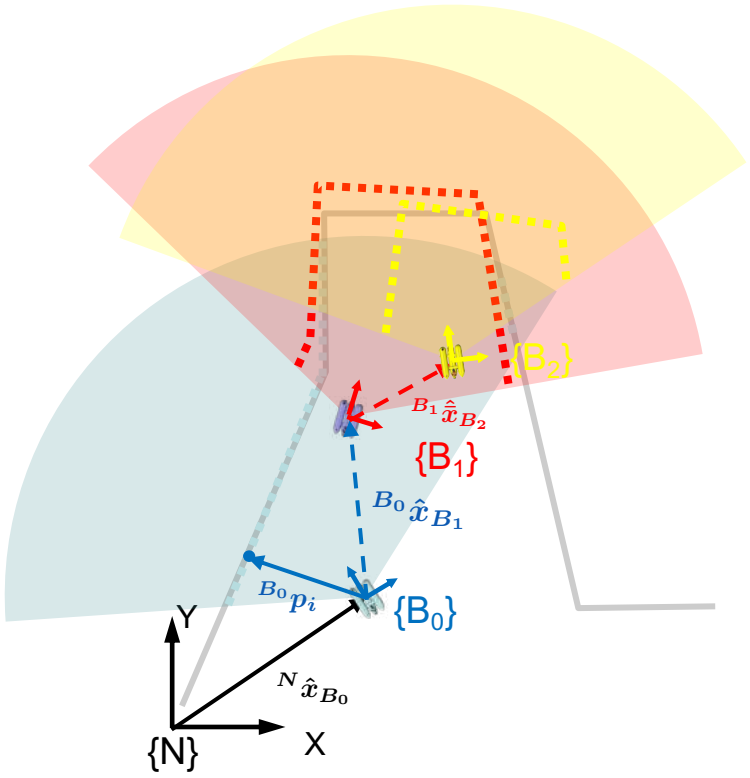
EKF Pose-Based SLAM

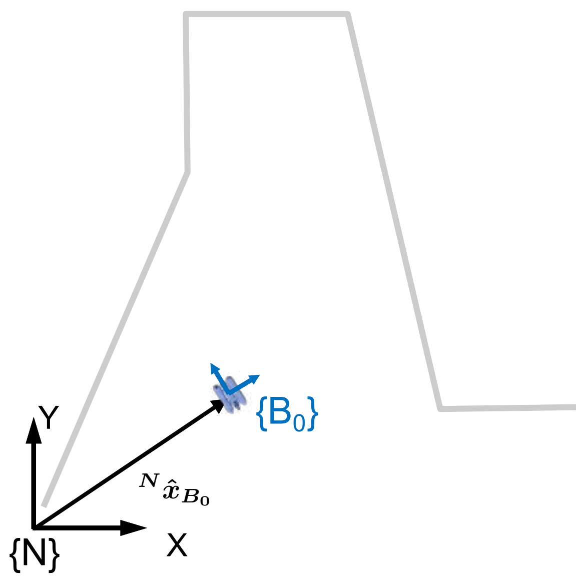


Spatial Relationships

$$ {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{B_0} \\ {}^{N}\hat{\boldsymbol{x}}_{B_1} \\ \vdots \\ {}^{N}\hat{\boldsymbol{x}}_{B_k} \end{bmatrix} \; ; \; {}^{N}\boldsymbol{P}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\boldsymbol{P}_{B_0} & {}^{N}\boldsymbol{P}_{B_0B_1} & \cdots & {}^{N}\boldsymbol{P}_{B_0B_k} \\ {}^{N}\boldsymbol{P}_{B_1B_0} & {}^{N}\boldsymbol{P}_{B_1} & \cdots & {}^{N}\boldsymbol{P}_{B_1B_k} \\ \vdots & \vdots & \ddots & \vdots \\ {}^{N}\boldsymbol{P}_{B_kB_0} & {}^{N}\boldsymbol{P}_{B_kB_1} & \cdots & {}^{N}\boldsymbol{P}_{B_k} \end{bmatrix} $$

$$ \mathcal{M} = \begin{bmatrix} {}^{B_0}\boldsymbol{S}_{\boldsymbol{0}} & {}^{B_1}\boldsymbol{S}_{\boldsymbol{1}} & \cdots & \underbrace{{}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}}_{\left[{}^{B_k}\boldsymbol{p}_0 \; \cdots \; {}^{B_k}\boldsymbol{p}_i \cdots \; {}^{B_k}\boldsymbol{p}_{np}\right]} \end{bmatrix} $$

- A map is represented by:
  - The robot poses referenced to the N-Frame
  - The covariances of the robot poses
  - The scans gathered from the poses
- The map is built incrementally

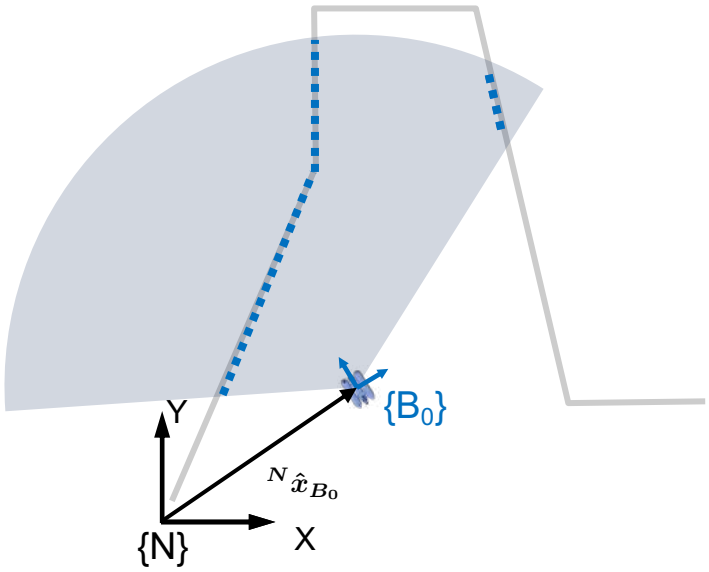


Method Localization($\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0$)

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$; // Read the Scan from the sensor
- $[{}^N\hat{\bar{\boldsymbol{x}}}_0, {}^N\bar{\boldsymbol{P}}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}]]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u}_k, \boldsymbol{Q}_k] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u}_k, \boldsymbol{Q}_k)$;
  - $[\boldsymbol{z}_m, \boldsymbol{R}_m] = GetMeasurement()$; // Read navigation sensors
  - $\boldsymbol{z}_p = [\,]; \boldsymbol{z}_p = [\,]; \mathcal{H}_p = [\,]$;
  - **if** *ScanAvailable* **then**
    - $[{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}] = GetScan()$;
    - $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = AddNewPose({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = [{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}]$; // Store the scan in the map
    - $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{\boldsymbol{x}}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - $c = 1$; $\mathcal{H}_p = [\,]$;
    - **for** $i = 1$ *to length*$(\mathcal{H}_o)$ **do** // for all overlaps
      - $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - $[{}^{B_j}\boldsymbol{S}_j, {}^{B_j}\boldsymbol{R}_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map
      - ${}^{B_j}\boldsymbol{x}_{B_k} = (\ominus {}^N\boldsymbol{x}_{B_j}) \oplus {}^N\boldsymbol{x}_{B_k}$; // Scan Displacement guess
      - ${}^{B_j}\boldsymbol{P}_{B_k} = \boldsymbol{J}_{1\oplus}\boldsymbol{J}_{\ominus} {}^N\boldsymbol{P}_{B_j}\boldsymbol{J}_{\ominus}^T\boldsymbol{J}_{1\oplus}^T + \boldsymbol{J}_{2\oplus} {}^N\boldsymbol{P}_{B_k}\boldsymbol{J}_{2\oplus}^T$
      - $[\boldsymbol{z}_r, \boldsymbol{R}_r] = Register({}^{B_j}\boldsymbol{S}_j, {}^{B_k}\boldsymbol{S}_k, {}^{B_j}\boldsymbol{x}_{B_k})$; // Scan displacement mean & cov
      - **if** $IndividuallyCompatible({}^{B_j}\boldsymbol{x}_{B_k}, {}^{B_j}\boldsymbol{P}_{B_k}, \boldsymbol{z}_r, \boldsymbol{R}_r, \alpha)$ **then**
        - $\boldsymbol{z}_p[c] = \boldsymbol{z}_r$; $\boldsymbol{R}_p[c] = \boldsymbol{R}_r$; // Accepted registration
        - $\mathcal{H}_p[c] = i$; $c = c + 1$;
  - $[\boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z}_m, \boldsymbol{R}_m, \boldsymbol{z}_p, \boldsymbol{R}_p)$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k, \mathcal{H}_p)$;

$$
{}^N\hat{\bar{\boldsymbol{x}}}_0 = \begin{bmatrix} {}^N\hat{\boldsymbol{x}}_{B_0} \\ \\ \\ \end{bmatrix} \;;\; {}^N\bar{\boldsymbol{P}}_2 = \begin{bmatrix} {}^N\boldsymbol{P}_{B_0} & & \\ & & \\ & & \end{bmatrix}
$$

$$
\mathcal{M}_0 = [\qquad]
$$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

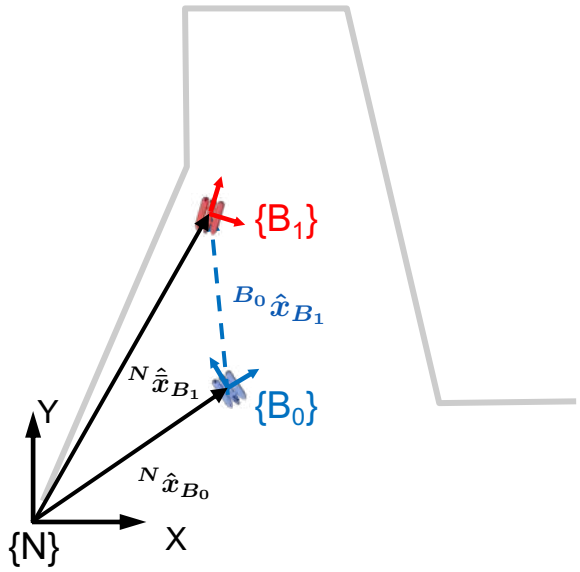
$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{\hat{x}}_0 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_0} \\ \\ \end{bmatrix} \; ; \; {}^N\bar{P}_0 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0} & \\ {}^N P_{B_0} & {}^N P_{B_0} & \\ & & \end{bmatrix} $$

$$ \mathcal{M}_0 = [\,[{}^{B_0}S_0, {}^{B_0}R_{S_0}] \quad] $$



Method `Localization`$(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$

$[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{\boldsymbol{S}_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{\boldsymbol{x}}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{\boldsymbol{S}_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[\boldsymbol{u_k}, \boldsymbol{Q_k}] = GetInput()$; // Get input to the motion model

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

$[\boldsymbol{z_m}, \boldsymbol{R_m}] = GetMeasurement()$; // Read navigation sensors

$\boldsymbol{z_p} = [\,]; \boldsymbol{z_p} = [\,]; \mathcal{H}_p = [\,]$;

if *ScanAvailable* **then**

$[{}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}, {}^{B_k}\boldsymbol{R}_{\boldsymbol{S}_k}] = GetScan()$;

$[{}^N\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, {}^N\bar{\boldsymbol{P}}_{\boldsymbol{k}}] = AddNewPose({}^N\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, {}^N\bar{\boldsymbol{P}}_{\boldsymbol{k}})$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}, {}^{B_k}\boldsymbol{R}_{\boldsymbol{S}_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}\boldsymbol{S_j}, {}^{B_j}\boldsymbol{R}_{\boldsymbol{S}_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}\boldsymbol{x}_{B_k} = (\ominus {}^N\boldsymbol{x}_{B_j}) \oplus {}^N\boldsymbol{x}_{B_k}$; // Scan Displacement guess

${}^{B_j}\boldsymbol{P}_{B_k} = \boldsymbol{J_{1\oplus}} \boldsymbol{J_{\ominus}} {}^N\boldsymbol{P}_{B_j} \boldsymbol{J_{\ominus}^T} \boldsymbol{J_{1\oplus}^T} + \boldsymbol{J_{2\oplus}} {}^N\boldsymbol{P}_{B_k} \boldsymbol{J_{2\oplus}^T}$

$[\boldsymbol{z_r}, \boldsymbol{R_r}] = Register({}^{B_j}\boldsymbol{S_j}, {}^{B_k}\boldsymbol{S_k}, {}^{B_j}\boldsymbol{x}_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}\boldsymbol{x}_{B_k}, {}^{B_j}\boldsymbol{P}_{B_k}, \boldsymbol{z_r}, \boldsymbol{R_r}, \alpha)$ **then**

$\boldsymbol{z_p}[c] = \boldsymbol{z_r}$; $\boldsymbol{R_p}[c] = \boldsymbol{R_r}$; // Accepted registration

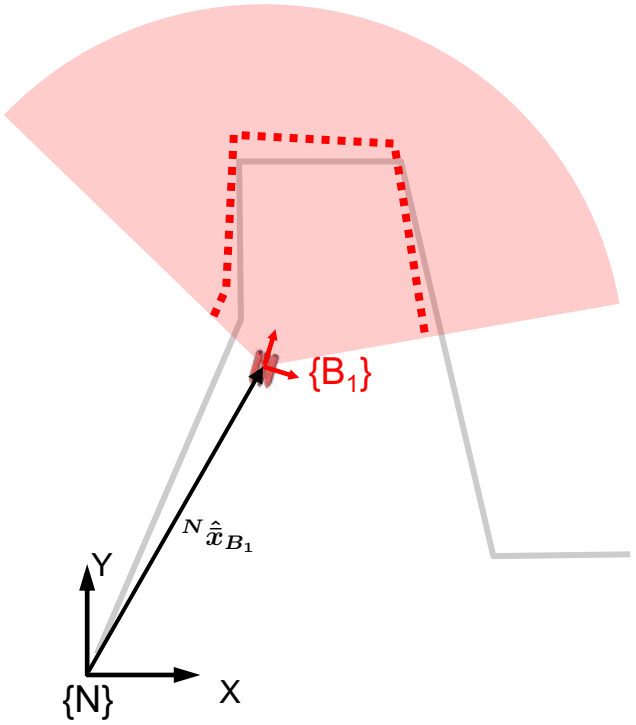
$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$;

$[{}^N\hat{\boldsymbol{x}}_{\boldsymbol{k}}, {}^N\boldsymbol{P_k}] = Update({}^N\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, {}^N\bar{\boldsymbol{P}}_{\boldsymbol{k}}, \boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \mathcal{H}_p)$;

$$ {}^N\hat{\bar{\boldsymbol{x}}}_1 = \begin{bmatrix} {}^N\hat{\boldsymbol{x}}_{B_0} \\ {}^N\hat{\bar{\boldsymbol{x}}}_{B_1} \end{bmatrix} \; ; \; {}^N\bar{\boldsymbol{P}}_1 = \begin{bmatrix} {}^N\boldsymbol{P}_{B_0} & {}^N\bar{\boldsymbol{P}}_{B_0B_1} \\ {}^N\boldsymbol{P}_{B_1B_0} & {}^N\boldsymbol{P}_{B_1} \end{bmatrix} $$

$$ \mathcal{M}_1 = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{\boldsymbol{S}_0}] \quad] \quad \boldsymbol{z_m} \quad \boldsymbol{R_m} $$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1$; $\mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus}\, {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus}\, {}^N P_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

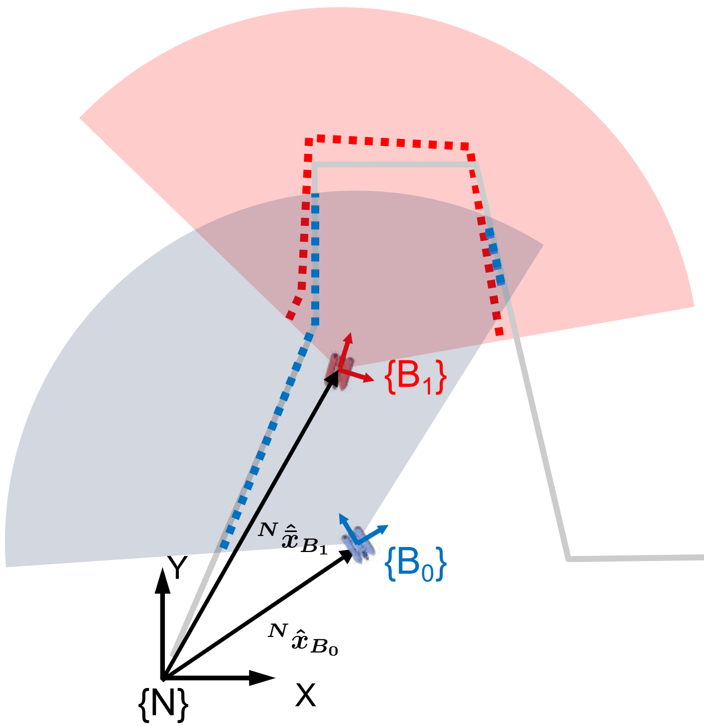
        $\mathcal{H}_p[c] = i$; $c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$
{}^N\hat{\hat{x}}_1 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{\hat{x}}_{B_1} \\ {}^N\hat{\hat{x}}_{B_1} \end{bmatrix} \quad ; \quad {}^N\bar{P}_1 = \begin{bmatrix} {}^N P_{B_0} & {}^N\bar{P}_{B_0B_1} & {}^N\bar{P}_{B_0B_1} \\ {}^N\bar{P}_{B_1B_0} & {}^N\bar{P}_{B_1} & {}^N\bar{P}_{B_1} \\ {}^N\bar{P}_{B_1B_0} & {}^N\bar{P}_{B_1} & {}^N\bar{P}_{B_1} \end{bmatrix}
$$

$$
\mathcal{M}_1 = \begin{bmatrix} [{}^{B_0}S_0, {}^{B0}R_{S_0}] & [{}^{B_1}S_1, {}^{B1}R_{S_1}] &] \end{bmatrix} \quad [z_m \quad R_m]
$$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1$; $\mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^N P_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then** // Accepted registration

        $z_p[c] = z_r$; $R_p[c] = R_r$;

        $\mathcal{H}_p[c] = i$; $c = c + 1$;

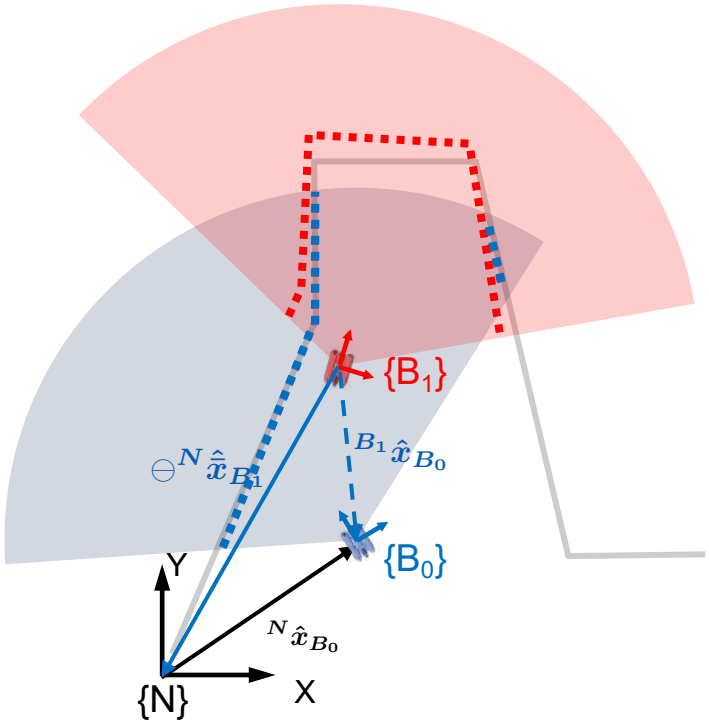
  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$
{}^N\hat{x}_1 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{x}_{B_1} \end{bmatrix} \; ; \; {}^N P_1 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N P_{B_0 B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \end{bmatrix}
$$

$$
\mathcal{M}_1 = [[{}^{B_0}S_0, {}^{B0}R_{S_0}]\ [{}^{B_1}S_1, {}^{B1}R_{S_1}]\quad] \quad [z_m \quad R_m]
$$

$$
\mathcal{H}_p = [0 \quad]
$$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{x}_1 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{x}_{B_1} \end{bmatrix} \; ; \; {}^N P_1 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N P_{B_0 B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \end{bmatrix} $$

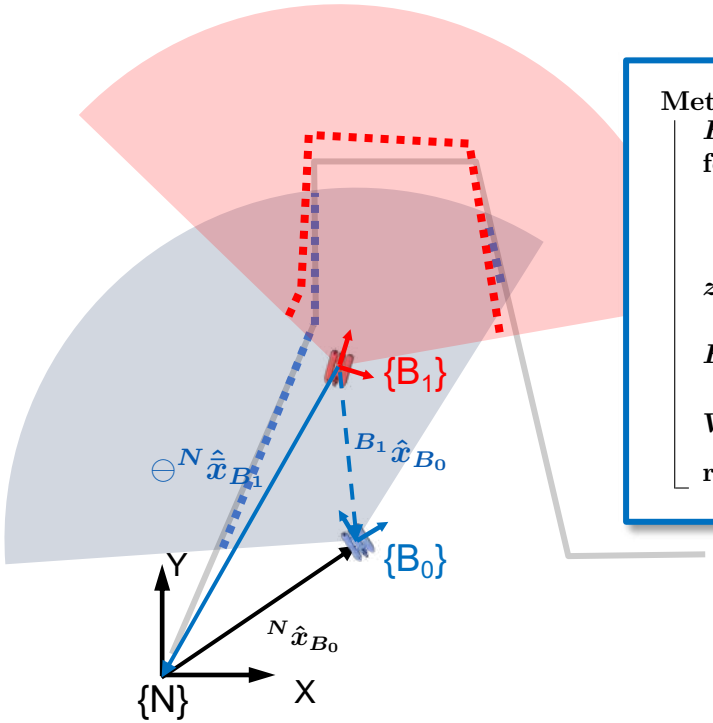
$$ \mathcal{M}_1 = \begin{bmatrix} [{}^{B_0}S_0, {}^{B0}R_{S_0}] & [{}^{B_1}S_1, {}^{B1}R_{S_1}] & & \end{bmatrix} \quad [z_m \quad R_m] $$

$$ \mathcal{H}_p = \begin{bmatrix} 0 & \end{bmatrix} $$

$$ {}^{B_1}x_{B_0} = (\ominus {}^N x_{B_2}) \oplus {}^N x_{B_0}) $$

$$ [z_{p_1}, R_{p_1}] = Register({}^{B_1}S_1, {}^{B_0}S_0, {}^{B_2}x_{B_0}) $$

$$ z_p = \begin{bmatrix} z_{p_1} \end{bmatrix} \; ; \; R_p = \begin{bmatrix} R_{p_1} \end{bmatrix} $$



Method `Localization`$(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$; // Read the Scan from the sensor
- $[{}^N\hat{\bar{\boldsymbol{x}}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}]]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u}_k, \boldsymbol{Q}_k] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u}_k, \boldsymbol{Q}_k)$;
  - $[\boldsymbol{z}_m, \boldsymbol{R}_m] = GetMeasurement()$; // Read navigation sensors
  - $\mathcal{H}_p[c] = i;\ c = c + 1$;
  - $[\boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z}_m, \boldsymbol{R}_m, \boldsymbol{z}_p, \boldsymbol{R}_p)$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k, \mathcal{H}_p)$;

Method `ObservationMatrix`$(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z}_m, \boldsymbol{R}_m, \boldsymbol{z}_p, \boldsymbol{R}_p)$

- $\boldsymbol{H}_p = \boldsymbol{0}$;
- **for** $i = 1$ *to* $length(\boldsymbol{z}_p)$ **do**
  - $\boldsymbol{H}_p[i, 1] = \boldsymbol{J}_{1\oplus}(\ominus {}^N\hat{\bar{\boldsymbol{x}}}_{B_k}, {}^N\hat{\boldsymbol{x}}_k[\mathcal{H}_p[i]])\boldsymbol{J}_{\ominus}({}^N\hat{\bar{\boldsymbol{x}}}_{B_k})$;
  - $\boldsymbol{H}_p[i, \mathcal{H}_p[i]] = \boldsymbol{J}_{2\oplus}({}^N\hat{\bar{\boldsymbol{x}}}_{B_k})$;
- $\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{z}_m \\ \boldsymbol{z}_p \end{bmatrix}$; $\boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_p \end{bmatrix}$; // Indep. measure&displacement noise
- $\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{J}_{h_x}(\hat{\bar{\boldsymbol{x}}}_k) \\ \boldsymbol{H}_p \end{bmatrix}$; // Stack measurements and Displ. observ.
- $\boldsymbol{V}_k = \begin{bmatrix} \boldsymbol{J}_{h_v}(\hat{\bar{\boldsymbol{x}}}_k) \\ \boldsymbol{V}_p \end{bmatrix}$;
- **return** $[\boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k]$;

$$
{}^N\hat{\boldsymbol{x}}_1 = \begin{bmatrix} {}^N\hat{\boldsymbol{x}}_{B_0} \\ {}^N\hat{\boldsymbol{x}}_{B_1} \\ {}^N\hat{\boldsymbol{x}}_{B_1} \end{bmatrix} \; ; \; {}^N\boldsymbol{P}_1 = \begin{bmatrix} {}^N\boldsymbol{P}_{B_0} & {}^N\boldsymbol{P}_{B_0B_1} & {}^N\boldsymbol{P}_{B_0B_1} \\ {}^N\boldsymbol{P}_{B_1B_0} & {}^N\boldsymbol{P}_{B_1} & {}^N\boldsymbol{P}_{B_1} \\ {}^N\boldsymbol{P}_{B_1B_0} & {}^N\boldsymbol{P}_{B_1} & {}^N\boldsymbol{P}_{B_1} \end{bmatrix}
$$

$$
\mathcal{M}_1 = [[{}^{B_0}\boldsymbol{S}_0, {}^{B0}\boldsymbol{R}_{S_0}]\ [{}^{B_1}\boldsymbol{S}_1, {}^{B1}\boldsymbol{R}_{S_1}]\ \ ]\quad [\boldsymbol{z}_m \quad \boldsymbol{R}_m]
$$

$$
\mathcal{H}_p = [0 \quad]
$$

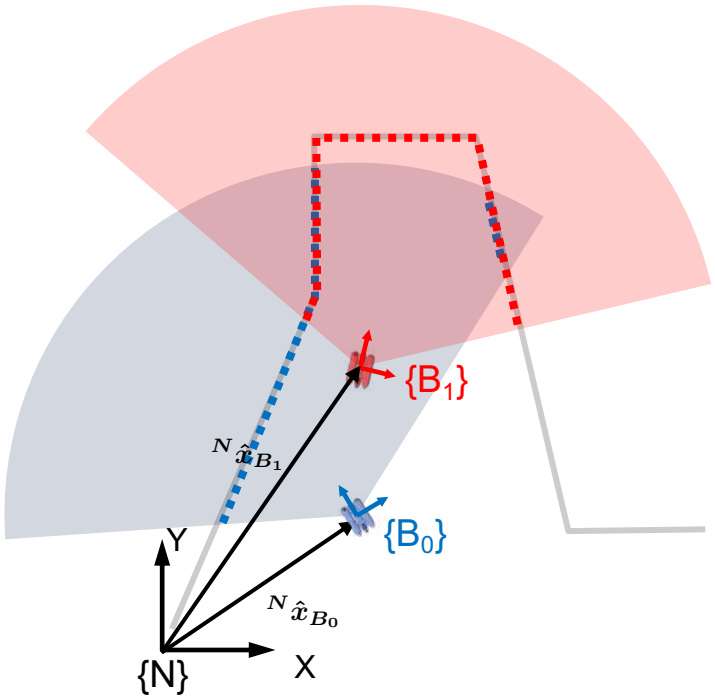
$$
{}^{B_1}\boldsymbol{x}_{B_0} = (\ominus {}^N\boldsymbol{x}_{B_2}) \oplus {}^N\boldsymbol{x}_{B_0})
$$

$$
[\boldsymbol{z}_{p_1}, \boldsymbol{R}_{p_1}] = Register({}^{B_1}\boldsymbol{S}_1, {}^{B_0}\boldsymbol{S}_0, {}^{B_2}\boldsymbol{x}_{B_0})
$$

$$
\boldsymbol{z}_p = [\boldsymbol{z}_{p_1}] \; ; \; \boldsymbol{R}_p = [\boldsymbol{R}_{p_1}]
$$

$$
\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{z}_m \\ \boldsymbol{z}_p \end{bmatrix} \; ; \; \boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{R}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_p \end{bmatrix}
$$

$$
\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{H}_m & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_p \end{bmatrix} \; ; \; \boldsymbol{V}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}
$$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus}\, {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus}\, {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{x}_1 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{x}_{B_1} \end{bmatrix} \; ; \; {}^N P_1 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N P_{B_0 B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N P_{B_1} \end{bmatrix} $$

$$ \mathcal{M}_1 = \begin{bmatrix} [{}^{B_0}S_0, {}^{B0}R_{S_0}] & [{}^{B_1}S_1, {}^{B1}R_{S_1}] & \end{bmatrix} \quad [z_m \quad R_m] $$

$$ \mathcal{H}_p = \begin{bmatrix} 0 & \end{bmatrix} $$

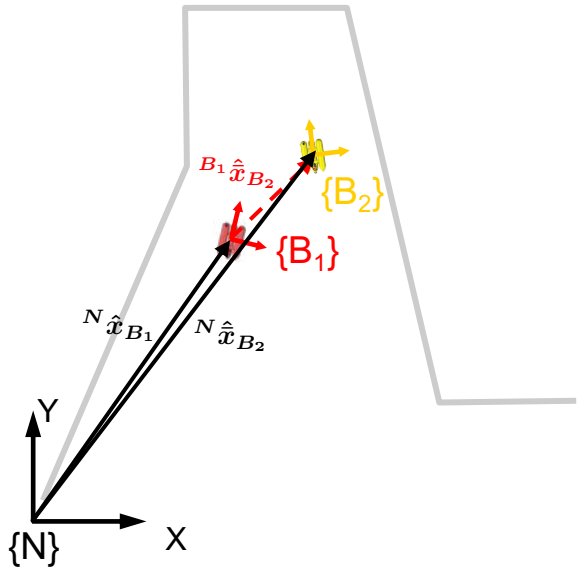
$$ {}^{B_1}x_{B_0} = (\ominus {}^N x_{B_2}) \oplus {}^N x_{B_0}) $$

$$ [z_{p_1}, R_{p_1}] = Register({}^{B_1}S_1, {}^{B_0}S_0, {}^{B_2}x_{B_0}) $$

$$ z_p = [z_{p_1}] \; ; \; R_p = [R_{p_1}] $$

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

$$ H_k = \begin{bmatrix} H_m & 0 \\ 0 & H_p \end{bmatrix} \; ; \; V_k = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} $$



Method $\texttt{Localization}(\hat{x}_0, P_0)$

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus}\, {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus}\, {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

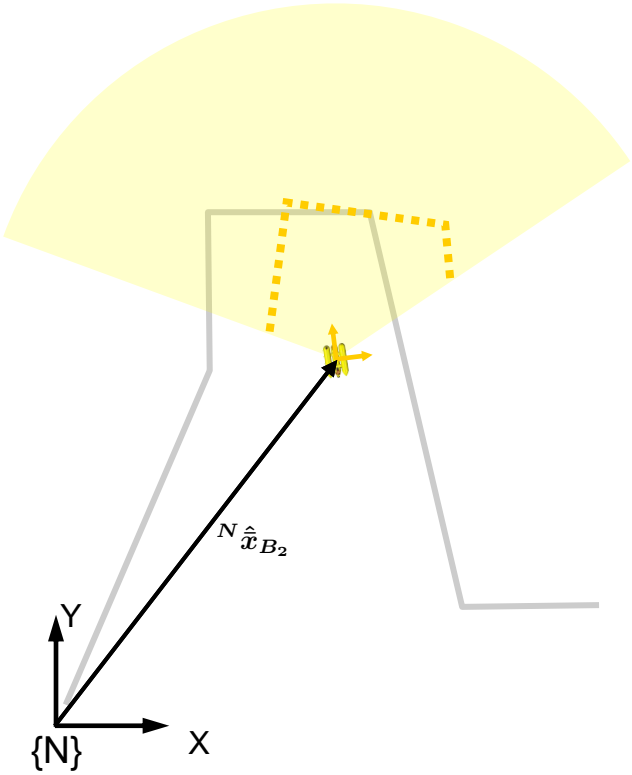
$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{\hat{x}}_2 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{\hat{x}}_{B_2} \end{bmatrix} \; ; \; {}^N\bar{P}_2 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N\bar{P}_{B_0 B_2} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N\bar{P}_{B_1, B_2} \\ {}^N\bar{P}_{B_2 B_0} & {}^N\bar{P}_{B_2 B_1} & {}^N\bar{P}_{B_2} \end{bmatrix} $$

$$ \mathcal{M}_2 = [[{}^{B_0}S_0, {}^{B0}R_{S_0}]\ [{}^{B_1}S_1, {}^{B1}R_{S_1}]\ \ \ ] \quad z_m \quad R_m $$

$$ \mathcal{H}_p = [0 \quad] $$



Method `Localization`$(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$

$[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[\boldsymbol{u}_k, \boldsymbol{Q}_k] = GetInput()$; // Get input to the motion model

  $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u}_k, \boldsymbol{Q}_k)$;

  $[\boldsymbol{z}_m, \boldsymbol{R}_m] = GetMeasurement()$; // Read navigation sensors

  $\boldsymbol{z}_p = [\,]; \boldsymbol{z}_p = [\,]; \mathcal{H}_p = [\,]$;

  if *ScanAvailable* **then**

    $[{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = AddNewPose({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{\boldsymbol{x}}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1$; $\mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}\boldsymbol{S}_j, {}^{B_j}\boldsymbol{R}_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}\boldsymbol{x}_{B_k} = (\ominus {}^N\boldsymbol{x}_{B_j}) \oplus {}^N\boldsymbol{x}_{B_k}$; // Scan Displacement guess

      ${}^{B_j}\boldsymbol{P}_{B_k} = \boldsymbol{J_{1\oplus}} \boldsymbol{J_\ominus} {}^N\boldsymbol{P}_{B_j} \boldsymbol{J}_\ominus^T \boldsymbol{J}_{1\oplus}^T + \boldsymbol{J_{2\oplus}} {}^N\boldsymbol{P}_{B_k} \boldsymbol{J}_{2\oplus}^T$

      $[\boldsymbol{z}_r, \boldsymbol{R}_r] = Register({}^{B_j}\boldsymbol{S}_j, {}^{B_k}\boldsymbol{S}_k, {}^{B_j}\boldsymbol{x}_{B_k})$; // Scan displacement mean & cov

      if $IndividuallyCompatible({}^{B_j}\boldsymbol{x}_{B_k}, {}^{B_j}\boldsymbol{P}_{B_k}, \boldsymbol{z}_r, \boldsymbol{R}_r, \alpha)$ **then**

        $\boldsymbol{z_p}[c] = \boldsymbol{z_r}$; $\boldsymbol{R_p}[c] = \boldsymbol{R_r}$; // Accepted registration

        $\mathcal{H}_p[c] = i$; $c = c + 1$;

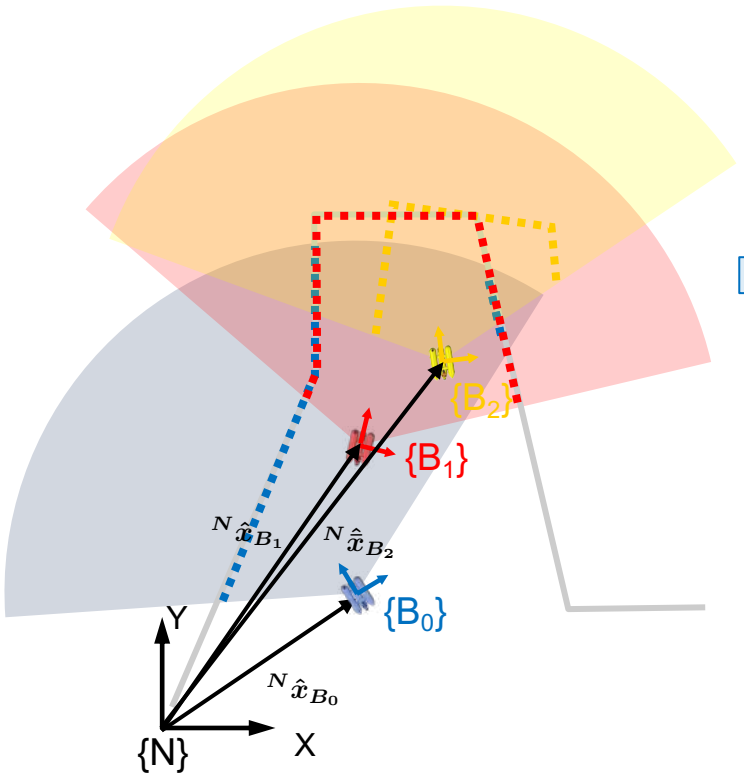
  $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$;

  $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k, \mathcal{H}_p)$;

$$
{}^N\hat{\bar{\boldsymbol{x}}}_2 = \begin{bmatrix} {}^N\hat{\boldsymbol{x}}_{B_0} \\ {}^N\hat{\boldsymbol{x}}_{B_1} \\ {}^N\hat{\bar{\boldsymbol{x}}}_{B_2} \\ {}^N\hat{\bar{\boldsymbol{x}}}_{B_2} \end{bmatrix} \quad ; \quad {}^N\bar{\boldsymbol{P}}_2 = \begin{bmatrix} {}^N\boldsymbol{P}_{B_0} & {}^N\boldsymbol{P}_{B_0B_1} & {}^N\bar{\boldsymbol{P}}_{B_0B_2} & {}^N\bar{\boldsymbol{P}}_{B_0B_2} \\ {}^N\boldsymbol{P}_{B_1B_0} & {}^N\boldsymbol{P}_{B_1} & {}^N\bar{\boldsymbol{P}}_{B_1,B_2} & {}^N\bar{\boldsymbol{P}}_{B_1B_2} \\ {}^N\bar{\boldsymbol{P}}_{B_2B_0} & {}^N\bar{\boldsymbol{P}}_{B_2B_1} & {}^N\bar{\boldsymbol{P}}_{B_2} & {}^N\bar{\boldsymbol{P}}_{B_2} \\ {}^N\bar{\boldsymbol{P}}_{B_2B_0} & {}^N\bar{\boldsymbol{P}}_{B_2B_1} & {}^N\bar{\boldsymbol{P}}_{B_2} & {}^N\bar{\boldsymbol{P}}_{B_2} \end{bmatrix}
$$

$$
\mathcal{M}_2 = \left[[{}^{B_0}\boldsymbol{S}_0, {}^{B0}\boldsymbol{R}_{S_0}] \; [{}^{B_1}\boldsymbol{S}_1, {}^{B1}\boldsymbol{R}_{S_1}] \; [{}^{B_2}\boldsymbol{S}_2, {}^{B2}\boldsymbol{R}_{S_2}] \right] \quad [\boldsymbol{z}_m \quad \boldsymbol{R}_m]
$$

$$
\mathcal{H}_p = \begin{bmatrix} 0 & \end{bmatrix}
$$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus}\, {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus}\, {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

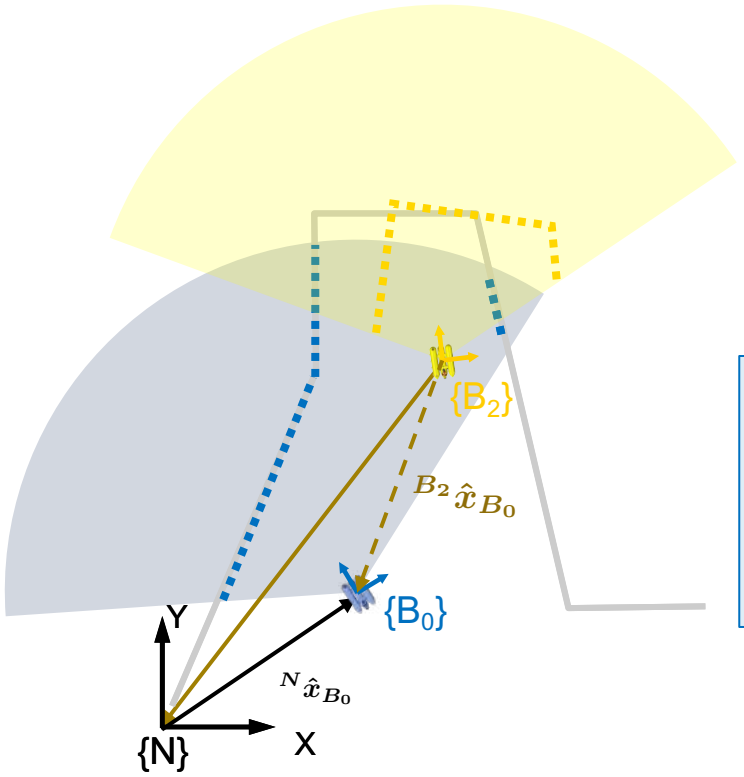
$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{\hat{x}}_2 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{\hat{x}}_{B_2} \\ {}^N\hat{\hat{x}}_{B_2} \end{bmatrix} \; ; \; {}^N\bar{P}_2 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N\bar{P}_{B_0 B_2} & {}^N\bar{P}_{B_0 B_2} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N\bar{P}_{B_1, B_2} & {}^N\bar{P}_{B_1 B_2} \\ {}^N\bar{P}_{B_2 B_0} & {}^N\bar{P}_{B_2 B_1} & {}^N\bar{P}_{B_2} & {}^N\bar{P}_{B_2} \\ {}^N\bar{P}_{B_2 B_0} & {}^N\bar{P}_{B_2 B_1} & {}^N\bar{P}_{B_2} & {}^N\bar{P}_{B_2} \end{bmatrix} $$

$$ \mathcal{M}_2 = \left[\left[{}^{B_0}S_0, {}^{B0}R_{S_0}\right] \left[{}^{B_1}S_1, {}^{B1}R_{S_1}\right] \left[{}^{B_2}S_2, {}^{B2}R_{S_2}\right]\right] \quad \left[z_m \quad R_m\right] $$

$$ \mathcal{H}_p = \begin{bmatrix} 0 & \boxed{1} \end{bmatrix} $$



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

$[{}^N\hat{x}_0, {}^N P_0] = [{}^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^N P_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^N P_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$ {}^N\hat{\hat{x}}_2 = \begin{bmatrix} {}^N\hat{x}_{B_0} \\ {}^N\hat{x}_{B_1} \\ {}^N\hat{\hat{x}}_{B_2} \\ {}^N\hat{\hat{x}}_{B_2} \end{bmatrix} \; ; \; {}^N\bar{P}_2 = \begin{bmatrix} {}^N P_{B_0} & {}^N P_{B_0 B_1} & {}^N\bar{P}_{B_0 B_2} & {}^N\bar{P}_{B_0 B_2} \\ {}^N P_{B_1 B_0} & {}^N P_{B_1} & {}^N\bar{P}_{B_1, B_2} & {}^N\bar{P}_{B_1 B_2} \\ {}^N\bar{P}_{B_2 B_0} & {}^N\bar{P}_{B_2 B_1} & {}^N\bar{P}_{B_2} & {}^N\bar{P}_{B_2} \\ {}^N\bar{P}_{B_2 B_0} & {}^N\bar{P}_{B_2 B_1} & {}^N\bar{P}_{B_2} & {}^N\bar{P}_{B_2} \end{bmatrix} $$

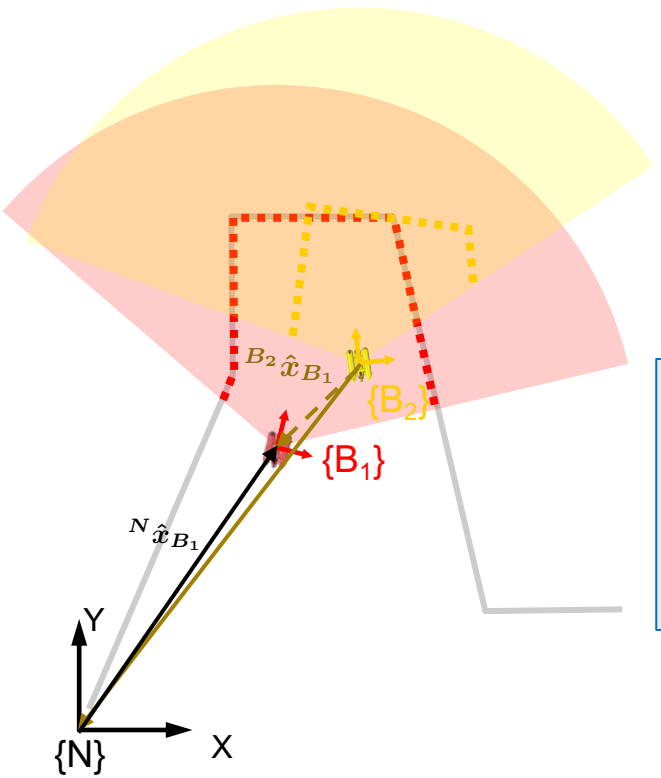
$$ \mathcal{M}_2 = \begin{bmatrix} [{}^{B_0}S_0, {}^{B0}R_{S_0}] & [{}^{B_1}S_1, {}^{B1}R_{S_1}] & [{}^{B_2}S_2, {}^{B2}R_{S_2}] \end{bmatrix} \quad \begin{bmatrix} z_m & R_m \end{bmatrix} $$

$$ \mathcal{H}_p = \begin{bmatrix} 0 & 1 \end{bmatrix} $$

$${}^{B_2}x_{B_0} = (\ominus {}^N x_{B_2}) \oplus {}^N x_{B_0}) \; ;$$

$$ [z_{p_1}, R_{p_1}] = Register({}^{B_2}S_2, {}^{B_0}S_0, {}^{B_2}x_{B_0}) \; ; $$

$$ z_p = \begin{bmatrix} z_{p_1} & z_{p_2} \end{bmatrix} \; ; $$



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

$[^N\hat{x}_0, {}^N P_0] = [^N\hat{x}_{B_0}, {}^N P_{B_0}]$; // Initialize SLAM state vector

$[^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[^N\hat{\bar{x}}_0, {}^N P_0] = AddNewPose(^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* **then**

$[^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose(^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans(^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1$; $\mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: $^{B_j}S_j$ overlaps $^{B_k}S_k$

$[^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

$^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scan Displacement guess

$^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^N P_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^N P_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register(^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

if $IndividuallyCompatible(^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i$; $c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

$[^N\hat{x}_k, {}^N P_k] = Update(^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$^N\hat{\bar{x}}_2 = \begin{bmatrix} ^N\hat{x}_{B_0} \\ ^N\hat{x}_{B_1} \\ ^N\hat{\bar{x}}_{B_2} \\ ^N\hat{\bar{x}}_{B_2} \end{bmatrix} \; ; \; {}^N\bar{P}_2 = \begin{bmatrix} ^N P_{B_0} & ^N P_{B_0B_1} & ^N\bar{P}_{B_0B_2} & ^N\bar{P}_{B_0B_2} \\ ^N P_{B_1B_0} & ^N P_{B_1} & ^N\bar{P}_{B_1,B_2} & ^N\bar{P}_{B_1B_2} \\ ^N\bar{P}_{B_2B_0} & ^N\bar{P}_{B_2B_1} & ^N\bar{P}_{B_2} & ^N\bar{P}_{B_2} \\ ^N\bar{P}_{B_2B_0} & ^N\bar{P}_{B_2B_1} & ^N\bar{P}_{B_2} & ^N\bar{P}_{B_2} \end{bmatrix}$$

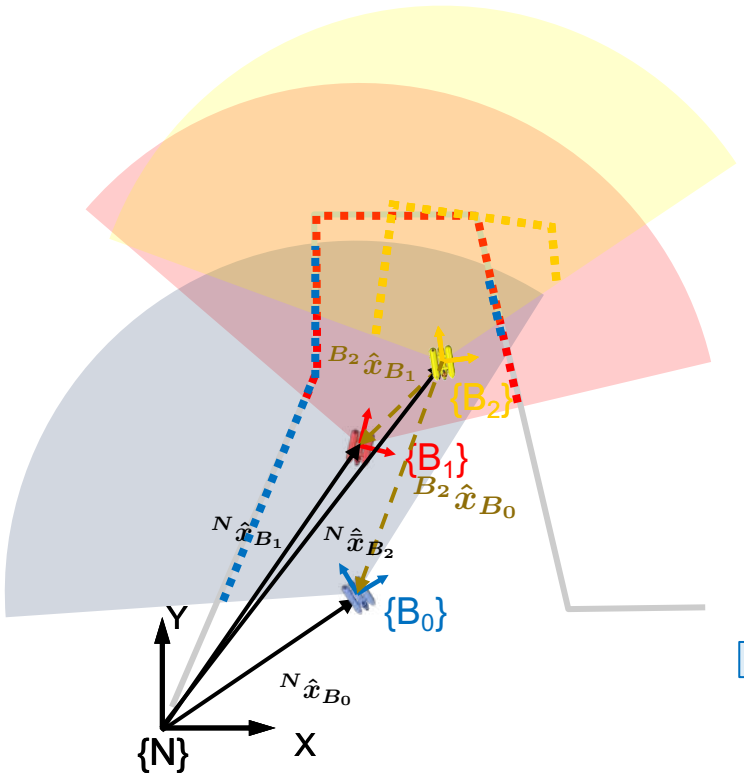
$$\mathcal{M}_2 = [[^{B_0}S_0, {}^{B0}R_{S_0}] \; [^{B_1}S_1, {}^{B1}R_{S_1}] \; [^{B_2}S_2, {}^{B2}R_{S_2}]] \quad [z_m \quad R_m]$$

$$\mathcal{H}_p = [0 \quad 1]$$

$$^{B_2}x_{B_0} = (\ominus {}^N x_{B_2}) \oplus {}^N x_{B_0}) \; ; \quad {}^{B_2}x_{B_1} = (\ominus {}^N x_{B_2}) \oplus {}^N x_{B_1})$$

$$[z_{p_1}, R_{p_1}] = Register(^{B_2}S_2, {}^{B_0}S_0, {}^{B_2}x_{B_0}) \; ; \; [z_{p_2}, R_{p_2}] = Register(^{B_2}S_2, {}^{B_1}S_1, {}^{B_2}x_{B_1})$$

$$z_p = [z_{p_1} \quad z_{p_2}] \; ; \; R_p = [R_{p_1} \quad R_{p_2}]$$



Method `Localization`$(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$; // Read the Scan from the sensor
- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}]]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u}_k, \boldsymbol{Q}_k] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u}_k, \boldsymbol{Q}_k)$;
  - $[\boldsymbol{z}_m, \boldsymbol{R}_m] = GetMeasurement()$; // Read navigation sensors
  - $\boldsymbol{z}_p = [\,]; \boldsymbol{z}_p = [\,]; \mathcal{H}_p = [\,]$;
  - **if** *ScanAvailable* **then**
    - $[{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}] = GetScan()$;
    - $[{}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k] = AddNewPose({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = [{}^{B_k}\boldsymbol{S}_k, {}^{B_k}\boldsymbol{R}_{S_k}]$; // Store the scan in the map
    - $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{\boldsymbol{x}}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - $c = 1$; $\mathcal{H}_p = [\,]$;
    - **for** $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps
      - $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - $[{}^{B_j}\boldsymbol{S}_j, {}^{B_j}\boldsymbol{R}_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map
      - ${}^{B_j}\boldsymbol{x}_{B_k} = (\ominus {}^N\boldsymbol{x}_{B_j}) \oplus {}^N\boldsymbol{x}_{B_k}$; // Scan Displacement guess
      - ${}^{B_j}\boldsymbol{P}_{B_k} = \boldsymbol{J}_{1\oplus}\boldsymbol{J}_{\ominus}\,{}^N\boldsymbol{P}_{B_j}\boldsymbol{J}_{\ominus}^T\boldsymbol{J}_{1\oplus}^T + \boldsymbol{J}_{2\oplus}\,{}^N\boldsymbol{P}_{B_k}\boldsymbol{J}_{2\oplus}^T$
      - $[\boldsymbol{z}_r, \boldsymbol{R}_r] = Register({}^{B_j}\boldsymbol{S}_j, {}^{B_k}\boldsymbol{S}_k, {}^{B_j}\boldsymbol{x}_{B_k})$; // Scan displacement mean & cov
      - **if** $IndividuallyCompatible({}^{B_j}\boldsymbol{x}_{B_k}, {}^{B_j}\boldsymbol{P}_{B_k}, \boldsymbol{z}_r, \boldsymbol{R}_r, \alpha)$ **then**
        - $\boldsymbol{z}_p[c] = \boldsymbol{z}_r$; $\boldsymbol{R}_p[c] = \boldsymbol{R}_r$; // Accepted registration
        - $\mathcal{H}_p[c] = i$; $c = c + 1$;
  - $[\boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z}_m, \boldsymbol{R}_m, \boldsymbol{z}_p, \boldsymbol{R}_p)$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\bar{\boldsymbol{x}}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z}_k, \boldsymbol{R}_k, \boldsymbol{H}_k, \boldsymbol{V}_k, \mathcal{H}_p)$;

$$
{}^N\hat{\bar{\boldsymbol{x}}}_2 = \begin{bmatrix} {}^N\hat{\boldsymbol{x}}_{B_0} \\ {}^N\hat{\boldsymbol{x}}_{B_1} \\ {}^N\hat{\bar{\boldsymbol{x}}}_{B_2} \\ {}^N\hat{\bar{\boldsymbol{x}}}_{B_2} \end{bmatrix} \; ; \; {}^N\bar{\boldsymbol{P}}_2 = \begin{bmatrix} {}^N\boldsymbol{P}_{B_0} & {}^N\boldsymbol{P}_{B_0B_1} & {}^N\bar{\boldsymbol{P}}_{B_0B_2} & {}^N\bar{\boldsymbol{P}}_{B_0B_2} \\ {}^N\boldsymbol{P}_{B_1B_0} & {}^N\boldsymbol{P}_{B_1} & {}^N\bar{\boldsymbol{P}}_{B_1,B_2} & {}^N\bar{\boldsymbol{P}}_{B_1B_2} \\ {}^N\bar{\boldsymbol{P}}_{B_2B_0} & {}^N\bar{\boldsymbol{P}}_{B_2B_1} & {}^N\bar{\boldsymbol{P}}_{B_2} & {}^N\bar{\boldsymbol{P}}_{B_2} \\ {}^N\bar{\boldsymbol{P}}_{B_2B_0} & {}^N\bar{\boldsymbol{P}}_{B_2B_1} & {}^N\bar{\boldsymbol{P}}_{B_2} & {}^N\bar{\boldsymbol{P}}_{B_2} \end{bmatrix}
$$

$$
\mathcal{M}_2 = \left[\left[{}^{B_0}\boldsymbol{S}_0, {}^{B0}\boldsymbol{R}_{S_0}\right]\left[{}^{B_1}\boldsymbol{S}_1, {}^{B1}\boldsymbol{R}_{S_1}\right]\left[{}^{B_2}\boldsymbol{S}_2, {}^{B2}\boldsymbol{R}_{S_2}\right]\right] \quad \left[\boldsymbol{z}_m \quad \boldsymbol{R}_m\right]
$$

$$
\mathcal{H}_p = \begin{bmatrix} 0 & 1 \end{bmatrix}
$$

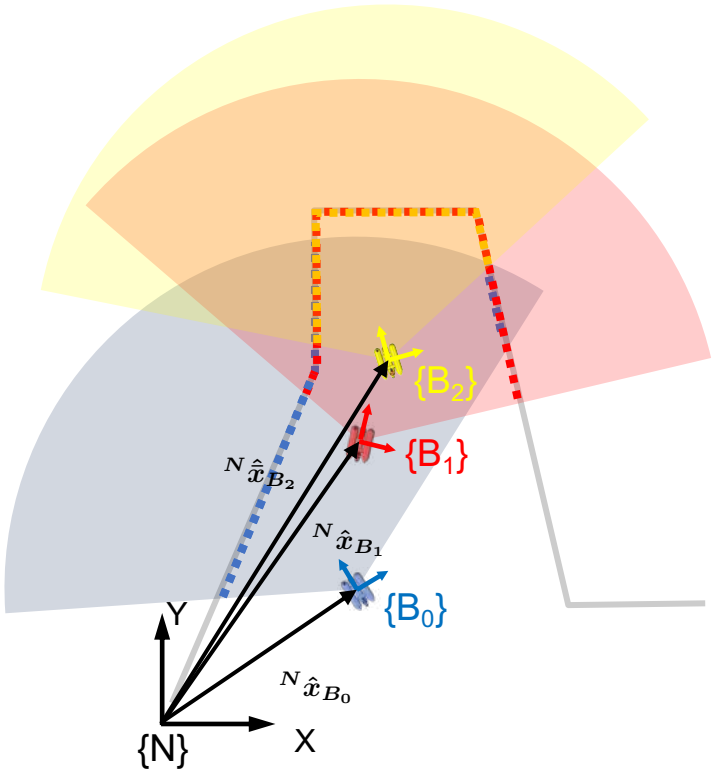
$$
{}^{B_2}\boldsymbol{x}_{B_0} = (\ominus {}^N\boldsymbol{x}_{B_2}) \oplus {}^N\boldsymbol{x}_{B_0}) \; ; \quad {}^{B_2}\boldsymbol{x}_{B_1} = (\ominus {}^N\boldsymbol{x}_{B_2}) \oplus {}^N\boldsymbol{x}_{B_1})
$$

$$
[\boldsymbol{z}_{p_1}, \boldsymbol{R}_{p_1}] = Register({}^{B_2}\boldsymbol{S}_2, {}^{B_0}\boldsymbol{S}_0, {}^{B_2}\boldsymbol{x}_{B_0}) \; ; \; [\boldsymbol{z}_{p_2}, \boldsymbol{R}_{p_2}] = Register({}^{B_2}\boldsymbol{S}_2, {}^{B_1}\boldsymbol{S}_1, {}^{B_2}\boldsymbol{x}_{B_1})
$$

$$
\boldsymbol{z}_p = \begin{bmatrix} \boldsymbol{z}_{p_1} & \boldsymbol{z}_{p_2} \end{bmatrix} \; ; \; \boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{R}_{p_1} & \boldsymbol{R}_{p_2} \end{bmatrix}
$$

$$
\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{z}_m \\ \boldsymbol{z}_p \end{bmatrix} \; ; \; \boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{R}_m & 0 \\ 0 & \boldsymbol{R}_p \end{bmatrix}
$$

$$
\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{H}_m & 0 \\ 0 & \boldsymbol{H}_p \end{bmatrix} \; ; \; \boldsymbol{V}_k = \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}
$$



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

$[^N\hat{x}_0, {}^NP_0] = [^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[^N\hat{\bar{x}}_0, {}^NP_0] = AddNewPose(^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose(^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans(^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1$; $\mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: $^{B_j}S_j$ overlaps $^{B_k}S_k$

      $[^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      $^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      $^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2\oplus} {}^NP_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register(^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

      if $IndividuallyCompatible(^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i$; $c = c + 1$;

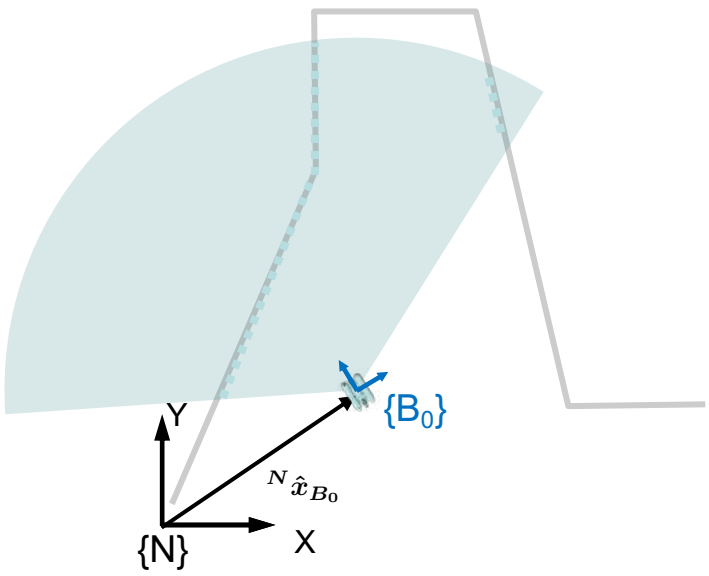
  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[^N\hat{x}_k, {}^NP_k] = Update(^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

$$^N\hat{x}_2 = \begin{bmatrix} ^N\hat{x}_{B_0} \\ ^N\hat{x}_{B_1} \\ ^N\hat{x}_{B_2} \\ \hline ^N\hat{x}_{B_2} \end{bmatrix} \quad ; \quad {}^NP_2 = \left[\begin{array}{ccc|c} {}^NP_{B_0} & {}^NP_{B_0B_1} & {}^NP_{B_0B_2} & {}^NP_{B_0B_2} \\ {}^NP_{B_1B_0} & {}^NP_{B_1} & {}^NP_{B_1,B_2} & {}^NP_{B_1B_2} \\ {}^NP_{B_2B_0} & {}^NP_{B_2B_1} & {}^NP_{B_2} & {}^NP_{B_2} \\ \hline {}^NP_{B_2B_0} & {}^NP_{B_2B_1} & {}^NP_{B_2} & {}^NP_{B_2} \end{array}\right]$$

$$\mathcal{M}_2 = \left[\left[^{B_0}S_0, {}^{B0}R_{S_0}\right] \left[^{B_1}S_1, {}^{B1}R_{S_1}\right] \left[^{B_2}S_2, {}^{B2}R_{S_2}\right]\right]$$

EKF Pose-Based SLAM



Add a New Viewpoint Pose

> Previous State Vector

$$ {}^{N}\hat{\hat{\boldsymbol{x}}}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{B_0} \\ \vdots \\ {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \end{bmatrix} \; ; \; {}^{N}\bar{\boldsymbol{P}}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\boldsymbol{P}_{B_0} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_{0,k}} \\ \vdots & \ddots & \vdots \\ {}^{N}\bar{\boldsymbol{P}}_{B_{k,0}} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_k} \end{bmatrix} $$

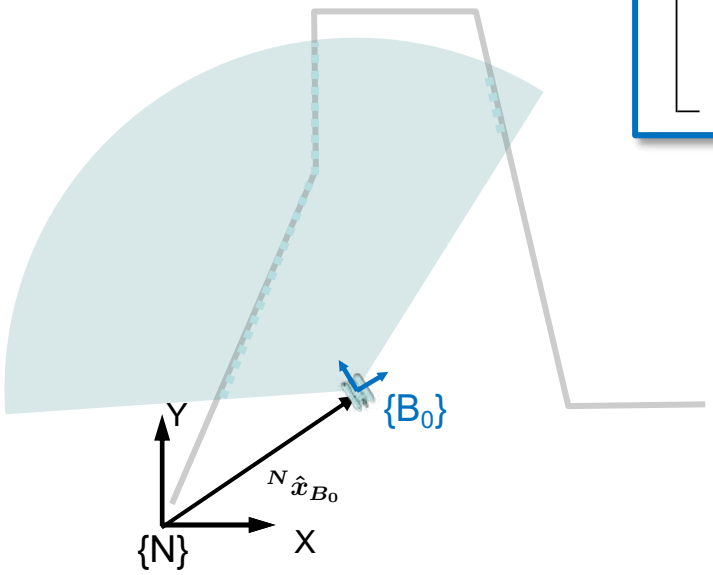
> Grow the state vector cloning the robot pose

$$ {}^{N}\hat{\hat{\boldsymbol{x}}}_{\boldsymbol{k}}{}^{+} = \left[\begin{array}{cccc} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \hline \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \end{array}\right] \cdot \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{B_0} \\ \vdots \\ {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \end{bmatrix} = \left[\begin{array}{c} {}^{N}\hat{\boldsymbol{x}}_{B_0} \\ \vdots \\ {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \\ \hline {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \end{array}\right] $$

$$ {}^{N}\boldsymbol{P}_{\boldsymbol{k}}{}^{+} = \left[\begin{array}{cccc} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ \hline \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \end{array}\right] \cdot \begin{bmatrix} {}^{N}\boldsymbol{P}_{B_0} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_{0,k}} \\ \vdots & \ddots & \vdots \\ {}^{N}\bar{\boldsymbol{P}}_{B_{k,0}} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_k} \end{bmatrix} \cdot \left[\begin{array}{cccc|c} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \boldsymbol{I} \end{array}\right] $$

$$ = \left[\begin{array}{ccc|c} {}^{N}\boldsymbol{P}_{B_0} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_{0,k}} & {}^{N}\bar{\boldsymbol{P}}_{B_{0,k}} \\ \vdots & \ddots & \vdots & \vdots \\ {}^{N}\bar{\boldsymbol{P}}_{B_{k,0}} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_k} & {}^{N}\bar{\boldsymbol{P}}_{B_k} \\ \hline {}^{N}\bar{\boldsymbol{P}}_{B_{k,0}} & \cdots & {}^{N}\bar{\boldsymbol{P}}_{B_k} & {}^{N}\bar{\boldsymbol{P}}_{B_k} \end{array}\right] $$

EKF Pose-Based SLAM



Add a New Viewpoint Pose

Method AddNewPose($^N\hat{x}_k$, ^NP_k)

$$^N\hat{x}_k^+ = \begin{bmatrix} ^N\hat{x}_k \\ \hline ^N\hat{x}_{B_k} \end{bmatrix}; \quad ^NP_k^+ = \left[\begin{array}{c|c} ^NP_k & ^NP_{0\ldots k,B_k} \\ \hline ^NP_{B_k,0\ldots k} & ^NP_{B_k} \end{array}\right];$$

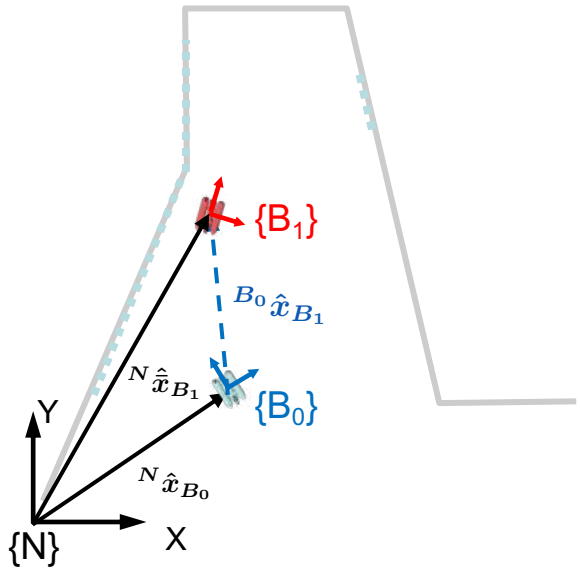
return $[^N\hat{\bar{x}}_k^+, {}^N\bar{P}_k^+]$

$$^N\hat{\bar{x}}_k^+ = \left[\begin{array}{ccc|c} I & 0 & \ldots & 0 \\ 0 & I & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & I \\ \hline 0 & 0 & \ldots & I \end{array}\right] \cdot \begin{bmatrix} ^N\hat{x}_{B_0} \\ \vdots \\ ^N\hat{\bar{x}}_{B_k} \end{bmatrix} = \begin{bmatrix} ^N\hat{x}_{B_0} \\ \vdots \\ ^N\hat{\bar{x}}_{B_k} \\ \hline ^N\hat{\bar{x}}_{B_k} \end{bmatrix}$$

$$^NP_k^+ = \left[\begin{array}{ccc|c} I & 0 & \ldots & 0 \\ 0 & I & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & I \\ \hline 0 & 0 & \ldots & I \end{array}\right] \cdot \begin{bmatrix} ^NP_{B_0} & \ldots & ^N\bar{P}_{B_{0,k}} \\ \vdots & \ddots & \vdots \\ ^N\bar{P}_{B_{k,0}} & \ldots & ^N\bar{P}_{B_k} \end{bmatrix} \cdot \left[\begin{array}{ccc|c} I & 0 & \ldots & 0 & 0 \\ 0 & I & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & I & I \end{array}\right]$$

$$= \left[\begin{array}{ccc|c} ^NP_{B_0} & \ldots & ^N\bar{P}_{B_{0,k}} & ^N\bar{P}_{B_{0,k}} \\ \vdots & \ddots & \vdots & \vdots \\ ^N\bar{P}_{B_{k,0}} & \ldots & ^N\bar{P}_{B_k} & ^N\bar{P}_{B_k} \\ \hline ^N\bar{P}_{B_{k,0}} & \ldots & ^N\bar{P}_{B_k} & ^N\bar{P}_{B_k} \end{array}\right]$$

EKF Pose-Based SLAM



Prediction

> Previous State Vector

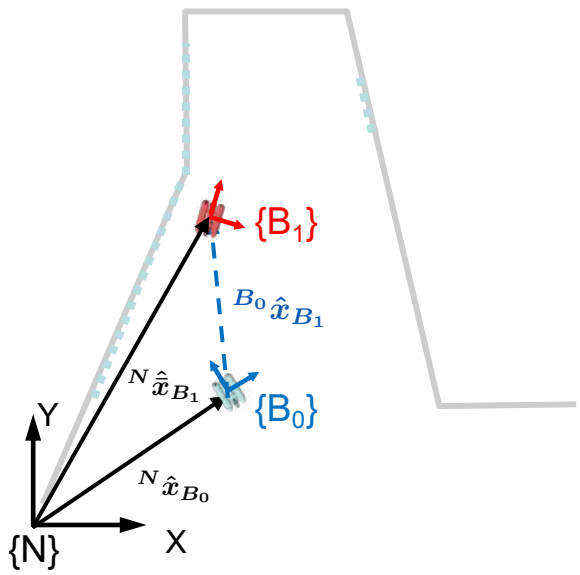
$$^{N}\hat{x}_{k-1} = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ \boxed{^{N}\hat{x}_{B_{k-1}}} \end{bmatrix} \; ; \; ^{N}P_{k-1} = \begin{bmatrix} ^{N}P_{B_0} & \cdots & ^{N}P_{B_{0,k-1}} \\ \vdots & \ddots & \vdots \\ ^{N}P_{B_{k-1,0}} & \cdots & ^{N}P_{B_{k-1}} \end{bmatrix}$$

> Predicted State Vector

$$^{N}\hat{\bar{x}}_{k} = \mathrm{f}(^{N}\hat{x}_{k-1}, u_k, \hat{w}_k) = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ \boxed{f(^{N}\hat{x}_{B_{k-1}}, u_k, \hat{w}_k)} \end{bmatrix}$$

$$^{N}\bar{P}_{k} = F_{1_k}\,^{N}P_{k-1}F_{1_k}^{T} + F_{2_k}Q_kF_{2_k}^{T}$$

EKF Pose-Based SLAM



Prediction

> Previous State Vector

$$^{N}\hat{x}_{k-1} = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ ^{N}\hat{x}_{B_{k-1}} \end{bmatrix} \; ; \; ^{N}P_{k-1} = \begin{bmatrix} ^{N}P_{B_0} & \cdots & ^{N}P_{B_{0,k-1}} \\ \vdots & \ddots & \vdots \\ ^{N}P_{B_{k-1,0}} & \cdots & ^{N}P_{B_{k-1}} \end{bmatrix}$$

> Predicted State Vector

$$^{N}\hat{\bar{x}}_{k} = \mathfrak{f}(^{N}\hat{x}_{k-1}, u_k, \hat{w}_k) = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ f(^{N}\hat{x}_{B_{k-1}}, u_k, \hat{w}_k) \end{bmatrix}$$

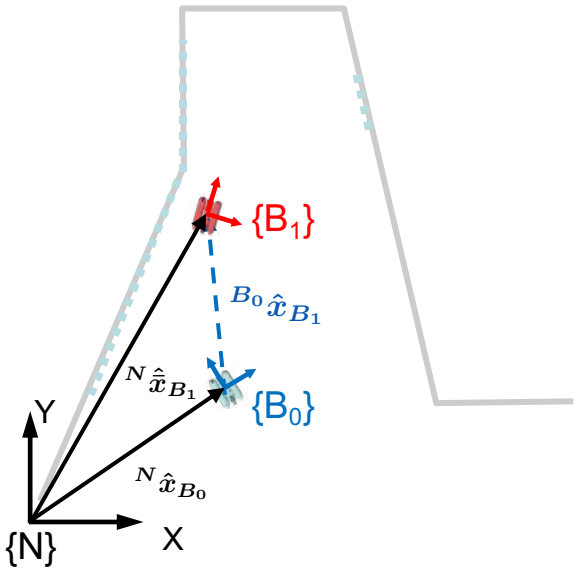
$$^{N}\bar{P}_{k} = F_{1_k} \, ^{N}P_{k-1} F_{1_k}^{T} + F_{2_k} Q_k F_{2_k}^{T}$$

$$F_{1_k} = \frac{\partial \mathfrak{f}(^{N}x_{k-1}, u_k, w_k)}{\partial ^{N}x_{k-1}}$$

$$= \begin{bmatrix} \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_0}} & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_{k-1}}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_0}} & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_{k-1}}} \\ \frac{\partial f(^{N}x_{B_{k-1}}, u_k, w_k)}{\partial ^{N}x_{B_0}} & \frac{\partial f(^{N}x_{B_{k-1}}, u_k, w_k)}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial f(^{N}x^{T}_{B_{k-1}}, u^{T}_k, w_k)}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial f(^{N}x^{T}_{B_{k-1}}, u^{T}_k, w_k)}{\partial ^{N}x_{B_{k-1}}} \end{bmatrix}$$

$$= \begin{bmatrix} I & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & I & 0 \\ 0 & \cdots & 0 & J_{f_x} \end{bmatrix}$$

EKF Pose-Based SLAM



Prediction

> Previous State Vector

$$^{N}\hat{x}_{k-1} = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ ^{N}\hat{x}_{B_{k-1}} \end{bmatrix} \; ; \; ^{N}P_{k-1} = \begin{bmatrix} ^{N}P_{B_0} & \cdots & ^{N}P_{B_{0,k-1}} \\ \vdots & \ddots & \vdots \\ ^{N}P_{B_{k-1,0}} & \cdots & ^{N}P_{B_{k-1}} \end{bmatrix}$$

> Predicted State Vector

$$^{N}\hat{\bar{x}}_{k} = \mathrm{f}(^{N}\hat{x}_{k-1}, u_k, \hat{w}_k) = \begin{bmatrix} ^{N}\hat{x}_{B_0} \\ \vdots \\ ^{N}\hat{x}_{B_{k-2}} \\ f(^{N}\hat{x}_{B_{k-1}}, u_k, \hat{w}_k) \end{bmatrix}$$

$$^{N}\bar{P}_{k} = F_{1_k}\,^{N}P_{k-1}F_{1_k}^{T} + F_{2_k}Q_kF_{2_k}^{T}$$

$$F_{2_k} = \frac{\partial \mathrm{f}(^{N}x_{k-1}, u_k, w_k)}{\partial w_k} = \begin{bmatrix} \frac{\partial ^{N}x_{B_0}}{\partial w_k} \\ \vdots \\ \frac{\partial ^{N}x_{B_{k-2}}}{\partial w_k} \\ \frac{\partial f(^{N}x_{B_{k-1}}, u_k, w_k)}{\partial w_k} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ J_{f_w} \end{bmatrix}$$

$$F_{1_k} = \frac{\partial \mathrm{f}(^{N}x_{k-1}, u_k, w_k)}{\partial ^{N}x_{k-1}}$$

$$= \begin{bmatrix} \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_0}} & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial ^{N}x_{B_0}}{\partial ^{N}x_{B_{k-1}}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_0}} & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial ^{N}x_{B_{k-2}}}{\partial ^{N}x_{B_{k-1}}} \\ \frac{\partial f(^{N}x_{B_{k-1}}, u_k, w_k)}{\partial ^{N}x_{B_0}} & \frac{\partial f(^{N}x_{B_{k-1}}, u_k, w_k)}{\partial ^{N}x_{B_1}} & \cdots & \frac{\partial f(^{N}x^{T}_{B_{k-1}}, u^{T}_k, w_k)}{\partial ^{N}x_{B_{k-2}}} & \frac{\partial f(^{N}x^{T}_{B_{k-1}}, u^{T}_k, w_k)}{\partial ^{N}x_{B_{k-1}}} \end{bmatrix}$$

$$= \begin{bmatrix} I & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & I & 0 \\ 0 & \cdots & 0 & J_{f_x} \end{bmatrix}$$

EKF Pose-Based SLAM

Prediction

$$
\begin{aligned}
{}^{N}\bar{P}_k &= F_{1_k}\,{}^{N}P_{k-1}F_{1_k}^T + F_{2_k}Q_kF_{2_k}^T \\
&= \begin{bmatrix} I & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & I & 0 \\ 0 & \dots & 0 & J_{f_x} \end{bmatrix}
\begin{bmatrix} {}^{N}P_{B_0} & \dots & {}^{N}P_{B_{0,k-1}} \\ \vdots & \ddots & \vdots \\ {}^{N}P_{B_{k-1,0}} & \dots & {}^{N}P_{B_{k-1}} \end{bmatrix}
\begin{bmatrix} I & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & I & 0 \\ 0 & \dots & 0 & J_{f_x}^T \end{bmatrix} \\
&+ \begin{bmatrix} 0 \\ \vdots \\ 0 \\ J_{f_w} \end{bmatrix} Q_k \begin{bmatrix} 0 & 0 & \dots & J_{f_w}^T \end{bmatrix} \\
&= \left[\begin{array}{ccc|c} {}^{N}P_{B_0} & \dots & {}^{N}P_{B_{0,k-2}} & {}^{N}P_{B_{0,k-1}}J_{f_x}^T \\ \vdots & \ddots & \vdots & \vdots \\ {}^{N}P_{B_{k-1,0}} & \dots & {}^{N}P_{B_{k-2}} & {}^{N}P_{B_{k-2,k-0}}J_{f_x}^T \\ \hline J_{f_x}\,{}^{N}P_{B_{k-1,0}} & \dots & J_{f_x}\,{}^{N}P_{B_{k-1,k-2}} & J_{f_x}\,{}^{N}P_{B_{k-1}}J_{f_x}^T \end{array}\right] + \left[\begin{array}{ccc|c} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \dots & J_{f_w}Q_kJ_{f_w}^T \end{array}\right] \\
&= \left[\begin{array}{c|c} {}^{N}P_{B_{0\dots k-2,0\dots k-2}} & {}^{N}P_{B_{0\dots k-2,k-1}}J_{f_x}^T \\ \hline J_{f_x}\,{}^{N}P_{B_{k-1,0\dots k-2}} & J_{f_x}\,{}^{N}P_{B_{k-1}}J_{f_x}^T + J_{f_w}Q_kJ_{f_w}^T \end{array}\right].
\end{aligned}
$$

EKF Pose-Based SLAM

Prediction

$$^{N}\bar{P}_k = F_{1_k}\,{}^{N}P_{k-1}F_{1_k}^T + F_{2_k}Q_kF_{2_k}^T$$

$$\begin{bmatrix} I & \dots & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^{N}P_{B_0} & \dots & {}^{N}P_{B_{0\,k-1}} \end{bmatrix} \begin{bmatrix} I & \dots & 0 & 0 \end{bmatrix}$$

Method Prediction$(^{N}\hat{x}_{k-1}, {}^{N}P_{k-1}, {}^{N}\hat{\bar{x}}_k, u_k, Q_k)$

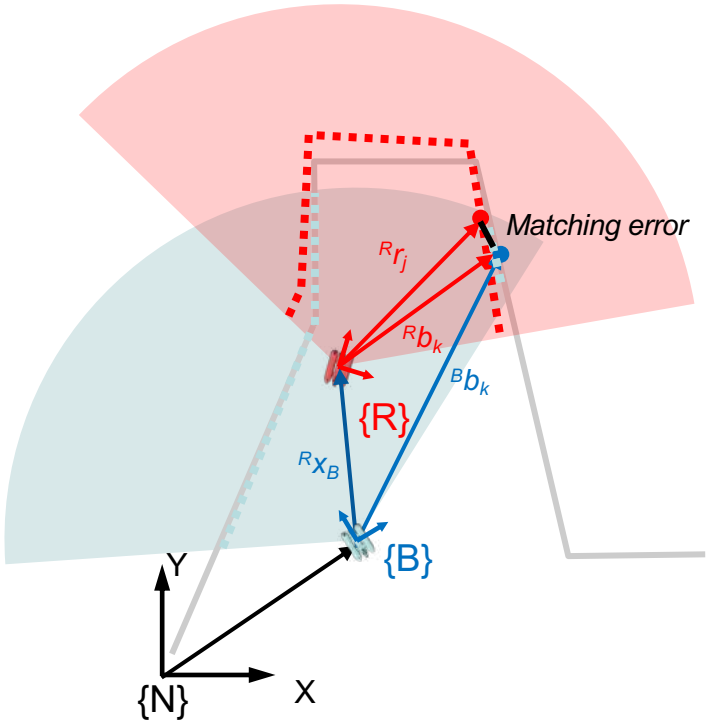
$$^{N}\hat{\bar{x}}_k = \begin{bmatrix} {}^{N}\hat{x}_{B_0} \\ \vdots \\ {}^{N}\hat{x}_{B_{k-2}} \\ f({}^{N}\hat{x}_{B_{k-1}}, u_k, \hat{w}_k) \end{bmatrix};$$

$$^{N}\bar{x}_k = \left[\begin{array}{c|c} {}^{N}P_{B_{1\dots k-2,1\dots k-2}} & {}^{N}P_{B_{1\dots k-2,k-1}}J_{f_x}^T \\ \hline J_{f_x}\,{}^{N}P_{B_{k-1,1\dots k-2}} & J_{f_x}\,{}^{N}P_{B_{k-1}}J_{f_x}^T + J_{f_w}Q_kJ_{f_w}^T \end{array}\right];$$

return $[^{N}\hat{\bar{x}}_k, {}^{N}\bar{P}_k]$

$$= \left[\begin{array}{c|c} {}^{N}P_{B_{0\dots k-2,0\dots k-2}} & {}^{N}P_{B_{0\dots k-2,k-1}}J_{f_x}^T \\ \hline J_{f_x}\,{}^{N}P_{B_{k-1,0\dots k-2}} & J_{f_x}\,{}^{N}P_{B_{k-1}}J_{f_x}^T + J_{f_w}Q_kJ_{f_w}^T \end{array}\right].$$

EKF Pose-Based SLAM



Registration

Function ICP($^{\mathcal{R}}S_r, ^{\mathcal{B}}S_b, ^{\mathcal{R}}x_B$)

// New Scan: $^{\mathcal{R}}S_r = [^{\mathcal{R}}r_1\ ^{\mathcal{R}}r_2\ \dots\ ^{\mathcal{R}}r_N\ ]$

// Reference Scan: $^{\mathcal{B}}S_b = [^{\mathcal{B}}b_1\ ^{\mathcal{B}}b_2\ \dots\ ^{\mathcal{B}}b_S\ ]$

// Initial Transformation guess: $^{\mathcal{R}}x_B$

$i = 1$;

while ($i < maxIterations$) **and**(**not** *converged*) **do**

  for $j = 1$ **to** N **do** // Every point in the R Scan

    $^{\mathcal{B}}c_j = \underset{^{\mathcal{B}}b_k \in {}^{\mathcal{B}}S_b}{\operatorname{argmin}} \{||^{\mathcal{R}}x_B \oplus {}^{\mathcal{B}}b_k - {}^{\mathcal{R}}r_j||\}$; // Compute the Nearest Neighbor

  end

  // LS minimization of the Matching error

  $^{\mathcal{R}}x_B = \underset{^{\mathcal{R}}x_B}{\operatorname{argmin}} \{\sum_j^N e_j^T \cdot e_j\}$ where $e_j = {}^{\mathcal{R}}x_B \oplus {}^{\mathcal{B}}c_j - {}^{\mathcal{R}}r_j$;

  $i = i + 1$;

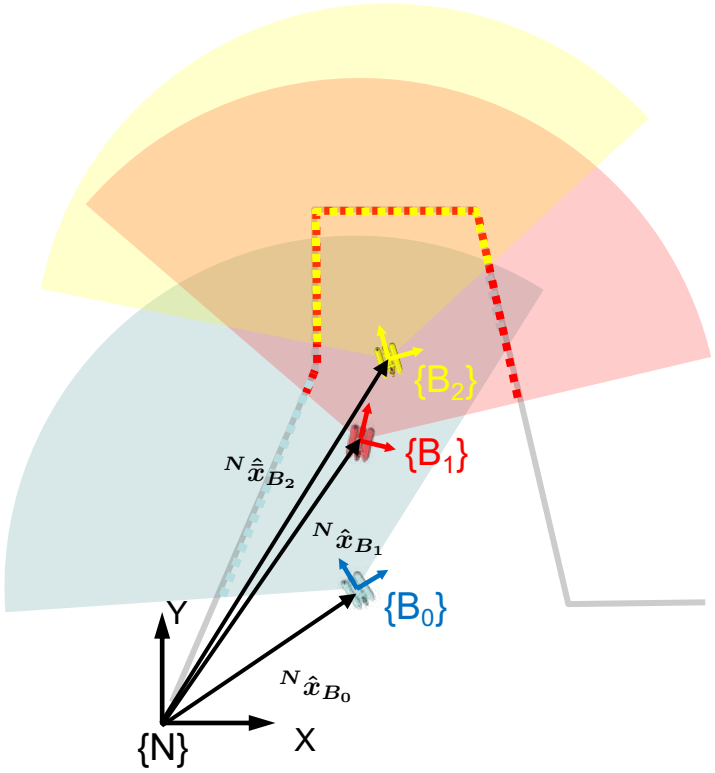
end

return $^{\mathcal{R}}x_B$;

end

Algorithm 1: Iterative Closest Point

EKF Pose-Based SLAM



$$B_0\text{–}frame \equiv \mathcal{B}\text{–}frame$$
$$B_1\text{–}frame \equiv \mathcal{R}\text{–}frame$$
$$B_2\text{–}frame \equiv \mathcal{Y}\text{–}frame$$

Pose Constraints

1. Compute overlapping scans

$$\mathcal{H}_p = OverlappingScans({}^{N}\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}}, \mathcal{M})$$
$$\mathcal{H}_p = [0\ 1\ 2]$$

2. Register overlapping scans

$$\begin{aligned}
[{}^{B_k}\boldsymbol{x}_{\mathcal{Y}}, {}^{B_k}\boldsymbol{R}_{\mathcal{Y}}] &= Registration({}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}, {}^{\mathcal{Y}}\boldsymbol{S}_{\boldsymbol{y}}) \\
[{}^{B_k}\boldsymbol{x}_{\mathcal{B}}, {}^{B_k}\boldsymbol{R}_{\mathcal{B}}] &= Registration({}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}, {}^{\mathcal{B}}\boldsymbol{S}_{\boldsymbol{b}}) \\
[{}^{B_k}\boldsymbol{x}_{\mathcal{R}}, {}^{B_k}\boldsymbol{R}_{\mathcal{R}}] &= Registration({}^{B_k}\boldsymbol{S}_{\boldsymbol{k}}, {}^{\mathcal{R}}\boldsymbol{S}_{\boldsymbol{r}}),
\end{aligned}$$

3. Observation equation

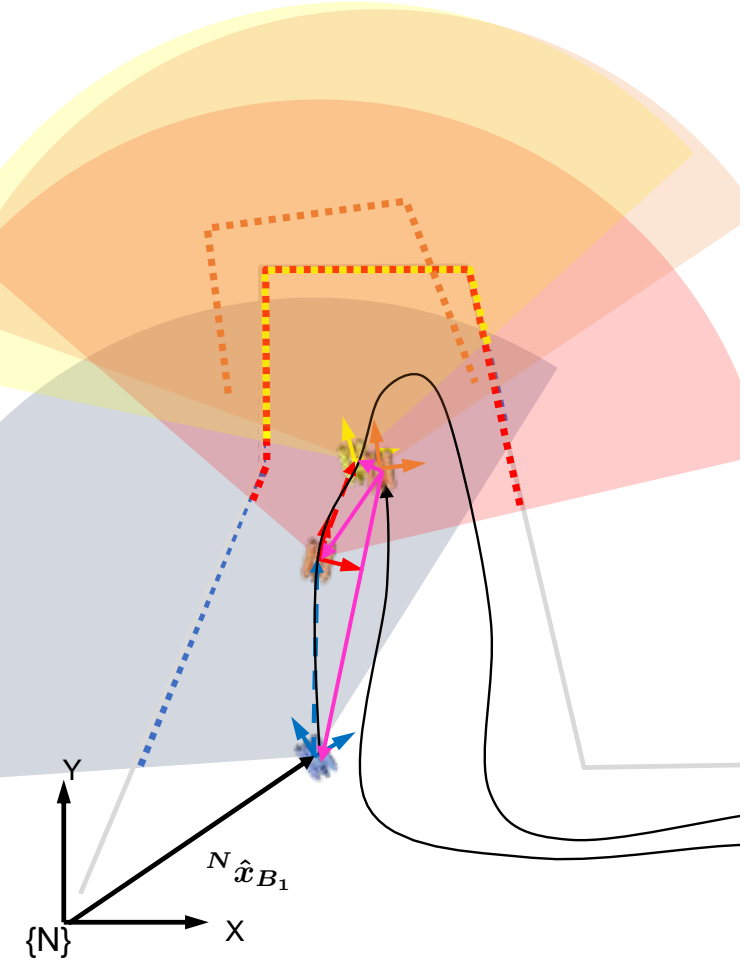
$$\boldsymbol{z}_{\boldsymbol{r}} = \boldsymbol{h}({}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{v}_{\boldsymbol{r}_k})$$

$$\begin{bmatrix} \boldsymbol{z}_{\boldsymbol{y}} \\ \boldsymbol{z}_{\boldsymbol{b}} \\ \boldsymbol{z}_{\boldsymbol{r}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_{\boldsymbol{r}}({}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{N}\boldsymbol{x}_{\mathcal{Y}}, \boldsymbol{v}_{\boldsymbol{y}_k}) \\ \boldsymbol{h}_{\boldsymbol{r}}({}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{N}\boldsymbol{x}_{\mathcal{B}}, \boldsymbol{v}_{\boldsymbol{b}_k}) \\ \boldsymbol{h}_{\boldsymbol{r}}({}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}, {}^{N}\boldsymbol{x}_{\mathcal{R}}, \boldsymbol{v}_{\boldsymbol{r}_k}) \end{bmatrix}$$

$$\begin{bmatrix} {}^{B_k}\boldsymbol{x}_{\mathcal{Y}} \\ {}^{B_k}\boldsymbol{x}_{\mathcal{B}} \\ {}^{B_k}\boldsymbol{x}_{\mathcal{R}} \end{bmatrix} = \begin{bmatrix} \left((\ominus {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}) \oplus {}^{N}\boldsymbol{x}_{\mathcal{Y}}\right) + \boldsymbol{v}_{\boldsymbol{y}_k} \\ \left((\ominus {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}) \oplus {}^{N}\boldsymbol{x}_{\mathcal{B}}\right) + \boldsymbol{v}_{\boldsymbol{b}_k} \\ \left((\ominus {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}}) \oplus {}^{N}\boldsymbol{x}_{\mathcal{R}}\right) + \boldsymbol{v}_{\boldsymbol{r}_k} \end{bmatrix}$$

EKF Pose-Based SLAM

Loop Closing



Pose Constraints

1. Compute overlapping scans

$$\mathcal{H}_p = OverlappingScans(^{N}\hat{\bar{x}}_{k}, \mathcal{M})$$
$$\mathcal{H}_p = [0\ 1\ 2]$$

2. Register overlapping scans

$$[^{B_k}x_{\mathcal{Y}}, {}^{B_k}R_{\mathcal{Y}}] = Registration(^{B_k}S_k, {}^{\mathcal{Y}}S_y)$$
$$[^{B_k}x_{\mathcal{B}}, {}^{B_k}R_{\mathcal{B}}] = Registration(^{B_k}S_k, {}^{\mathcal{B}}S_b)$$
$$[^{B_k}x_{\mathcal{R}}, {}^{B_k}R_{\mathcal{R}}] = Registration(^{B_k}S_k, {}^{\mathcal{R}}S_r),$$

3. Observation equation

$$z_r = h(^{N}\bar{x}_k, v_{r_k})$$

$$\begin{bmatrix} z_y \\ z_b \\ z_r \end{bmatrix} = \begin{bmatrix} h_r(^{N}\bar{x}_{B_k}, {}^{N}x_{\mathcal{Y}}, v_{y_k}) \\ h_r(^{N}\bar{x}_{B_k}, {}^{N}x_{\mathcal{B}}, v_{b_k}) \\ h_r(^{N}\bar{x}_{B_k}, {}^{N}x_{\mathcal{R}}, v_{r_k}) \end{bmatrix}$$

$$\begin{bmatrix} {}^{B_k}x_{\mathcal{Y}} \\ {}^{B_k}x_{\mathcal{B}} \\ {}^{B_k}x_{\mathcal{R}} \end{bmatrix} = \begin{bmatrix} ((\ominus {}^{N}\bar{x}_{B_k}) \oplus {}^{N}x_{\mathcal{Y}}) + v_{y_k} \\ ((\ominus {}^{N}\bar{x}_{B_k}) \oplus {}^{N}x_{\mathcal{B}}) + v_{b_k} \\ ((\ominus {}^{N}\bar{x}_{B_k}) \oplus {}^{N}x_{\mathcal{R}}) + v_{r_k} \end{bmatrix}$$

Observation Jacobians

$$\boldsymbol{H_k} = \left.\frac{\partial \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_k)}{\partial {}^N\bar{\boldsymbol{x}}_k}\right|_{{}^N\bar{\boldsymbol{x}}_k = {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{w}_k = \boldsymbol{0}}$$

$$= \begin{array}{c|cccccccc} & 1 & 2 & \dots & y & \dots & b & \dots & r & \dots & n_p \\ \hline 1 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{B}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{R}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \\ 2 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{B}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{R}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \\ 3 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial \boldsymbol{h}_r({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{B}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{R}}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \end{array}$$

$$= \begin{array}{c|cccccccccc} & 1 & 2 & \dots & y & \dots & b & \dots & r & \dots & n_p \\ \hline 1 & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{J_{2_\oplus}} & \dots & \dots & \dots & \dots & \dots & \boldsymbol{J_{1_\oplus}} \boldsymbol{J_\ominus} \\ 2 & \boldsymbol{0} & \boldsymbol{0} & \dots & \dots & \dots & \boldsymbol{J_{2_\oplus}} & \dots & \dots & \dots & \boldsymbol{J_{1_\oplus}} \boldsymbol{J_\ominus} \\ 3 & \boldsymbol{0} & \boldsymbol{0} & \dots & \dots & \dots & \dots & \dots & \boldsymbol{J_{2_\oplus}} & \dots & \boldsymbol{J_{1_\oplus}} \boldsymbol{J_\ominus} \end{array}$$

$$\boldsymbol{V_k} = \left.\frac{\partial \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{v}_k}\right|_{\bar{\boldsymbol{x}}_k = {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{w}_k = \boldsymbol{0}}$$

$$= \begin{bmatrix} \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{r_k}} \\ \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{r_k}} \\ \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{r_k}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

EKF Pose-Based SLAM

Observation Jacobians

$$\boldsymbol{H_k} = \left.\frac{\partial \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_k)}{\partial {}^N\bar{\boldsymbol{x}}_k}\right|_{{}^N\bar{\boldsymbol{x}}_k = {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{w}_k = \boldsymbol{0}}$$

$$= \begin{array}{c|ccc} & 1 & 2 & \dots & y \\ \hline 1 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} \\ 2 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} \\ 3 & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{B_0}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial {}^N\boldsymbol{x}_{B_1}} & \dots & \frac{\partial \boldsymbol{h}_r({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{y_k})}{\partial {}^N\boldsymbol{x}_{\mathcal{Y}}} \end{array}$$

$$= \begin{array}{c|cccccccccc} & 1 & 2 & \dots & y & \dots & b & \dots & r & \dots & n_p \\ \hline 1 & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{J_{2\oplus}} & \dots & \dots & \dots & \dots & \dots & \boldsymbol{J_{1\oplus}} \boldsymbol{J_\ominus} \\ 2 & \boldsymbol{0} & \boldsymbol{0} & \dots & \dots & \dots & \boldsymbol{J_{2\oplus}} & \dots & \dots & \dots & \boldsymbol{J_{1\oplus}} \boldsymbol{J_\ominus} \\ 3 & \boldsymbol{0} & \boldsymbol{0} & \dots & \dots & \dots & \dots & \dots & \boldsymbol{J_{2\oplus}} & \dots & \boldsymbol{J_{1\oplus}} \boldsymbol{J_\ominus} \end{array}$$

$$\boldsymbol{V_k} = \left.\frac{\partial \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_k)}{\partial \boldsymbol{v}_k}\right|_{\bar{\boldsymbol{x}}_k = {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{w}_k = \boldsymbol{0}}$$

$$= \begin{bmatrix} \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{Y}} + \boldsymbol{v}_{y_k})}{\partial \boldsymbol{v}_{r_k}} \\ \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{B}} + \boldsymbol{v}_{b_k})}{\partial \boldsymbol{v}_{r_k}} \\ \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{y_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{b_k}} & \frac{\partial((\ominus {}^N\bar{\boldsymbol{x}}_{B_k}) \oplus {}^N\boldsymbol{x}_{\mathcal{R}} + \boldsymbol{v}_{r_k})}{\partial \boldsymbol{v}_{r_k}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

Method `ObservationMatrix`$(\mathcal{H}_p, {}^N\hat{\bar{\boldsymbol{x}}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$

- $\boldsymbol{H_p} = \boldsymbol{0}$;
- **for** $i = 1$ *to* $length(\boldsymbol{z_p})$ **do**
  - $\boldsymbol{H_p}[i, 1] = \boldsymbol{J_{1\oplus}}(\ominus {}^N\hat{\bar{\boldsymbol{x}}}_{B_k}, {}^N\hat{\boldsymbol{x}}_k[\mathcal{H}_p[i]]) \boldsymbol{J_\ominus}({}^N\hat{\bar{\boldsymbol{x}}}_{B_k})$;
  - $\boldsymbol{H_p}[i, \mathcal{H}_p[i]] = \boldsymbol{J_{2\oplus}}({}^N\hat{\bar{\boldsymbol{x}}}_{B_k})$;
- $\boldsymbol{z_k} = \begin{bmatrix} \boldsymbol{z_m} \\ \boldsymbol{z_p} \end{bmatrix}$; $\boldsymbol{R_k} = \begin{bmatrix} \boldsymbol{R_m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_p} \end{bmatrix}$;
- $\boldsymbol{H_k} = \begin{bmatrix} \boldsymbol{J_{h_x}}(\hat{\bar{\boldsymbol{x}}}_k) \\ \boldsymbol{H_p} \end{bmatrix}$;
- $\boldsymbol{V_k} = \begin{bmatrix} \boldsymbol{J_{h_v}}(\hat{\bar{\boldsymbol{x}}}_k) \\ \boldsymbol{V_p} \end{bmatrix}$;
- **return** $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}]$;

Method `h`$({}^N\hat{\boldsymbol{x}}_k, \mathcal{H}_p)$

- $h_{mf} = parent.h(\hat{\bar{\boldsymbol{x}}}_k)$;
- **for** $i = 1$ *to* $length(\mathcal{H}_p)$ **do**
  - $j = \mathcal{H}_p[i]$;
  - **if** $j \neq 0$ **then**
    - $h_{mf} = \begin{bmatrix} h_{mf} \\ \ominus {}^N\hat{\bar{\boldsymbol{x}}}_{B_k} \oplus {}^N\boldsymbol{x}_k[j] \end{bmatrix}$;
- **return** $\boldsymbol{h_{mf}}$;

Update

1. The Standard EKF Update equations are used:

$$\boldsymbol{K}_k = {}^{\boldsymbol{N}}\bar{\boldsymbol{P}}_k \boldsymbol{H}_k^T (\boldsymbol{H}_k {}^{\boldsymbol{N}}\bar{\boldsymbol{P}}_k \boldsymbol{H}_k^T + \boldsymbol{V}_k \boldsymbol{R}_k \boldsymbol{V}_k^T)^{-1}$$

$${}^{\boldsymbol{N}}\hat{\boldsymbol{x}}_{\boldsymbol{k}} = {}^{\boldsymbol{N}}\bar{\hat{\boldsymbol{x}}}_{\boldsymbol{k}} + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{h}({}^{\boldsymbol{N}}\bar{\hat{\boldsymbol{x}}}_{\boldsymbol{k}}, \boldsymbol{z}_{\boldsymbol{p}}, \mathcal{H}_p) \right)$$

$${}^{\boldsymbol{N}}\boldsymbol{P}_{\boldsymbol{k}} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) {}^{\boldsymbol{N}}\bar{\boldsymbol{P}}_k (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^T$$

Pose based EKF SLAM using Point Clouds

Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner

Method Localization($\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0$)

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$;; // Read the Scan from the sensor
- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [{}^{B_0}\boldsymbol{S}_0]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u_k}, \boldsymbol{Q_k}] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u_k}, \boldsymbol{Q_k})$;
  - $[\boldsymbol{z_m}, \boldsymbol{R_m}] = GetMeasurement()$; // Read navigation sensors
  - **if** *ScanAvailable* **then**
    - $[{}^{B_k}\boldsymbol{S_k}, {}^{B_k}\boldsymbol{R}_{\boldsymbol{S_k}}] = GetScan()$;
    - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k] = AddNewPose({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = {}^{B_k}\boldsymbol{S_k}$; // Store the scan in the map
    - $\mathcal{H}_p = OverlappingScans({}^N\hat{\boldsymbol{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - **for** $i = 1$ *to* $length(\mathcal{H}_p)$ **do** // for all overlaps
      - $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map
      - ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_k}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess
      - $[\boldsymbol{z_p}, \boldsymbol{R_p}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
      - $i = i + 1$; // Go to next pair
  - $\mathcal{H}_p = DataAssociation({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_p}, \boldsymbol{R_p})$; // select compatible registrations
  - $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\boldsymbol{x}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \mathcal{H}_p)$;

> To implement a PEKFSLAM we need to:

1. Program the ***GetInput(·)*** function
2. Define the **motion model:** ${}^N\bar{\boldsymbol{x}}_{B_k} = \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)$
   - Compute the motion model Jacobians:

$$\boldsymbol{J}_{\boldsymbol{f_x}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial {}^N\boldsymbol{x}_{B_{k-1}}} \;,\; \boldsymbol{J}_{\boldsymbol{f_w}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial \boldsymbol{w_k}}$$

3. Program the ***GetScan()*** function
4. Define the **observation model:**

$$\boldsymbol{z_k} = \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{p_k}})$$

$$\begin{bmatrix} \boldsymbol{z_{ij}} \\ \vdots \\ \boldsymbol{z_{pq}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{ij_k}}) \\ \vdots \\ \boldsymbol{h_{pq}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{pq_k}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta x_{ij}} \\ \vdots \\ \boldsymbol{\Delta x_{pq}} \end{bmatrix}$$

   - Compute the observation Jacobians:

$\forall i, j$ compute:

$$\boldsymbol{J}_{\boldsymbol{h_x}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \;,\; \boldsymbol{J}_{\boldsymbol{h_v}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial \boldsymbol{v_k}}$$

5. Implement the ***Register(·)*** *function*

GetInput()

$$\boldsymbol{u_k} = \begin{bmatrix} {}^{N}\phi_B \\ {}^{N}\theta_B \\ {}^{N}\psi_B \end{bmatrix}$$

Motion Model

State Vector:

$$ {}^{N}\boldsymbol{x}_{B_k} = [\underbrace{{}^{N}x_B \ {}^{N}y_B \ {}^{N}z_B}_{{}^{N}\boldsymbol{\eta_1}_{B_k}} \ \underbrace{{}^{B}u_B \ {}^{B}v_B \ {}^{B}w_B}_{{}^{B}\boldsymbol{\nu}_k} \ {}^{N}\boldsymbol{x}_{B_n} \ \dots \ {}^{N}\boldsymbol{x}_{B_1}]^T $$

Motion Model:

$$ {}^{N}\bar{\boldsymbol{x}}_{B_k} = \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right) = \begin{bmatrix} {}^{N}\boldsymbol{\eta_1}_{B_{k-1}} + {}^{N}\boldsymbol{R_B}(\boldsymbol{u_k} + \boldsymbol{w}_{\eta_{2_k}})\left({}^{B}\boldsymbol{\nu_{k-1}}\Delta t + \boldsymbol{w}_{\dot{\nu}_k}\frac{\boldsymbol{\Delta t^2}}{\boldsymbol{2}}\right) \\ {}^{B}\boldsymbol{\nu_{k-1}} + \boldsymbol{w}_{\dot{\nu}_k}\Delta t \end{bmatrix} $$

where $\boldsymbol{u_k} = \begin{bmatrix} {}^{N}\phi_B & {}^{N}\theta_B & {}^{N}\psi_B \end{bmatrix}^T$

$$ \boldsymbol{w_k} = \begin{bmatrix} \boldsymbol{w}^{\boldsymbol{T}}_{\boldsymbol{\eta_{2_k}}} & \boldsymbol{w}^{\boldsymbol{T}}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}} \end{bmatrix}^T = \begin{bmatrix} w_\phi & w_\theta & w_\psi & w_{\dot{u}} & w_{\dot{v}} & w_{\dot{w}} \end{bmatrix}^T $$

$$ \boldsymbol{w_k} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q_k}); \boldsymbol{Q_k} = diag(\sigma^2_{w_\phi}, \sigma^2_{w_\theta}, \sigma^2_{w_\psi}, \sigma^2_{w_{\dot{u}}}, \sigma^2_{w_{\dot{v}}}, \sigma^2_{w_{\dot{w}}}) $$

Jacobians:

$$ \boldsymbol{J}_{\boldsymbol{f_x}} = \frac{\partial \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial {}^{N}\boldsymbol{x}_{B_{k-1}}} = \begin{bmatrix} \boldsymbol{I_{3\times 3}} & {}^{N}\boldsymbol{R_B}(\boldsymbol{u_k})\Delta t \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{I_{3\times 3}} \end{bmatrix} $$

$$ \boldsymbol{J}_{\boldsymbol{f_w}} = \frac{\partial \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial \boldsymbol{w_k}} = \begin{bmatrix} \boldsymbol{J_R} \cdot {}^{B}\hat{\boldsymbol{\nu}}_{\boldsymbol{k-1}}\Delta t & {}^{N}\boldsymbol{R_B}(\boldsymbol{u_k})\frac{\Delta t^2}{2} \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{I_{3\times 3}}\Delta t \end{bmatrix} $$

where $\boldsymbol{J_R} = \left.\dfrac{\partial {}^{N}\boldsymbol{R_B}(\boldsymbol{u_k} + \boldsymbol{w}_{\eta_{2_k}})}{\partial \boldsymbol{w}_{\eta_{2_k}}}\right|_{{}^{N}\boldsymbol{x}_{B_{k-1}} = {}^{N}\hat{\boldsymbol{x}}_{B_{k-1}}, \boldsymbol{w_k}=\boldsymbol{0}}$

View poses

> The State Vector contains only the position: $^{N}x_k = [\eta_{1_{B_k}}^T \ \nu_1^T]^T$

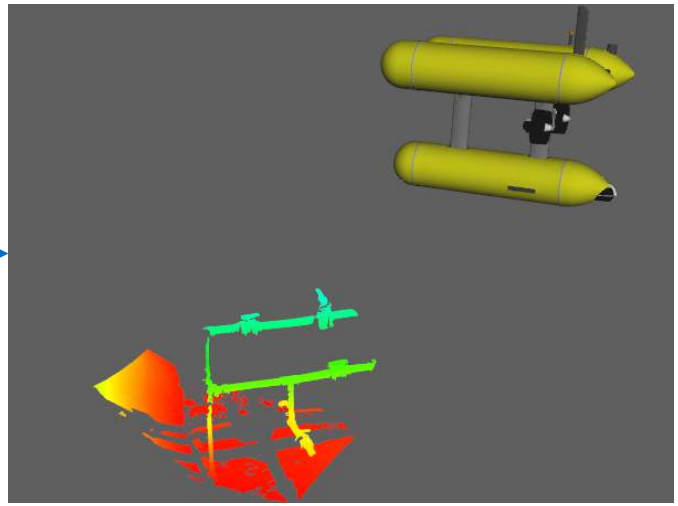
> A View Pose can be computed combining the position and the input attitude:

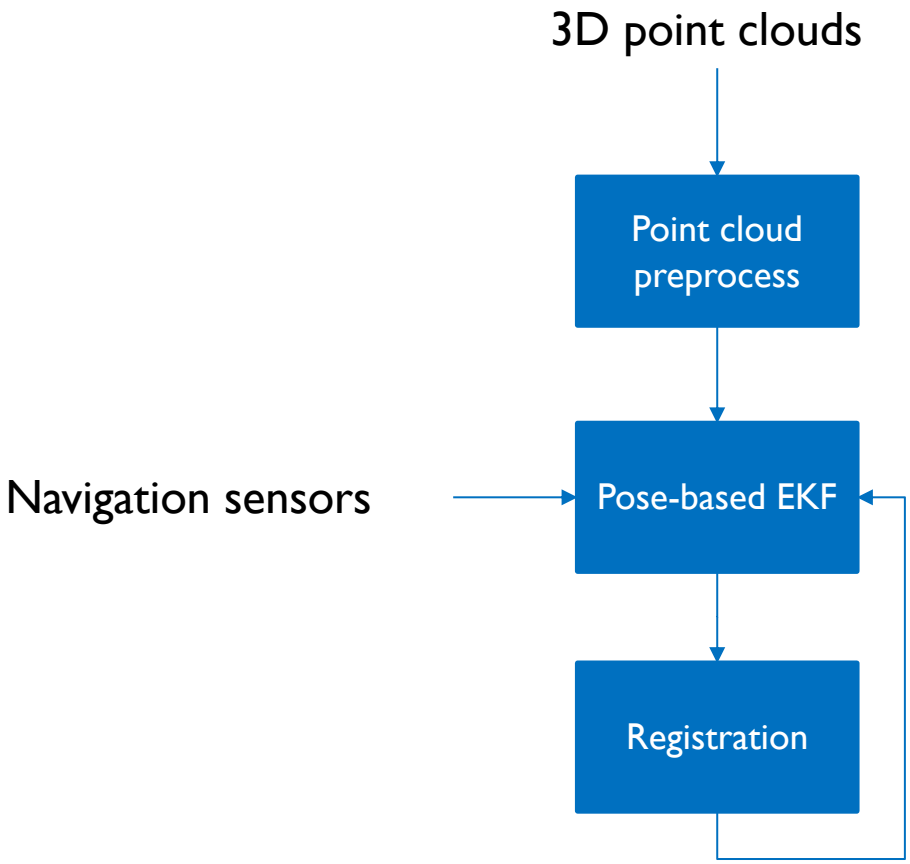
$$\left.\begin{array}{rl} ^{N}x_k & = [\eta_{1_{B_k}}^T \ \nu_1^T]^T \\ u_k & = \eta_{2_{B_k}} = [\phi \quad \theta \quad \psi]^T \\ w_{\eta_2} & = N(0, R_{AHRS}) \end{array}\right\} \Rightarrow \eta_i = \begin{bmatrix} \eta_{1_i} \\ \eta_{2_i} \end{bmatrix} = \underbrace{\begin{bmatrix} I \\ 0 \end{bmatrix}}_{F} f(x_{k-1}, u_k, w_k) + \underbrace{\begin{bmatrix} 0 \\ I \end{bmatrix}}_{G} (u_k + w_{\eta_2})$$

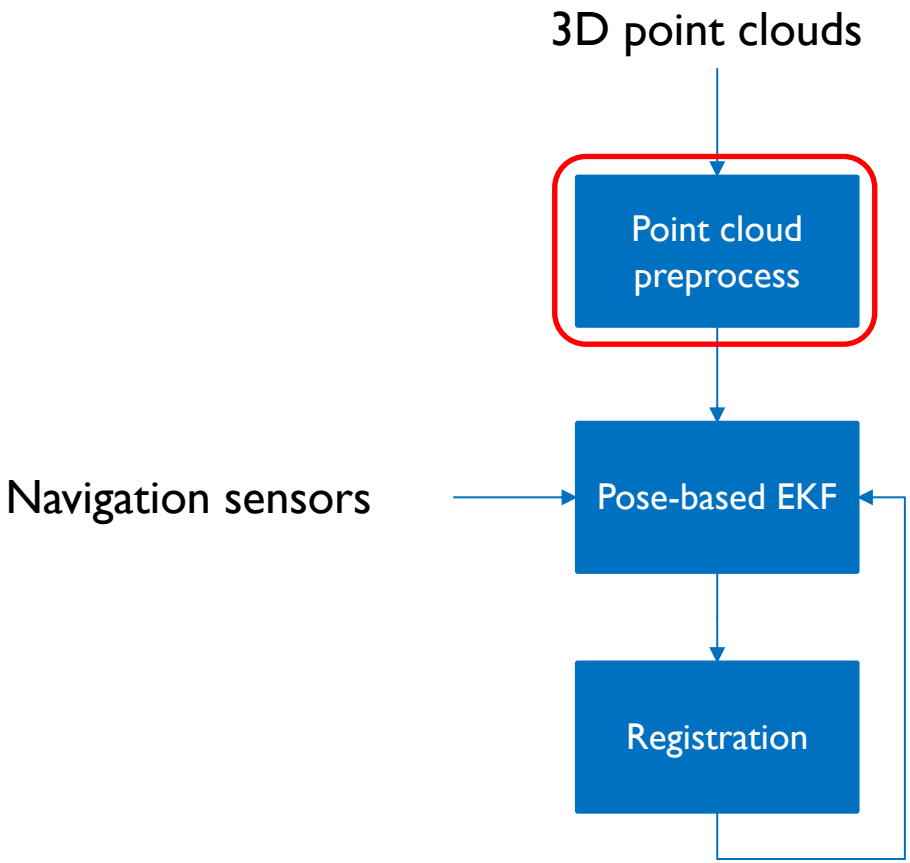
> Adding a new View Pose can be achieved by:

$$\bar{x}_k^+ = \begin{bmatrix} F f(x_{k-1}, u_k, w_k) + G(u_k + w_{\eta_2}) \\ x_k \end{bmatrix}; F = \begin{bmatrix} I \\ 0 \end{bmatrix}; \ G = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

GetScan()

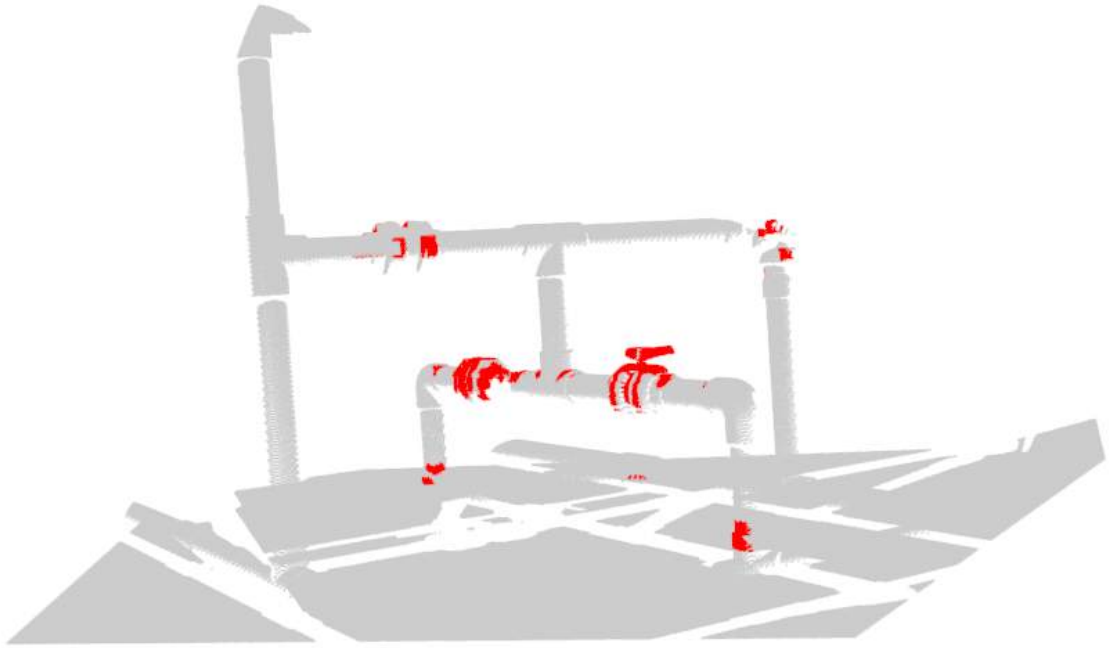



3D point clouds
Point cloud preprocess
Navigation sensors
Pose-based EKF
Registration


3D point clouds
Point cloud preprocess
Navigation sensors
Pose-based EKF
Registration

Point cloud preprocess

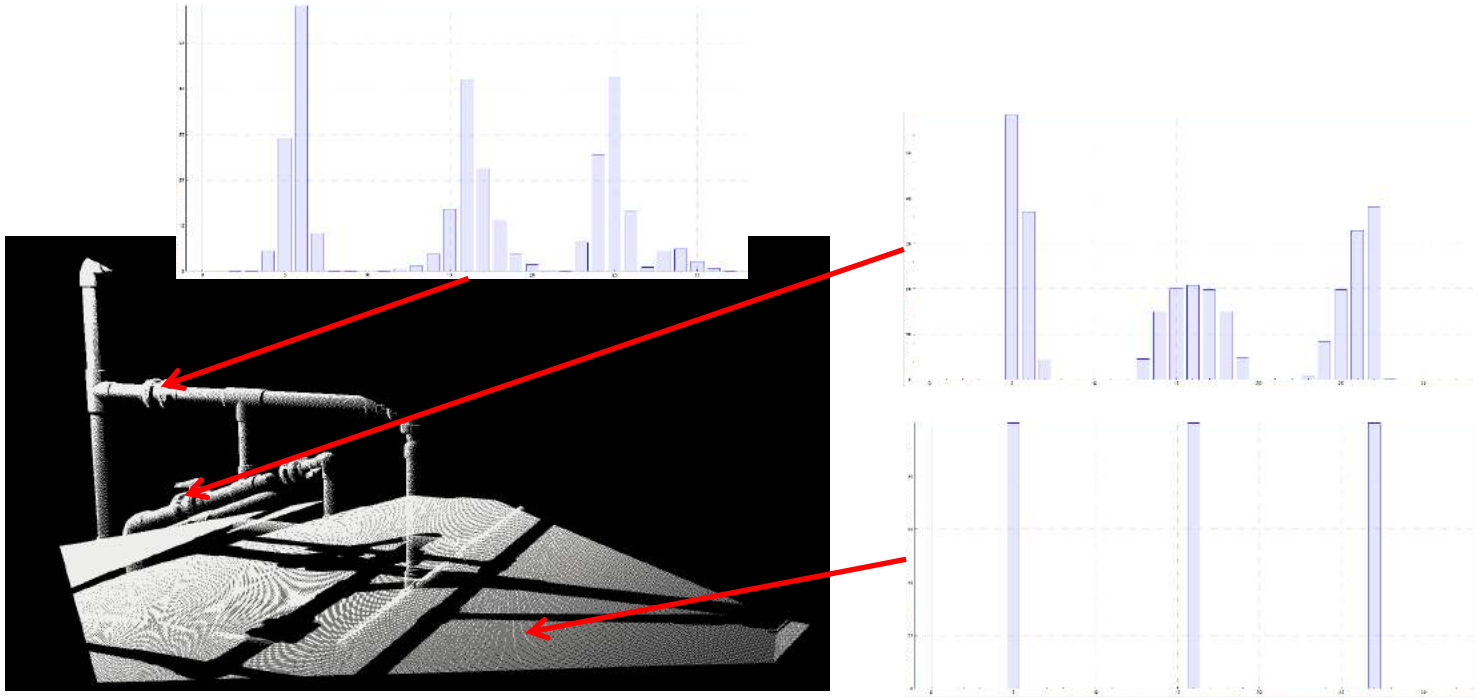
- Key points extraction
  - Remove planar surfaces (RANSAC, Fischler et al. 1981)
  - Remove points with curvature (Pauly et al., 2002) below threshold

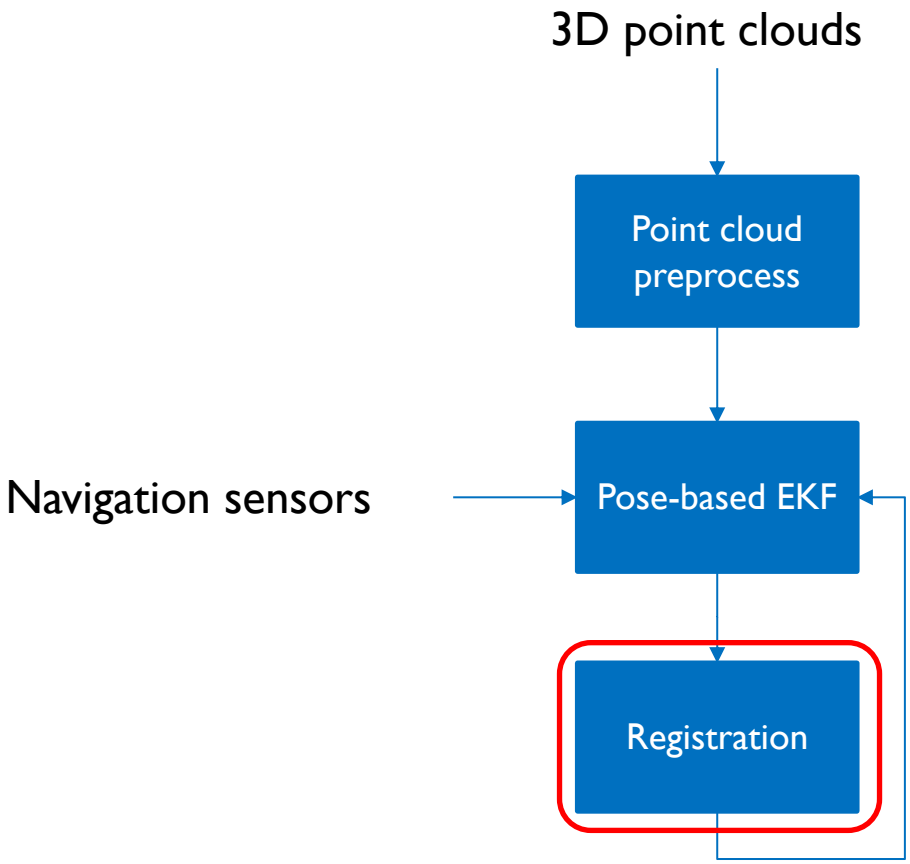


Point cloud preprocess

- Key points extraction
  - Remove planar surfaces (RANSAC, Fischler et al. 1981)
  - Remove points with curvature (Pauly et al., 2002) below threshold

- Feature extraction: Fast point feature histogram (Rusu et al., 2009)



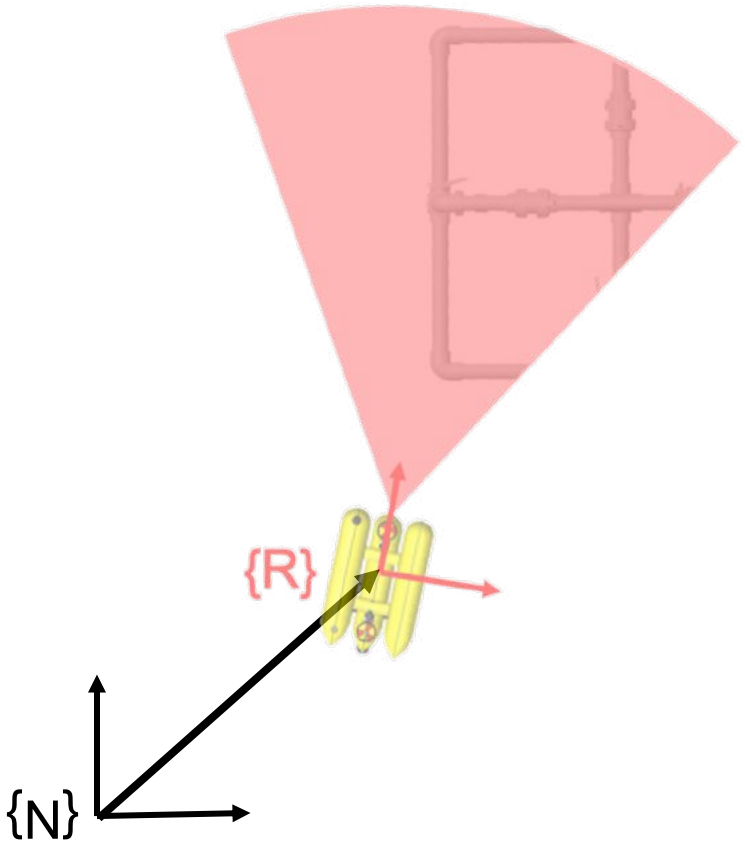

3D point clouds
Point cloud preprocess
Navigation sensors
Pose-based EKF
Registration

EKF Pose-Based SLAM

> The AUV takes 1 scans and adds the View-Pose to the map

$$\mathcal{M} = [{}^{R}\boldsymbol{S}_{R}]$$

$${}^{N}\hat{\boldsymbol{x}}_{k} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{R} \\ {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \end{bmatrix}$$

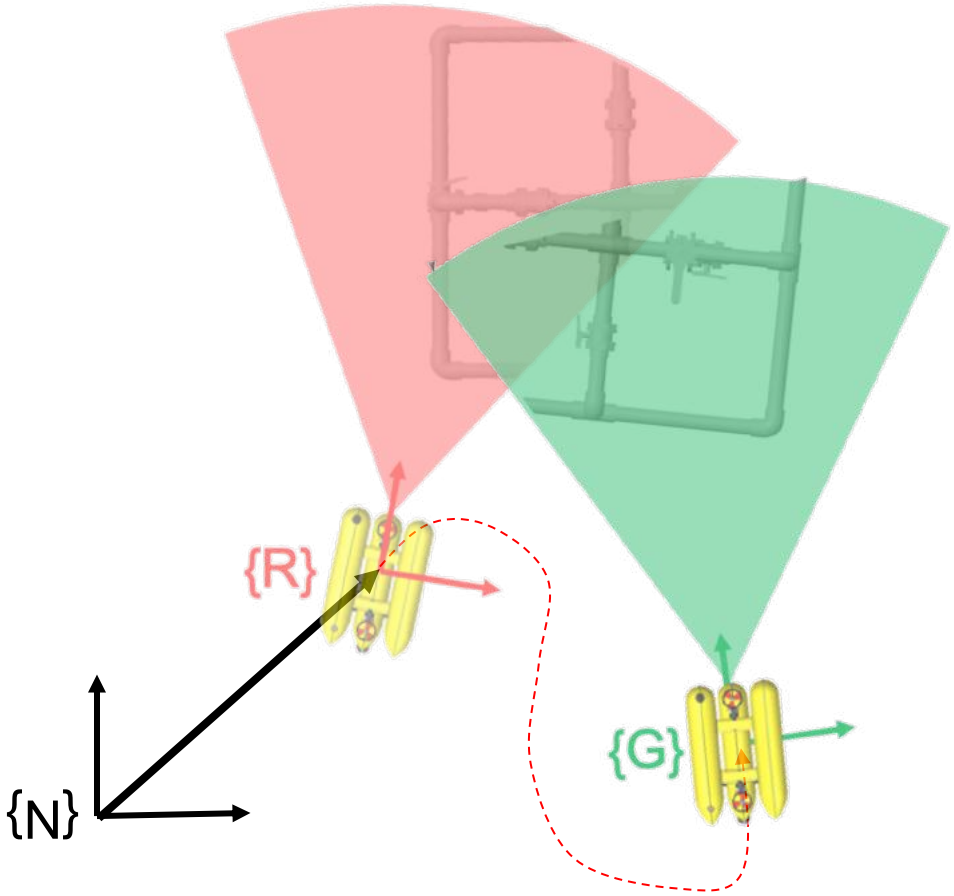


EKF Pose-Based SLAM

> The AUV moves and take another scans from the new View pose

$$\mathcal{M} = [{}^{R}\boldsymbol{S}_{\boldsymbol{R}},\ {}^{G}\boldsymbol{S}_{\boldsymbol{G}}]$$

$${}^{N}\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{R}} \\ {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{G}} \\ {}^{N}\hat{\bar{\boldsymbol{x}}}_{\boldsymbol{B_k}} \end{bmatrix}$$



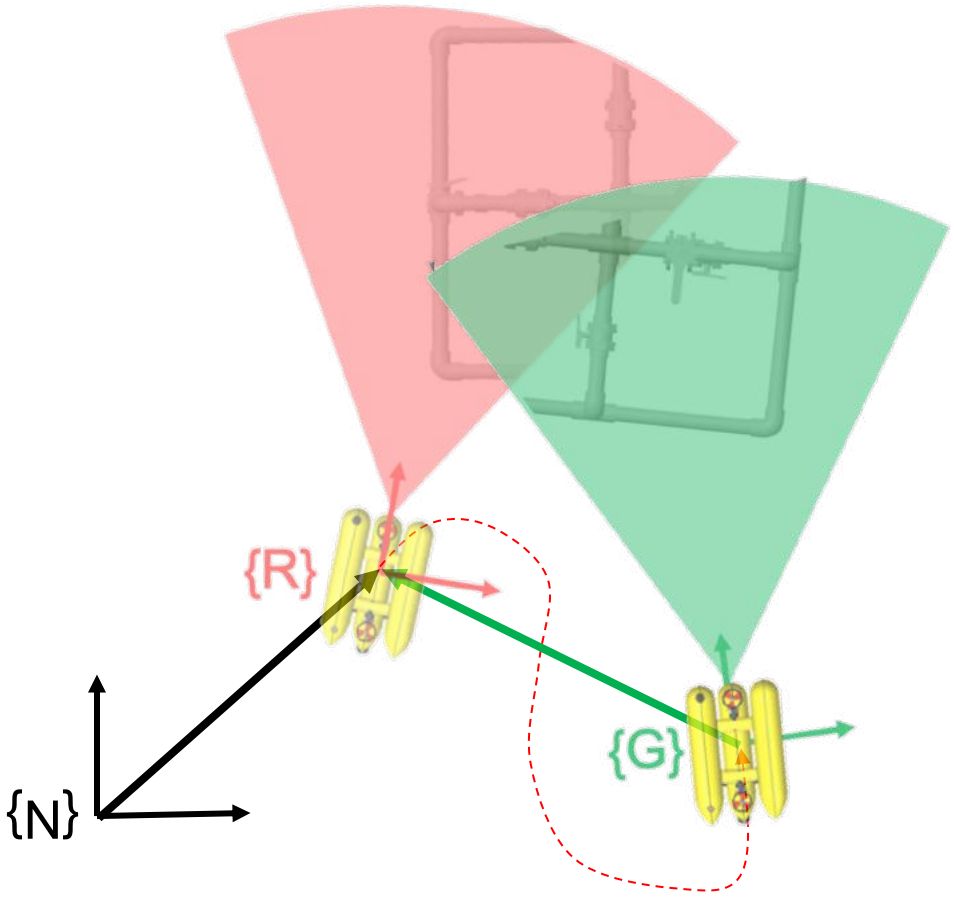
EKF Pose-Based SLAM

> The Observation function computes the expected robot displacement between scans.

$$\mathcal{M} = [{}^{R}S_{R},\ {}^{G}S_{G}]$$

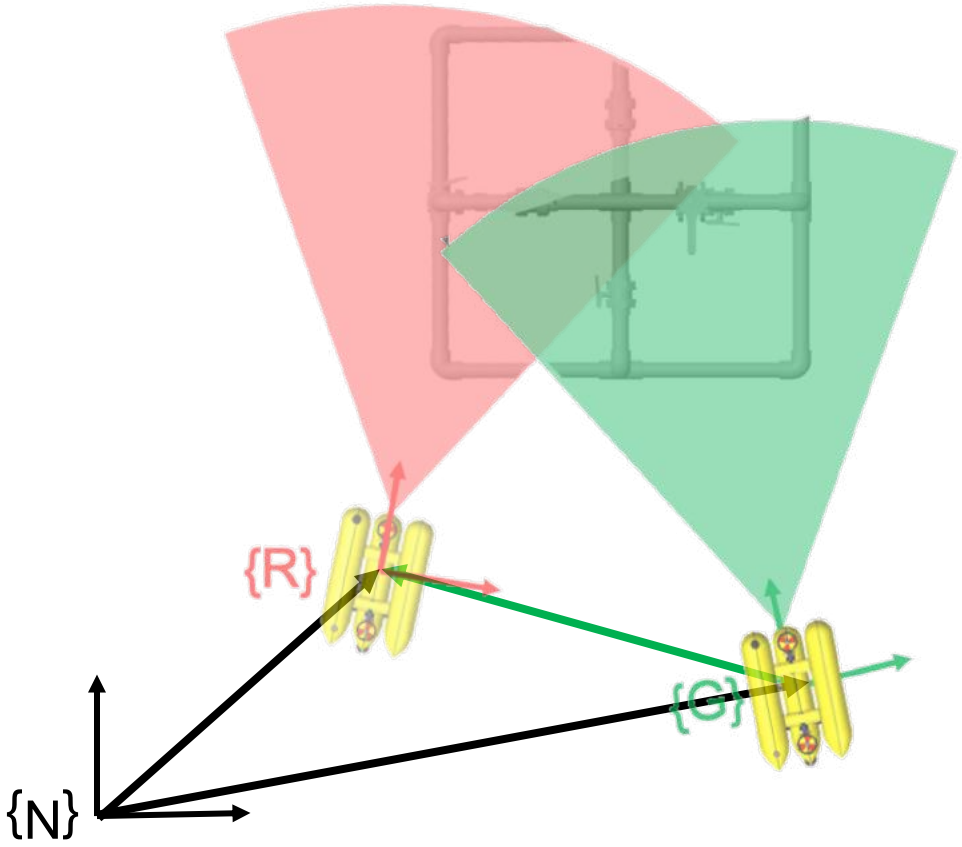
$${}^{N}\hat{\hat{x}}_k = \begin{bmatrix} {}^{N}\hat{x}_R \\ {}^{N}\hat{x}_G \\ {}^{N}\hat{\hat{x}}_{B_k} \end{bmatrix}$$

$$z_{RG} = h_{RG}({}^{N}x_k, v_k) = \ominus {}^{N}x_R \oplus {}^{N}x_G + v_{RG}$$



EKF Pose-Based SLAM

> Scans are registered to get the robot displacement & the PEKFSLAM is updated



$$\mathcal{M} = [{}^{R}S_{R},\ {}^{G}S_{G}]$$

$${}^{N}\hat{\boldsymbol{x}}_{k} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{R} \\ {}^{N}\hat{\boldsymbol{x}}_{G} \\ {}^{N}\hat{\boldsymbol{x}}_{B_k} \end{bmatrix}$$

$$z_{RG} = h_{RG}({}^{N}x_{k}, v_{k}) = \ominus {}^{N}x_{R} \oplus {}^{N}x_{G} + v_{RG}$$

$$[\boldsymbol{z_{RG}}, \boldsymbol{R_{RG}}] = Register({}^{R}S_{R}, {}^{G}S_{G})$$

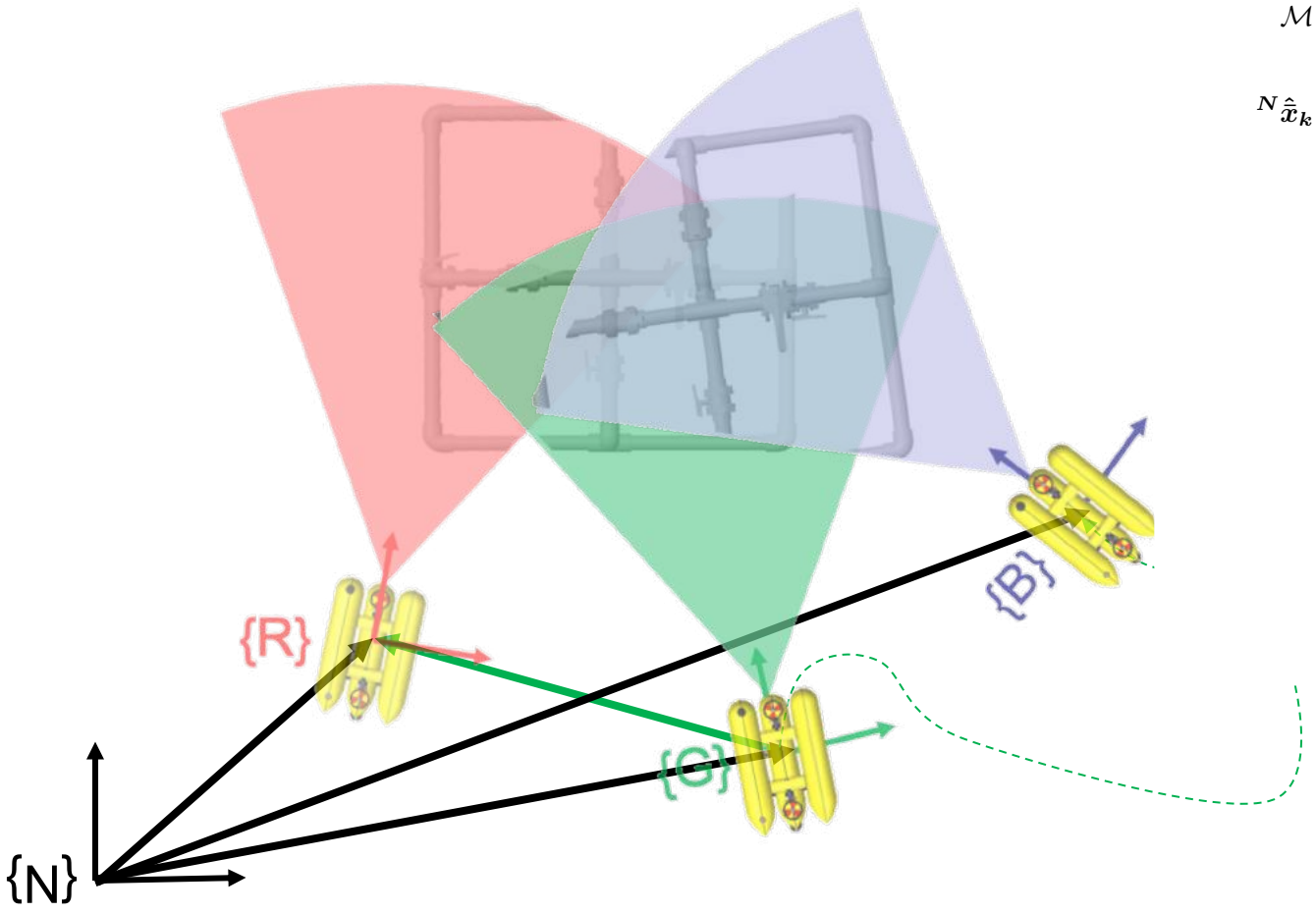
$${}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{G}} \\ {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{R}} \\ {}^{N}\hat{\boldsymbol{x}}_{\boldsymbol{B_k}} \end{bmatrix} = Update(\cdot)$$

EKF Pose-Based SLAM

> The AUV moves and take another scans from the new View pose

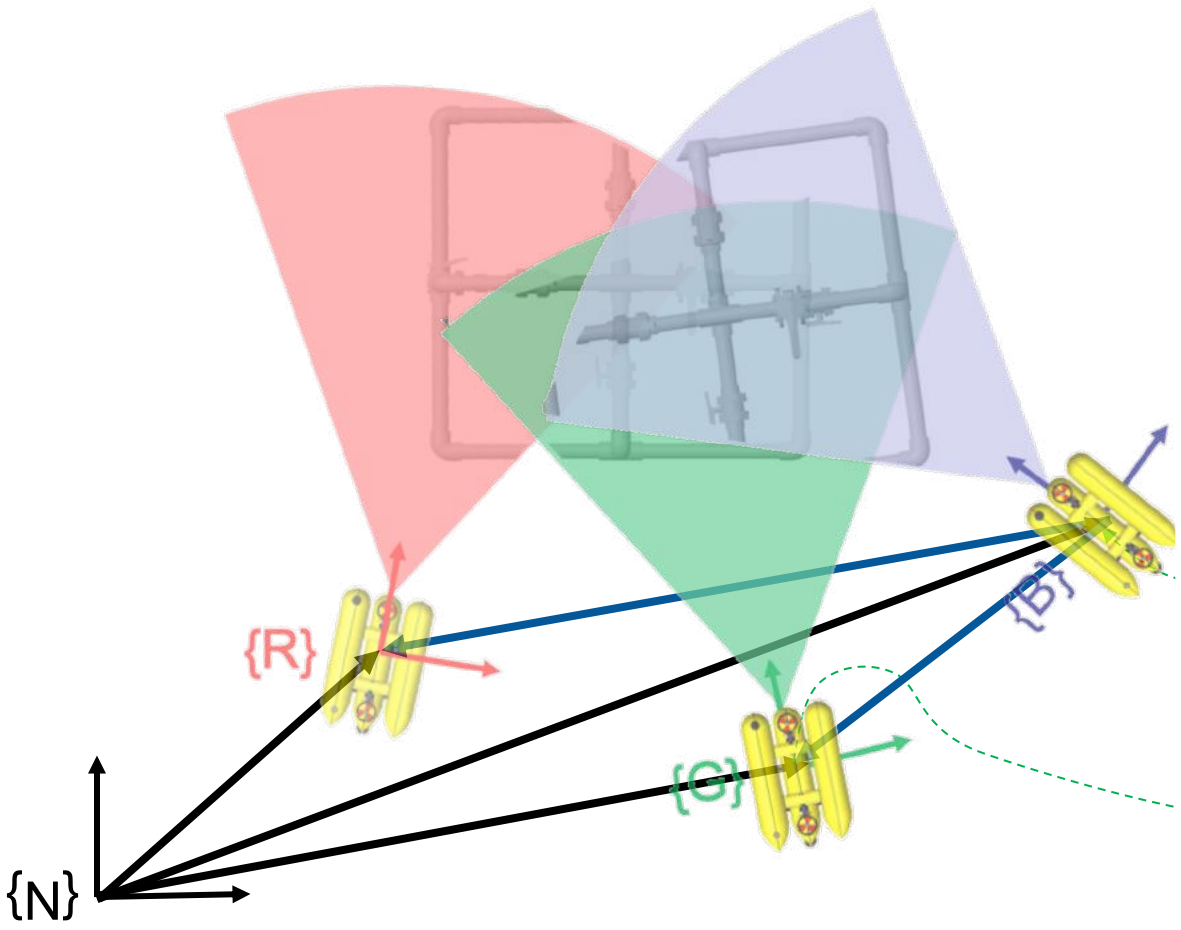
$$\mathcal{M} = [{}^{R}\boldsymbol{S}_{R},\ {}^{G}\boldsymbol{S}_{G},\ {}^{B}\boldsymbol{S}_{B}]$$

$${}^{N}\hat{\hat{\boldsymbol{x}}}_{k} = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_{R} \\ {}^{N}\hat{\boldsymbol{x}}_{G} \\ {}^{N}\hat{\boldsymbol{x}}_{B} \\ {}^{N}\hat{\hat{\boldsymbol{x}}}_{B_k} \end{bmatrix}$$



EKF Pose-Based SLAM

> The Observation function computes the expected robot displacement between scans.



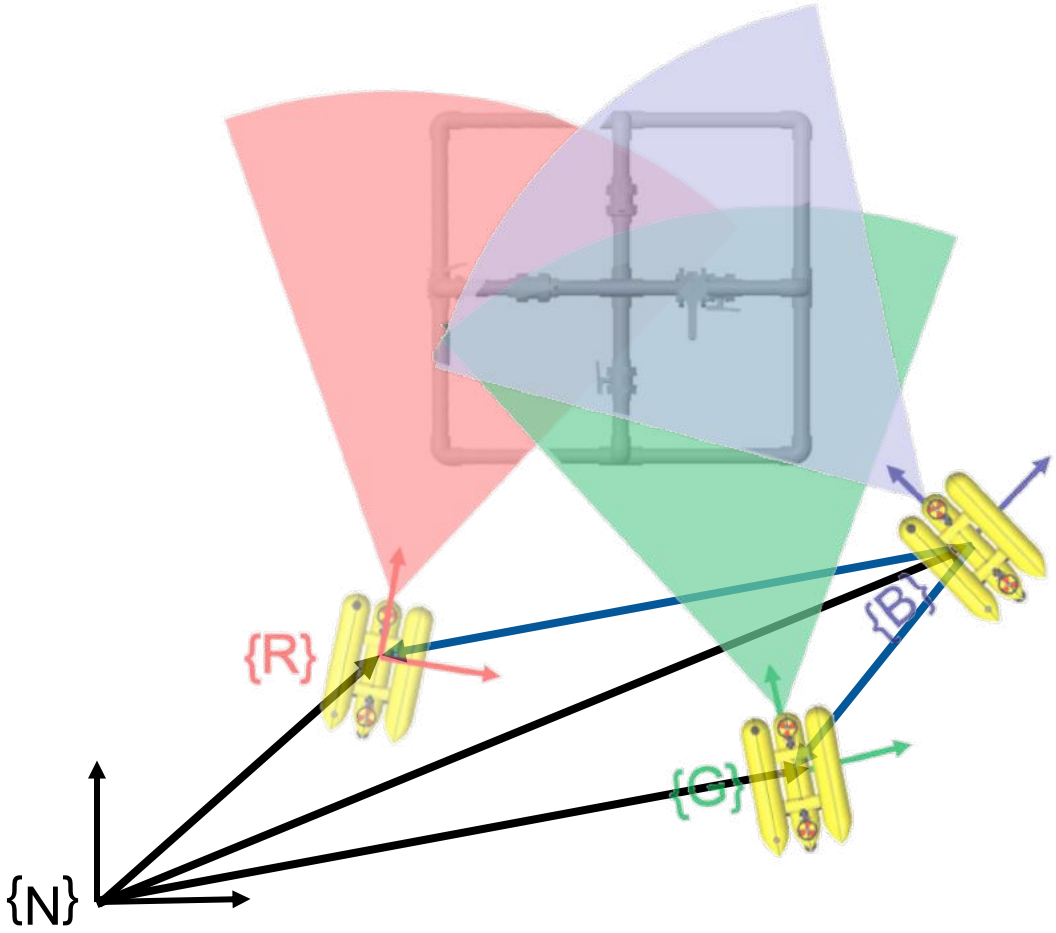
$$\mathcal{M} = [{}^{R}S_{R},\ {}^{G}S_{G},\ {}^{B}S_{B}]$$

$${}^{N}\hat{\bar{x}}_k = \begin{bmatrix} {}^{N}\hat{x}_R \\ {}^{N}\hat{x}_G \\ {}^{N}\hat{x}_B \\ {}^{N}\hat{\bar{x}}_{B_k} \end{bmatrix}$$

$$z_k = \begin{bmatrix} h_{BG}({}^{N}x_k, v_k) \\ h_{BR}({}^{N}x_k, v_k) \end{bmatrix}$$

EKF Pose-Based SLAM

> Scans are registered to get the robot displacement & the PEKFSLAM is updated



$$\mathcal{M} = [{}^{R}S_{R},\ {}^{G}S_{G},\ {}^{B}S_{B}]$$

$${}^{N}\hat{\bar{x}}_k = \begin{bmatrix} {}^{N}\hat{x}_R \\ {}^{N}\hat{x}_G \\ {}^{N}\hat{x}_B \\ {}^{N}\hat{\bar{x}}_{B_k} \end{bmatrix}$$

$$z_k = \begin{bmatrix} h_{BG}({}^{N}x_k, v_k) \\ h_{BR}({}^{N}x_k, v_k) \end{bmatrix} = \begin{bmatrix} \ominus {}^{N}x_B \oplus {}^{N}x_G + v_{BG} \\ \ominus {}^{N}x_B \oplus {}^{N}x_R + v_{BR} \end{bmatrix}$$

$$[\boldsymbol{z_{BG}}, \boldsymbol{R_{BG}}] = Register({}^{B}S_B, {}^{G}S_G)$$

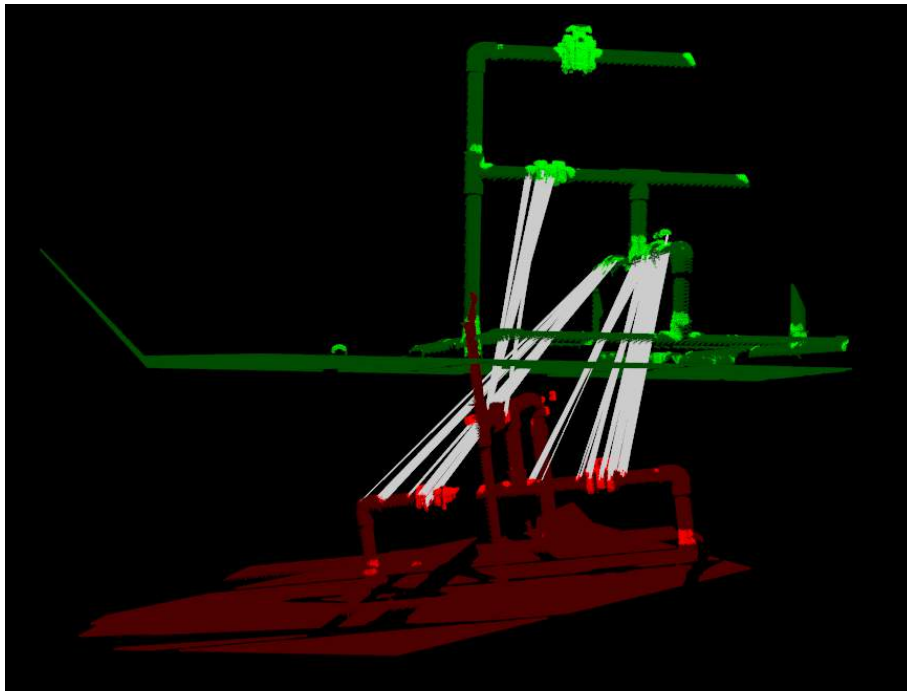
$$[\boldsymbol{z_{BR}}, \boldsymbol{R_{BR}}] = Register({}^{B}S_B, {}^{R}S_R)$$

$$\boldsymbol{z_k} = \begin{bmatrix} \boldsymbol{z_{BG}} \\ \boldsymbol{z_{BR}} \end{bmatrix} \ ; \ \boldsymbol{R_k} = \begin{bmatrix} \boldsymbol{R_{BG}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_{BR}} \end{bmatrix}$$

$${}^{N}\hat{\boldsymbol{x}}_k = \begin{bmatrix} {}^{N}\hat{\boldsymbol{x}}_G \\ {}^{N}\hat{\boldsymbol{x}}_R \\ {}^{N}\hat{\boldsymbol{x}}_B \\ {}^{N}\hat{\boldsymbol{x}}_{B_k} \end{bmatrix} = Update(\ldots, \boldsymbol{z_k}, \boldsymbol{R_k})$$

Registration algorithm

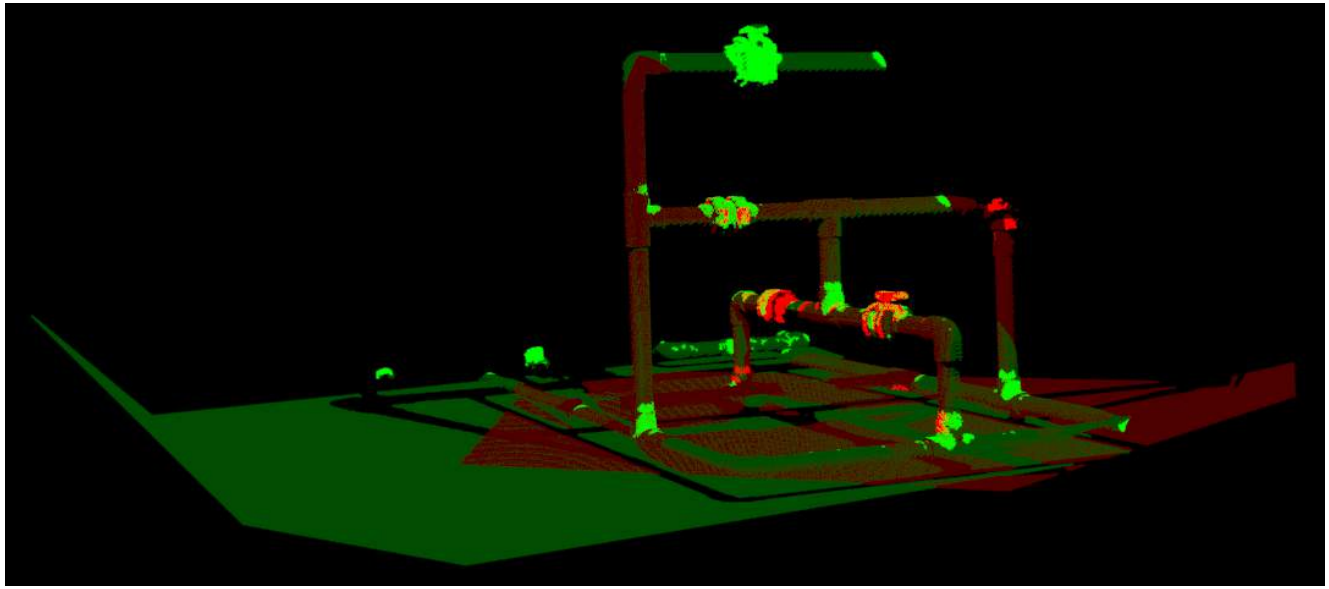
- Coarse registration
  - Feature association
  - Roto-translation using Singular Value Decomposition (J. Besl and McKay, 1992)



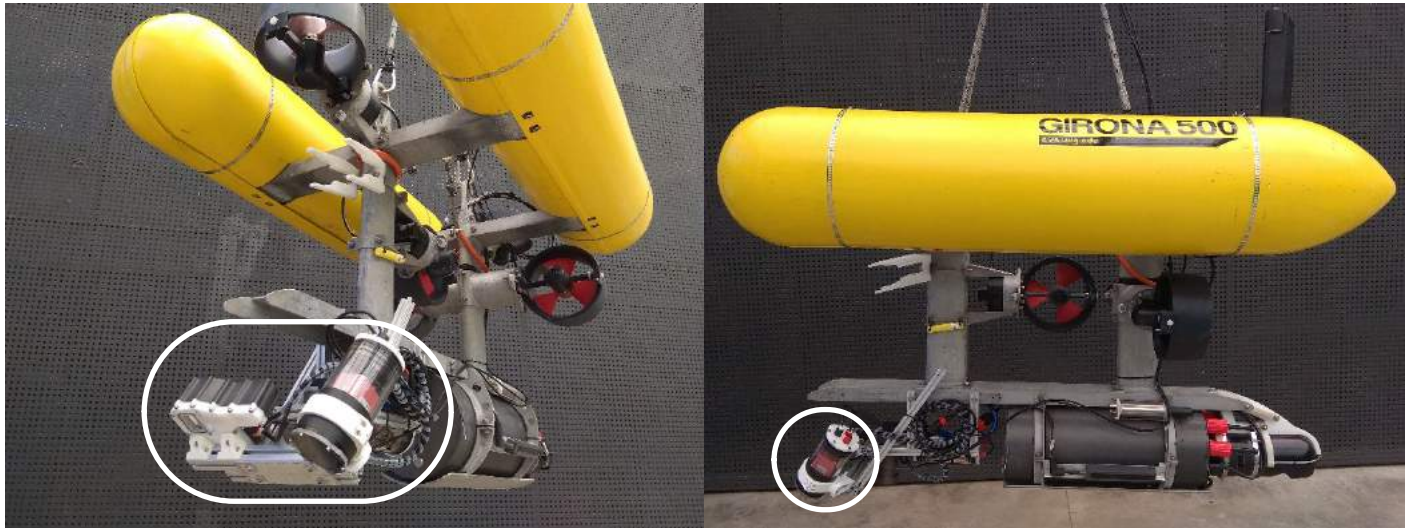
Registration algorithm

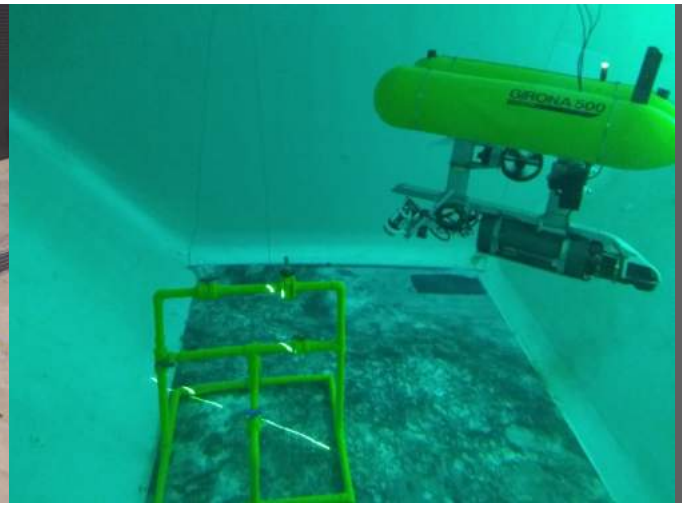
- Coarse registration
  - Feature association
  - Roto-translation using Singular Value Decomposition (J. Besl and McKay, 1992)

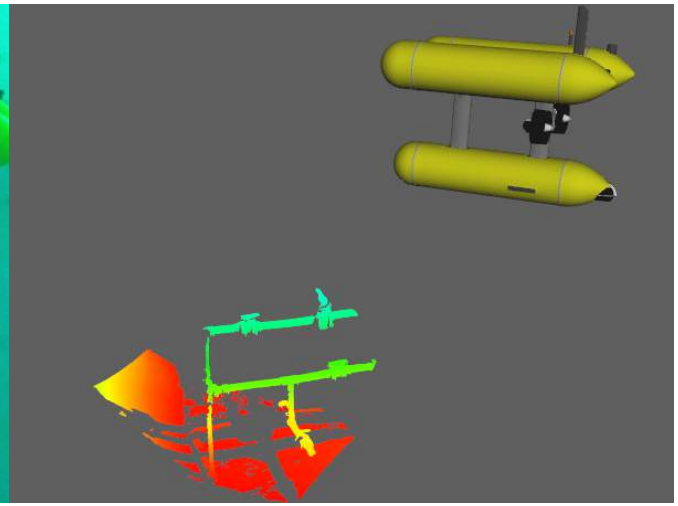
- Fine registration: Point to point ICP



Experiments and results

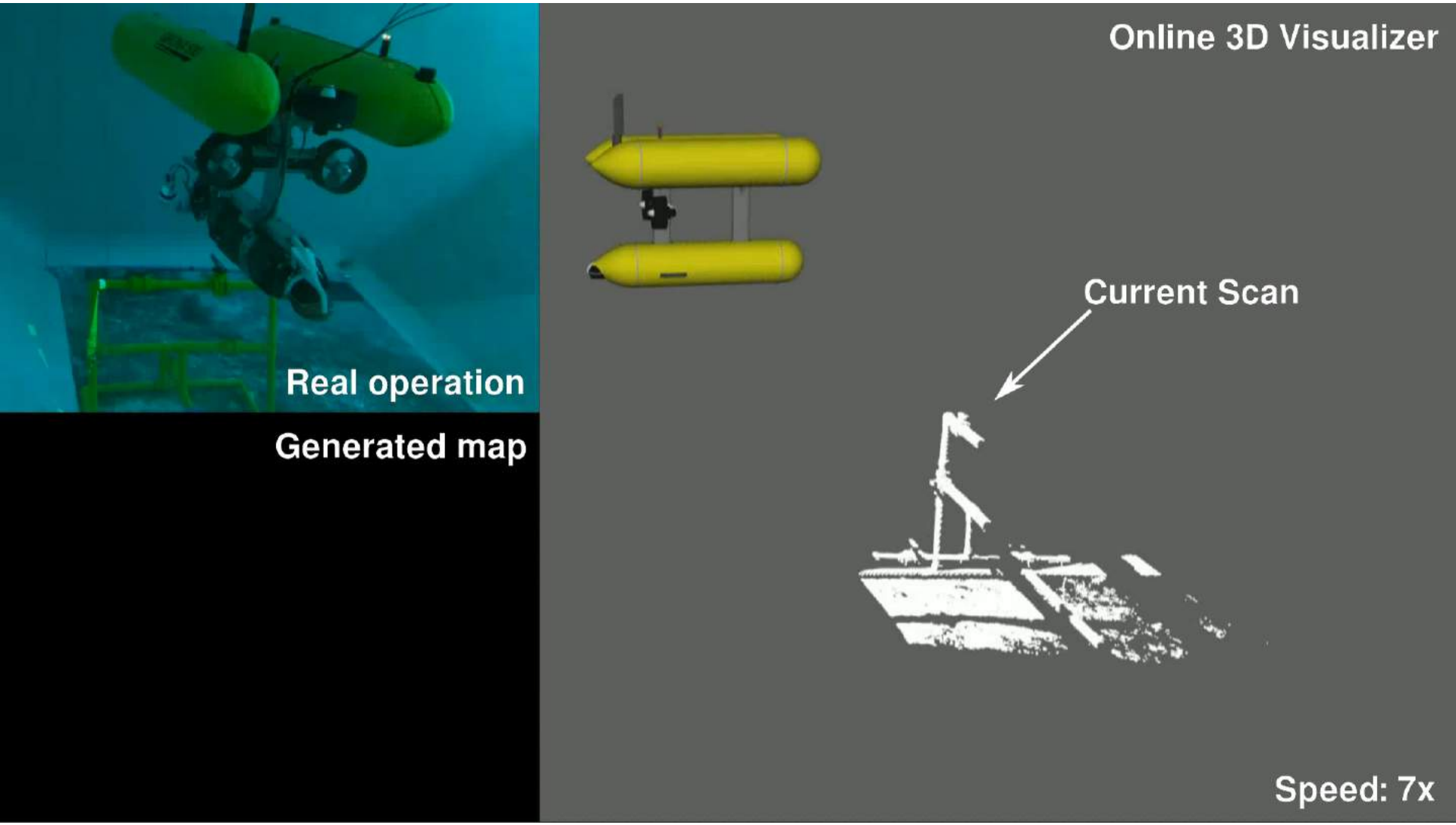






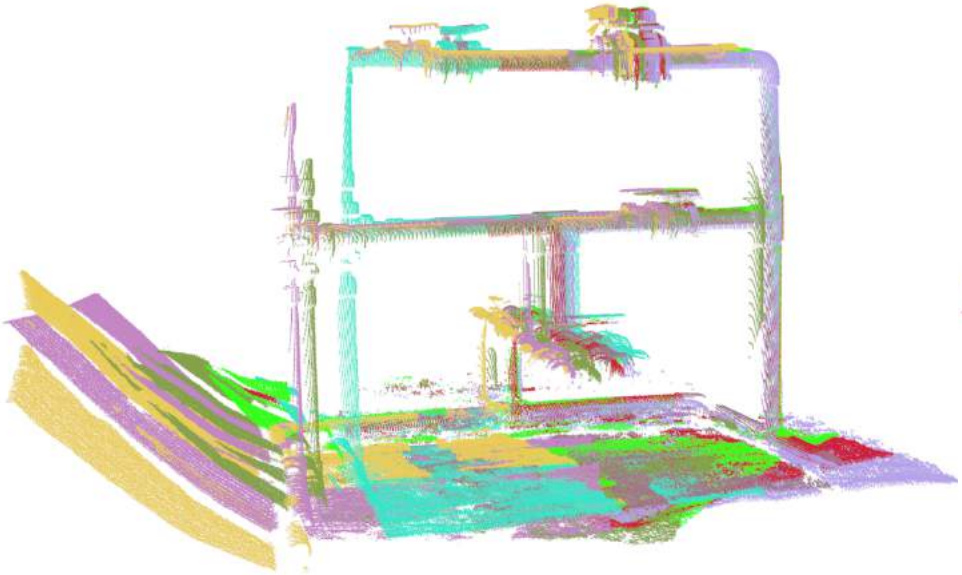
Pose based EKF SLAM using Point Clouds

Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner

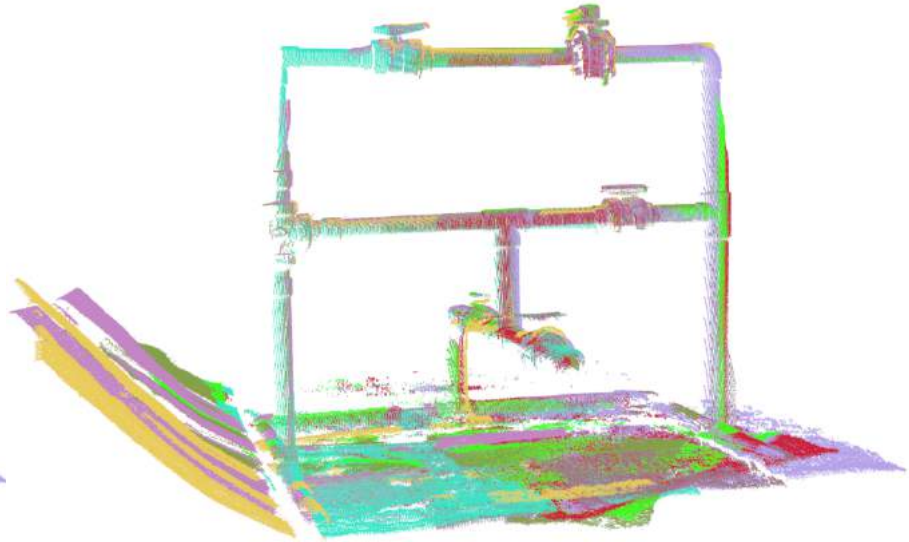


Experiments and results

Dead reckoning

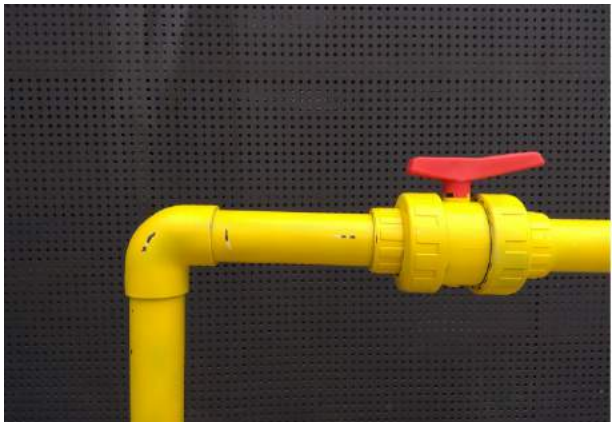
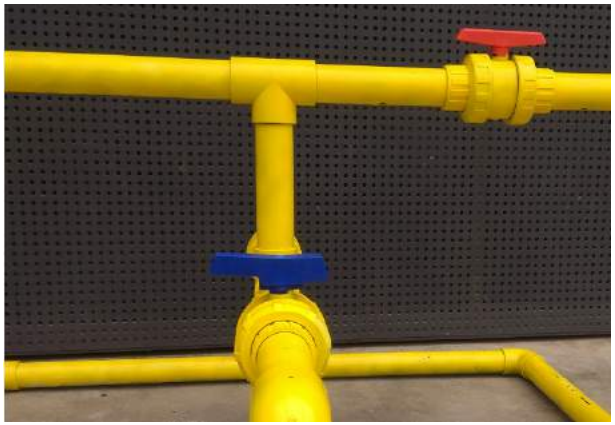


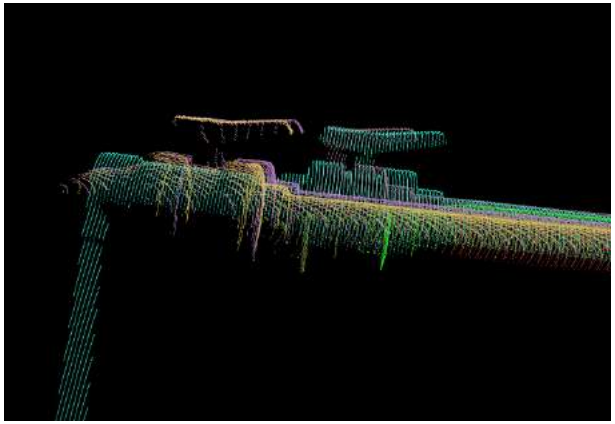
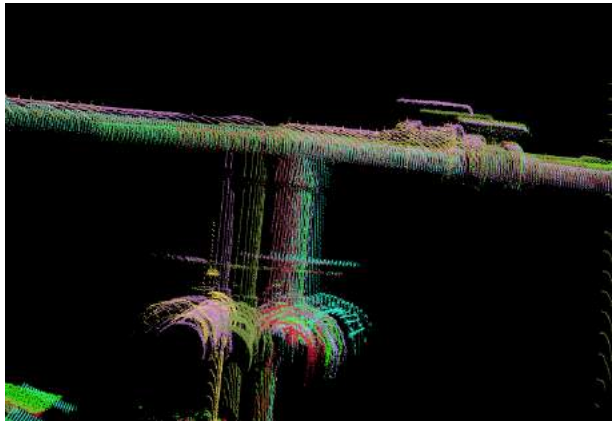
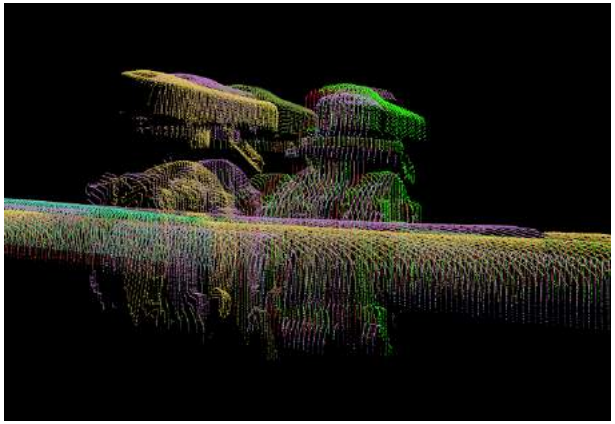
SLAM



Experiments and results

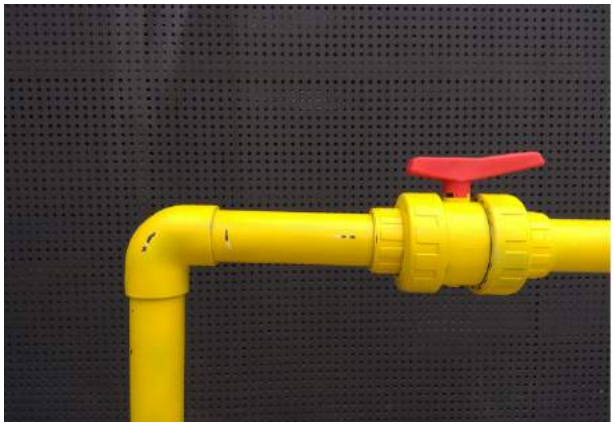
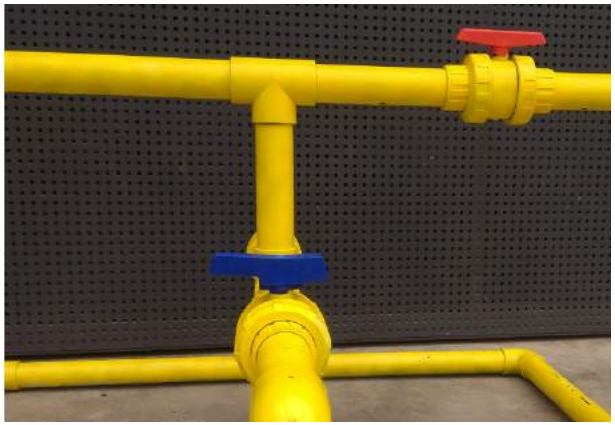
Structure details: dead reckoning navigation

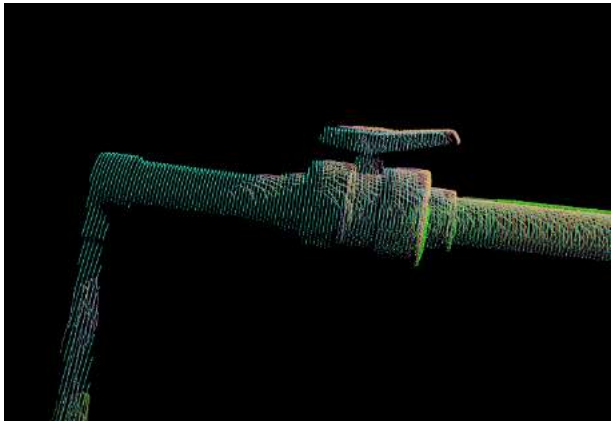
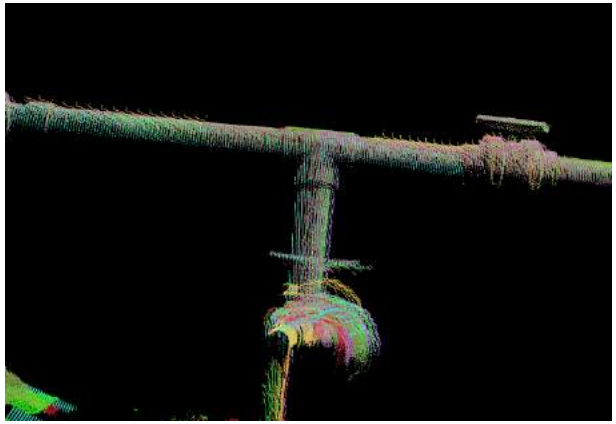
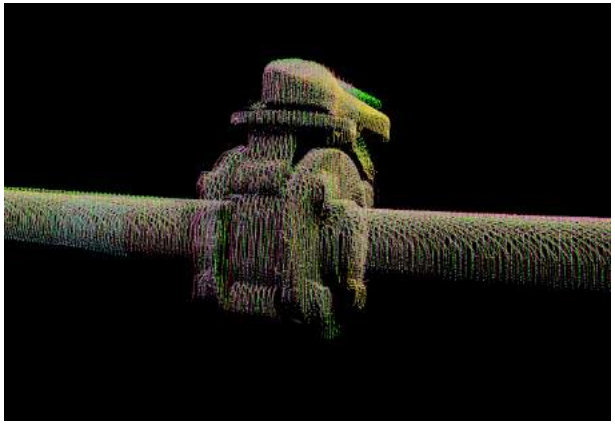



Experiments and results

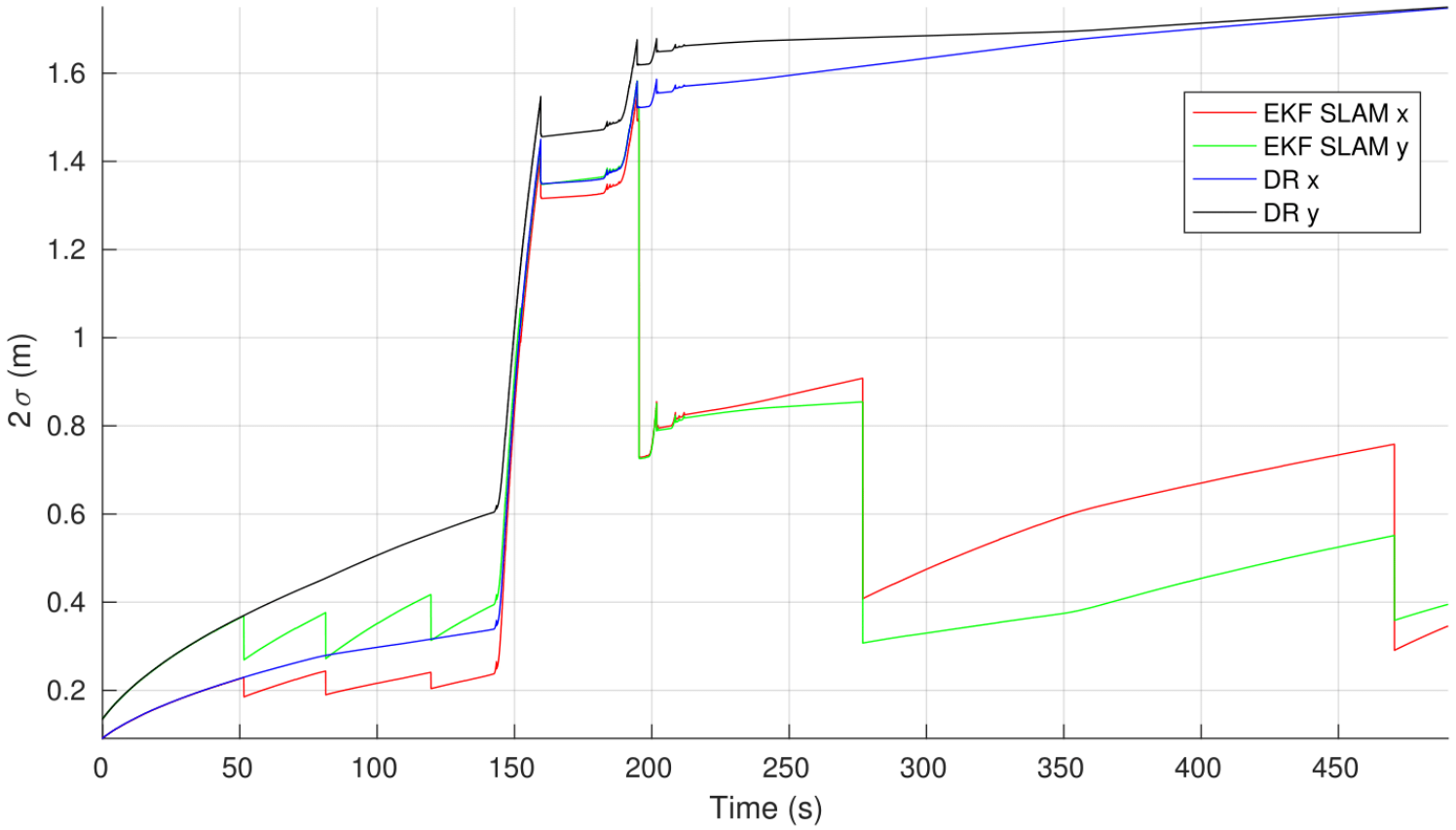
Structure details: SLAM

Experiments and results

Robot uncertainty



Universitat de Girona

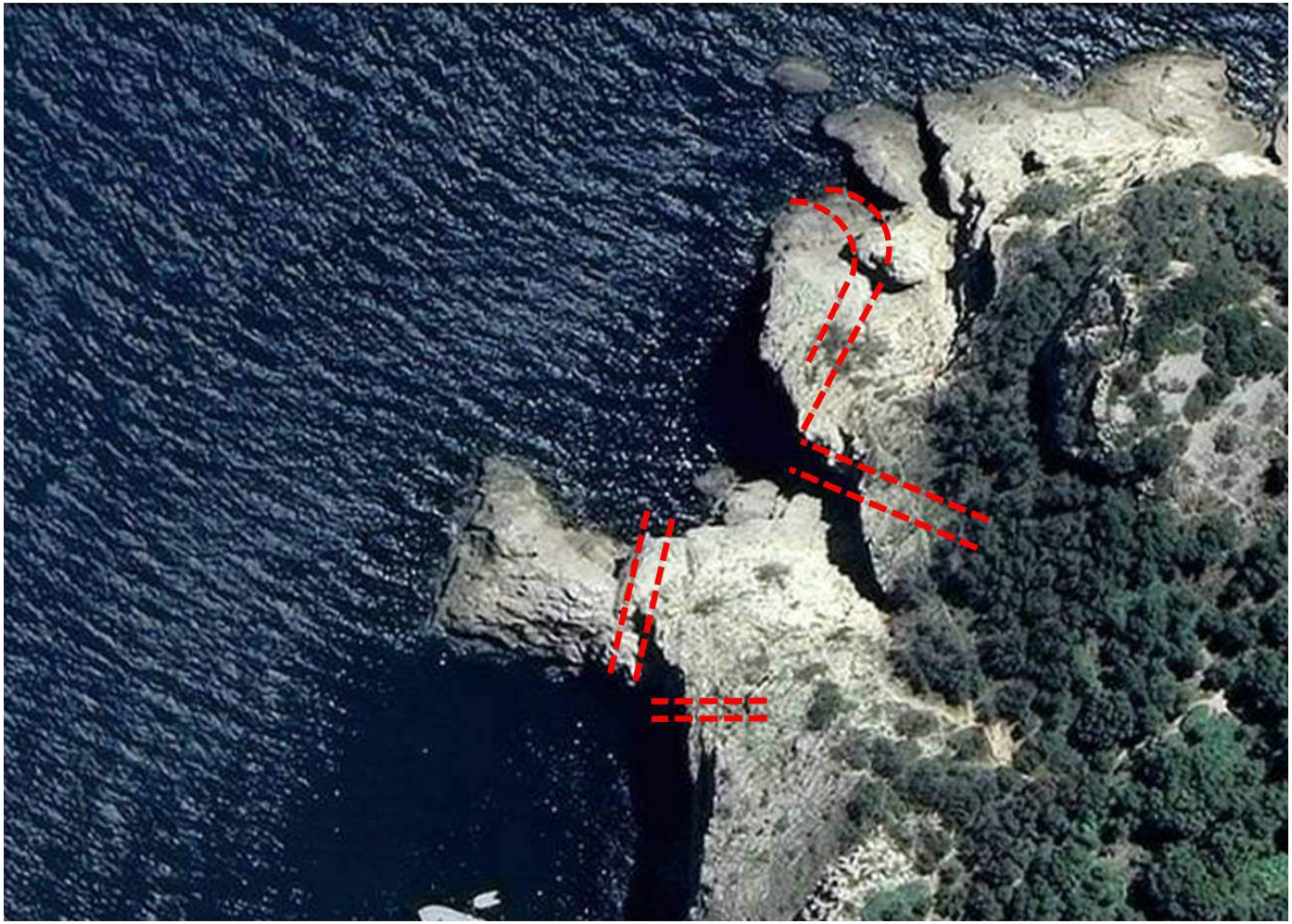
GIRONA UNDERWATER VISION AND ROBOTICS

Sonar Scan Matching for Simultaneous Localization and Mapping in Confined Underwater Environments

A. Mallios
P. Ridao & D. Ribas
pere@eia.udg.edu
http://cirs.udg.edu

SLAM objectives

*"The development of a **localization and mapping** technique for an autonomous underwater vehicle that navigates in **confined** and possibly **unstructured** environments."*



Method Localization($\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0$)

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$;; // Read the Scan from the sensor
- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [{}^{B_0}\boldsymbol{S}_0]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u_k}, \boldsymbol{Q_k}] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u_k}, \boldsymbol{Q_k})$;
  - $[\boldsymbol{z_m}, \boldsymbol{R_m}] = GetMeasurement()$; // Read navigation sensors
  - **if** *ScanAvailable* **then**
    - $[{}^{B_k}\boldsymbol{S_k}, {}^{B_k}\boldsymbol{R}_{S_k}] = GetScan()$;
    - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k] = AddNewPose({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = {}^{B_k}\boldsymbol{S_k}$; // Store the scan in the map
    - $\mathcal{H}_p = OverlappingScans({}^N\hat{\boldsymbol{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - **for** $i = 1$ *to* $length(\mathcal{H}_p)$ **do** // for all overlaps
      - $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map
      - ${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scans Displacement guess
      - $[\boldsymbol{z_p}, \boldsymbol{R_p}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
      - $i = i + 1$; // Go to next pair
  - $\mathcal{H}_p = DataAssociation({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_p}, \boldsymbol{R_p})$; // select compatible registrations
  - $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\boldsymbol{x}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \mathcal{H}_p)$;

> To implement a PEKFSLAM we need to:

1. Program the ***GetInput(·)*** function
2. Define the **motion model:** ${}^N\bar{\boldsymbol{x}}_{B_k} = \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)$
   - Compute the motion model Jacobians:

$$\boldsymbol{J}_{\boldsymbol{f_x}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial {}^N\boldsymbol{x}_{B_{k-1}}} \quad , \quad \boldsymbol{J}_{\boldsymbol{f_w}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial \boldsymbol{w_k}}$$

3. Program the ***GetScan()*** function
4. Define the **observation model:**

$$\boldsymbol{z_k} = \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{p_k}})$$

$$\begin{bmatrix} \boldsymbol{z_{ij}} \\ \vdots \\ \boldsymbol{z_{pq}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{ij_k}}) \\ \vdots \\ \boldsymbol{h_{pq}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{pq_k}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta x_{ij}} \\ \vdots \\ \boldsymbol{\Delta x_{pq}} \end{bmatrix}$$

   - Compute the observation Jacobians:

$\forall i, j$ compute:

$$\boldsymbol{J}_{\boldsymbol{h_x}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \quad , \quad \boldsymbol{J}_{\boldsymbol{h_v}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial \boldsymbol{v_k}}$$

5. Implement the ***Register(·)*** *function*

Motion Model

> The vehicle state is predicted using a constant velocity **motion model**:

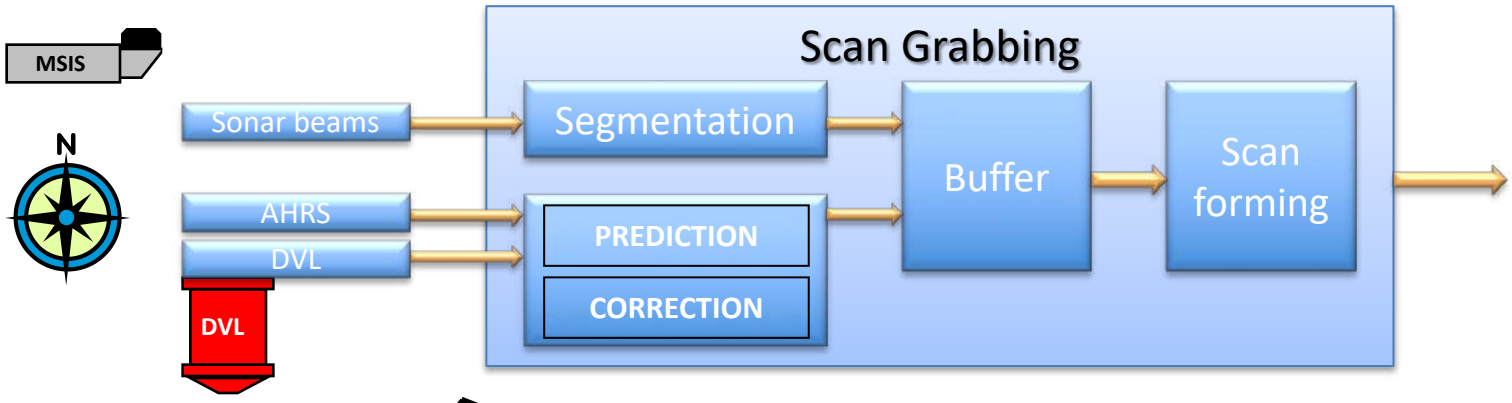
$$ {}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}} = \underbrace{[{}^{N}x_B \ {}^{N}y_B \ {}^{N}z_B \ {}^{N}\psi_B}_{\text{Pose } {}^{N}\eta_{B_{k-1}}} | \underbrace{{}^{B}u_B \ {}^{B}v_B \ {}^{B}w_B \ {}^{B}r_B}_{\text{Velocity } {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}}]^T $$

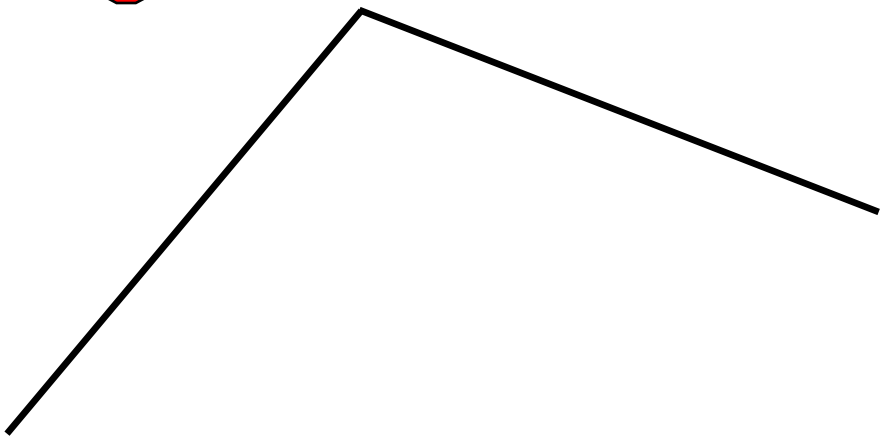
$$ {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}} = \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right) = \begin{bmatrix} {}^{N}\boldsymbol{\eta}_{\boldsymbol{B_{k-1}}} \oplus \left({}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}\Delta t + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\frac{\boldsymbol{\Delta t^2}}{\boldsymbol{2}}\right) \\ {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}} + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\Delta t \end{bmatrix} $$

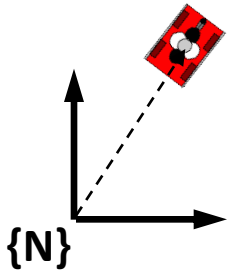
where $\boldsymbol{u_k}$ is not used

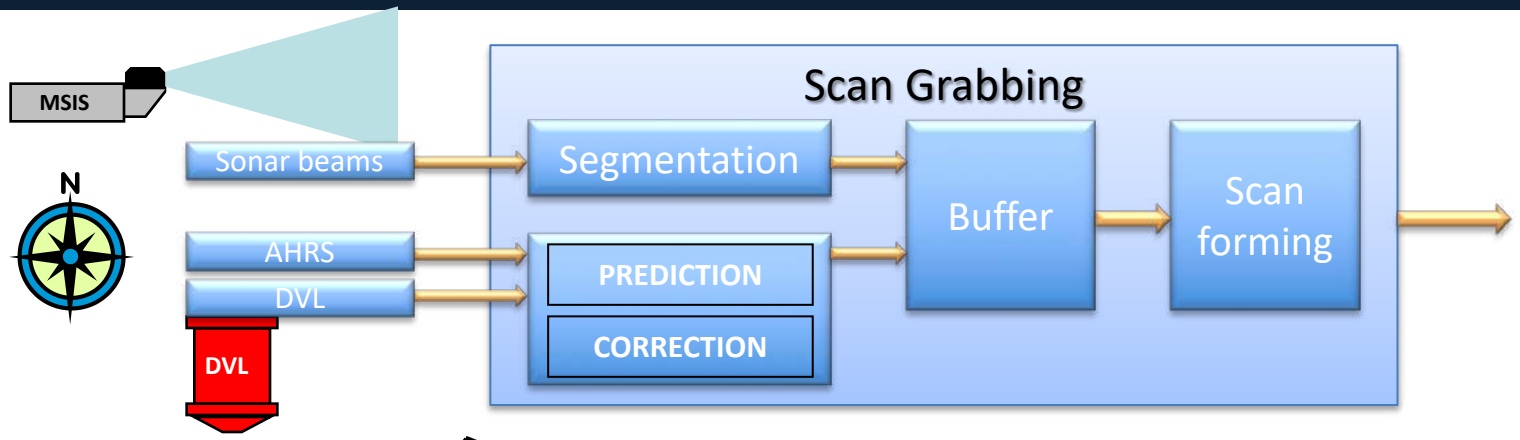
and $\boldsymbol{w_k} = \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}} = [w_{\dot{u}} \ w_{\dot{v}} \ w_{\dot{w}} \ w_{\dot{r}}]^T$ is the acceleration noise

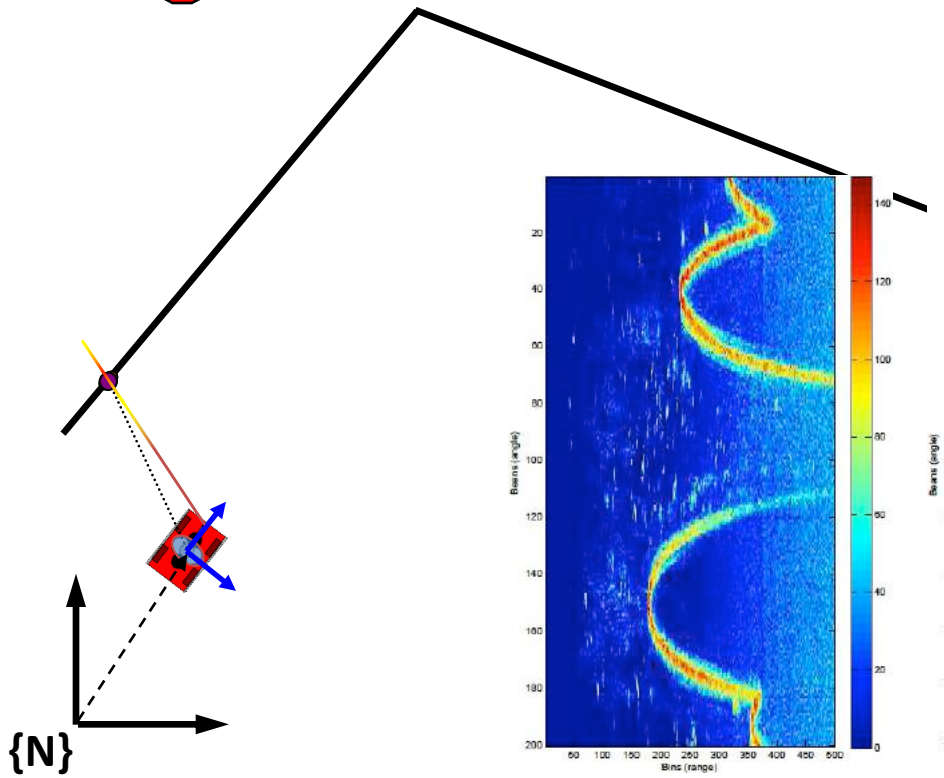
> Thefore the **GetInput()** is not used since u_k is not used by the motion model

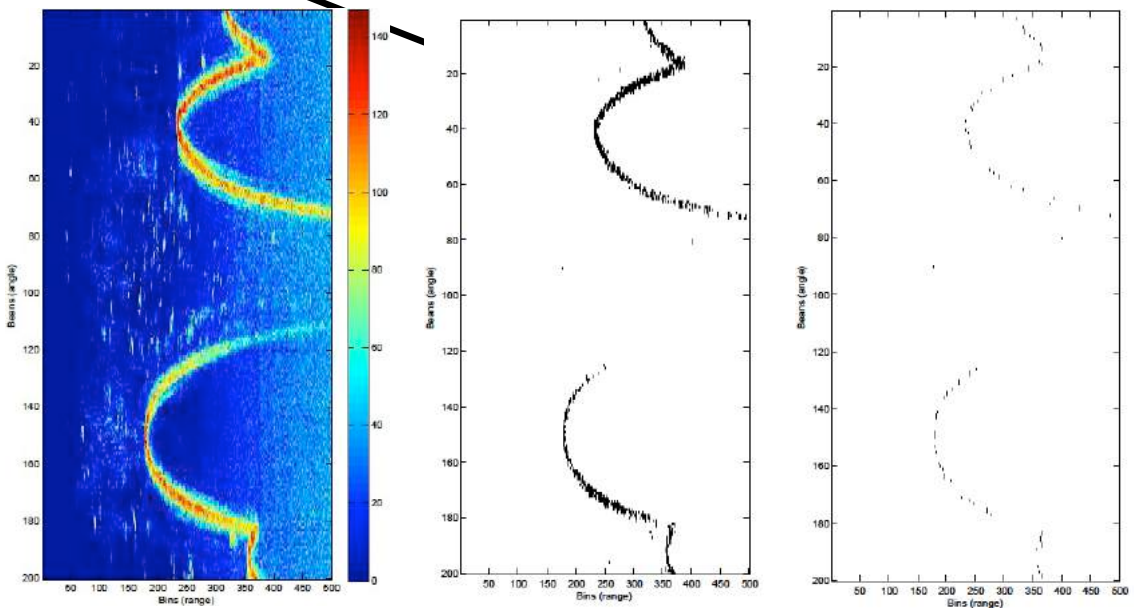

MSIS
N
Sonar beams
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming

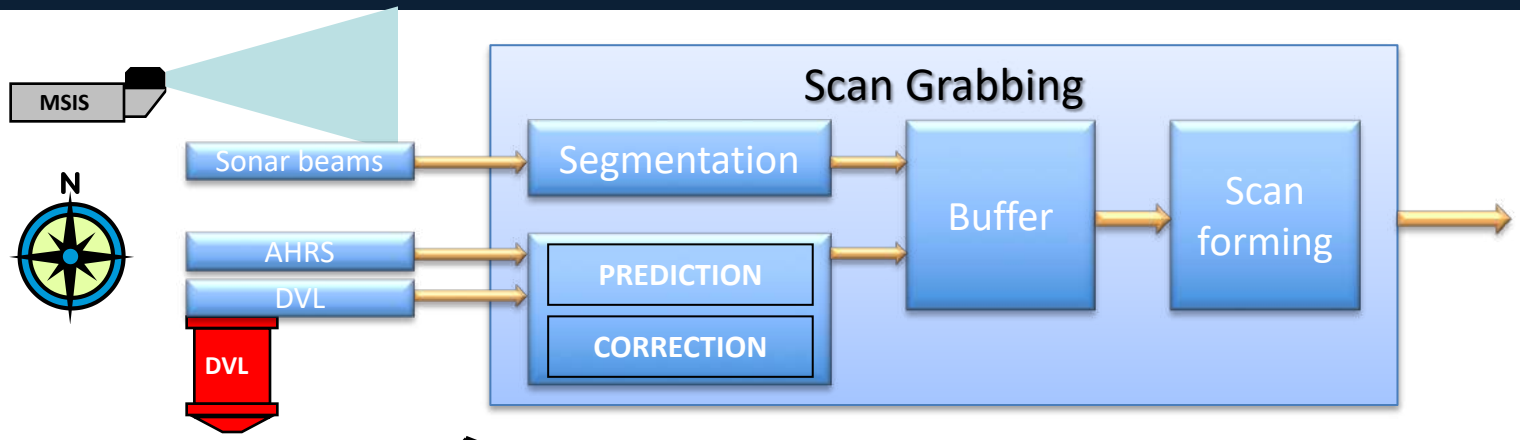


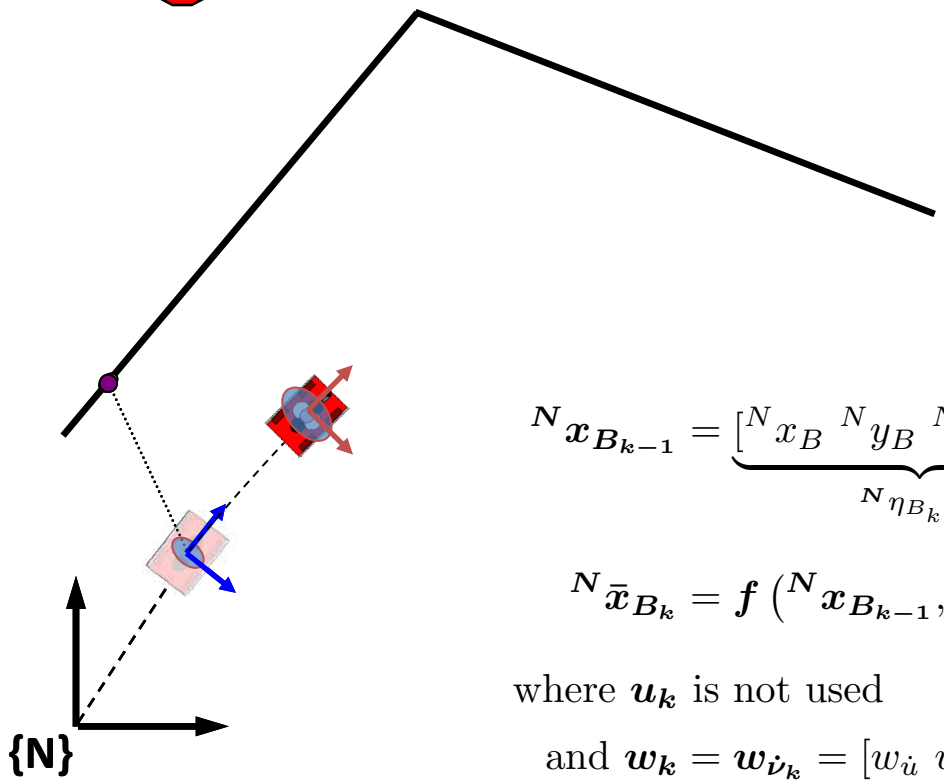

{N}


MSIS
N
Sonar beams
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming


{N}







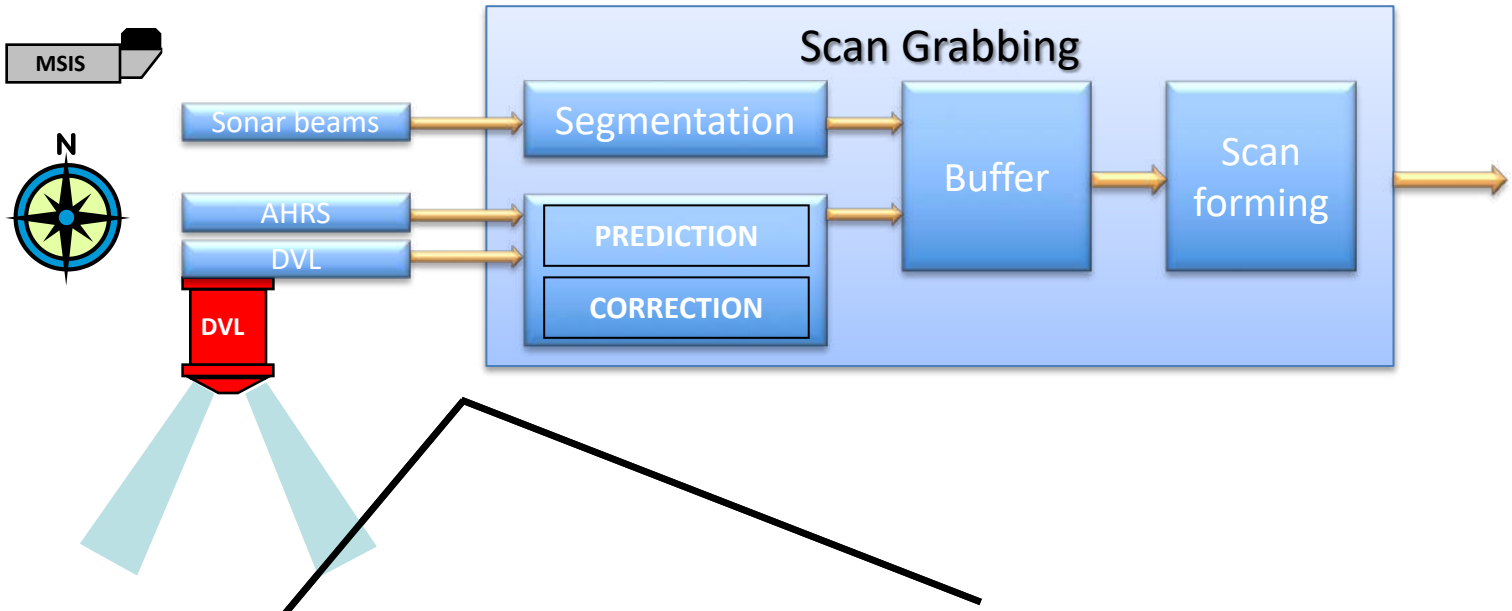
Constant Velocity Kinematic Model:

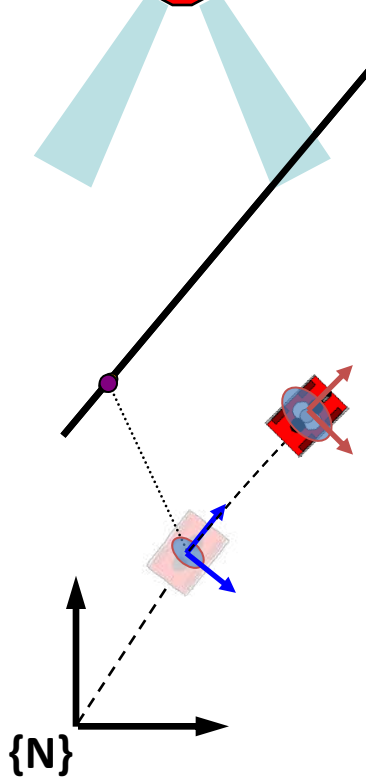
$$ {}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}} = [\underbrace{{}^{N}x_B \ {}^{N}y_B \ {}^{N}z_B \ {}^{N}\psi_B}_{{}^{N}\eta_{B_{k-1}}} | \underbrace{{}^{B}u_B \ {}^{B}v_B \ {}^{B}w_B \ {}^{B}r_B}_{{}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}}]^T $$

$$ {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}} = \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right) = \begin{bmatrix} {}^{N}\boldsymbol{\eta}_{\boldsymbol{B_{k-1}}} \oplus ({}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}\Delta t + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\frac{\boldsymbol{\Delta t^2}}{\boldsymbol{2}}) \\ {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}} + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\Delta t \end{bmatrix} $$

where $\boldsymbol{u_k}$ is not used

and $\boldsymbol{w_k} = \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}} = [w_{\dot{u}} \ w_{\dot{v}} \ w_{\dot{w}} \ w_{\dot{r}}]^T$ is the acceleration noise



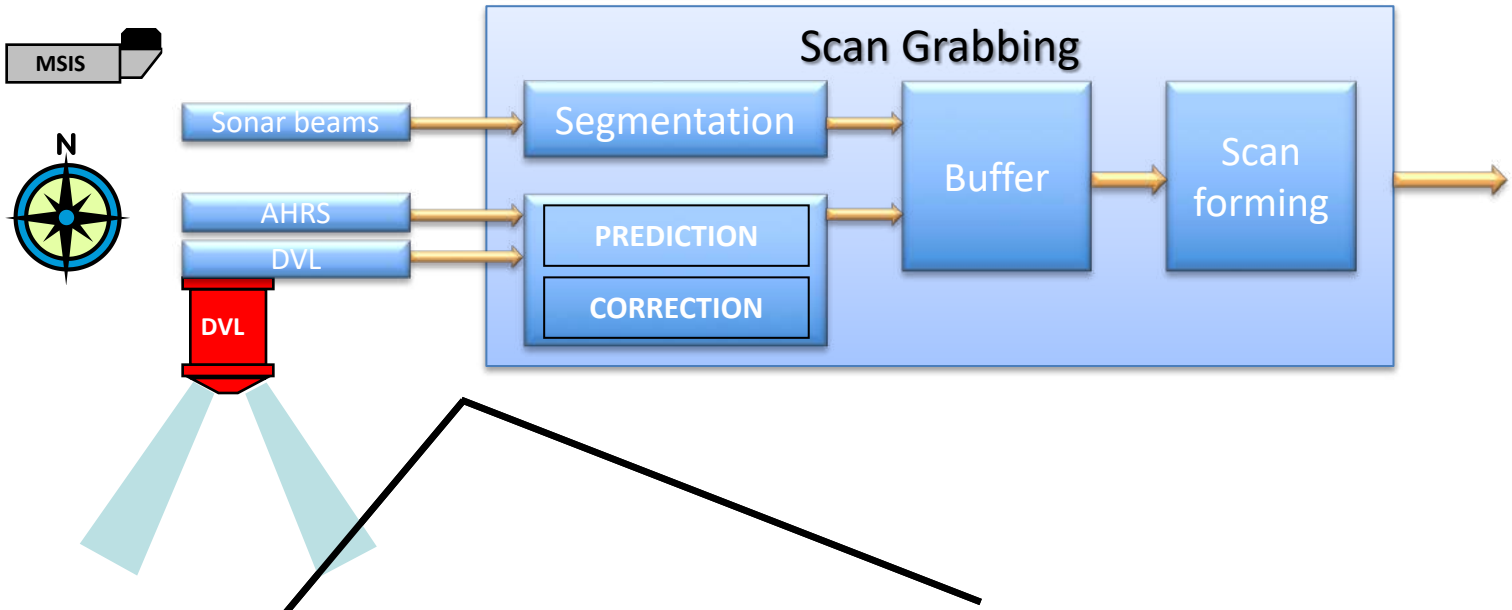


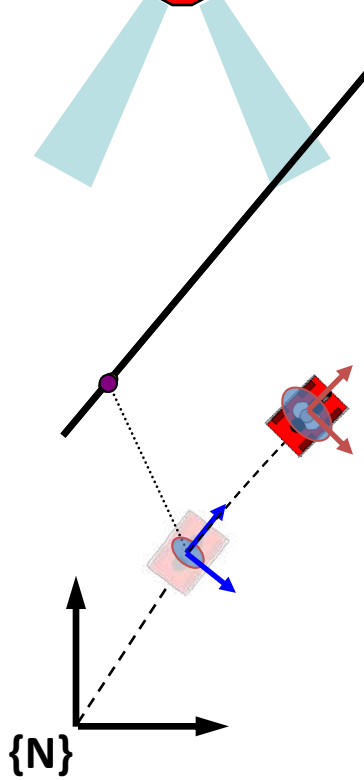
Measurements: DVL, Depth & Compass

$\boldsymbol{z_{DVL}} = \begin{bmatrix} z_u & z_v & z_w \end{bmatrix}^T$ is the lineal velocity observation

$\boldsymbol{v_{DVL}} = \begin{bmatrix} v_u & v_v & v_w \end{bmatrix}^T = \mathcal{N}(\mathbf{0}, \boldsymbol{R_{DVL}})$ is the noise of the velocity observation

$$\boldsymbol{z_{DVL}} = \boldsymbol{H_k} \cdot {}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B_k}} + \boldsymbol{v_{DVL}} \Rightarrow \begin{bmatrix} z_u \\ z_v \\ z_w \\ z_{depth} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^N x_B \\ {}^N y_B \\ {}^N z_B \\ {}^N \psi_B \\ {}^B u_B \\ {}^B v_B \\ {}^B w_B \\ {}^B r_B \end{bmatrix} + \begin{bmatrix} v_u \\ v_v \\ v_w \end{bmatrix}$$



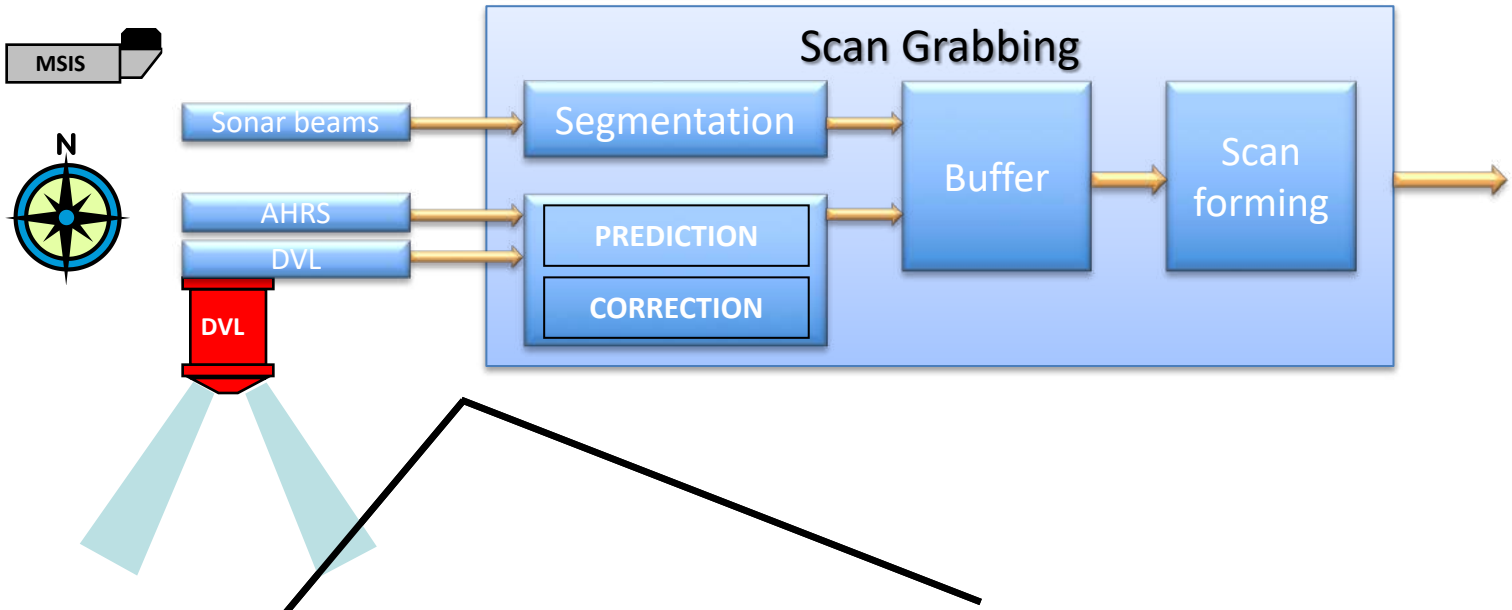


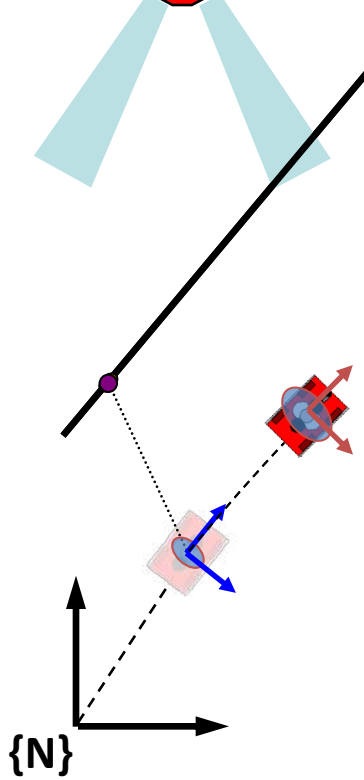
Measurements: DVL, Depth & Compass

$\boldsymbol{z_{depth}} = z_z$ is the depth observation

$\boldsymbol{v_{depth}} = v_z = \mathcal{N}(\mathbf{0}, \sigma^2_{depth})$ is the noise of the depth observation

$$\boldsymbol{z_{DVL}} = \boldsymbol{H_k} \cdot {}^{\boldsymbol{N}}\boldsymbol{x}_{\boldsymbol{B_k}} + \boldsymbol{v_{depth}} \Rightarrow z_z = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^N x_B \\ {}^N y_B \\ {}^N z_B \\ {}^N \psi_B \\ {}^B u_B \\ {}^B v_B \\ {}^B w_B \\ {}^B r_B \end{bmatrix} + v_z$$



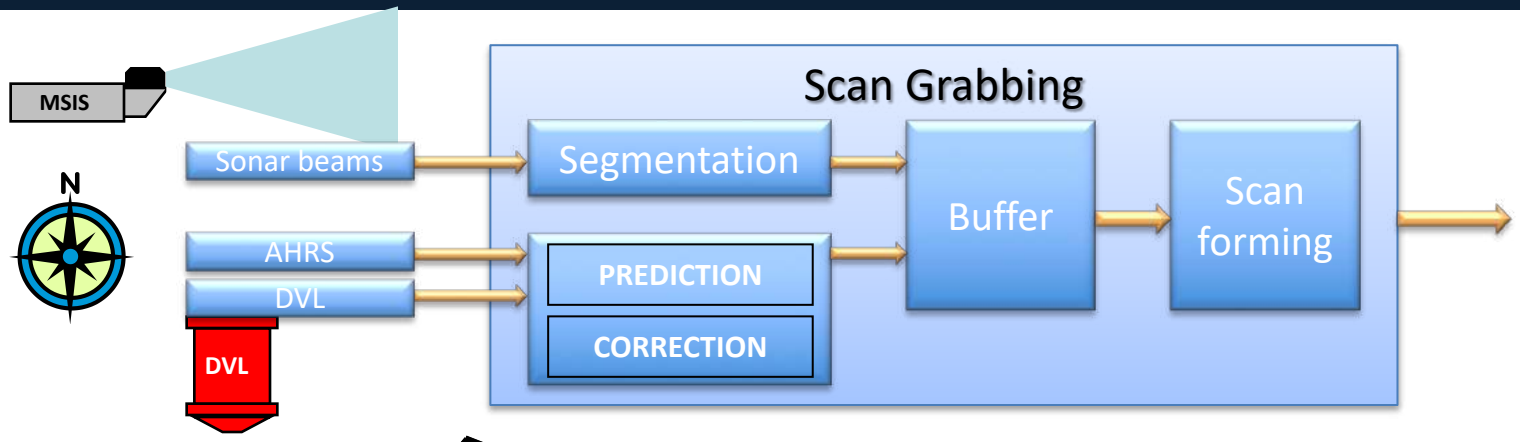


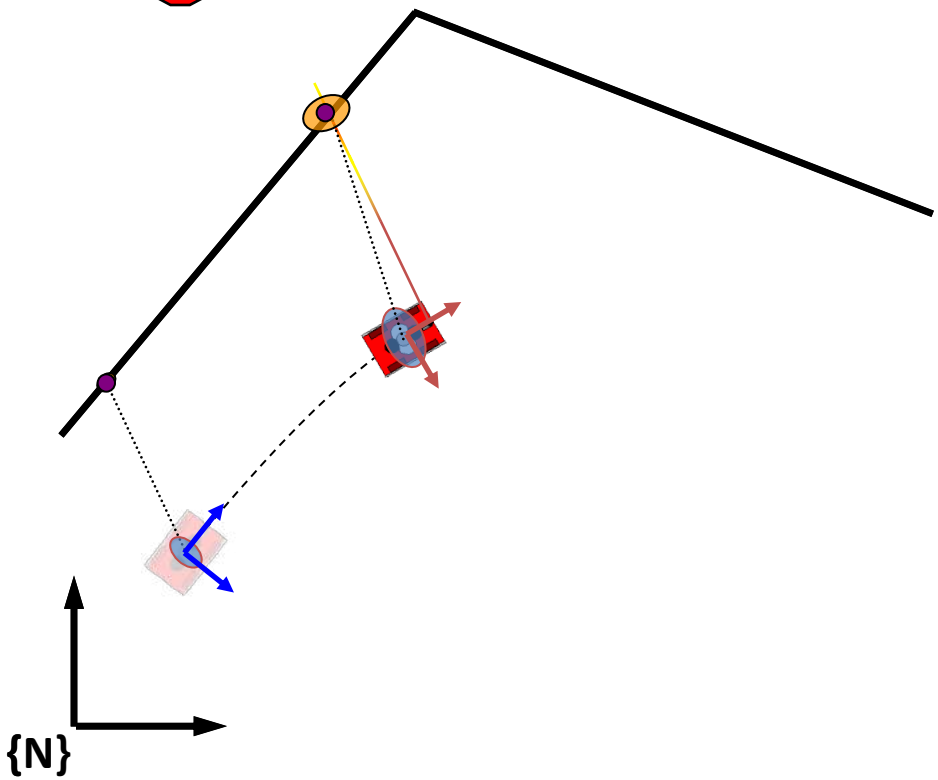
Measurements: DVL, Depth & Compass

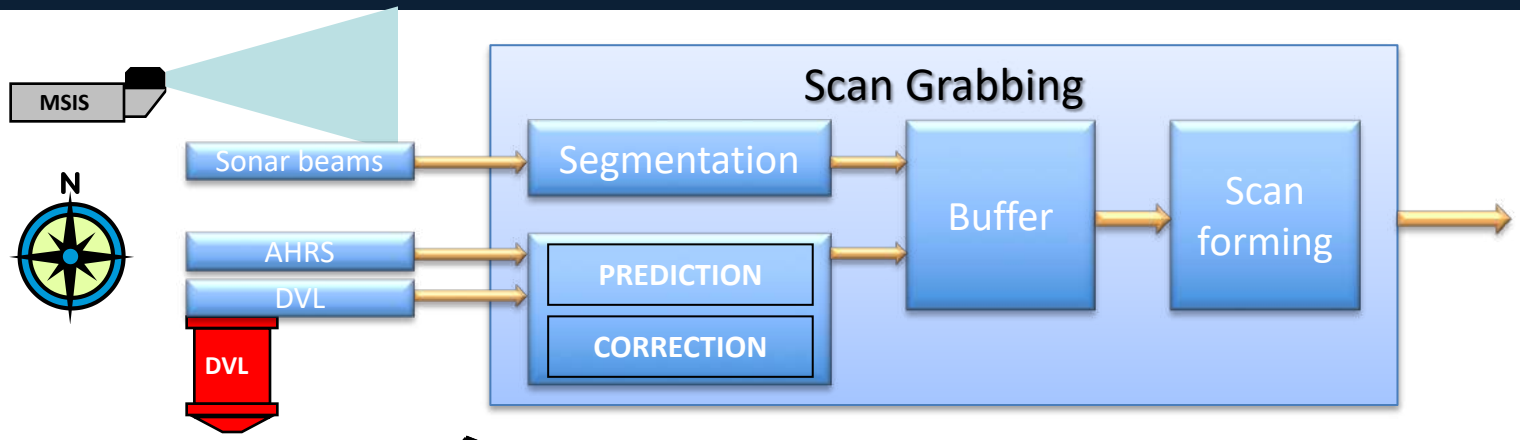
$\boldsymbol{z_{compass}} = z_\psi$ is the Yaw observation

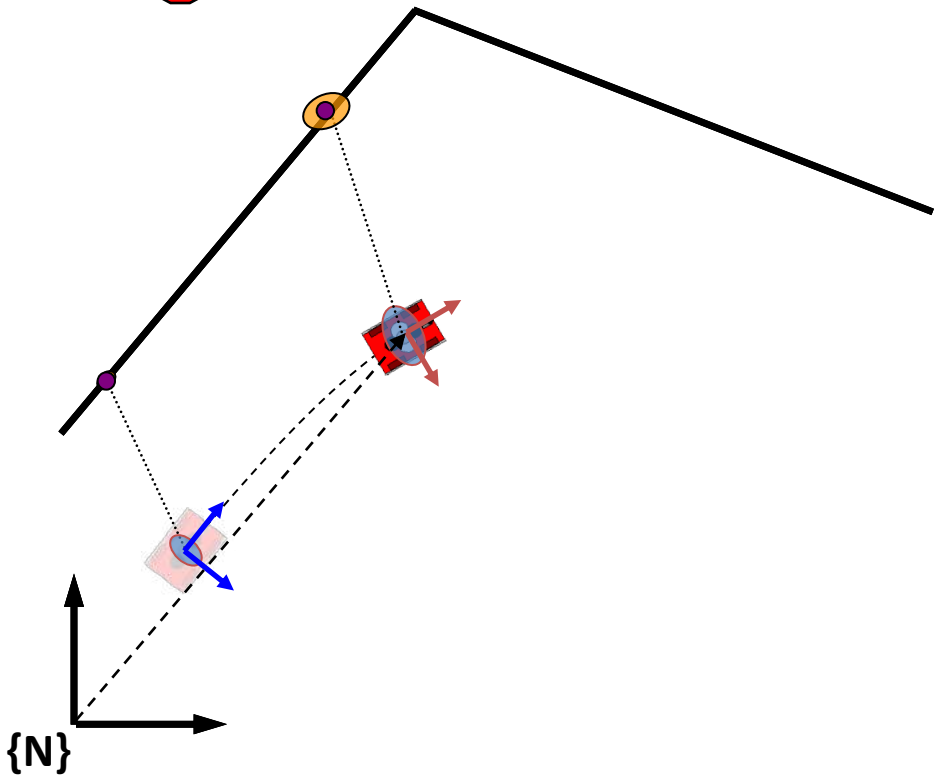
$\boldsymbol{v_{compass}} = v_\psi = \mathcal{N}(\mathbf{0}, \sigma^2_{compass})$ is the noise of the Yaw observation

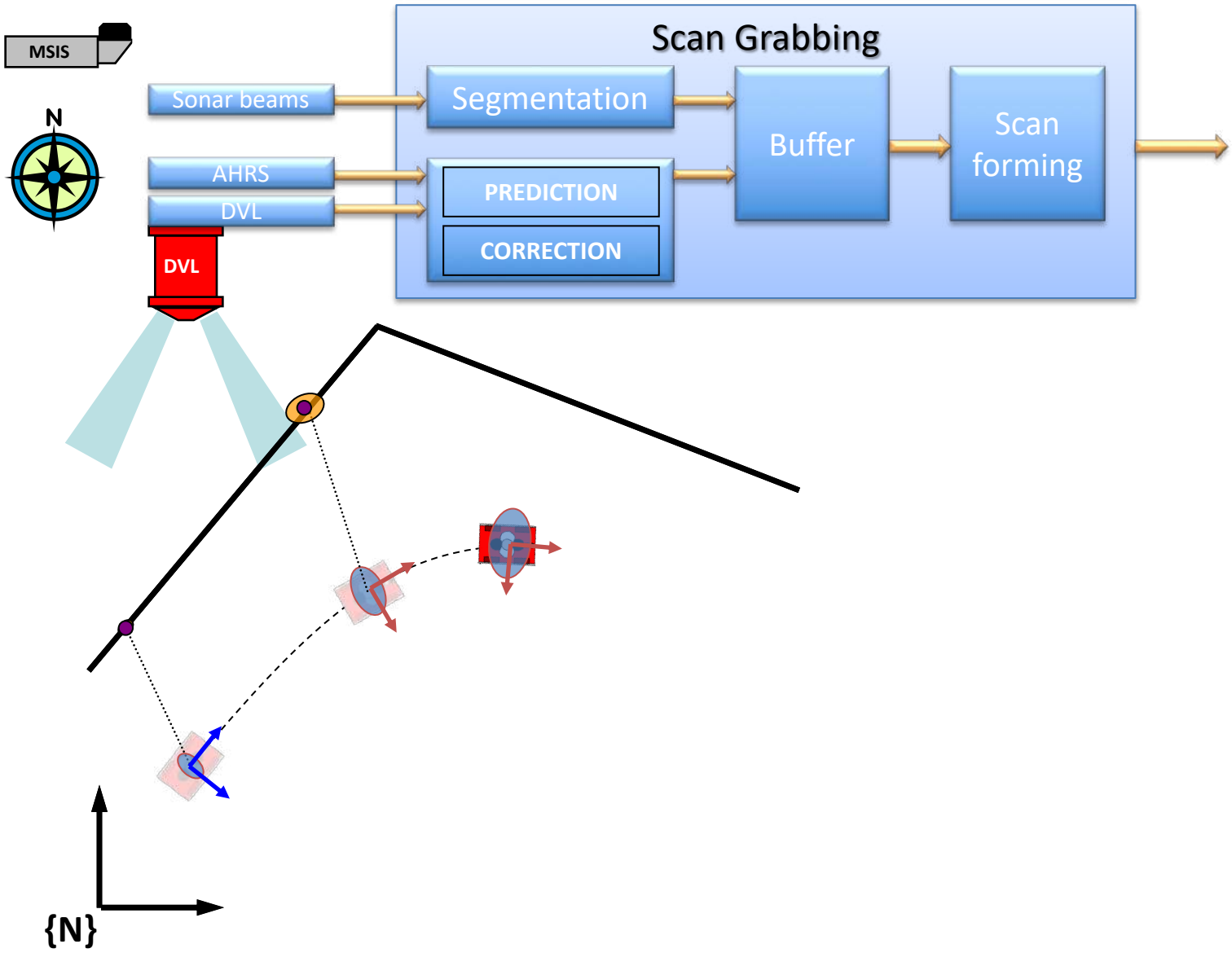
$$\boldsymbol{z_{compass}} = \boldsymbol{H_k} \cdot {}^{N}\boldsymbol{x}_{B_k} + \boldsymbol{v_{compass}} \Rightarrow z_\psi = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^{N}x_B \\ {}^{N}y_B \\ {}^{N}z_B \\ {}^{N}\psi_B \\ {}^{B}u_B \\ {}^{B}v_B \\ {}^{B}w_B \\ {}^{B}r_B \end{bmatrix} + v_\psi$$

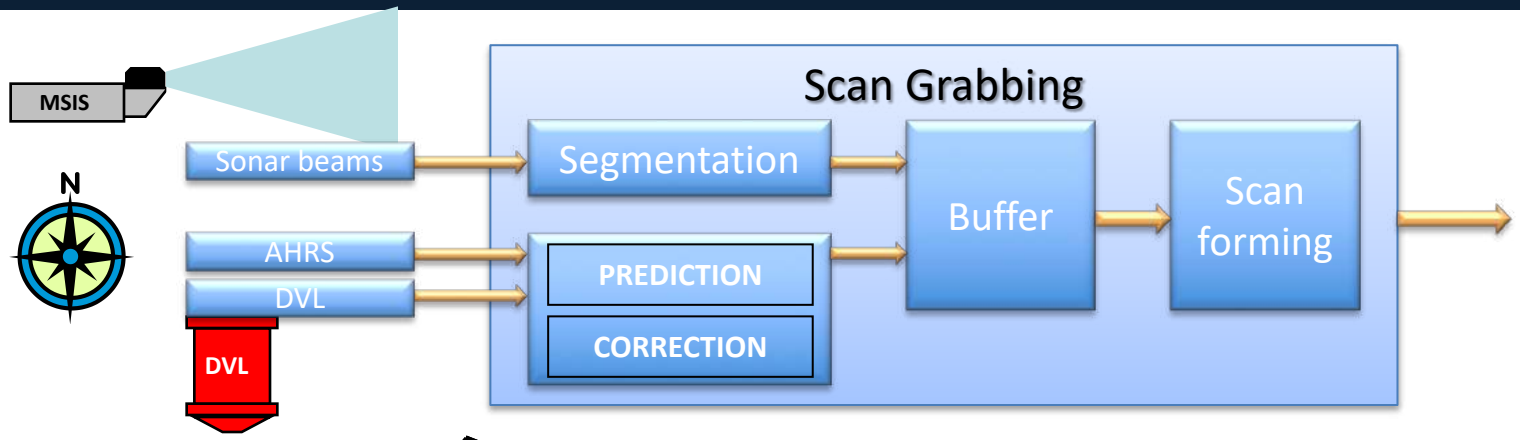

MSIS
N
Sonar beams
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming

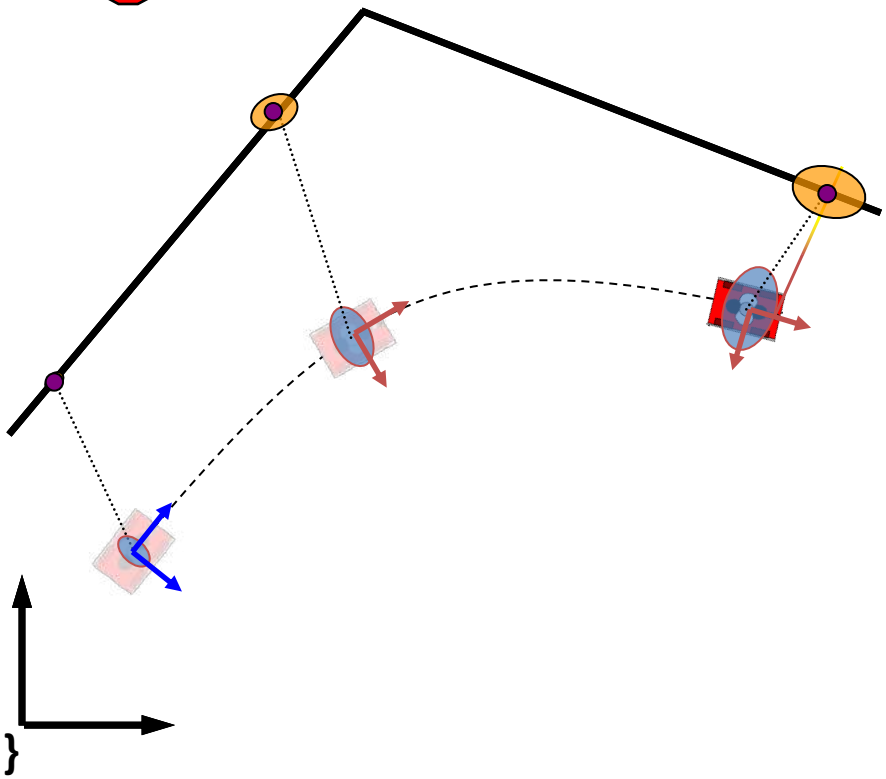

{N}

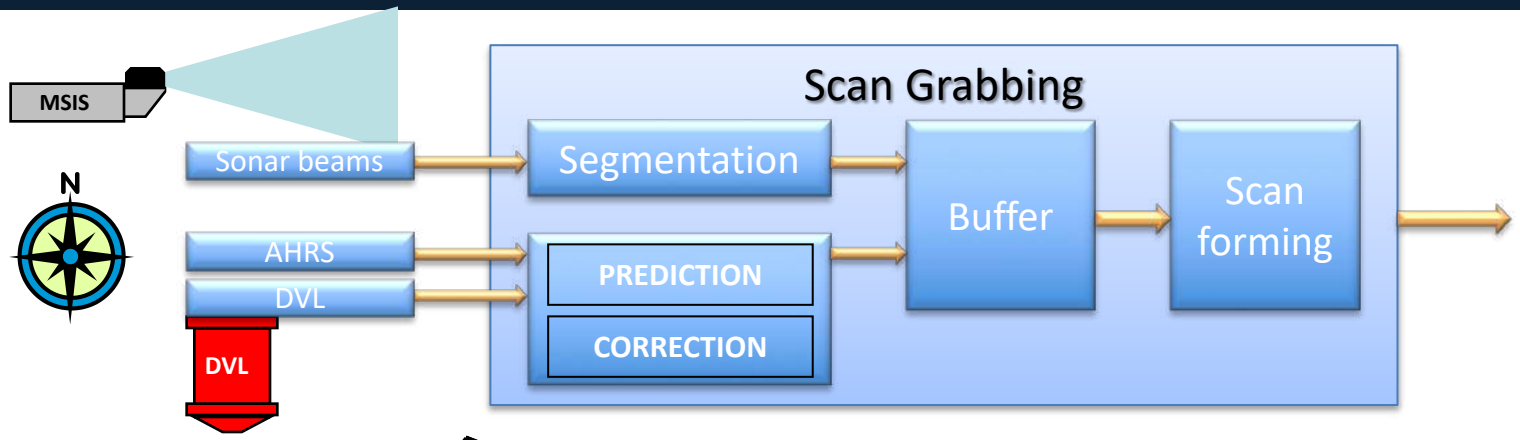

MSIS
Sonar beams
N
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming

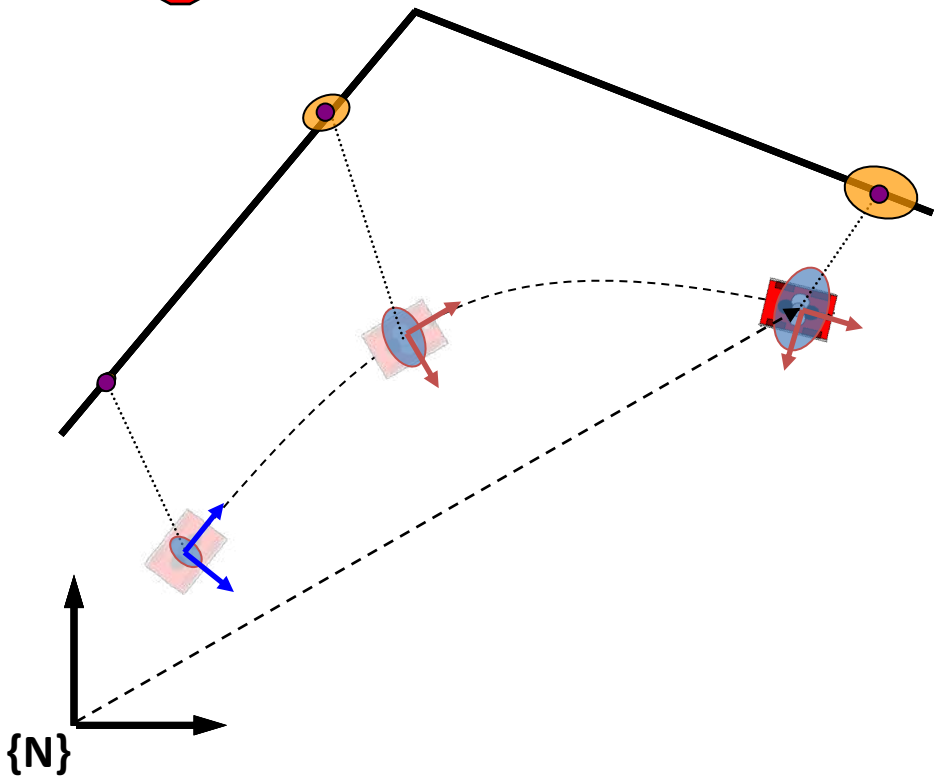

{N}

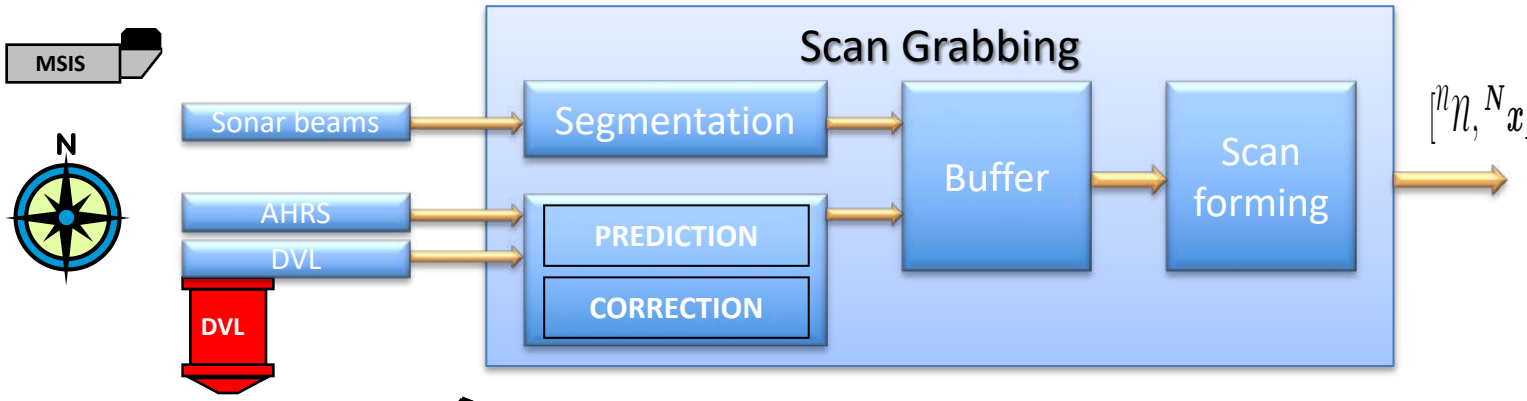

MSIS
N
Sonar beams
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming
{N}

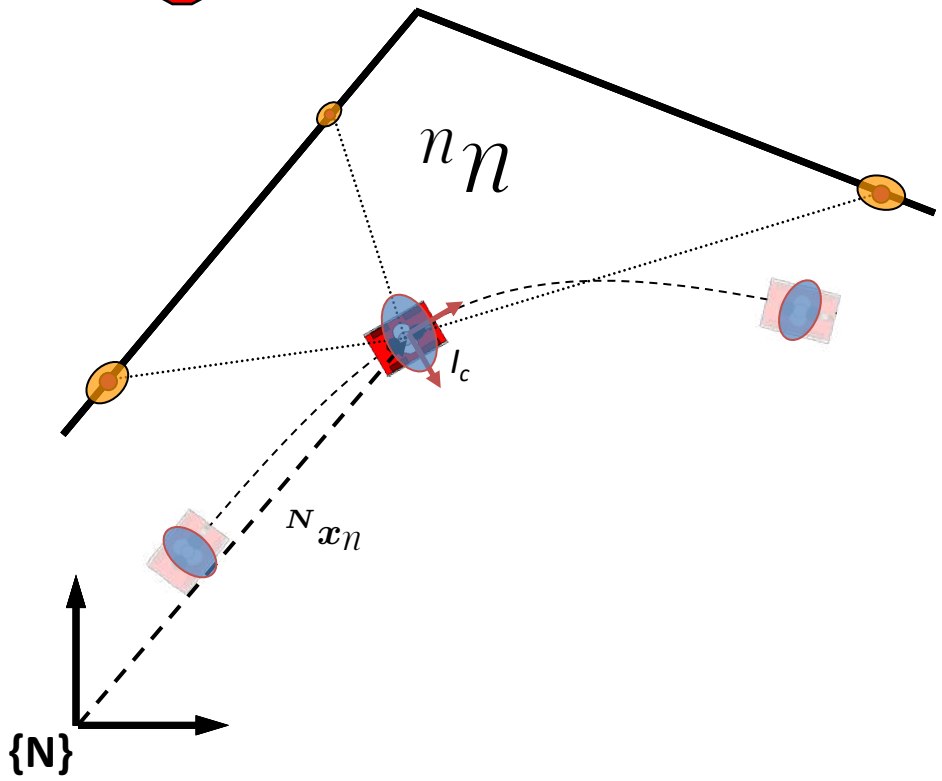

MSIS
Sonar beams
N
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming


{N}


MSIS
Sonar beams
N
AHRS
DVL
DVL
Scan Grabbing
Segmentation
PREDICTION
CORRECTION
Buffer
Scan forming

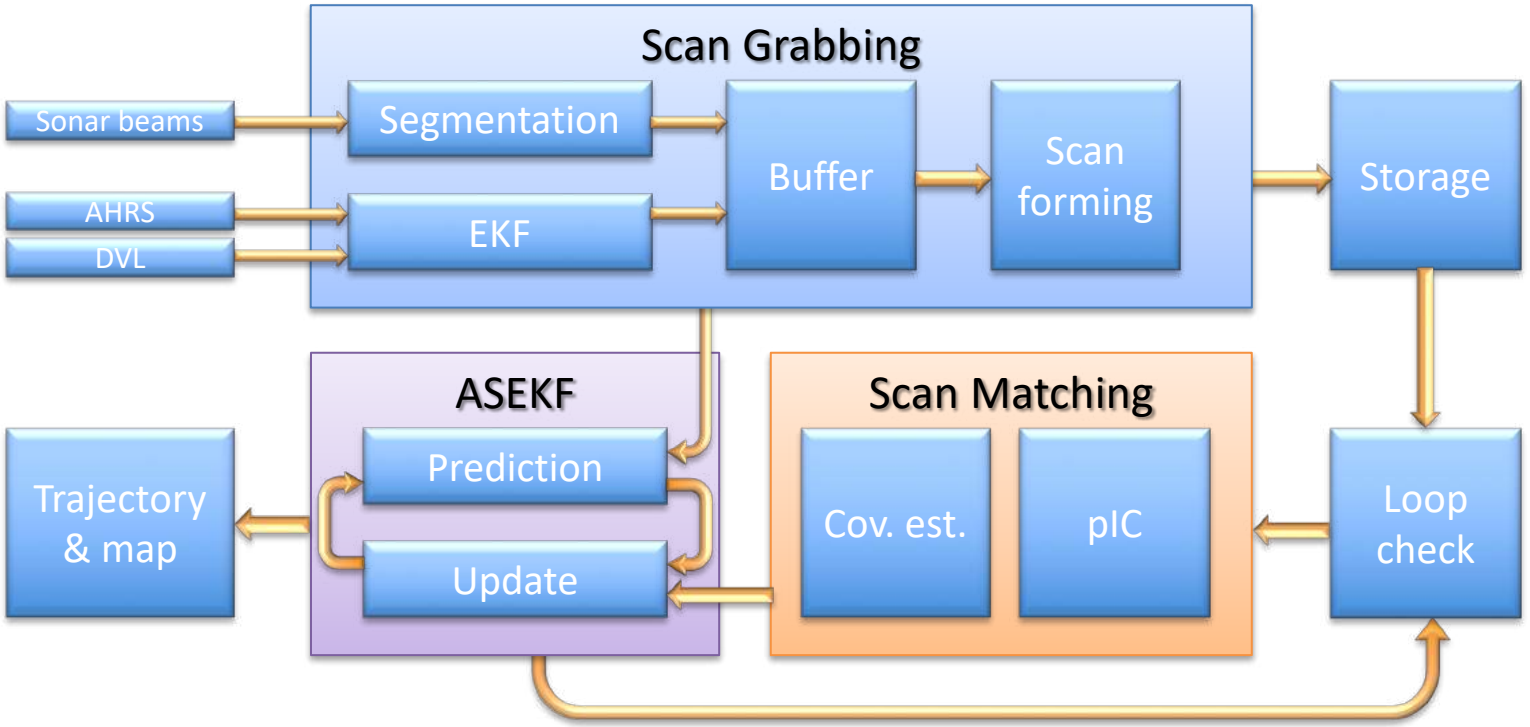

{N}



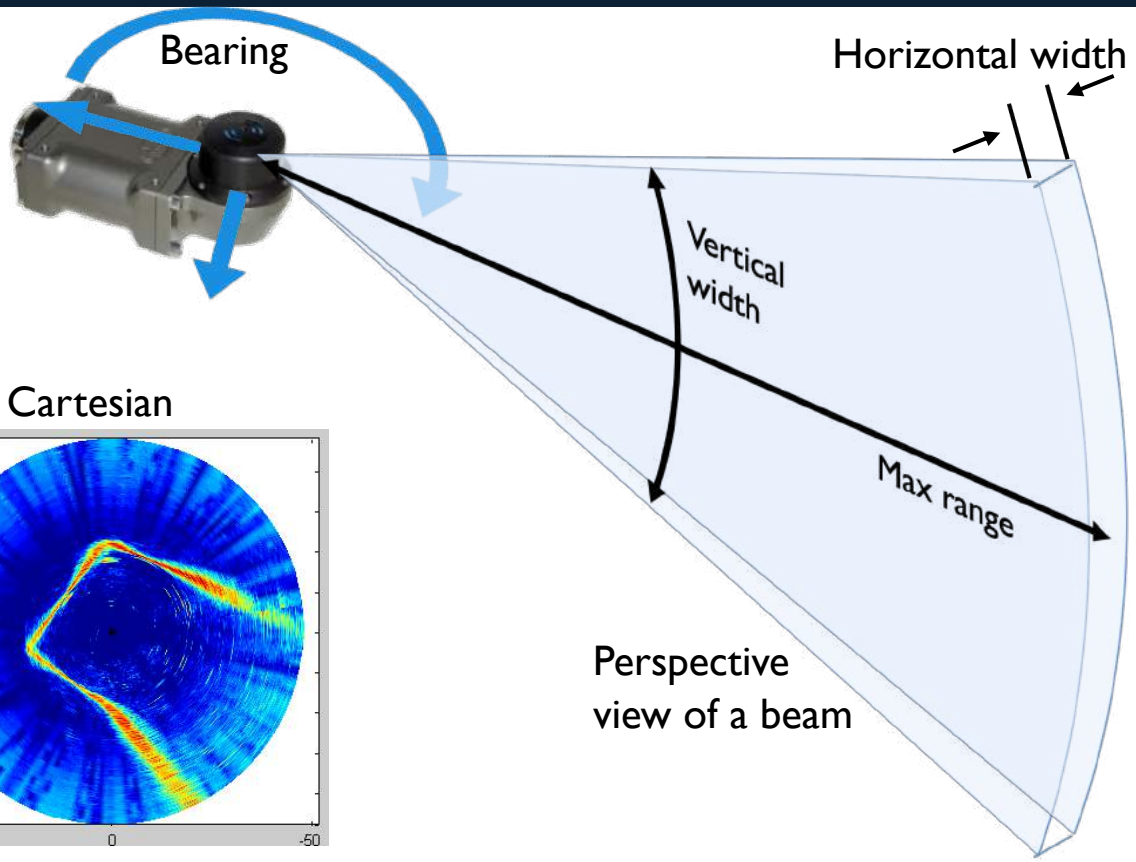


- Complete scan sector acquired with the robot displacement.
- The uncertainty of the dead-reckoning process is distributed accordingly and propagated to the scan points.
- The beams in a full scan are then referenced to the I_c frame, which corresponds to the position at the center of the trajectory [*Burguera et al, 2008*]

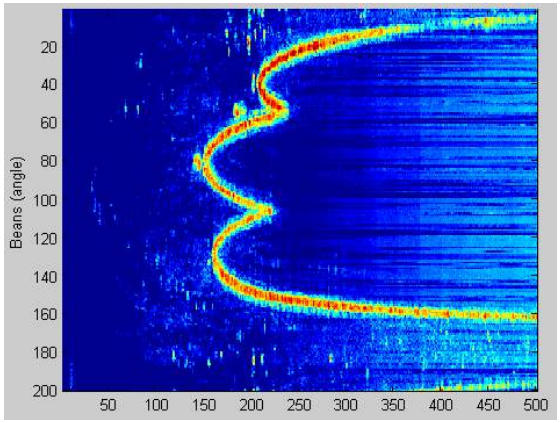
Algorithm block diagram



Working with MSIS: Sensor model



Polar



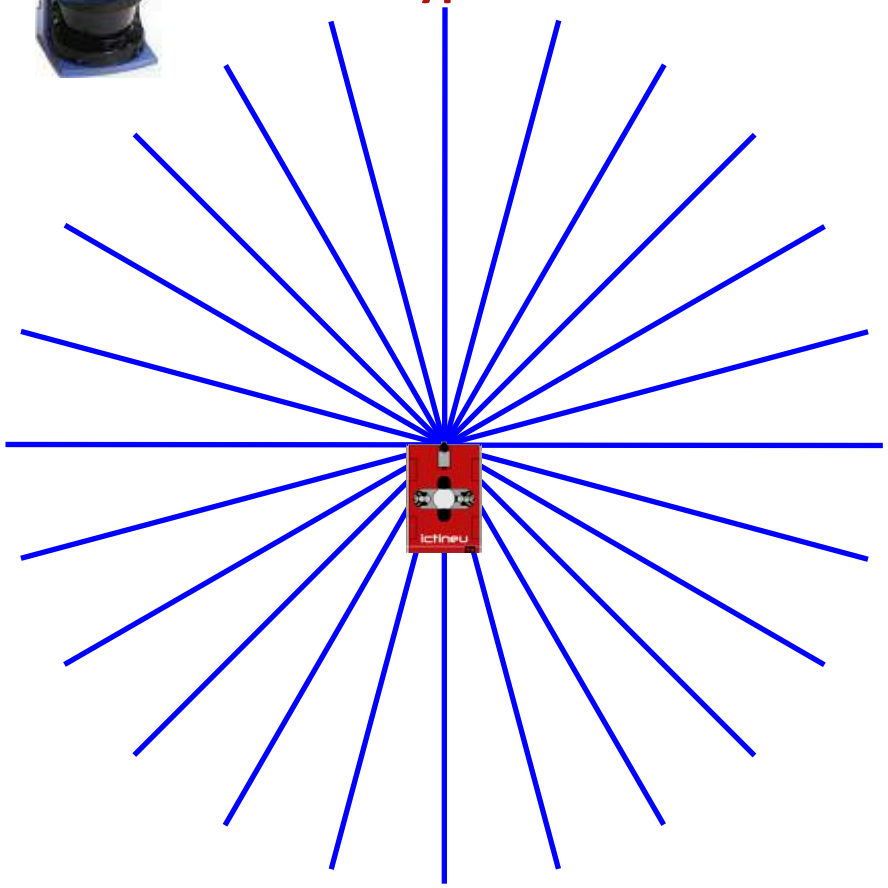
Cartesian

Tritech Miniking Imaging Sonar

- Max range: 100 m
- Configurable scan sector 0 – 360°
- Vertical width: 40°
- Horizontal width: 3°
- **Update rate: complete scan in ~14s** (50m, 360°)

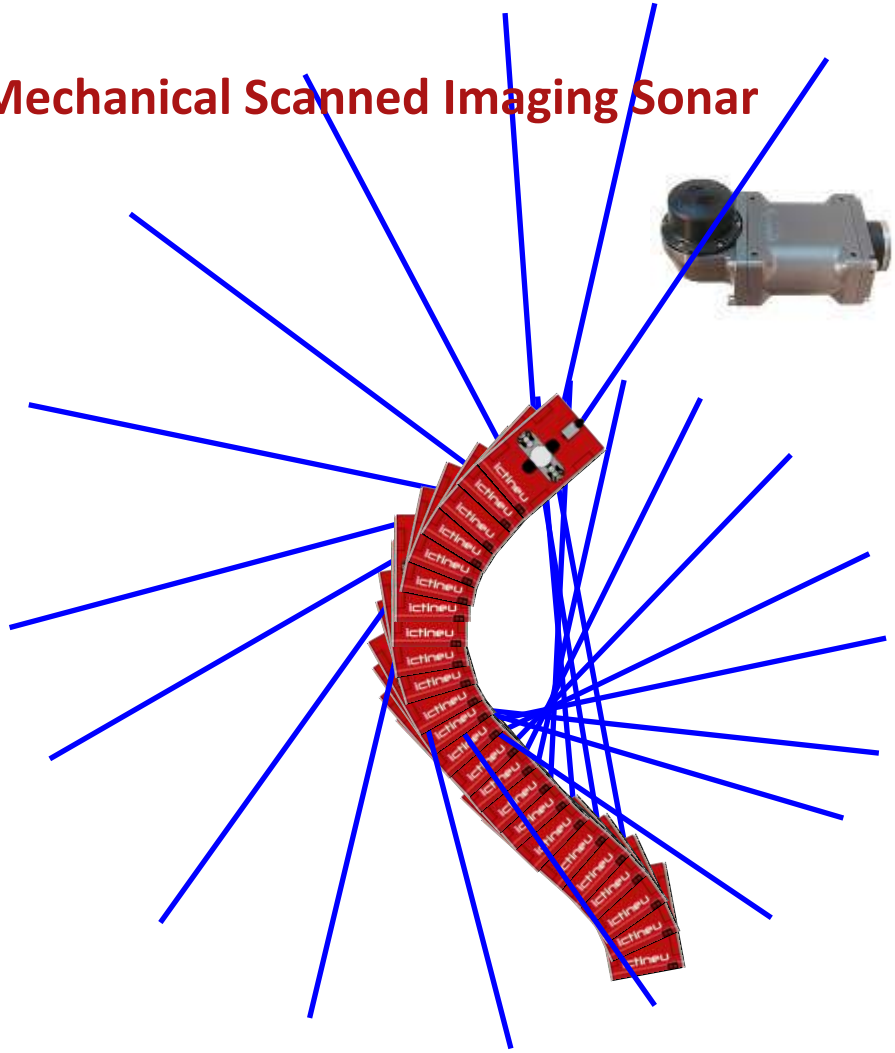
Working with MSIS: Motion distortion on acoustic data

Laser type scanner



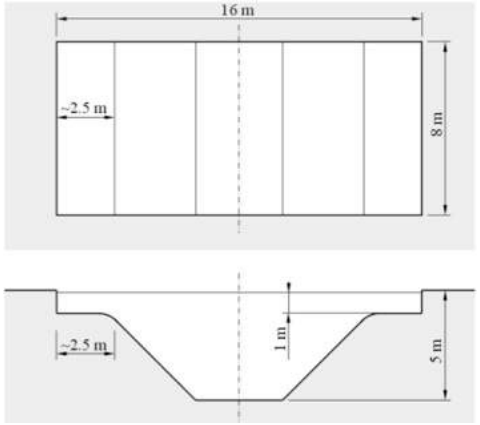
Full scan in time step *k*

Mechanical Scanned Imaging Sonar



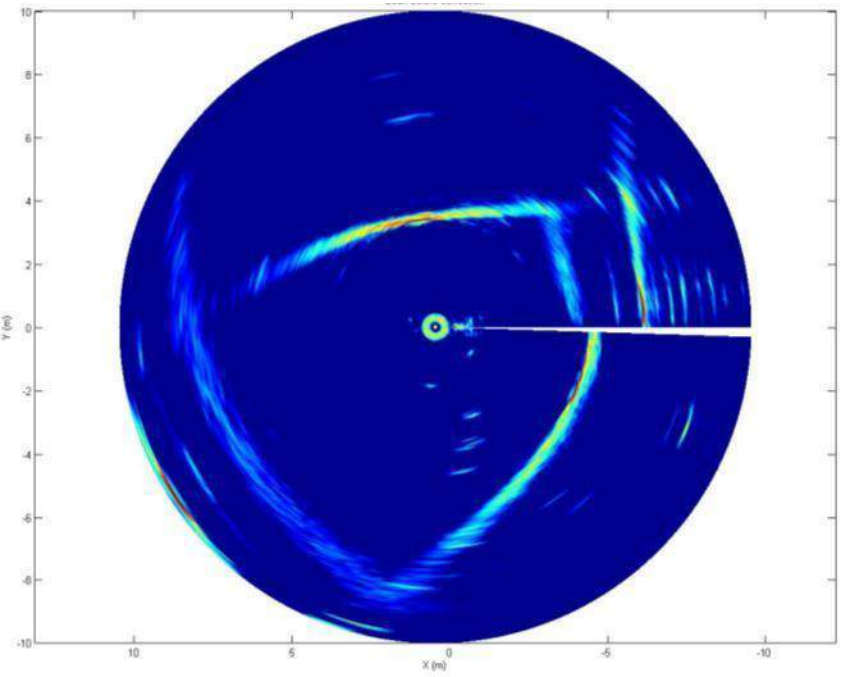
Full scan in time steps {*k* → *k+m*}

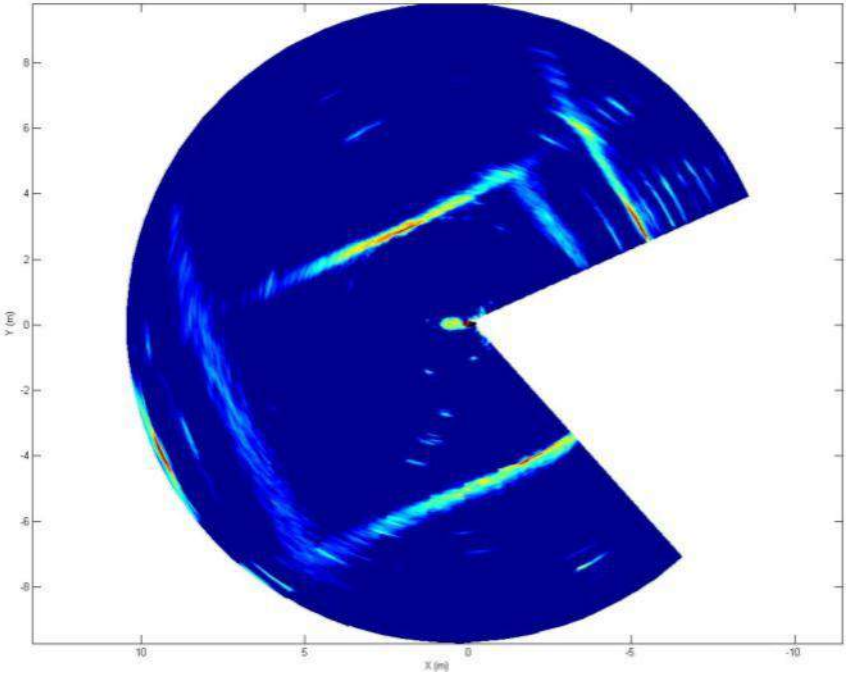
Working with MSIS: Motion distortion on acoustic data



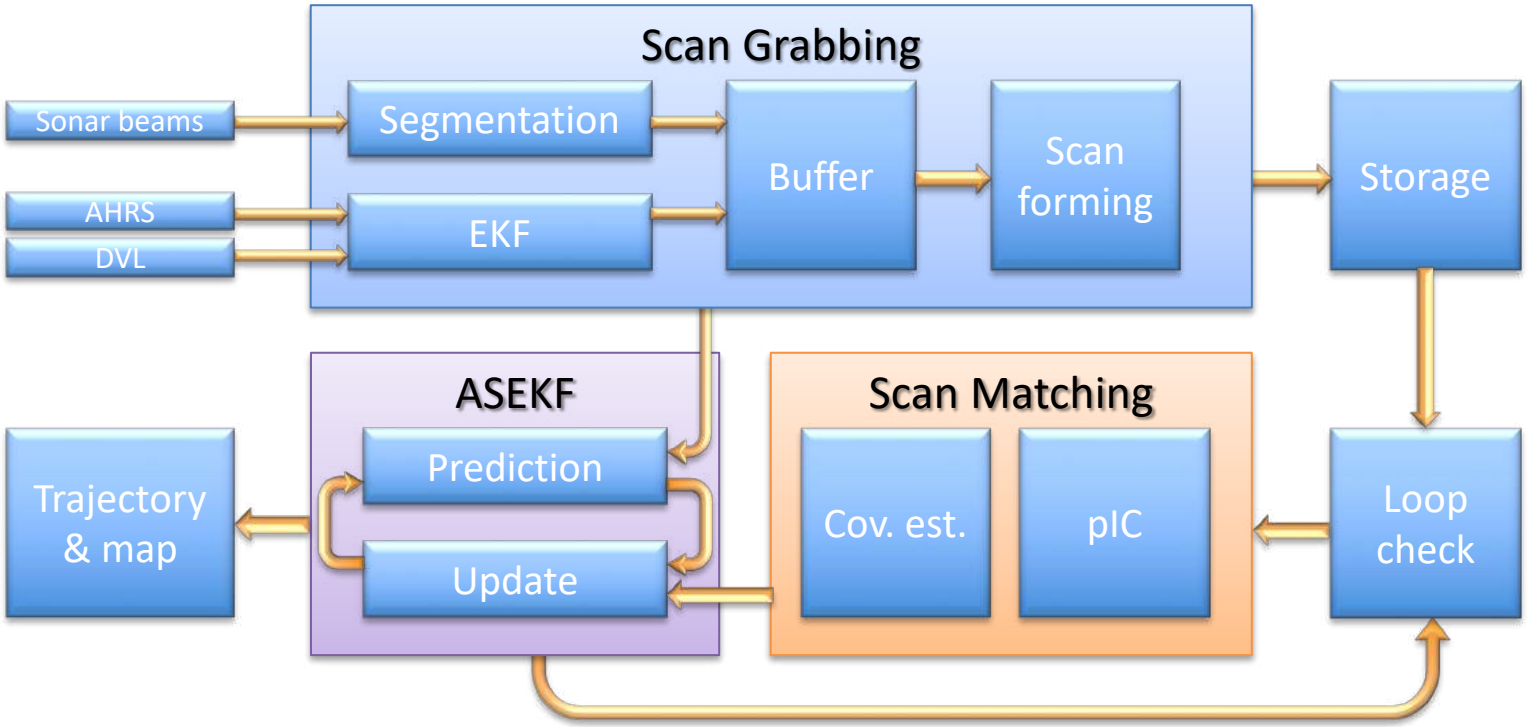
Distorted Data

Corrected Data



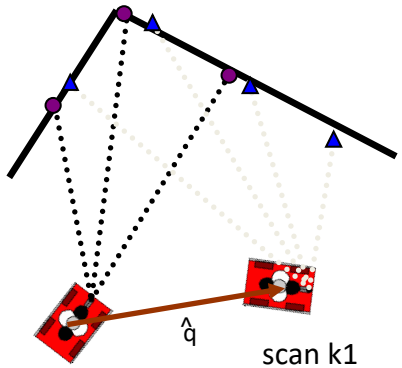


Algorithm block diagram

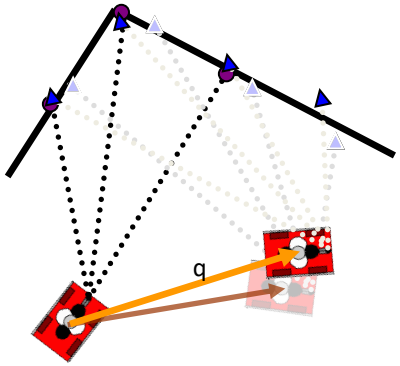


Probabilistic Registration

- Compute the relative displacement of a vehicle between two scans by maximizing the overlap between range & bearing measurements.

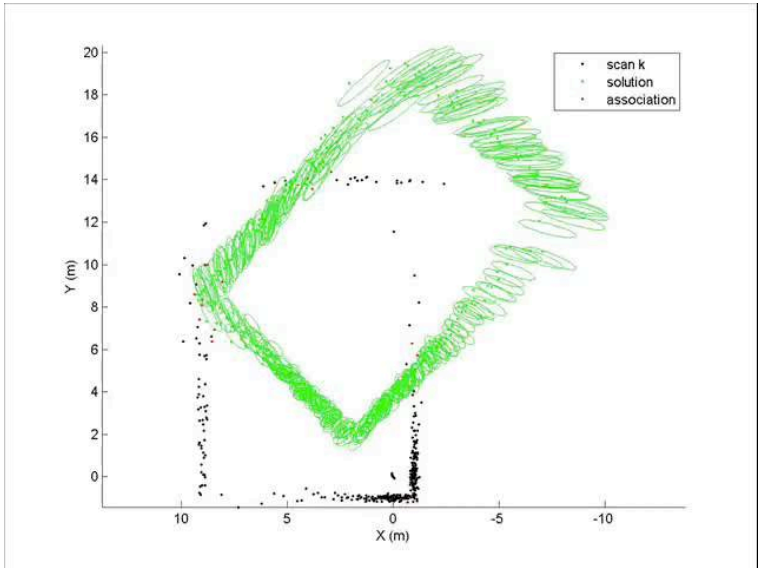


Scan Matching



do{
- Association
- Minimization

}**while** (!convergence)



Function pIC($^{\mathcal{N}}\eta, {}^{\mathcal{R}}\mathcal{R}, {}^{\mathcal{R}}\hat{\mathbf{x}}_\eta, {}^{\mathcal{R}}P_\eta$)

// New Scan: $^{\mathcal{N}}\eta = [[^{\mathcal{N}}\hat{\mathbf{n}}_1, {}^{\mathcal{N}}P_{n_1}] [^{\mathcal{N}}\hat{\mathbf{n}}_2, {}^{\mathcal{N}}P_{n_2}] \dots [^{\mathcal{N}}\hat{\mathbf{n}}_n, {}^{\mathcal{N}}P_{n_n}]]$

// Reference Scan: $^{\mathcal{R}}\mathcal{R} = [[^{\mathcal{R}}\hat{\mathbf{r}}_1, {}^{\mathcal{R}}P_{r_1}] [^{\mathcal{R}}\hat{\mathbf{r}}_2, {}^{\mathcal{R}}P_{r_2}] \dots [^{\mathcal{R}}\hat{\mathbf{r}}_i, {}^{\mathcal{R}}P_{r_i}]]$

// Initial Transformation guess: $\mathcal{N}(^{\mathcal{R}}\hat{\mathbf{x}}_\eta, {}^{\mathcal{R}}P\mathbf{x}_\eta)$

1 $i = 1$;

2 while $(i < maxIterations)$ and (not *converged*) do

3 for $j = 1$ to n do // For each point in the New Scan

4 $^{\mathcal{R}}\hat{\mathbf{n}}_j = {}^{\mathcal{R}}\hat{\mathbf{x}}_\eta \oplus {}^{\mathcal{N}}\hat{\mathbf{n}}_j$; // Transform to *R–frame*

5 $^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_\eta J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j} J_{2\oplus}^T$;

6 $\mathbf{A} = \{{}^{\mathcal{R}}\mathbf{r_k} \in {}^{\mathcal{R}}\mathcal{R} / \mathcal{D}_M^2(^{\mathcal{R}}\hat{\mathbf{r}}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{\mathbf{n}}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\}$; // Set of IC points

7 $[^{\mathcal{N}}\hat{c}_j, {}^{G}P_{c_j}] = \underset{{}^{\mathcal{R}}\mathbf{r_k} \in \mathbf{A}}{\operatorname{argmin}} \{\mathcal{D}_M^2(^{\mathcal{N}}\hat{\mathbf{r}}_k, {}^{\mathcal{N}}P_{r_k}, {}^{G}\hat{\mathbf{n}}_j, {}^{\mathcal{R}}P_{n_j})\}$; // Closest IC point

8 $^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{\mathbf{n}}_j$; // Mean matching error

9 $^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{b_j}$;

10 end

11 $[^{\mathcal{R}}\hat{\mathbf{x}}_\eta, {}^{\mathcal{R}}P\mathbf{x}_\eta] = \underset{{}^{\mathcal{R}}\mathbf{x}\mathbf{x}_\eta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{j=1}^{n} {}^{\mathcal{R}}\hat{e}_j^T {}^{\mathcal{R}}P_{e_j}^{-1} {}^{\mathcal{R}}\hat{e}_j \right\}$;

12 $i = i + 1$;

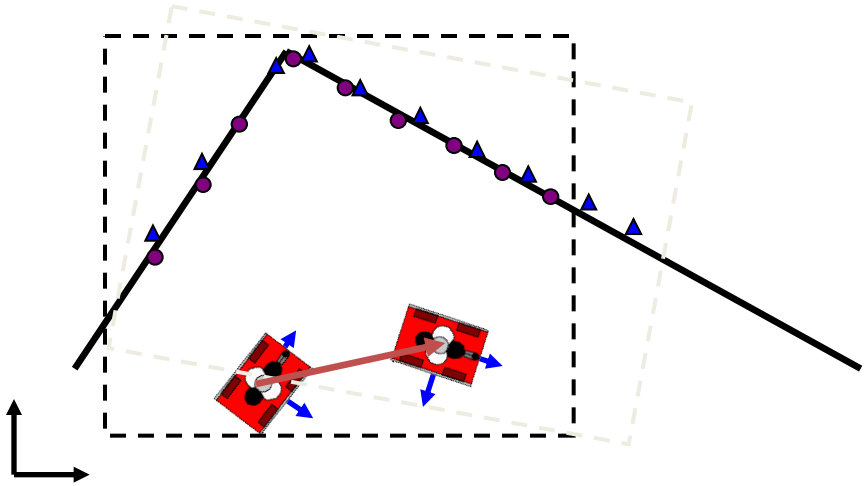
13 end

14 return $[^{\mathcal{R}}\hat{\mathbf{x}}_\eta, {}^{\mathcal{R}}P_\eta]$;

end

Algorithm 1: Probabilistic Iterative Closest Point

Input parameters



Function pIC($^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}}$)

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N], {}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N], {}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not $converged$) do

  for $j = 1$ to N do — // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ — // Transform to R–$frame$

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ — // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ — // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ — // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ — // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ — // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

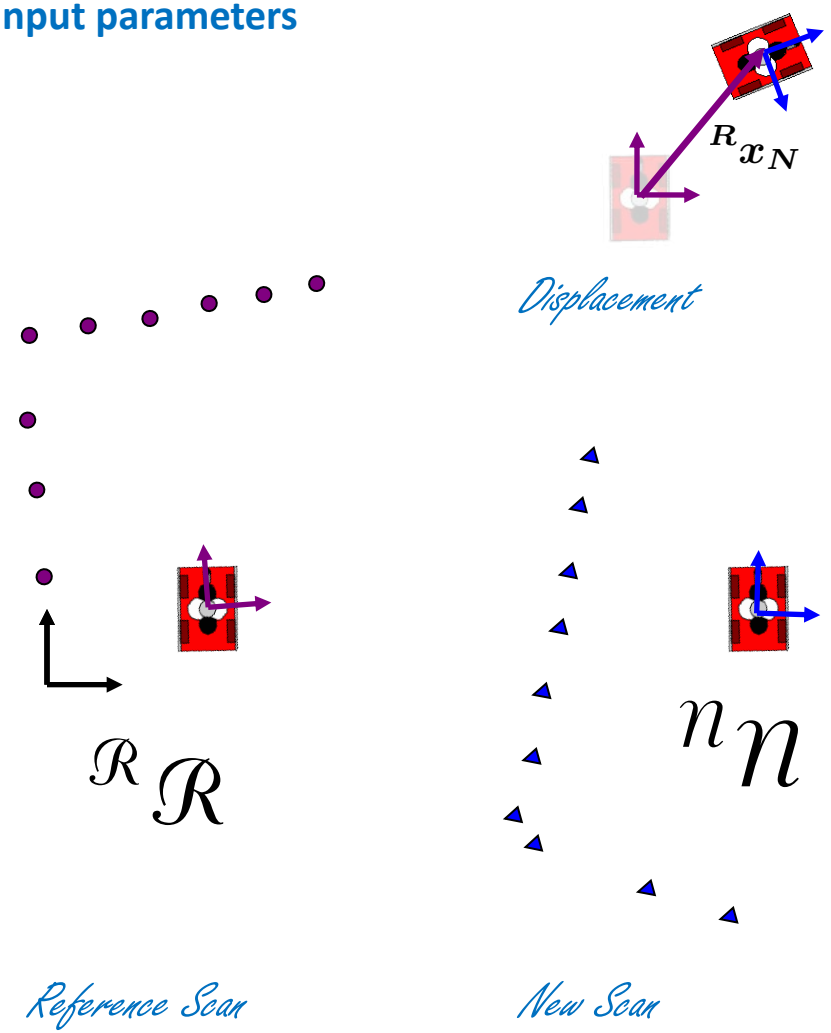
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

Input parameters



Function pIC($^{\mathcal{N}}S_n, ^{\mathcal{N}}R_n, ^{\mathcal{R}}S_r, ^{\mathcal{R}}R_r, ^{\mathcal{R}}x_{\mathcal{N}}$)

// New Scan: $^{\mathcal{N}}S_n = [^{\mathcal{N}}\hat{n}_1\ ^{\mathcal{N}}\hat{n}_2,\ldots,^{\mathcal{N}}\hat{n}_N], ^{\mathcal{N}}R_n = [^{\mathcal{N}}P_{n_1}\ ^{\mathcal{N}}P_{n_2},\ldots,^{\mathcal{N}}P_{n_N}]$

// Ref Scan: $^{\mathcal{R}}S_r = [^{\mathcal{R}}\hat{r}_1\ ^{\mathcal{R}}\hat{r}_2,\ldots,^{\mathcal{R}}\hat{r}_N], ^{\mathcal{R}}R_n = [^{\mathcal{R}}P_{r_1}\ ^{\mathcal{R}}P_{r_2},\ldots,^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[^{\mathcal{R}}\hat{x}_{\mathcal{N}}, ^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not *converged*) do

  for $j = 1$ to N do — // For each point in the New Scan

    $^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ — // Transform to *R–frame*

    $^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ — // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ — // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ — // Matching point

    $^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ — // Mean matching error

    $^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ — // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

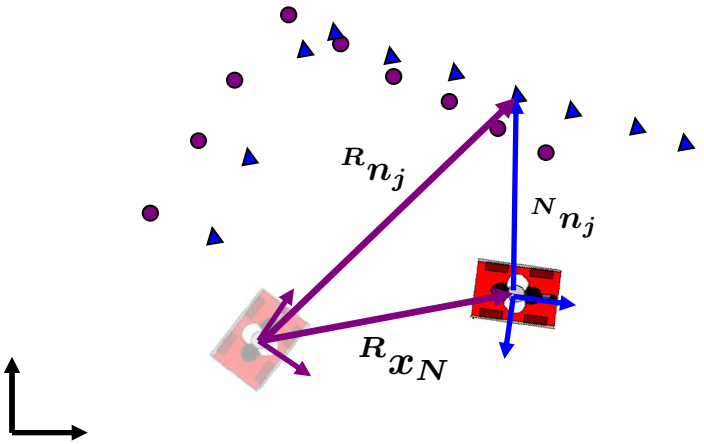
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

Reference New Scan to the R frame



Function pIC($^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}}$)

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N], {}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N], {}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not *converged*) do

  for $j = 1$ to N do — // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ — // Transform to *R–frame*

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ — // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ — // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ — // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ — // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ — // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

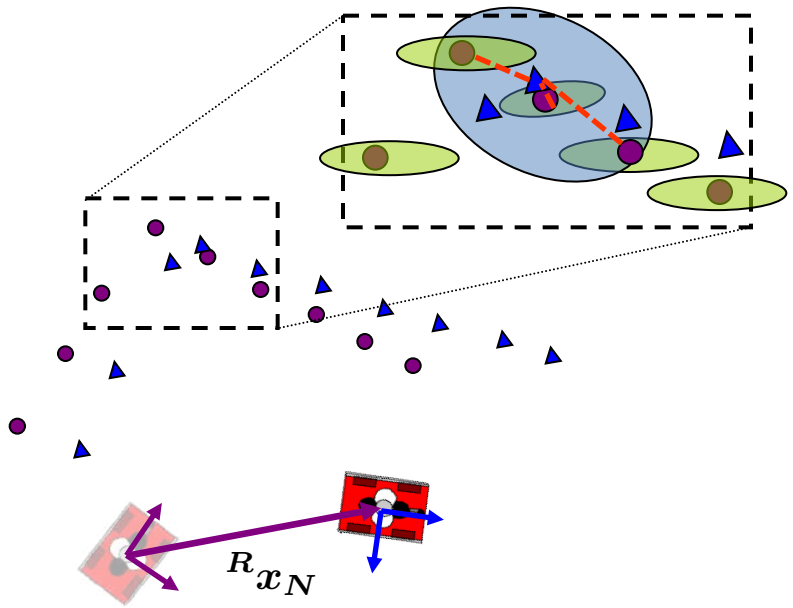
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

For each new point **Find the Set of Compatibles** reference points



Function $\texttt{pIC}({}^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}})$

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N]$, ${}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N]$, ${}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not $converged$) do

  for $j = 1$ to N do // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ // Transform to R–$frame$

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

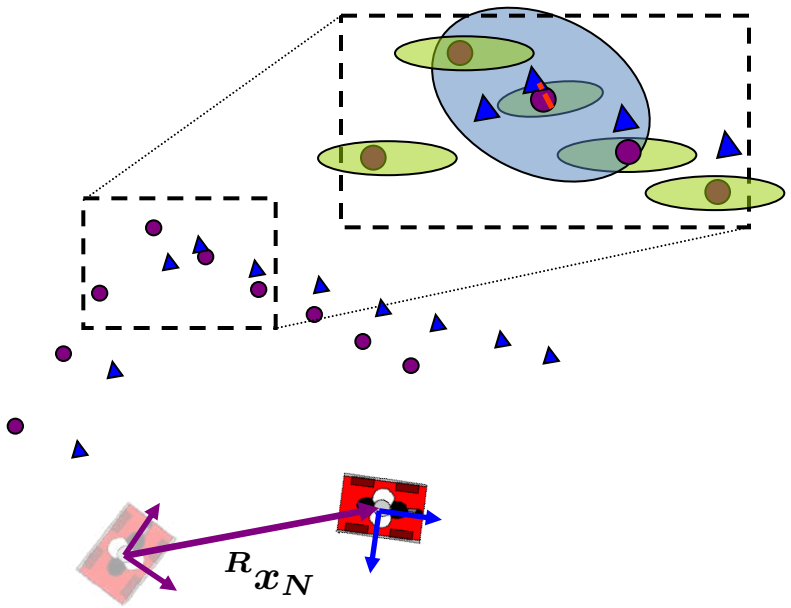
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

Select the **Nearest Neighbor** among compatible points



Function $\mathrm{pIC}({}^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}})$

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N]$, ${}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N]$, ${}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not $converged$) do

  for $j = 1$ to N do // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ // Transform to R–$frame$

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

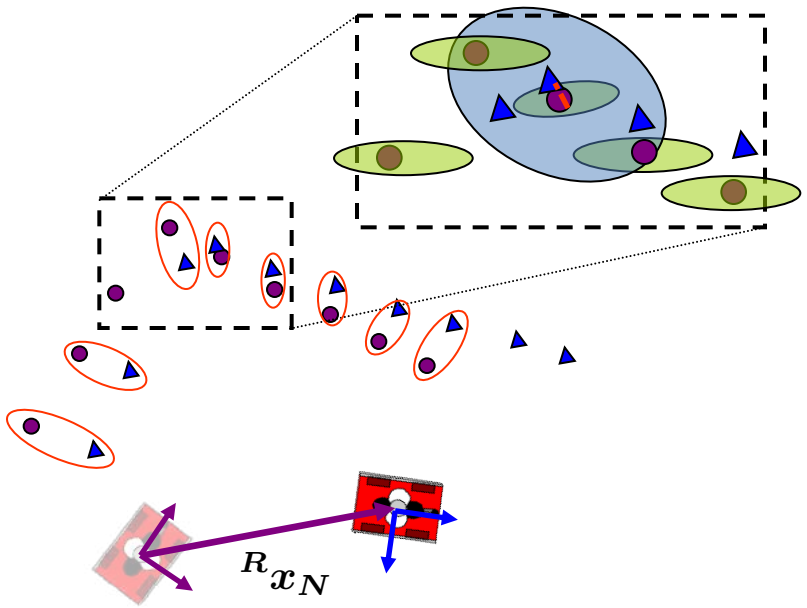
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

for all the points: **Correspondence = Nearest Neighbor**



Function $\mathrm{pIC}({}^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}})$

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N]$, ${}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N]$, ${}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not $converged$) do

  for $j = 1$ to N do // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ // Transform to R–$frame$

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

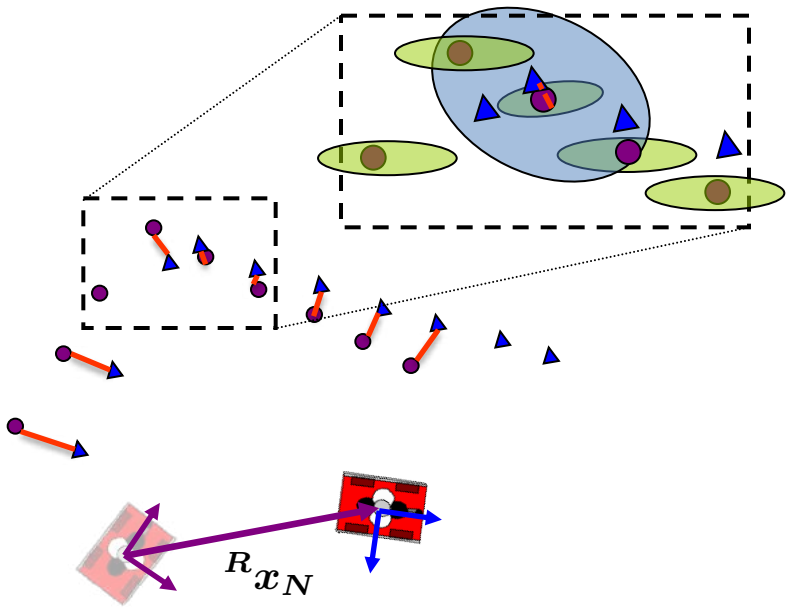
  $i = i + 1;$

end

return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

Compute the **matching error**



Function $\texttt{pIC}({}^{\mathcal{N}}S_n, {}^{\mathcal{N}}R_n, {}^{\mathcal{R}}S_r, {}^{\mathcal{R}}R_r, {}^{\mathcal{R}}x_{\mathcal{N}})$

// New Scan: ${}^{\mathcal{N}}S_n = [{}^{\mathcal{N}}\hat{n}_1\ {}^{\mathcal{N}}\hat{n}_2, \ldots, {}^{\mathcal{N}}\hat{n}_N], {}^{\mathcal{N}}R_n = [{}^{\mathcal{N}}P_{n_1}\ {}^{\mathcal{N}}P_{n_2}, \ldots, {}^{\mathcal{N}}P_{n_N}]$

// Ref Scan: ${}^{\mathcal{R}}S_r = [{}^{\mathcal{R}}\hat{r}_1\ {}^{\mathcal{R}}\hat{r}_2, \ldots, {}^{\mathcal{R}}\hat{r}_N], {}^{\mathcal{R}}R_n = [{}^{\mathcal{R}}P_{r_1}\ {}^{\mathcal{R}}P_{r_2}, \ldots, {}^{\mathcal{R}}P_{r_N}]$

// Initial Transformation guess: $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}]$

$i = 1;$

while $(i < maxIterations)$ and (not $converged$) do

  for $j = 1$ to N do // For each point in the New Scan

    ${}^{\mathcal{R}}\hat{n}_j = {}^{\mathcal{R}}\hat{x}_{\mathcal{N}} \oplus {}^{\mathcal{N}}\hat{n}_j;$ // Transform to R–$frame$

    ${}^{\mathcal{R}}P_{n_j} = J_{1\oplus}\,{}^{\mathcal{R}}P_{\mathcal{N}}\,J_{1\oplus}^T + J_{2\oplus}\,{}^{\mathcal{N}}P_{n_j}\,J_{2\oplus}^T;$ // Cov of the matching error

    $A = \{{}^{\mathcal{R}}r_k \in {}^{\mathcal{R}}S_r / \mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j}) \le \chi_\alpha^2\};$ // Set of IC points

    $[{}^{\mathcal{R}}\hat{c}_j, {}^{\mathcal{R}}P_{c_j}] = \underset{{}^{\mathcal{R}}r_k \in A}{\operatorname{argmin}}\{\mathcal{D}_M^2({}^{\mathcal{R}}\hat{r}_k, {}^{\mathcal{R}}P_{r_k}, {}^{\mathcal{R}}\hat{n}_j, {}^{\mathcal{R}}P_{n_j})\};$ // Matching point

    ${}^{\mathcal{R}}\hat{e}_j = {}^{\mathcal{R}}\hat{c}_j - {}^{\mathcal{R}}\hat{n}_j;$ // Mean matching error

    ${}^{\mathcal{R}}P_{e_j} = {}^{\mathcal{R}}P_{c_j} + {}^{\mathcal{R}}P_{n_j};;$ // Cov of the matching error

  end

  $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}] = {}^{\mathcal{R}}x_{\mathcal{N}}\operatorname{argmin}\{\frac{1}{2}\sum_{j=1}^{\mathcal{N}} {}^{\mathcal{R}}\hat{e}_j{}^T \cdot {}^{\mathcal{R}}P_{e_j}{}^{-1} \cdot {}^{\mathcal{R}}\hat{e}_j\};$

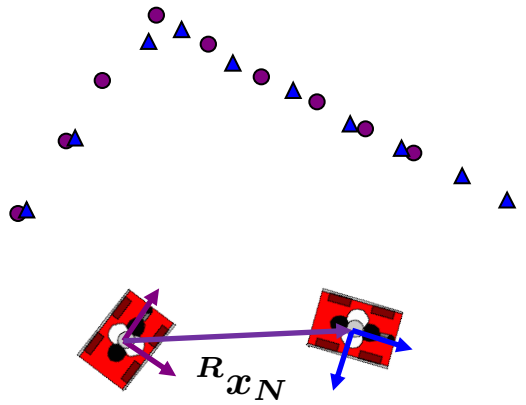
  $i = i + 1;$

end

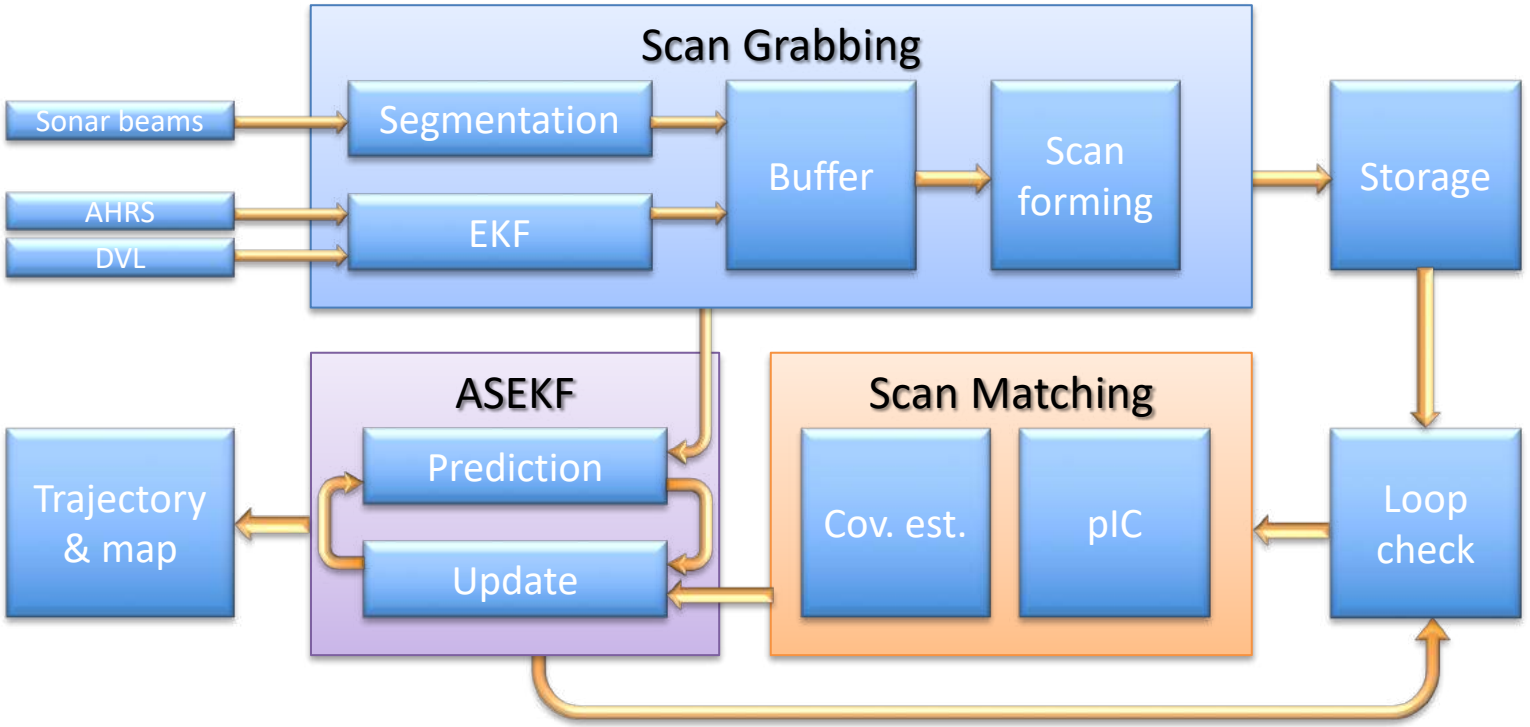
return $[{}^{\mathcal{R}}\hat{x}_{\mathcal{N}}, {}^{\mathcal{R}}P_{\mathcal{N}}];$

end

Choose the Displacement to **minimizing the LS** of the **Mahalanobis Distance** of the matching error



Algorithm block diagram



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* do

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* then

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ do // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2_\oplus} {}^NP_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

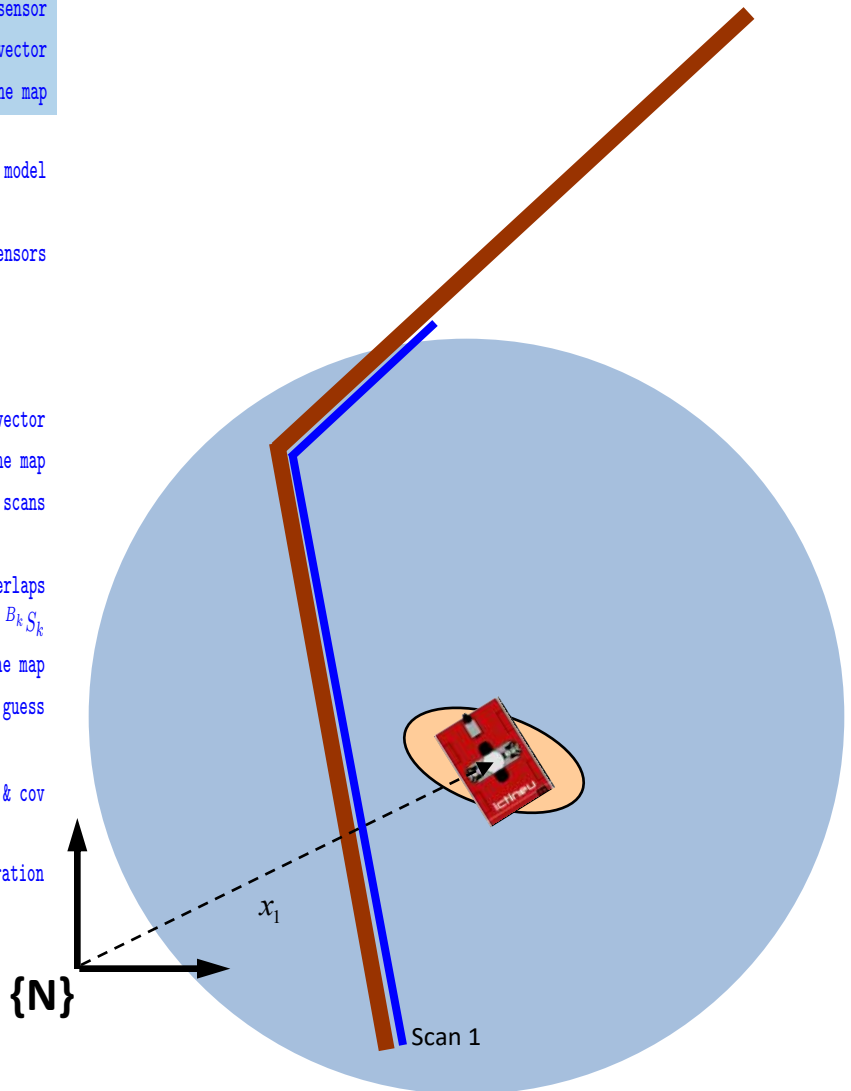
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ then

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

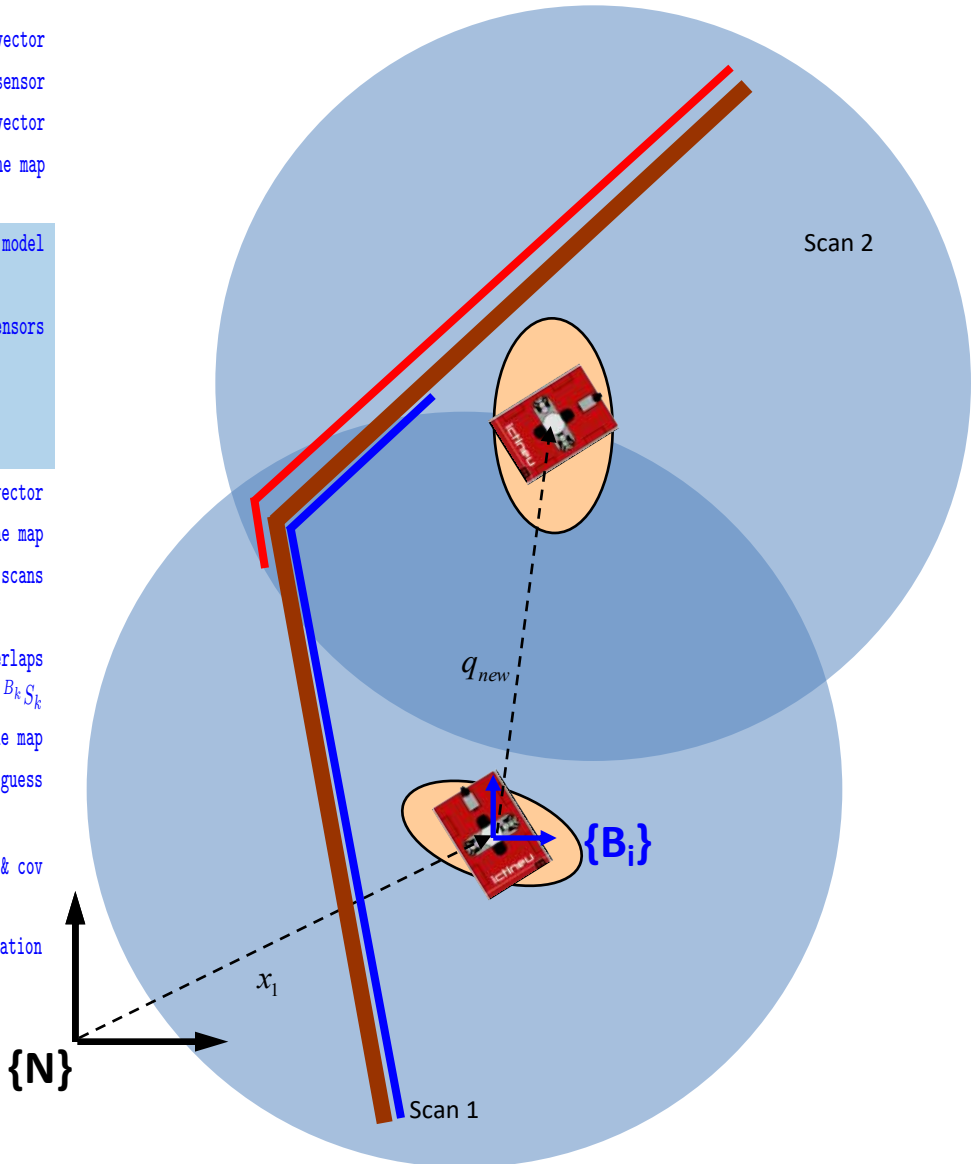
$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* do

- $[u_k, Q_k] = GetInput()$; // Get input to the motion model
- $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = Prediction({}^N\hat{x}_{k-1}, {}^NP_{k-1}, u_k, Q_k)$;
- $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors
- $z_p = [\,]; z_p = [\,]; \mathcal{H}_p = [\,]$;
- if *ScanAvailable* then
  - $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;
  - $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector
  - $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map
  - $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
  - $c = 1; \mathcal{H}_p = [\,]$;
  - for $i = 1$ *to* $length(\mathcal{H}_o)$ do // for all overlaps
    - $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
    - $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map
    - ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess
    - ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{2_\oplus}{}^NP_{B_k}J_{2\oplus}^T$
    - $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
    - if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ then // Accepted registration
      - $z_p[c] = z_r$; $R_p[c] = R_r$;
      - $\mathcal{H}_p[c] = i; c = c + 1$;
- $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;
- $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N\bar{P}_0] = AddNewPose({}^N\hat{x}_0, {}^NP_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* do

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* then

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ do // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{2_\oplus}{}^NP_{B_k}J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

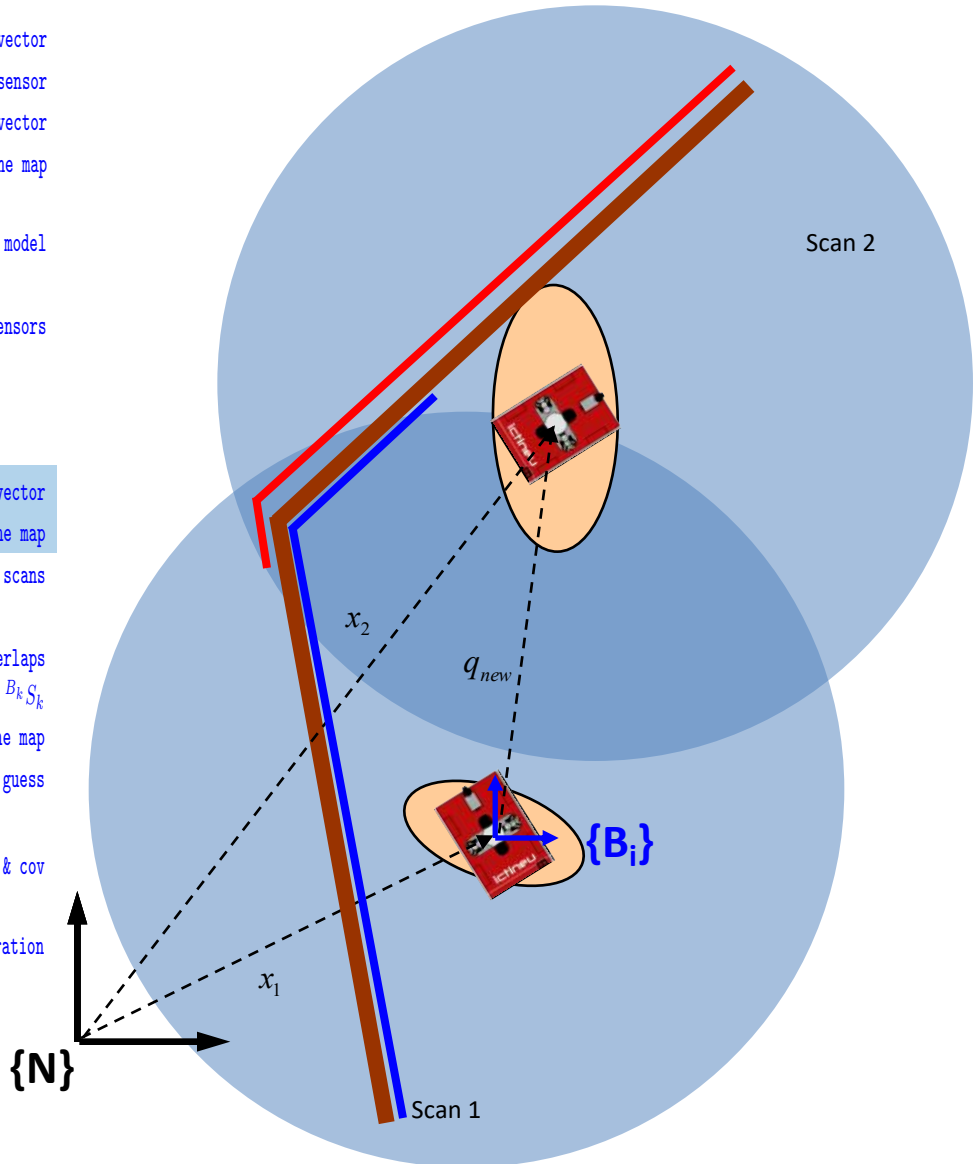
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ then

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N\bar{P}_0] = AddNewPose({}^N\hat{x}_0, {}^NP_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2_\oplus} {}^NP_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

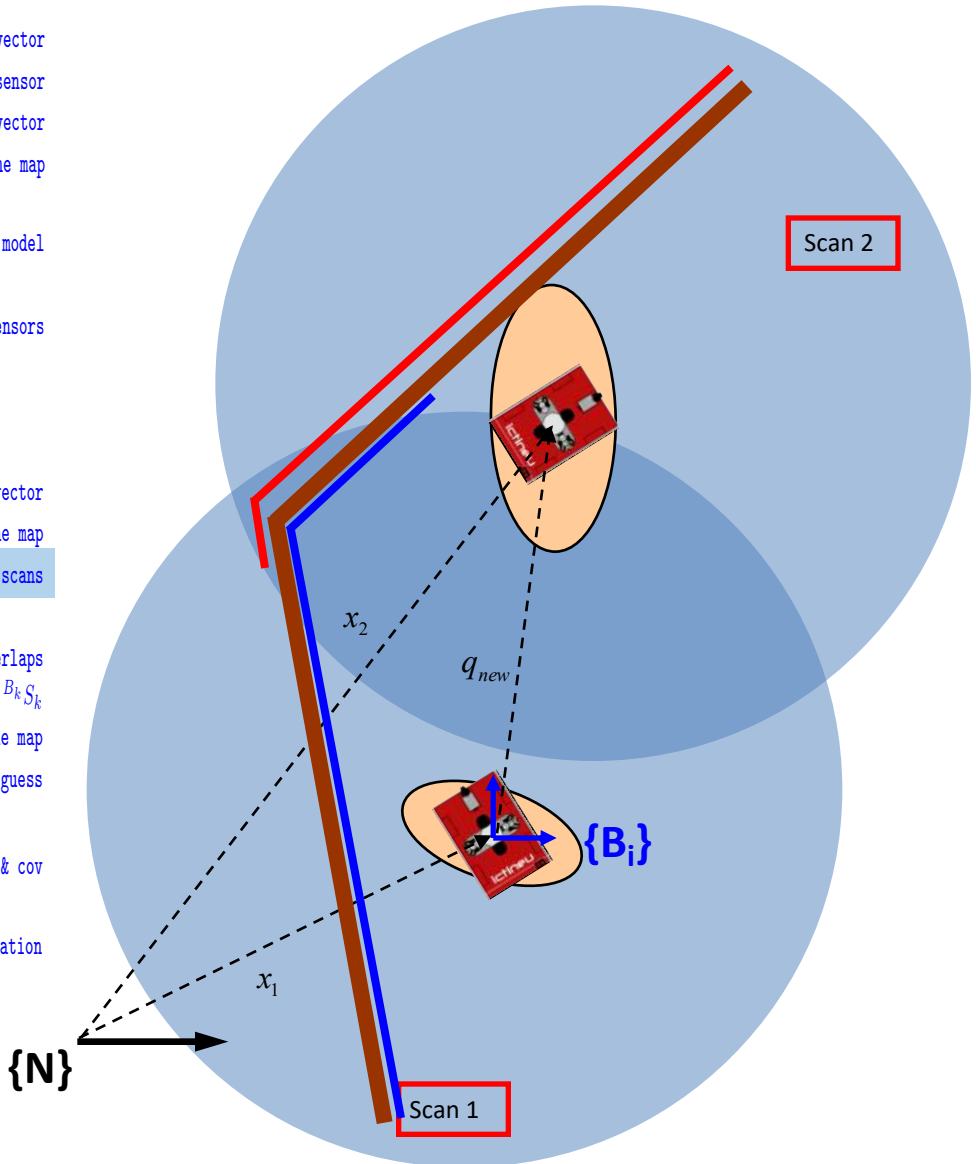
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N\bar{P}_0] = AddNewPose({}^N\hat{x}_0, {}^NP_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

$\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

$c = 1; \mathcal{H}_p = [\,]$;

for $i = 1$ *to* $length(\mathcal{H}_o)$ do // for all overlaps

$j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

$[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2_\oplus} {}^NP_{B_k} J_{2\oplus}^T$

$[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

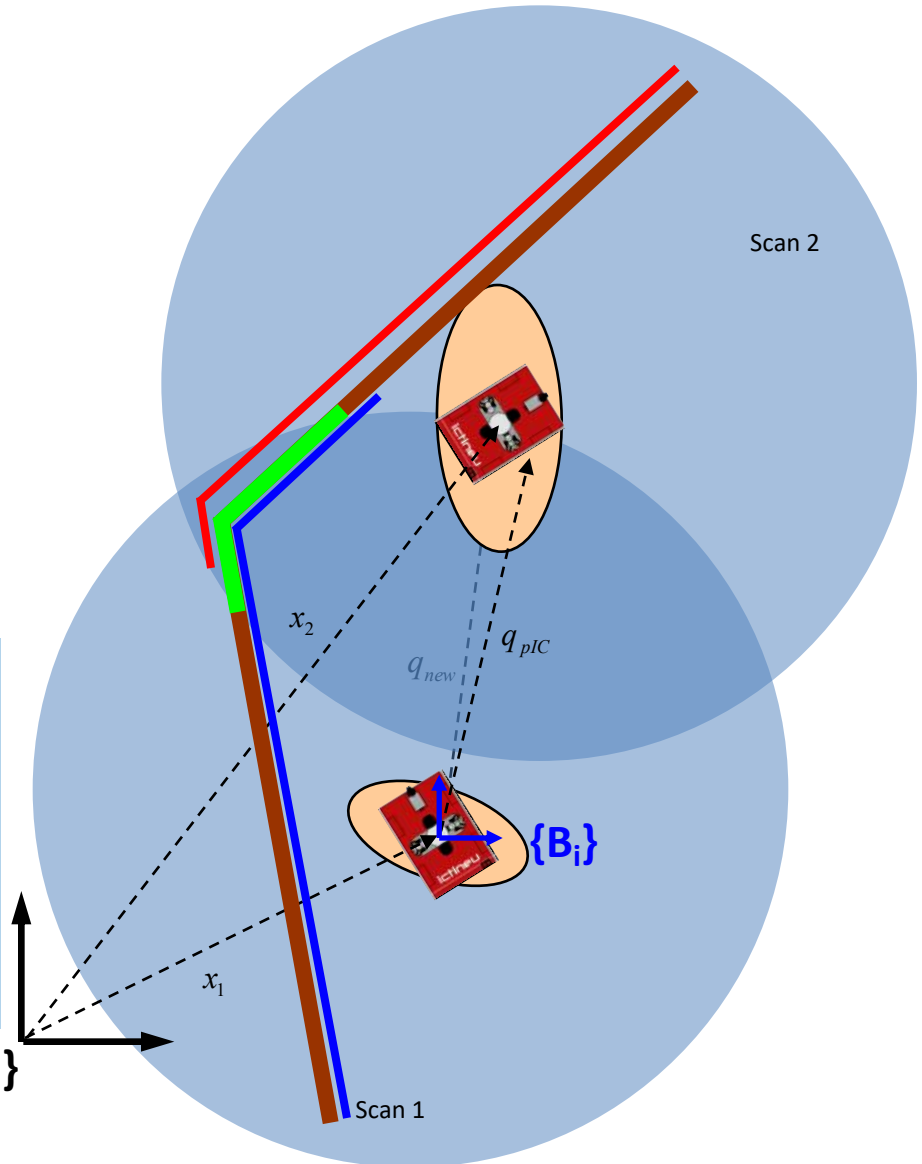
if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ then

$z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

$\mathcal{H}_p[c] = i; c = c + 1$;

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



Method $\texttt{Localization}(\hat{x}_0, P_0)$

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N\bar{P}_0] = AddNewPose({}^N\hat{x}_0, {}^NP_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2_\oplus} {}^NP_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

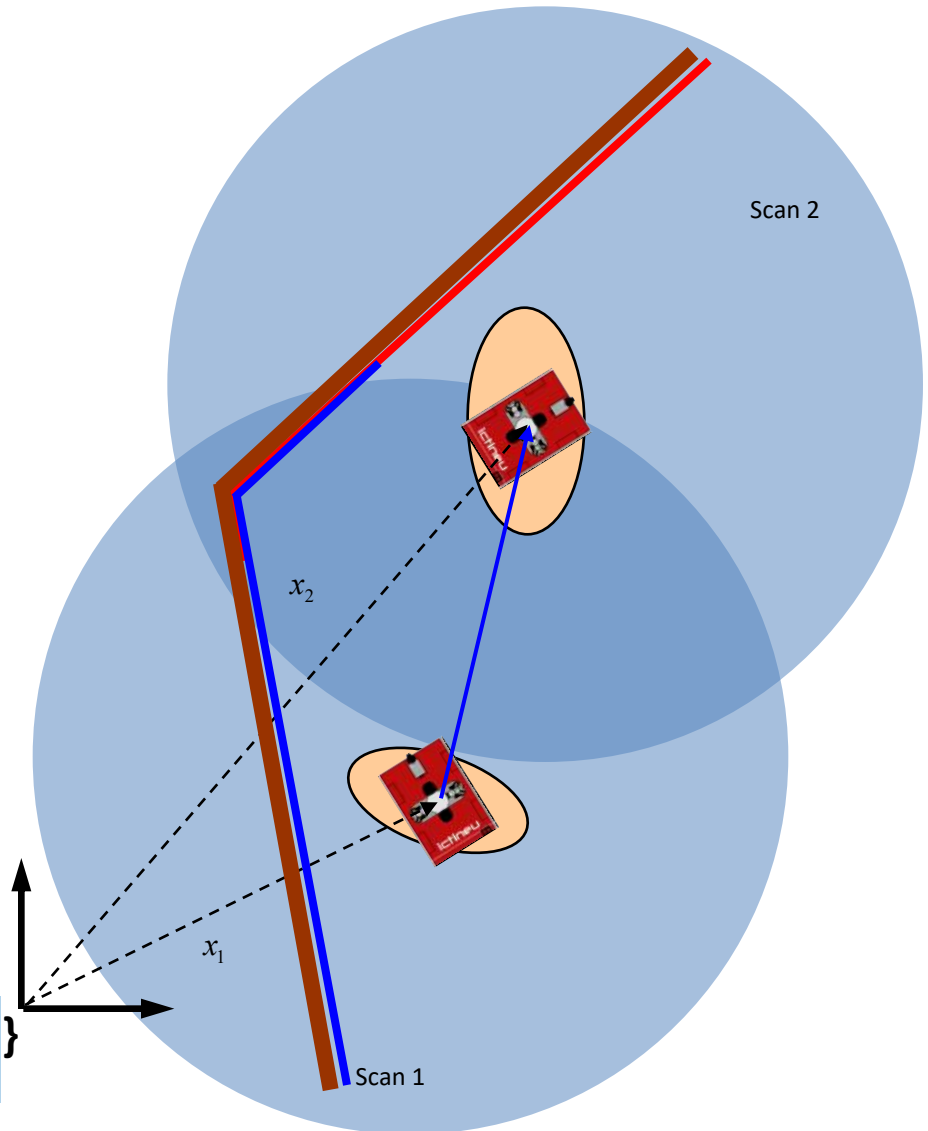
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^N\bar{P}_0] = AddNewPose({}^N\hat{x}_0, {}^NP_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus} J_{\ominus} {}^NP_{B_j} J_{\ominus}^T J_{1\oplus}^T + J_{2_\oplus} {}^NP_{B_k} J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

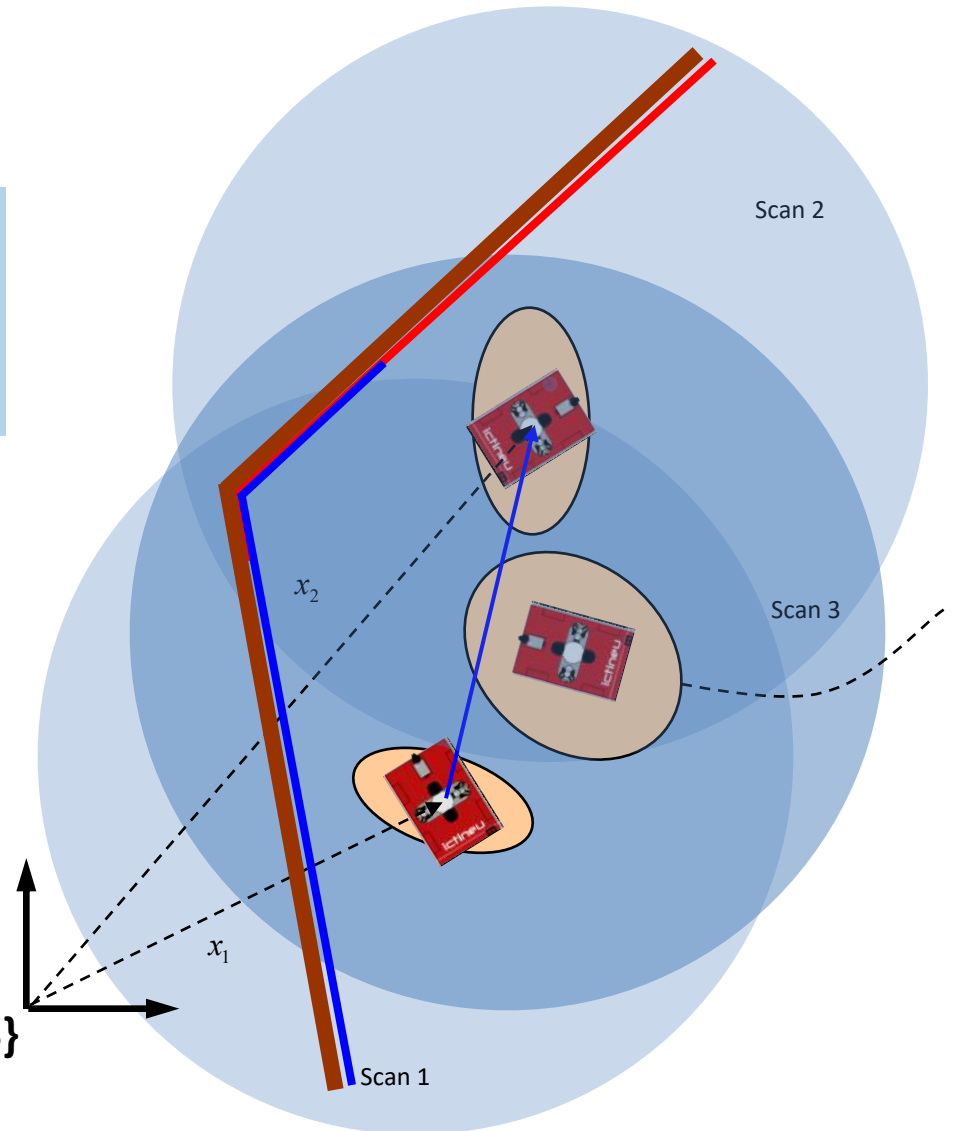
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



Method $\text{Localization}(\hat{x}_0, P_0)$

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\hat{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{2_\oplus}{}^NP_{B_k}J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

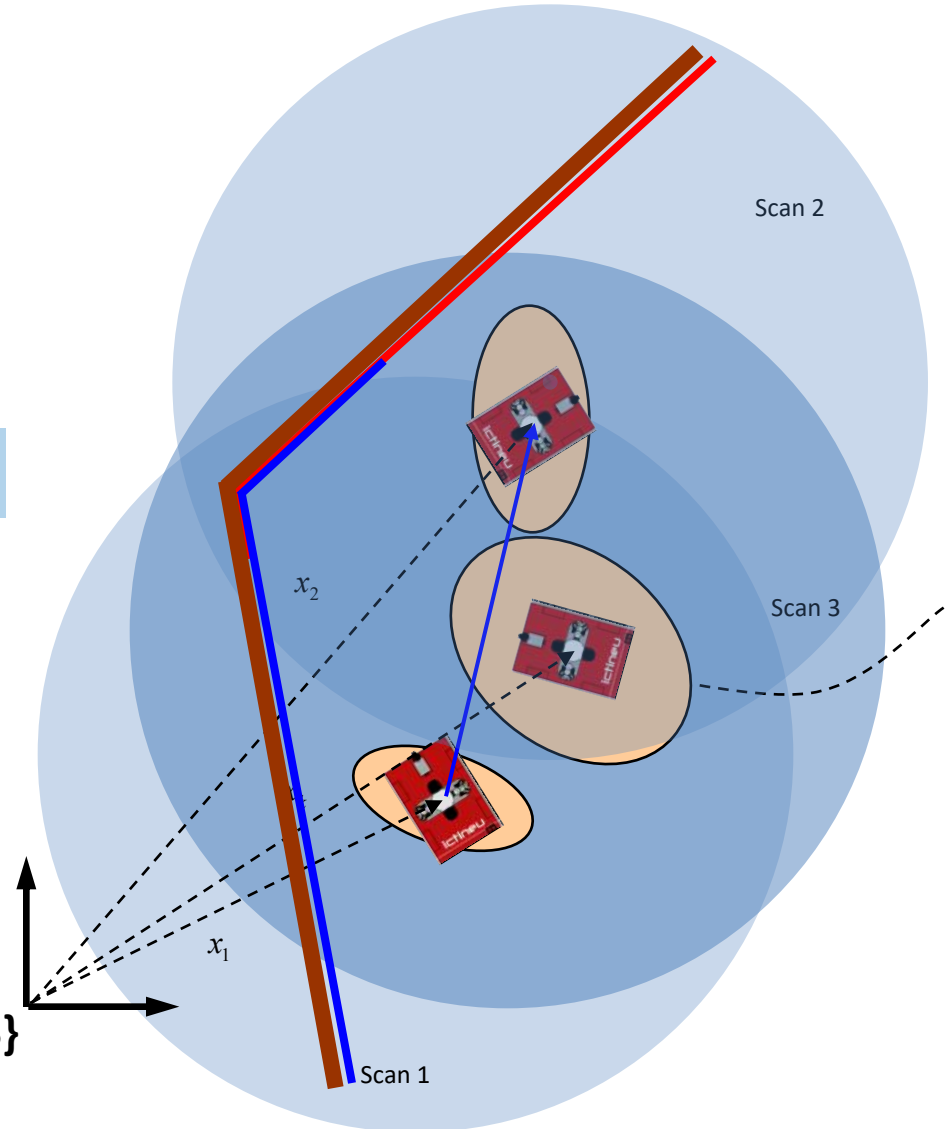
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* **do**

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* **then**

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{2_\oplus}{}^NP_{B_k}J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

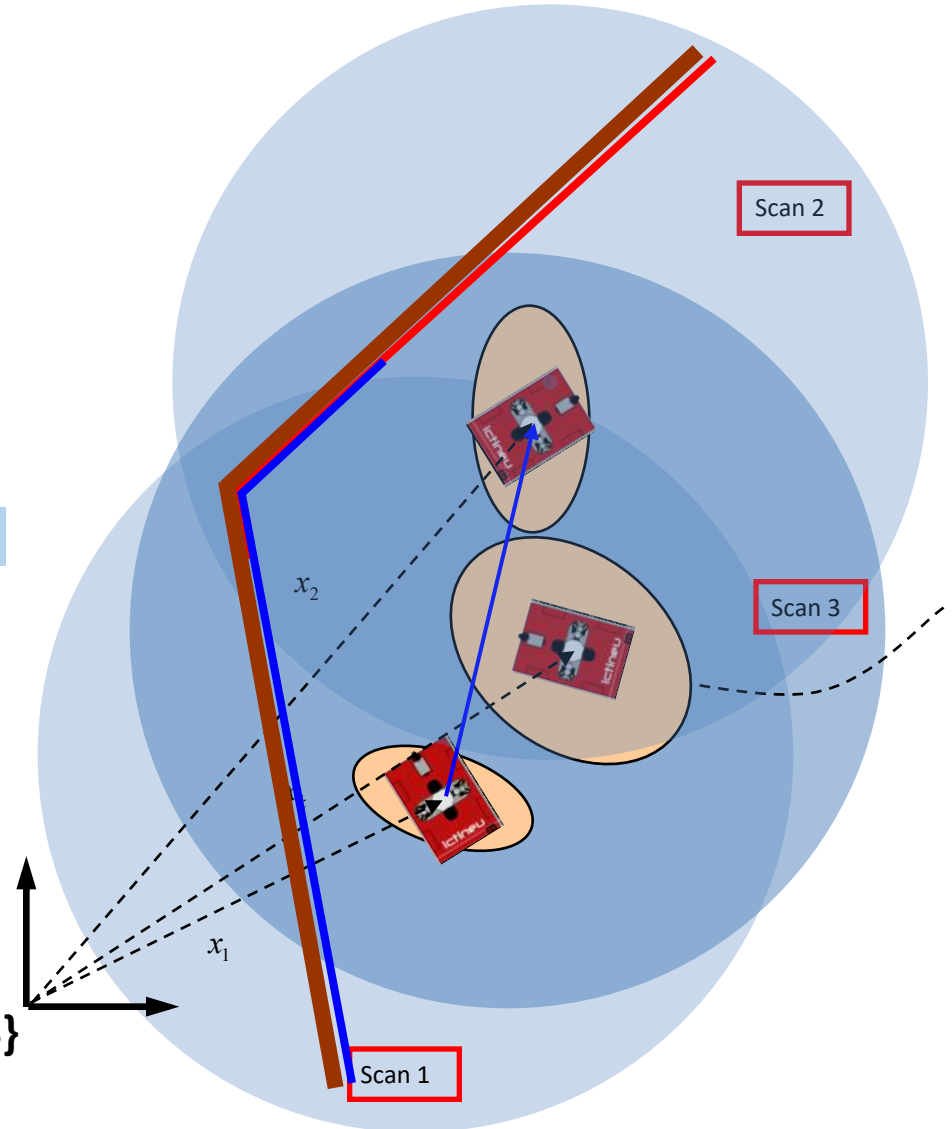
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i; c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

$[{}^N\hat{\bar{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

for $k = 1$ *to steps* do

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

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

  if *ScanAvailable* then

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map

    $\mathcal{H}_o = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    $c = 1; \mathcal{H}_p = [\,]$;

    for $i = 1$ *to* $length(\mathcal{H}_o)$ do // for all overlaps

      $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess

      ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{2_\oplus}{}^NP_{B_k}J_{2\oplus}^T$

      $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

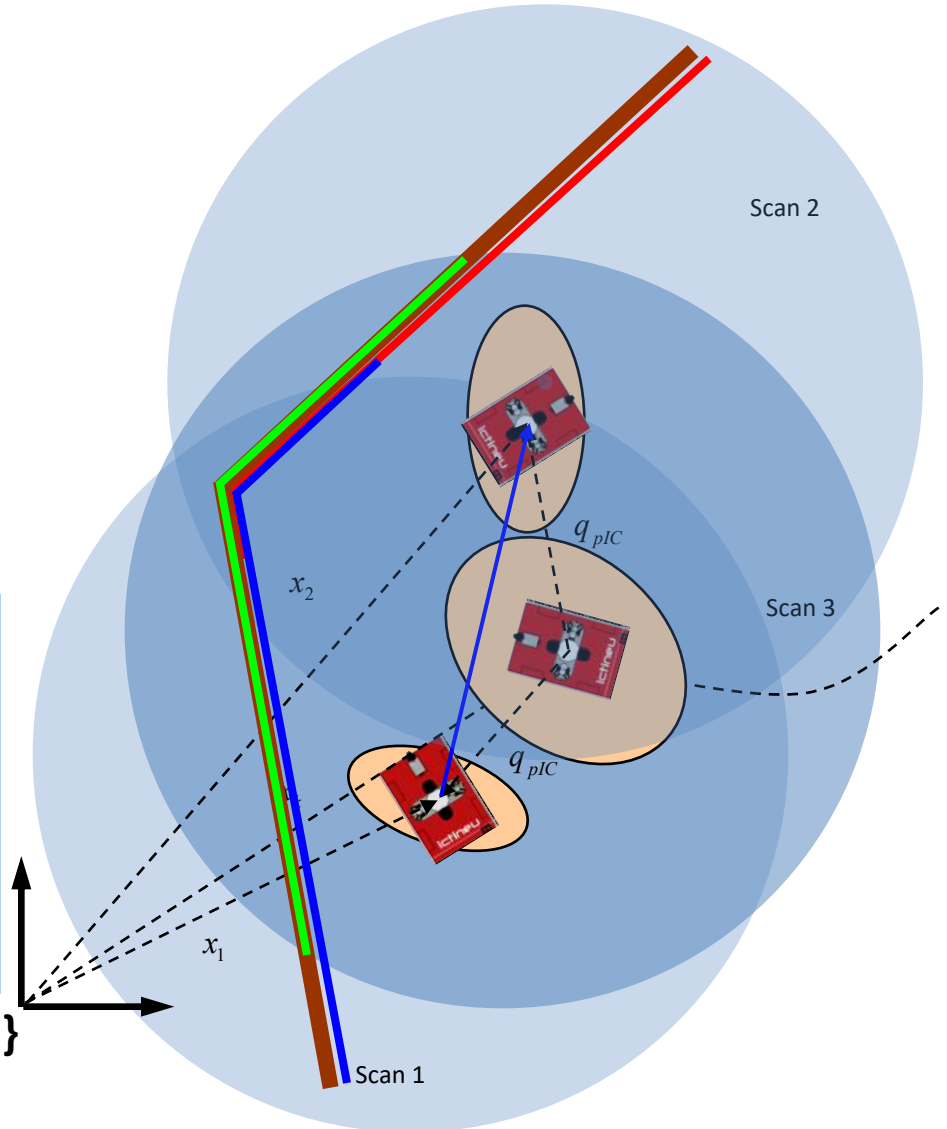
      if $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ then

        $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration

        $\mathcal{H}_p[c] = i$; $c = c + 1$;

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;



Method $\texttt{Localization}(\hat{x}_0, P_0)$

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

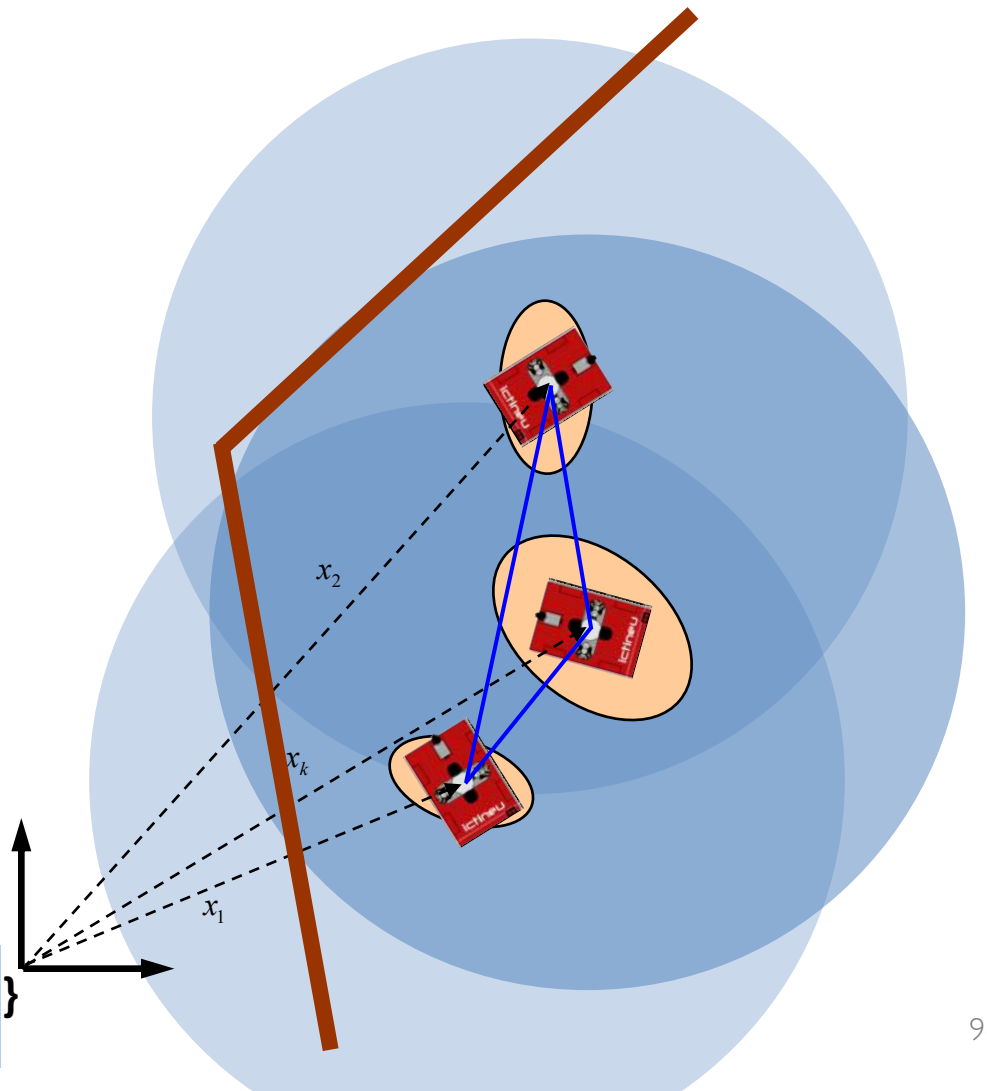
$[{}^{B_0}S_0, {}^{B_0}R_{S_0}] = GetScan()$; // Read the Scan from the sensor

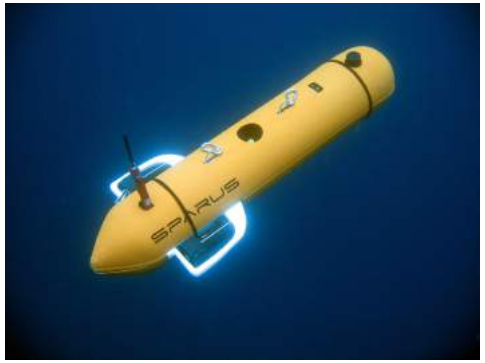
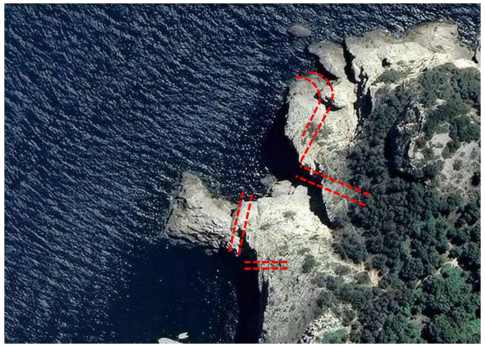
$[{}^N\hat{\hat{x}}_0, {}^NP_0] = AddNewPose({}^N\hat{x}_0, {}^N\bar{P}_0)$; // Grow the state vector

$\mathcal{M} = [[{}^{B_0}S_0, {}^{B_0}R_{S_0}]]$; // Store the scan in the map

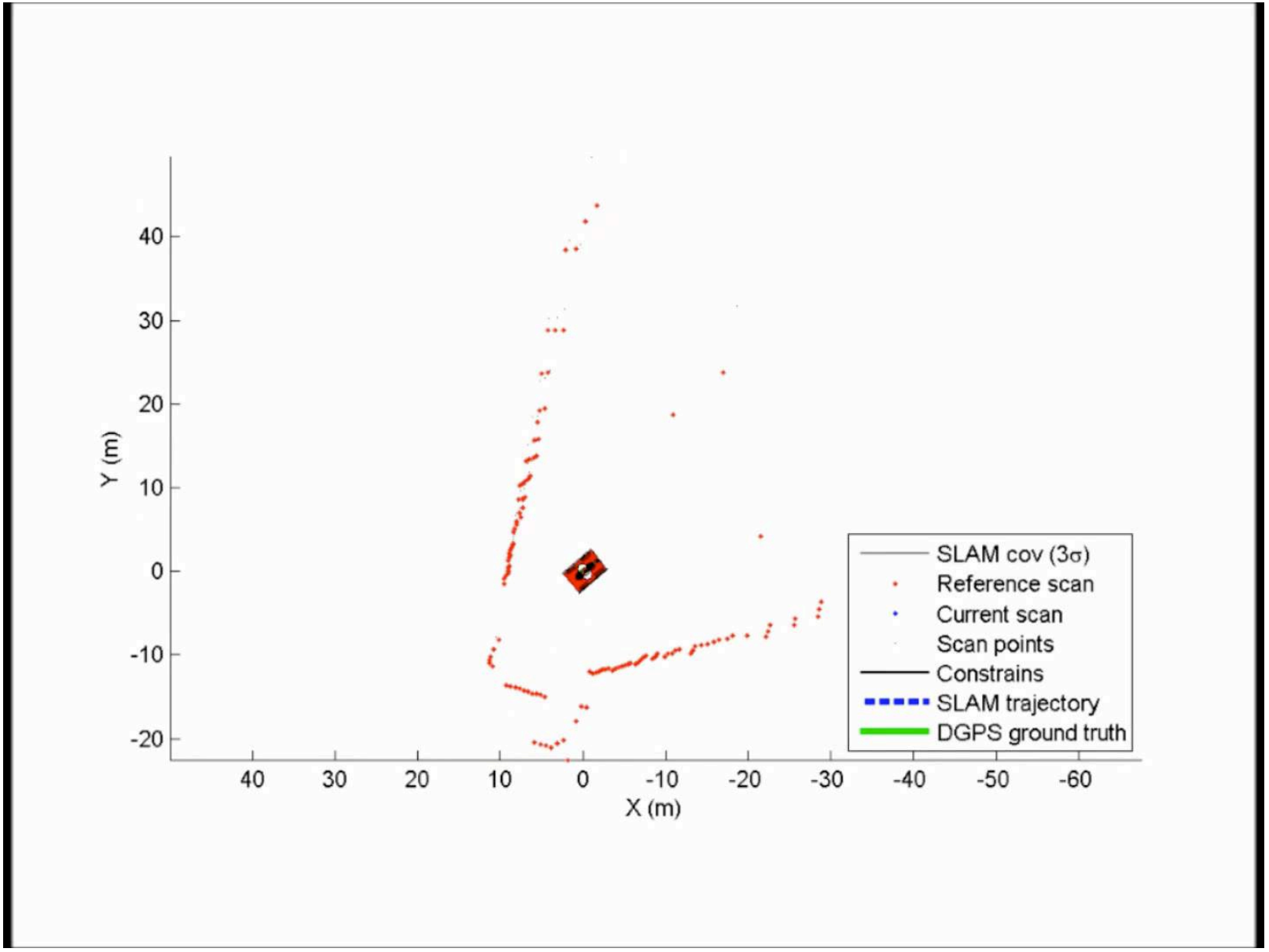
for $k = 1$ *to steps* **do**

- $[u_k, Q_k] = GetInput()$; // Get input to the motion model
- $[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = Prediction({}^N\hat{x}_{k-1}, {}^NP_{k-1}, u_k, Q_k)$;
- $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors
- $z_p = [\,]; z_p = [\,]; \mathcal{H}_p = [\,]$;
- **if** *ScanAvailable* **then**
  - $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;
  - $[{}^N\hat{\hat{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector
  - $\mathcal{M}[k] = [{}^{B_k}S_k, {}^{B_k}R_{S_k}]$; // Store the scan in the map
  - $\mathcal{H}_o = OverlappingScans({}^N\hat{\hat{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
  - $c = 1; \mathcal{H}_p = [\,]$;
  - **for** $i = 1$ *to* $length(\mathcal{H}_o)$ **do** // for all overlaps
    - $j = \mathcal{H}_o[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
    - $[{}^{B_j}S_j, {}^{B_j}R_{S_j}] = \mathcal{M}[j]$; // Get the scans from the map
    - ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scan Displacement guess
    - ${}^{B_j}P_{B_k} = J_{1\oplus}J_{\ominus}{}^NP_{B_j}J_{\ominus}^TJ_{1\oplus}^T + J_{\ominus}{}^NP_{B_k}J_{\ominus}^T$
    - $[z_r, R_r] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
    - **if** $IndividuallyCompatible({}^{B_j}x_{B_k}, {}^{B_j}P_{B_k}, z_r, R_r, \alpha)$ **then**
      - $z_p[c] = z_r$; $R_p[c] = R_r$; // Accepted registration
      - $\mathcal{H}_p[c] = i$; $c = c + 1$;
- $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\hat{x}}_k, z_m, R_m, z_p, R_p)$;
- $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\hat{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

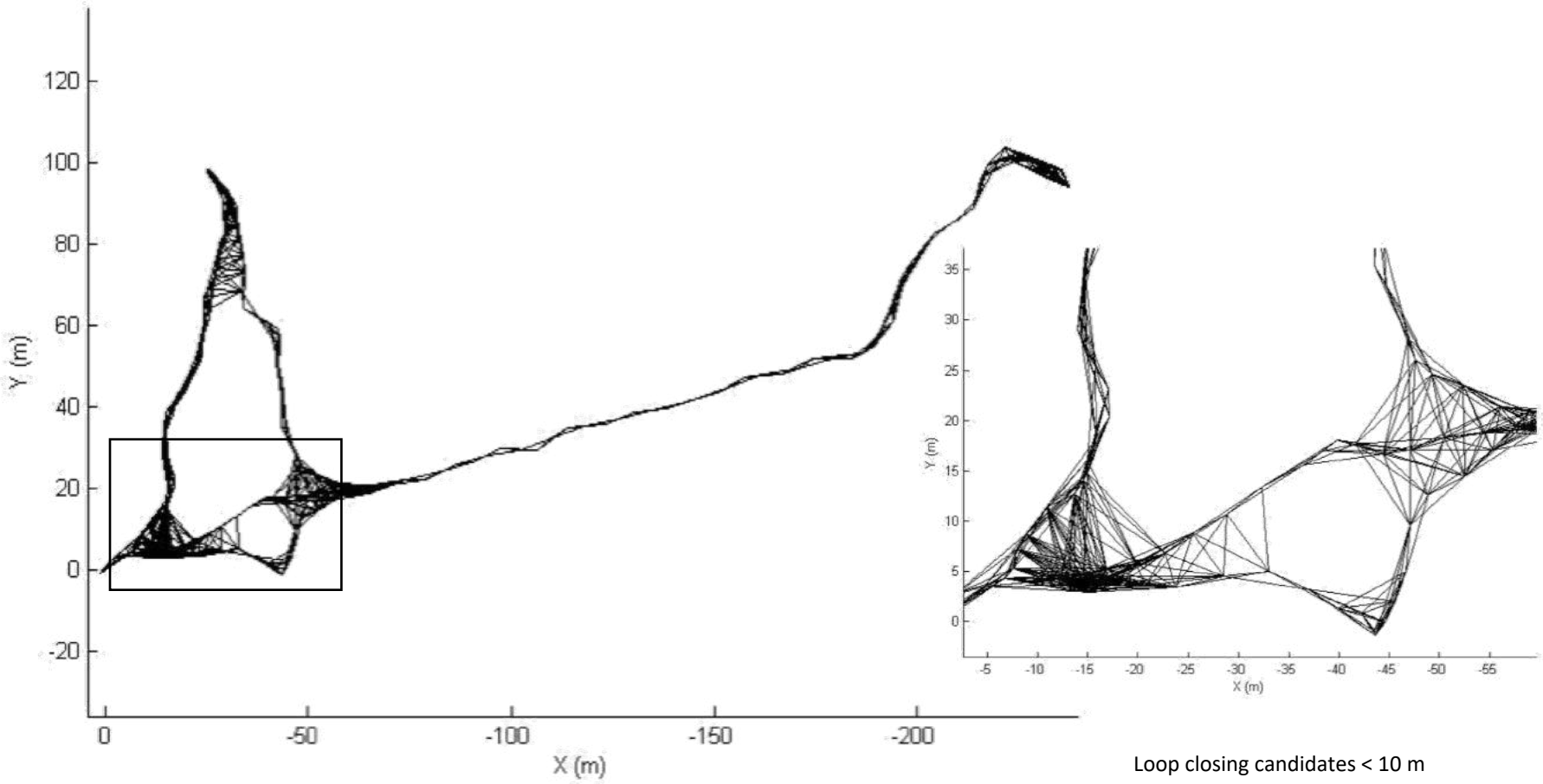


Vehicles	ICTINEU^{AUV}	SPARUS I	
			
Datasets	Marina	U/W tunnel	U/W caves system
			
	Path length: 600 m Exp. time: 53 min	Path length: 100 m Exp. time: 12 min	Path length: 500 m Exp. time: 32 min

Sonar points over dead-reckoning


40
30
20
10
0
-10
-20
Y (m)
40
30
20
10
0
-10
-20
-30
-40
-50
-60
X (m)
SLAM cov (3σ)
Reference scan
Current scan
Scan points
Constrains
SLAM trajectory
DGPS ground truth

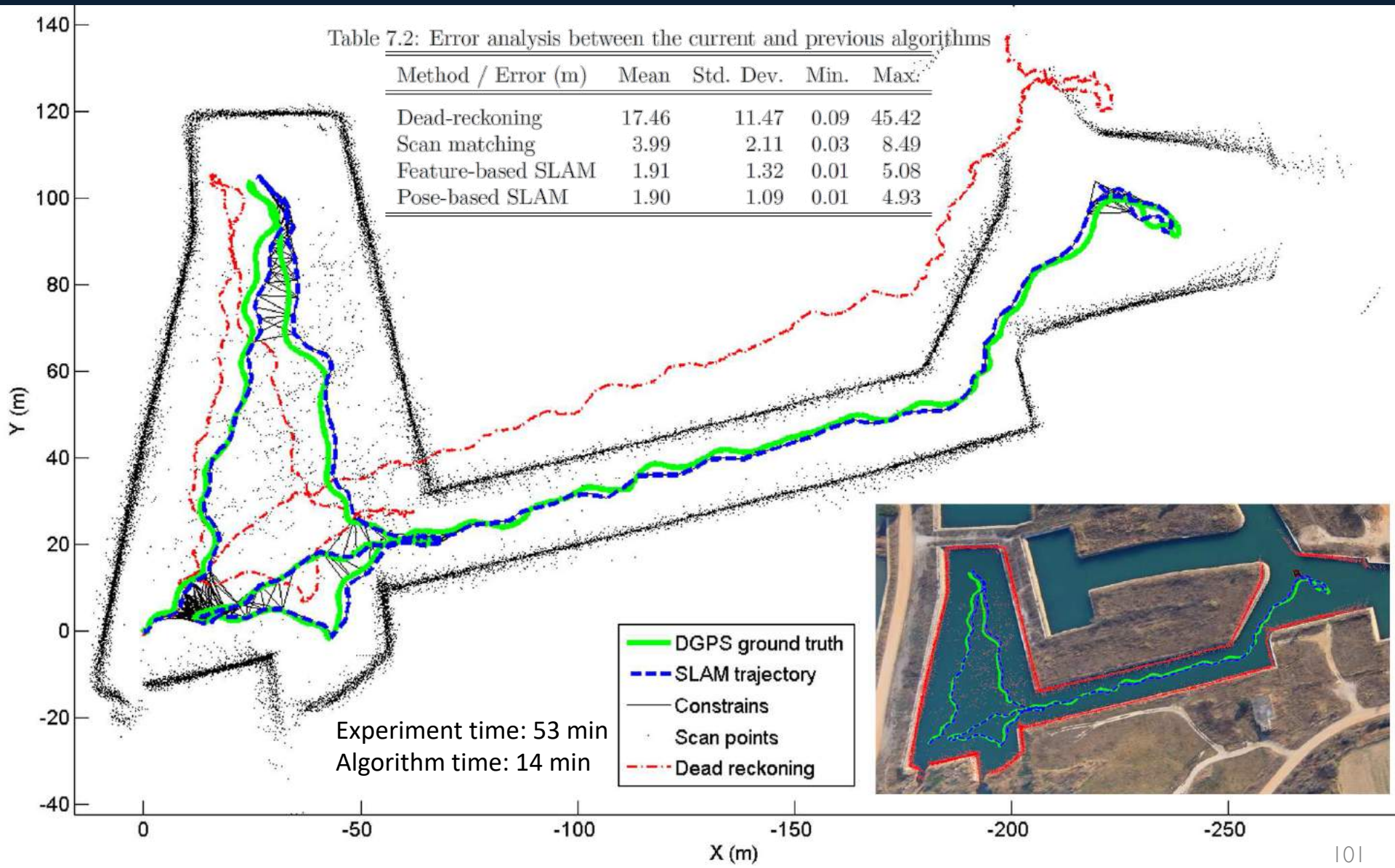
SLAM network of constraints



Loop closing candidates < 10 m

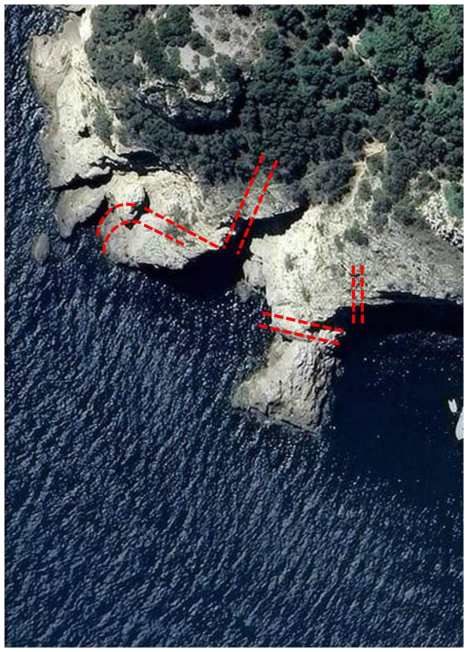
Table 7.2: Error analysis between the current and previous algorithms

Method / Error (m)	Mean	Std. Dev.	Min.	Max.
Dead-reckoning	17.46	11.47	0.09	45.42
Scan matching	3.99	2.11	0.03	8.49
Feature-based SLAM	1.91	1.32	0.01	5.08
Pose-based SLAM	1.90	1.09	0.01	4.93



PEKFSLAM Using MSISpiC

Sonar Scan Matching for Simultaneous Localization and Mapping in Confined Underwater Environments



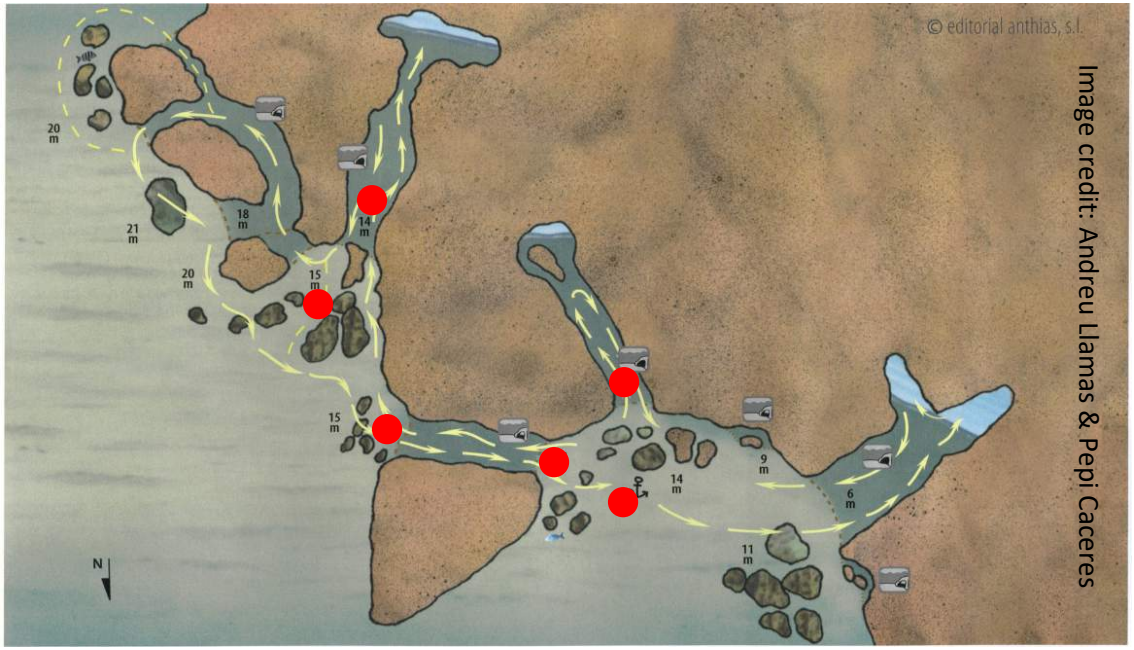
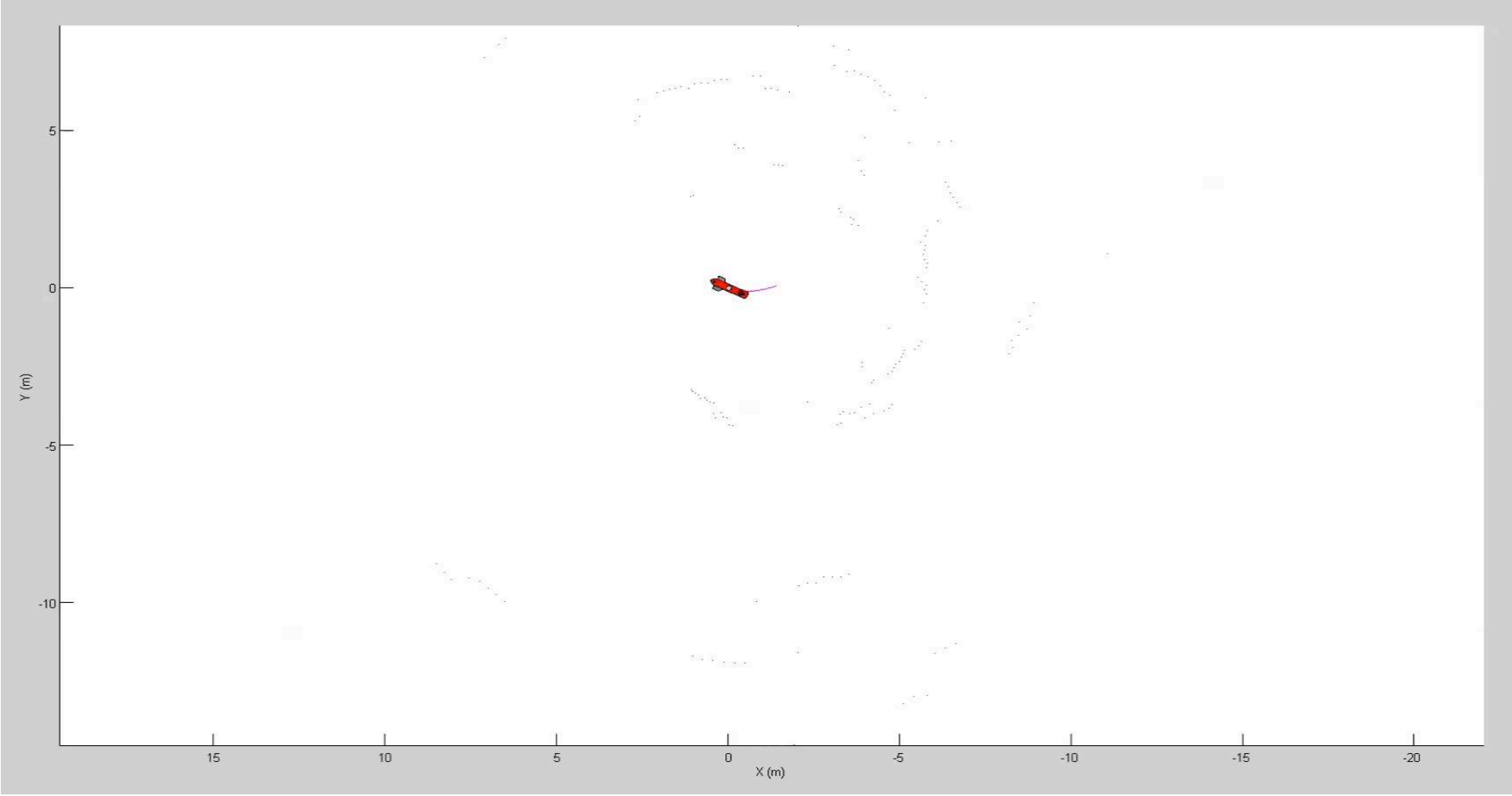
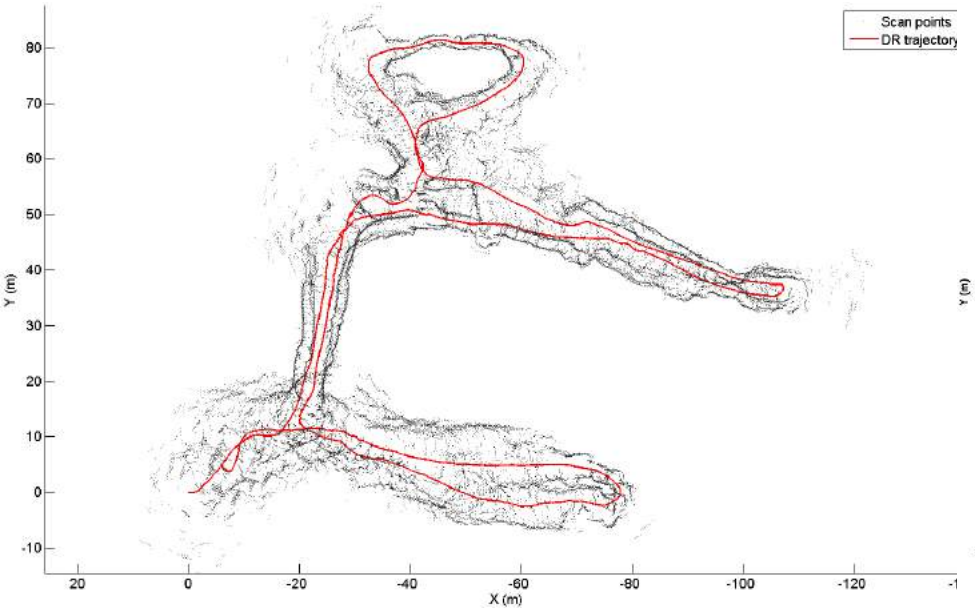


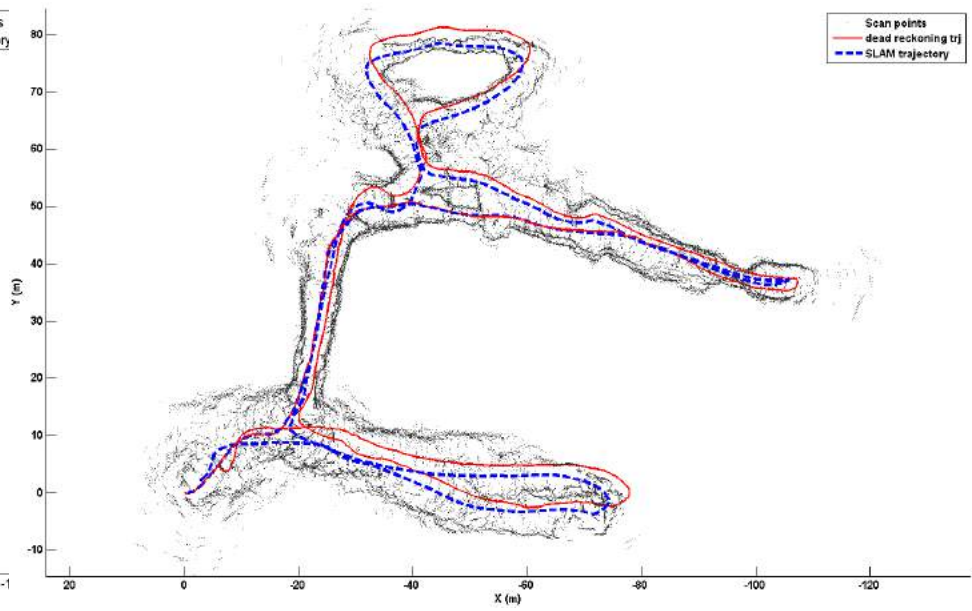
Image credit: Andreu Llamas & Pepi Caceres


5
0
-5
-10
Y (m)
15
10
5
0
-5
-10
-15
-20
X (m)

Map over dead-reckoning

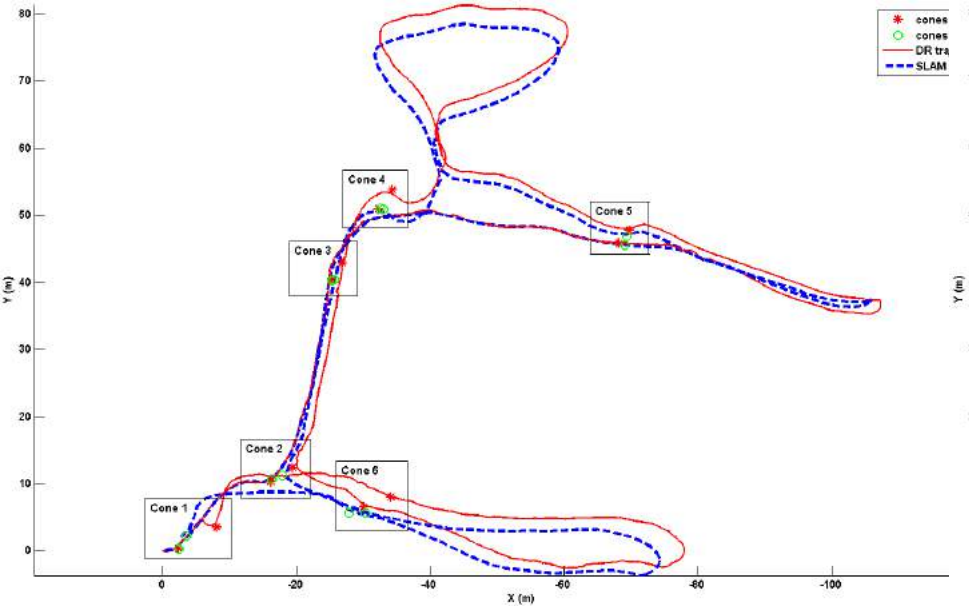


Map over SLAM

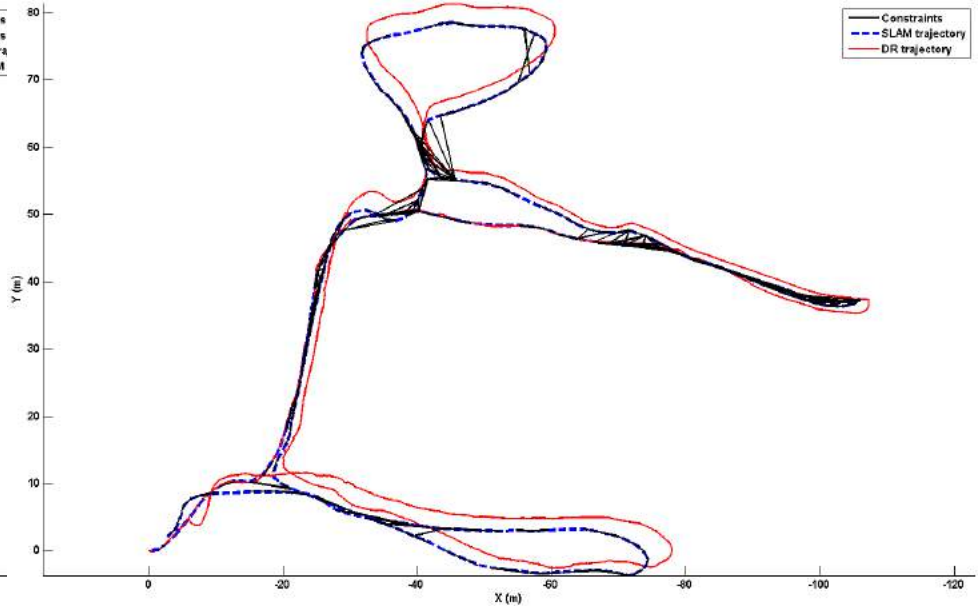


Experiment time: 32 min
Algorithm time: 19 min

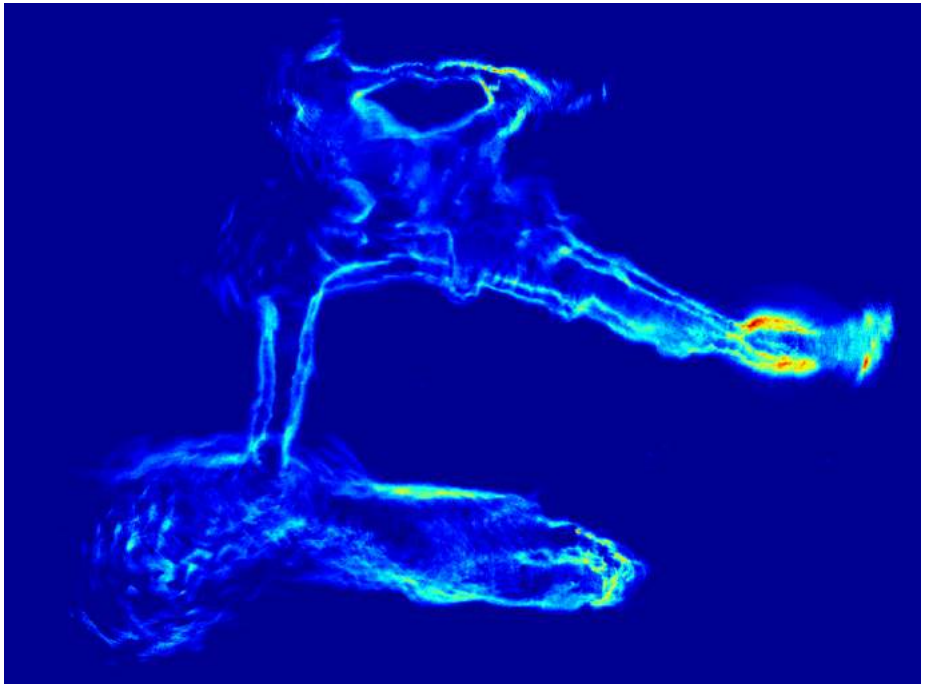
Cones positions (ground truth points)



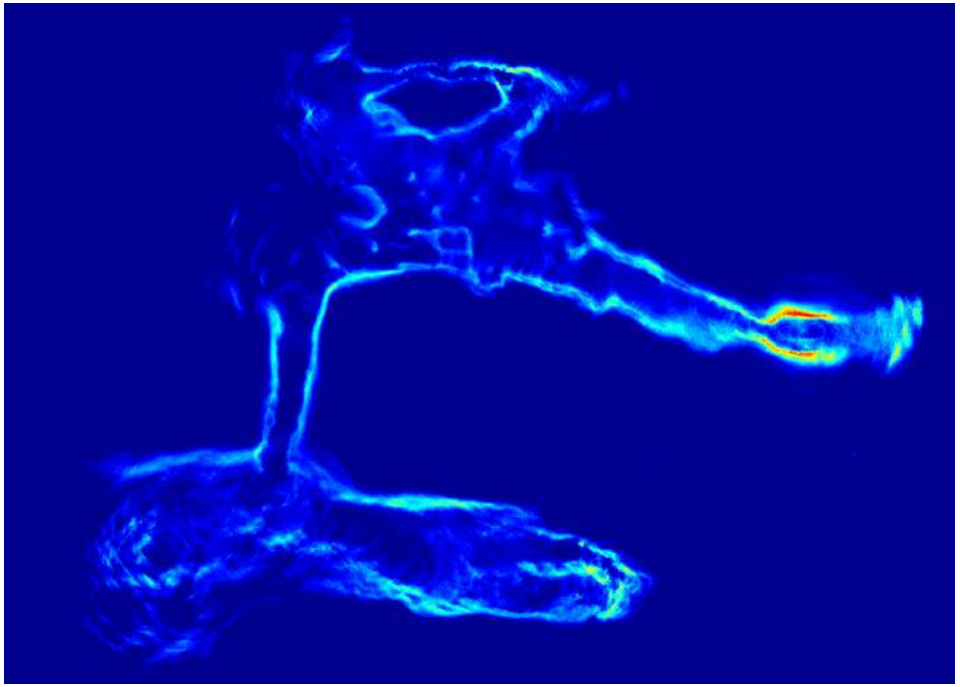
Map of constraints



Acoustic map over dead-reckoning



Acoustic map over SLAM



Hand made map

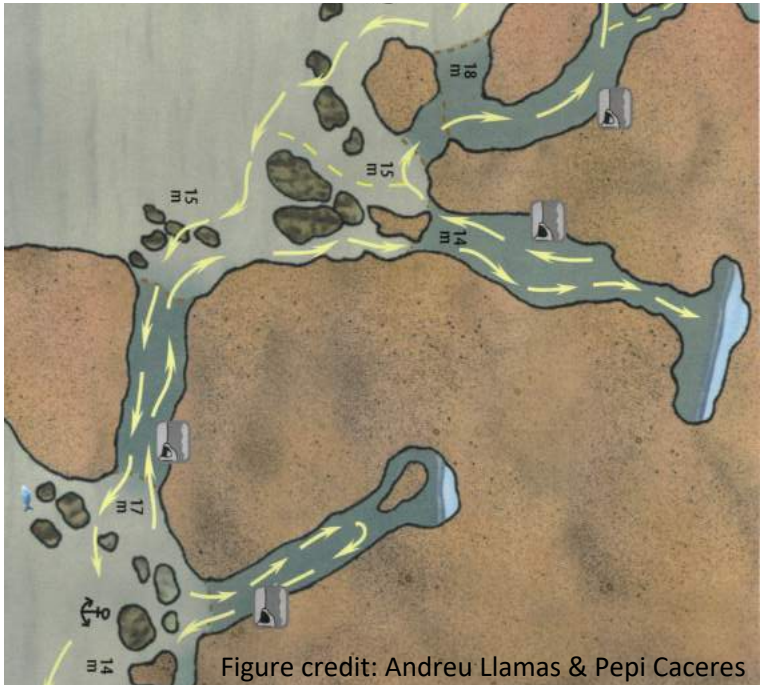
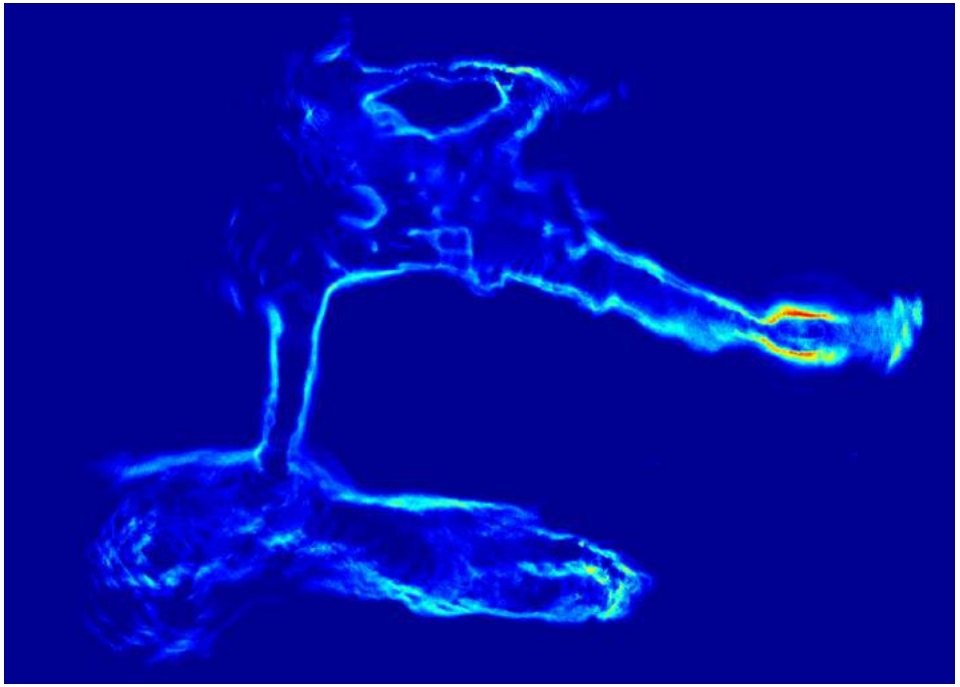


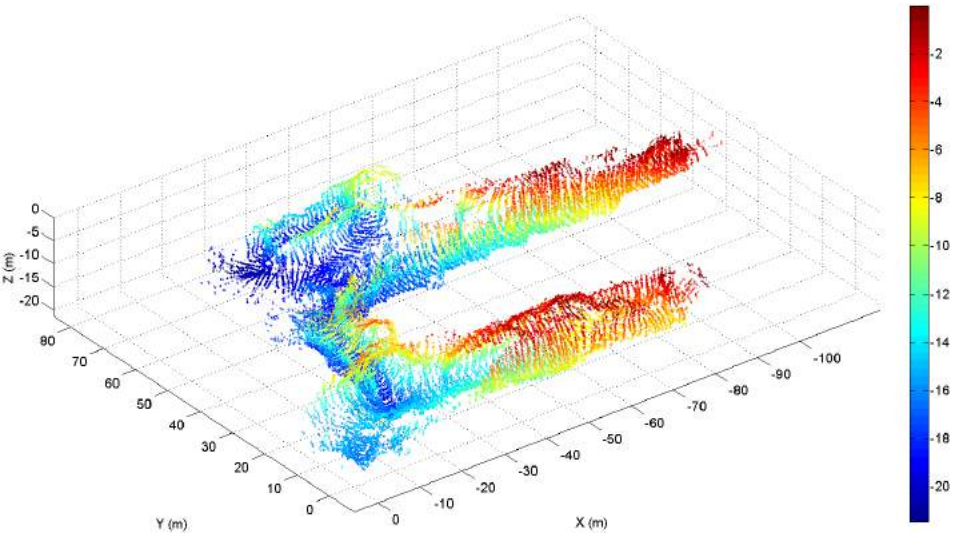
Figure credit: Andreu Llamas & Pepi Caceres

Acoustic map over SLAM



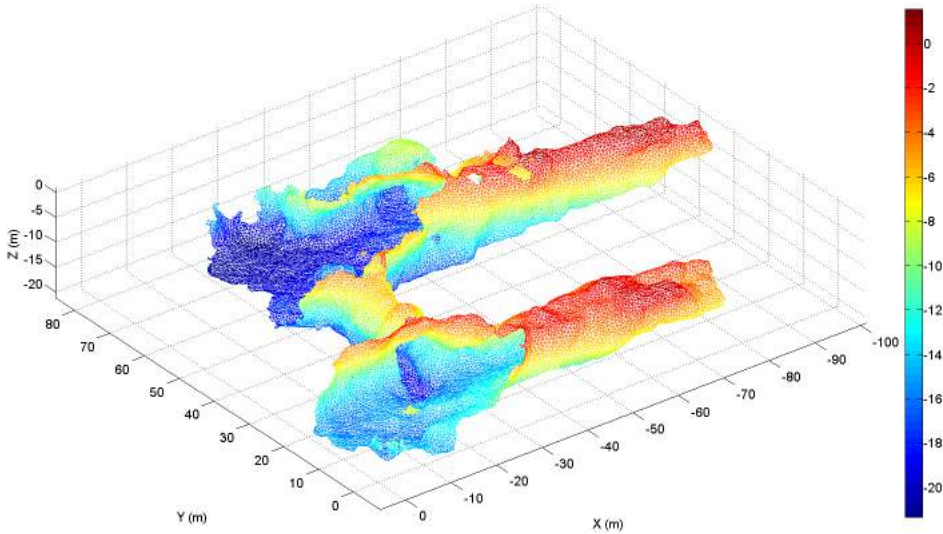
Maps not in scale

Cloud points from profiler

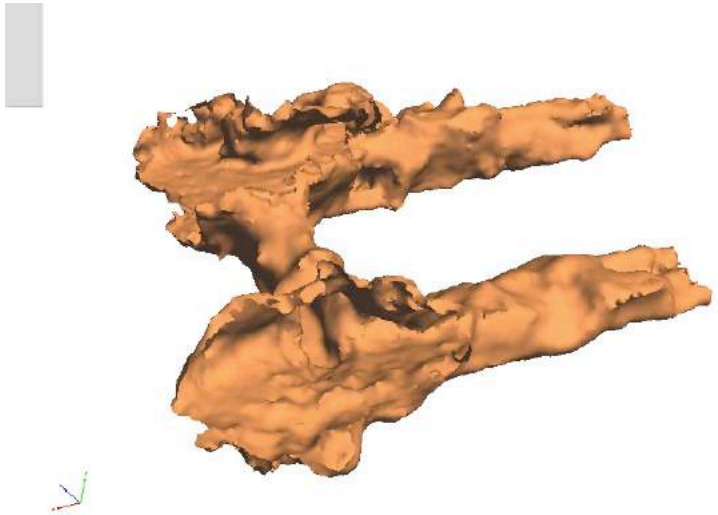


Tritech Super SeaKing DFP
Frequency: 1.1 MHz
Range: 10 m
Beam width: 1 deg
Scan rate (360-deg sector): 4 sec

Meshed points

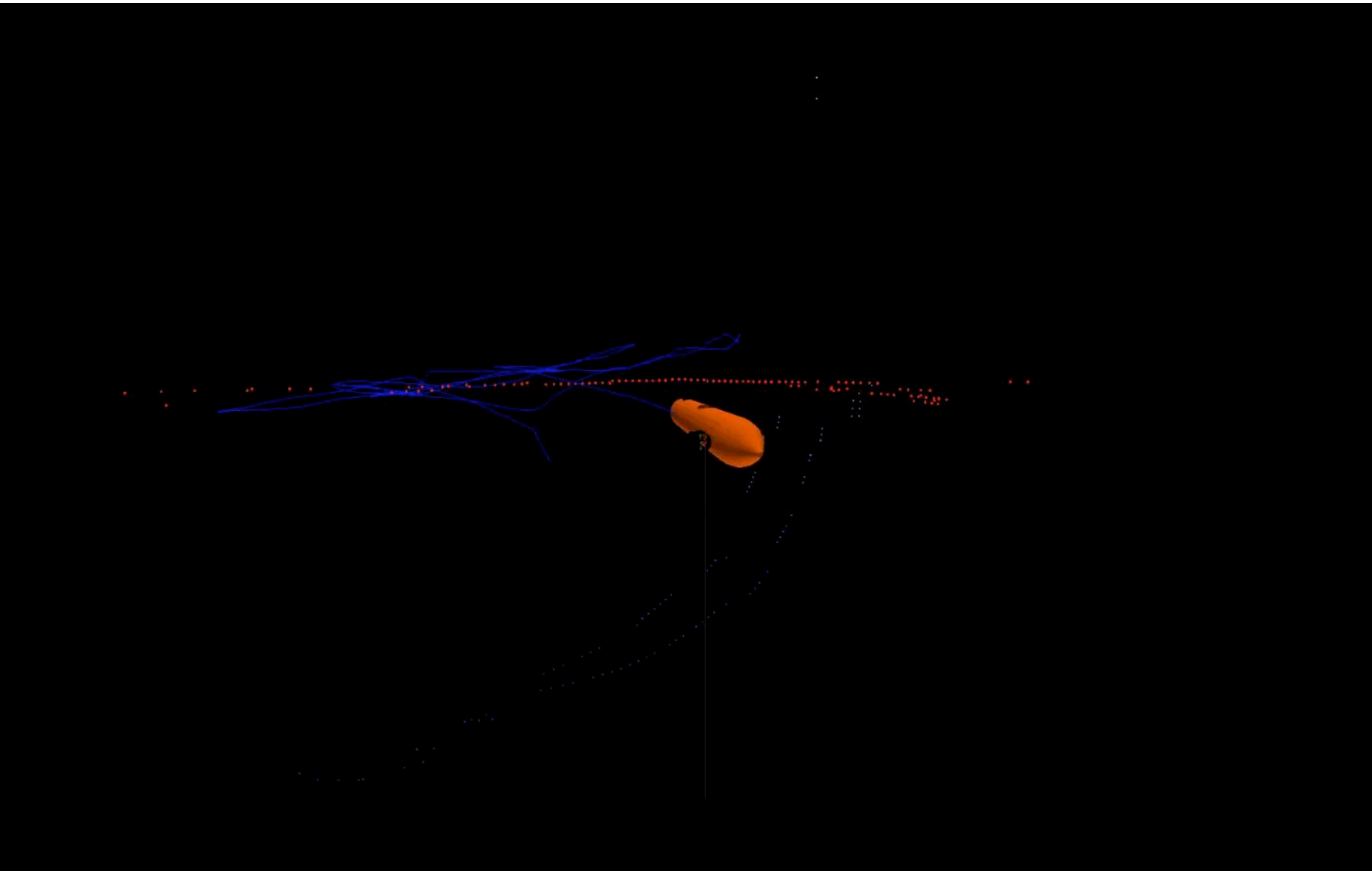


(Splat-based surface reconstruction from defect-laden point sets. Campos, R. *et al*, 2013)



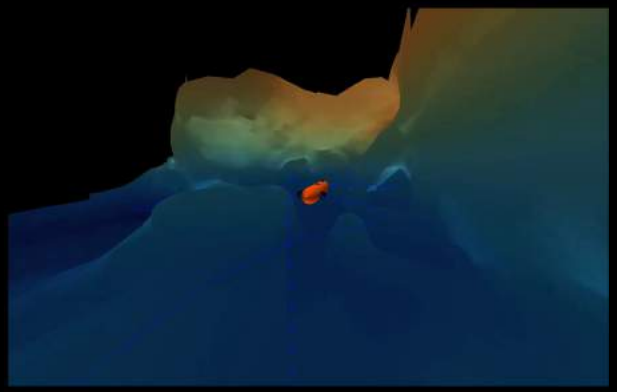
PEKFSLAM Using MSISpiC

Sonar Scan Matching for Simultaneous Localization and Mapping in Confined Underwater Environments



PEKFSLAM Using MSISpiC

Sonar Scan Matching for Simultaneous Localization and Mapping in Confined Underwater Environments






Universitat de Girona
Computer Vision and Robotics Group

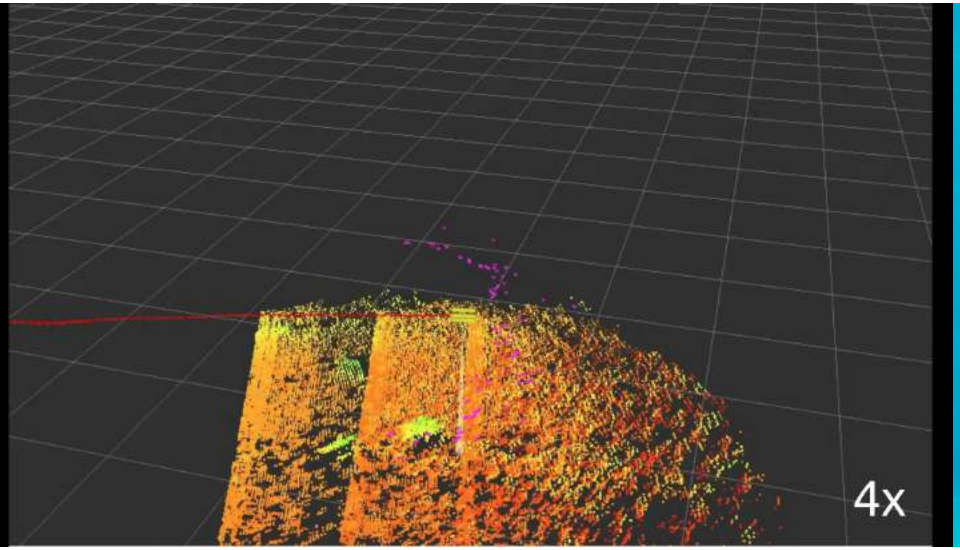
Probabilistic ICP for Bathymetry-based SLAM

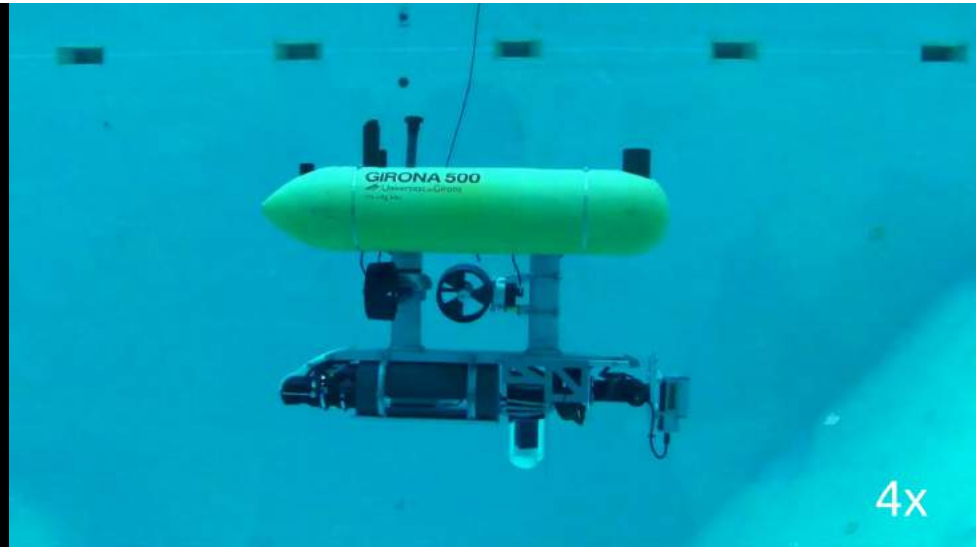
Albert Palomer Vila

Directors: Pere Ridao
David Ribas

PEKFSLAM Using Multibeam 3D Point Clouds

Probabilistic ICP for Bathymetry-based SLAM





- Divide mission into submaps
- 3D Gaussian point cloud
- Probabilistic 3D registration
- Multibeam mounted on a pan and tilt
- Pose-based EKF SLAM
- Also usable in 2.5D configuration

Method Localization($\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0$)

- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = [{}^N\hat{\boldsymbol{x}}_{B_0}, {}^N\boldsymbol{P}_{B_0}]$; // Initialize SLAM state vector
- $[{}^{B_0}\boldsymbol{S}_0, {}^{B_0}\boldsymbol{R}_{S_0}] = GetScan()$;; // Read the Scan from the sensor
- $[{}^N\hat{\boldsymbol{x}}_0, {}^N\boldsymbol{P}_0] = AddNewPose({}^N\hat{\boldsymbol{x}}_0, {}^N\bar{\boldsymbol{P}}_0)$; // Grow the state vector
- $\mathcal{M} = [{}^{B_0}\boldsymbol{S}_0]$; // Store the scan in the map
- **for** $k = 1$ *to steps* **do**
  - $[\boldsymbol{u_k}, \boldsymbol{Q_k}] = GetInput()$; // Get input to the motion model
  - $[{}^N\bar{\hat{\boldsymbol{x}}}_k, {}^N\boldsymbol{P}_k] = Prediction({}^N\hat{\boldsymbol{x}}_{k-1}, {}^N\boldsymbol{P}_{k-1}, \boldsymbol{u_k}, \boldsymbol{Q_k})$;
  - $[\boldsymbol{z_m}, \boldsymbol{R_m}] = GetMeasurement()$; // Read navigation sensors
  - **if** *ScanAvailable* **then**
    - $[{}^{B_k}\boldsymbol{S_k}, {}^{B_k}\boldsymbol{R}_{\boldsymbol{S_k}}] = GetScan()$;
    - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = AddNewPose({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = {}^{B_k}\boldsymbol{S_k}$; // Store the scan in the map
    - $\mathcal{H}_p = OverlappingScans({}^N\hat{\boldsymbol{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - **for** $i = 1$ *to* $length(\mathcal{H}_p)$ **do** // for all overlaps
      - $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map
      - ${}^{B_j}x_{B_k} = (\ominus {}^N x_{B_j}) \oplus {}^N x_{B_k}$; // Scans Displacement guess
      - $[\boldsymbol{z_p}, \boldsymbol{R_p}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
      - $i = i + 1$; // Go to next pair
  - $\mathcal{H}_p = DataAssociation({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_p}, \boldsymbol{R_p})$; // select compatible registrations
  - $[\boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\boldsymbol{x}}_k, \boldsymbol{z_m}, \boldsymbol{R_m}, \boldsymbol{z_p}, \boldsymbol{R_p})$;
  - $[{}^N\hat{\boldsymbol{x}}_k, {}^N\boldsymbol{P}_k] = Update({}^N\hat{\boldsymbol{x}}_k, {}^N\bar{\boldsymbol{P}}_k, \boldsymbol{z_k}, \boldsymbol{R_k}, \boldsymbol{H_k}, \boldsymbol{V_k}, \mathcal{H}_p)$;

> To implement a PEKFSLAM we need to:

1. Program the ***GetInput(·)*** function
2. Define the **motion model:** ${}^N\bar{\boldsymbol{x}}_{B_k} = \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)$
   - Compute the motion model Jacobians:

$$\boldsymbol{J}_{\boldsymbol{f_x}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial {}^N\boldsymbol{x}_{B_{k-1}}} \;,\; \boldsymbol{J}_{\boldsymbol{f_w}} = \frac{\partial \boldsymbol{f}\left({}^N\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right)}{\partial \boldsymbol{w_k}}$$

3. Program the ***GetScan()*** function
4. Define the **observation model:**

$$\boldsymbol{z_k} = \boldsymbol{h}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{p_k}})$$

$$\begin{bmatrix} \boldsymbol{z_{ij}} \\ \vdots \\ \boldsymbol{z_{pq}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{ij_k}}) \\ \vdots \\ \boldsymbol{h_{pq}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v}_{\boldsymbol{pq_k}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta x_{ij}} \\ \vdots \\ \boldsymbol{\Delta x_{pq}} \end{bmatrix}$$

   - Compute the observation Jacobians:

$\forall i, j$ compute:

$$\boldsymbol{J}_{\boldsymbol{h_x}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial {}^N\boldsymbol{x}_{B_k}} \;,\; \boldsymbol{J}_{\boldsymbol{h_v}} = \frac{\partial \boldsymbol{h_{ij}}({}^N\bar{\boldsymbol{x}}_k, \boldsymbol{v_k})}{\partial \boldsymbol{v_k}}$$

5. Implement the ***Register(·)*** *function*

Motion Model

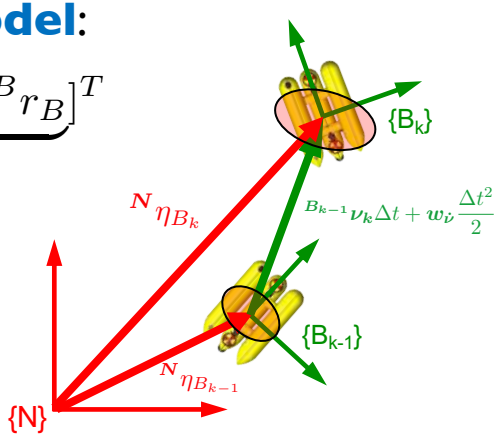
> The vehicle state is predicted using a constant velocity **motion model**:

$$ {}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}} = [\underbrace{{}^{N}x_B \; {}^{N}y_B \; {}^{N}z_B \; {}^{N}\phi_B \; {}^{N}\theta_B \; {}^{N}\psi_B}_{\text{Pose } {}^{N}\eta_{B_{k-1}}} | \underbrace{{}^{B}u_B \; {}^{B}v_B \; {}^{B}w_B \; {}^{B}p_B \; {}^{B}q_B \; {}^{B}r_B}_{\text{Velocity } {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}}]^T $$

$$ {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}} = \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right) = \begin{bmatrix} {}^{N}\boldsymbol{\eta}_{\boldsymbol{B_{k-1}}} \oplus ({}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}\Delta t + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\frac{\boldsymbol{\Delta t^2}}{\boldsymbol{2}}) \\ {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}} + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\Delta t \end{bmatrix} $$

where $\boldsymbol{u_k}$ is not used

and $\boldsymbol{w_k} = \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}} = [w_{\dot{u}} \; w_{\dot{v}} \; w_{\dot{w}} \; w_{\dot{r}}]^T$ is the acceleration noise



Motion Model

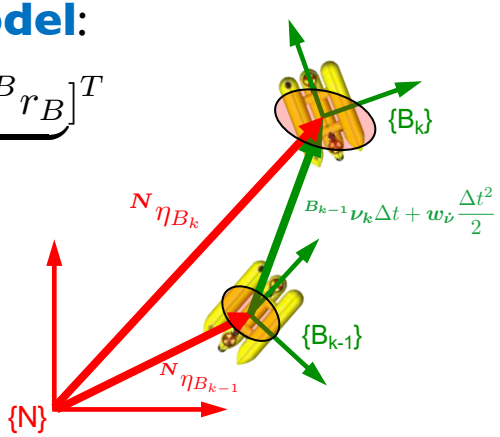
> The vehicle state is predicted using a constant velocity **motion model**:

$$ {}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}} = [\underbrace{{}^{N}x_B \ {}^{N}y_B \ {}^{N}z_B \ {}^{N}\phi_B \ {}^{N}\theta_B \ {}^{N}\psi_B}_{\text{Pose } {}^{N}\eta_{B_{k-1}}} | \underbrace{{}^{B}u_B \ {}^{B}v_B \ {}^{B}w_B \ {}^{B}p_B \ {}^{B}q_B \ {}^{B}r_B}_{\text{Velocity } {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}}]^T $$

$$ {}^{N}\bar{\boldsymbol{x}}_{\boldsymbol{B_k}} = \boldsymbol{f}\left({}^{N}\boldsymbol{x}_{\boldsymbol{B_{k-1}}}, \boldsymbol{u_k}, \boldsymbol{w_k}\right) = \begin{bmatrix} {}^{N}\boldsymbol{\eta}_{\boldsymbol{B_{k-1}}} \oplus ({}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}}\Delta t + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\frac{\boldsymbol{\Delta t^2}}{\boldsymbol{2}}) \\ {}^{B}\boldsymbol{\nu}_{\boldsymbol{k-1}} + \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}}\Delta t \end{bmatrix} $$

where $\boldsymbol{u_k}$ is not used

and $\boldsymbol{w_k} = \boldsymbol{w}_{\dot{\boldsymbol{\nu}}_{\boldsymbol{k}}} = [w_{\dot{u}} \ w_{\dot{v}} \ w_{\dot{w}} \ w_{\dot{r}}]^T$ is the acceleration noise



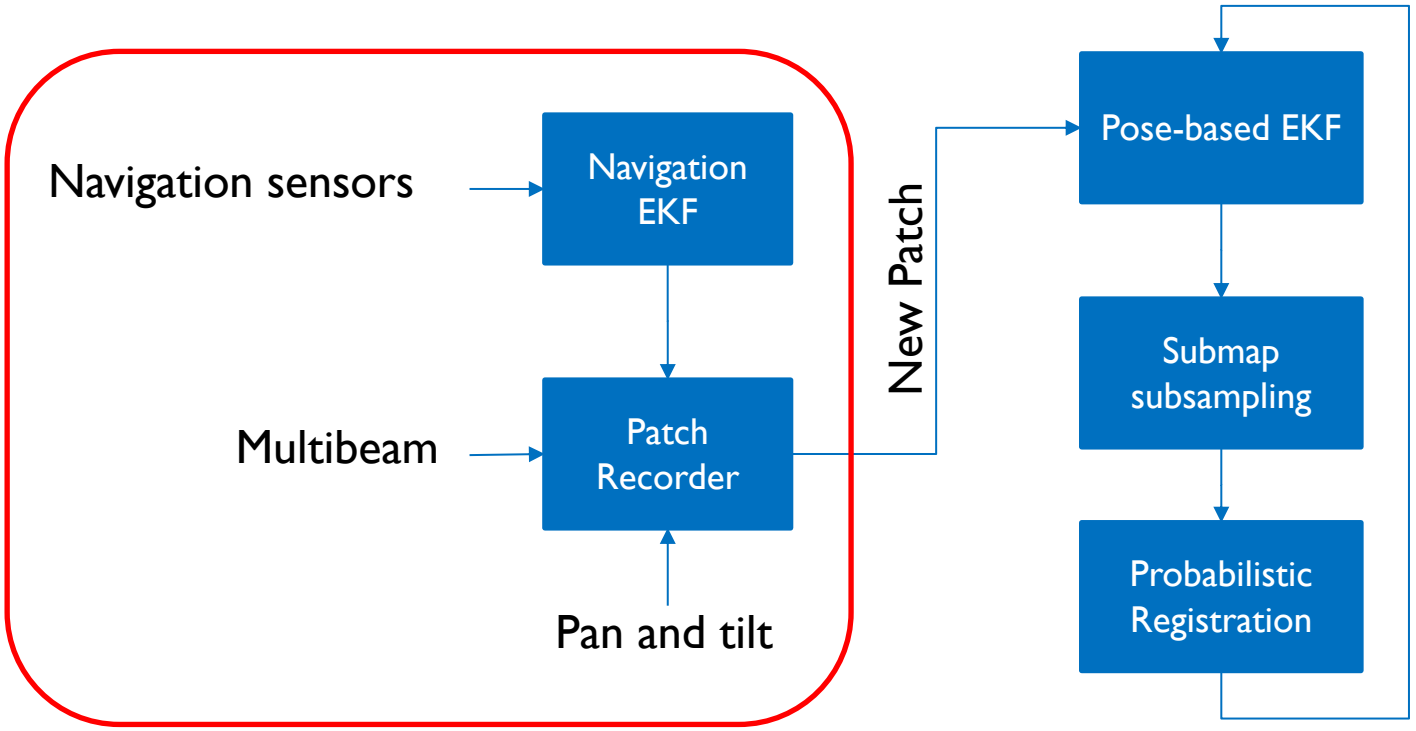
Measurements Observation Model

$$ \boldsymbol{z_{AHRS}} = \begin{bmatrix} z_\phi \\ z_\theta \\ z_\psi \end{bmatrix} = \begin{bmatrix} \boldsymbol{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} \end{bmatrix} \begin{bmatrix} {}^{N}\boldsymbol{\eta_1} \\ {}^{N}\boldsymbol{\eta_2} \\ {}^{B}\boldsymbol{\nu_1} \\ {}^{B}\boldsymbol{\nu_2} \end{bmatrix} + \boldsymbol{v_{\eta_2}} $$

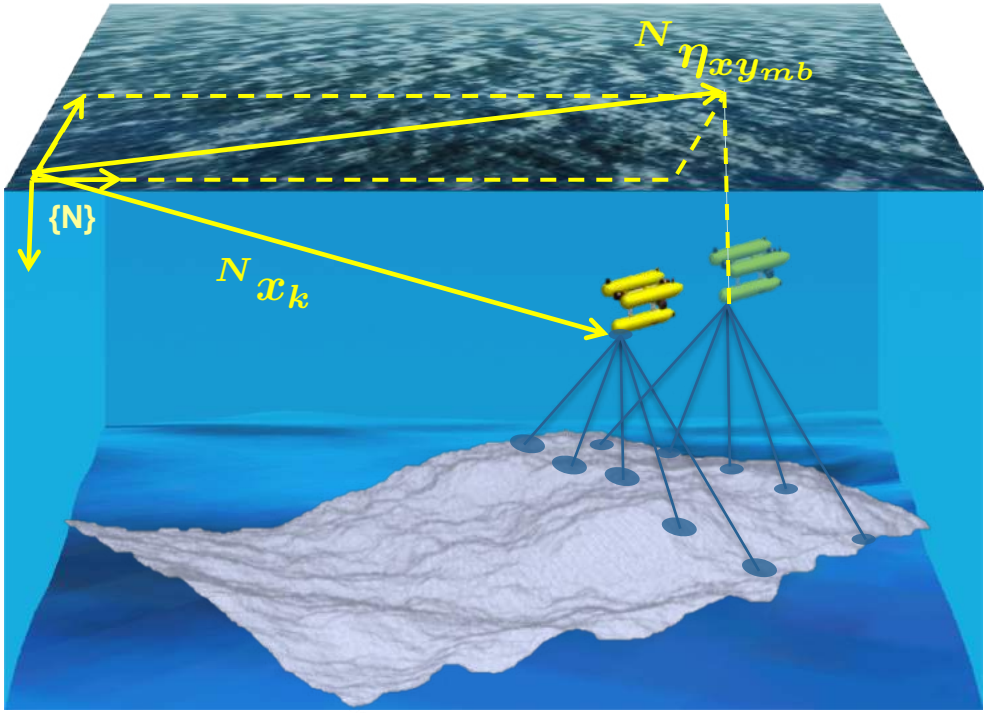
$$ z_{depth} = \begin{bmatrix} 0 & 0 & 1 & \boldsymbol{0_{1\times3}} & \boldsymbol{0_{1\times3}} & \boldsymbol{0_{1\times3}} \end{bmatrix} \begin{bmatrix} {}^{N}\boldsymbol{\eta_1} \\ {}^{N}\boldsymbol{\eta_2} \\ {}^{B}\boldsymbol{\nu_1} \\ {}^{B}\boldsymbol{\nu_2} \end{bmatrix} + v_{depth} $$

$$ \boldsymbol{z_{DVL}} = \begin{bmatrix} z_u \\ z_v \\ z_w \end{bmatrix} = \begin{bmatrix} \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times3}} \end{bmatrix} \begin{bmatrix} {}^{N}\boldsymbol{\eta_1} \\ {}^{N}\boldsymbol{\eta_2} \\ {}^{B}\boldsymbol{\nu_1} \\ {}^{B}\boldsymbol{\nu_2} \end{bmatrix} + \boldsymbol{v_{DVL}} $$

GetScan()


Navigation sensors
Navigation EKF
Multibeam
Patch Recorder
Pan and tilt
New Patch
Pose-based EKF
Submap subsampling
Probabilistic Registration

GetScan()

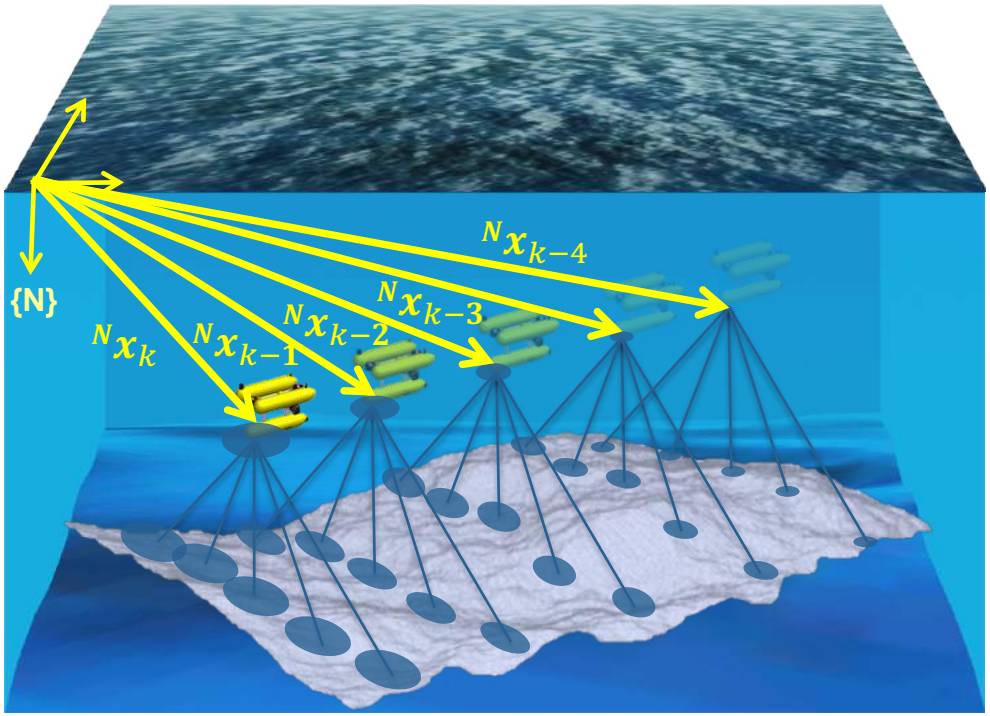


Trajectory

> Augmented state vector with the last MB XY.

$$\begin{bmatrix} {}^{N}\eta_{B_k} \\ {}^{B}\nu_k \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) = \begin{bmatrix} {}^{N}\eta_{B_{k-1}} \oplus \left({}^{B}\nu_{k-1}\Delta t + w_{\dot{\nu}_k}\frac{\Delta t^2}{2}\right) \\ {}^{B}\nu_{k-1} + w_{\dot{\nu}_k}\Delta t \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix}$$

GetScan()



Trajectory

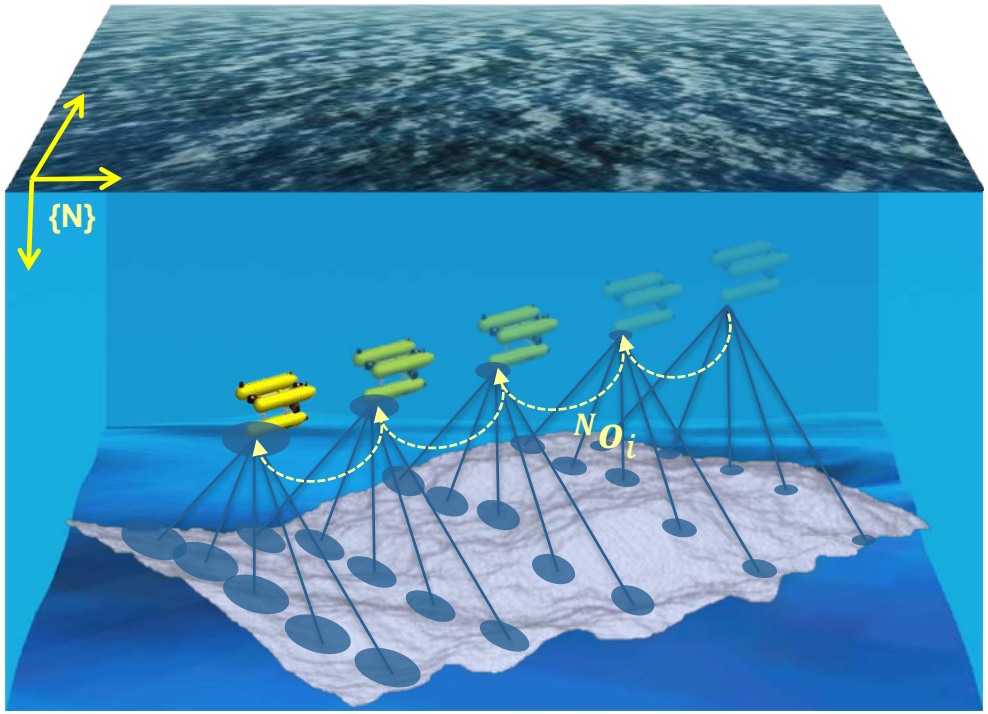
> Augmented state vector with the last MB XY.

$$\begin{bmatrix} {}^{N}\eta_{B_k} \\ {}^{B}\nu_k \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) = \begin{bmatrix} {}^{N}\eta_{B_{k-1}} \oplus \left({}^{B}\nu_{k-1}\Delta t + w_{\dot{\nu}_k}\frac{\Delta t^2}{2}\right) \\ {}^{B}\nu_{k-1} + w_{\dot{\nu}_k}\Delta t \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix}$$

> At each MB reading it is updated:

$$\begin{bmatrix} {}^{N}\eta_{B_k} \\ {}^{B}\nu_k \\ {}^{N}\eta_{xy_{B_{k-1}}} \end{bmatrix} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) = \begin{bmatrix} {}^{N}\eta_{B_{k-1}} \oplus \left({}^{B}\nu_{k-1}\Delta t + w_{\dot{\nu}_k}\frac{\Delta t^2}{2}\right) \\ {}^{B}\nu_{k-1} + w_{\dot{\nu}_k}\Delta t \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix}$$

GetScan()



Trajectory

> Augmented state vector with the last MB XY.

$$\begin{bmatrix} {}^{N}\eta_{B_k} \\ {}^{B}\nu_k \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) = \begin{bmatrix} {}^{N}\eta_{B_{k-1}} \oplus \left({}^{B}\nu_{k-1}\Delta t + w_{\dot{\nu}_k}\frac{\Delta t^2}{2}\right) \\ {}^{B}\nu_{k-1} + w_{\dot{\nu}_k}\Delta t \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix}$$

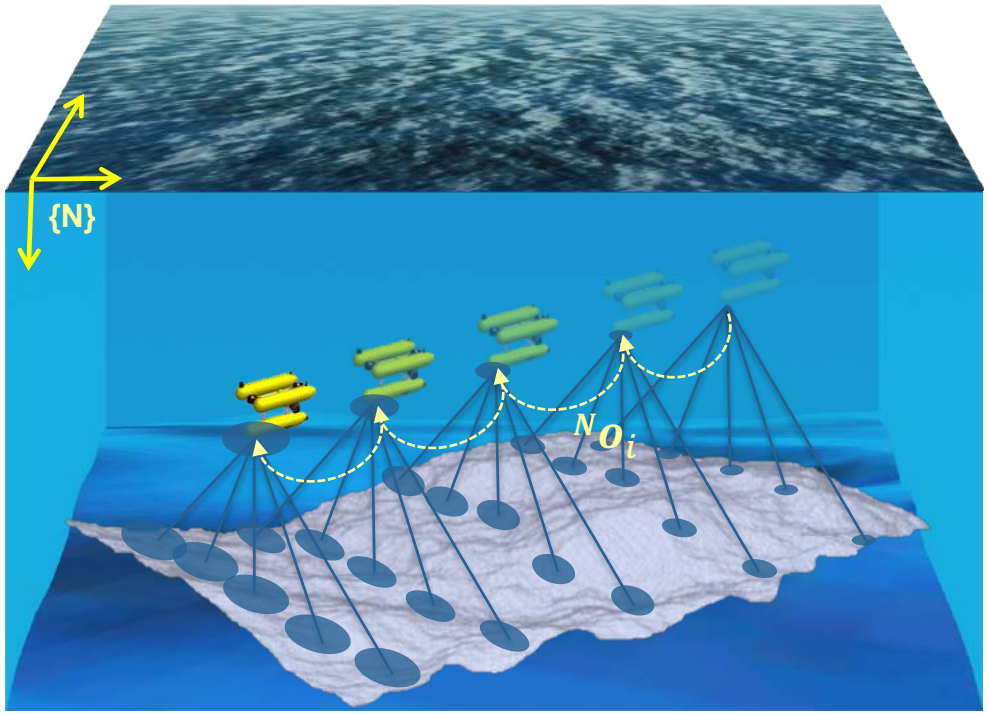
> At each MB reading it is updated:

$$\begin{bmatrix} {}^{N}\eta_{B_k} \\ {}^{B}\nu_k \\ {}^{N}\eta_{xy_{B_{k-1}}} \end{bmatrix} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) = \begin{bmatrix} {}^{N}\eta_{B_{k-1}} \oplus \left({}^{B}\nu_{k-1}\Delta t + w_{\dot{\nu}_k}\frac{\Delta t^2}{2}\right) \\ {}^{B}\nu_{k-1} + w_{\dot{\nu}_k}\Delta t \\ {}^{N}\eta_{xy_{mb}} \end{bmatrix}$$

> For all MB readings their relative MB reading displacements are computed.

$$^{N}\boldsymbol{o_k} = \begin{bmatrix} {}^{N}\boldsymbol{\eta_{xy_{B_k}}} - {}^{N}\boldsymbol{\eta_{xymb}} \\ {}^{N}z_{B_k} \\ {}^{N}\boldsymbol{\eta_{2_{B_k}}} \end{bmatrix}$$

GetScan()



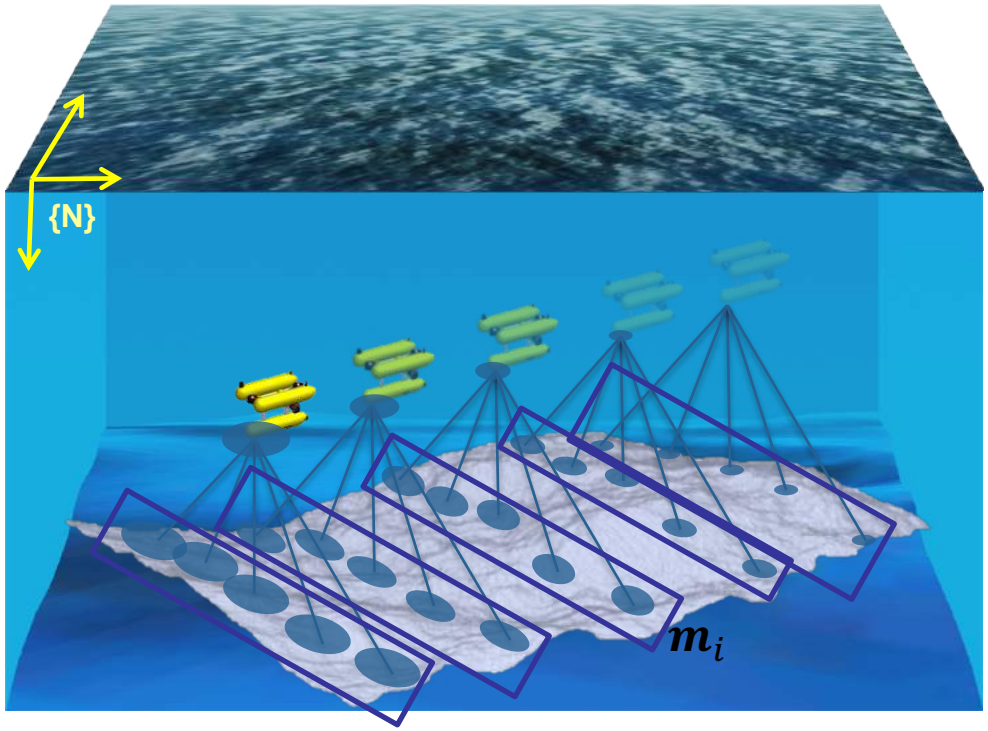
> Store the Incremental Trajectory.

$$^{N}o = \{^{N}o_1 \dots ^{N}o_n\} \; ; \; ^{N}o_i = \mathcal{N}(^{N}\hat{o}_i, ^{N}P_{o_i})$$

> Point Cloud Characterization

$$S = [\quad ^{N}O \quad]$$

GetScan()



> Store the Incremental Trajectory.

$$^{N}\boldsymbol{o} = \{^{N}\boldsymbol{o_1} \dots ^{N}\boldsymbol{o_n}\} \;;\; ^{N}\boldsymbol{o_i} = \mathcal{N}(^{N}\hat{\boldsymbol{o}}_{\boldsymbol{i}}, ^{N}\boldsymbol{P}_{\boldsymbol{o_i}})$$

> Store polar MB profiles.

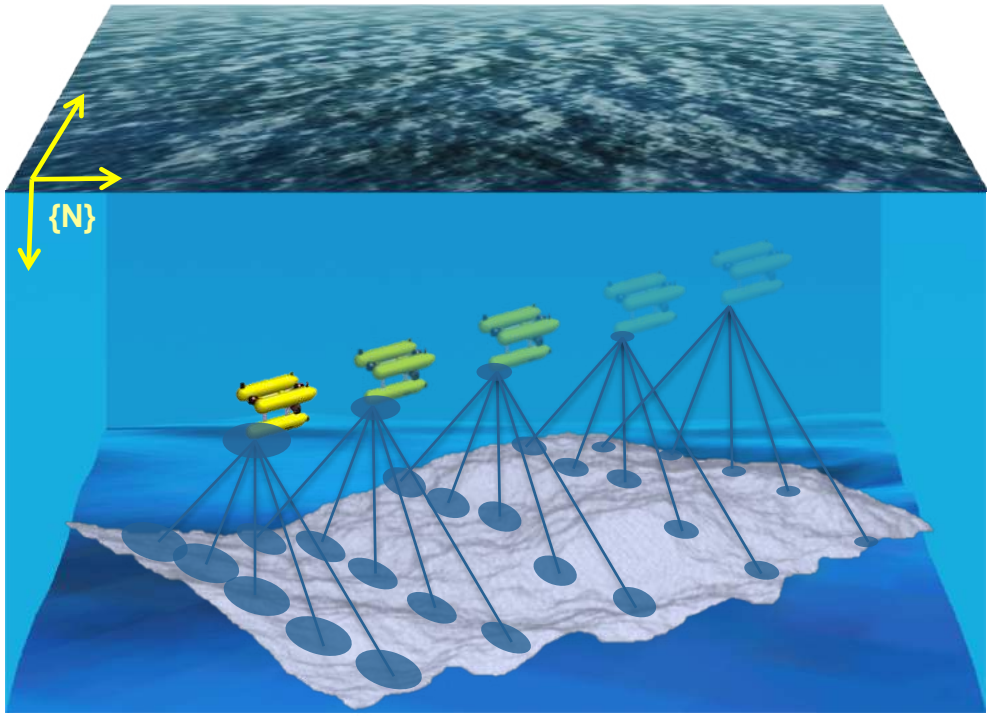
$$\boldsymbol{M} = \{\boldsymbol{m_1} \dots \boldsymbol{m_n}\} \;;\; \boldsymbol{m_i} = \{^{B_1}\boldsymbol{\delta_1} \dots ^{B_m}\boldsymbol{\delta_m}\}$$

$$^{B_i}\boldsymbol{\delta_i} = [\rho_i \;\; \theta_i]^T = \mathcal{N}(^{B_i}\hat{\boldsymbol{\delta}}_{\boldsymbol{i}}, ^{N}\boldsymbol{P}_{\boldsymbol{\delta_i}})$$

> Point Cloud Characterization

$$\boldsymbol{S} = [\quad ^{N}\boldsymbol{O} \quad]$$

GetScan()



> Store the Incremental Trajectory.

$$^{N}\boldsymbol{o} = \{^{N}\boldsymbol{o_1} \dots ^{N}\boldsymbol{o_n}\} \;;\; ^{N}\boldsymbol{o_i} = \mathcal{N}(^{N}\hat{\boldsymbol{o}}_{\boldsymbol{i}}, ^{N}\boldsymbol{P}_{o_i})$$

> Store polar MB profiles.

$$\boldsymbol{M} = \{\boldsymbol{m_1} \dots \boldsymbol{m_n}\} \;;\; \boldsymbol{m_i} = \{^{B_1}\boldsymbol{\delta_1} \dots ^{B_m}\boldsymbol{\delta_m}\}$$

$$^{B_i}\boldsymbol{\delta_i} = [\rho_i \;\; \theta_i]^T = \mathcal{N}(^{B_i}\hat{\boldsymbol{\delta}}_{\boldsymbol{i}}, ^{N}\boldsymbol{P}_{\boldsymbol{\delta_i}})$$

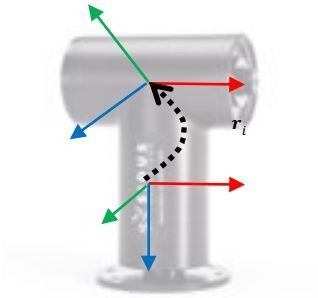
> Store Pan&Tilt Transformation.

$$\boldsymbol{R} = \{^{B_i}\boldsymbol{r_1} \dots ^{B_i}\boldsymbol{r_n}\} \;;\; ^{B_i}\boldsymbol{r_i} = [^{B_i}\phi_i \;\; ^{B_i}\theta_i \;\; ^{B_i}\psi_i]^T$$

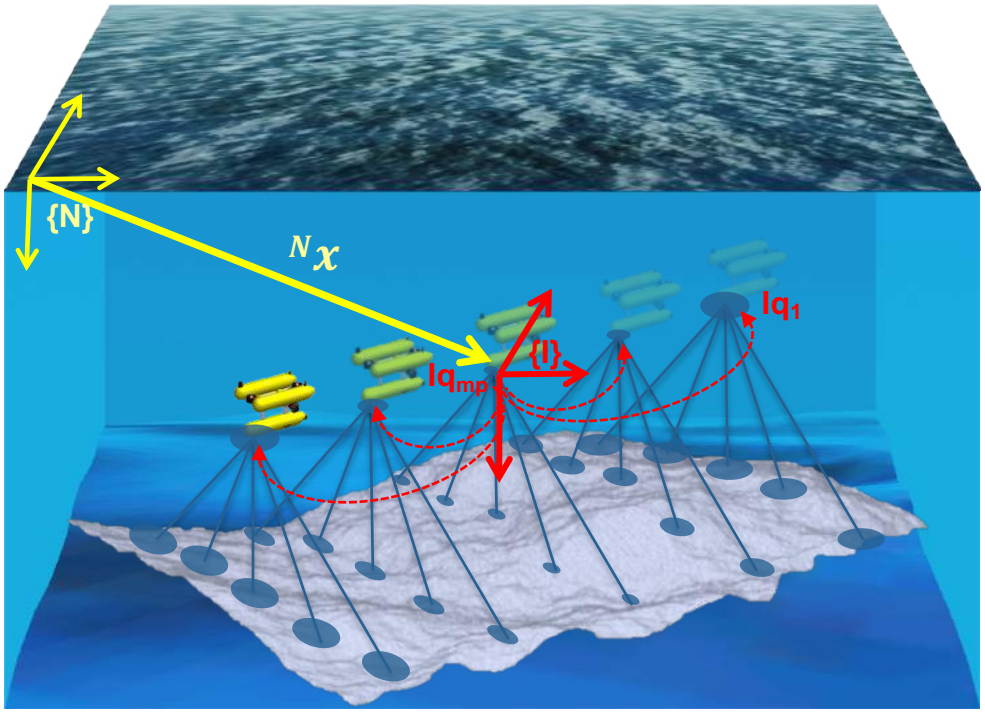
$$^{B_i}\boldsymbol{r_i} = \mathcal{N}(^{B_i}\hat{\boldsymbol{r}}_{\boldsymbol{i}}, ^{B_i}\boldsymbol{P}_{\boldsymbol{r_i}})$$

> Point Cloud Characterization

$$\boldsymbol{S} = [\quad ^{N}\boldsymbol{O} \quad]$$

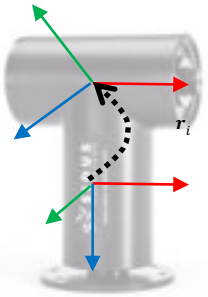


GetScan()



> Point Cloud Characterization

$$S = [\,^N x \quad ^N O \quad ^I W \quad]$$



> Store the Incremental Trajectory.

$$^N o = \{^N o_1 \dots ^N o_n\} \;;\; ^N o_i = \mathcal{N}(^N \hat{o}_i, ^N P_{o_i})$$

> Store polar MB profiles.

$$M = \{m_1 \dots m_n\} \;;\; m_i = \{^{B_1}\delta_1 \dots ^{B_m}\delta_m\}$$
$$^{B_i}\delta_i = [\rho_i \;\; \theta_i]^T = \mathcal{N}(^{B_i}\hat{\delta}_i, ^N P_{\delta_i})$$

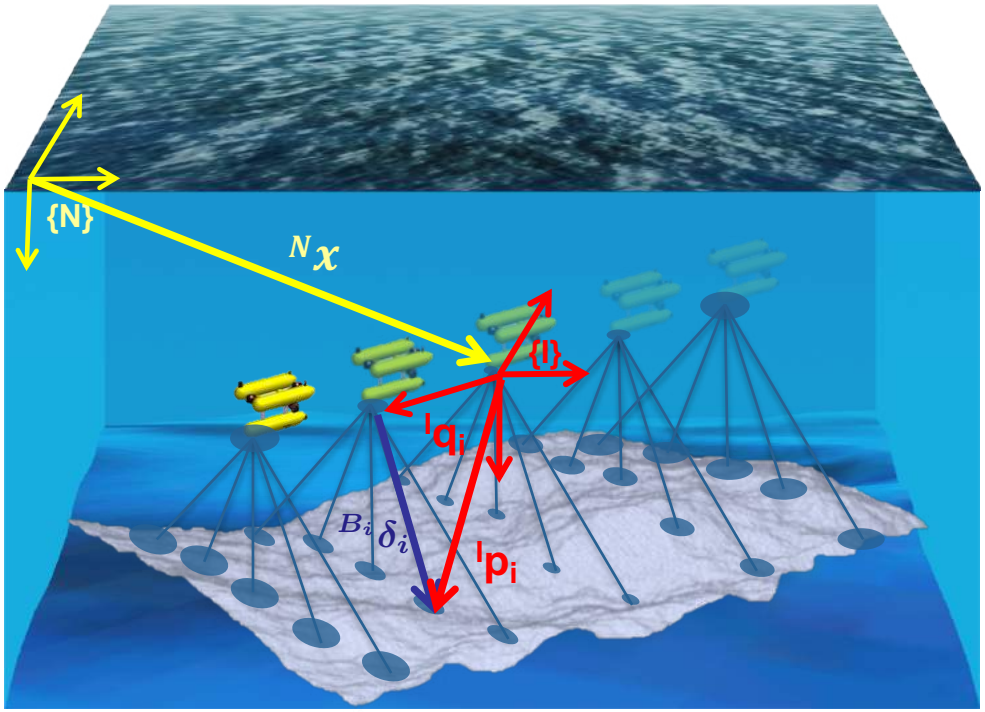
> Store Pan&Tilt Transformation.

$$R = \{^{B_i}r_1 \dots ^{B_i}r_n\} \;;\; ^{B_i}r_i = [^{B_i}\phi_i \;\; ^{B_i}\theta_i \;\; ^{B_i}\psi_i]^T$$
$$^{B_i}r_i = \mathcal{N}(^{B_i}\hat{r}_i, ^{B_i}P_{r_i})$$

> The scan is referenced to centroide to reduce the uncertainty

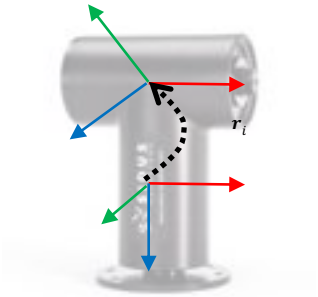
$$k > mp \Rightarrow ^I q_k = \begin{bmatrix} ^N q_{xy_{k-1}} + ^N o_{xy_k} \\ ^N o_{z_k} - ^N o_{mp_z} \\ o_{k_2} \end{bmatrix}$$
$$k = mp \Rightarrow ^I q_{mp} = [0_{3\times 1} \;\; o_{mp_2}]$$
$$k < mp \Rightarrow ^I q_k = \begin{bmatrix} ^N q_{xy_{k-1}} - ^N o_{xy_k} \\ ^N o_{z_k} - ^N o_{mp_z} \\ o_{k_2} \end{bmatrix}$$

GetScan()



> Point Cloud Characterization

$$S = \begin{bmatrix} {}^{N}x & {}^{N}O & {}^{I}W \end{bmatrix}$$



> Store the Incremental Trajectory.

$${}^{N}o = \{{}^{N}o_1 \dots {}^{N}o_n\} \; ; \; {}^{N}o_i = \mathcal{N}({}^{N}\hat{o}_i, {}^{N}P_{o_i})$$

> Store polar MB profiles.

$$M = \{m_1 \dots m_n\} \; ; \; m_i = \{{}^{B_1}\delta_1 \dots {}^{B_m}\delta_m\}$$

$${}^{B_i}\delta_i = [\rho_i \; \theta_i]^T = \mathcal{N}({}^{B_i}\hat{\delta}_i, {}^{N}P_{\delta_i})$$

> Store Pan&Tilt Transformation.

$$R = \{{}^{B_i}r_1 \dots {}^{B_i}r_n\} \; ; \; {}^{B_i}r_i = [{}^{B_i}\phi_i \; {}^{B_i}\theta_i \; {}^{B_i}\psi_i]^T$$

$${}^{B_i}r_i = \mathcal{N}({}^{B_i}\hat{r}_i, {}^{B_i}P_{r_i})$$

> The scan is referenced to centroide to reduce the uncertainty

$$k > mp \Rightarrow {}^{I}q_k = \begin{bmatrix} {}^{N}q_{xy_{k-1}} + {}^{N}o_{xy_k} \\ {}^{N}o_{z_k} - {}^{N}o_{mp_z} \\ o_{k_2} \end{bmatrix}$$

$$k = mp \Rightarrow {}^{I}q_{mp} = [0_{3\times 1} \; o_{mp_2}]$$

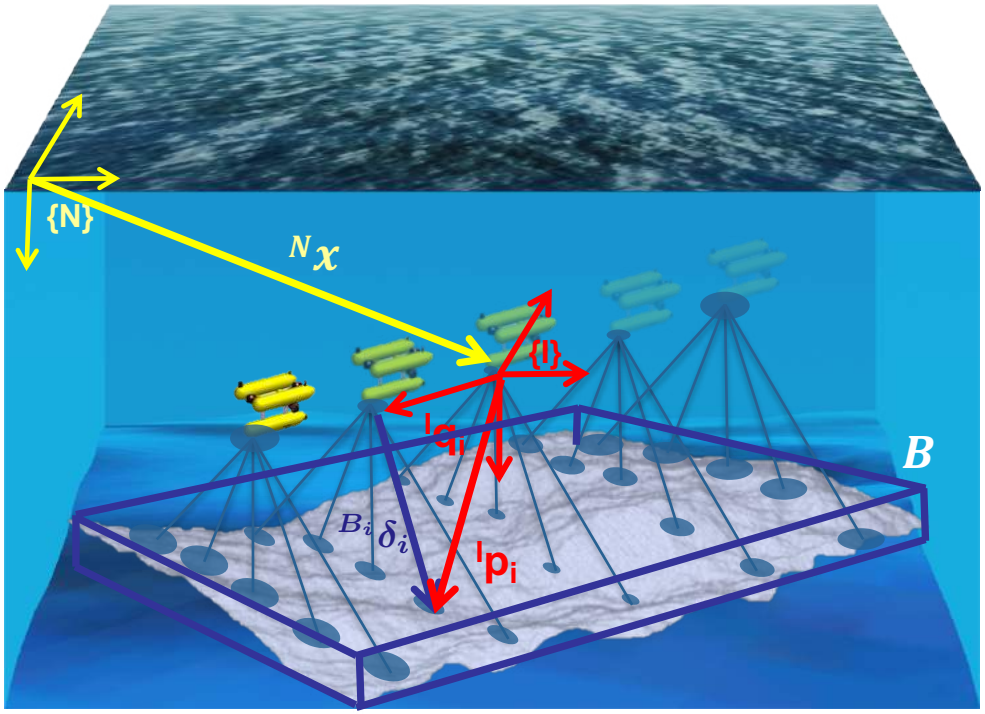
$$k < mp \Rightarrow {}^{I}q_k = \begin{bmatrix} {}^{N}q_{xy_{k-1}} - {}^{N}o_{xy_k} \\ {}^{N}o_{z_k} - {}^{N}o_{mp_z} \\ o_{k_2} \end{bmatrix}$$

> MB points are converted to Cartesian

$${}^{I}W = \{{}^{I}p_1 \dots {}^{I}p_n\} \; ; \; {}^{I}p_i = \mathcal{N}({}^{I}\hat{p}_i, {}^{I}P_{p_i})$$

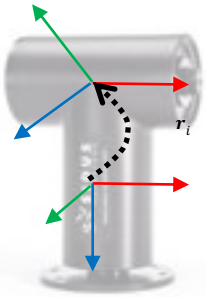
$${}^{I}p_i = {}^{I}q_i \oplus ({}^{B_i}r_i \oplus p2c({}^{B_i}\delta_i))$$

GetScan()



> Point Cloud Characterization

$$S = \begin{bmatrix} {}^{N}x & {}^{N}O & {}^{I}W & {}^{I}B \end{bmatrix}$$



> Store the Incremental Trajectory.

$${}^{N}o = \{{}^{N}o_1 \dots {}^{N}o_n\} \; ; \; {}^{N}o_i = \mathcal{N}({}^{N}\hat{o}_i, {}^{N}P_{o_i})$$

> Store polar MB profiles.

$$M = \{m_1 \dots m_n\} \; ; \; m_i = \{{}^{B_1}\delta_1 \dots {}^{B_m}\delta_m\}$$

$${}^{B_i}\delta_i = [\rho_i \; \theta_i]^T = \mathcal{N}({}^{B_i}\hat{\delta}_i, {}^{N}P_{\delta_i})$$

> Store Pan&Tilt Transformation.

$$R = \{{}^{B_i}r_1 \dots {}^{B_i}r_n\} \; ; \; {}^{B_i}r_i = [{}^{B_i}\phi_i \; {}^{B_i}\theta_i \; {}^{B_i}\psi_i]^T$$

$${}^{B_i}r_i = \mathcal{N}({}^{B_i}\hat{r}_i, {}^{B_i}P_{r_i})$$

> The scan is referenced to centroide to reduce the uncertainty

$$k > mp \Rightarrow {}^{I}q_k = \begin{bmatrix} {}^{N}q_{xy_{k-1}} + {}^{N}o_{xy_k} \\ {}^{N}o_{z_k} - {}^{N}o_{mp_z} \\ o_{k_2} \end{bmatrix}$$

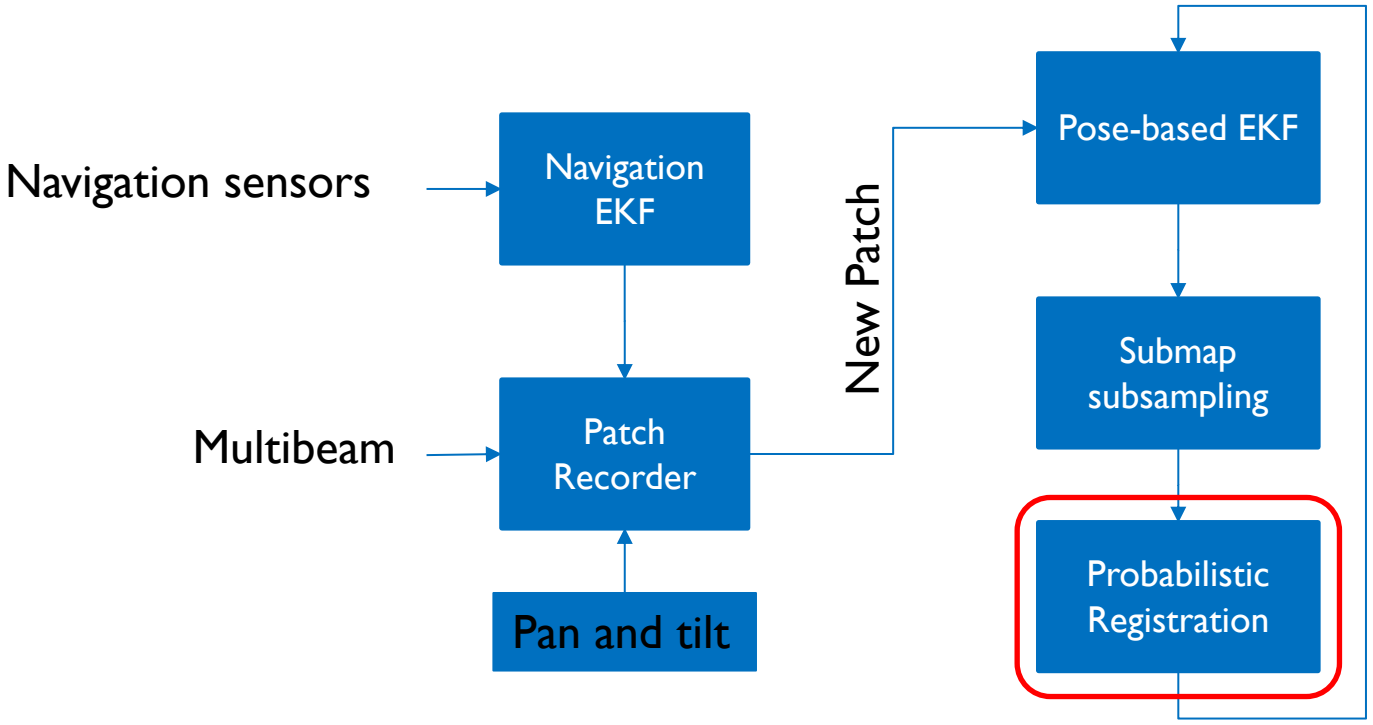
$$k = mp \Rightarrow {}^{I}q_{mp} = [0_{3\times 1} \; o_{mp_2}]$$

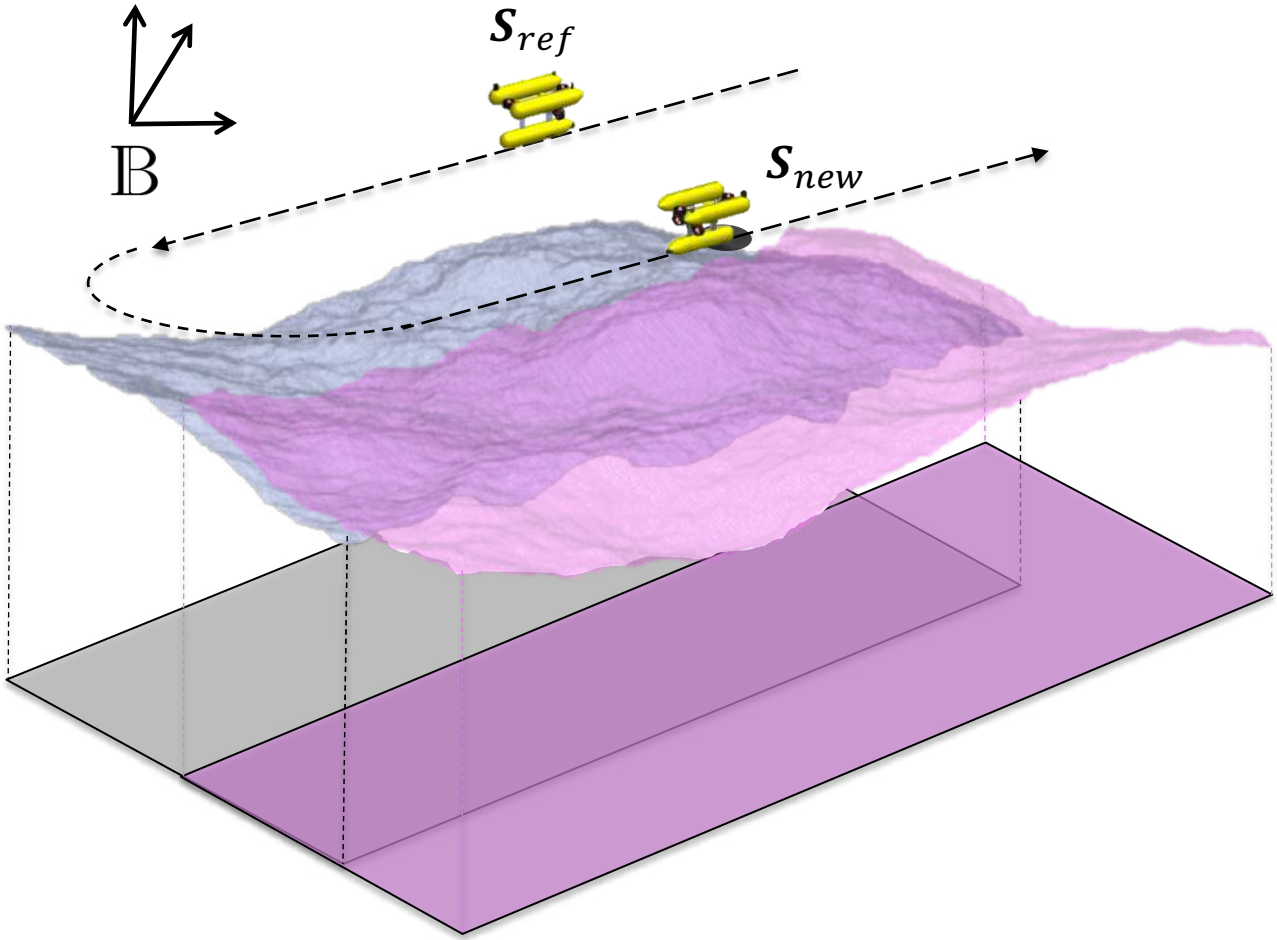
$$k < mp \Rightarrow {}^{I}q_k = \begin{bmatrix} {}^{N}q_{xy_{k-1}} - {}^{N}o_{xy_k} \\ {}^{N}o_{z_k} - {}^{N}o_{mp_z} \\ o_{k_2} \end{bmatrix}$$

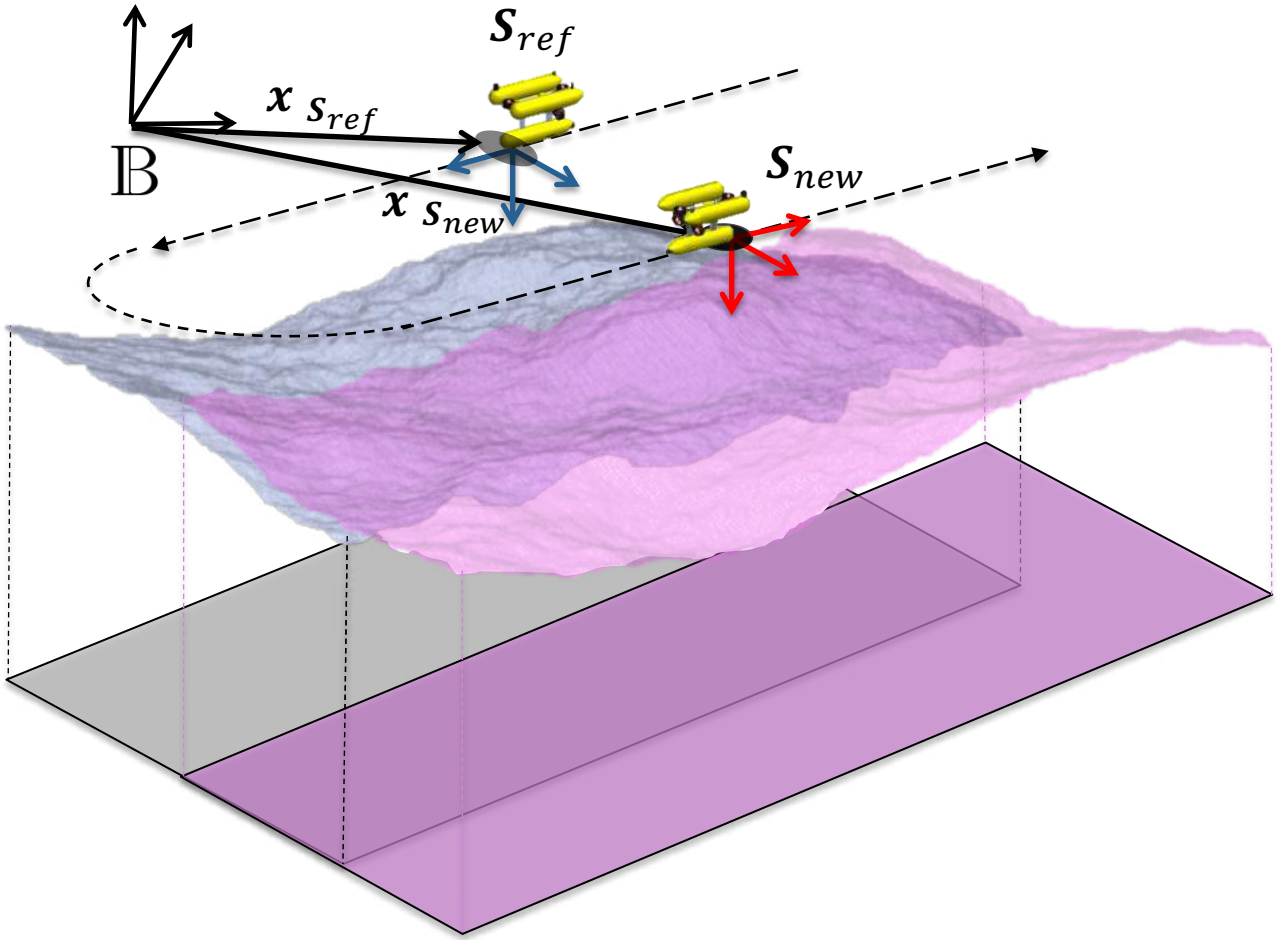
> MB points are converted to Cartesian

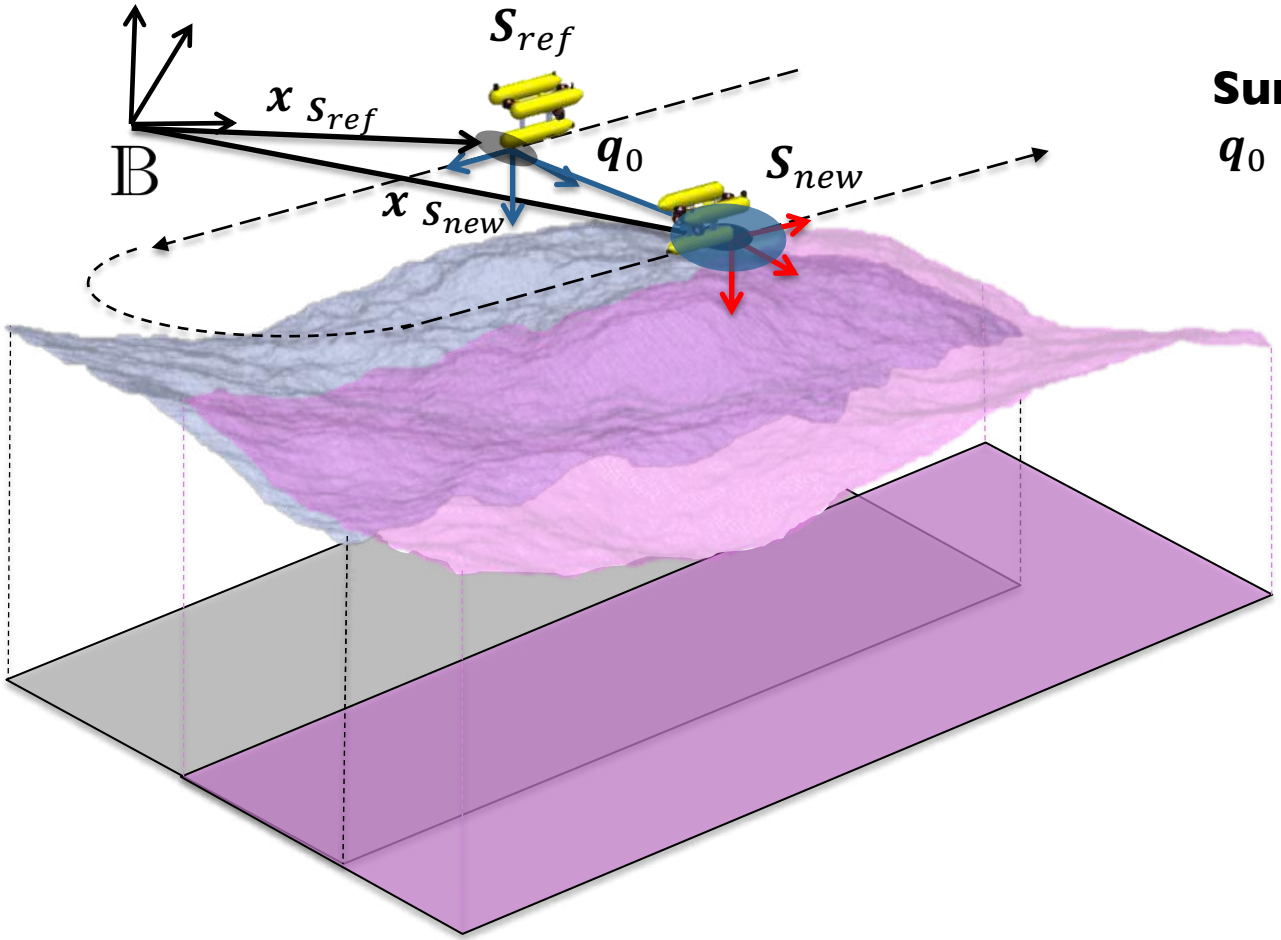
$${}^{I}W = \{{}^{I}p_1 \dots {}^{I}p_n\} \; ; \; {}^{I}p_i = \mathcal{N}({}^{I}\hat{p}_i, {}^{I}P_{p_i})$$

$${}^{I}p_i = {}^{I}q_i \oplus ({}^{B_i}r_i \oplus p2c({}^{B_i}\delta_i))$$


Navigation sensors
Navigation EKF
Multibeam
Patch Recorder
Pan and tilt
New Patch
Pose-based EKF
Submap subsampling
Probabilistic Registration

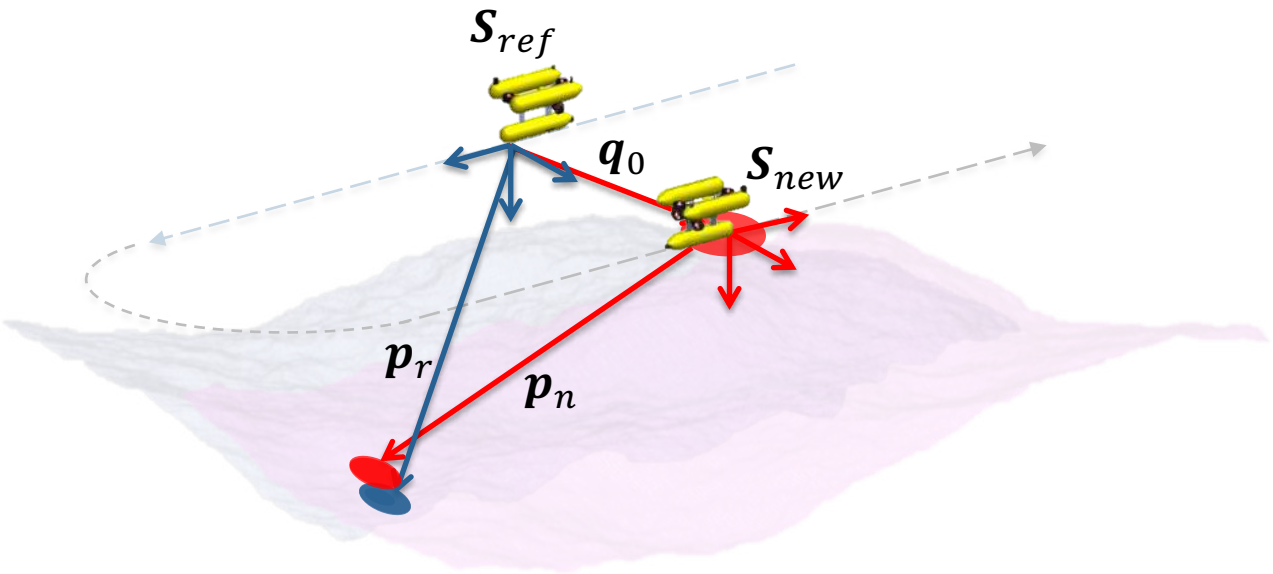

S_{ref}
S_{new}
$\mathbb{B}$


S_{ref}
$x_{S_{ref}}$
$\mathbb{B}$
$x_{S_{new}}$
S_{new}



Surfaces' Relative Pose

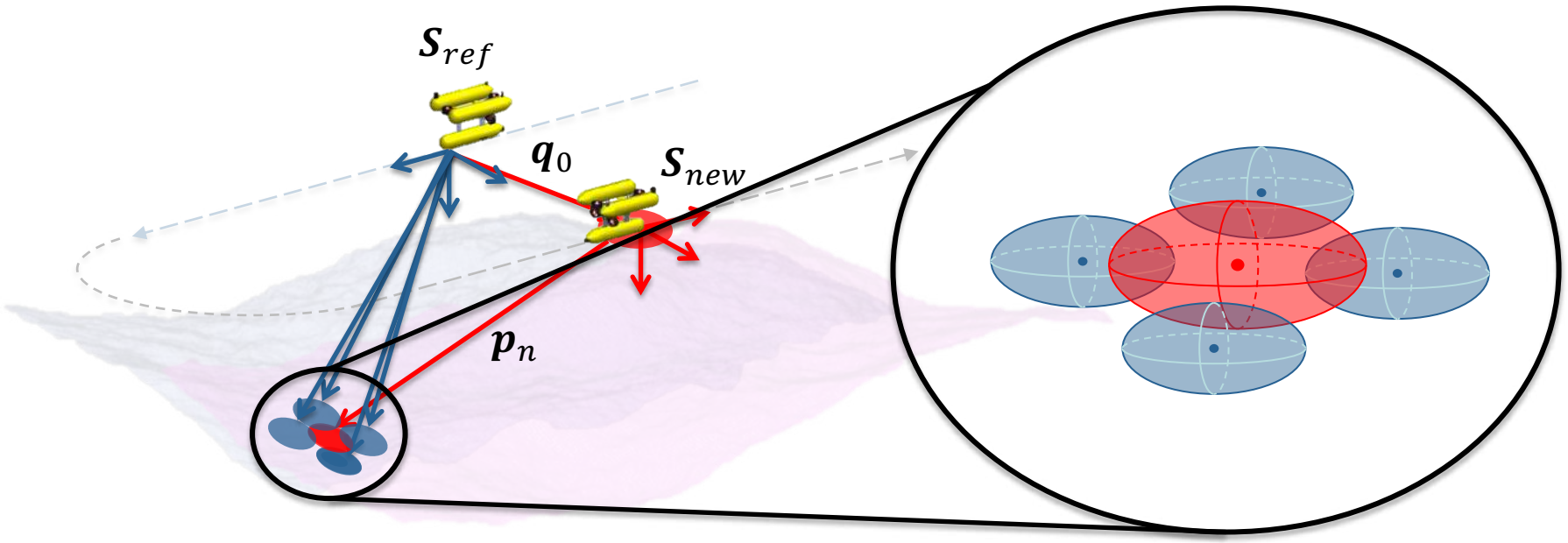
$$q_0 = \ominus x_{S_{ref}} \oplus x_{S_{new}}$$



$$\hat{\mathbf{e}}_{rn} = \hat{\mathbf{p}}_r - \hat{\mathbf{q}}_0 \oplus \hat{\mathbf{p}}_n$$

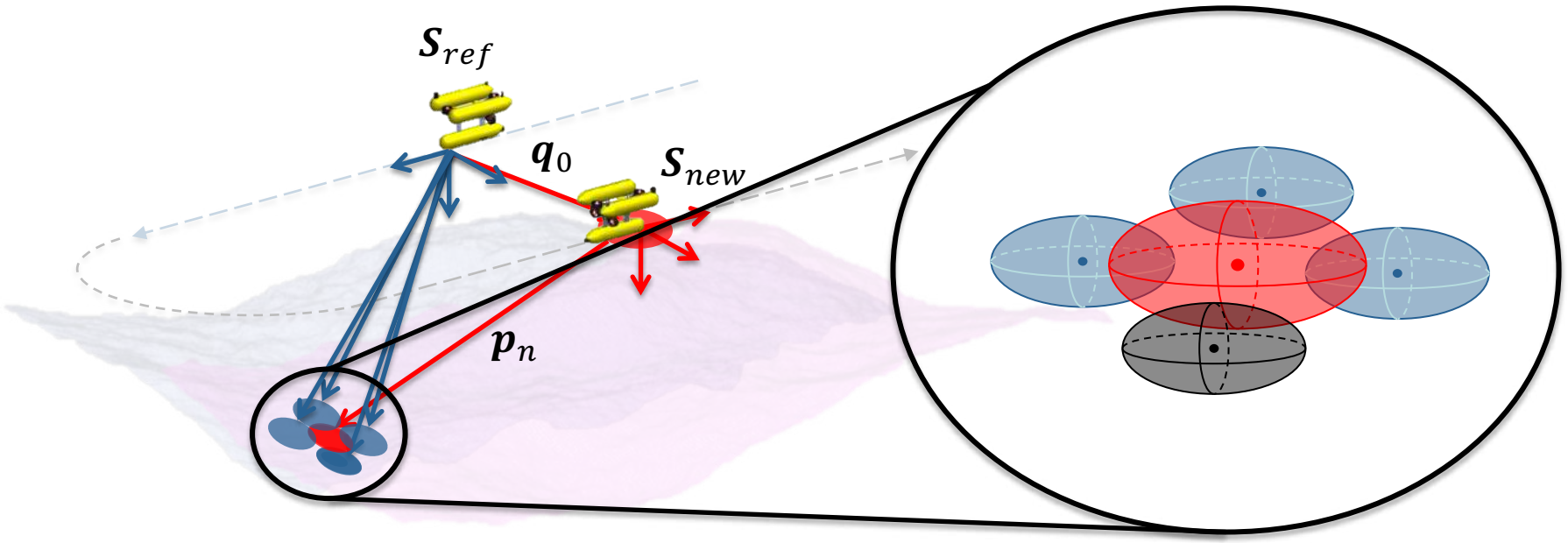
$$\mathbf{P}_{rn} = \mathbf{P}_r + \mathbf{J}_{1\oplus}\mathbf{P}_{q_0}\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{P}_n\mathbf{J}_{2\oplus}^T$$

If $D^2 = \hat{\mathbf{e}}_{rn}^T \mathbf{P}_{rn}^{-1} \hat{\mathbf{e}}_{rn} < \chi_{d,\alpha}^2$ ➡ **Points are Compatible**

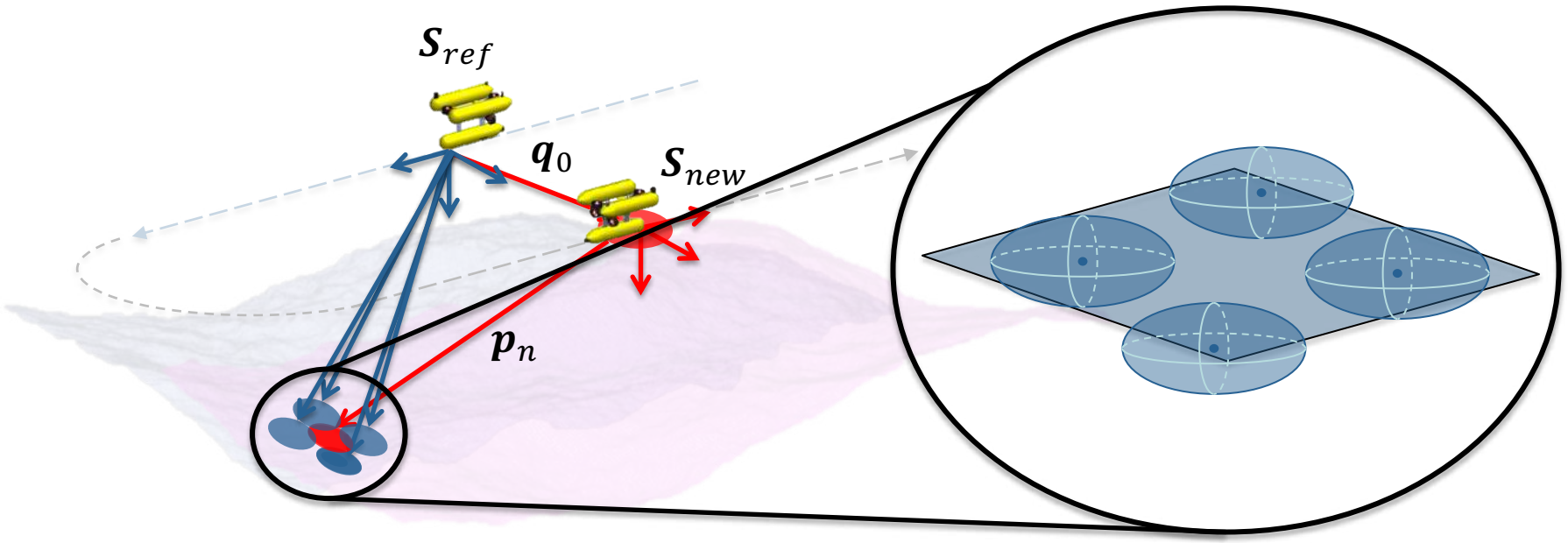


1st step ⇨ Point to point, minimum Mahalanobis association

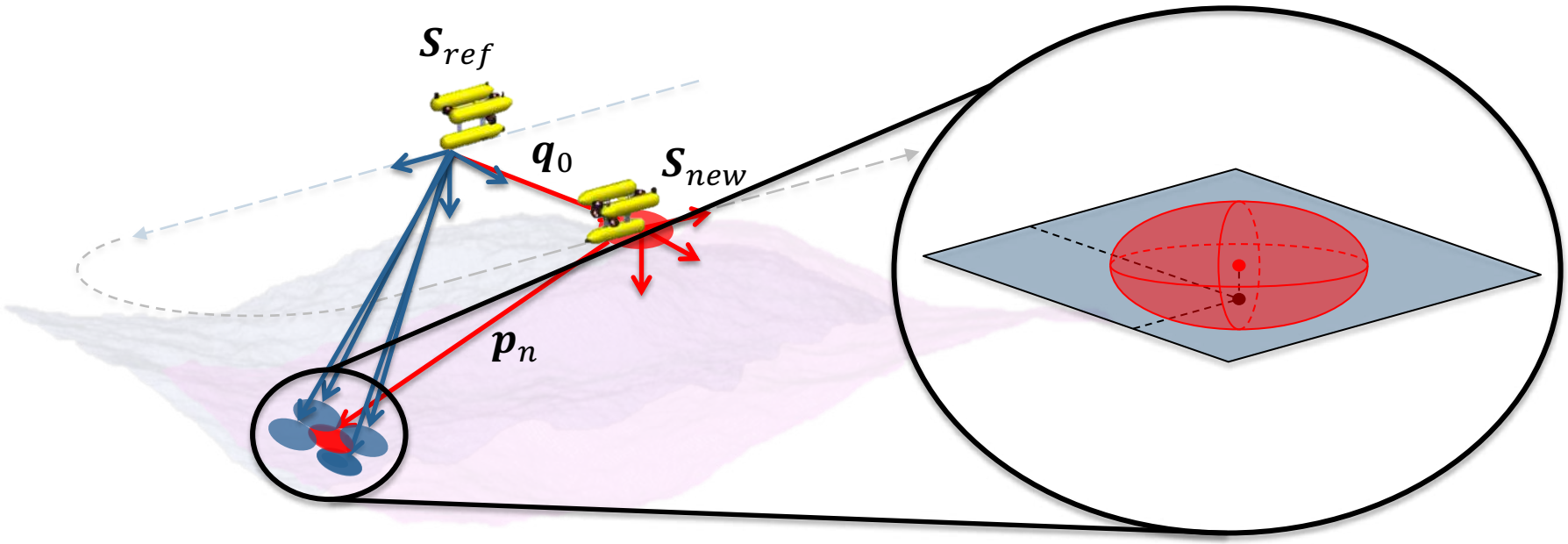
2nd step ⇨ Point to plane, all individually compatible points



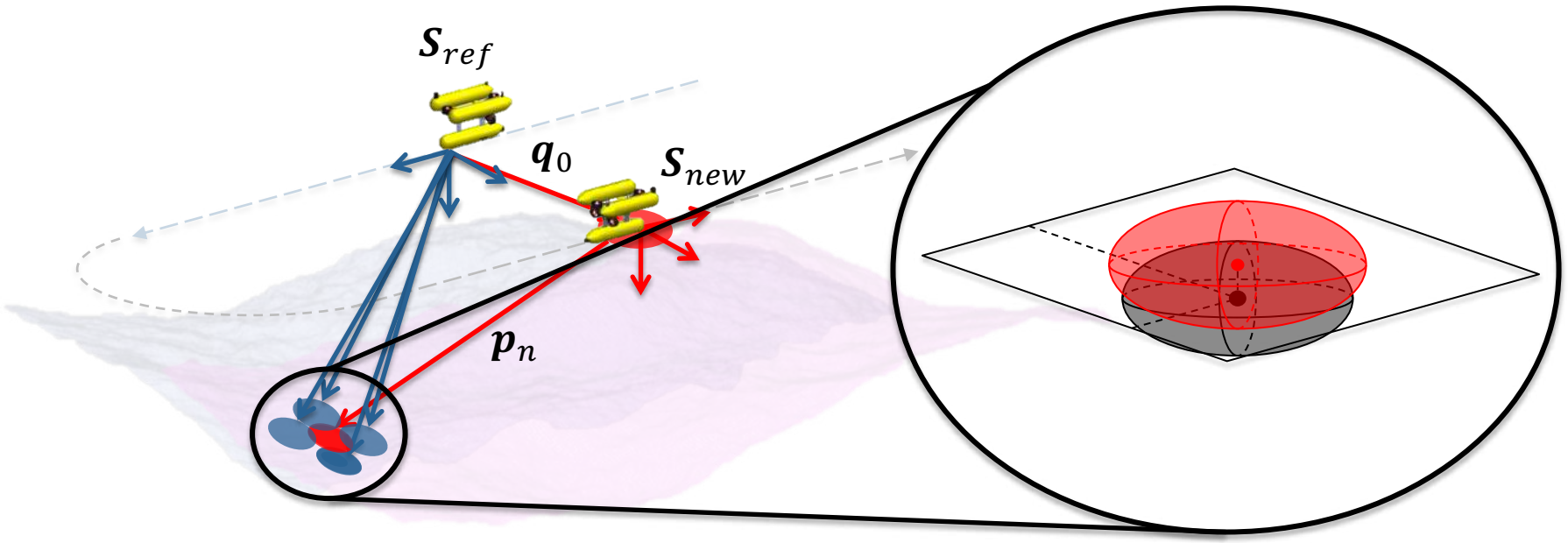
- The point with the minimum Mahalanobis distance is chosen for the point-to-point association.
- Correction of large displacements



- Next a plane fitting for all the candidate points is estimated.
- The plane projection of the point is computed and used for probabilistic ICP.

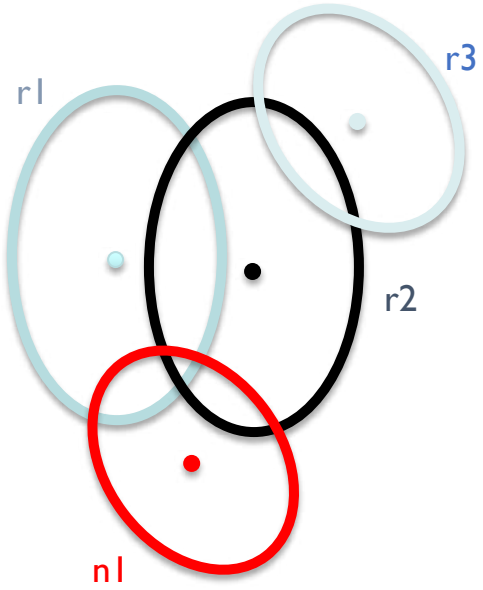


- Next a plane fitting for all the candidate points is estimated.
- The plane projection of the point is computed and used for probabilistic ICP.

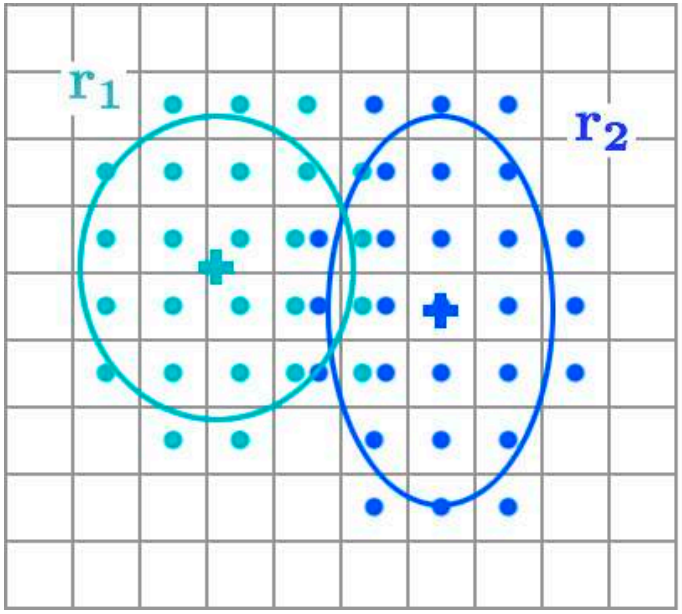


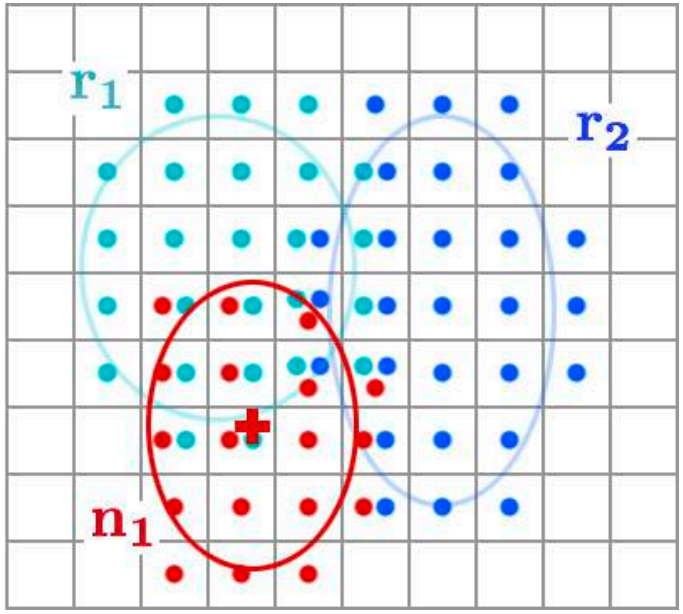
$$q_{\min} = \underset{q}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_i\left(\varepsilon^T P_\varepsilon^{-1}\varepsilon\right)\right\} \Rightarrow \text{Least squares}$$

- Being ε a vector of all $\hat{e}_{rn}$ and P_ε the block diagonal with their corresponding covariances

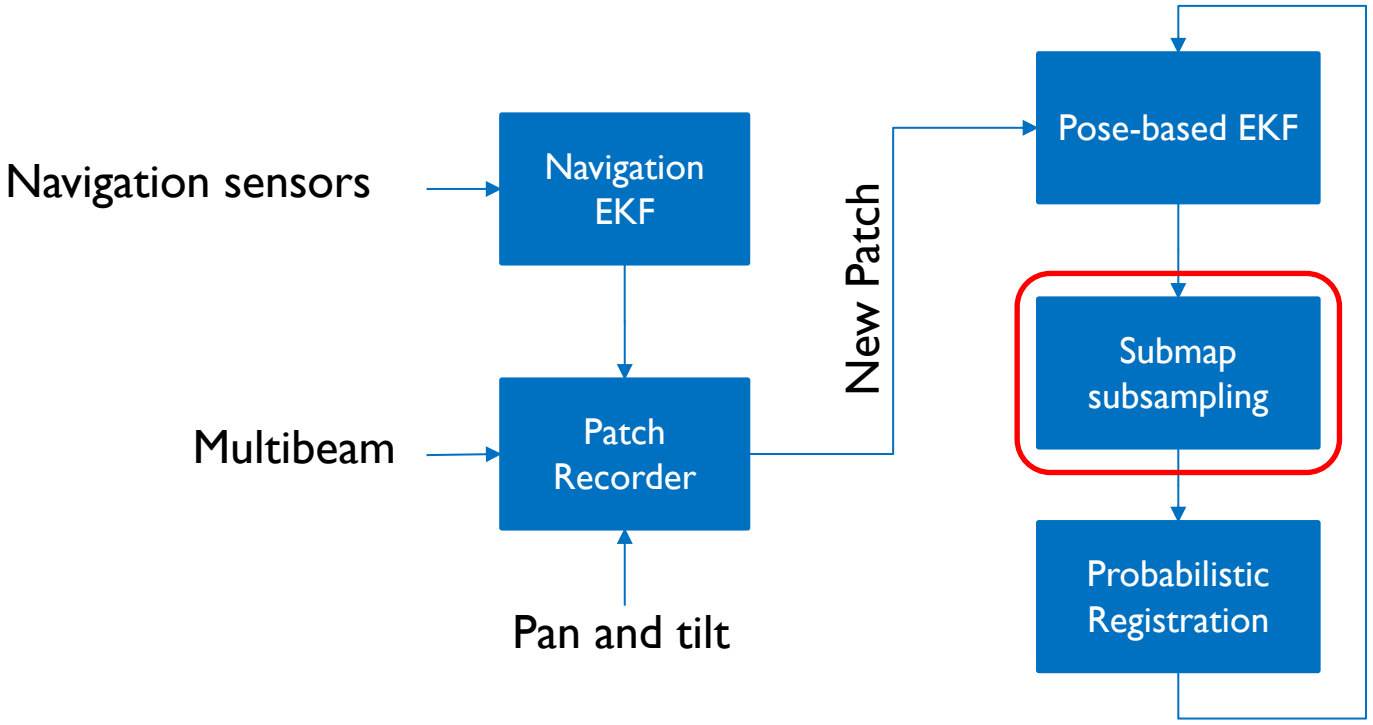


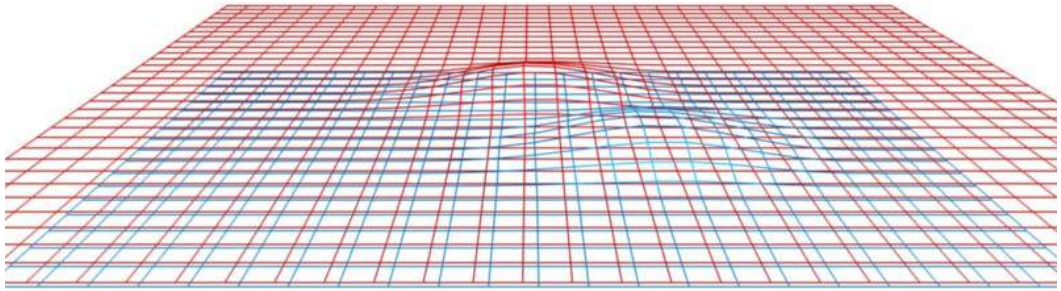
- Determining correspondences between points ↑ computational cost $O(n^2)$
- Marking in a grid the threshold uncertainty of all the points, only the ones sharing cells may be compatible → $O(n)$



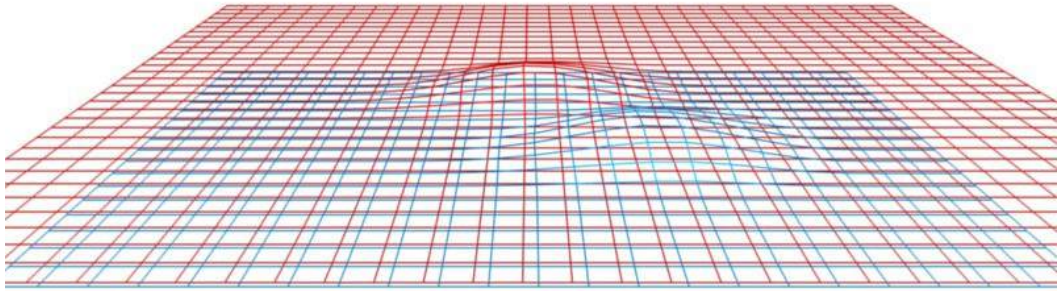


- The points (and its uncertainty) of the reference scan are rendered in a grid and each cell is marked with the points that occupy it.
- Each point of the new scan is rendered in the same grid. The compatibility test will be computed for the points sharing some cells.

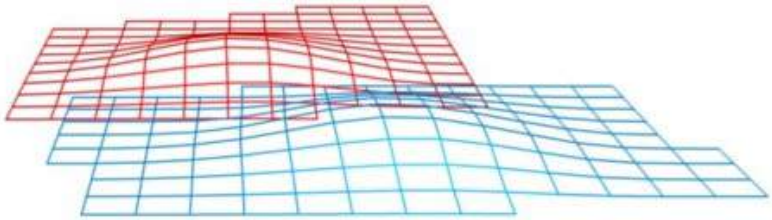

Navigation sensors
Navigation EKF
Multibeam
Patch Recorder
Pan and tilt
New Patch
Pose-based EKF
Submap subsampling
Probabilistic Registration

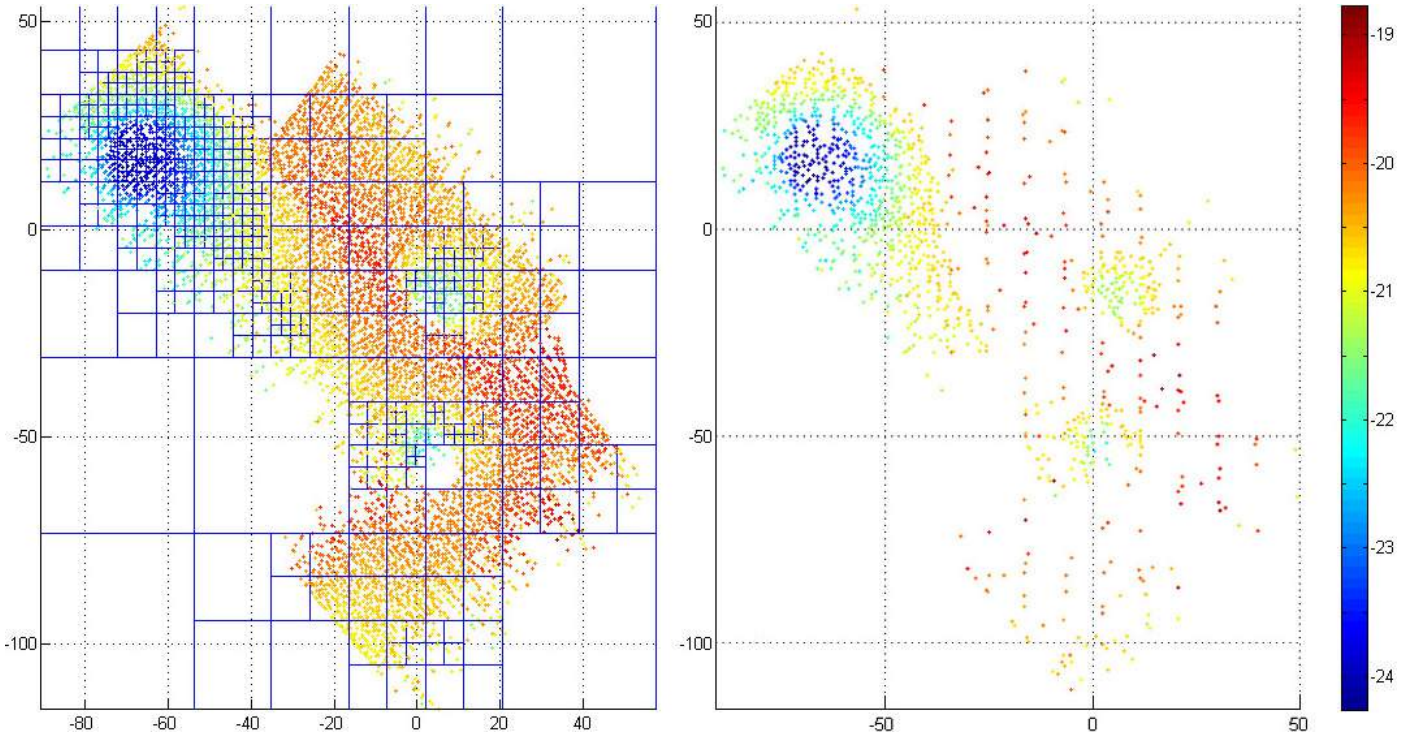


- ICP tends to associate each point with its neighbour (according to a certain metric) → Features not relevant within the neighbourhood.
- Flat areas may lead to poor matchings when prevailing over the features.



- ICP tends to associate each point with its neighbour (according to a certain metric) → Features not relevant within the neighbourhood.
- Flat areas may lead to poor matches when prevailing over the features.



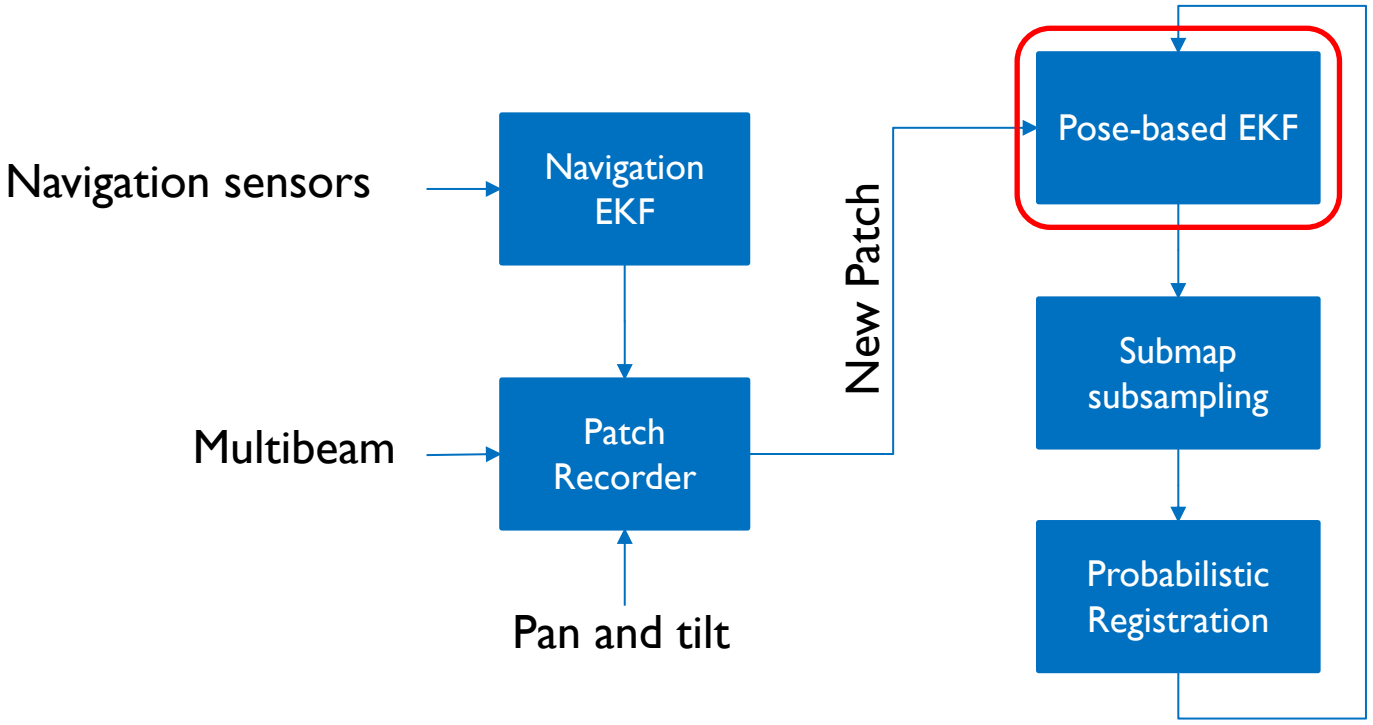


Octree structure + heuristic

- Distance from principal plane
- Distance from mean
- PCA eigenvalues
- Curvature
- PCA eigenvector
- Difference of normals

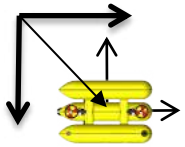
EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

  if *ScanAvailable* then

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

    $\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

      $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

      $[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

      $i = i + 1$; // Go to next pair

    end

  end

  $\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

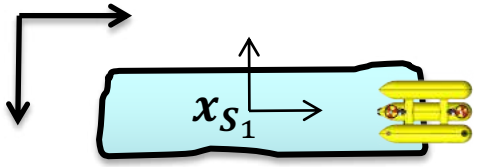
  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

  $[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

  $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

  if *ScanAvailable* then

    $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

    $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

    $\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

    $\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

    for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

      $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

      ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

      ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

      $[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

      $i = i + 1$; // Go to next pair

    end

  end

  $\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

  $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

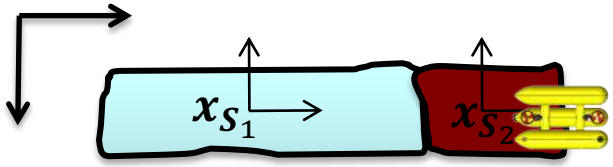
  $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

$\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

$j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

$[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

$i = i + 1$; // Go to next pair

end

end

$\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

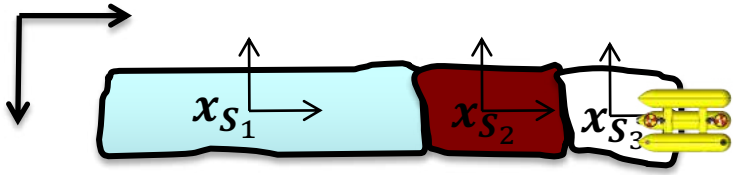
$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

$\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

$j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

$[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

$i = i + 1$; // Go to next pair

end

end

$\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

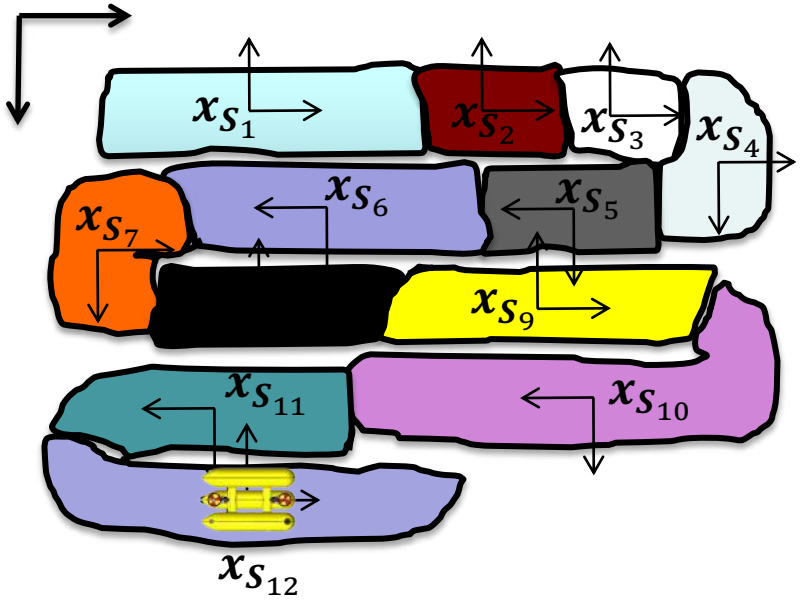
$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

$\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

$j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

$[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

$i = i + 1$; // Go to next pair

end

end

$\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

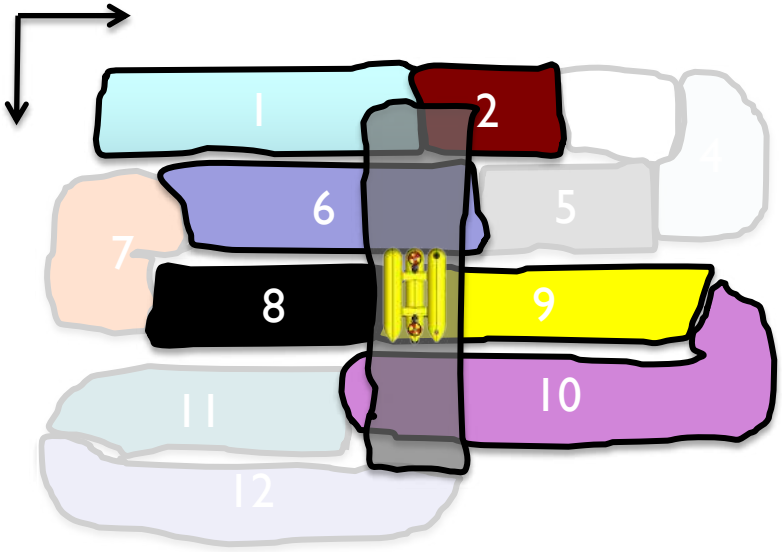
$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



Patches overlapping 13

1
2
6
8
9
10

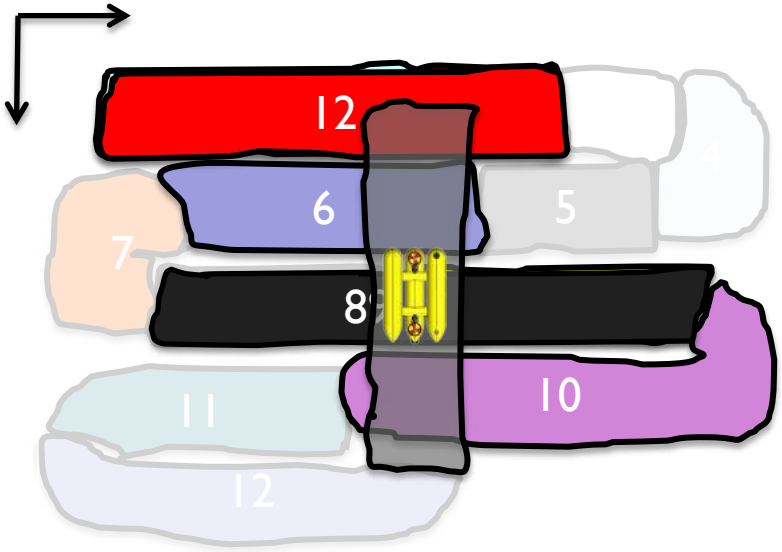
Method Localization$(\hat{x}_0, P_0)$

- $[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector
- $\mathcal{M} = []$; // INITIALIZE the map
- for $k = 1$ *to steps* do
  - $[u_k, Q_k] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = Prediction({}^N\hat{x}_{k-1}, {}^NP_{k-1}, u_k, Q_k)$;
  - $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors
  - if *ScanAvailable* then
    - $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;
    - $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map
    - $\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps
      - $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map
      - ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess
      - $[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
      - $i = i + 1$; // Go to next pair
    - end
  - end
  - $\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations
  - $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;
  - $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;
- end

end

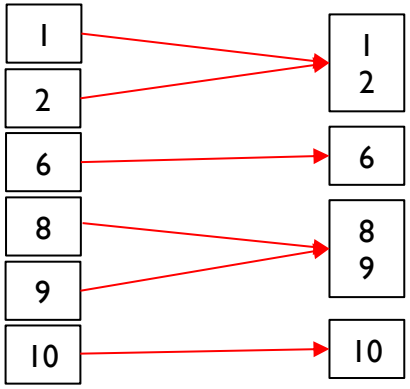
EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



Patches overlapping 13

Submaps fused for registration



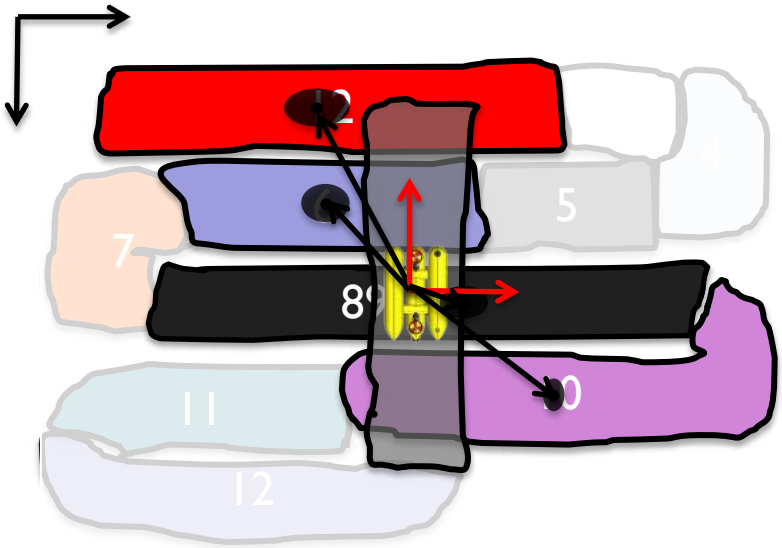
Method Localization$(\hat{x}_0, P_0)$

- $[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector
- $\mathcal{M} = []$; // INITIALIZE the map
- for $k = 1$ *to steps* do
  - $[u_k, Q_k] = GetInput()$; // Get input to the motion model
  - $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = Prediction({}^N\hat{x}_{k-1}, {}^NP_{k-1}, u_k, Q_k)$;
  - $[z_m, R_m] = GetMeasurement()$; // Read navigation sensors
  - if *ScanAvailable* then
    - $[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;
    - $[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector
    - $\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map
    - $\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans
    - for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps
      - $j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$
      - ${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map
      - ${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess
      - $[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov
      - $i = i + 1$; // Go to next pair
    - end
  - end
  - $\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations
  - $[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;
  - $[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;
- end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

$\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

$j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

$[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

$i = i + 1$; // Go to next pair

end

end

$\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

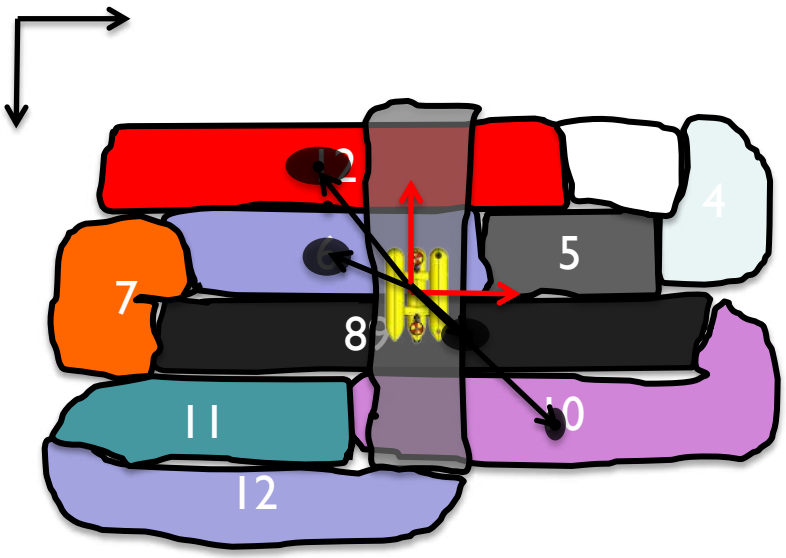
$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM



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

$[{}^N\hat{x}_0, {}^NP_0] = [{}^N\hat{x}_{B_0}, {}^NP_{B_0}]$; // Initialize SLAM state vector

$\mathcal{M} = []$; // INITIALIZE the map

for $k = 1$ *to steps* do

$[u_k, Q_k] = GetInput()$; // Get input to the motion model

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

$[z_m, R_m] = GetMeasurement()$; // Read navigation sensors

if *ScanAvailable* then

$[{}^{B_k}S_k, {}^{B_k}R_{S_k}] = GetScan()$;

$[{}^N\hat{\bar{x}}_k, {}^N\bar{P}_k] = AddNewPose({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k)$; // Grow the state vector

$\mathcal{M}[k] = {}^{B_k}S_k$; // Store the scan in the map

$\mathcal{H}_p = OverlappingScans({}^N\hat{\bar{x}}_k, \mathcal{M})$; // Get pairs of overlapping scans

for $i = 1$ *to* $length(\mathcal{H}_p)$ do // for all overlaps

$j = \mathcal{H}_p[i]$; // Get the pair: ${}^{B_j}S_j$ overlaps ${}^{B_k}S_k$

${}^{B_j}S_j = \mathcal{M}[j]$; ${}^{B_k}S_k = \mathcal{M}[k]$; // Get the scans from the map

${}^{B_j}x_{B_k} = (\ominus {}^Nx_{B_j}) \oplus {}^Nx_{B_k}$; // Scans Displacement guess

$[z_{p_i}, R_{p_i}] = Register({}^{B_j}S_j, {}^{B_k}S_k, {}^{B_j}x_{B_k})$; // Scan displacement mean & cov

$i = i + 1$; // Go to next pair

end

end

$\mathcal{H}_p = DataAssociation({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_p, R_p)$; // select compatible registrations

$[z_k, R_k, H_k, V_k] = ObservationMatrix(\mathcal{H}_p, {}^N\hat{\bar{x}}_k, z_m, R_m, z_p, R_p)$;

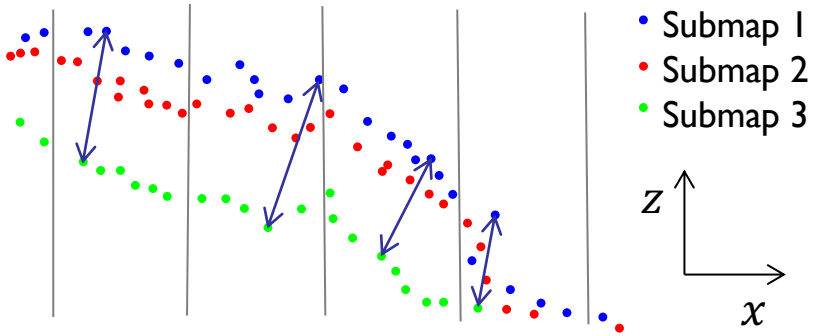
$[{}^N\hat{x}_k, {}^NP_k] = Update({}^N\hat{\bar{x}}_k, {}^N\bar{P}_k, z_k, R_k, H_k, V_k, \mathcal{H}_p)$;

end

end

Error assessment

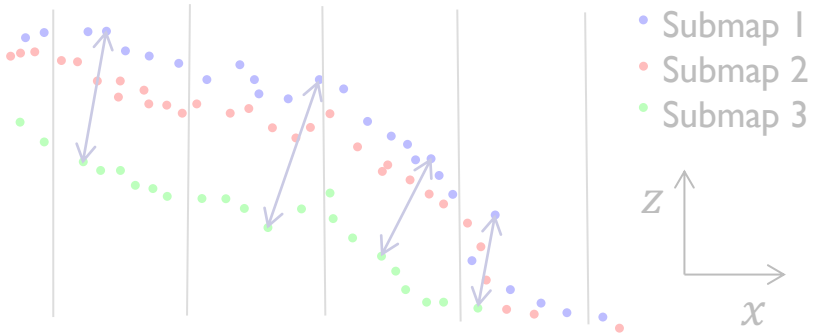
Consistency-based error evaluation (CBEE)



Thickness of the submap

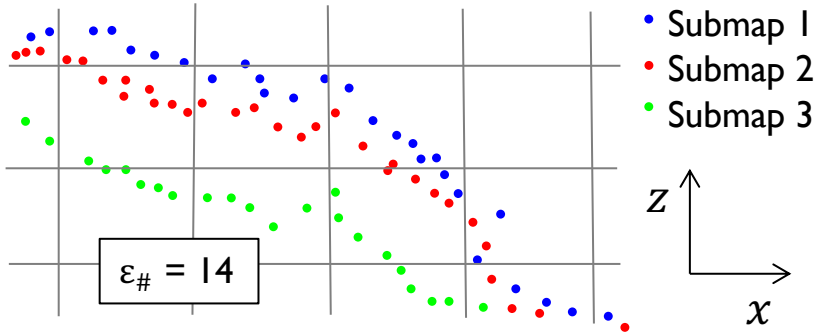
Error assessment

Consistency-based error evaluation (CBEE)



Thickness of the submap

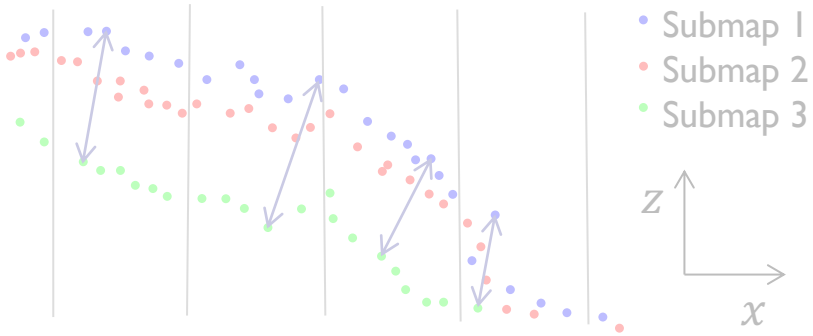
Number of cells (#Cells)



Total number of 3D cells of the final map

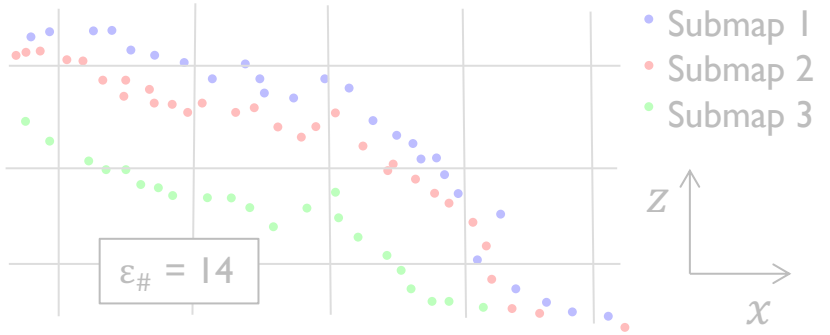
Error assessment

Consistency-based error evaluation (CBEE)



Thickness of the submap

Number of cells (#Cells)



Total number of 3D cells of the final map

Statistics

- Error reduction:

$$\varepsilon^{\downarrow} = \frac{\varepsilon_{\text{dead reckoning}} - \varepsilon_{\text{SLAM}}}{\varepsilon_{\text{dead reckoning}}}$$

- Mean error (2.5D):

$$\varepsilon_{\mu} = \frac{1}{n}\sum_{\forall i} \varepsilon_i$$

- Total error (2.5D):

$$\varepsilon_{\Sigma} = \sum_{\forall i} \varepsilon_i$$

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Tasmania Dataset (2.5D)

- SIRIUS AUV (ACFR)
- Survey area of 350 x 150 m
- Lawn Mower type survey
- Multiple loop closures

ACFR
AUSTRALIAN CENTRE FOR FIELD ROBOTICS

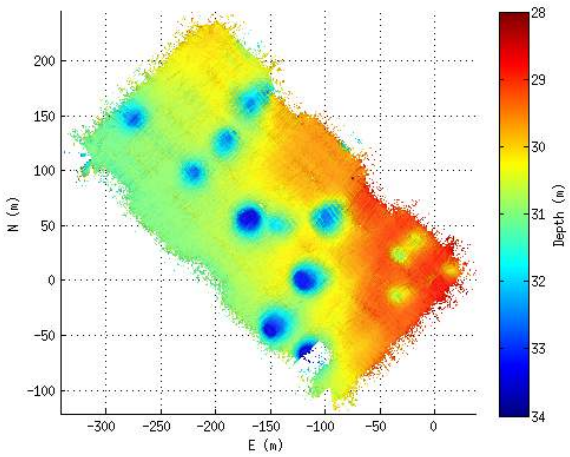
This dataset is a courtesy of the ACFR [Barkby et al., 2011]

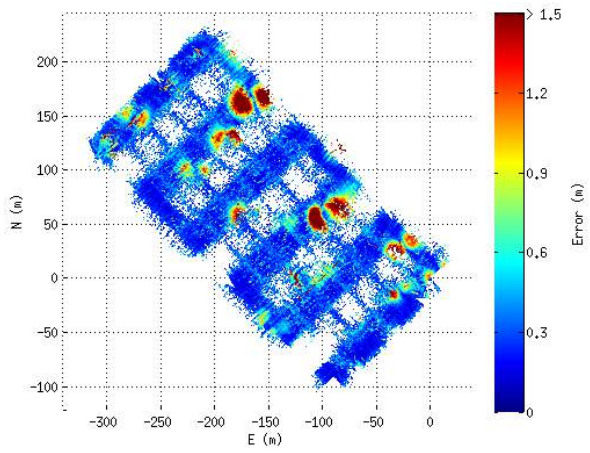
EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Tasmania Dataset (2.5D)

Dead reckoning





CBEE

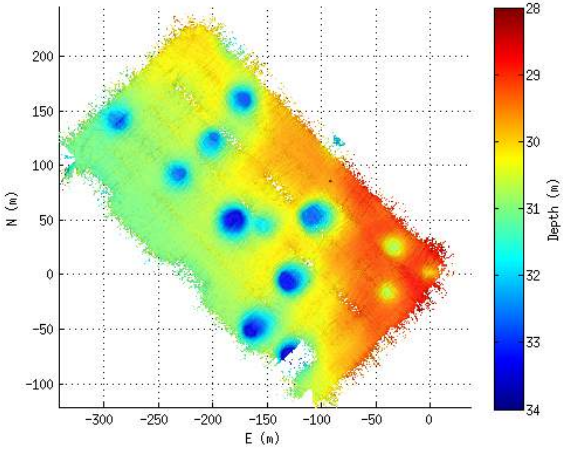
$\varepsilon_\Sigma = 70986.2$

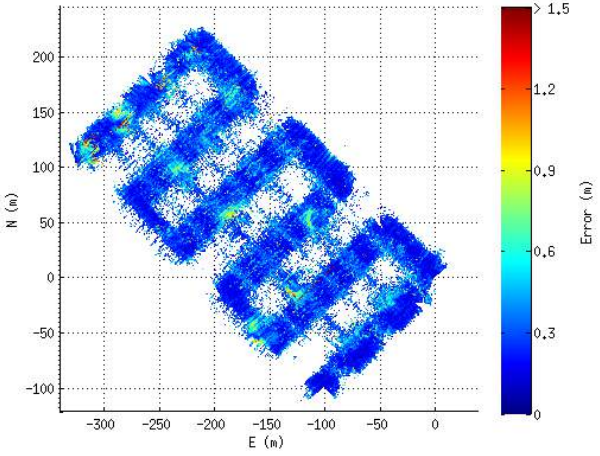
$\varepsilon_\mu = 0.3988$

#Cells

$\varepsilon_\# = 373121$

SLAM





CBEE

$\varepsilon_\Sigma = 57521.8$

$\varepsilon_\Sigma^{\downarrow} = 18.97\%$

$\varepsilon_\mu = 0.3223$

$\varepsilon_\mu^{\downarrow} = 19.2\%$

#Cells

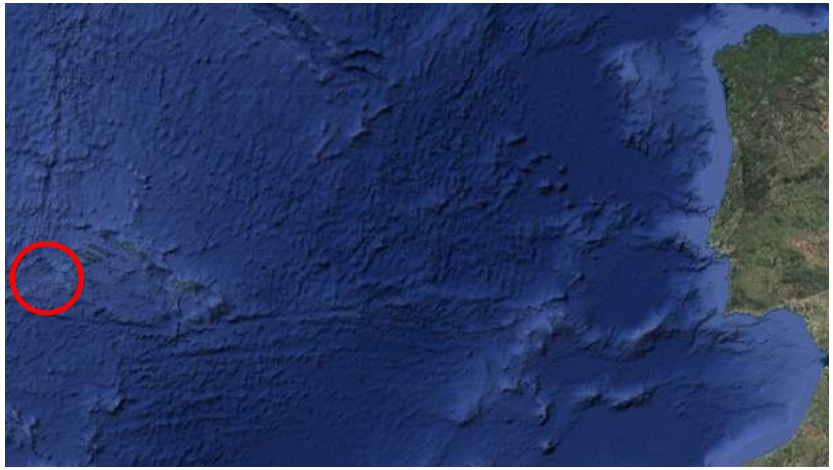
$\varepsilon_\# = 36{,}5014$

$\varepsilon_\#^{\downarrow} = 2.17\%$

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Porto Pim Dataset - I (2.5D)



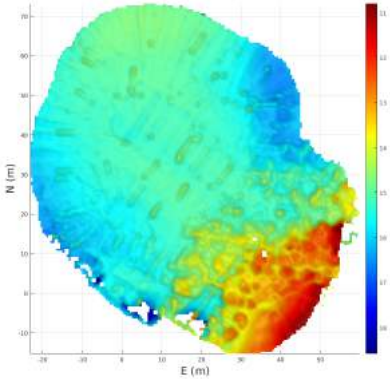
- Girona 500 (UdG)
- Three experiments
- Circular type surveys
- Multiple loop closures

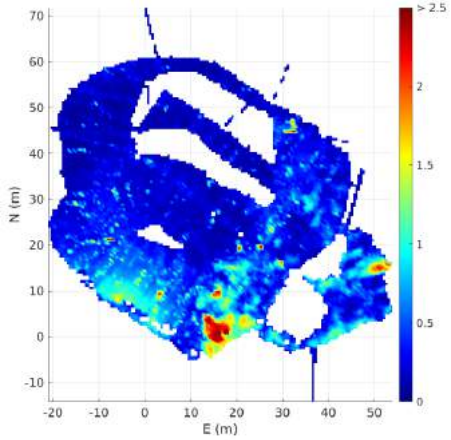
EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Porto Pim Dataset - I (2.5D)

Dead reckoning





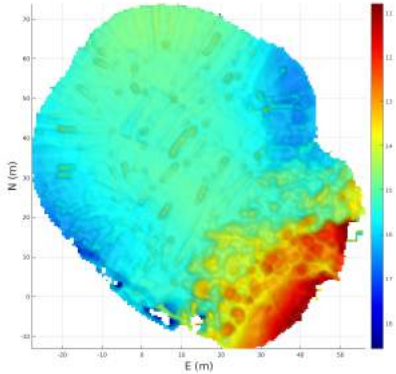
CBEE

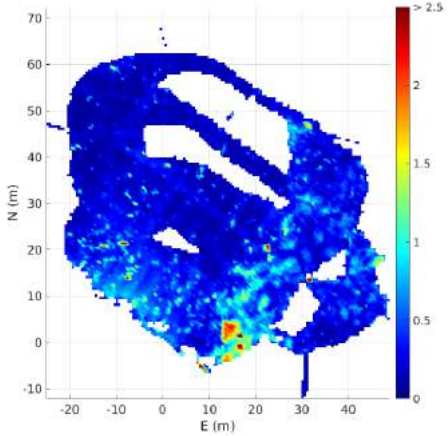
$\varepsilon_\Sigma = 4886.67$

#Cells

$\varepsilon_\# = 51319$

SLAM





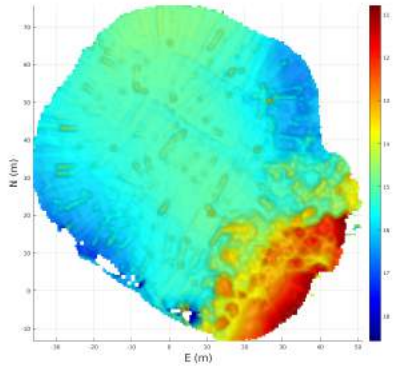
CBEE

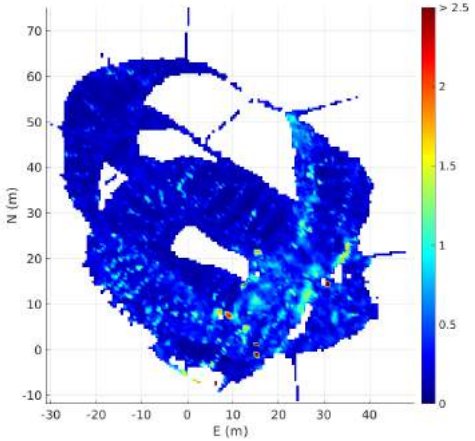
$\varepsilon_\Sigma = 4274.04, \varepsilon_\Sigma^{\downarrow} = 14.29\%$

#Cells

$\varepsilon_\# = 49491, \varepsilon_\#^{\downarrow} = 3.56\%$

USBL





CBEE

$\varepsilon_\Sigma = 3523.93, \varepsilon_\Sigma^{\downarrow} = 29.33\%$

#Cells

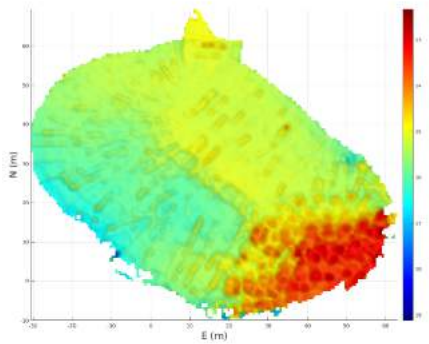
$\varepsilon_\# = 49914, \varepsilon_\#^{\downarrow} = 2.74\%$

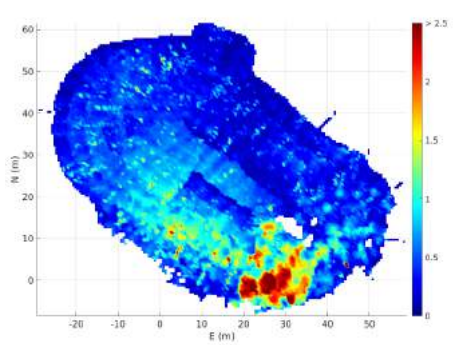
EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Porto Pim Dataset - I (2.5D)

Dead reckoning





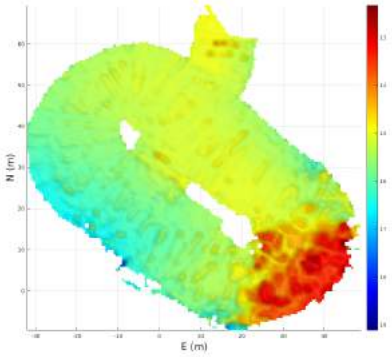
CBEE

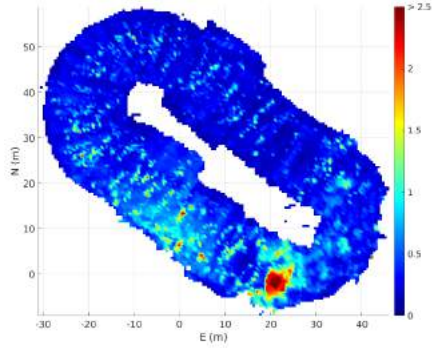
$\varepsilon_{\Sigma} = 8400.57$

#Cells

$\varepsilon_{\#} = 52735$

SLAM





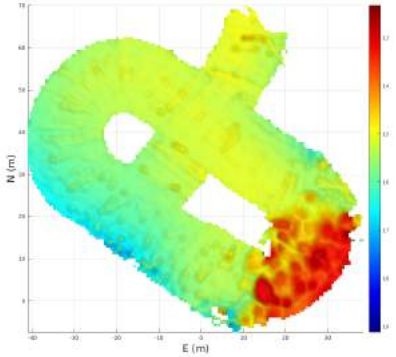
CBEE

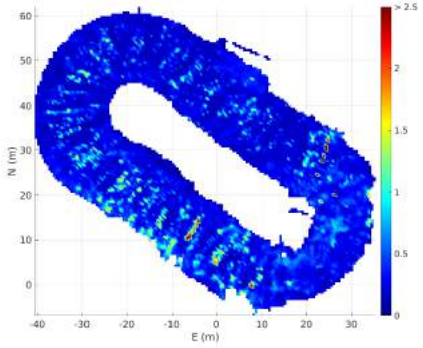
$\varepsilon_{\Sigma} = 4925.65, \varepsilon_{\Sigma}^{\downarrow} = 41.37\%$

#Cells

$\varepsilon_{\#} = 40342, \varepsilon_{\#}^{\downarrow} = 23.50\%$

USBL





CBEE

$\varepsilon_{\Sigma} = 3674.66, \varepsilon_{\Sigma}^{\downarrow} = 56.26\%$

#Cells

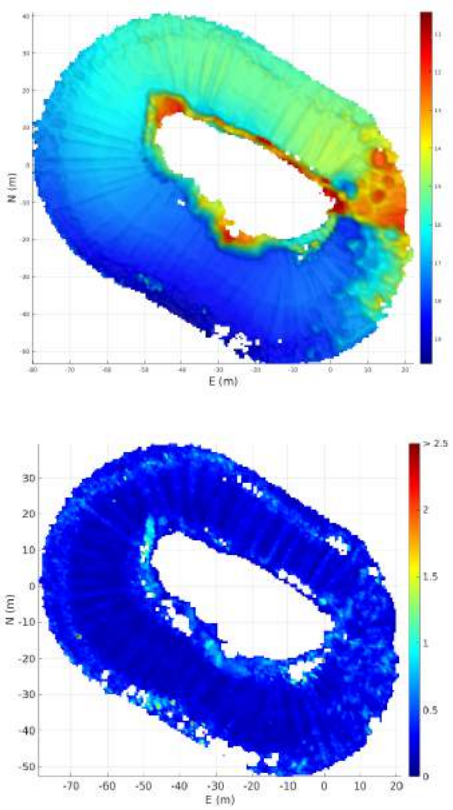
$\varepsilon_{\#} = 36209, \varepsilon_{\#}^{\downarrow} = 32.34\%$

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Porto Pim Dataset - I (2.5D)

Dead reckoning



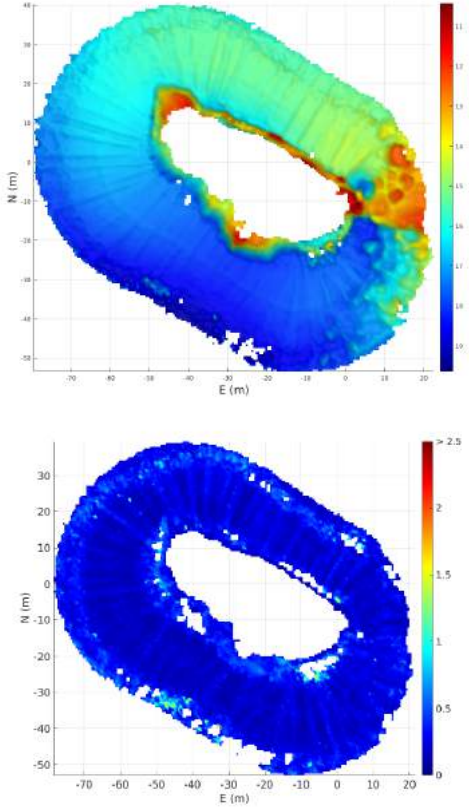
CBEE

$\varepsilon_{\Sigma} = 6379.63$

#Cells

$\varepsilon_{\#} = 63354$

SLAM



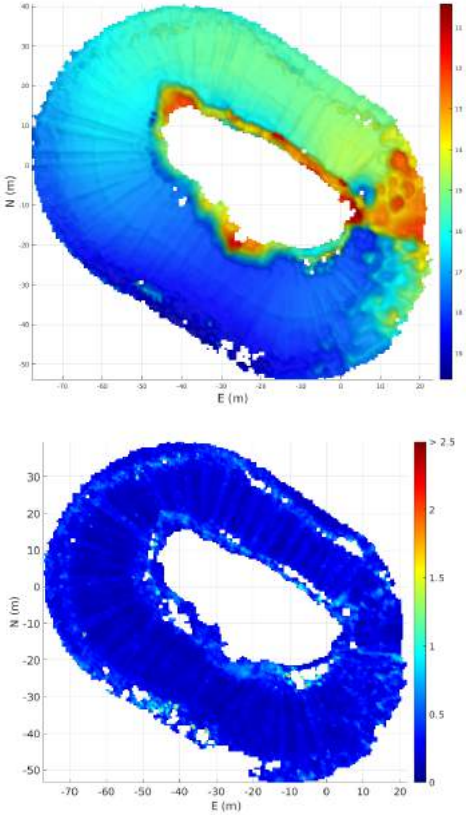
CBEE

$\varepsilon_{\Sigma} = 5836.97, \varepsilon_{\Sigma}^{\downarrow} = 8.51\%$

#Cells

$\varepsilon_{\#} = 60443, \varepsilon_{\#}^{\downarrow} = 4.59\%$

USBL



CBEE

$\varepsilon_{\Sigma} = 6092.49, \varepsilon_{\Sigma}^{\downarrow} = 4.5\%$

#Cells

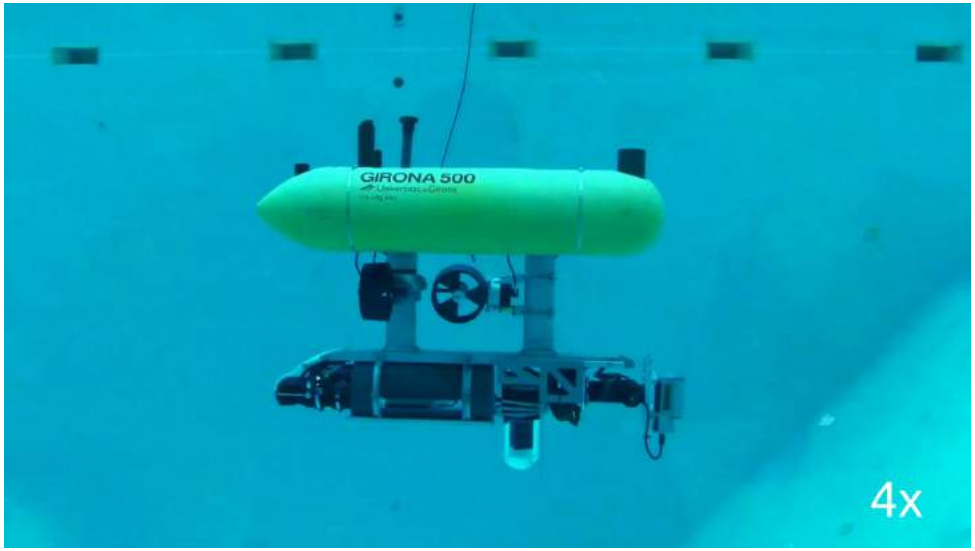
$\varepsilon_{\#} = 62125, \varepsilon_{\#}^{\downarrow} = 1.49\%$

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

St. Feliu de Guíxols Dataset (3D)

- Girona 500 (UdG)
- Survey area of 60 x 20 m
- Zero-shaped trajectory
- Multiple loop closures

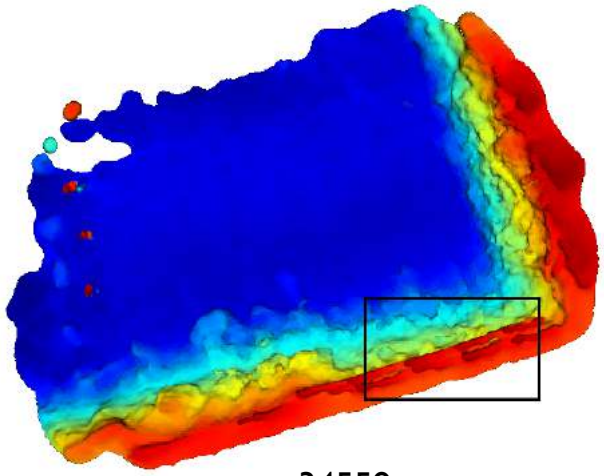


EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

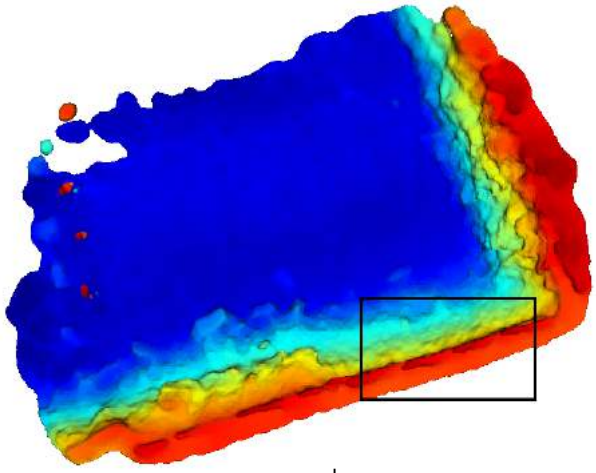
St. Feliu de Guíxols Dataset (3D)

Dead reckoning



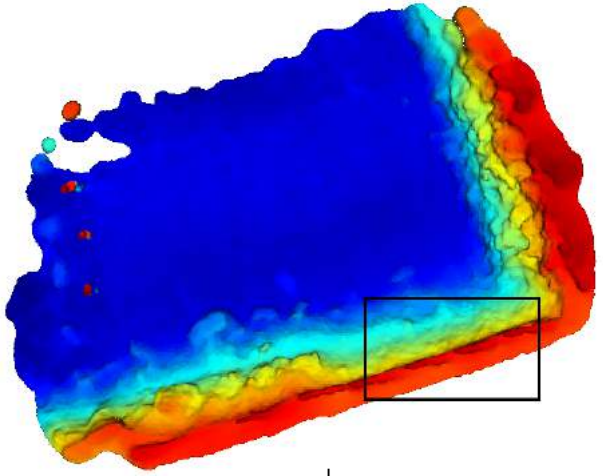
$\varepsilon_{\#} = 34559$

USBL



$\varepsilon_{\#} = 32057, \varepsilon_{\#}^{\downarrow} = 7.24\%$

SLAM



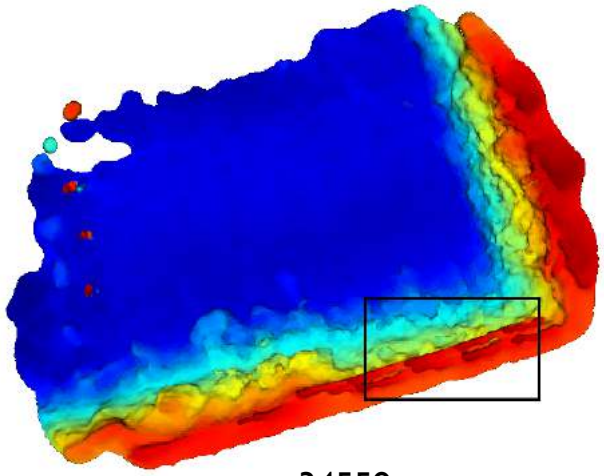
$\varepsilon_{\#} = 32570, \varepsilon_{\#}^{\downarrow} = 5.76\%$

EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

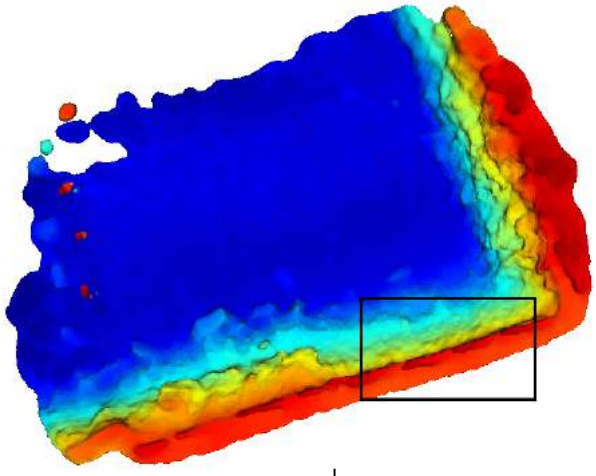
St. Feliu de Guíxols Dataset (3D)

Dead reckoning



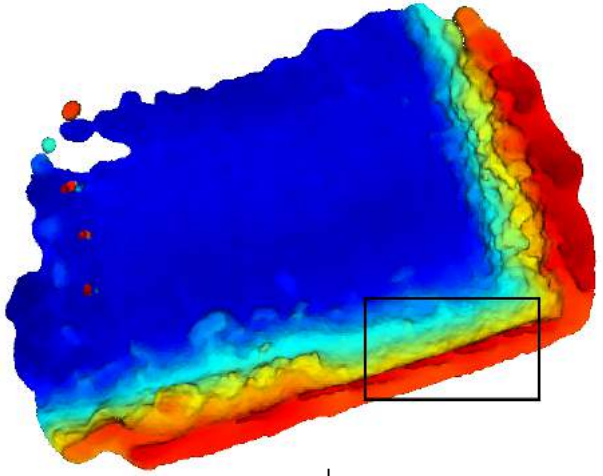
$\varepsilon_{\#} = 34559$

USBL

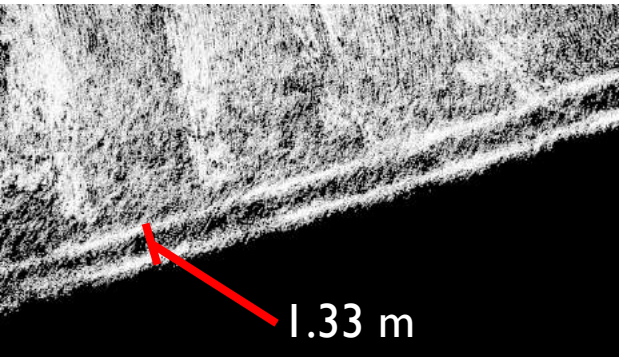


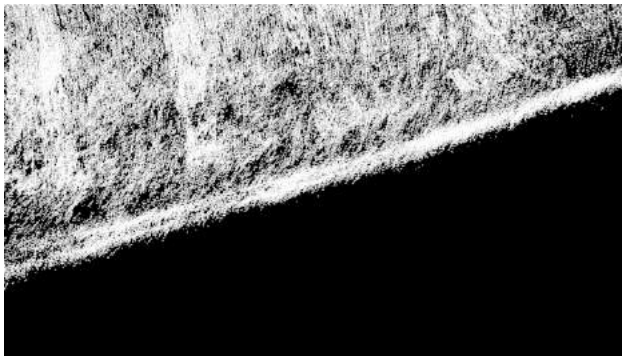
$\varepsilon_{\#} = 32057, \varepsilon_{\#}^{\downarrow} = 7.24\%$

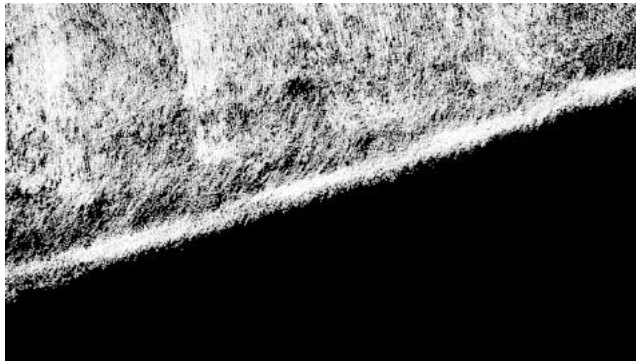
SLAM



$\varepsilon_{\#} = 32570, \varepsilon_{\#}^{\downarrow} = 5.76\%$





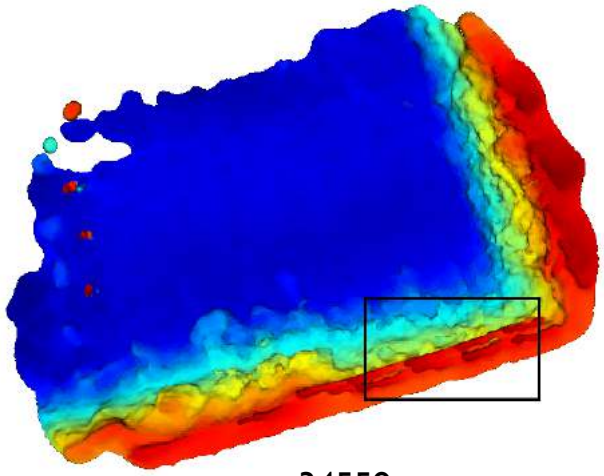


EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

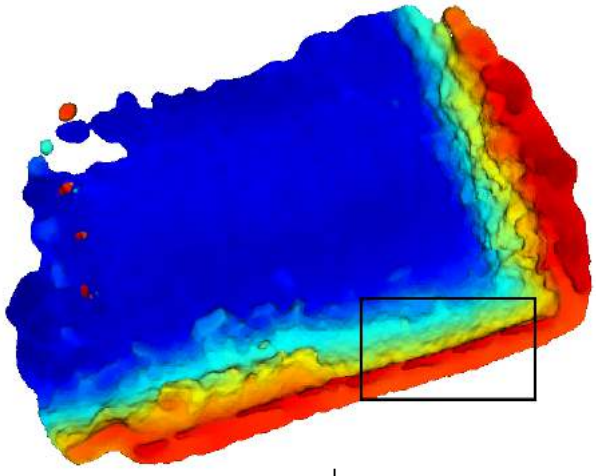
St. Feliu de Guíxols Dataset (3D)

Dead reckoning



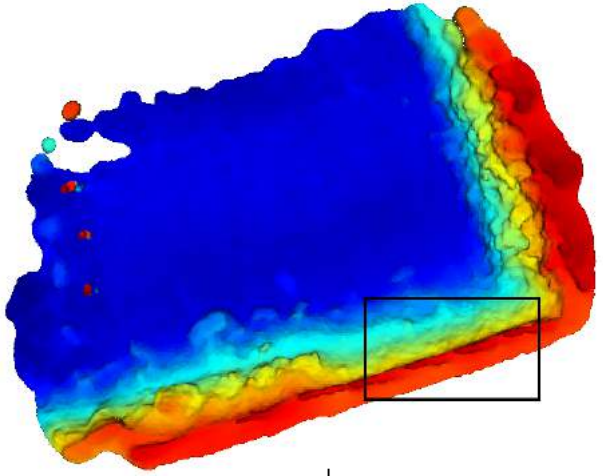
$\varepsilon_{\#} = 34559$

USBL

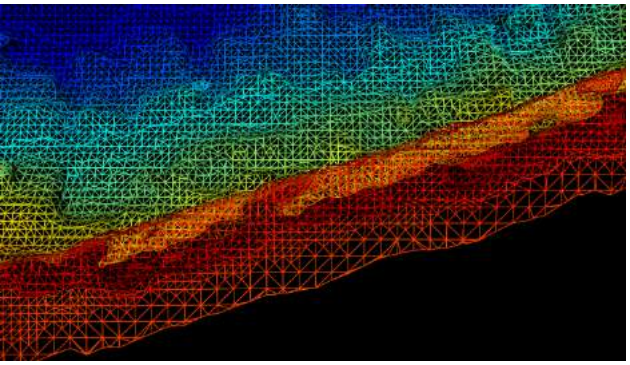


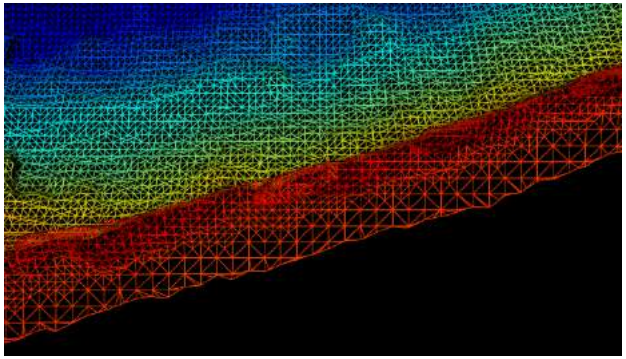
$\varepsilon_{\#} = 32057, \varepsilon_{\#}^{\downarrow} = 7.24\%$

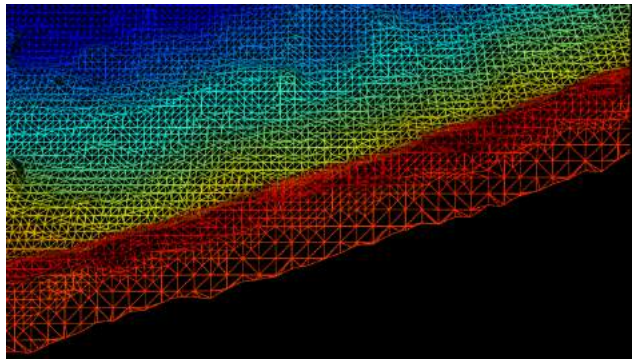
SLAM



$\varepsilon_{\#} = 32570, \varepsilon_{\#}^{\downarrow} = 5.76\%$







EKF Pose-Based SLAM

Underwater 3D multibeam SLAM

Summary

- Divide the mission into submaps
- 3D Gaussian point clouds
- Probabilistic registration
- Association computational cost in linear time
- Pose Based EKF SLAM Framework
- Extensive experimental evaluation

Conclusions

- Produces more consistent maps than dead reckoning navigation
- Map consistency equivalent to USBL