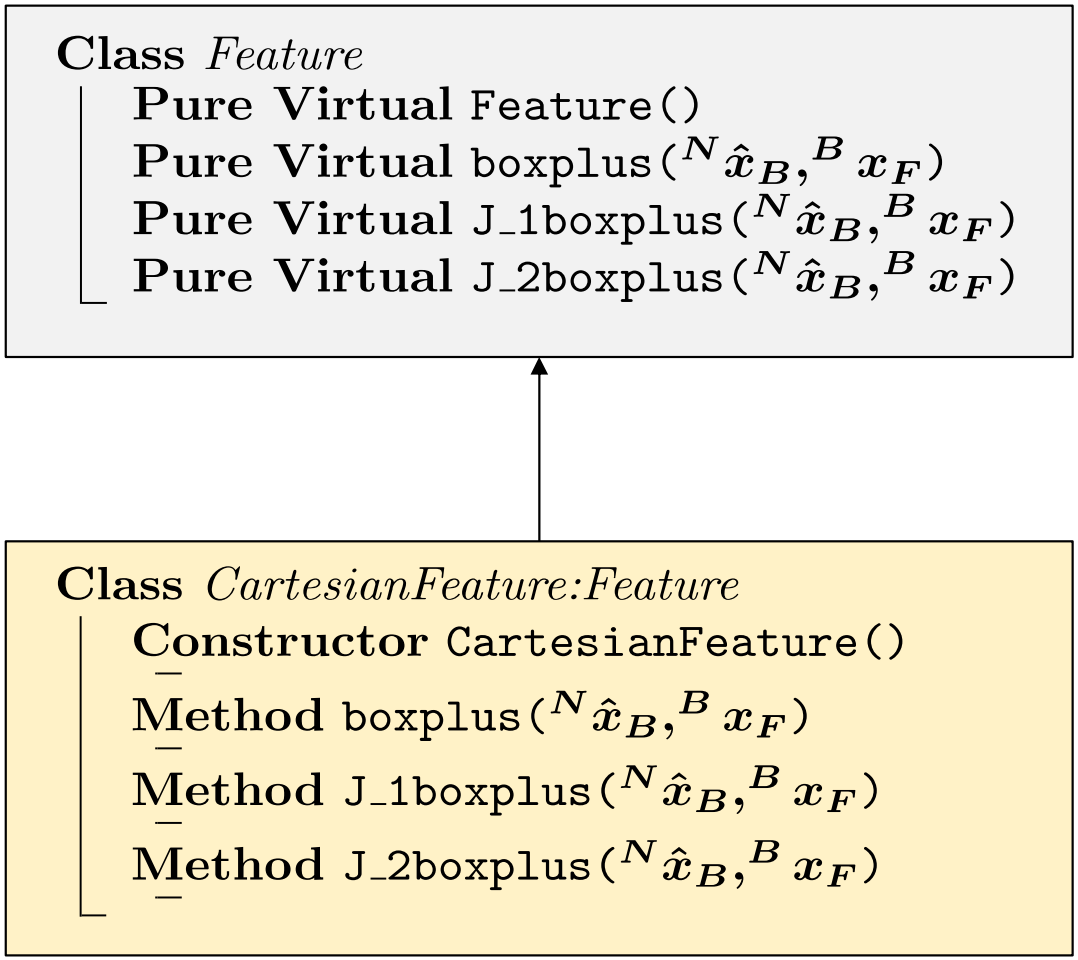
Whose Jacobian are given by:

$$
\begin{aligned}
\boldsymbol{J}_{1\boxplus} &= \frac{\partial {}^N\boldsymbol{x}_B \boxplus \left({}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k\right)}{\partial {}^N\boldsymbol{x}_B} \\
&= \left.\frac{\partial \boldsymbol{F}\left({}^N\boldsymbol{x}_B \oplus \boldsymbol{F}^T \cdot {}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k\right)}{\partial {}^N\boldsymbol{x}_B}\right|_{{}^N\boldsymbol{x}_{B_k} = {}^N\hat{\boldsymbol{x}}_{B_k}, {}^B\boldsymbol{v}_k = \boldsymbol{0}} = \boldsymbol{F} \cdot \boldsymbol{J}_{1\oplus}({}^N\hat{\boldsymbol{x}}_{B_k}, {}^B\boldsymbol{x}_F)
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{J}_{2\boxplus} &= \frac{\partial {}^N\boldsymbol{x}_B \boxplus \left({}^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k\right)}{\partial {}^B\boldsymbol{x}_F} \\
&= \left.\frac{\partial \boldsymbol{F}\left({}^N\boldsymbol{x}_B \oplus \boldsymbol{F}^T \cdot {}^B\boldsymbol{x}_F\right)}{\partial {}^N\boldsymbol{x}_B}\right|_{{}^N\boldsymbol{x}_{B_k} = {}^N\hat{\boldsymbol{x}}_{B_k}, {}^B\boldsymbol{v}_k = \boldsymbol{0}} = \boldsymbol{F} \cdot \boldsymbol{J}_{2\oplus}({}^N\hat{\boldsymbol{x}}_{B_k}, {}^B\boldsymbol{x}_F) \cdot \boldsymbol{F}^T
\end{aligned}
$$

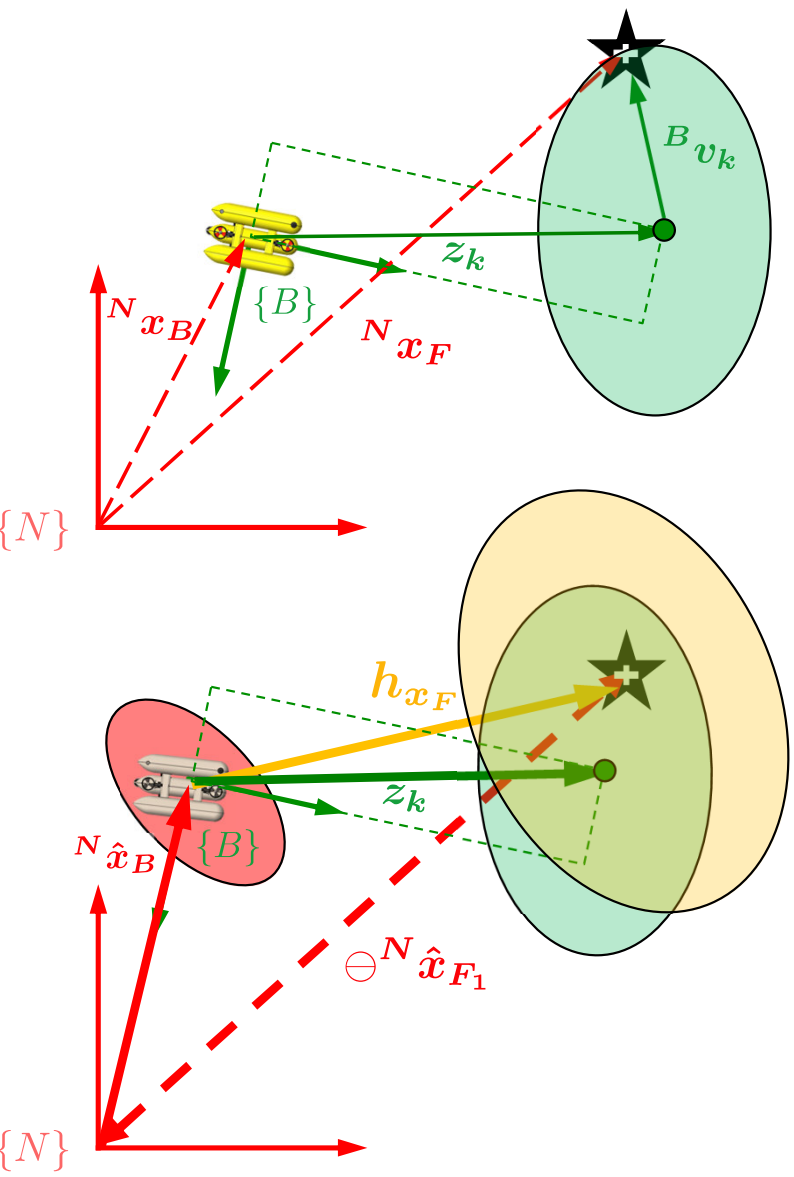
EKF Map Based Localization

3D Point Feature: Implementation



EKF Map Based Localization

3D Point Feature Observed in Cartesian



Pose: $^{N}\boldsymbol{x_B} = [^{N}x_B \;\; ^{N}y_B \;\; ^{N}z_B \;\; ^{N}\psi_B]^T \;;\; ^{N}\boldsymbol{x_B} \equiv \mathcal{N}(^{N}\hat{\boldsymbol{x}}_B, ^{N}\boldsymbol{P_B})$

Storage: $^{B}\boldsymbol{x_F} = \begin{bmatrix} ^{B}x_F & ^{B}y_F & ^{B}z_F \end{bmatrix}^T \;;\; ^{B}\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = ^{B}\boldsymbol{x_F} + ^{B}\boldsymbol{v_k} \;;\; ^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

> Sensor Model:

$$
\begin{aligned}
^{B}\boldsymbol{z_k} &= h_{\boldsymbol{x_F}}(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{v_k}) \\
&= \left((\ominus ^{N}\boldsymbol{x_B}) \boxplus ^{N}\boldsymbol{x_F}\right) + ^{B}\boldsymbol{v_k}
\end{aligned}
$$

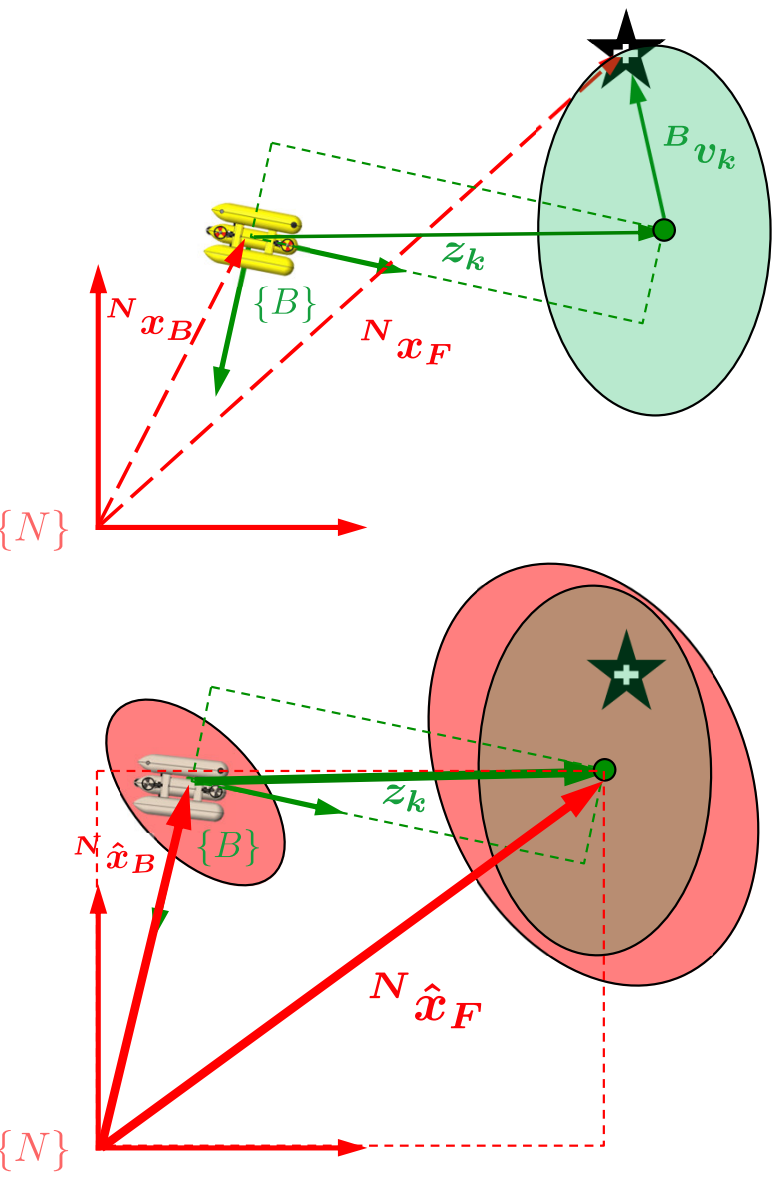
> Observation Jacobians:

$$
\begin{aligned}
\boldsymbol{H_k} &= \frac{\partial h_{\boldsymbol{x_F}}(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{v_k})}{\partial ^{N}\boldsymbol{x_B}} \\
&= \left.\frac{\partial(\ominus ^{N}\boldsymbol{x_B} \boxplus ^{N}\boldsymbol{x_F}) + ^{B}\boldsymbol{v_k}}{\partial ^{N}\boldsymbol{x_B}}\right|_{^{N}\boldsymbol{x_B}=^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{v_k}=0} = \boldsymbol{J_{1\boxplus}}(^{N}\hat{\boldsymbol{x}}_B, ^{N}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}(^{N}\hat{\boldsymbol{x}}_B)
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{V_k} &= \frac{\partial h_{\boldsymbol{x_F}}(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{v_k})}{\partial ^{B}\boldsymbol{v_k}} \\
&= \left.\frac{\partial(\ominus ^{N}\boldsymbol{x_B} \boxplus ^{N}\boldsymbol{x_F}) + ^{B}\boldsymbol{v_k}}{\partial ^{B}\boldsymbol{v_k}}\right|_{^{N}\boldsymbol{x_B}=^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{v_k}=0} = \boldsymbol{I}
\end{aligned}
$$

EKF Map Based Localization

3D Point Feature Observed in Cartesian



Pose: $^{N}\boldsymbol{x_B} = [^{N}x_B \;\; ^{N}y_B \;\; ^{N}z_B \;\; ^{N}\psi_B]^T \;;\; ^{N}\boldsymbol{x_B} \equiv \mathcal{N}(^{N}\hat{\boldsymbol{x}}_B, ^{N}\boldsymbol{P_B})$

Storage: $^{B}\boldsymbol{x_F} = [^{B}x_F \;\; ^{B}y_F \;\; ^{B}z_F]^T \;;\; ^{B}\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $z_k = ^{B}\boldsymbol{x_F} + ^{B}\boldsymbol{v_k} \;;\; ^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

> Inverted Sensor Model:

$$^{N}\boldsymbol{x_F} = g(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{x_F}, ^{B}\boldsymbol{v_k}) = ^{N}\boldsymbol{x_B} \boxplus (^{B}\boldsymbol{x_F} + ^{B}\boldsymbol{v_k})$$

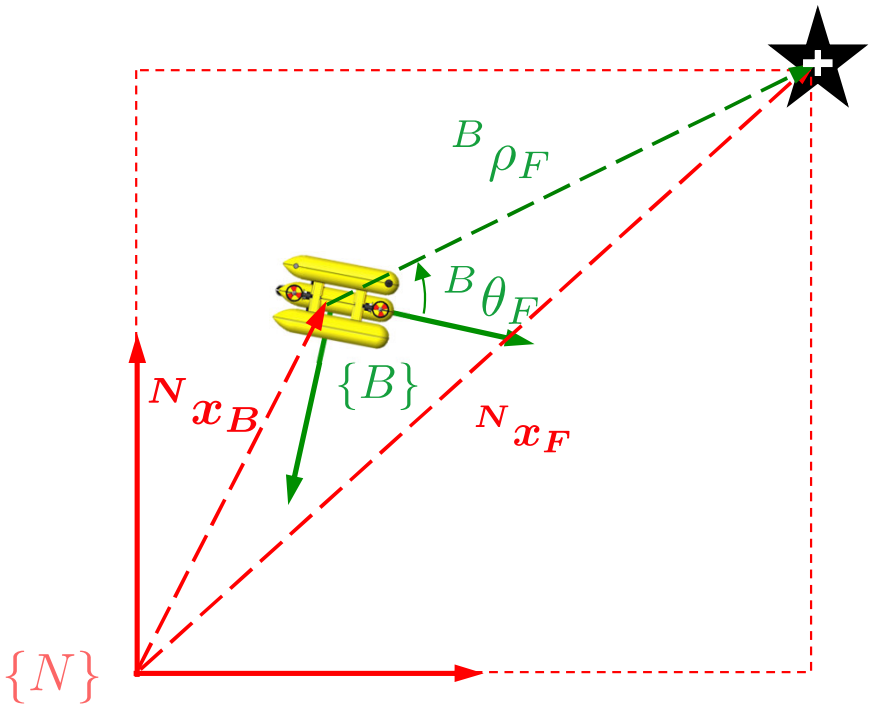
> Jacobians:

$$\boldsymbol{J_{g_x}} = \frac{\partial g(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{x_F}, ^{B}\boldsymbol{v_k})}{\partial ^{N}\boldsymbol{x_B}} = \left.\frac{\partial ^{N}\boldsymbol{x_B} \boxplus (^{B}\boldsymbol{x_F} + ^{B}\boldsymbol{v_k})}{\partial ^{N}\boldsymbol{x_B}}\right|_{^{N}\boldsymbol{x_B}=^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{v_k}=\boldsymbol{0}} = \boldsymbol{J_{1\boxplus}}(^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{x_F})$$

$$\boldsymbol{J_{g_v}} = \frac{\partial g(^{N}\boldsymbol{x_B}, ^{B}\boldsymbol{x_F}, \boldsymbol{v_k})}{\partial ^{B}\boldsymbol{v_k}} = \left.\frac{\partial ^{N}\boldsymbol{x_B} \boxplus (^{B}\boldsymbol{x_F} + ^{B}\boldsymbol{v_k})}{\partial \boldsymbol{v_k}}\right|_{^{N}\boldsymbol{x_B}=^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{v_k}=\boldsymbol{0}} = \boldsymbol{J_{2\boxplus}}(^{N}\hat{\boldsymbol{x}}_B, ^{B}\boldsymbol{x_F})$$

EKF Map Based Localization

3D Point Feature Observed in Spherical

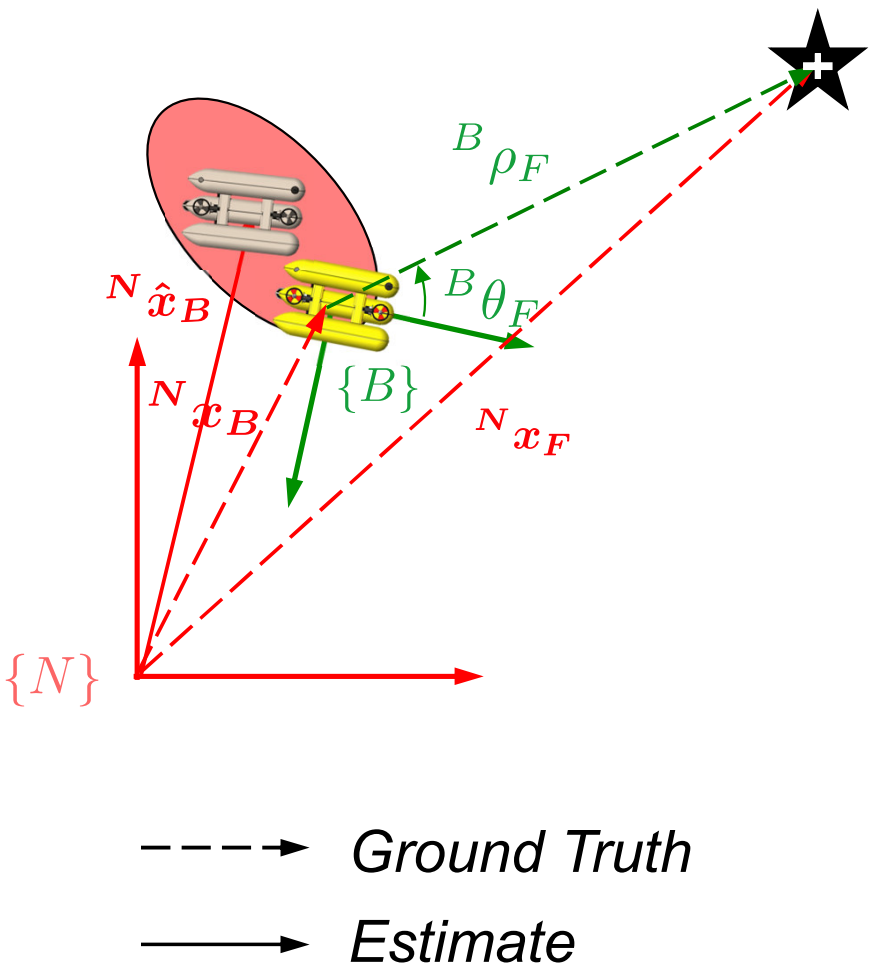


Ground Truth

> **Ground Truth** The robot is at $^{N}\mathbf{x}_{B}$ and the feature si at $^{B}\mathbf{x}_{F}$.

EKF Map Based Localization

3D Point Feature Observed in Spherical



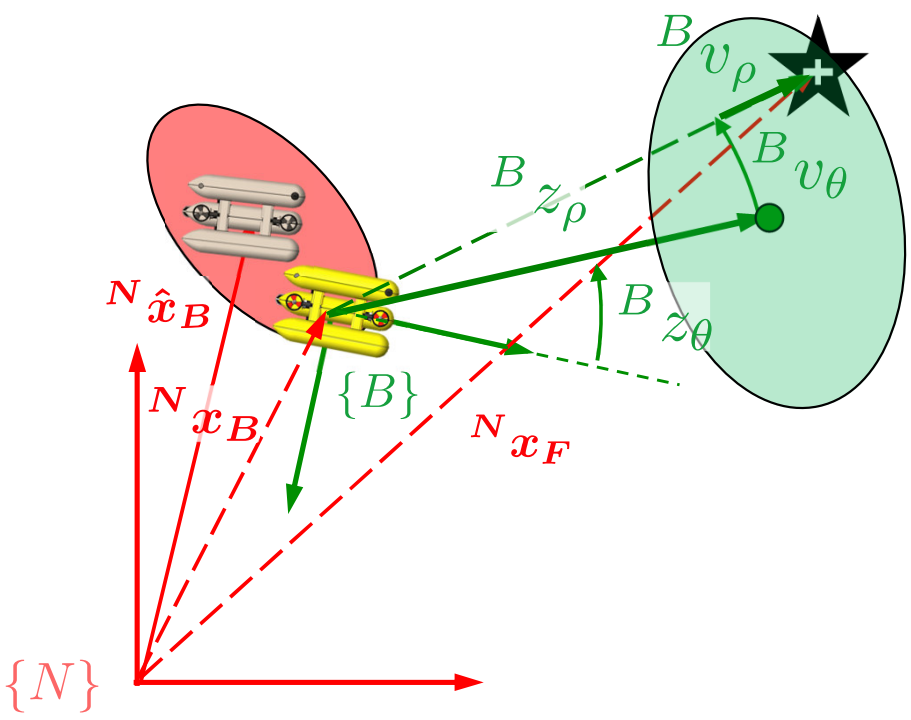
> **Ground Truth** The robot is at $^{N}\mathbf{x}_{B}$ and the feature si at $^{B}\mathbf{x}_{F}$.

> The **robot pose estimate** is:

$$[^{N}\hat{\boldsymbol{x}}_{B}, ^{N}\boldsymbol{P}_{B}]$$

EKF Map Based Localization

3D Point Feature Observed in Spherical



Ground Truth

Estimate

> **Ground Truth** The robot is at $^{N}\mathbf{x}_{B}$ and the feature si at $^{B}\mathbf{x}_{F}$.

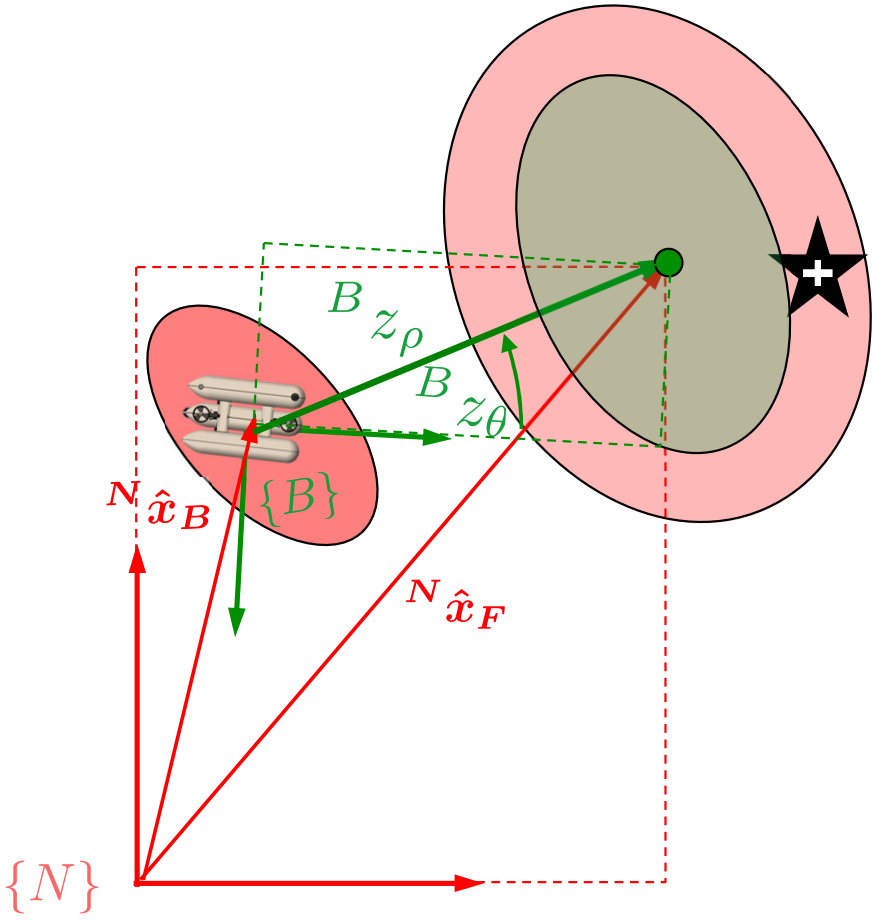
> The **robot pose estimate** is:

$$[^{N}\hat{\boldsymbol{x}}_{B}, ^{N}\boldsymbol{P}_{B}]$$

> The robot **makes a polar observation** $\mathbf{z}_{k}$ perturbed by a noise $^{B}\mathbf{v}_{k}$

EKF Map Based Localization

3D Point Feature Observed in Spherical



Ground Truth

Estimate

> **Ground Truth** The robot is at $^N\mathbf{x}_B$ and the feature si at $^B\mathbf{x}_F$.

> The **robot pose estimate** is:

$$[^N\hat{\boldsymbol{x}}_B, ^N\boldsymbol{P}_B]$$

> The robot **makes an observation** $\mathbf{z}_k$ perturbed by a noise $^B\mathbf{v}_k$

> **The Cartesian feature estimate** is:

$$^N\boldsymbol{x}_F = {}^N\boldsymbol{x}_B \boxplus s2c(^B\boldsymbol{x}_F + {}^B\boldsymbol{v}_k)$$

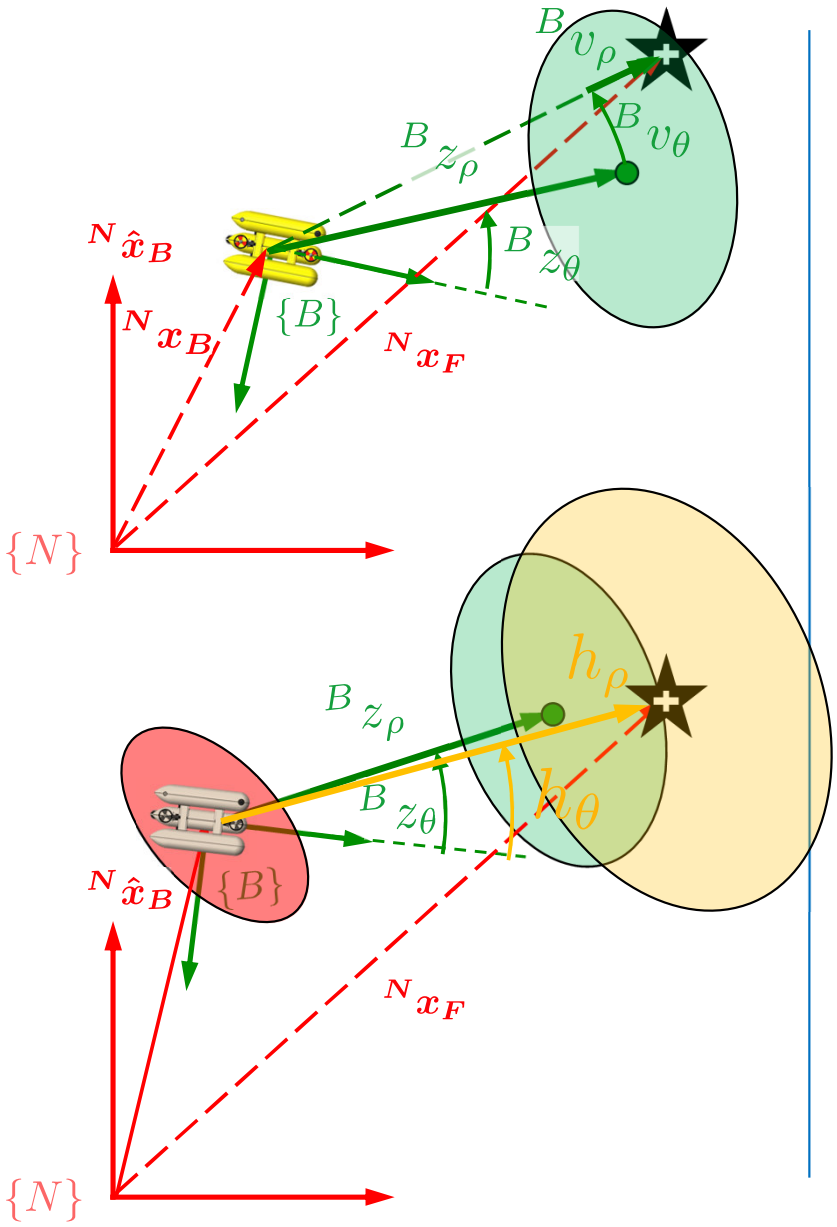
$$^N\hat{\boldsymbol{x}}_F = {}^N\hat{\boldsymbol{x}}_B \boxplus s2c(\boldsymbol{z}_k)$$

$$^N\boldsymbol{P}_F = \boldsymbol{J}_{1\boxplus} \cdot {}^N\boldsymbol{P}_B \cdot \boldsymbol{J}_{1\boxplus}^T + \boldsymbol{J}_{2\boxplus} \cdot \boldsymbol{J}_{s2c} \cdot {}^B\boldsymbol{P}_F \cdot \boldsymbol{J}_{s2c}^T \cdot \boldsymbol{J}_{2\boxplus}^T$$

where $\boxplus$ is the pose-Feature compounding operation

EKF Map Based Localization

3D Point Feature Observed in Spherical



Pose: $^N\boldsymbol{x_B} = [^N x_B \;\; ^N y_B \;\; ^N z_B \;\; ^N \psi_B]^T \;;\; ^N\boldsymbol{x_B} \equiv \mathcal{N}(^N\hat{\boldsymbol{x}}_B, ^N\boldsymbol{P_B})$

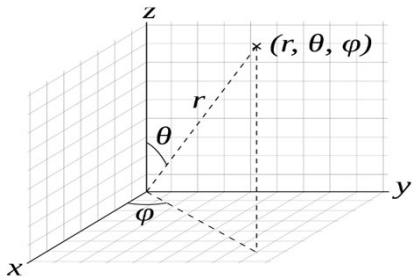
Storage: $^N\boldsymbol{x_F} = [^N x_F \;\; ^N y_F \;\; ^N z_F]^T \;;\; ^N\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = s2c(^B\boldsymbol{x_F} + ^B\boldsymbol{v_k}) \;;\; ^B\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{^B\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

$$^B\boldsymbol{x_F} = [^B\rho_F \;\; ^B\theta_F \;\; ^B\varphi_F \;\;]^T$$

> Sensor Model:

$$\begin{aligned}\boldsymbol{z_k} &= h_{\boldsymbol{x_F}}(^N\boldsymbol{x_B}, ^B\boldsymbol{v_k}) \\ &= c2s\left(\ominus ^N\boldsymbol{x_B} \boxplus ^N\boldsymbol{x_F}\right) + ^B\boldsymbol{v_k}\end{aligned}$$



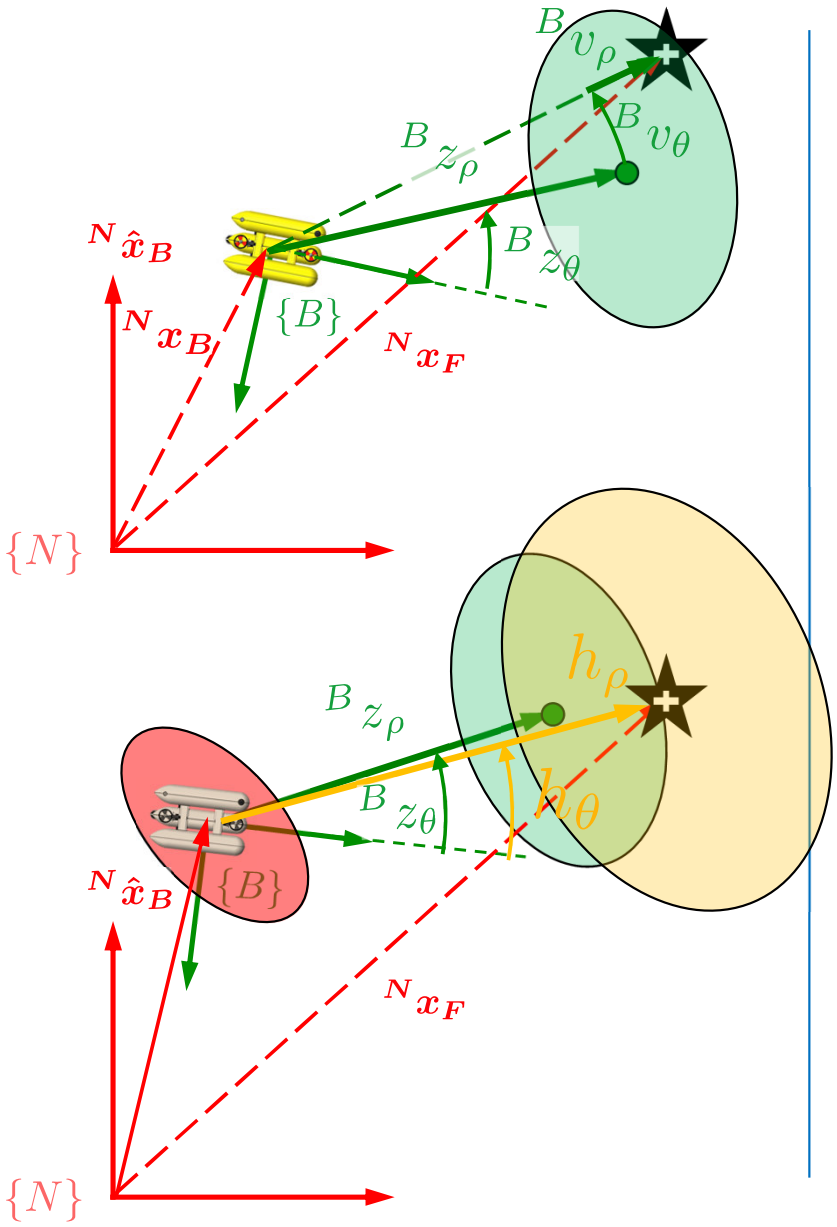
Where:

$$\begin{bmatrix}\rho\\ \theta\\ \varphi\end{bmatrix} = c_{2s}\left(\begin{bmatrix}x\\ y\\ z\end{bmatrix}\right) = \begin{bmatrix}\sqrt{x^2+y^2+z^2}\\ atan2(\sqrt{x^2+y^2}, z)\\ atan2(y,x)\end{bmatrix}\Bigg|_{[x\;\;y\;\;z]=[\hat{x}\;\;\hat{y}\;\;\hat{z}]}$$

$$\boldsymbol{J_{c_{2s}}} = \begin{bmatrix}\frac{\hat{x}}{\sqrt{\hat{x}^2+\hat{y}^2+\hat{z}^2}} & \frac{\hat{y}}{\sqrt{\hat{x}^2+\hat{y}^2+\hat{z}^2}} & \frac{\hat{z}}{\sqrt{\hat{x}^2+\hat{y}^2+\hat{z}^2}}\\ \frac{-\hat{y}}{\hat{x}^2+\hat{y}^2} & \frac{\hat{x}}{\hat{x}^2+\hat{y}^2} & 0\\ \frac{\hat{x}\hat{z}}{(\hat{x}^2+\hat{y}^2+\hat{z}^2)\sqrt{\hat{x}^2+\hat{y}^2}} & \frac{\hat{y}\hat{z}}{(\hat{x}^2+\hat{y}^2+\hat{z}^2)\sqrt{\hat{x}^2+\hat{y}^2}} & \frac{-\sqrt{\hat{x}^2+\hat{y}^2}}{(\hat{x}^2+\hat{y}^2+\hat{z}^2)}\end{bmatrix}$$

EKF Map Based Localization

3D Point Feature Observed in Spherical



Pose: $^N\boldsymbol{x_B} = [^N x_B \;\; ^N y_B \;\; ^N z_B \;\; ^N \psi_B]^T \;;\; ^N\boldsymbol{x_B} \equiv \mathcal{N}(^N\hat{\boldsymbol{x}}_B, ^N\boldsymbol{P_B})$

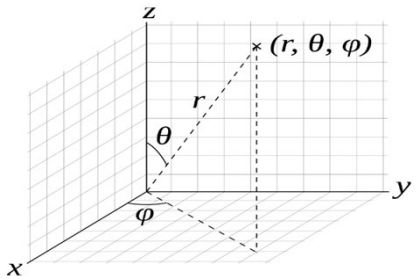
Storage: $^N\boldsymbol{x_F} = [^N x_F \;\; ^N y_F \;\; ^N z_F]^T \;;\; ^N\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = s2c(^B\boldsymbol{x_F} + ^B\boldsymbol{v_k}) \;;\; ^B\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{^B\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

$$^B\boldsymbol{x_F} = [^B\rho_F \;\; ^B\theta_F \;\; ^B\varphi_F \;\;]^T$$

> Sensor Model:

$$\begin{aligned}\boldsymbol{z_k} &= h_{\boldsymbol{x_F}}(^N\boldsymbol{x_B}, ^B\boldsymbol{v_k}) \\ &= c2s\left(\ominus ^N\boldsymbol{x_B} \boxplus ^N\boldsymbol{x_F}\right) + ^B\boldsymbol{v_k}\end{aligned}$$



> Observation Jacobians:

$$\boldsymbol{H_k} = \frac{\partial h_{\boldsymbol{x_F}}(^N\boldsymbol{x_B}, ^B\boldsymbol{v_k})}{\partial ^N\boldsymbol{x_B}} = \left.\frac{\partial c2s\left(\ominus ^N\boldsymbol{x_B} \boxplus ^N\boldsymbol{x_F}\right) + ^B\boldsymbol{v_k}}{\partial ^N\boldsymbol{x_B}}\right|_{^N\boldsymbol{x_B}=^N\hat{\boldsymbol{x}}_B, ^B\boldsymbol{v_k}=\boldsymbol{0}}$$

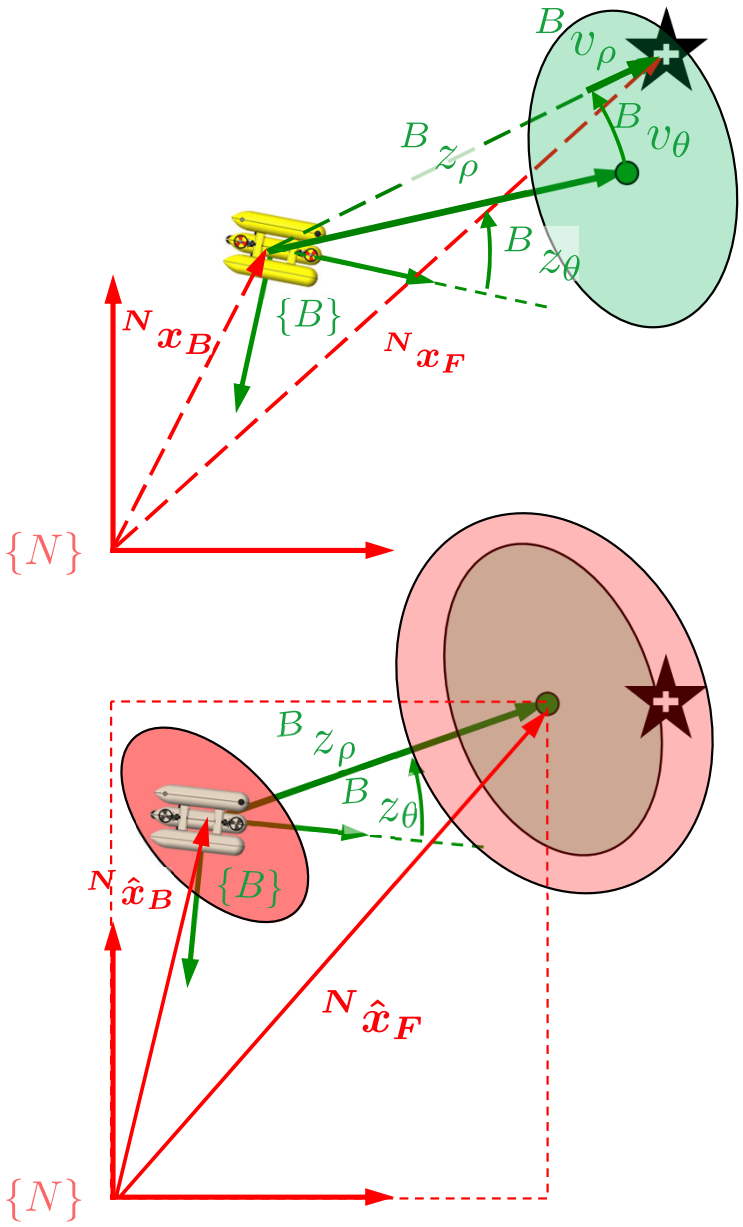
$$= \boldsymbol{J_{c2s}}\left(\ominus ^N\hat{\boldsymbol{x}}_B \boxplus ^N\boldsymbol{x_F}\right) \cdot \boldsymbol{J_{1\boxplus}}(\ominus ^N\hat{\boldsymbol{x}}_B, ^B\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}(^N\hat{\boldsymbol{x}}_B)$$

$$\boldsymbol{V_k} = \frac{\partial h_{\boldsymbol{x_F}}(^N\boldsymbol{x_B}, ^B\boldsymbol{v_k})}{\partial ^B\boldsymbol{v_k}}$$

$$= \left.\frac{\partial c2s\left((\ominus ^N\boldsymbol{x_B}) \boxplus ^N\boldsymbol{x_F}\right) + ^B\boldsymbol{v_k}}{\partial \boldsymbol{v_k}}\right|_{^B\boldsymbol{v_k}=\boldsymbol{0}} = \boldsymbol{I}$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model



Pose: ${}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T \;;\; {}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$

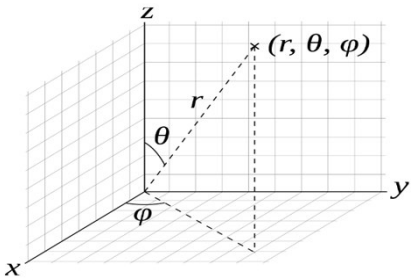
Storage: ${}^{N}\boldsymbol{x_F} = [{}^{N}x_F \;\; {}^{N}y_F \;\; {}^{N}z_F]^T \;;\; {}^{N}\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}) \;;\; {}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

$${}^{B}\boldsymbol{x_F} = [{}^{B}\rho_F \;\; {}^{B}\theta_F \;\; {}^{B}\varphi_F \;]^T$$

> Inverted Sensor Model:

$$\begin{aligned} {}^{N}\boldsymbol{x_F} &= g({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_F}, \boldsymbol{v_k}) \\ &= {}^{N}\boldsymbol{x_B} \boxplus s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k} \end{aligned}$$



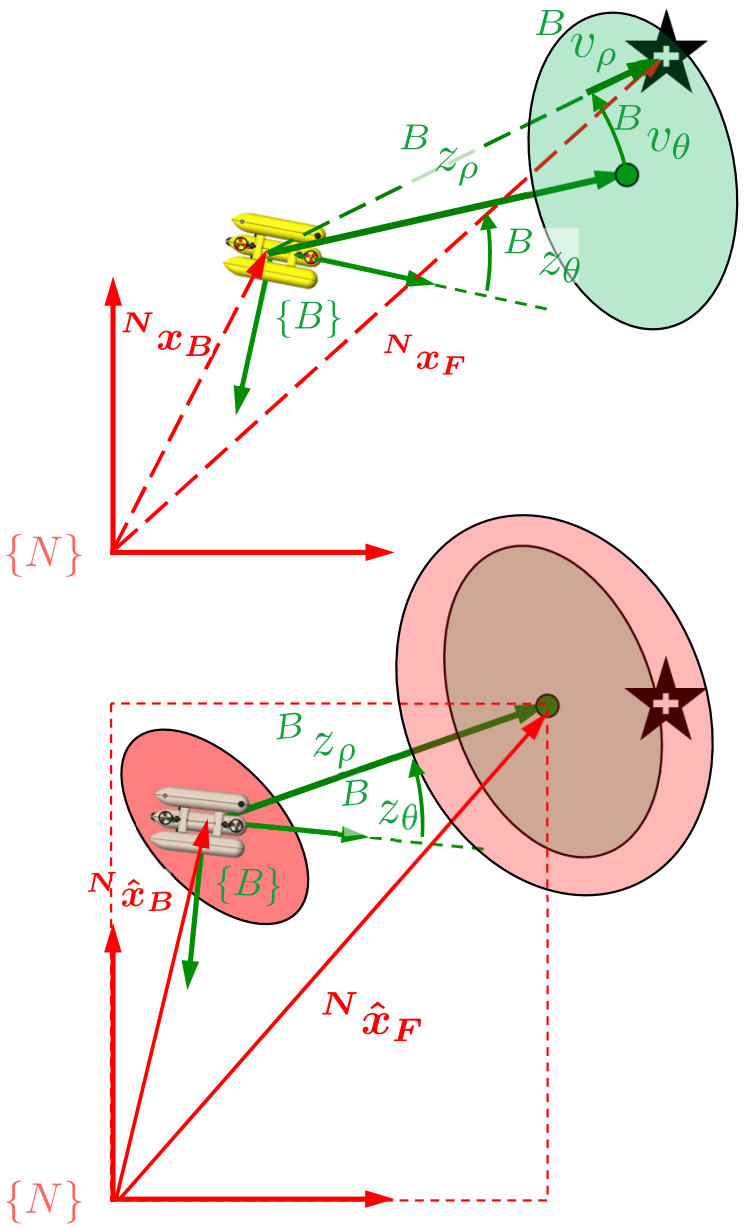
where:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = s_{2c}\left(\begin{bmatrix} \rho \\ \theta \\ \varphi \end{bmatrix}\right) = \begin{bmatrix} \rho\cos\varphi\,\sin\theta \\ \rho\sin\varphi\,\sin\theta \\ \rho\cos\theta \end{bmatrix}\Bigg|_{[\rho\;\theta\;\varphi]=[\hat{\rho}\;\hat{\theta}\;\hat{\varphi}]}$$

$$\boldsymbol{J_{s2c}} = \begin{bmatrix} \sin\hat{\theta}\cos\hat{\varphi} & \hat{\rho}\cos\hat{\theta}\cos\hat{\varphi} & -\hat{\rho}\sin\hat{\theta}\sin\hat{\varphi} \\ \sin\hat{\theta}\sin\hat{\varphi} & \hat{\rho}\cos\hat{\theta}\sin\hat{\varphi} & \hat{\rho}\sin\hat{\theta}\cos\hat{\varphi} \\ \cos\hat{\theta} & -\hat{\rho}\sin\hat{\theta} & 0 \end{bmatrix}$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model



Pose: ${}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T \;;\; {}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_B, {}^{N}\boldsymbol{P_B})$

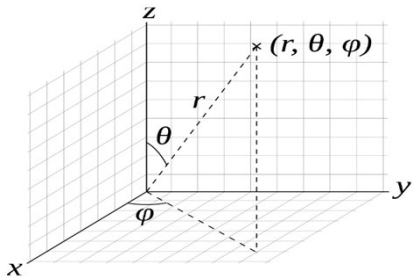
Storage: ${}^{N}\boldsymbol{x_F} = [{}^{N}x_F \;\; {}^{N}y_F \;\; {}^{N}z_F]^T \;;\; {}^{N}\boldsymbol{x_F} \in \mathbb{R}^3$

Observation: $\boldsymbol{z_k} = s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k}) \;;\; {}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$

$${}^{B}\boldsymbol{x_F} = [{}^{B}\rho_F \;\; {}^{B}\theta_F \;\; {}^{B}\varphi_F \;]^T$$

> Inverted Sensor Model:

$$\begin{aligned} {}^{N}\boldsymbol{x_F} &= g({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_F}, \boldsymbol{v_k}) \\ &= {}^{N}\boldsymbol{x_B} \boxplus s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k} \end{aligned}$$



> Jacobians:

$$\boldsymbol{J_{g_x}} = \frac{\partial g({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_F}, {}^{B}\boldsymbol{v_k})}{\partial {}^{N}\boldsymbol{x_B}} = \left.\frac{\partial {}^{N}\boldsymbol{x_B} \boxplus s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})}{\partial {}^{N}\boldsymbol{x_B}}\right|_{{}^{N}\boldsymbol{x_B}={}^{N}\hat{\boldsymbol{x}}_B, {}^{B}\boldsymbol{v_k}=0}$$

$$= \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_B, s2c({}^{B}\boldsymbol{x_F}))$$

$$\boldsymbol{J_{g_v}} = \frac{\partial g({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_F}, \boldsymbol{v_k})}{\partial {}^{B}\boldsymbol{v_k}} = \left.\frac{\partial {}^{N}\boldsymbol{x_B} \boxplus s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})}{\partial \boldsymbol{v_k}}\right|_{{}^{N}\boldsymbol{x_B}={}^{N}\hat{\boldsymbol{x}}_B, {}^{B}\boldsymbol{v_k}=0}$$

$$= \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_B, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J_{s2c}}({}^{B}\boldsymbol{x_F})$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model

Sensor Model of a Cartesian 3D Point Feature

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$
$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Sensor Model:

$$\boldsymbol{z_k} = h_{\boldsymbol{x_F}}({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{v_k})$$
$$= \left((\ominus {}^{N}\boldsymbol{x_B}) \boxplus {}^{N}\,\boldsymbol{x_F}\right) + {}^{B}\boldsymbol{v_k}$$

> Jacobians:

$$\boldsymbol{H_k} = \boldsymbol{J}_{1\boxplus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{x_F}) \cdot \boldsymbol{J}_{\ominus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$$
$$\boldsymbol{V_k} = \boldsymbol{I}$$

Observed in Spherical

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{N}\boldsymbol{x_F} = \left[{}^{N}x_F \;\; {}^{N}y_F \;\; {}^{N}z_F\right]^T$$
$${}^{N}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = c2s({}^{B}\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k})$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Sensor Model:

$$\boldsymbol{z_k} = h_{\boldsymbol{x_F}}({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{v_k})$$
$$= c2s\left(\ominus {}^{N}\boldsymbol{x_B} \boxplus {}^{N}\,\boldsymbol{x_F}\right) + {}^{B}\boldsymbol{v_k}$$

> Jacobians:

$$\boldsymbol{H_k} = \boldsymbol{J}_{c2s}\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}\right) \cdot \boldsymbol{J}_{1\boxplus}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J}_{\ominus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$$
$$\boldsymbol{V_k} = \boldsymbol{I}$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model

Sensor Model of a Cartesian 3D Point Feature

Observed in Cartesian	Observed in Spherical

- Using the pose-feature compounding operation the expression of the **observation equation** and the observation **Jacobians** are **independent of the feature type.**
- What **changes** is the particular value of the **pose-feature compounding** operator**, its Jacobians, the o2s(), s2o()** and **their Jacobians**

Observation: $z_k = {}^B x_F + {}^B v_k$

$${}^B v_k \equiv \mathcal{N}(0, \underbrace{{}^B P_F}_{R_k})$$

> Sensor Model:

$$z_k = h_{x_F}({}^N x_B, {}^B v_k)$$
$$= s2o\left(\ominus {}^N x_B \boxplus {}^N x_F\right) + {}^B v_k$$

> Jacobians:

$$H_k = J_{s2o}\left(\ominus {}^N \hat{x}_B \boxplus {}^N x_F\right) \cdot J_{1\boxplus}({}^N \hat{x}_B, {}^B x_F) \cdot J_{\ominus}({}^N \hat{x}_B)$$
$$V_k = I$$

Observation: $z_k = c2s({}^B x_F + {}^B v_k)$

$${}^B v_k \equiv \mathcal{N}(0, \underbrace{{}^B P_F}_{R_k})$$

> Sensor Model:

$$z_k = h_{x_F}({}^N x_B, {}^B v_k)$$
$$= s2o\left(\ominus {}^N x_B \boxplus {}^N x_F\right) + {}^B v_k$$

> Jacobians:

$$H_k = J_{s2o}\left(\ominus {}^N \hat{x}_B \boxplus {}^N x_F\right) \cdot J_{1\boxplus}({}^N \hat{x}_B, {}^B x_F) \cdot J_{\ominus}({}^N \hat{x}_B)$$
$$V_k = I$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model

Inverted Sensor Model of a Cartesian 3D Point Feature

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$
$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Inverted Sensor Model:

$${}^{N}\boldsymbol{x_F} = g({}^{N}\boldsymbol{x_B}, {}^{B}\,\boldsymbol{x_F}, \boldsymbol{v_k})$$
$$= {}^{N}\,\boldsymbol{x_B} \boxplus ({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})$$

> Jacobians:

$$\boldsymbol{J_{g_x}} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F})$$
$$\boldsymbol{J_{g_v}} = \boldsymbol{I}$$

Observed in Spherical

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{N}\boldsymbol{x_F} = \left[{}^{N}x_F \;\; {}^{N}y_F \;\; {}^{N}z_F\right]^T$$
$${}^{N}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = c2s({}^{B}\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k})$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Inverted Sensor Model:

$${}^{N}\boldsymbol{x_F} = g({}^{N}\boldsymbol{x_B}, {}^{B}\,\boldsymbol{x_F}, \boldsymbol{v_k})$$
$$= {}^{N}\,\boldsymbol{x_B} \boxplus s2c({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})$$

> Jacobians:

$$\boldsymbol{J_{g_x}} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, s2c({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{s2c}}\left({}^{N}\boldsymbol{x_F}\right)$$
$$\boldsymbol{J_{g_v}} = \boldsymbol{I}$$

EKF Map Based Localization

3D Point Feature Observed in Spherical: Inverted Sensor Model

Inverted Sensor Model of a Cartesian 3D Point Feature

- Using the pose-feature compounding operation the expression of the **Inverted Sensor Model** and the its **Jacobians** are **independent of the feature type.**
- What **changes** is the particular value of the **pose-feature compounding** operator**, its Jacobians, the o2s(), s2o()** and **their Jacobians**

Observed in Cartesian

$$\textbf{Observation: } z_k = {}^B x_F + {}^B v_k$$

$${}^B v_k \equiv \mathcal{N}(0, \underbrace{{}^B P_F}_{R_k})$$

> Inverted Sensor Model:

$$\begin{aligned} {}^N x_F &= g({}^N x_B, {}^B x_F, v_k) \\ &= {}^N x_B \boxplus o2s({}^B x_F + {}^B v_k) \end{aligned}$$

> Jacobians:

$$J_{g_x} = J_{2\boxplus}({}^N \hat{x}_B, o2s({}^B x_F)) \cdot J_{o2s}({}^N x_F)$$

$$J_{g_v} = I$$

Observed in Spherical

$$\textbf{Observation: } z_k = c2s({}^B x_F + {}^B v_k)$$

$${}^B v_k \equiv \mathcal{N}(0, \underbrace{{}^B P_F}_{R_k})$$

> Inverted Sensor Model:

$$\begin{aligned} {}^N x_F &= g({}^N x_B, {}^B x_F, v_k) \\ &= {}^N x_B \boxplus o2s({}^B x_F + {}^B v_k) \end{aligned}$$

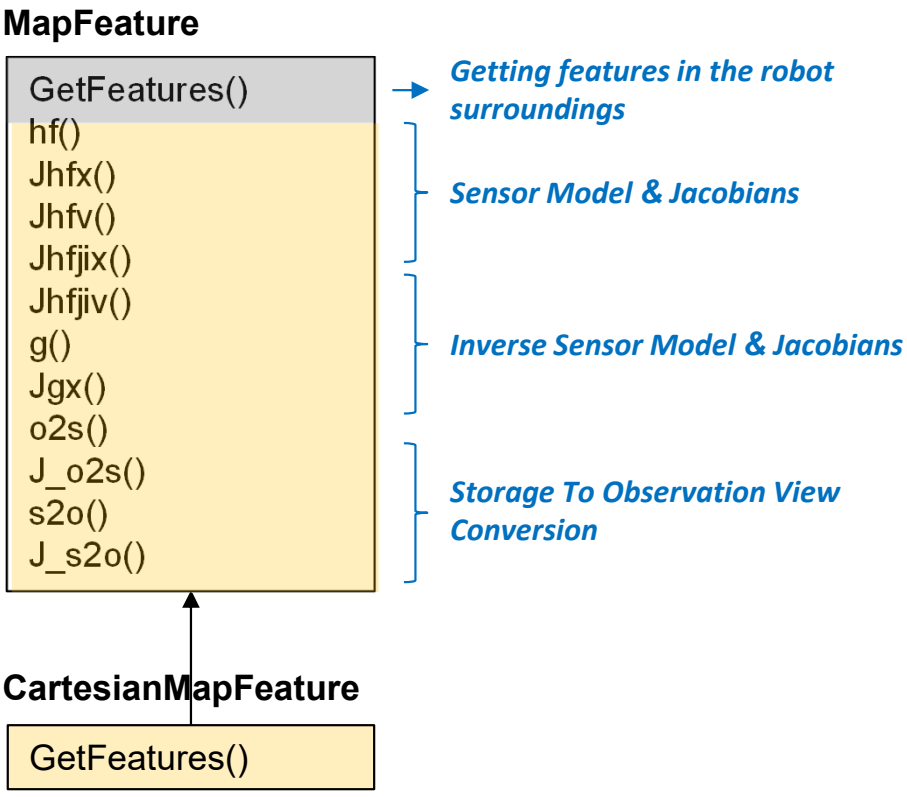
> Jacobians:

$$J_{g_x} = J_{2\boxplus}({}^N \hat{x}_B, o2s({}^B x_F)) \cdot J_{o2s}({}^N x_F)$$

$$J_{g_v} = I$$

3D Cartesian Point Feature Sensor Model

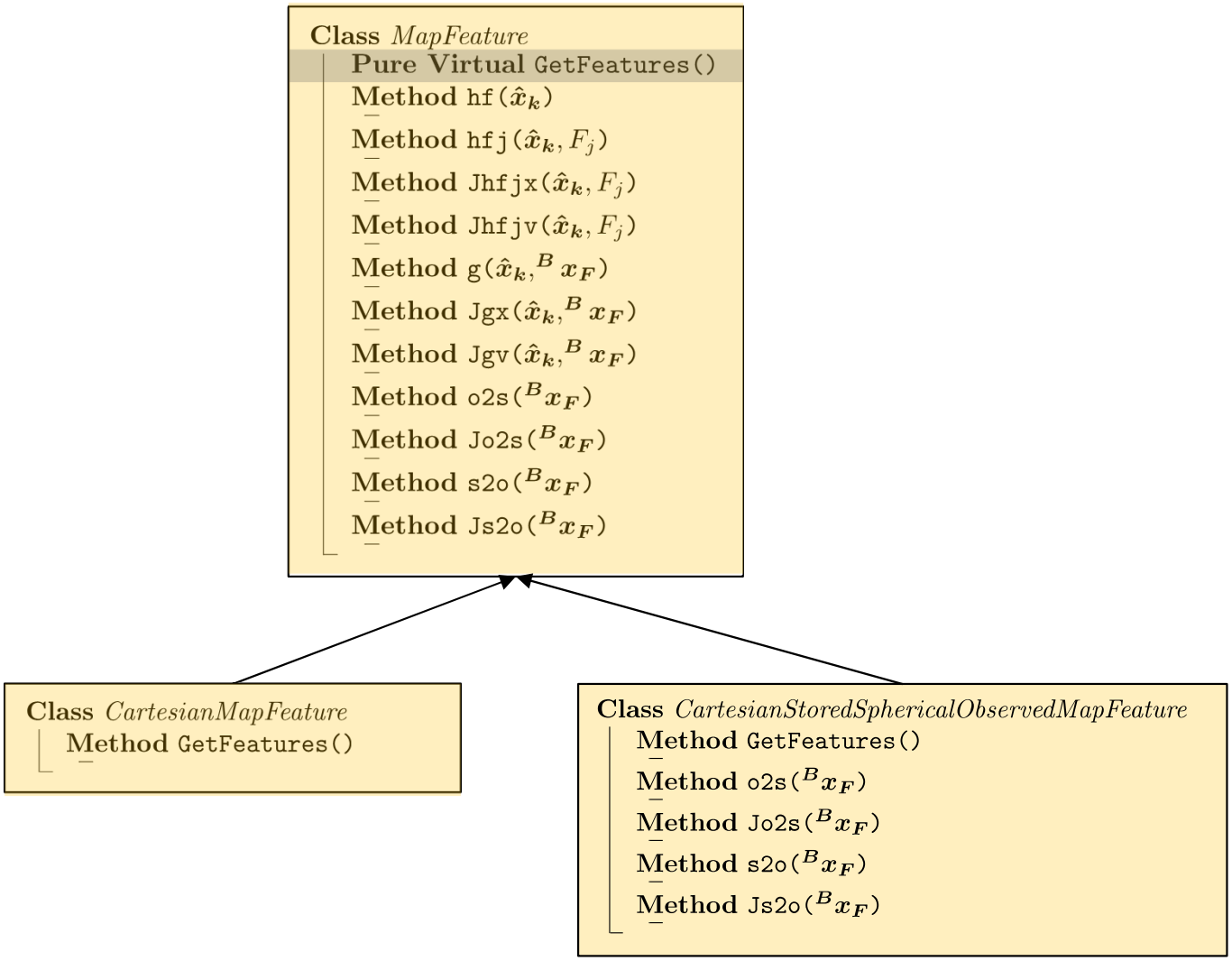
Observed in Cartesian



Class *MapFeature*

- **Pure Virtual** `GetFeatures()` $\rightarrow [\boldsymbol{z_f}, \boldsymbol{R_f}, \boldsymbol{H_f}, \boldsymbol{V_f}]$;
- **Method** `hf(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}$`)`
  - $h = []$; **for** $i = 0$ *to* n_z **do**
    - $F_j = H[i]$;
    - **if** Fj *is not spurius* **then**
      - $h = [h\ hfj(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j)]$;
  - **return** h
- **Method** `hfj(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - ${}^{N}\boldsymbol{x}_{\boldsymbol{F}} = M[F_j]$;
  - **return** $s2o\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\boldsymbol{x}_{\boldsymbol{F}}\right)$
- **Method** `Jhfjx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J_p} = \boldsymbol{J_{s2o}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\boldsymbol{x}_{\boldsymbol{F}}) \cdot \boldsymbol{J}_{1\boxplus}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x}_{\boldsymbol{F}}) \cdot \boldsymbol{J}_{\ominus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p}\ \boldsymbol{J_{np}}]$
- **Method** `Jhfjv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - **return** $[\boldsymbol{I}_{d\times d}]$; `// d:dimension of a single feature observation`
- **Method** `g(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - **return** ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus o2s({}^{B}\boldsymbol{x}_{\boldsymbol{F}})$
- **Method** `Jgx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - $\boldsymbol{J_p} = \boldsymbol{J}_{1\boxplus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x}_{\boldsymbol{F}}))$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p}\ \boldsymbol{J_{np}}]$
- **Method** `Jgv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J} = \boldsymbol{J}_{2\boxplus}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x}_{\boldsymbol{F}})) \cdot \boldsymbol{J_{o2s}}({}^{B}\boldsymbol{x}_{\boldsymbol{F}})$;
  - **return** $\boldsymbol{J}$
- **Method** `o2s(`${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - **return** ${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$
- **Method** `Jo2s(`${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - **return** $\boldsymbol{I}$
- **Method** `s2o(`${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - **return** ${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$
- **Method** `Js2o(`${}^{B}\boldsymbol{x}_{\boldsymbol{F}}$`)`
  - **return** $\boldsymbol{I}$

EKF Map Based Localization



3D Cartesian Point Feature
Sensor Model

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$
$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Sensor Model:

$$\boldsymbol{z_k} = h_{\boldsymbol{x_F}}({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{v_k})$$
$$= s2o\left(\ominus {}^{N}\boldsymbol{x_B} \boxplus {}^{N}\,\boldsymbol{x_F}\right) + {}^{B}\boldsymbol{v_k}$$

> Jacobians:

$$\boldsymbol{H_k} = \boldsymbol{J_{s2o}}\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}\right) \cdot \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$$
$$\boldsymbol{V_k} = \boldsymbol{I}$$

Class *MapFeature*

- **Pure Virtual** `GetFeatures()` $\rightarrow [\boldsymbol{z_f}, \boldsymbol{R_f}, \boldsymbol{H_f}, \boldsymbol{V_f}]$;
- **Method** `hf(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}$`)`
  - $h = []$; **for** $i = 0$ *to* n_z **do**
    - $F_j = H[i]$;
    - **if** Fj *is not spurius* **then**
      - $h = [h \; hfj(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j)]$;
  - **return** h
- **Method** `hfj(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - ${}^{N}\boldsymbol{x_F} = M[F_j]$;
  - **return** $s2o\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}\right)$
- **Method** `Jhfjx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J_p} = \boldsymbol{J_{s2o}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}) \cdot \boldsymbol{J_{1\boxplus}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jhfjv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - **return** $\boldsymbol{I_{d\times d}}$; `// d:dimension of a feature observation`
- **Method** `g(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - **return** ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus o2s({}^{B}\boldsymbol{x_F})$
- **Method** `Jgx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - $\boldsymbol{J_p} = \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F}))$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jgv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}({}^{B}\boldsymbol{x_F})$;
  - **return** $\boldsymbol{J}$
- **Method** `o2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Jo2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$
- **Method** `s2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Js2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$

3D Cartesian Point Feature

Sensor Model

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$

$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$

$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$

$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Sensor Model:

$$\begin{aligned}\boldsymbol{z_k} &= h_{\boldsymbol{x_F}}({}^{\boldsymbol{N}}\boldsymbol{x_B}, {}^{\boldsymbol{B}}\boldsymbol{v_k}) \\ &= s2o\left(\ominus {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right) + {}^{\boldsymbol{B}}\boldsymbol{v_k}\end{aligned}$$

> Jacobians:

$$\boldsymbol{H_k} = \boldsymbol{J_{s2o}}\left(\ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right) \cdot \boldsymbol{J}_{1\boxplus}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{B}}\boldsymbol{x_F}) \cdot \boldsymbol{J}_{\ominus}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$$

$$\boldsymbol{V_k} = \boldsymbol{I}$$

Class *MapFeature*

- **Pure Virtual** `GetFeatures()` $\rightarrow [\boldsymbol{z_f}, \boldsymbol{R_f}, \boldsymbol{H_f}, \boldsymbol{V_f}]$;
- **Method** `hf(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}$`)`
  - $h = []$; **for** $i = 0$ *to* n_z **do**
    - $F_j = H[i]$;
    - **if** Fj *is not spurius* **then**
      - $h = [h \; hfj(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j)]$;
  - **return** h
- **Method** `hfj(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - ${}^{\boldsymbol{N}}\boldsymbol{x_F} = M[F_j]$;
  - **return** $s2o\left(\ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right)$
- **Method** `Jhfjx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J_p} = \boldsymbol{J_{s2o}}(\ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}) \cdot \boldsymbol{J}_{1\boxplus}(\ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{B}}\boldsymbol{x_F}) \cdot \boldsymbol{J}_{\ominus}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jhfjv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - **return** $\boldsymbol{I_{d\times d}}$; `// d:dimension of a feature observation`
- **Method** `g(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{\boldsymbol{B}}\,\boldsymbol{x_F}$`)`
  - ${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - **return** ${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus o2s({}^{\boldsymbol{B}}\boldsymbol{x_F})$
- **Method** `Jgx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{\boldsymbol{B}}\,\boldsymbol{x_F}$`)`
  - $\boldsymbol{J_p} = \boldsymbol{J}_{1\boxplus}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{\boldsymbol{B}}\boldsymbol{x_F}))$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jgv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{\boldsymbol{B}}\,\boldsymbol{x_F}$`)`
  - ${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J} = \boldsymbol{J}_{2\boxplus}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{\boldsymbol{B}}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}({}^{\boldsymbol{B}}\boldsymbol{x_F})$;
  - **return** $\boldsymbol{J}$
- **Method** `o2s(`${}^{\boldsymbol{B}}\boldsymbol{x_F}$`)`
  - **return** ${}^{\boldsymbol{B}}\boldsymbol{x_F}$
- **Method** `Jo2s(`${}^{\boldsymbol{B}}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$
- **Method** `s2o(`${}^{\boldsymbol{B}}\boldsymbol{x_F}$`)`
  - **return** ${}^{\boldsymbol{B}}\boldsymbol{x_F}$
- **Method** `Js2o(`${}^{\boldsymbol{B}}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$

3D Cartesian Point Feature
Inverted Sensor Model

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$
$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Inverted Sensor Model:

$${}^{N}\boldsymbol{x_F} = g({}^{N}\boldsymbol{x_B}, {}^{B}\,\boldsymbol{x_F}, \boldsymbol{v_k})$$
$$= {}^{N}\,\boldsymbol{x_B} \boxplus o2s({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})$$

> Jacobians:

$$\boldsymbol{J_{g_x}} = \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F}))$$
$$\boldsymbol{J_{g_v}} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}\left({}^{N}\boldsymbol{x_F}\right)$$

Class *MapFeature*

- **Pure Virtual** `GetFeatures()` $\rightarrow [\boldsymbol{z_f}, \boldsymbol{R_f}, \boldsymbol{H_f}, \boldsymbol{V_f}]$;
- **Method** `hf(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}$`)`
  - $h = []$; **for** $i = 0$ *to* n_z **do**
    - $F_j = H[i]$;
    - **if** Fj *is not spurius* **then**
      - $h = [h \; hfj(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j)]$;
  - **return** h
- **Method** `hfj(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - ${}^{N}\boldsymbol{x_F} = M[F_j]$;
  - **return** $s2o\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}\right)$
- **Method** `Jhfjx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J_p} = \boldsymbol{J_{s2o}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}) \cdot \boldsymbol{J_{1\boxplus}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jhfjv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - **return** $\boldsymbol{I_{d\times d}}$; `// d:dimension of a feature observation`
- **Method** `g(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - **return** ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus o2s({}^{B}\boldsymbol{x_F})$
- **Method** `Jgx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - $\boldsymbol{J_p} = \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F}))$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jgv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}({}^{B}\boldsymbol{x_F})$;
  - **return** $\boldsymbol{J}$
- **Method** `o2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Jo2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$
- **Method** `s2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Js2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$

3D Cartesian Point Feature
Inverted Sensor Model

Observed in Cartesian

$$\textbf{Pose: } {}^{N}\boldsymbol{x_B} = [{}^{N}x_B \;\; {}^{N}y_B \;\; {}^{N}z_B \;\; {}^{N}\psi_B]^T$$
$${}^{N}\boldsymbol{x_B} \equiv \mathcal{N}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{N}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{B}\boldsymbol{x_F} = \left[{}^{B}x_F \;\; {}^{B}y_F \;\; {}^{B}z_F\right]^T$$
$${}^{B}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{B}\,\boldsymbol{x_F} + {}^{B}\,\boldsymbol{v_k}$$
$${}^{B}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{B}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

> Inverted Sensor Model:

$${}^{N}\boldsymbol{x_F} = g({}^{N}\boldsymbol{x_B}, {}^{B}\,\boldsymbol{x_F}, \boldsymbol{v_k})$$
$$= {}^{N}\,\boldsymbol{x_B} \boxplus o2s({}^{B}\boldsymbol{x_F} + {}^{B}\boldsymbol{v_k})$$

> Jacobians:

$$\boldsymbol{J_{g_x}} = \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F}))$$
$$\boldsymbol{J_{g_v}} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}\left({}^{N}\boldsymbol{x_F}\right)$$

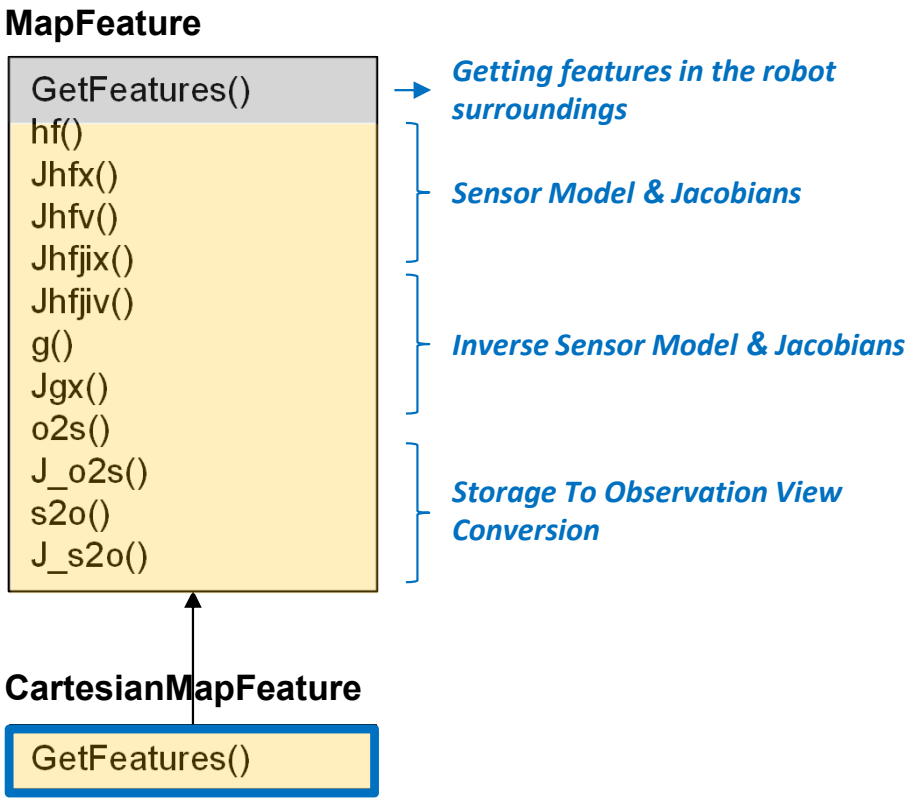
Class *MapFeature*

- **Pure Virtual** `GetFeatures()` $\rightarrow [\boldsymbol{z_f}, \boldsymbol{R_f}, \boldsymbol{H_f}, \boldsymbol{V_f}]$;
- **Method** `hf(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}$`)`
  - $h = []$; **for** $i = 0$ *to* n_z **do**
    - $F_j = H[i]$;
    - **if** Fj *is not spurius* **then**
      - $h = [h \; hfj(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j)]$;
  - **return** h
- **Method** `hfj(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - ${}^{N}\boldsymbol{x_F} = M[F_j]$;
  - **return** $s2o\left(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}\right)$
- **Method** `Jhfjx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J_p} = \boldsymbol{J_{s2o}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{N}\,\boldsymbol{x_F}) \cdot \boldsymbol{J_{1\boxplus}}(\ominus {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{B}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}})$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jhfjv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, F_j$`)`
  - **return** $\boldsymbol{I_{d\times d}}$; `// d:dimension of a feature observation`
- **Method** `g(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - **return** ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus o2s({}^{B}\boldsymbol{x_F})$
- **Method** `Jgx(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - $\boldsymbol{J_p} = \boldsymbol{J_{1\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F}))$;
  - $\boldsymbol{J_{np}} = \boldsymbol{0}$;
  - **return** $[\boldsymbol{J_p} \; \boldsymbol{J_{np}}]$
- **Method** `Jgv(`$\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^{B}\,\boldsymbol{x_F}$`)`
  - ${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}} = GetRobotPose(\hat{\boldsymbol{x}}_{\boldsymbol{k}})$;
  - $\boldsymbol{J} = \boldsymbol{J_{2\boxplus}}({}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, o2s({}^{B}\boldsymbol{x_F})) \cdot \boldsymbol{J_{o2s}}({}^{B}\boldsymbol{x_F})$;
  - **return** $\boldsymbol{J}$
- **Method** `o2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Jo2s(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$
- **Method** `s2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** ${}^{B}\boldsymbol{x_F}$
- **Method** `Js2o(`${}^{B}\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{I}$

3D Cartesian Point Feature

Observed in Cartesian

Implementation



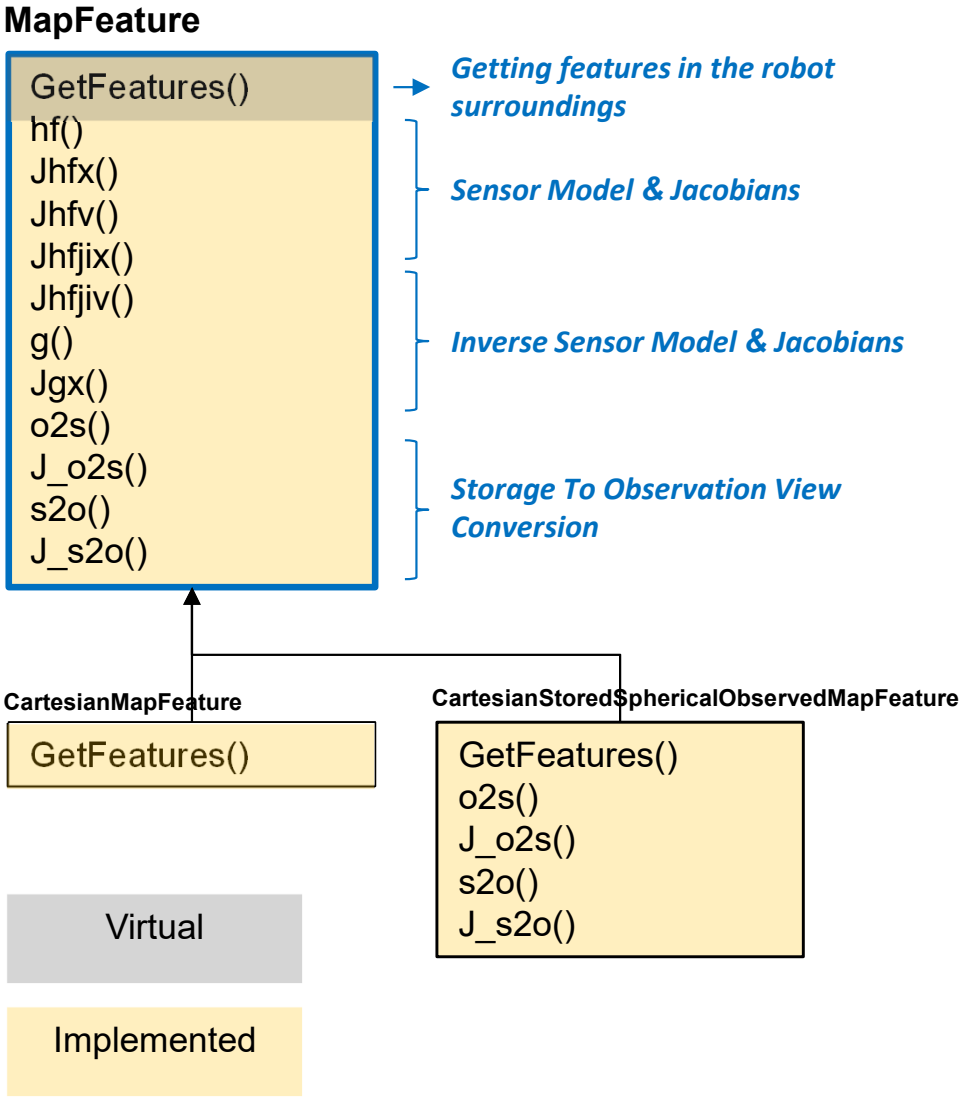
Class *CartesianMapFeature*

- **Method** `GetFeatures()`
  - $[\boldsymbol{z_f}, \boldsymbol{R_f}] = robot.ReadCartesianFeature();$
  - **return** $[\boldsymbol{Z_f}, \boldsymbol{R_f}]$

3D Cartesian Point Feature

Observed in Spherical

Implementation



Class *MapFeature*

Pure Virtual `GetFeatures()` $\rightarrow [z_f, R_f, H_f, V_f]$;

Method `hf`$(\hat{x}_k)$

- $h = []$; **for** $i = 0$ *to* n_z **do**
  - $F_j = H[i]$;
  - **if** Fj *is not spurius* **then**
    - $h = [h\ hfj(\hat{x}_k, F_j)]$;
- **return** h

Method `hfj`$(\hat{x}_k, F_j)$

- ${}^N\hat{x}_B = GetRobotPose(\hat{x}_k)$;
- ${}^N x_F = M[F_j]$;
- **return** $s2o\left(\ominus {}^N\hat{x}_B \boxplus {}^N x_F\right)$

Method `Jhfjx`$(\hat{x}_k, F_j)$

- ${}^N\hat{x}_B = GetRobotPose(\hat{x}_k)$;
- $J_p = J_{s2o}(\ominus {}^N\hat{x}_B \boxplus {}^N x_F) \cdot J_{1\boxplus}(\ominus {}^N\hat{x}_B, {}^B x_F) \cdot J_{\ominus}({}^N\hat{x}_B)$;
- $J_{np} = \mathbf{0}$;
- **return** $[J_p\ J_{np}]$

Method `Jhfjv`$(\hat{x}_k, F_j)$

- **return** $I_{d\times d}$; `// d:dimension of a feature observation`

Method `g`$(\hat{x}_k, {}^B x_F)$

- ${}^N\hat{x}_B = GetRobotPose(\hat{x}_k)$;
- **return** ${}^N\hat{x}_B \boxplus o2s({}^B x_F)$

Method `Jgx`$(\hat{x}_k, {}^B x_F)$

- $J_p = J_{1\boxplus}({}^N\hat{x}_B, o2s({}^B x_F))$;
- $J_{np} = \mathbf{0}$;
- **return** $[J_p\ J_{np}]$

Method `Jgv`$(\hat{x}_k, {}^B x_F)$

- ${}^N\hat{x}_B = GetRobotPose(\hat{x}_k)$;
- $J = J_{2\boxplus}({}^N\hat{x}_B, o2s({}^B x_F)) \cdot J_{o2s}({}^B x_F)$;
- **return** J

Method `o2s`$({}^B x_F)$

- **return** ${}^B x_F$

Method `Jo2s`$({}^B x_F)$

- **return** I

Method `s2o`$({}^B x_F)$

- **return** ${}^B x_F$

Method `Js2o`$({}^B x_F)$

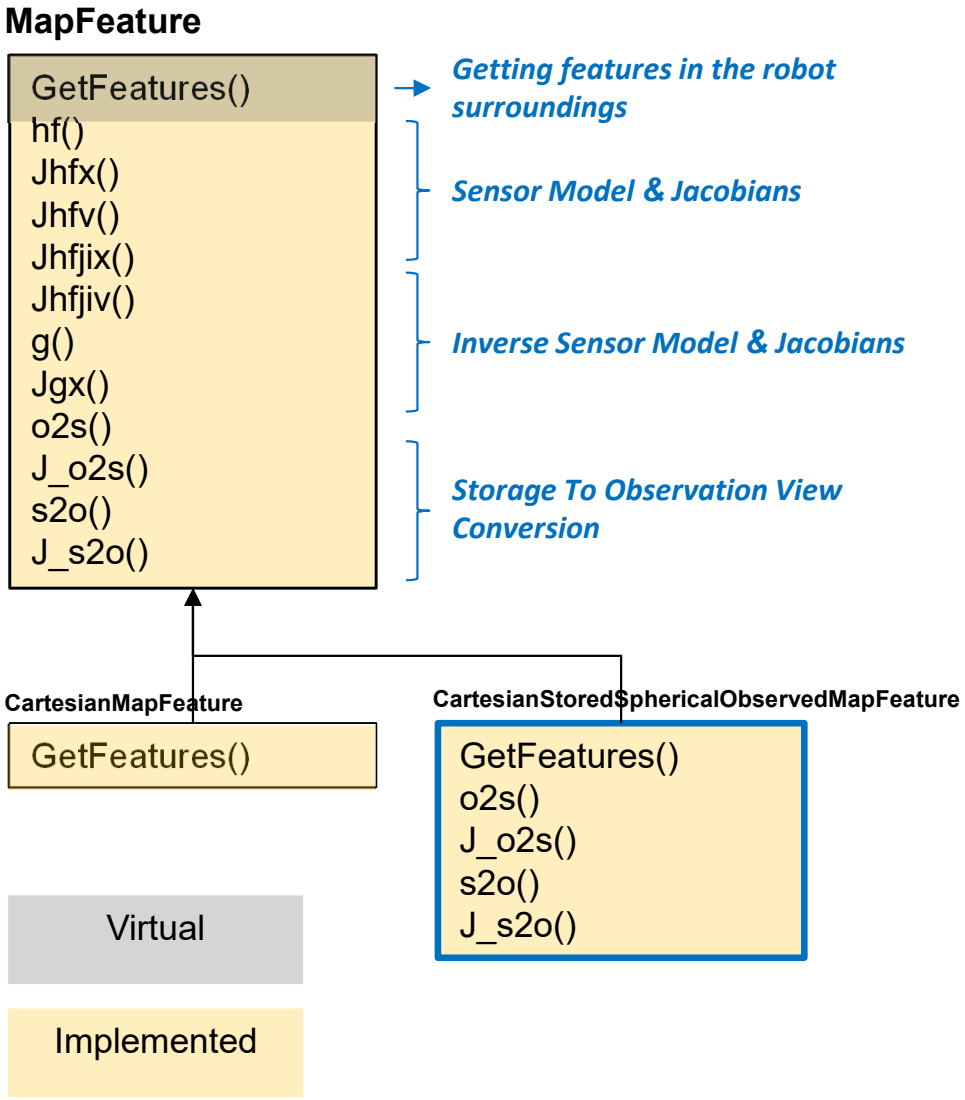
- **return** I

This is the same than in the previous case

3D Cartesian Point Feature

Observed in Spherical

Implementation

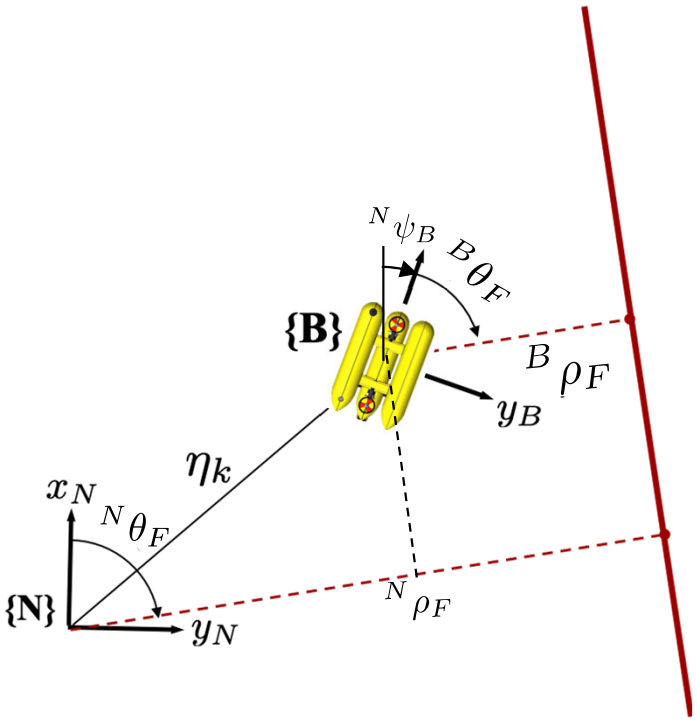


Class *CartesianStoredSphericalObservedMapFeature*

- **Method** `GetFeatures()`
  - $[\boldsymbol{z_f}, \boldsymbol{R_f}] = robot.SphericalFeature();$
  - **return** $[\boldsymbol{z_f}, \boldsymbol{R_f}]$
- **Method** `o2s(`$^B\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{s2c}(^B\boldsymbol{x_F})$
- **Method** `Jo2s(`$^B\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{J_{s2c}}$
- **Method** `s2o(`$^B\boldsymbol{x_F}$`)`
  - **return** $c2s(^B\boldsymbol{x_F})$
- **Method** `Js2o(`$^B\boldsymbol{x_F}$`)`
  - **return** $\boldsymbol{J_{c2s}}$

EKF Map Based Localization

2D Line Feature



> 4DPose-to-2Dline compounding:

$$\textbf{Pose: } {}^{\boldsymbol{N}}\boldsymbol{x_B} = [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N \psi_B]^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_B} \equiv \mathcal{N}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{N}}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{\boldsymbol{B}}\boldsymbol{x_F} = \left[{}^B\rho_F \;\; {}^B\theta_F\right]^T$$

$${}^{\boldsymbol{B}}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k} \;;\; {}^{\boldsymbol{B}}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{\boldsymbol{B}}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k})$$

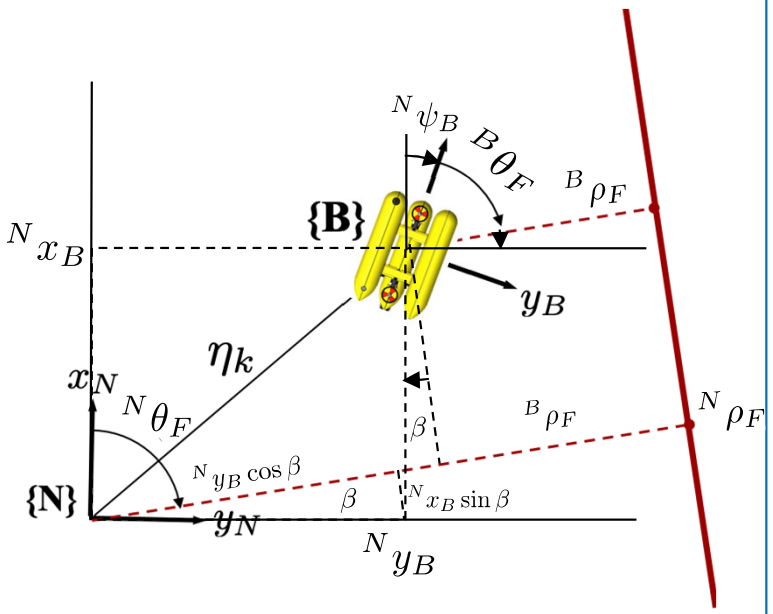
$${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{F}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\boldsymbol{x_F}$$

$${}^{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{I}} = \boldsymbol{J}_{1\boxplus}{}^{\boldsymbol{N}}\boldsymbol{P_B}\boldsymbol{J}_{1\boxplus}^T + \boldsymbol{J}_{2\boxplus}{}^{\boldsymbol{B}}\boldsymbol{P_F}\boldsymbol{J}_{2\boxplus}^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\boldsymbol{x_F} \quad ?$$

EKF Map Based Localization

2D Line Feature



$$\beta = \frac{\pi}{2} - {}^N\psi_B + {}^B\theta_F$$
$$\sin\beta = \cos({}^N\psi_B + {}^B\theta_F)$$
$$\cos\beta = \sin({}^N\psi_B + {}^B\theta_F)$$

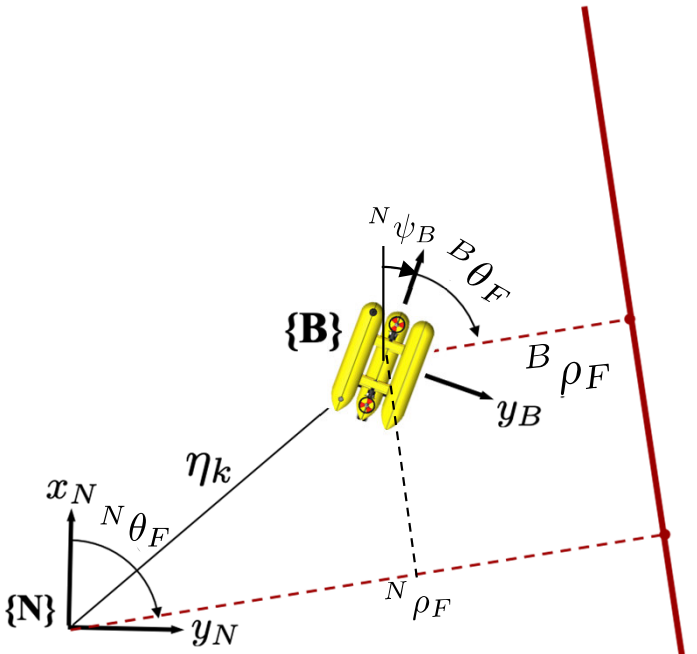
> 4DPose-to-2Dline compounding:

$$\textbf{Pose: } {}^{\boldsymbol{N}}\boldsymbol{x_B} = [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N\psi_B]^T$$
$${}^{\boldsymbol{N}}\boldsymbol{x_B} \equiv \mathcal{N}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{N}}\boldsymbol{P_B})$$
$$\textbf{Storage: } {}^{\boldsymbol{B}}\boldsymbol{x_F} = \left[{}^B\rho_F \;\; {}^B\theta_F\right]^T$$
$${}^{\boldsymbol{B}}\boldsymbol{x_F} \in \mathbb{R}^3$$
$$\textbf{Observation: } \boldsymbol{z_k} = {}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k} \;;\; {}^{\boldsymbol{B}}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{\boldsymbol{B}}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$
$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k})$$
$${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{F}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\boldsymbol{x_F}$$
$${}^{\boldsymbol{N}}\boldsymbol{P_L} = \boldsymbol{J_{1\boxplus}}\,{}^{\boldsymbol{N}}\boldsymbol{P_B}\boldsymbol{J_{1\boxplus}^T} + \boldsymbol{J_{2\boxplus}}\,{}^{\boldsymbol{B}}\boldsymbol{P_F}\boldsymbol{J_{2\boxplus}^T}$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\boldsymbol{x_F} = \begin{bmatrix} {}^B\rho_F + {}^N\hat{x}_B\cos({}^N\hat{\psi}_B + {}^B\theta_F) + {}^N\hat{y}_B\sin({}^N\hat{\psi}_B + {}^B\theta_F) \\ {}^N\hat{\psi}_B + {}^B\theta_F \end{bmatrix}$$

EKF Map Based Localization

2D Polar Line Feature



> 4DPose-to-2Dline compounding:

$$\textbf{Pose: } {}^{\boldsymbol{N}}\boldsymbol{x_B} = [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N\psi_B]^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_B} \equiv \mathcal{N}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{N}}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{\boldsymbol{B}}\boldsymbol{x_F} = \left[{}^B\rho_F \;\; {}^B\theta_F\right]^T$$

$${}^{\boldsymbol{B}}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{\boldsymbol{B}}\,\boldsymbol{x_F} + {}^{\boldsymbol{B}}\,\boldsymbol{v_k}\;;\; {}^{\boldsymbol{B}}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{\boldsymbol{B}}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\,\boldsymbol{v_k})$$

$${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{F}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\,\boldsymbol{x_F}$$

$${}^{\boldsymbol{N}}\boldsymbol{P_L} = \boldsymbol{J_{1\boxplus}}\,{}^{\boldsymbol{N}}\boldsymbol{P_B}\boldsymbol{J_{1\boxplus}}^T + \boldsymbol{J_{2\boxplus}}\,{}^{\boldsymbol{B}}\boldsymbol{P_F}\boldsymbol{J_{2\boxplus}}^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\,\boldsymbol{x_F} = \begin{bmatrix} {}^B\rho_F + {}^N\hat{x}_B\cos({}^N\hat{\psi}_B + {}^B\,\theta_F) + {}^N\hat{y}_B\sin({}^N\hat{\psi}_B + {}^B\,\theta_F) \\ {}^N\hat{\psi}_B + {}^B\,\theta_F \end{bmatrix}$$

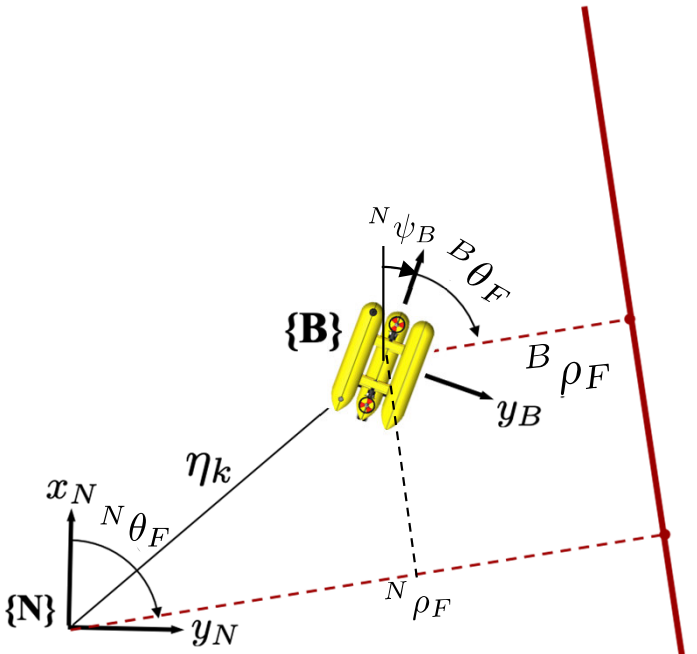
$$\boldsymbol{J_{1\boxplus}} = \left.\frac{\partial {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k})}{\partial {}^{\boldsymbol{N}}\boldsymbol{x_B}}\right|_{\substack{{}^{\boldsymbol{N}}\boldsymbol{x_B} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \\ {}^{\boldsymbol{B}}\boldsymbol{v_k} = \boldsymbol{0}}}$$

$$= \left.\frac{\partial \begin{bmatrix} {}^B\rho_F + {}^B v_\rho + {}^N x_B\cos({}^N\psi_B + {}^B\,\theta_F + {}^B v_\theta) + {}^N y_B\sin({}^N\psi_B + {}^B\,\theta_F + {}^B\,v_\theta) \\ {}^N\psi_B + {}^B\,\theta_F + {}^B v_\theta \end{bmatrix}}{\partial [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N\psi_B]}\right|_{\substack{{}^{\boldsymbol{N}}\boldsymbol{x_B} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \\ {}^{\boldsymbol{B}}\boldsymbol{v_k} = \boldsymbol{0}}}$$

$$= \begin{bmatrix} \cos({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) & \sin({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) & 0 & -{}^N\hat{x}_B\sin({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) + {}^N\hat{y}_B\cos({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

EKF Map Based Localization

2D Polar Line Feature



> 4DPose-to-2Dline compounding:

$$\textbf{Pose: } {}^{\boldsymbol{N}}\boldsymbol{x_B} = [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N\psi_B]^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_B} \equiv \mathcal{N}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{N}}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{\boldsymbol{B}}\boldsymbol{x_F} = \left[{}^B\rho_F \;\; {}^B\theta_F\right]^T$$

$${}^{\boldsymbol{B}}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{\boldsymbol{B}}\,\boldsymbol{x_F} + {}^{\boldsymbol{B}}\,\boldsymbol{v_k}\;;\;\; {}^{\boldsymbol{B}}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{\boldsymbol{B}}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\,\boldsymbol{v_k})$$

$${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{F}} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\,\boldsymbol{x_F}$$

$${}^{\boldsymbol{N}}\boldsymbol{P_L} = \boldsymbol{J_{1\boxplus}}\,{}^{\boldsymbol{N}}\boldsymbol{P_B}\boldsymbol{J_{1\boxplus}^T} + \boldsymbol{J_{2\boxplus}}\,{}^{\boldsymbol{B}}\boldsymbol{P_F}\boldsymbol{J_{2\boxplus}^T}$$

$${}^{\boldsymbol{N}}\boldsymbol{x_F} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{B}}\,\boldsymbol{x_F} = \begin{bmatrix} {}^B\rho_F + {}^N\hat{x}_B\cos({}^N\hat{\psi}_B + {}^B\,\theta_F) + {}^N\hat{y}_B\sin({}^N\hat{\psi}_B + {}^B\,\theta_F) \\ {}^N\hat{\psi}_B + {}^B\,\theta_F \end{bmatrix}$$

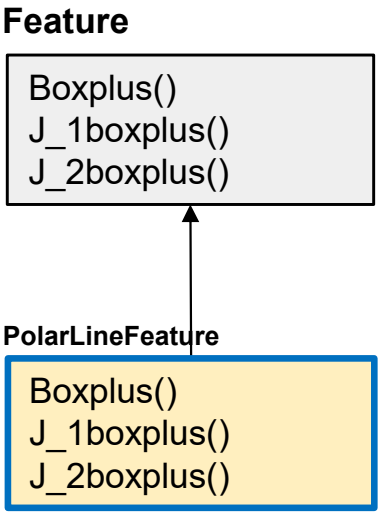
$$\boldsymbol{J_{2\boxplus}} = \left.\frac{\partial {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus ({}^{\boldsymbol{B}}\boldsymbol{x_F} + {}^{\boldsymbol{B}}\boldsymbol{v_k})}{\partial {}^{\boldsymbol{B}}\boldsymbol{v_k}}\right|_{\substack{{}^{\boldsymbol{N}}\boldsymbol{x_B} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_B \\ {}^{\boldsymbol{B}}\boldsymbol{v_k} = \boldsymbol{0}}}$$

$$= \left.\frac{\partial \begin{bmatrix} {}^B\rho_F + {}^Bv_\rho + {}^N x_B\cos({}^N\psi_B + {}^B\,\theta_F + {}^Bv_\theta) + {}^N y_B\sin({}^N\psi_B + {}^B\,\theta_F + {}^B\,v_\theta) \\ {}^N\psi_B + {}^B\,\theta_F + {}^Bv_\theta \end{bmatrix}}{\partial [{}^Bv_\rho \;\; {}^Bv_\theta]}\right|_{\substack{{}^{\boldsymbol{N}}\boldsymbol{x_B} = {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_B \\ {}^{\boldsymbol{B}}\boldsymbol{v_k} = \boldsymbol{0}}}$$

$$= \begin{bmatrix} 1 & -{}^N\hat{x}_B\sin({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) + {}^N\hat{y}_B\cos({}^N\hat{\psi}_B + {}^B\,\hat{\theta}_F) \\ 0 & 1 \end{bmatrix}$$

EKF Map Based Localization

2D Polar Line Feature: Implementation



Class *PolarLineFeature:Feature*

Method `boxplus`$({}^{N}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_F})$

$[{}^{B}\hat{\rho}_F, {}^{B}\hat{\theta}_F] = {}^{B}\boldsymbol{x_F}$

$${}^{N}\boldsymbol{x_F} = \begin{bmatrix} {}^{B}\hat{\rho}_F + {}^{N}x_B \cos({}^{N}\hat{\psi}_B + {}^{B}\theta_F) + {}^{N}\hat{y}_B \sin({}^{N}\hat{\psi}_B + {}^{B}\theta_F) \\ {}^{N}\hat{\psi}_B + {}^{B}\theta_F \end{bmatrix};$$

return ${}^{N}\boldsymbol{x_F}$

Method `J_1boxplus`$({}^{A}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_C})$

$$\boldsymbol{J_{1\boxplus}} = \begin{bmatrix} \cos({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) & \sin({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) & 0 & -{}^{N}\hat{x}_B \sin({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) + {}^{N}\hat{y}_B \cos({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

return $\boldsymbol{J_{1\boxplus}}$

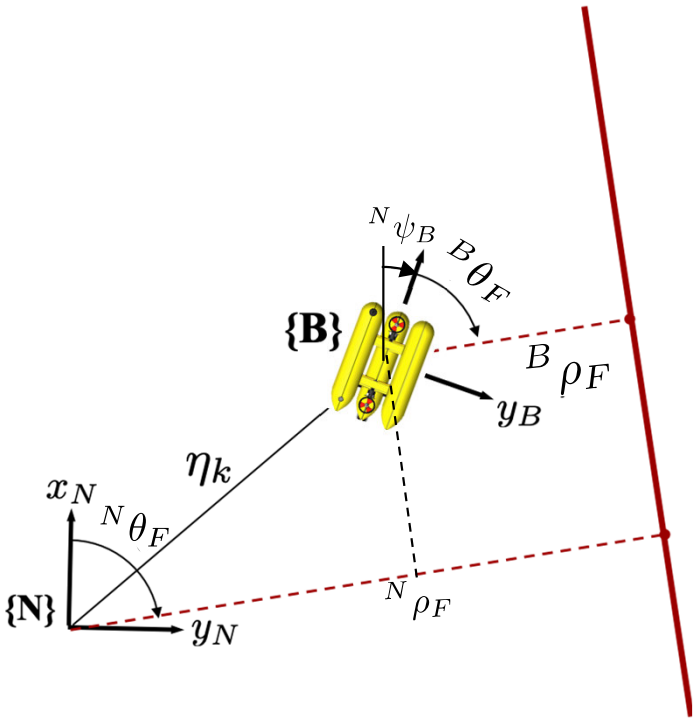
Method `J_2boxplus`$({}^{A}\boldsymbol{x_B}, {}^{B}\boldsymbol{x_C})$

$$\boldsymbol{J_{2\boxplus}} = \begin{bmatrix} 1 & -{}^{N}\hat{x}_B \sin({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) + {}^{N}\hat{y}_B \cos({}^{N}\hat{\psi}_B + {}^{B}\hat{\theta}_F) \\ 0 & 1 \end{bmatrix};$$

return $\boldsymbol{J_{2\boxplus}}$

EKF Map Based Localization

2D Line Feature: Observation Equation



> Sensor Model:

$$\textbf{Pose: } {}^{\boldsymbol{N}}\boldsymbol{x_B} = [{}^N x_B \;\; {}^N y_B \;\; {}^N z_B \;\; {}^N \psi_B]^T$$

$${}^{\boldsymbol{N}}\boldsymbol{x_B} \equiv \mathcal{N}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{N}}\boldsymbol{P_B})$$

$$\textbf{Storage: } {}^{\boldsymbol{B}}\boldsymbol{x_F} = \left[{}^B\rho_F \;\; {}^B\theta_F\right]^T$$

$${}^{\boldsymbol{B}}\boldsymbol{x_F} \in \mathbb{R}^3$$

$$\textbf{Observation: } \boldsymbol{z_k} = {}^{\boldsymbol{B}}\,\boldsymbol{x_F} + {}^{\boldsymbol{B}}\,\boldsymbol{v_k} \;;\; {}^{\boldsymbol{B}}\boldsymbol{v_k} \equiv \mathcal{N}(\boldsymbol{0}, \underbrace{{}^{\boldsymbol{B}}\boldsymbol{P_F}}_{\boldsymbol{R_k}})$$

$$\begin{aligned} {}^{\boldsymbol{B}}\boldsymbol{z_k} &= h_{\boldsymbol{x_F}}({}^{\boldsymbol{N}}\boldsymbol{x_B}, {}^{\boldsymbol{N}}\,\boldsymbol{x_F}, {}^{\boldsymbol{B}}\boldsymbol{v_k}) \\ &= s2o\left(\ominus {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right) + {}^{\boldsymbol{B}}\boldsymbol{v_k} \end{aligned}$$

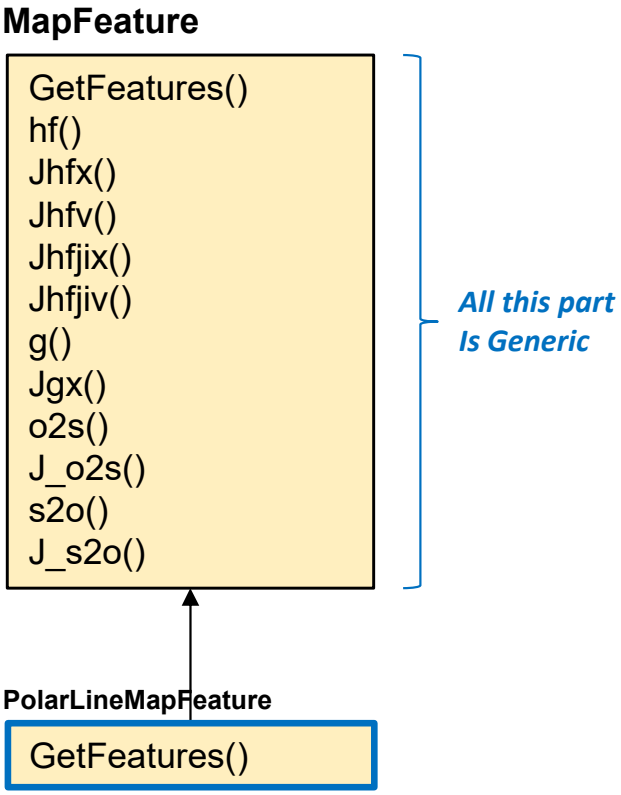
> Jacobians:

$$\begin{aligned} \boldsymbol{H_k} &= \frac{\partial h_{\boldsymbol{x_F}}({}^{\boldsymbol{N}}\boldsymbol{x_B}, {}^{\boldsymbol{B}}\boldsymbol{v_k})}{\partial {}^{\boldsymbol{N}}\boldsymbol{x_B}} = \frac{\partial s2o\left(\ominus {}^{\boldsymbol{N}}\boldsymbol{x_B} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right) + {}^{\boldsymbol{B}}\boldsymbol{v_k}}{\partial {}^{\boldsymbol{N}}\boldsymbol{x_B}} \\ &= \boldsymbol{J_{1\boxplus}}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}}, {}^{\boldsymbol{B}}\boldsymbol{x_F}) \cdot \boldsymbol{J_{\ominus}}({}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}})\boldsymbol{J_{s2o}}\left(\ominus {}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{B}} \boxplus {}^{\boldsymbol{N}}\,\boldsymbol{x_F}\right) \end{aligned}$$

$$\boldsymbol{V_k} = \frac{\partial h_{\boldsymbol{x_F}}({}^{\boldsymbol{N}}\boldsymbol{x_B}, {}^{\boldsymbol{B}}\boldsymbol{v_k})}{\partial {}^{\boldsymbol{B}}\boldsymbol{v_k}} = \boldsymbol{I}$$

EKF Map Based Localization

2D Line Feature: Implementation



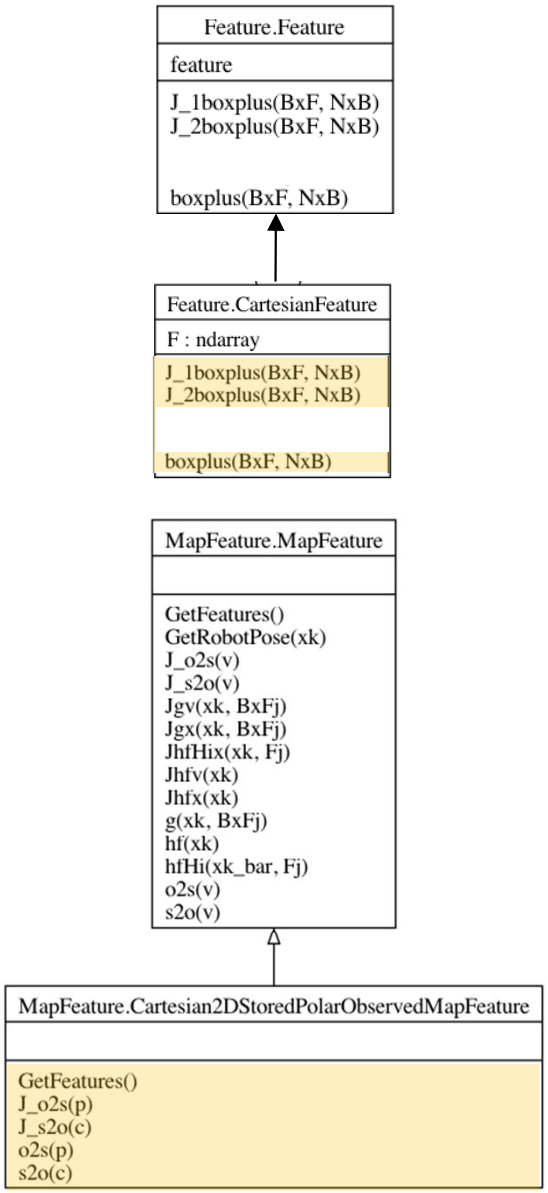
Class *PolarLineMapFeature*

  Method `GetFeatures()`

    $[\boldsymbol{z_f}, \boldsymbol{R_f}] = robot.GetPolarLineFeature();$

    return $[\boldsymbol{z_f}, \boldsymbol{R_f}]$

EKF Map Based Localization



How to add a New Type of Feature
(i.e. Cartesian 2D Feature observed in Polar)

1 Define the **Feature Storage**: ${}^{N}\boldsymbol{x}_{\boldsymbol{F}_i} = [{}^{N}x \quad {}^{N}y]^T$

2 Define its **Observation**: $\boldsymbol{z}_k = {}^{B}\boldsymbol{x}_{\boldsymbol{F}_i} = [{}^{B}\rho_F \quad {}^{B}\theta_F]^T$

3 Define its **pose-feature compounding**:

$$ {}^{N}\boldsymbol{x}_{\boldsymbol{F}_i} = {}^{N}\boldsymbol{x}_{\boldsymbol{B}_k} \boxplus ({}^{B}\boldsymbol{x}_{\boldsymbol{F}_i} + {}^{B}\boldsymbol{v}_k) $$

4 Define the pose-feature **compounding Jacobians**:

$$ \boldsymbol{J}_{1\boxplus} = \left. \frac{\partial {}^{N}\boldsymbol{x}_{\boldsymbol{B}_k} \boxplus ({}^{B}\boldsymbol{x}_{\boldsymbol{F}_i} + {}^{B}\boldsymbol{v}_k)}{\partial {}^{N}\boldsymbol{x}_{\boldsymbol{B}_k}} \right|_{\substack{{}^{N}\boldsymbol{x}_{\boldsymbol{B}_k} = {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}_k} \\ {}^{B}\boldsymbol{v}_k = \boldsymbol{0}}} $$

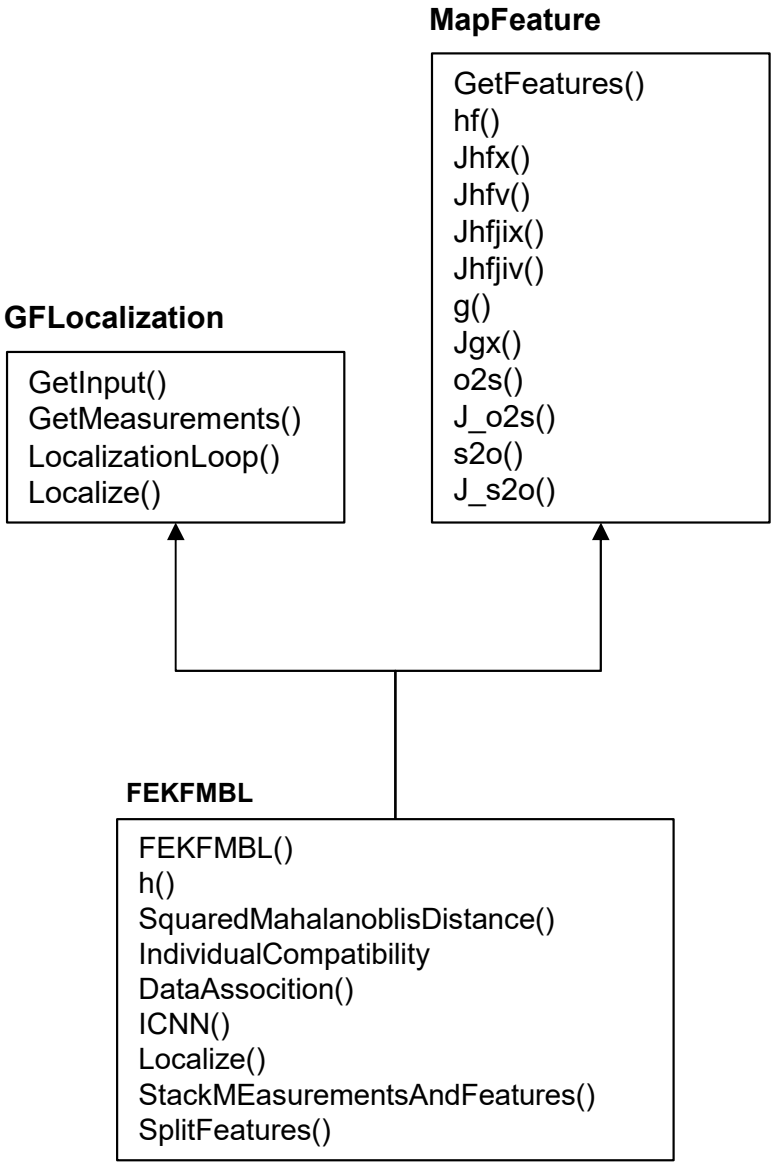
$$ \boldsymbol{J}_{2\boxplus} = \left. \frac{\partial {}^{N}\boldsymbol{x}_{\boldsymbol{B}_k} \boxplus ({}^{B}\boldsymbol{x}_{\boldsymbol{F}_i} + {}^{B}\boldsymbol{v}_k)}{\partial {}^{B}\boldsymbol{v}_k} \right|_{\substack{{}^{N}\boldsymbol{x}_{\boldsymbol{B}_k} = {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B}_k} \\ {}^{B}\boldsymbol{v}_k = \boldsymbol{0}}} $$

5 Program the **Cartesian2DStoredPolarObservedMap Feature class** inheriting from the **MapFeature class**

- Override the **o2s() ,J_o2s() ,s2o() ,J_s2o()** methods to call **p2c() ,J_p2c() ,c2p() ,J_c2p()**
- Override **GetFeatures** to read the Features from the robot

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map

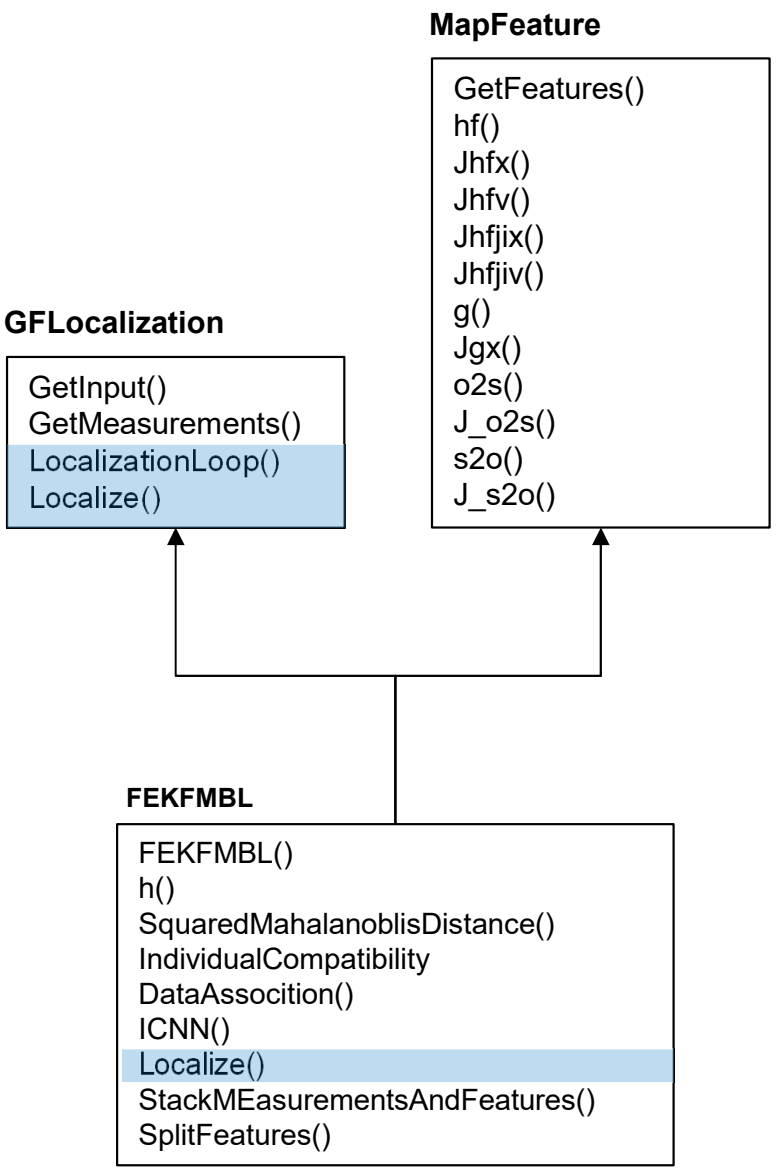


Class *FEKFMBL*

- **Method** `FEKFMBL`$(xBpose\_dim, xB\_dim, xF\_dim, zfi\_dim, M, \alpha,)$
- **Method** `h`$(\boldsymbol{x_k})$
- **Method** `SquaredMahalanobisDistance`$(\boldsymbol{h_{f_j}}, \boldsymbol{P_{f_j}}, \boldsymbol{z_{f_i}}, \boldsymbol{R_{f_i}})$
- **Method** `IndividualCompatibility`$(\mathcal{D}^2_{ij}, dof, \alpha)$
- **Method** `DataAssociation`$(\hat{\bar{\boldsymbol{x}}}_k, \bar{\boldsymbol{P}}_k, \boldsymbol{z_f}, \boldsymbol{R_f})$
- **Method** `ICNN`$(\boldsymbol{h_f}, \boldsymbol{P_{h_f}}, \boldsymbol{z_f}, \boldsymbol{R_f})$
- **Method** `Localize`$(\boldsymbol{x_{k-1}}, \boldsymbol{u_k})$
- **Method** `StackMeasurementAndFeatures`$(\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$
- **Method** `SplitFeatures`$(\boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map



Initialization

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0];$

for $k = 1$ *to* ∞ **do**

$[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput();$

$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k);$

$[z_m, R_m, H_m, V_m] = GetMeasurements();$

$[z_f, R_f, H_f, V_f] = GetFeatures();$

$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f);$

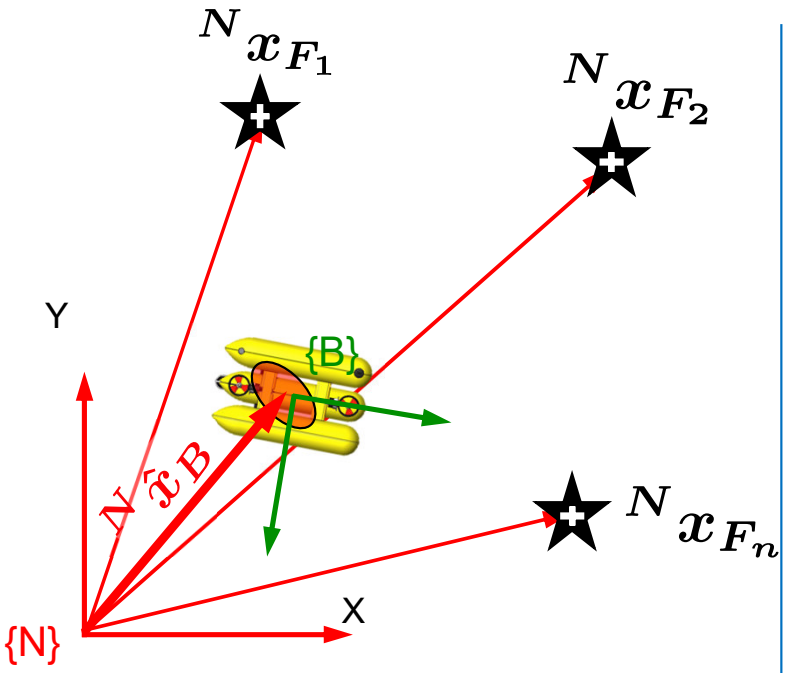
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
$StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p);$

$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k);$

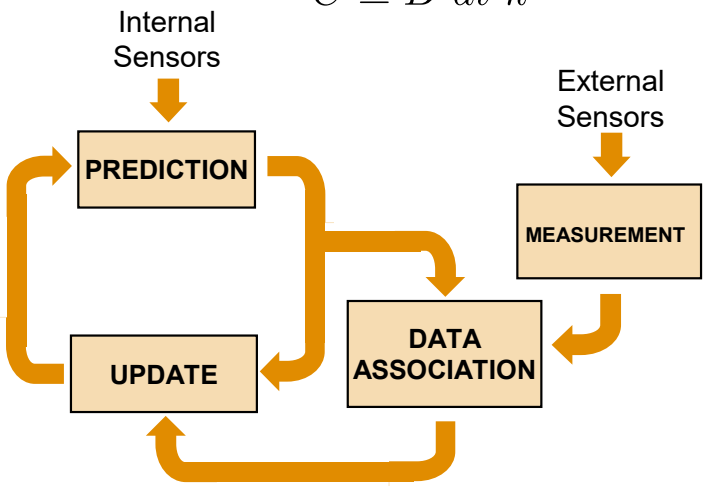
return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map



$B \equiv B \text{ at } k-1$
$C \equiv B \text{ at } k$



Initialization

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0];$
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

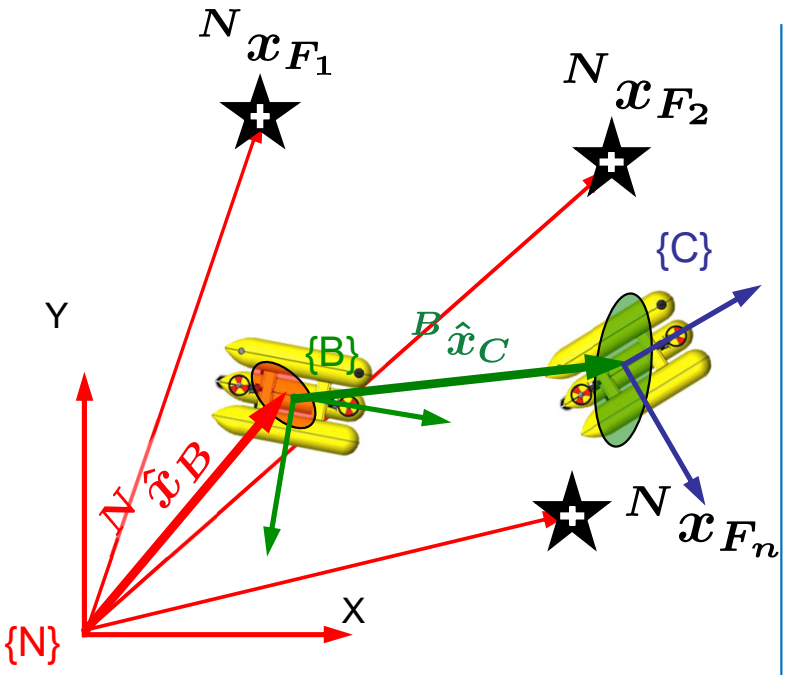
Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput();$
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k);$
$[z_m, R_m, H_m, V_m] = GetMeasurements();$
$[z_f, R_f, H_f, V_f] = GetFeatures();$
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f);$
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p);$
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k);$

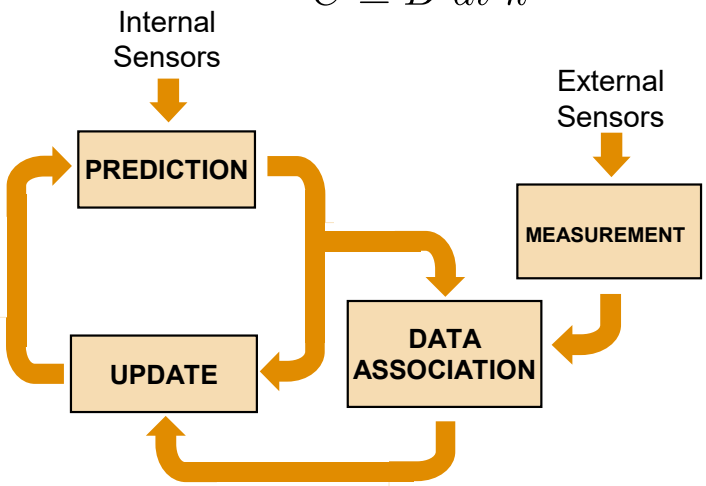
return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map



$B \equiv B \ at \ k-1$
$C \equiv B \ at \ k$



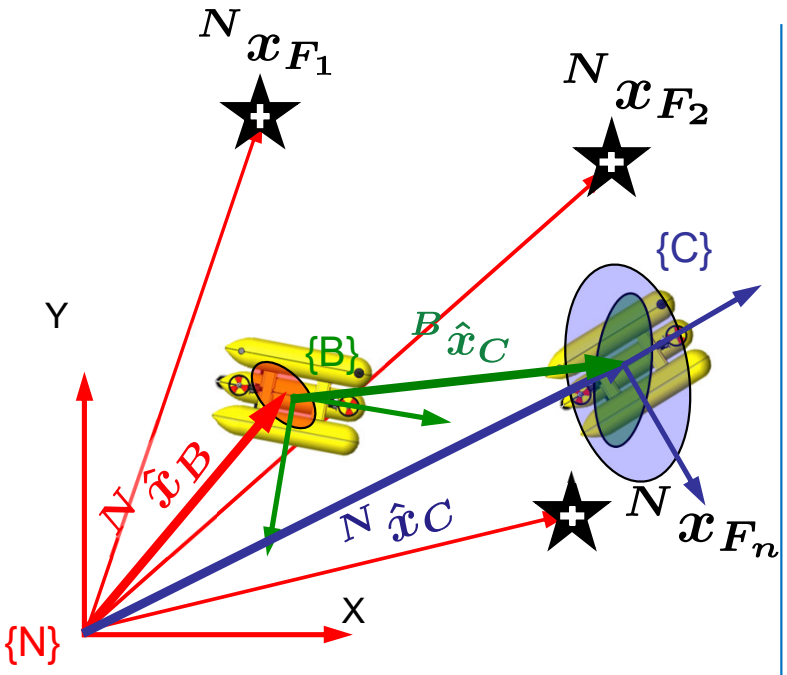
Get Input

Method `LocalizationLoop`$(\hat{x}_0, P_0)$
  $[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
  for $k = 1 \ to \ \infty$ **do**
    $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

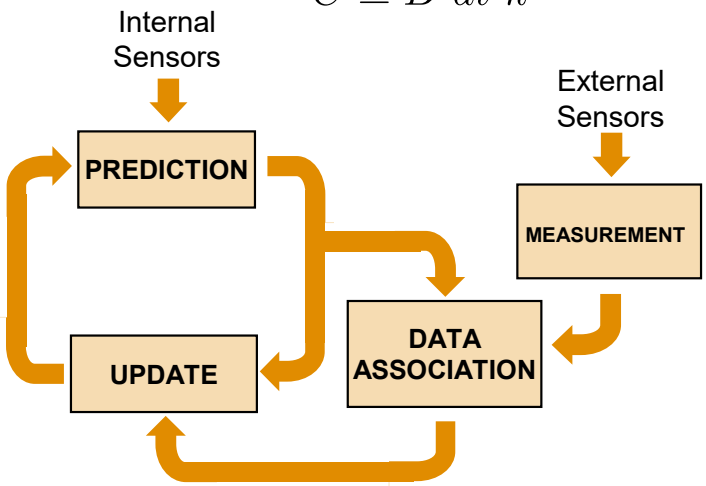
Method `Localize`(x_{k-1}, u_k)
  $[uk, Qk] = GetInput()$;
  $[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
  $[z_m, R_m, H_m, V_m] = GetMeasurements()$;
  $[z_f, R_f, H_f, V_f] = GetFeatures()$;
  $\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
  $[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
    $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
  $[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;
return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map



$B \equiv B \text{ at } k-1$
$C \equiv B \text{ at } k$



Prediction

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

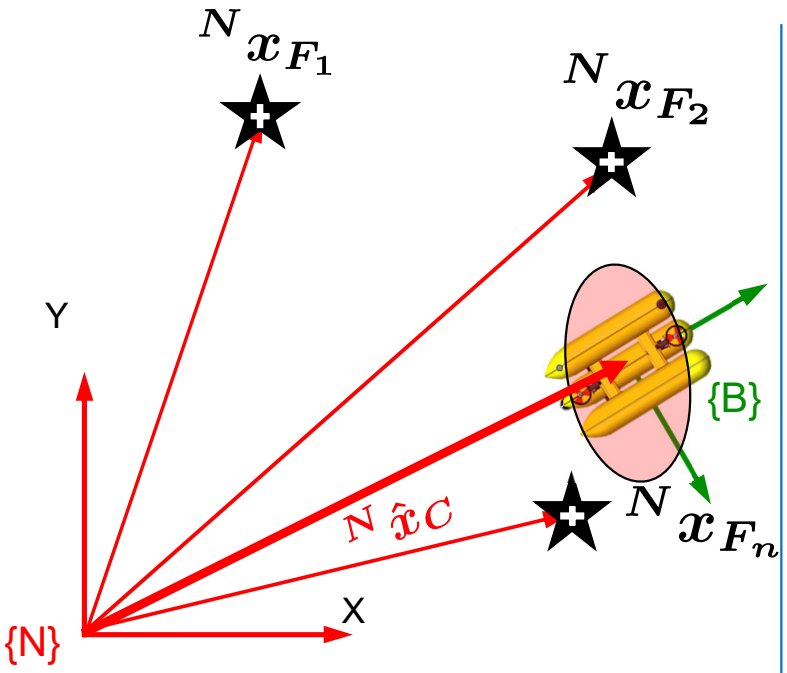
Method `Localize`(x_{k-1}, u_k)

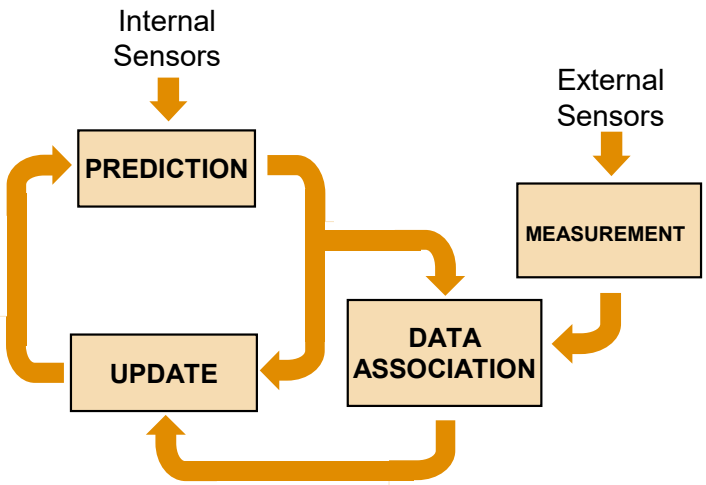
$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map





Get Features

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

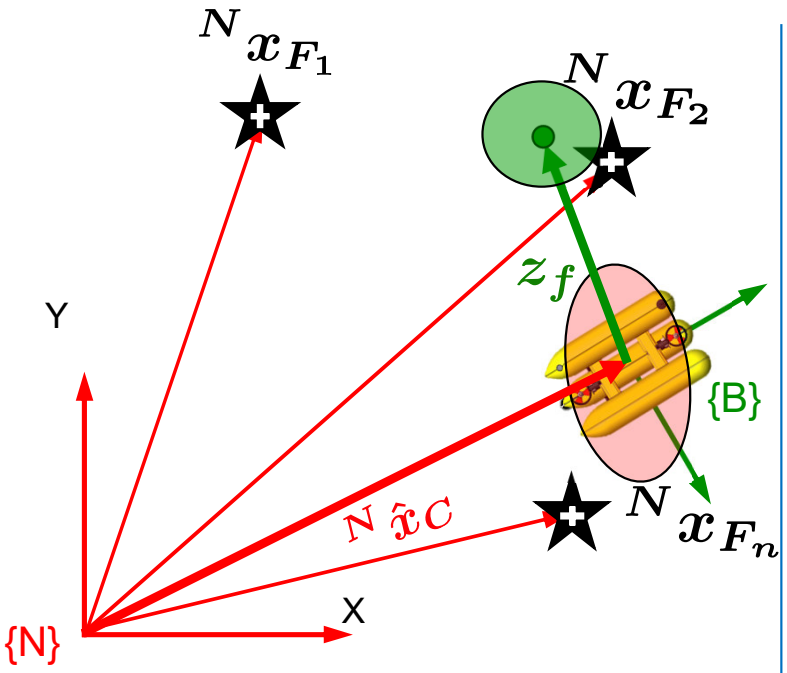
Method `Localize`(x_{k-1}, u_k)

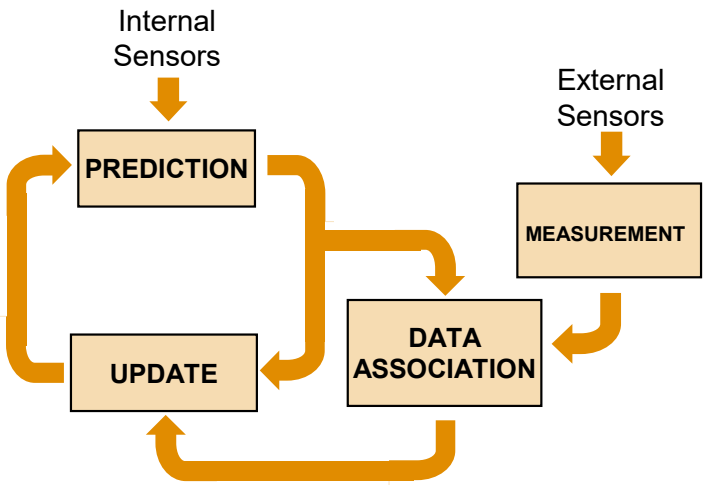
$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map





Get Features

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
$[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

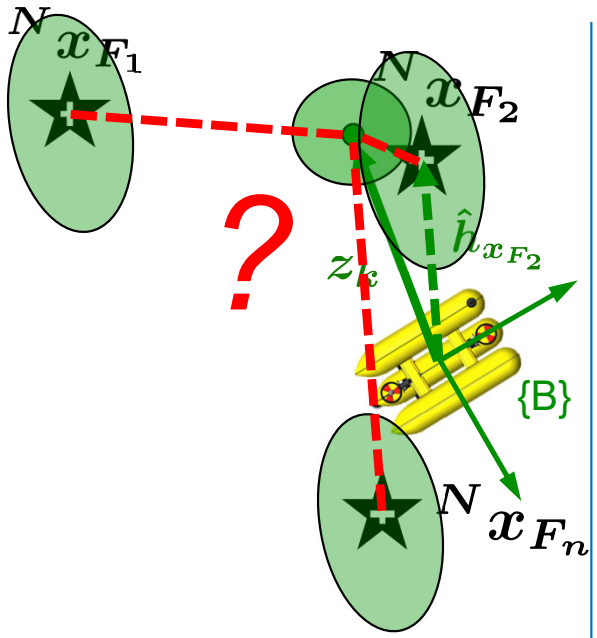
Method `Localize`(x_{k-1}, u_k)

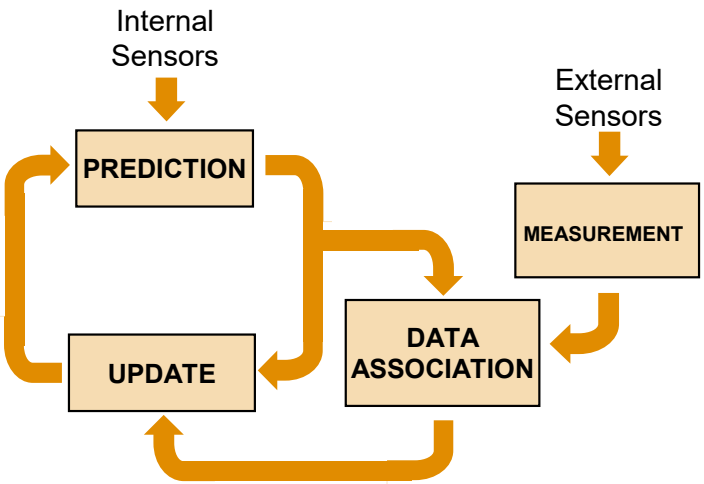
$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
$StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map





Data Association

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput()$;
$[\bar{\hat{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\bar{\hat{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

$z_f = [z_1]$
$\mathcal{H}p = [F_2]$

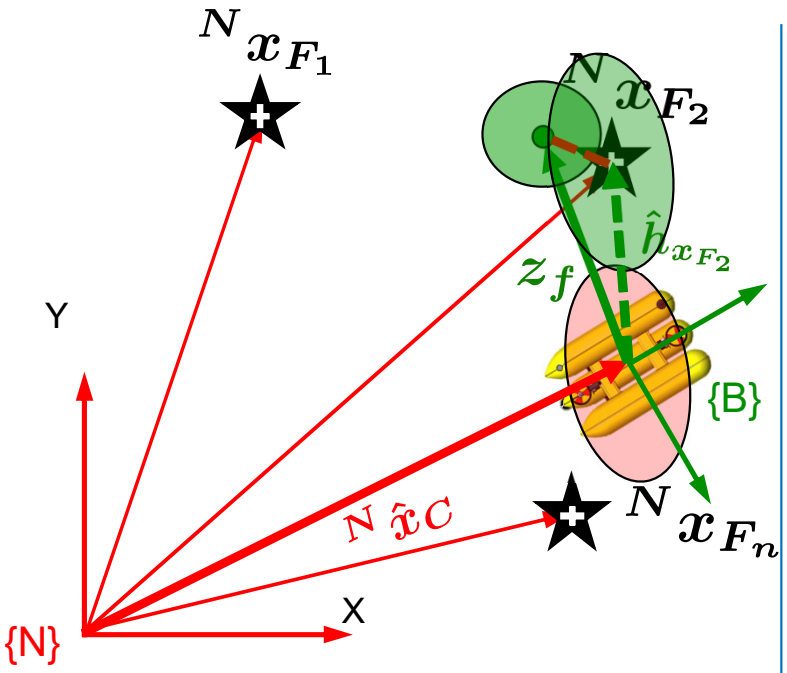
Z_1 is an observation of F_2

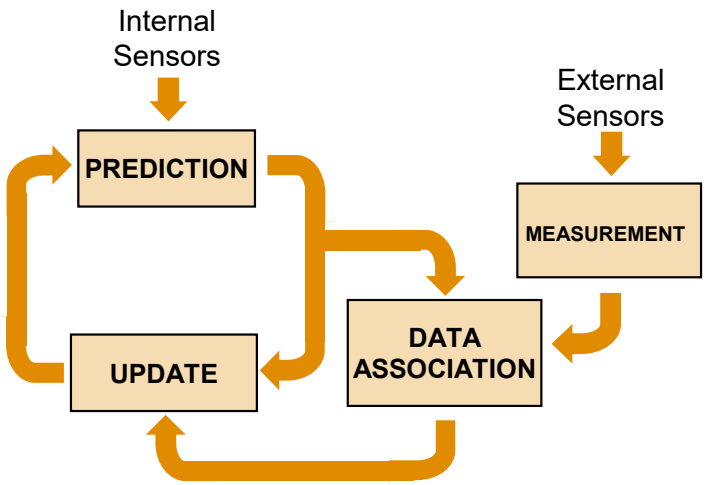
Method `DataAssociation`$(\bar{\hat{x}}_k, \bar{P}_k, z_f, R_f)$

$\mathcal{H}_p = ICNN(\bar{\hat{x}}_k, \bar{P}_k, z_f, R_f)$;
return $\mathcal{H}_p$;

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map





Paired Observations

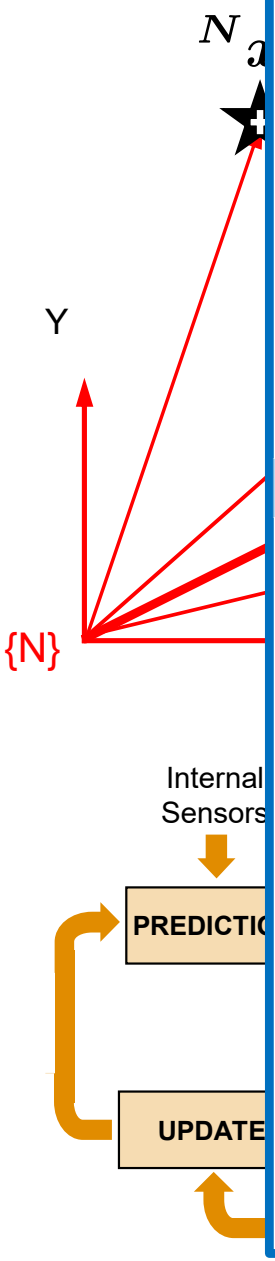
Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
$StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$



Method `StackMeasurementAndFeatures`$(\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$

$[\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}] = SplitFeatures(\boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$;

if *empty* $\boldsymbol{z_m}$ **then**

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = [\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}]$;

else if *empty* $\boldsymbol{z_p}$ **then**

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = [\boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{H_m}, \boldsymbol{V_m}]$;

else

$$\boldsymbol{z_k} = \begin{bmatrix} \boldsymbol{z_m} \\ \boldsymbol{z_p} \end{bmatrix};\ \boldsymbol{R_k} = \begin{bmatrix} \boldsymbol{R_m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_p} \end{bmatrix};$$

$$\boldsymbol{H_k} = \begin{bmatrix} \boldsymbol{H_m} \\ \boldsymbol{H_p} \end{bmatrix};\ \boldsymbol{V_k} = \begin{bmatrix} \boldsymbol{V_m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V_p} \end{bmatrix};$$

return $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}]$

Method `SplitFeatures`$(\boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$

$\boldsymbol{H_p} = \boldsymbol{V_p} = \boldsymbol{z_p} = \boldsymbol{R_p} = \boldsymbol{z_{np}} = \boldsymbol{R_{np}} = []$;

for $i = 1$ *to* $length(\mathcal{H}_p)$ **do** `// For all candidate associations`

$F_j = \mathcal{H}_p[i]$; `//` $\boldsymbol{z_{f_i}}$ `is paired with` $\boldsymbol{x_{F_j}}$

if $jisnotnone$ **then** `// Paired?`

$$\boldsymbol{z_p} = \begin{bmatrix} \boldsymbol{z_p} \\ \boldsymbol{z_{f_i}} \end{bmatrix};$$

$$\boldsymbol{R_p} = \begin{bmatrix} \boldsymbol{R_p} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_{f_i}} \end{bmatrix};$$ `// Noises are independent`

$$\boldsymbol{H_p} = \begin{bmatrix} \boldsymbol{H_p} \\ Jhfj(\hat{\boldsymbol{x}}_k, \boldsymbol{F_j}) \end{bmatrix};$$

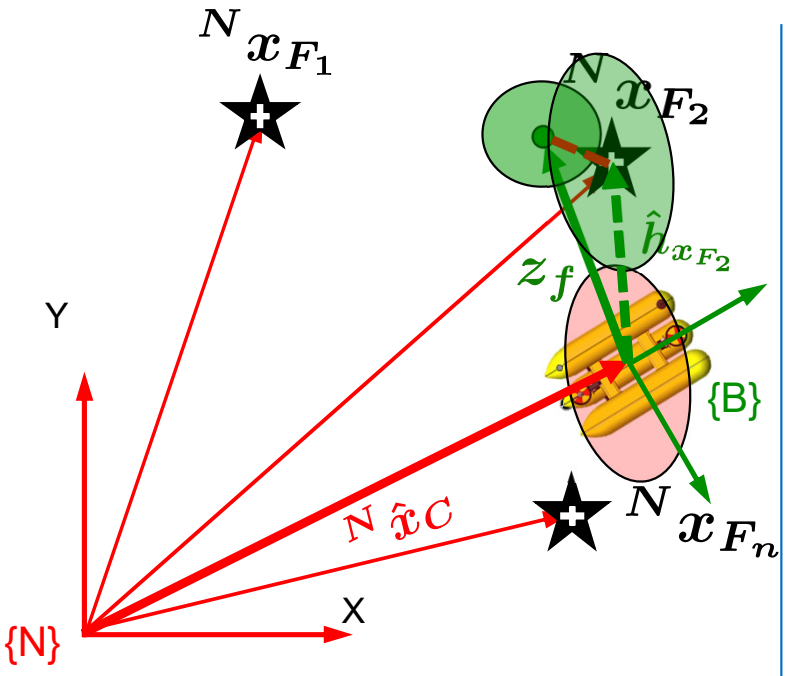
$$\boldsymbol{V_p} = \begin{bmatrix} \boldsymbol{V_p} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix};$$

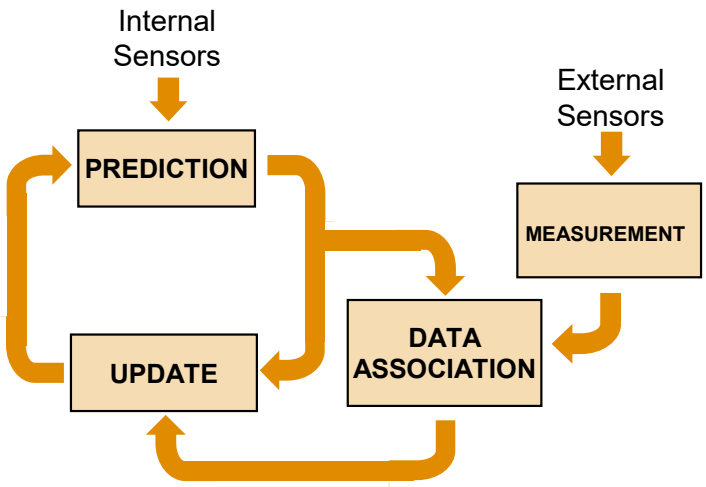
return $[\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}]$

$, \boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$;

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map





Paired Observations

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

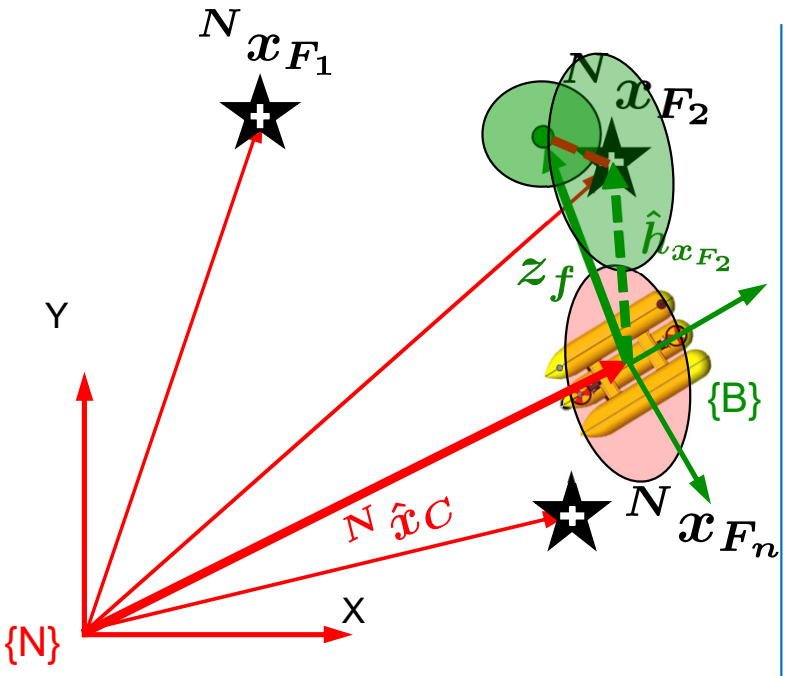
Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

EKF Map Based Localization

FEKFMBL: Feature-based EKF Localization using a priori Map



Paired Observations

Method `LocalizationLoop`$(\hat{x}_0, P_0)$

$[\hat{x}_{k-1}, P_{k-1}] = [\hat{x}_0, P_0]$;
for $k = 1$ *to* ∞ **do**
  $[\hat{x}_k, P_k] = Localize(\hat{x}_{k-1}, P_{k-1})$

Method `Localize`(x_{k-1}, u_k)

$[uk, Qk] = GetInput()$;
$[\hat{\bar{x}}_k, \bar{P}_k] = Prediction(\hat{x}_{k-1}, P_{k-1}, u_k, Q_k)$;
$[z_m, R_m, H_m, V_m] = GetMeasurements()$;
$[z_f, R_f, H_f, V_f] = GetFeatures()$;
$\mathcal{H}_p = DataAssociation(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$;
$[z_k, R_k, H_k, V_k, z_{np}, R_{np}] =$
  $StackMeasurementsAndFeatures(z_m, R_m, H_m, V_m, z_f, R_f, \mathcal{H}_p)$;
$[\hat{x}_k, P_k] = Update(\hat{x}_k, P_{k-1}, z_k, R_k, H_k, V_k)$;

return $[\hat{x}_k, P_k]$

Internal Sensors

PR

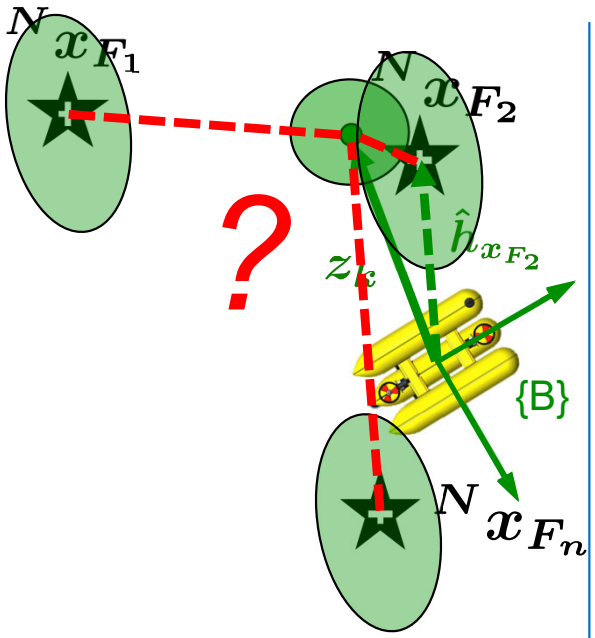
Method `h`(x_k)

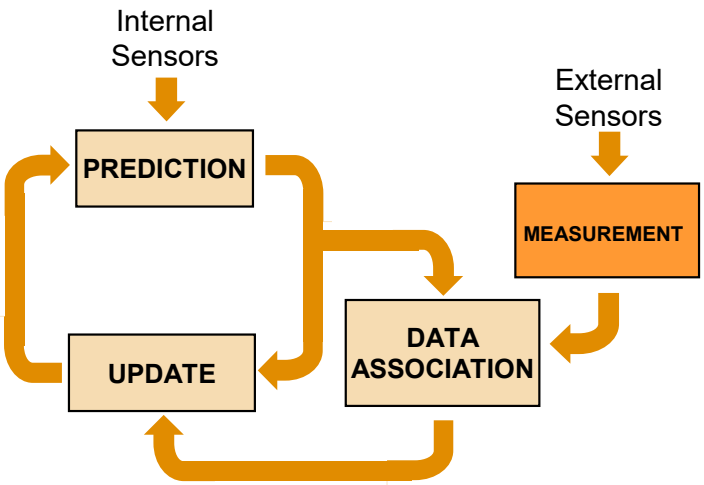
$h_x = \begin{bmatrix} h_m(x_k) \\ h_f(x_k) \end{bmatrix}$;
return h_x

Method `Update`$(\hat{\bar{x}}_k, \bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$

$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + V_k R_k V_k^T)^{-1}$;
$\hat{x}_k = \hat{\bar{x}}_k + K_k \left(z_k - h(\hat{\bar{x}}_k, \mathcal{H}_p)\right)$;
$P_k = (I - K_k H_k) \bar{P}_k (I - K_k H_k)^T$;
return $[\hat{x}_k, P_k]$;

EKF Map Based Localization





Data Association

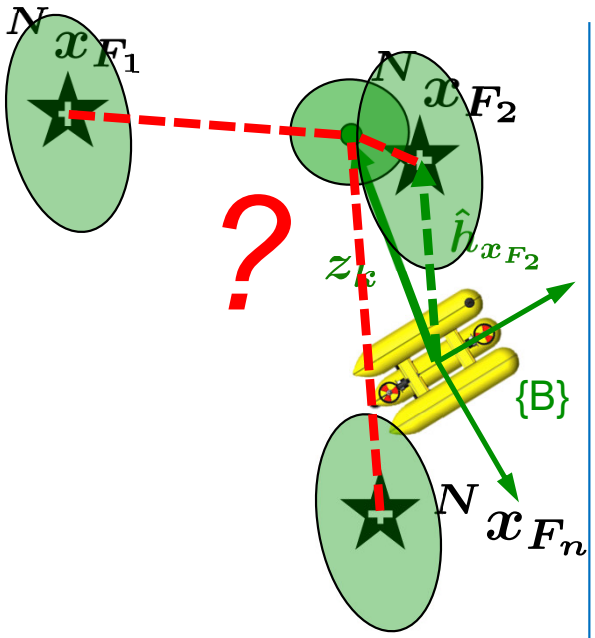
Question: Given an observation data, what map feature better corresponds to this observation?

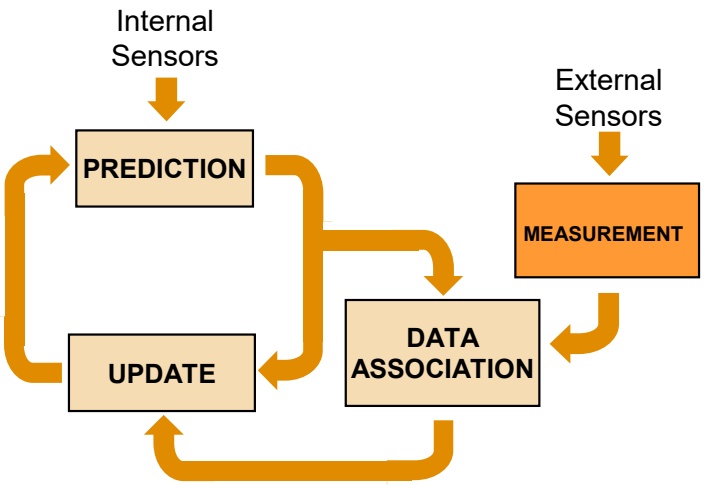
Answer?: Associate the measurement with nearest the feature.

What means Near & Far?

Grover will help us!

EKF Map Based Localization

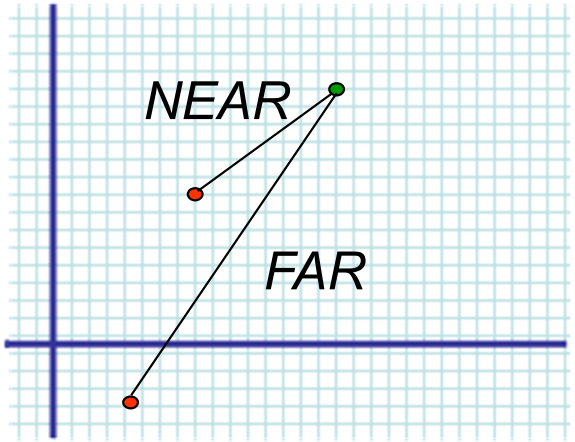




Data Association

Question: Given an observation data, what map feature better corresponds to this observation?

Answer?: Associate the measurement with nearest the feature (euclidean distance).



D_e=10 [m]

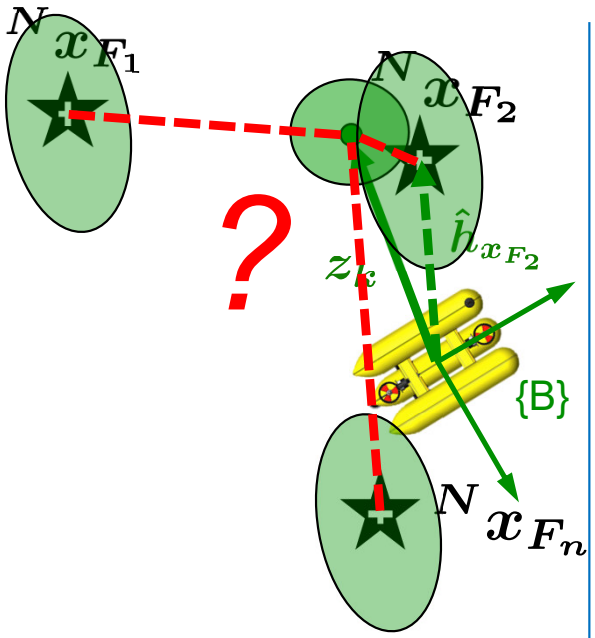
NEAR

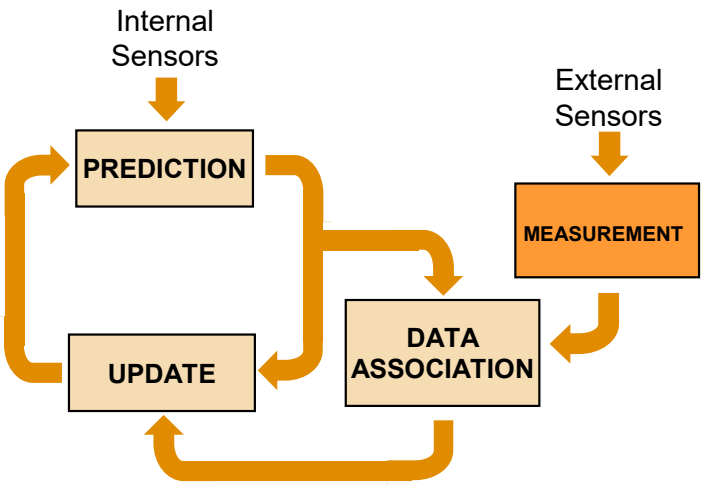
FAR

$$D_e = \sqrt{(f_x - z_x)^2 + (f_y - z_y)^2}$$

$$D_e^2 = \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix}^T$$

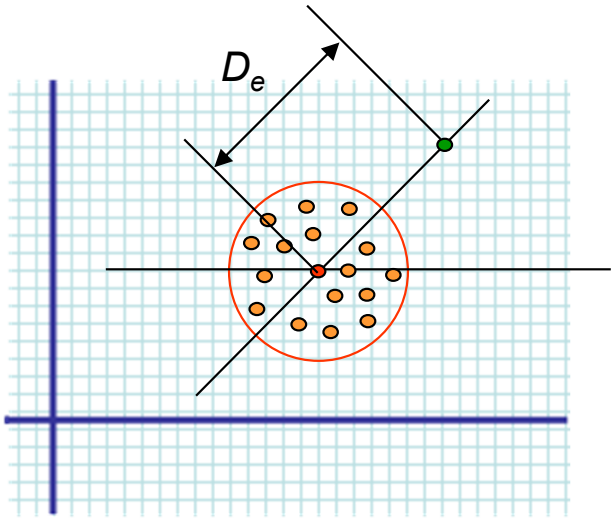
EKF Map Based Localization





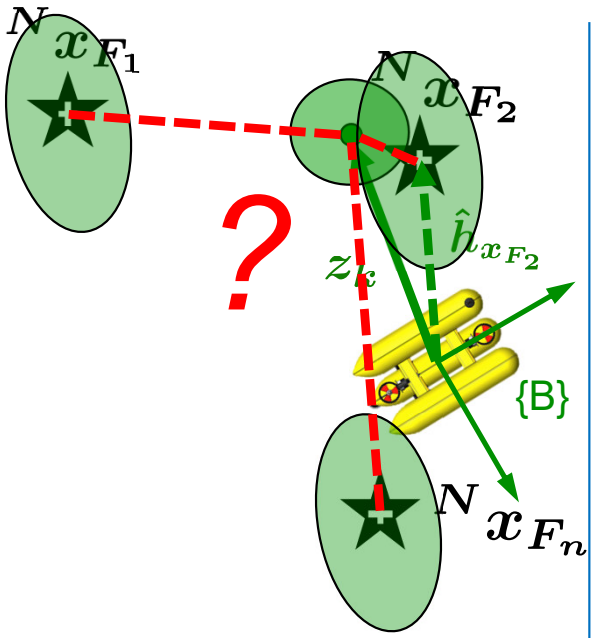
Data Association

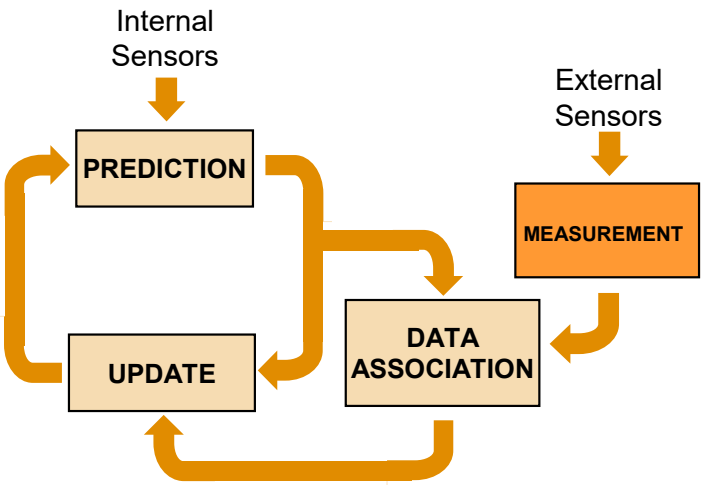
Question: But what happens if the feature is not a data point but a gaussian random vector? What means near? And far?
Answer?: We can compute the distance wrt the mean.



D_e=10 [m]

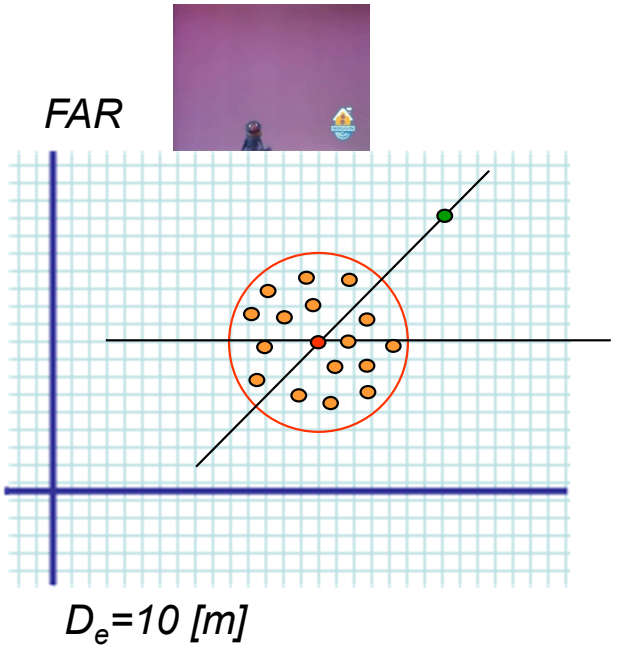
EKF Map Based Localization

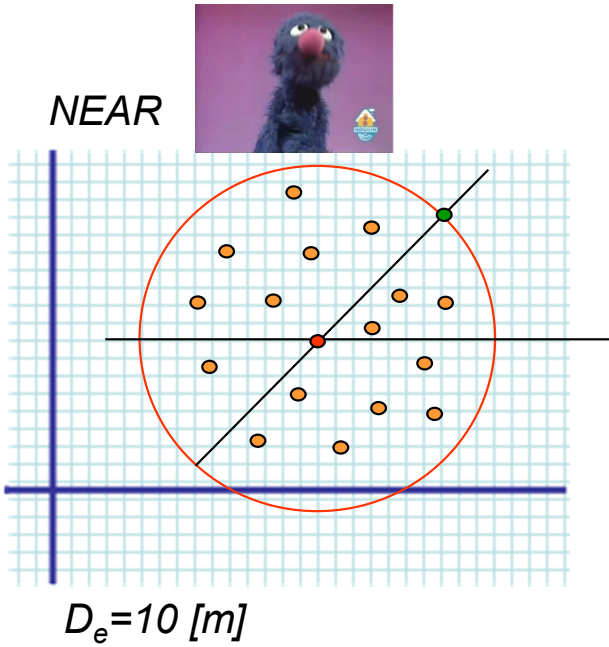




Data Association

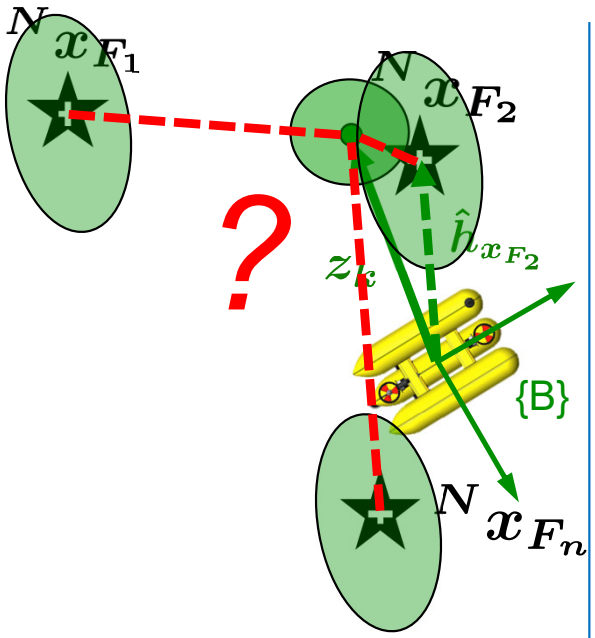
Question: But what happens if the feature is not a data point but a gaussian random vector? What means near? And far?
Answer?: We can compute the distance wrt the mean.

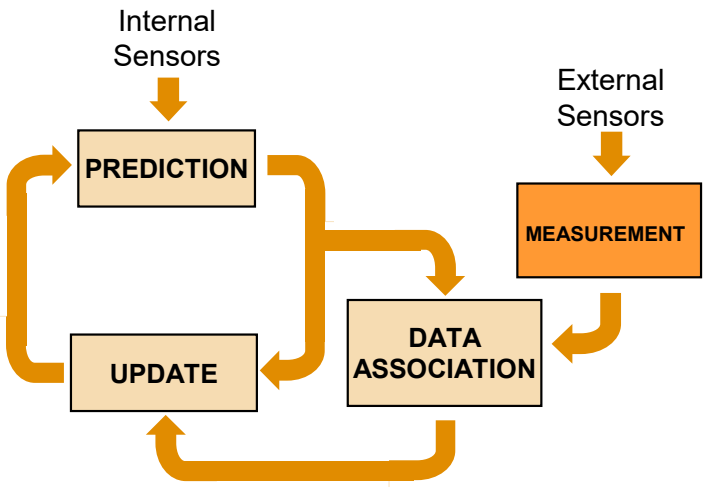




Problem: Both cases show the same euclidean distance!

EKF Map Based Localization

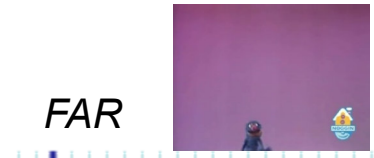




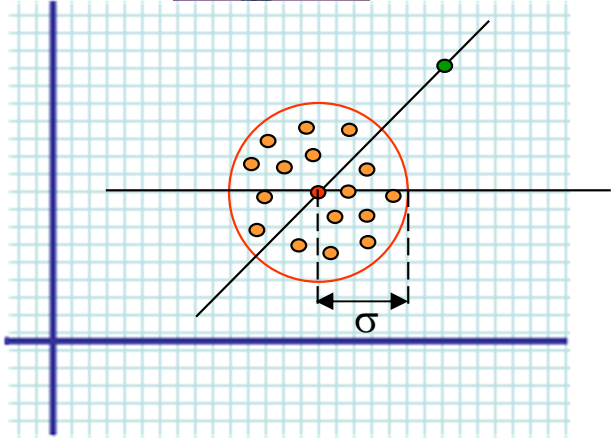
Data Association

Question: How can we distinguish both cases?

Answer?: We can normalize the distance using σ. Now this distance is called standarized distance D_s.

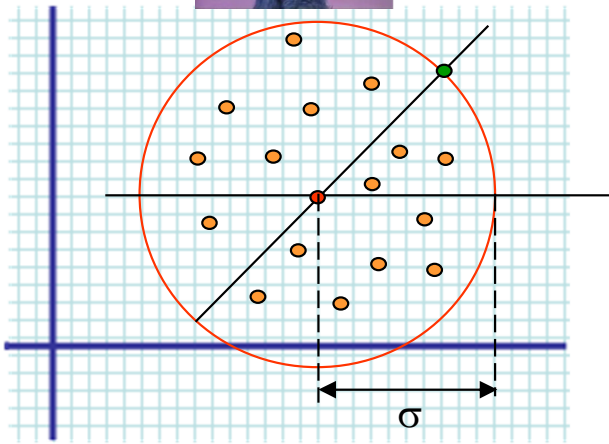


FAR



D_e=10 [m], D_s=2

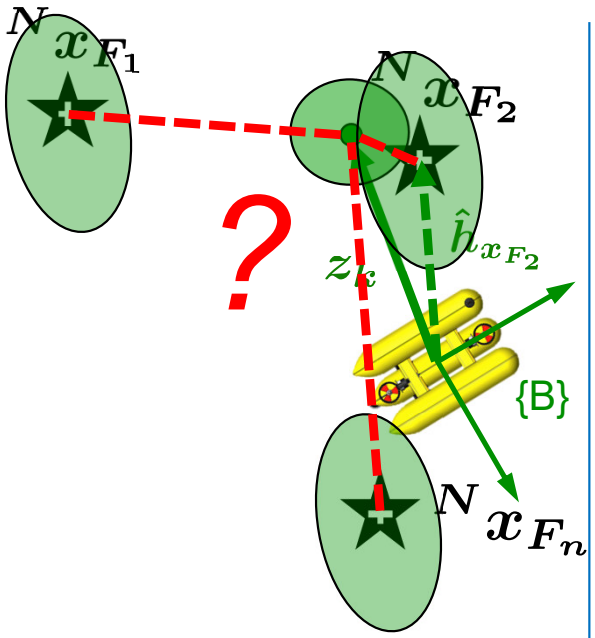
NEAR

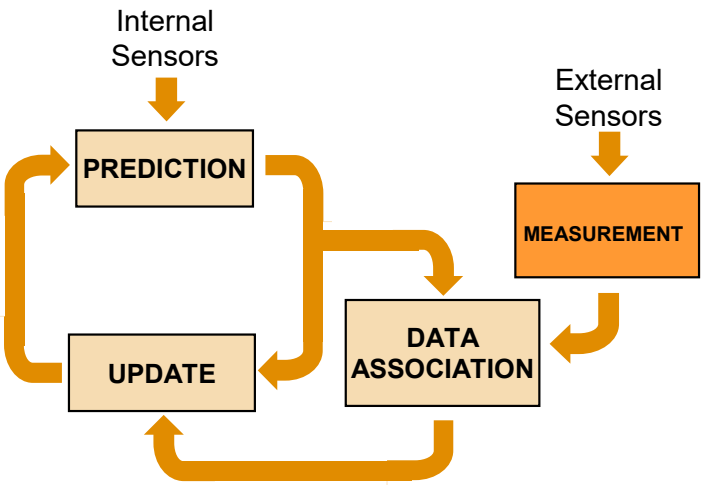


D_e=10 [m], D_s=1

$$D_s^2 = \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma} & 0 \\ 0 & \frac{1}{\sigma} \end{bmatrix} \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix}^T$$

EKF Map Based Localization

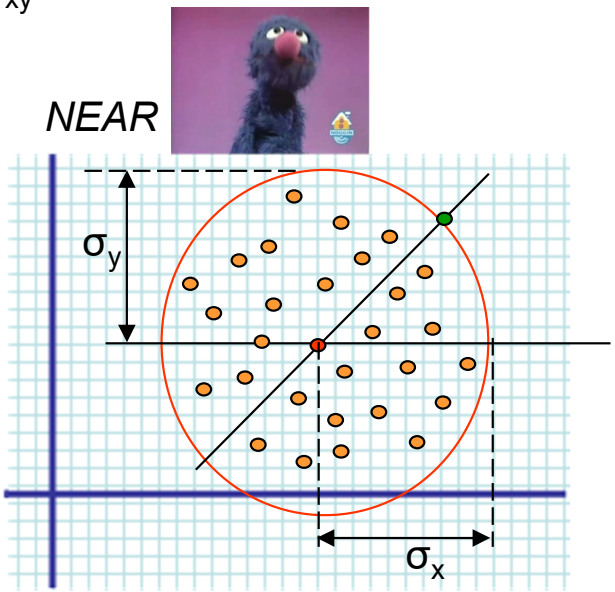




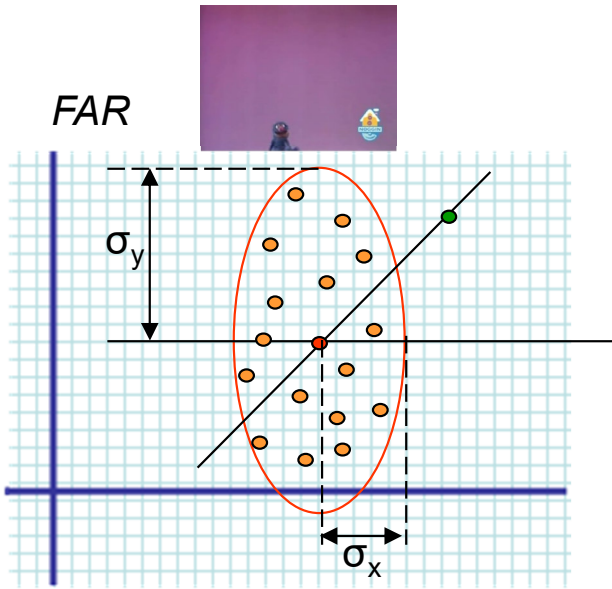
Data Association

Question: Does it work when $\sigma_x \neq \sigma_y$?

Answer: Yes it works while both variables are uncorrelated $\sigma_{xy}=0$



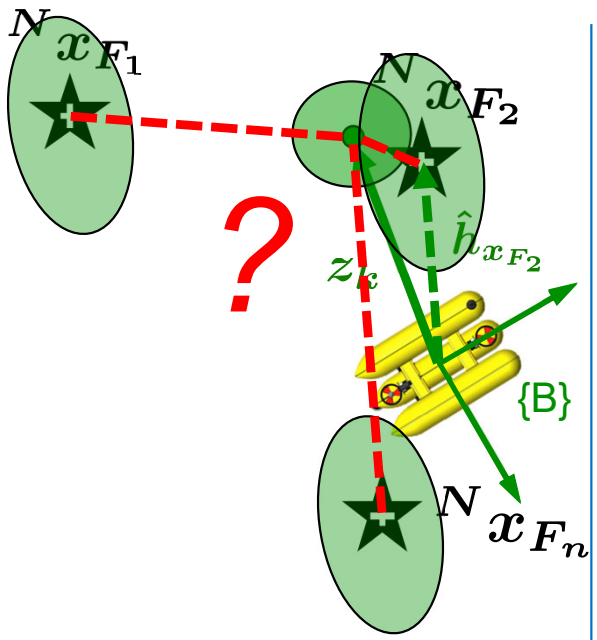
D_e=10 [m], D_s=1

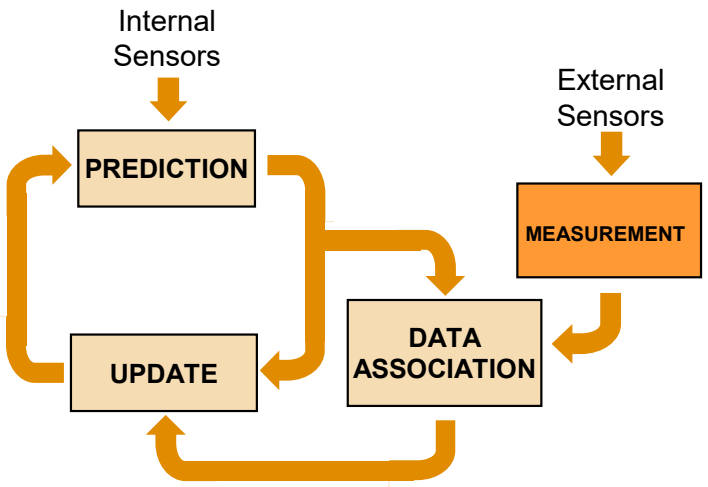


D_e=10 [m], D_s=2,5

$$D_s^2 = \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_x} & 0 \\ 0 & \frac{1}{\sigma_y} \end{bmatrix} \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix}^T$$

EKF Map Based Localization



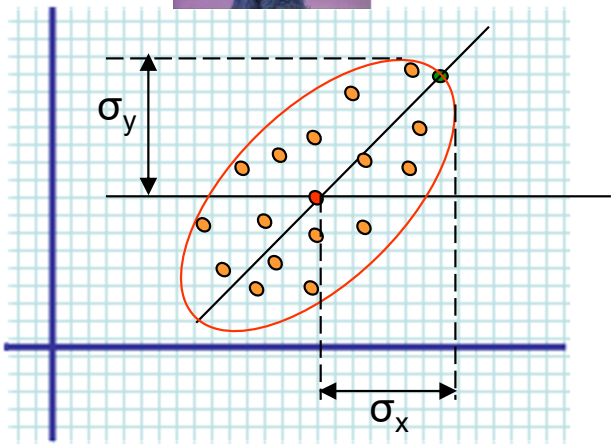


Data Association

Question: What happens if x is correlated with y?

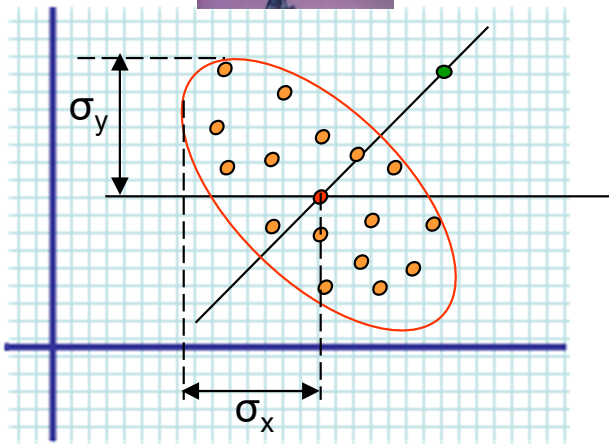
Answer: In this case we have to use the mahalanobis distance which takes into account corrrelations.

NEAR



D_e=10 [m], D_s=2.5, D_m=1

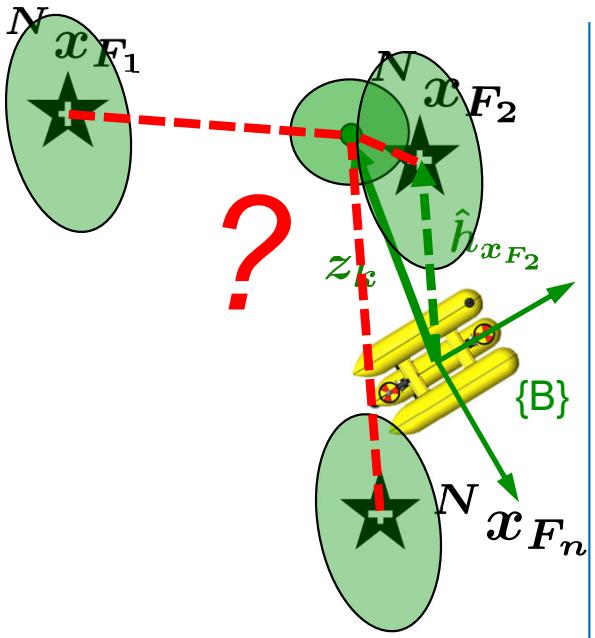
FAR

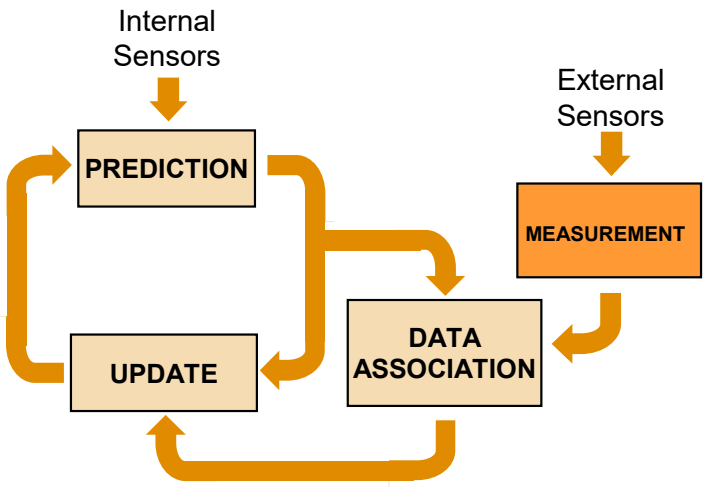


D_e=10 [m], D_s=2.5, D_m=2

$$D_s^2 = \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix} \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix}^{-1} \begin{bmatrix} (f_x - z_x) & (f_y - z_y) \end{bmatrix}^T$$

EKF Map Based Localization



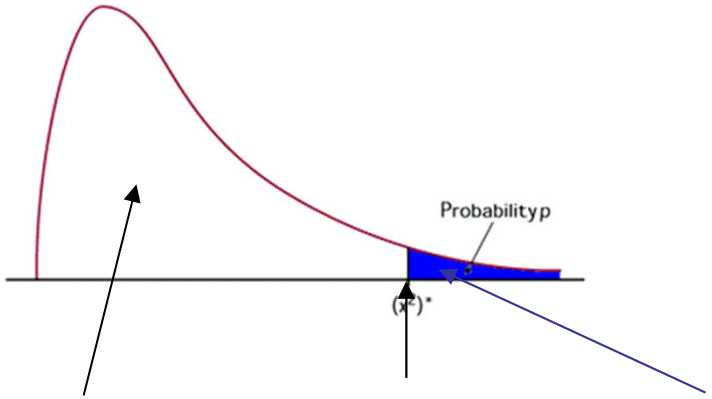


Testing Individual Compatibility

> The innovation v_{ij} measures the discrepancy between the feature an the observation.

> The Mahalanobis distance D_{ij} is computed as the addition of the squares of grv, and its distribution follows a chi-square pdf.

$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij} \Rightarrow D_{ij}^2 = \chi_d^2$$

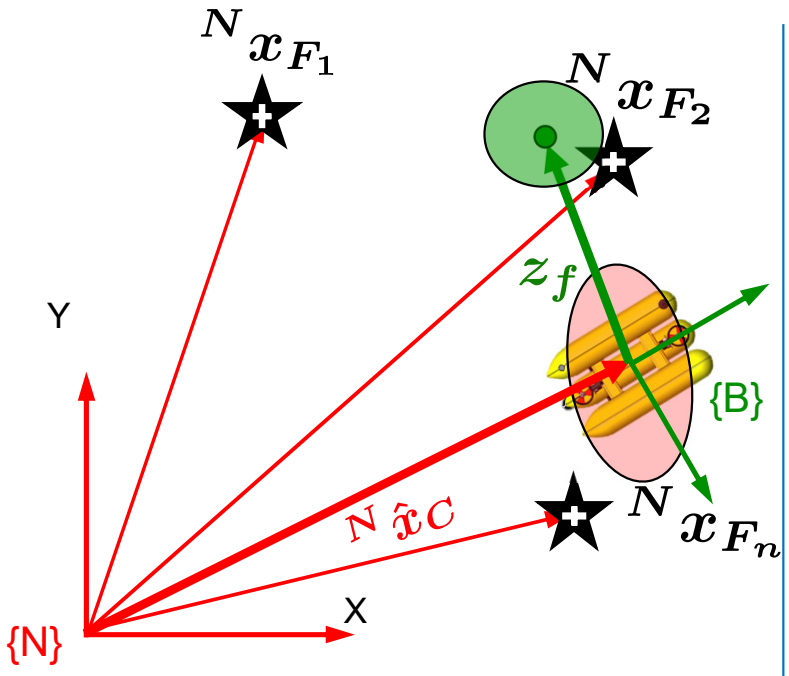


$$P(D_{ij}^2 \le \chi_{d,\alpha}^2) \qquad \chi_{d,\alpha}^2 \qquad P(D_{ij}^2 > \chi_{d,\alpha}^2)$$

> The probability of having a Dij greater than $\chi_{d,\alpha}^2$ when z it is an actual measurement of F is so small that we wil consider that there is no correspondence between them.

$$D_{ij}^2 > \chi_{d,\alpha}^2 \Rightarrow P(D_{ij}^2 > \chi_{d,\alpha}^2) < \alpha \Rightarrow REJECT$$

EKF Map Based Localization



ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j:

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\hat{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\hat{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

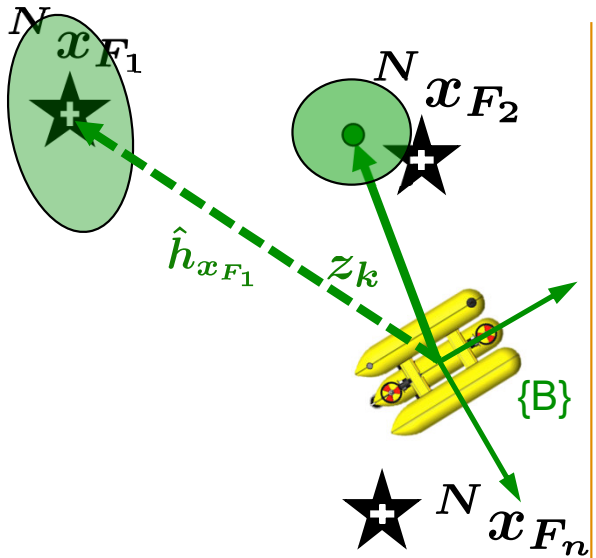
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi_{d,\alpha}^2$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization



ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j: *j=1*

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

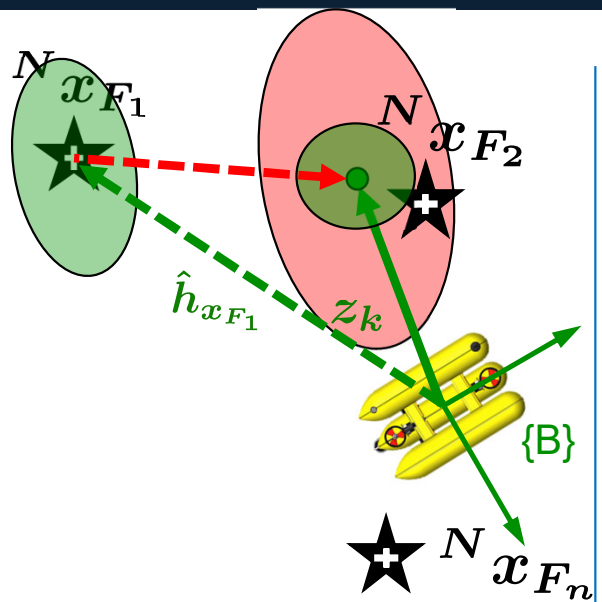
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi_{d,\alpha}^2$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization



ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j: *j=1*

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T + V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

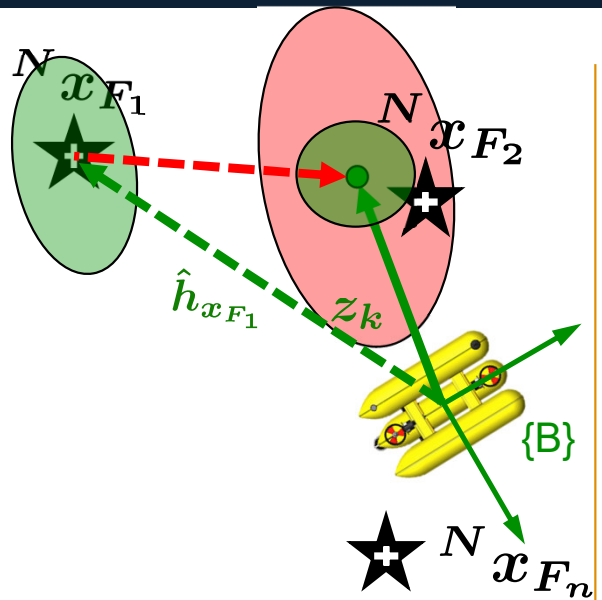
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

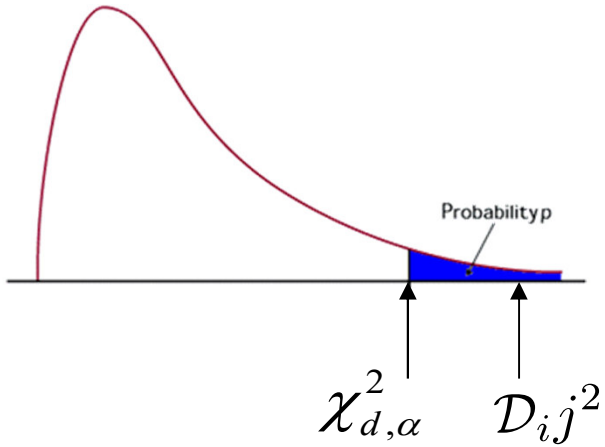
2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi_{d,\alpha}^2$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization





REJECT!

ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j: *j=1*

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

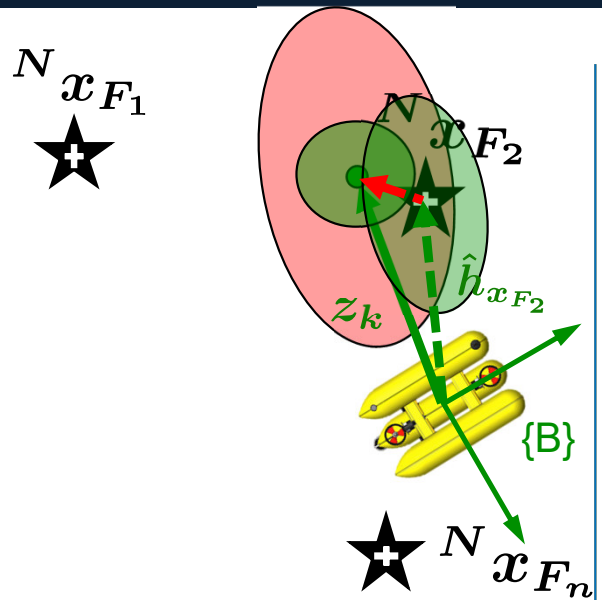
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

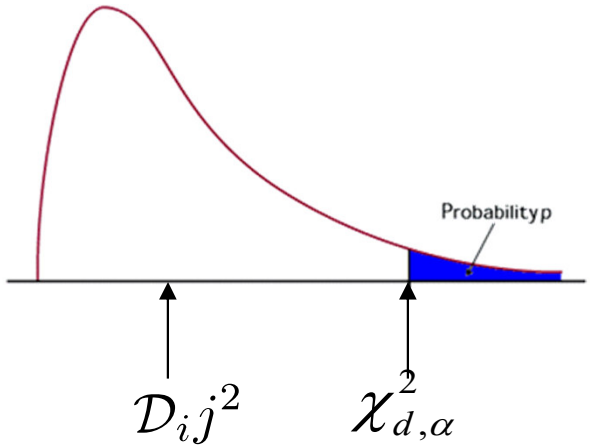
2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi^2_{d,\alpha}$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization





ACCEPT!

ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j: *j=2*

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

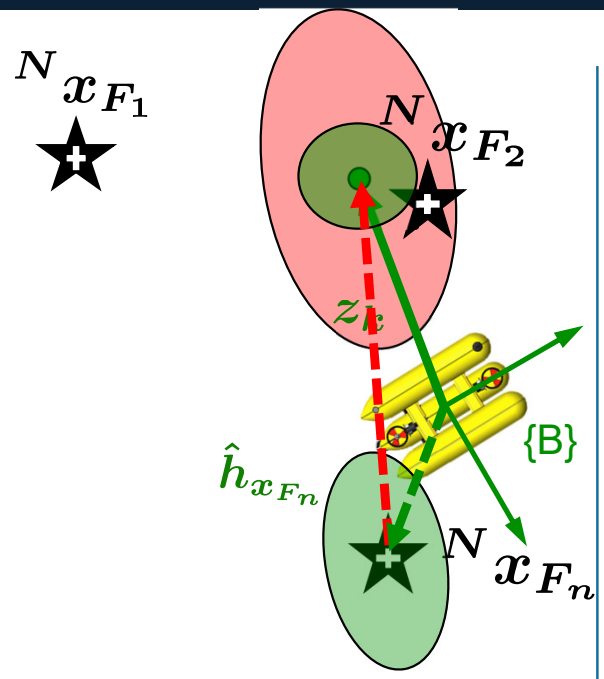
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

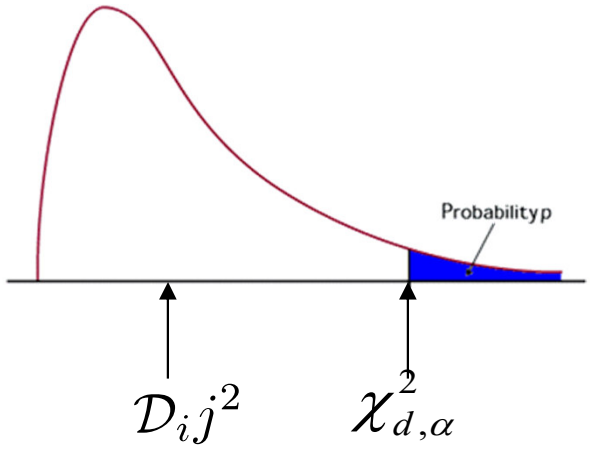
2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi^2_{d,\alpha}$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization





REJECT!

ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j: *j=3*

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

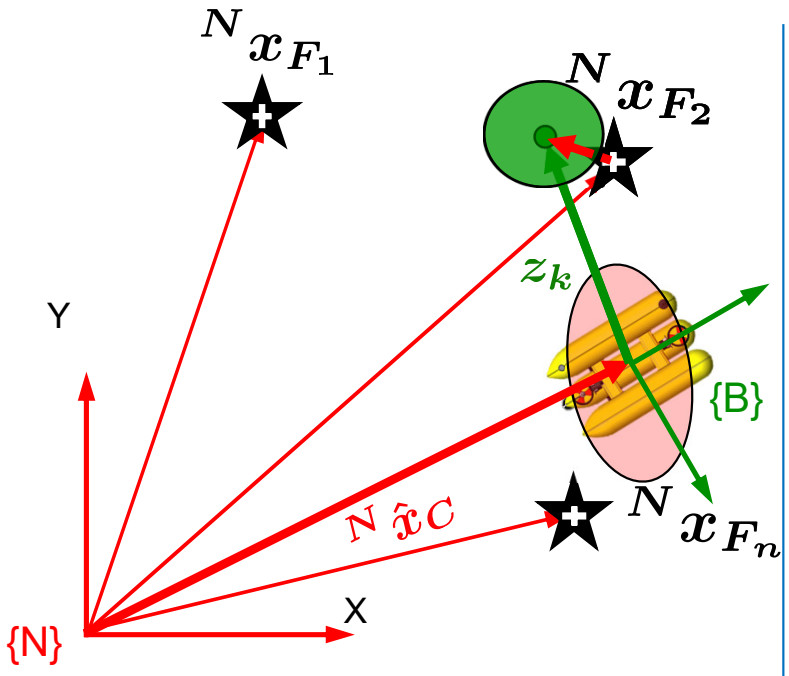
$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi^2_{d,\alpha}$$

3. Select the compatible feature with smallest distance

EKF Map Based Localization



ICNN (Individual Compatibility Nearest Neighbour)

1. Take a measurement z_i with uncertainty R_i

2. For every map feature x_j:

2.1. Compute the robot related position of the feature

$$[h_j(^N\hat{\bar{x}}_R), H_j^N P_R H_j^T] \quad \text{where} \quad H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$

2.2. Compute the innovation & its uncertainty

$$\nu_{ij} = z_i - h_j(^N\hat{\bar{x}}_R)$$
$$S_{ij} = H_j^N P_R H_j^T - V_k R_k V_k^T$$

$$H_j = \frac{\partial h_j(^N x_R, v_k)}{^N x_R}$$
$$V_k = \frac{\partial h_j(^N x_R, v_k)}{v_k}$$

2.3. Compute the mahalanobis distance

$$D_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

2.4. Does it pass the compatibility test?

$$D_{ij}^2 < \chi_{d,\alpha}^2$$

3. Select the compatible feature with smallest distance

Method `DataAssociation`$(\hat{\bar{x}}_k, \bar{P}_k, z_f, R_f)$

- **for** $i = 1$ *to* n **do** // Feature x_{F_i}
  - $h_{F_i} = hfj(x_k, i)$;
  - $P_{F_i} = Jhfj(x_k, i)(P_k, i)Jhfj(x_k, i).T$;
- $\mathcal{H}_p = ICNN(h_F, P_F, z_f, R_f)$;
- **return** $\mathcal{H}_p$;

Method `ICNN`(h_f, P_{h_f}, z_f, R_f)

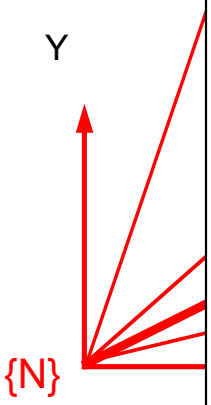
- $\mathcal{H}_p = []$;
- **for** $j = 1$ *to* m **do** // Observation z_{f_j}
  - $nearest = none$; // none ≡ non compatible with any feature
  - $\mathcal{D}^2_{min} = \infty$;
  - **for** $i = 1$ *to* n **do** // Feature x_{F_i}
    - $\mathcal{D}^2_{ij} = SquaredMahalanobisDistance(h_{F_i}, P_{F_i}, z_{f_i}, R_{f_i})$;
    - **if** $IndividualCompatibility(\mathcal{D}^2_{ij}, dof, \alpha)$ & $\mathcal{D}^2_{ij} < \mathcal{D}^2_{min}$ **then**
      - $nearest = i$;
      - $\mathcal{D}^2_{min} = \mathcal{D}^2_{ij}$;
  - $\mathcal{H}_p = [\mathcal{H}_p \; nearest]$;
- **return** $\mathcal{H}_p$

Method `SquaredMahalanobisDistance`$(h_{f_j}, P_{f_j}, z_{f_i}, R_{f_i})$

- $\nu_{ij} = z_{f_i} - h_{f_j}$;
- $S_{ij} = R_{f_i} + P_{f_j}$;
- $\mathcal{D}^2_{ij} = \nu_{ij} S^{-1}_{ij} \nu^T_{ij}$;
- **return** $\mathcal{D}^2_{ij}$

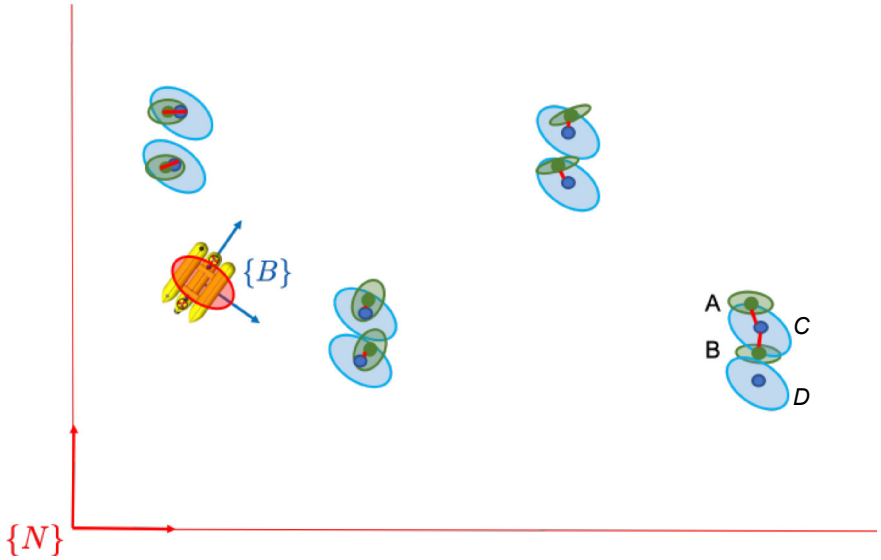
Method `IndividualCompatibility`$(\mathcal{D}^2_{ij}, dof, \alpha)$

- **return** $(\mathcal{D}^2_{ij} \leq \chi^2_{dof,\alpha})$; // Compatibility test

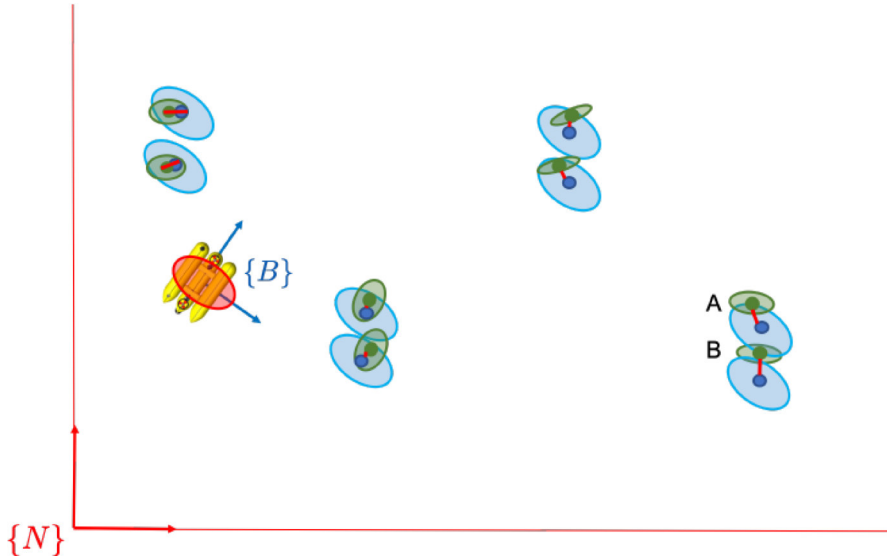


EKF Map Based Localization

ICNN fallacy



- *C is IC with A and B*
- *C is more close to B than A*
- *C is wrongly associated with B and A remain un-paired*



- *C is IC with A and B*
- *Pairing A with C, would allow pairing B with D*

EKF Map Based Localization

Algorithm 16: Joint Compatibility Branch and Bound (JCBB)

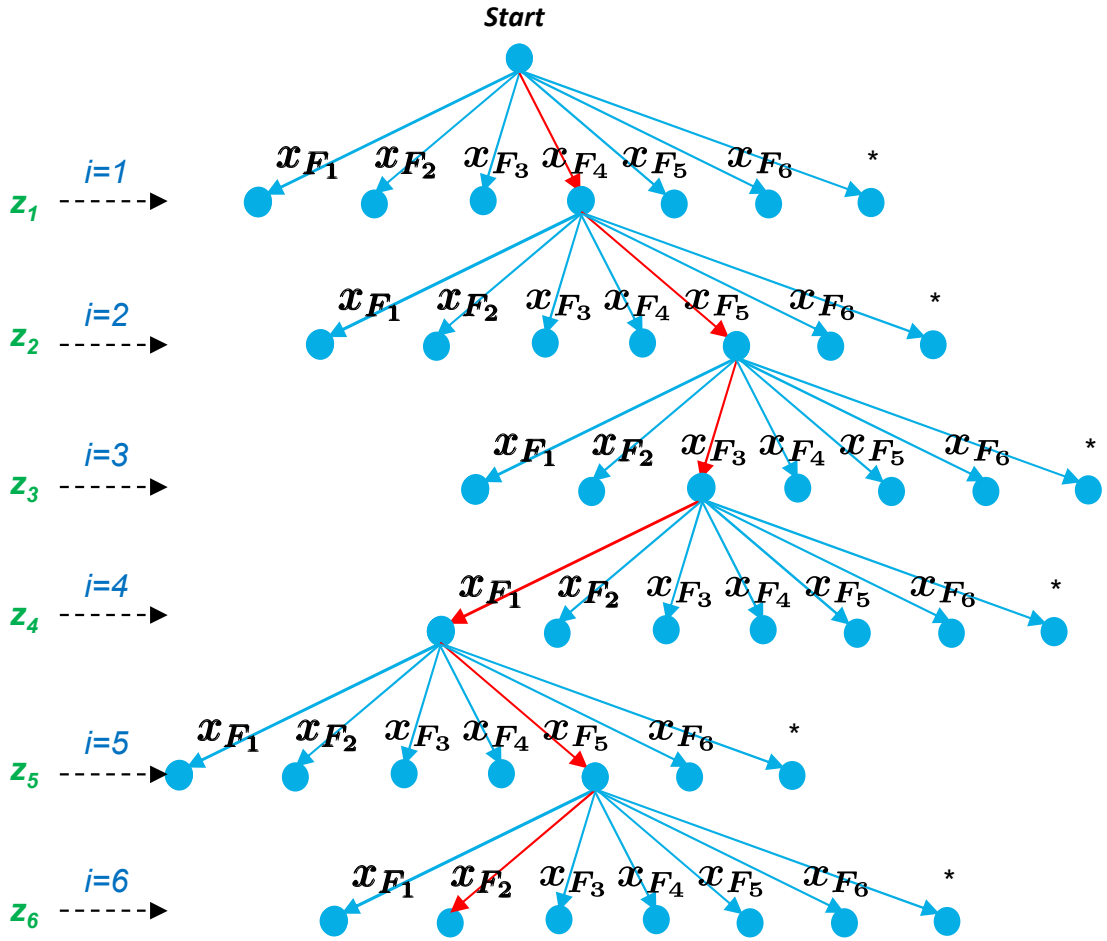
Function JCBB($\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z}_{f_{1\ldots m}}, \boldsymbol{R}_{f_{1\ldots m}}$)
  return $RJCBB\left(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, [\,], [\,], 1, \boldsymbol{z}_{f_{1\ldots m}}, \boldsymbol{R}_{f_{1\ldots m}}\right)$;
Function RJCBB($\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \mathcal{H}_p, \mathcal{H}_B, i, \boldsymbol{z}_{f_{1\ldots m}}, \boldsymbol{R}_{f_{1\ldots m}}$)
  if $i > m$ **then** // Leaf node?
    if $pairings(\mathcal{H}_p) > pairings(\mathcal{H}_B)$ **then**
      $\mathcal{H}_B = \mathcal{H}_p$;
  else
    for $j = 1$ *to* n **do**
      if $IndividuallyCompatible(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z}_{f_i}, \boldsymbol{R}_{f_i}, \boldsymbol{x}_{F_j}, \alpha)$ **and**
      $JointlyCompatible(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z}_{f_{1\ldots i}}, \boldsymbol{R}_{f_{1\ldots i}}, \boldsymbol{x}_{F_{1\ldots n}}, \alpha)$ **then**
        // pairing ($\boldsymbol{z}_{f_i}, \boldsymbol{x}_{F_j}$) accepted
        $\mathcal{H}_B = RJCBB\left(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, [\mathcal{H}_p\ j], \mathcal{H}_B, i+1, \boldsymbol{z}_{f_{1\ldots m}}, \boldsymbol{R}_{f_{1\ldots m}}\right)$
    if $pairings(\mathcal{H}_p) + m - i < pairings(\mathcal{H}_B)$ **then**
      // star node, $\boldsymbol{z}_{f_i}$ not paired
      $\mathcal{H}_B = RJCBB\left(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, [\mathcal{H}_p\ 0], \mathcal{H}_B, i+1, \boldsymbol{z}_{f_{1\ldots m}}, \boldsymbol{R}_{f_{1\ldots m}}\right)$
  return $\mathcal{H}_B$;

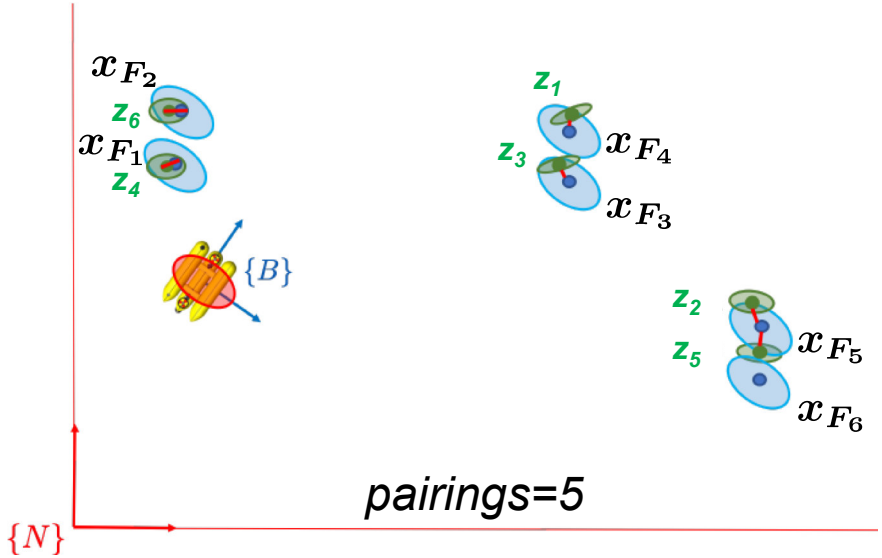
Function JointlyCompatible($\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z}_{\boldsymbol{f}}, \boldsymbol{R}_{\boldsymbol{f}}, \alpha$)
  $\hat{\boldsymbol{h}}_{\boldsymbol{p}} = h(^{N}\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \mathcal{H}_p)$; // Predicted paired feature observations
  $[\boldsymbol{z}_{\boldsymbol{p}}, \boldsymbol{R}_{\boldsymbol{p}}, \boldsymbol{H}_{\boldsymbol{p}}, \boldsymbol{V}_{\boldsymbol{p}}, \boldsymbol{z}_{\boldsymbol{np}}, \boldsymbol{R}_{\boldsymbol{np}}] = SplitFeatures(\boldsymbol{z}_{\boldsymbol{f}}, \boldsymbol{R}_{\boldsymbol{f}}, \mathcal{H}_p)$;
  $\boldsymbol{P}_{\boldsymbol{p}} = \boldsymbol{H}_{\boldsymbol{p}} \bar{\boldsymbol{P}}_{\boldsymbol{k}} \boldsymbol{H}_{\boldsymbol{p}}^{\boldsymbol{T}}$; // Covariance of $\boldsymbol{h}_{\boldsymbol{p}}$
  $\hat{\boldsymbol{\nu}}_{\boldsymbol{p}} = \boldsymbol{z}_{\boldsymbol{p}} - \hat{\boldsymbol{h}}_{\boldsymbol{p}}$; // Innovation
  $\boldsymbol{S}_{\boldsymbol{p}} = \boldsymbol{P}_{\boldsymbol{p}} + \boldsymbol{R}_{\boldsymbol{p}}$; // Innovation uncertainty
  $\mathcal{D}_p^2 = \hat{\boldsymbol{\nu}}_{\boldsymbol{p}} \boldsymbol{S}_{\boldsymbol{p}}^{-1} \hat{\boldsymbol{\nu}}_{\boldsymbol{p}}^{\boldsymbol{T}}$; // Mahalanobis distance
  return $(\mathcal{D}_p^2 \leq \chi_{d,\alpha}^2)$; // Compatibility test

Algorithm 1: χ_d^2 Joint compatibility test

EKF Map Based Localization

Joint Compatibility Branch and Bound (JCBB)



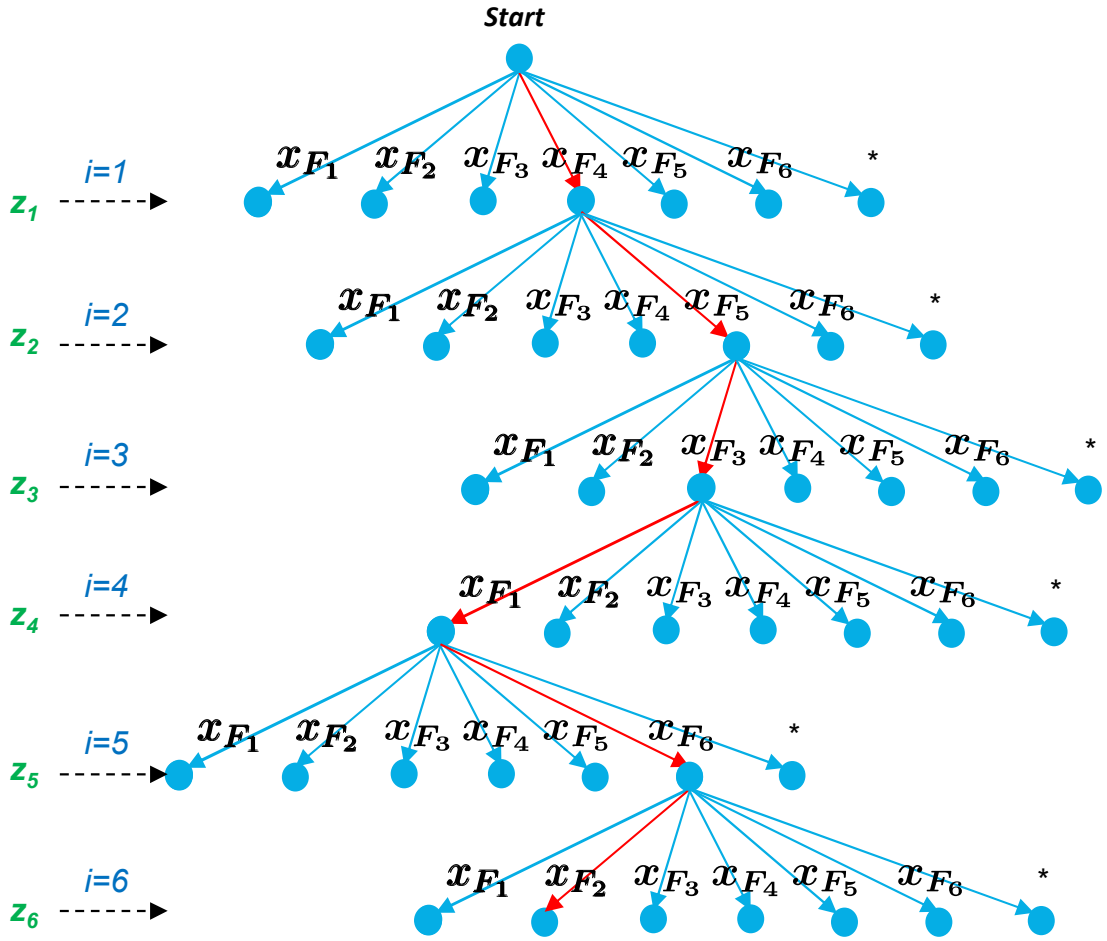


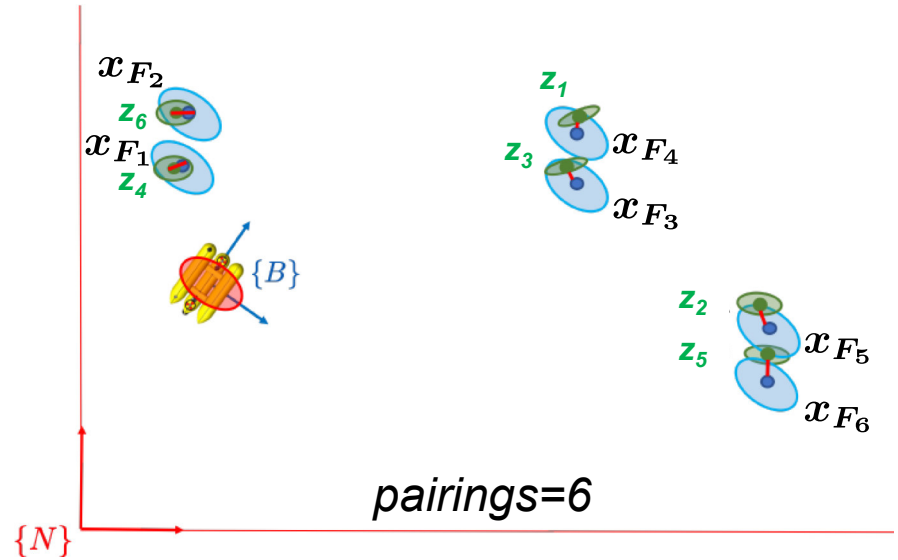
Algorithm 16: Joint Compatibility Branch and Bound (JCBB)

1 **Function** JCBB($\hat{\bar{x}}_k, \bar{P}_k, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}$)
2 **return** $RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\,], [\,], 1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$;
3 **Function** RJCBB($\hat{\bar{x}}_k, \bar{P}_k, \mathcal{H}_p, \mathcal{H}_B, i, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}$)
4 **if** $i > m$ **then** // Leaf node?
5 **if** $pairings(\mathcal{H}_p) > pairings(\mathcal{H}_B)$ **then**
6 $\mathcal{H}_B = \mathcal{H}_p$;
7 **else**
8 **for** $j = 1$ *to* n **do**
9 **if** $IndividuallyCompatible(\hat{\bar{x}}_k, \bar{P}_k, z_{f_i}, R_{f_i}, x_{F_j}, \alpha)$ **and** $JointlyCompatible(\hat{\bar{x}}_k, \bar{P}_k, z_{f_{1\ldots i}}, R_{f_{1\ldots i}}, x_{F_{1\ldots n}}, \alpha)$ **then**
// pairing (z_{f_i},x_{F_j}) accepted
10 $\mathcal{H}_B = RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\mathcal{H}_p\ j], \mathcal{H}_B, i+1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$
11 **if** $pairings(\mathcal{H}_p) + m - i < pairings(\mathcal{H}_B)$ **then**
// star node, z_{f_i} not paired
12 $\mathcal{H}_B = RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\mathcal{H}_p\ 0], \mathcal{H}_B, i+1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$
13 **return** $\mathcal{H}_B$;

EKF Map Based Localization

Joint Compatibility Branch and Bound (JCBB)



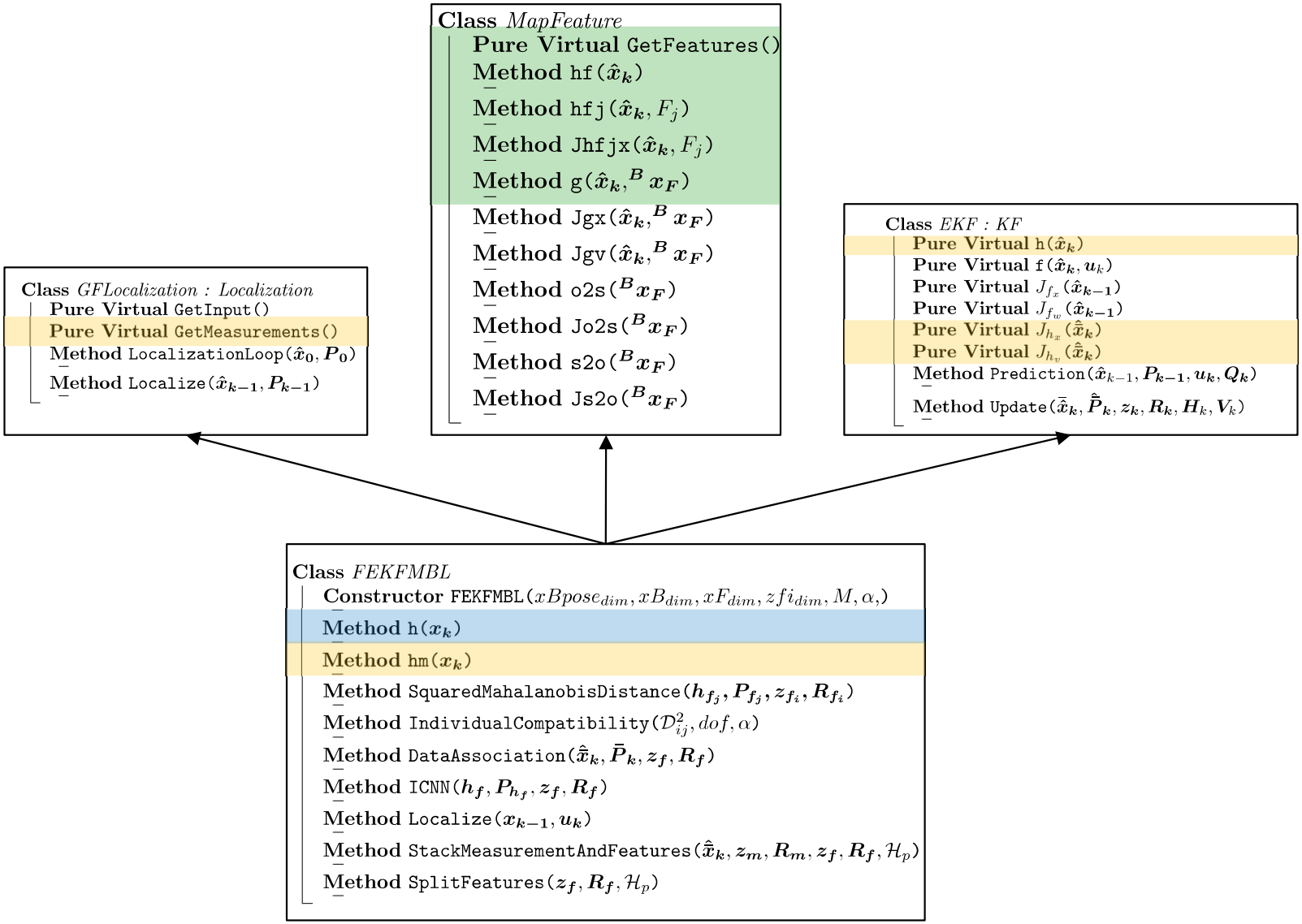


Algorithm 16: Joint Compatibility Branch and Bound (JCBB)

Function JCBB($\hat{\bar{x}}_k, \bar{P}_k, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}$)
  return $RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\,], [\,], 1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$;
Function RJCBB($\hat{\bar{x}}_k, \bar{P}_k, \mathcal{H}_p, \mathcal{H}_B, i, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}$)
  if $i > m$ **then** // Leaf node?
    if $pairings(\mathcal{H}_p) > pairings(\mathcal{H}_B)$ **then**
      $\mathcal{H}_B = \mathcal{H}_p$;
  else
    for $j = 1$ *to* n **do**
      if $IndividuallyCompatible(\hat{\bar{x}}_k, \bar{P}_k, z_{f_i}, R_{f_i}, x_{F_j}, \alpha)$ **and**
       $JointlyCompatible(\hat{\bar{x}}_k, \bar{P}_k, z_{f_{1\ldots i}}, R_{f_{1\ldots i}}, x_{F_{1\ldots n}}, \alpha)$ **then**
        // pairing (z_{f_i},x_{F_j}) accepted
        $\mathcal{H}_B = RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\mathcal{H}_p\ j], \mathcal{H}_B, i+1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$
    if $pairings(\mathcal{H}_p) + m - i < pairings(\mathcal{H}_B)$ **then**
      // star node, z_{f_i} not paired
      $\mathcal{H}_B = RJCBB\left(\hat{\bar{x}}_k, \bar{P}_k, [\mathcal{H}_p\ 0], \mathcal{H}_B, i+1, z_{f_{1\ldots m}}, R_{f_{1\ldots m}}\right)$
  return $\mathcal{H}_B$;

EKF Map Based Localization

FEKFMBL: Implementation



EKF Map Based Localization

Class *FEKFMBL:GFLocalization, MapFeature*

Method `h`$(\boldsymbol{x_k})$

$h_x = \begin{bmatrix} h_m(\boldsymbol{x_k}) \\ hf(\boldsymbol{x_k}) \end{bmatrix}$;

return h_x

Method `hm`$(\boldsymbol{x_k})$

$h_m = parent.h(\boldsymbol{x_k})$;

return h_m

Method `SquaredMahalanobisDistance`$(\boldsymbol{h_{f_j}}, \boldsymbol{P_{f_j}}, \boldsymbol{z_{f_i}}, \boldsymbol{R_{f_i}})$

$\boldsymbol{\nu_{ij}} = \boldsymbol{z_{f_i}} - \boldsymbol{h_{f_j}}$;

$\boldsymbol{S_{ij}} = \boldsymbol{R_{f_i}} + \boldsymbol{P_{f_j}}$;

$\boldsymbol{\mathcal{D}^2_{ij}} = \boldsymbol{\nu_{ij}} \boldsymbol{S_{ij}^{-1}} \boldsymbol{\nu_{ij}^T}$;

return $\boldsymbol{\mathcal{D}^2_{ij}}$

Method `IndividualCompatibility`$(\mathcal{D}^2_{ij}, dof, \alpha)$

return $(\mathcal{D}^2_{ij} \leq \chi^2_{dof,\alpha})$; `// Compatibility test`

Method `DataAssociation`$(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z_f}, \boldsymbol{R_f})$

for $i = 1$ *to* n **do** `// Feature` $\boldsymbol{x_{F_i}}$

$\boldsymbol{h_{F_i}} = hfj(\boldsymbol{x_k}, i)$;

$\boldsymbol{P_{F_i}} = Jhfj(\boldsymbol{x_k}, i)(\boldsymbol{P_k}, i)Jhfj(\boldsymbol{x_k}, i).T$;

$\mathcal{H}_p = ICNN(\boldsymbol{h_F}, \boldsymbol{P_F}, \boldsymbol{z_f}, \boldsymbol{R_f})$;

// $\mathcal{H}_p = JCBB(\boldsymbol{h_F}, \boldsymbol{P_F}, \boldsymbol{z_f}, \boldsymbol{R_f})$ `could also be used instead of ICNN`

return $\mathcal{H}_p$;

Method `ICNN`$(\boldsymbol{h_f}, \boldsymbol{P_{h_f}}, \boldsymbol{z_f}, \boldsymbol{R_f})$

$\mathcal{H}_p = []$;

for $j = 1$ *to* m **do** `// Observation` $\boldsymbol{z_{f_j}}$

$nearest = 0$; `// 0 ≡ non compatible with any feature`

$\mathcal{D}^2_{min} = \infty$;

for $i = 1$ *to* n **do** `// Feature` $\boldsymbol{x_{F_i}}$

$\mathcal{D}^2_{ij} = SquaredMahalanobisDistance(\boldsymbol{h_{F_i}}, \boldsymbol{P_{F_i}}, \boldsymbol{z_{f_i}}, \boldsymbol{R_{f_i}})$;

if $IndividuallyCompatibility(\mathcal{D}^2_{ij}, dof, \alpha)$ & $\mathcal{D}^2_{ij} < \mathcal{D}^2_{min}$ **then**

$nearest = i$;

$\mathcal{D}^2_{min} = \mathcal{D}^2_{ij}$;

$\mathcal{H}_p = [\mathcal{H}_p \; nearest]$;

return $\mathcal{H}_p$

Method `Localize`$(\boldsymbol{x_{k-1}}, \boldsymbol{u_k})$

$[\boldsymbol{uk}, \boldsymbol{Qk}] = GetInput()$;

$[\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}] = Prediction(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}, \boldsymbol{P_{k-1}}, \boldsymbol{u_k}, \boldsymbol{Q_k})$;

$[\boldsymbol{zk}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = GetMeasurements()$;

$\mathcal{H}_p = DataAssociation(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z_f}, \boldsymbol{R_f})$;

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}] =$
$StackMeasurementsAndFeatures(\boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{H_m}, \boldsymbol{V_m}, \boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$;

$[\hat{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{P_k}] = Update(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{P_{k-1}}, \boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k})$;

return $[\hat{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{P_k}]$

Method `StackMeasurementAndFeatures`$(\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$

$[\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}] = SplitFeatures(\boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$;

if *empty* $\boldsymbol{z_m}$ **then**

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = [\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}]$;

else if *empty* $\boldsymbol{z_p}$ **then**

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = [\boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{H_m}, \boldsymbol{V_m}]$;

else

$\boldsymbol{z_k} = \begin{bmatrix} \boldsymbol{z_m} \\ \boldsymbol{z_p} \end{bmatrix}$; $\boldsymbol{R_k} = \begin{bmatrix} \boldsymbol{R_m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_p} \end{bmatrix}$;

$\boldsymbol{H_k} = \begin{bmatrix} \boldsymbol{H_m} \\ \boldsymbol{H_p} \end{bmatrix}$; $\boldsymbol{V_k} = \begin{bmatrix} \boldsymbol{V_m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V_p} \end{bmatrix}$;

return $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}]$

Method `SplitFeatures`$(\boldsymbol{z_f}, \boldsymbol{R_f}, \mathcal{H}_p)$

$\boldsymbol{H_p} = \boldsymbol{V_p} = \boldsymbol{z_p} = \boldsymbol{R_p} = \boldsymbol{z_{np}} = \boldsymbol{R_{np}} = []$;

for $i = 1$ *to* $length(\mathcal{H}_p)$ **do** `// For all candidate associations`

$j = \mathcal{H}_p[i]$; `//` $\boldsymbol{z_{f_i}}$ `is paired with` $\boldsymbol{x_{F_j}}$

if $j \neq 0$ **then** `// Paired?`

$\boldsymbol{z_p} = \begin{bmatrix} \boldsymbol{z_p} \\ FeatureObservation(\boldsymbol{z_f}, i) \end{bmatrix}$;

$\boldsymbol{R_p} = \begin{bmatrix} \boldsymbol{R_p} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_{f_i}} \end{bmatrix}$; `// Noises are independent`

$\boldsymbol{H_p} = \begin{bmatrix} \boldsymbol{H_p} \\ JfHix(\hat{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{j}) \end{bmatrix}$;

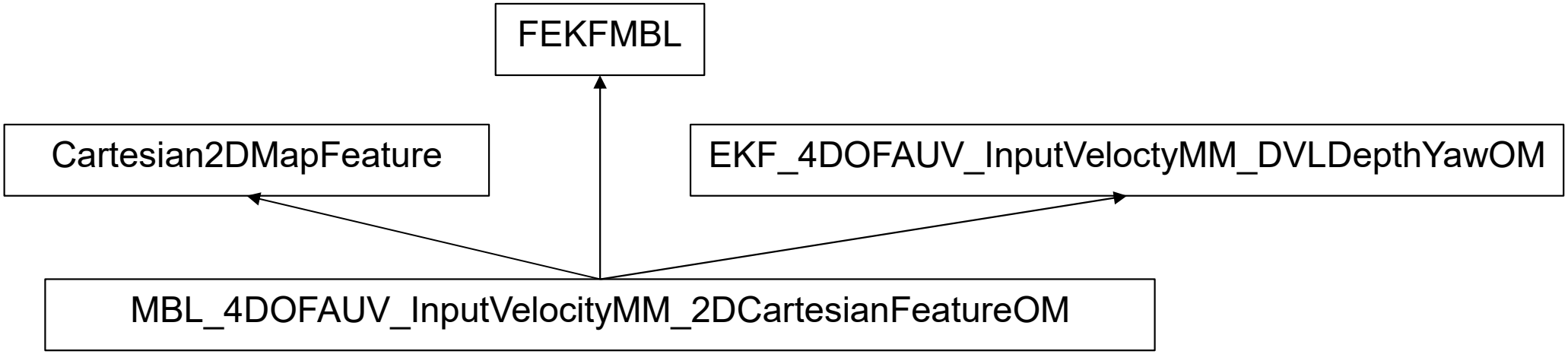
$\boldsymbol{V_p} = \begin{bmatrix} \boldsymbol{V_p} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$;

return $[\boldsymbol{z_p}, \boldsymbol{R_p}, \boldsymbol{H_p}, \boldsymbol{V_p}, \boldsymbol{z_{np}}, \boldsymbol{R_{np}}]$

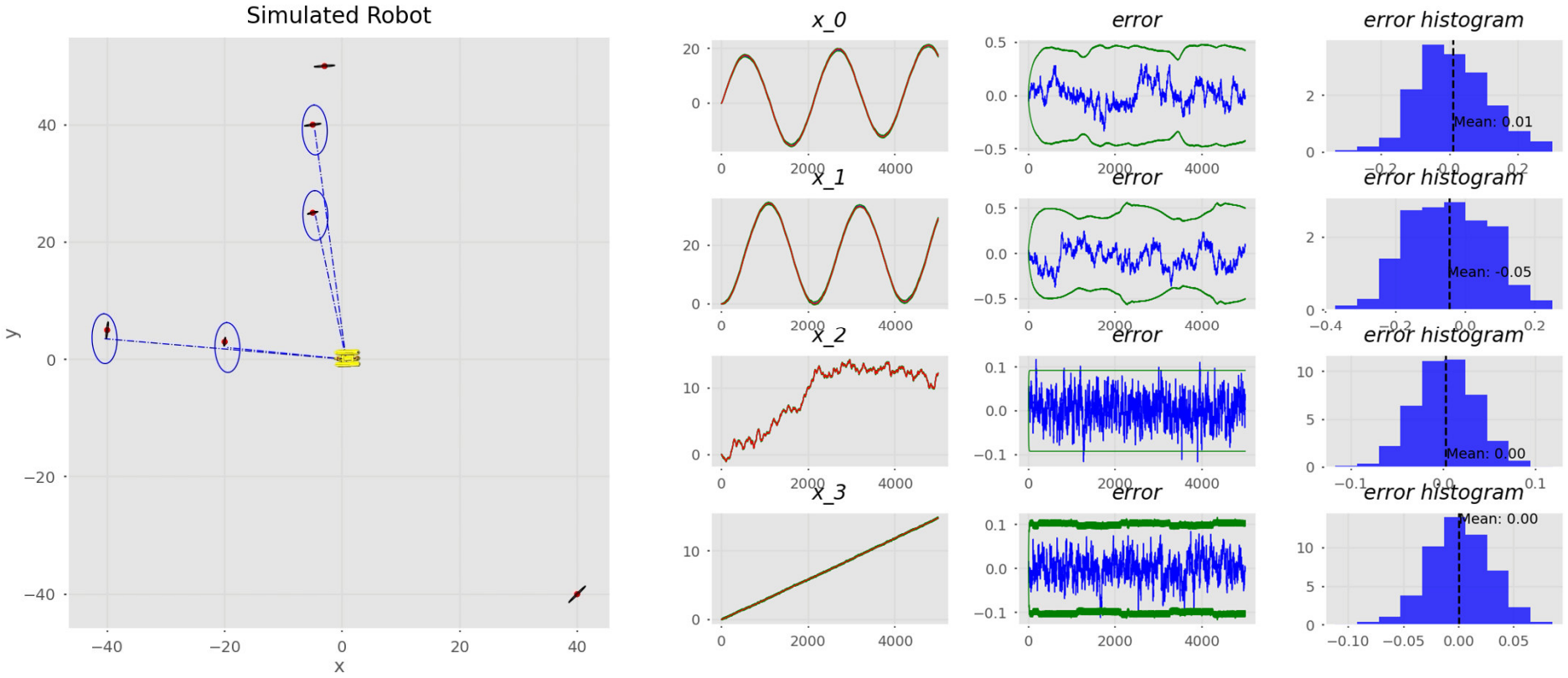
4 DOF AUV Localization

- **Motion Model:** **Input Velocity**
- **Observation Mosel :** **2D Point Features Stored in Cartesian Observed In Cartesian**

Class $MBL\_4DOFAUV\_InputVelocityMM\_2DCartesianFeatureOM:$ $Cartesian2DMapFeature,\ FEKFMBL,$ $EKF\_4DOFAUV\_InputVelocityMM\_DVLDepthYawOM$



EKF Map Based Localization



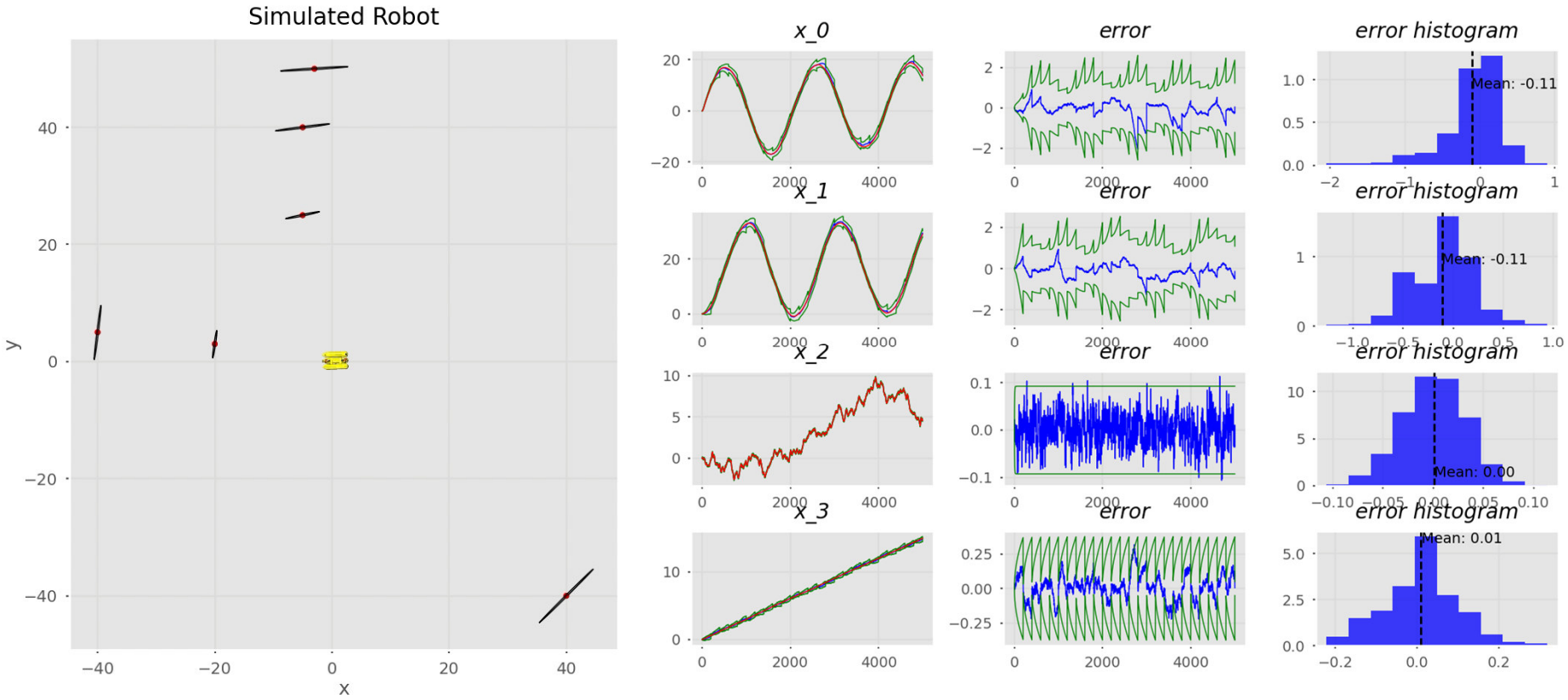
Input

- **DVL:** 1 every 0.1 s, $R_{DVL=}diag([0.1^2, 0.1^2, 0.1^2])$
- **Gyro:** 1 every 0.1 s, σ_{yaw}= 5º/s

Measurement

- **Depth:** 1 every 0.1 s, σ_{depth}= 0.1 m
- **Compass:** None, σ_{yaw}= 1º
- **Feature:** 1 every 5 s, $R_{Fxy=}diag([0.5^2\,m^2, 1^2\,m^2])$

EKF Map Based Localization



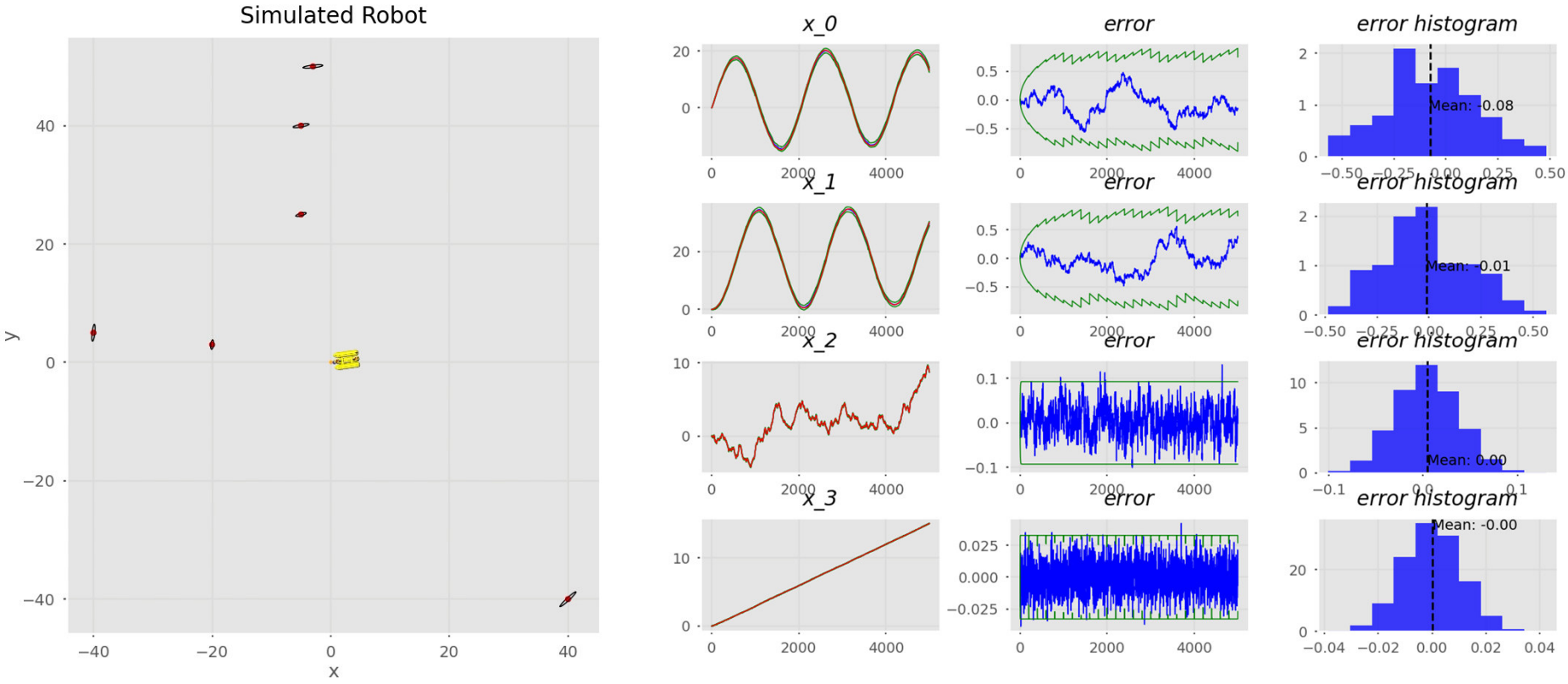
Input

- **DVL:** 1 every 0.1 s, $R_{DVL=}diag([0.1^2, 0.1^2, 0.1^2])$
- **Gyro:** 1 every 0.1 s, σ_{yaw}= 5º/s

Measurement

- **Depth:** 1 every 0.1 s, σ_{depth}= 0.1 m
- **Compass:** None, σ_{yaw}= 1º
- **Feature:** 1 every 200 s, $R_{Fxy=}diag([0.5^2\,m^2, 1^2\,m^2])$

EKF Map Based Localization



Input

- **DVL:** 1 every 0.1 s, $R_{DVL=}diag([0.1^2, 0.1^2, 0.1^2])$
- **Gyro:** 1 every 0.1 s, σ_{yaw}= 5º/s

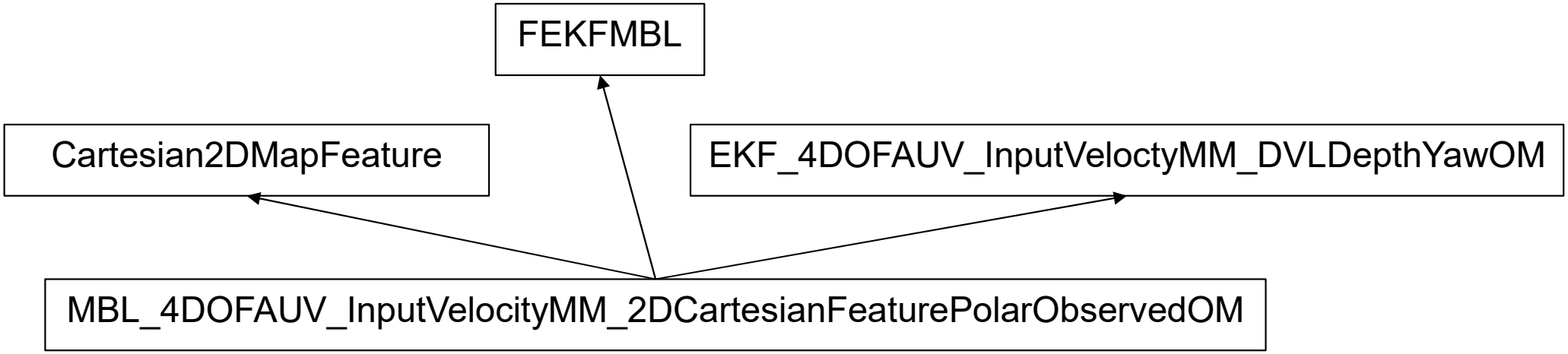
Measurement

- **Depth:** 1 every 0.1 s, σ_{depth}= 0.1 m
- **Compass:** 1 every 1 s, σ_{yaw}= 1º
- **Feature:** 1 every 200 s, $R_{Fxy=}diag([0.5^2\,m^2, 1^2\,m^2])$

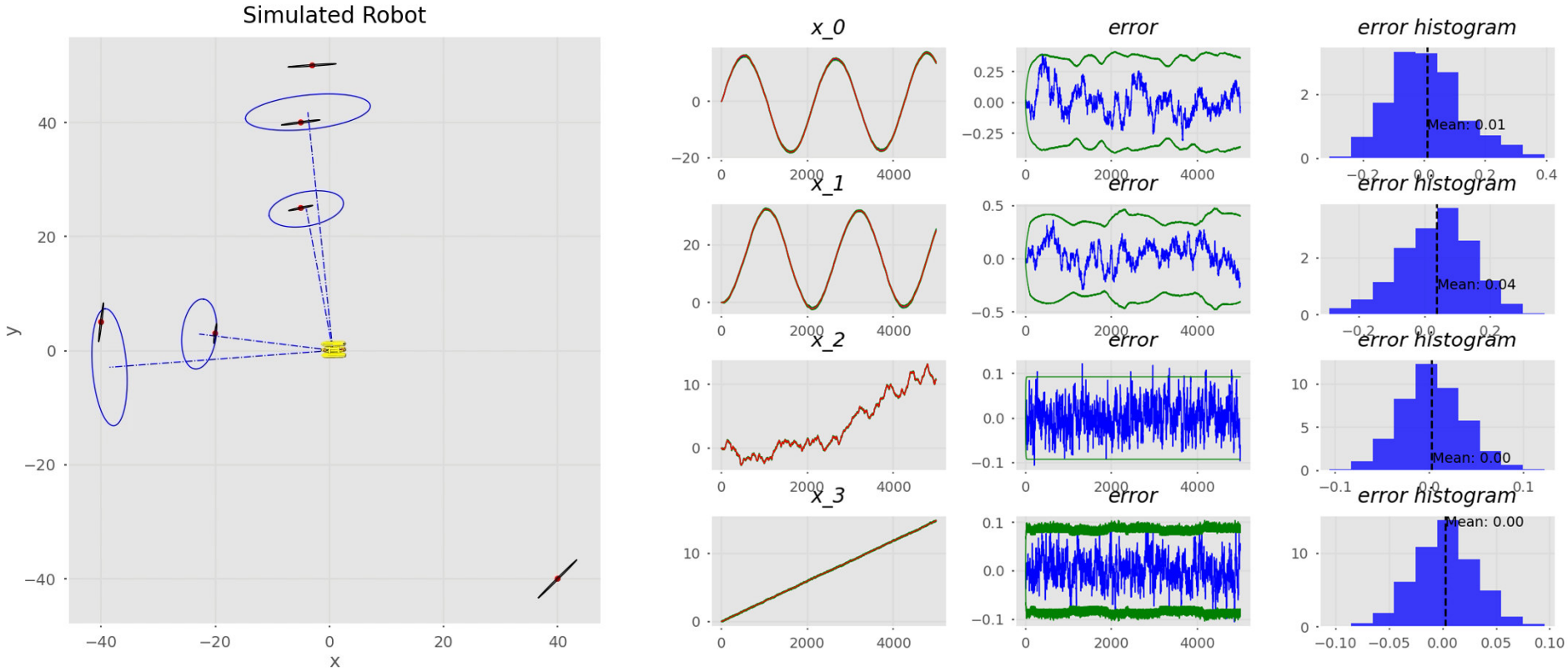
4 DOF AUV Localization

- **Motion Model:** **Input Velocity**
- **Observation Mosel :** **2D Point Features Stored in Cartesian Observed In Polar**

Class $MBL\_4DOFAUV\_InputVelocityMM\_2DCartesianFeaturePolarObservedOM$: $Cartesian2DStoredPolarObservedMapFeature$, $FEKFMBL$, $EKF\_4DOFAUV\_InputVelocityMM\_DVLDepthYawOM$



EKF Map Based Localization



Input

- **DVL:** 1 every 0.1 s, $R_{DVL=}diag([0.1^2, 0.1^2, 0.1^2])$
- **Gyro:** 1 every 0.1 s, σ_{yaw}= 5º/s

Measurement

- **Depth:** 1 every 0.1 s, σ_{depth}= 0.1 m
- **Compass:** None, σ_{yaw}= 1º
- **Feature:** 1 every 5 s, $R_{Fpolar=}diag([1^2\,m, 5^2\,deg^2])$