2022
Localization, Navigation and Smart Mobility Project

Esat Ince
Usevalad Milasheuski

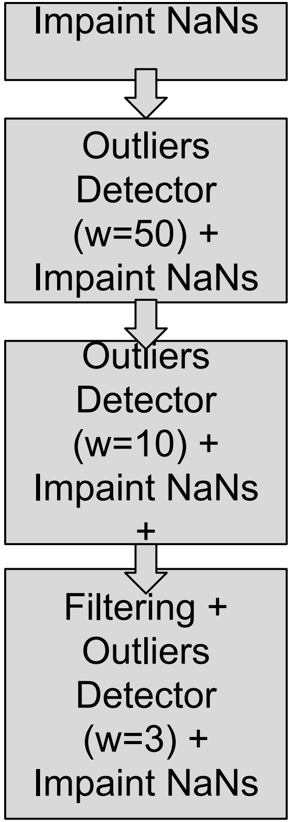
Problem

Using TDOA measurements from UWB kit, apply snapshot localization and filter tracking to estimate the position of AGV.

Approach

WNLS was used to perform a snapshot localization; EKF (NCV/NCA models) were used for Bayesian tracking. Since the ground truth is not provided, a "smoothed" dataset was generated for performance evaluation.

Outliers and NaN values removal

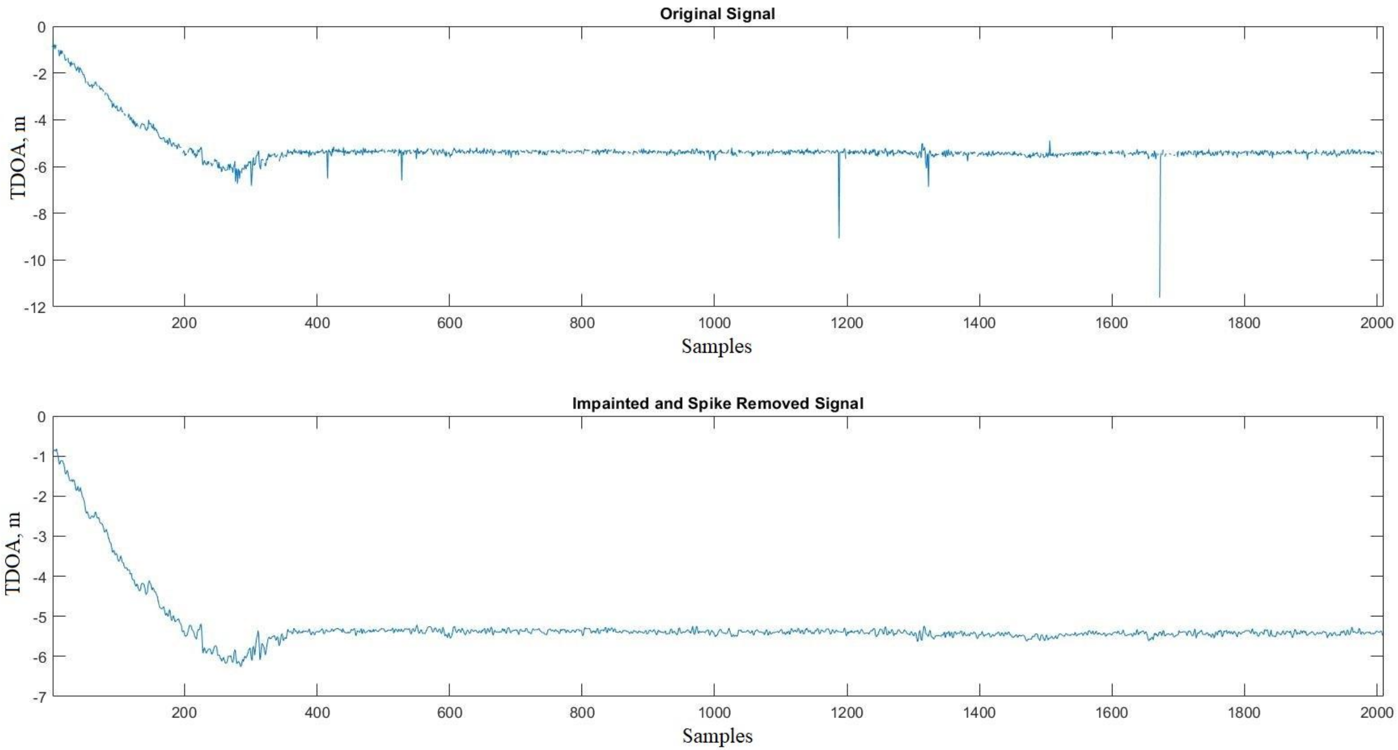


Having interpolated initial NaNs, we applied a method windowed method, which detects outliers according to the moving median. All the identified outliers were replaced with NaN values and linearly interpolated again. These cascade was applied three times with the reducing window size wrt. the previous one, At the final step, Savitzky-Golay filter (ord. 3, framelen. 5) was applied to smoothen the data and reduce the noise.

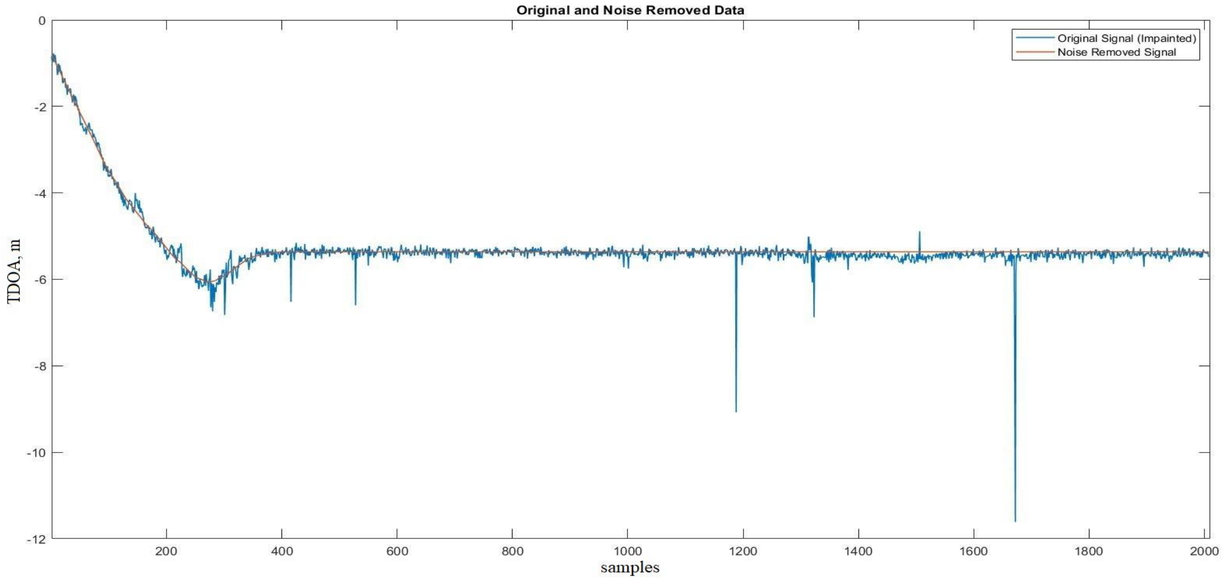
NaN Removal

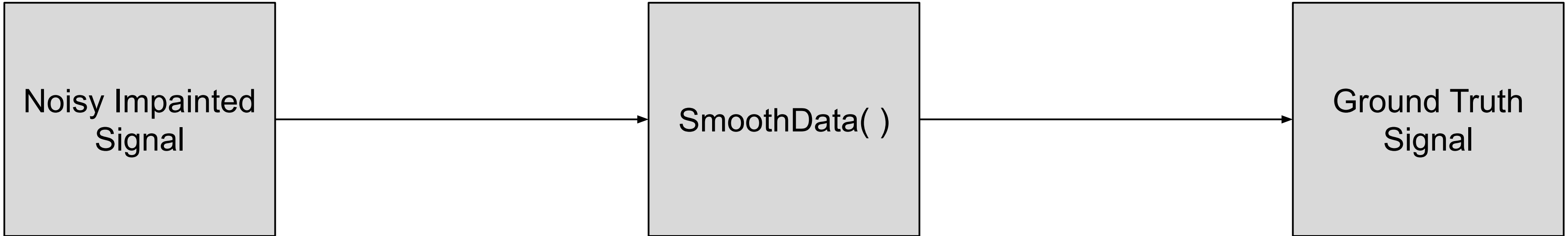
impaintnans(.): A custom MATLAB File Exchange function that drops the part of the known array except those who are connected to NaNs, and using Least Squares approach it paints the NaN values.

Example of the initial and preprocessed signal



Ground truth generation





TDOA Measurements

	Measurement	Jacobian
Method	$h_i(\boldsymbol{u})$	$[\mathbf{H}(\boldsymbol{u})]_i = \frac{\partial h_i(\boldsymbol{u})}{\partial \boldsymbol{u}}$
TDOA	$d_i - d_j = \lvert \boldsymbol{u} - \boldsymbol{s}_i \rvert - \lvert \boldsymbol{u} - \boldsymbol{s}_j \rvert$	$\frac{u_x - s_{i,x}}{d_i} - \frac{u_x - s_{j,x}}{d_j}, \frac{u_y - s_{i,y}}{d_i} - \frac{u_y - s_{j,y}}{d_j}$

$$h_{12}(\mathbf{u}) = \sqrt{(s_{x1} - u_x)^2 + (s_{y1} - u_y)^2} - \sqrt{(s_{x2} - u_x)^2 + (s_{y2} - u_y)^2}$$

WNLS

Iterative procedure:

- Initialization k=0:

$$\hat{\mathbf{u}}^{(k)} = \hat{\mathbf{u}}^{(0)}$$

- For iteration k=1,2,...:

1. Computation of:

$$\mathbf{H}^{(k)} = \left.\frac{\partial \mathbf{h}(\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\hat{\mathbf{u}}^{(k-1)}}$$

$$\Delta\rho_i^{(k)} = \rho_i - h_i\left(\hat{\mathbf{u}}^{(k-1)}\right)$$

$$\Delta\boldsymbol{\rho}^{(k)} = \left[\Delta\rho_i^{(k)}\right]_{i=1}^{N}$$

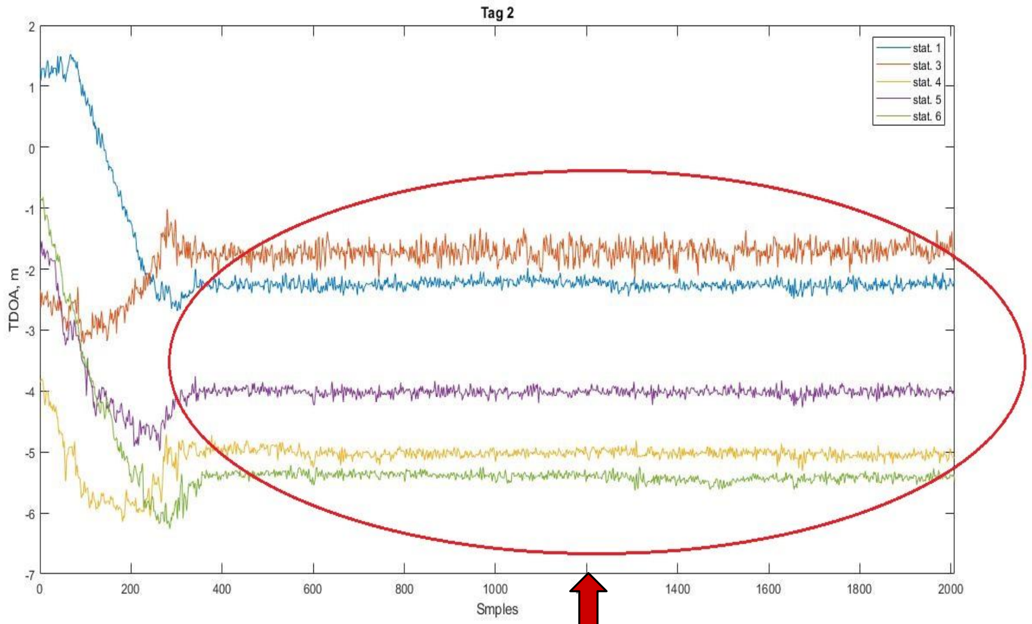
1. Inversion:

$$\Delta\mathbf{u}^{(k)} = \left(\mathbf{H}^{(k)^{\mathrm{T}}}\,\mathbf{R}^{-1}\,\mathbf{H}^{(k)}\right)^{-1}\mathbf{H}^{(k)^{\mathrm{T}}}\,\mathbf{R}^{-1}\,\Delta\boldsymbol{\rho}^{(k)}$$

2. Update the solutior..

$$\hat{\mathbf{u}}^{(k)} = \hat{\mathbf{u}}^{(k-1)} + \Delta\mathbf{u}^{(k)}$$

- Repeat till $\left|\Delta\mathbf{u}^{(k)}\right| < \varepsilon$ or k = num_iter_max



This part of the signal was used in order to empirically estimate variations for each station.

Tracking Filter

- **Prediction** of position u_t is made by past observations $\boldsymbol{\rho}_{1:t-1}$ using the **motion model**.

$$\mathbf{u}_t = \mathbf{f}_t(\mathbf{u}_{t-1}) + \mathbf{w}_{t-1} \longrightarrow p(\mathbf{u}_t | \boldsymbol{\rho}_{1:t-1}) \quad \text{Prior Pdf}$$

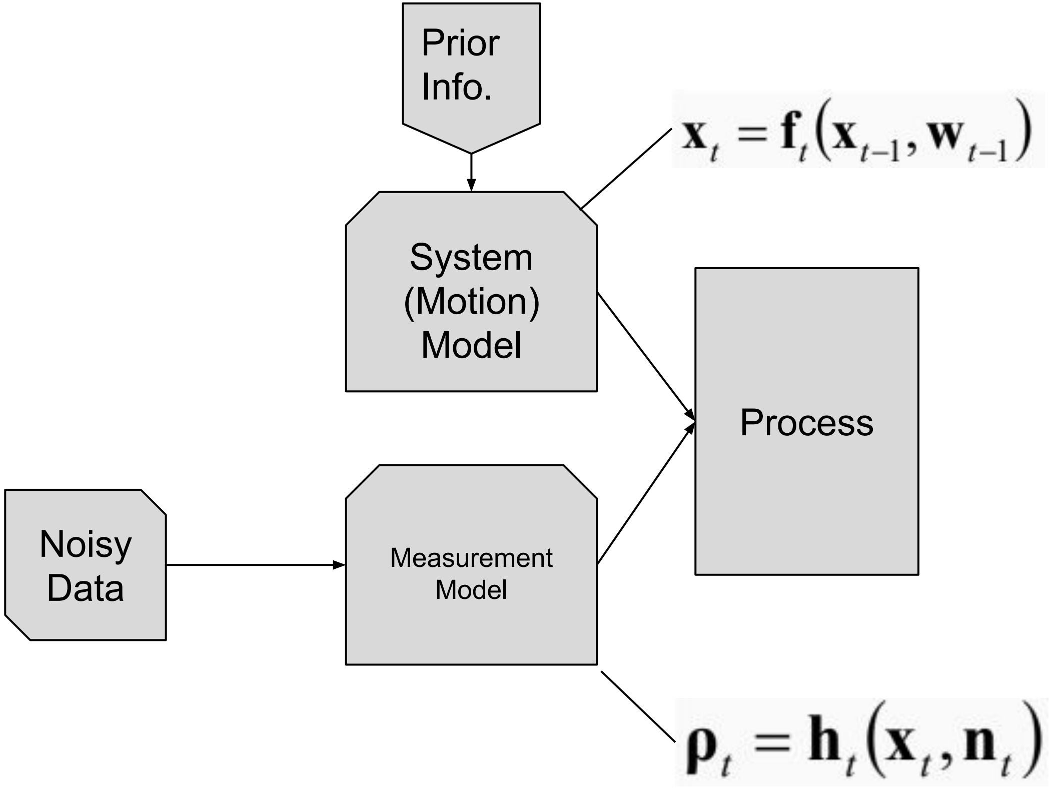
- **Likelihood** is evaluated from current measurement $\boldsymbol{\rho}_t$ using **measurement model**.

$$\boldsymbol{\rho}_t = \mathbf{h}_t(\mathbf{u}_t) + \mathbf{n}_t \longrightarrow p(\boldsymbol{\rho}_t | \mathbf{u}_t) \quad \text{Likelihood}$$

- Estimate is updated using the **Bayes** rule.

$$\boxed{\underbrace{p(\mathbf{u}_t | \boldsymbol{\rho}_{1:t})}_{\text{Posterior pdf}} = \Gamma_t \cdot \underbrace{p(\mathbf{u}_t | \boldsymbol{\rho}_{1:t-1})}_{\text{Prior pdf}} \cdot \underbrace{p(\boldsymbol{\rho}_t | \mathbf{u}_t)}_{\text{Likelihood}}}$$

Tracking Filter for Mobile Positioning



Why EKF?

Claims:

- Measurement model is nonlinear
- Noise pdf is Gaussian

EKF is used for Gaussian pdf's and nonlinear models. We linearize the model around the current location fix and approximate the pdf's as Gaussians.

EKF Model

Prediction:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{f}_t\left(\hat{\mathbf{x}}_{t-1|t-1}\right)$$

$$\mathbf{C}_{t|t-1} = \hat{\mathbf{F}}_t \mathbf{C}_{t-1|t-1} \hat{\mathbf{F}}_t^T + \mathbf{Q}_{t-1}$$

Update:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{G}_t \underbrace{\left(\boldsymbol{\rho}_t - \mathbf{h}_t\left(\hat{\mathbf{x}}_{t|t-1}\right)\right)}_{\text{Innovation } \varepsilon_{t|t-1}}$$

$$\mathbf{C}_{t|t} = \mathbf{C}_{t|t-1} - \mathbf{G}_t \hat{\mathbf{H}}_t \mathbf{C}_{t|t-1}$$

$$\mathbf{G}_t = \mathbf{C}_{t|t-1} \hat{\mathbf{H}}_t^T \left(\hat{\mathbf{H}}_t \mathbf{C}_{t|t-1} \hat{\mathbf{H}}_t^T + \mathbf{R}_t\right)^{-1}$$

Motion Model

Since there are no velocity or acceleration sensors, there are three main motion models that can be used.

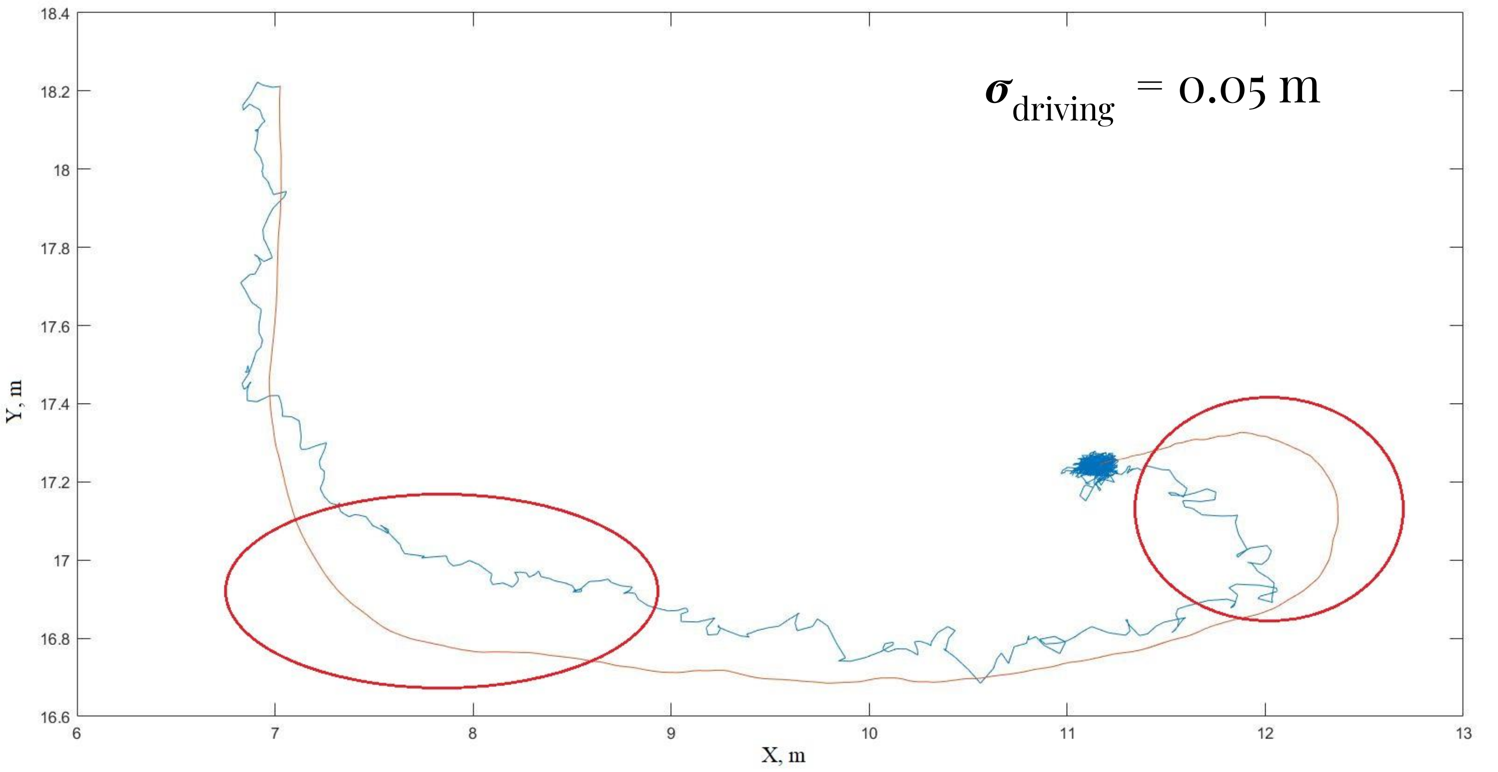
1. **Random Walk Model**
2. **Nearly Constant Velocity Model**
3. **Nearly Constant Acceleration Model (Random Jerk)**

Why Random Jerk Model?

- Random walk model uses a driving process of zero-mean random velocity, which is a suitable model for a pedestrian but not for a car.

- Nearly Constant Velocity Model and Nearly Constant Acceleration Model is suitable for a car. However, Nearly Constant Velocity model cannot perform well during sharp turns, quick stops or accelerations. We implemented both of the models and showed why Nearly Constant Velocity is performing worse.

EKF (Nearly constant velocity)



EKF (Nearly constant acceleration)

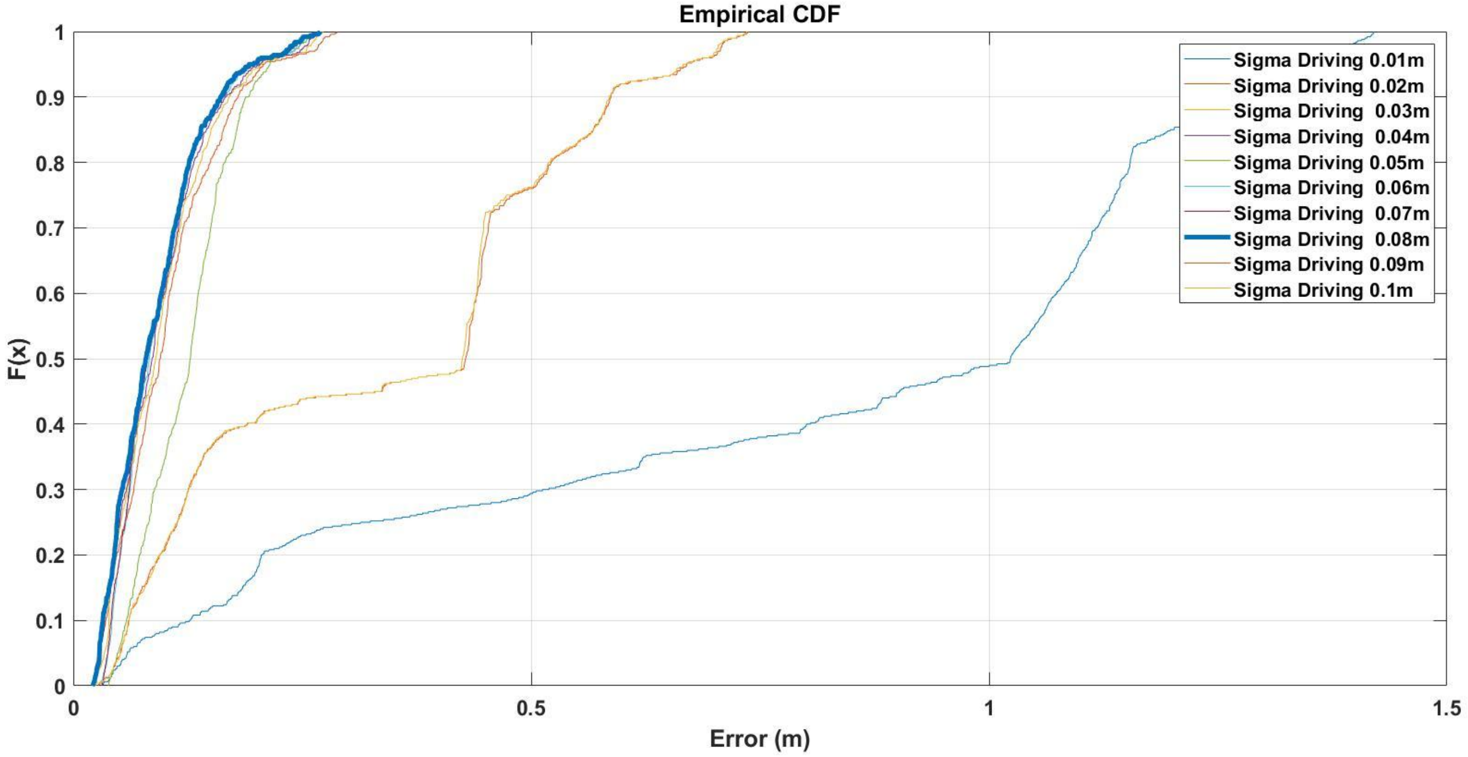
$$\mathbf{x}_t = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{v}_t \\ \mathbf{a}_t \end{bmatrix} = \begin{bmatrix} u_{x,t} \\ u_{y,t} \\ v_{x,t} \\ v_{y,t} \\ a_{x,t} \\ a_{y,t} \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \mathbf{u}_t \\ \mathbf{v}_t \\ \mathbf{a}_t \end{bmatrix}}_{\mathbf{x}_t} = \underbrace{\begin{bmatrix} \mathbf{I}_2 & T\mathbf{I}_2 & \frac{T^2}{2}\mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 & T\mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}}_{\mathbf{F}} \cdot \underbrace{\begin{bmatrix} \mathbf{u}_{t-1} \\ \mathbf{v}_{t-1} \\ \mathbf{a}_{t-1} \end{bmatrix}}_{\mathbf{x}_{t-1}} + \underbrace{\begin{bmatrix} \frac{T^3}{6}\mathbf{I}_2 \\ \frac{T^2}{2}\mathbf{I}_2 \\ T\mathbf{I}_2 \end{bmatrix}}_{\mathbf{L}} \cdot \boxed{\mathbf{w}_{j,t-1}}$$

$$\mathbf{w}_{j,t} = \begin{bmatrix} w_{jx,t} \\ w_{jy,t} \end{bmatrix} = \frac{\mathbf{a}_t - \mathbf{a}_{t-1}}{T}$$

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{L}\mathbf{w}_{j,t-1} \qquad \mathbf{w}_{j,t} \sim \mathcal{N}\left(\mathbf{0}, \sigma_j^2 \mathbf{I}_2\right) \Rightarrow \mathbf{x}_t \sim \mathcal{N}\left(\mathbf{F}\mathbf{x}_{t-1}, \boxed{\sigma_j^2 \mathbf{L}\mathbf{L}^{\mathrm{T}}}\right) \rightarrow \mathbf{Q}$$

EKF (Nearly Constant Acceleration)



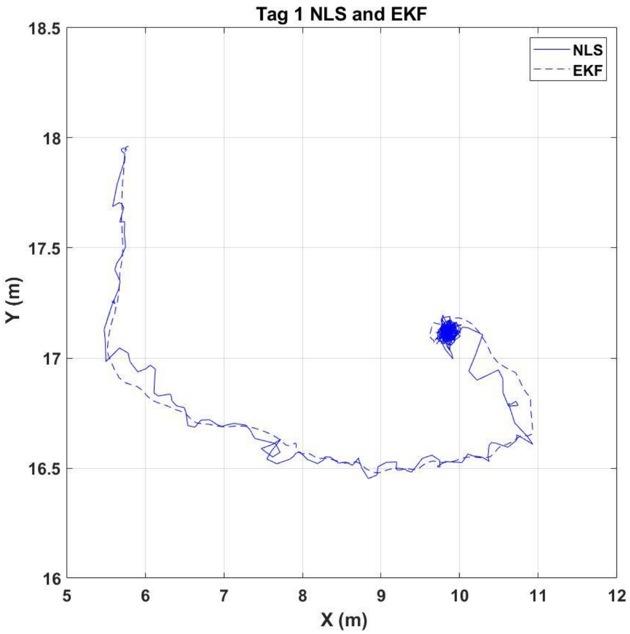
Results : Noise Properties

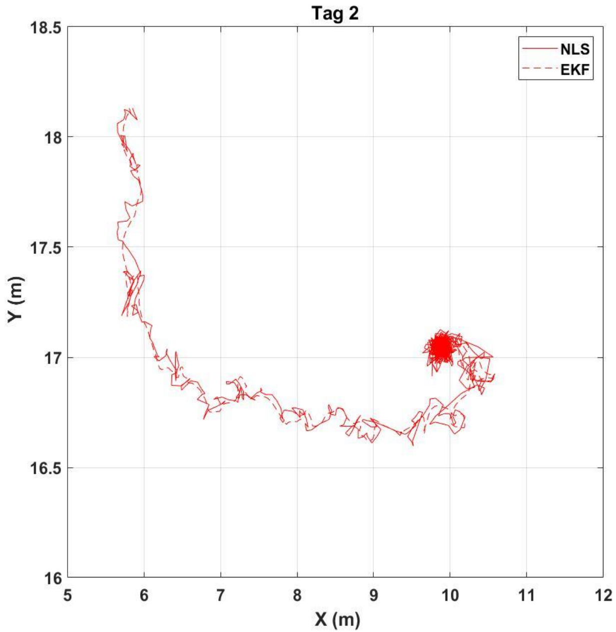
We found:

For 4 tags, on the average: $\boldsymbol{\mu}_{noise} = 0.38$ cm

For 4 tags, on the average: $\boldsymbol{\sigma}_{noise} = 8.24$ cm

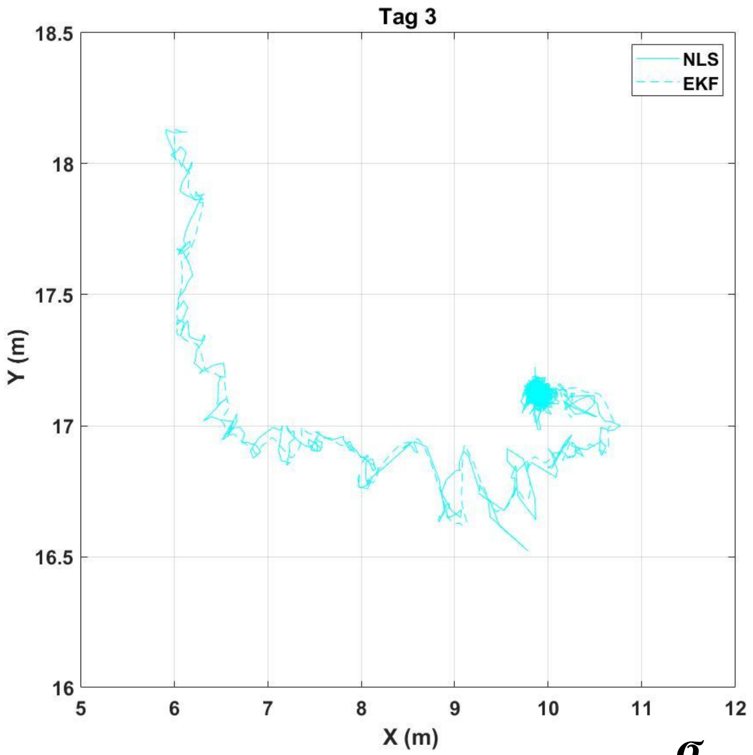
Results: WNLS and EKF for Each Tag

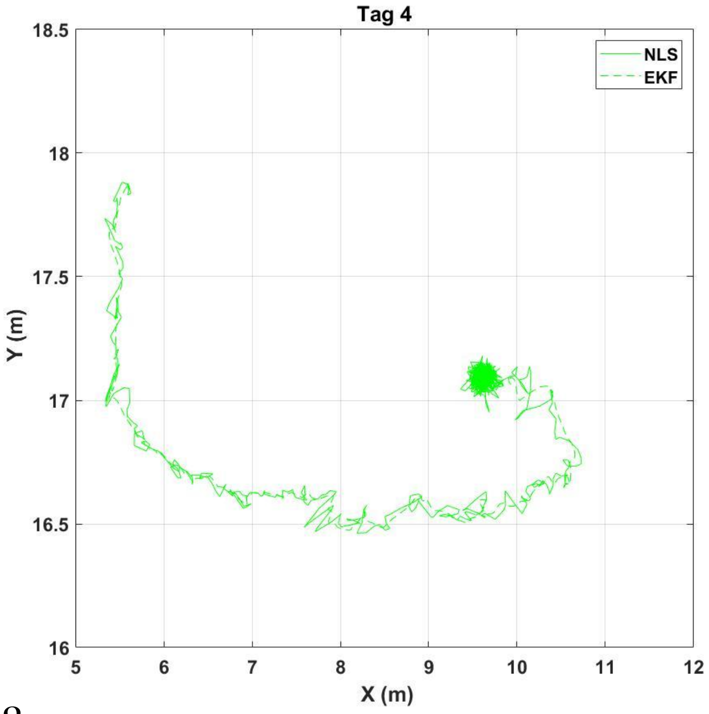




$$\boldsymbol{\sigma}_{\text{driving}} = 0.08 \text{ m}$$

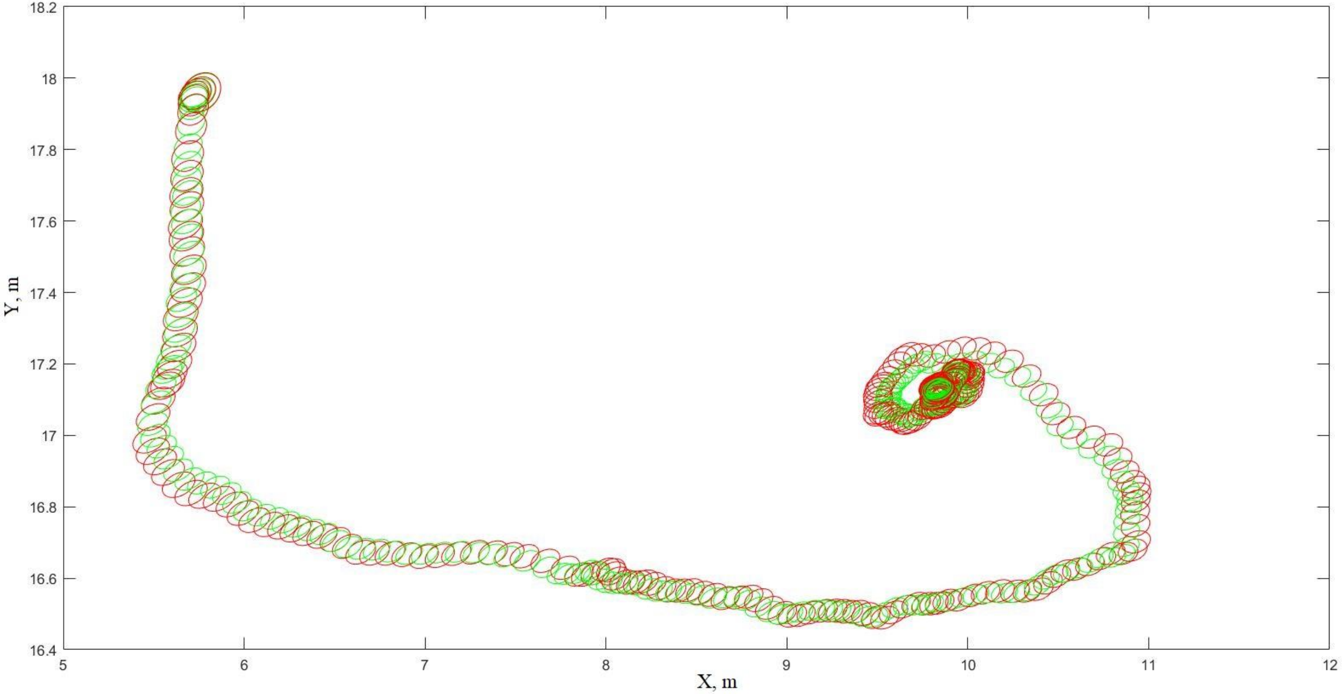
Results: WNLS and EKF for Each Tag





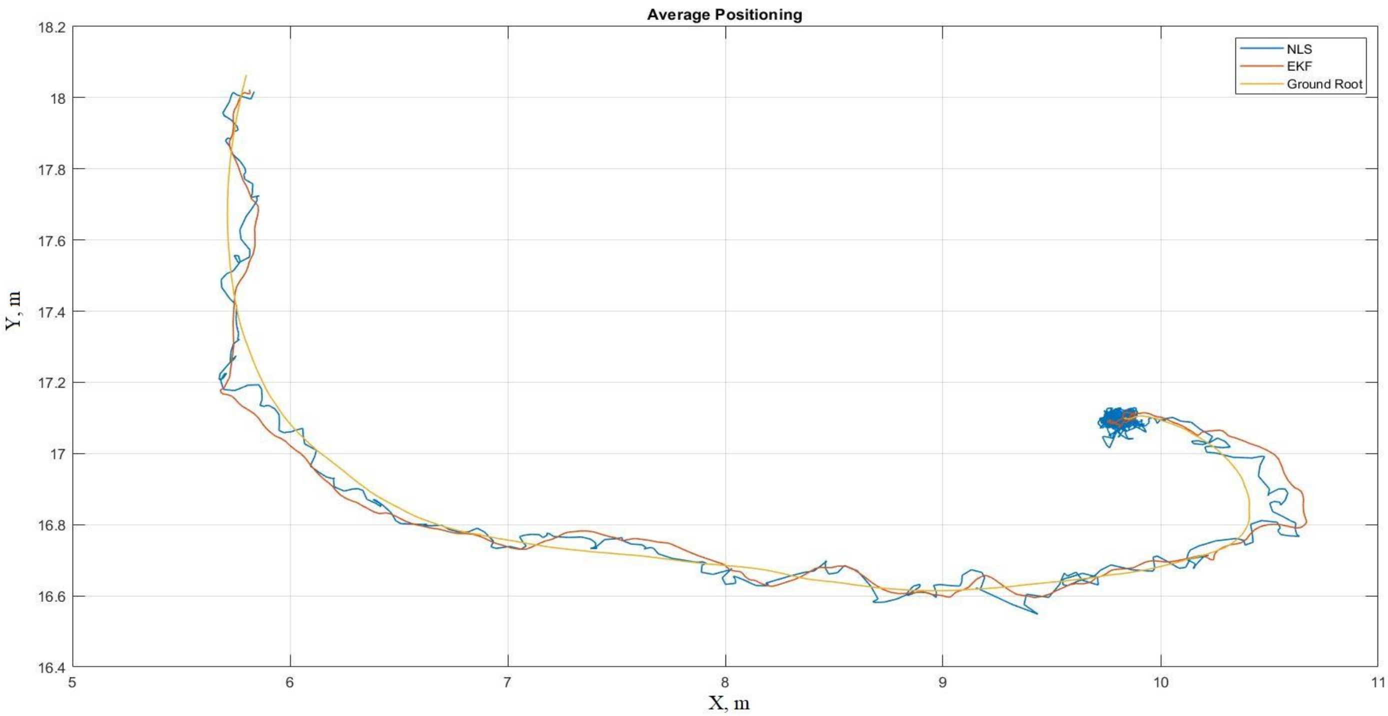
$$\sigma_{\text{driving}} = 0.08 \text{ m}$$

Results: Covariance Circles for Tag 1



$$\sigma_{\text{driving}} = 0.08 \text{ m}$$

Results: Overall Positioning WNLS-EKF-Original Position



$$\boldsymbol{\sigma}_{\text{driving}} = 0.08 \text{ m}$$

Results: Performance

DURING MOTION

EKF(NCA)

σ_x	6.32 cm
σ_y	1.86 cm
σ_H	6.59 cm

WNLS

σ_x	5.42 cm
σ_y	1.82 cm
σ_H	5.72 cm

EKF(NCV)

σ_x	7.25 cm
σ_y	2.02 cm
σ_H	7.73 cm

Results: Performance

STATIC PHASE

EKF(NCA)

σ_x	1.50 cm
σ_y	0.12 cm
σ_H	1.505 cm

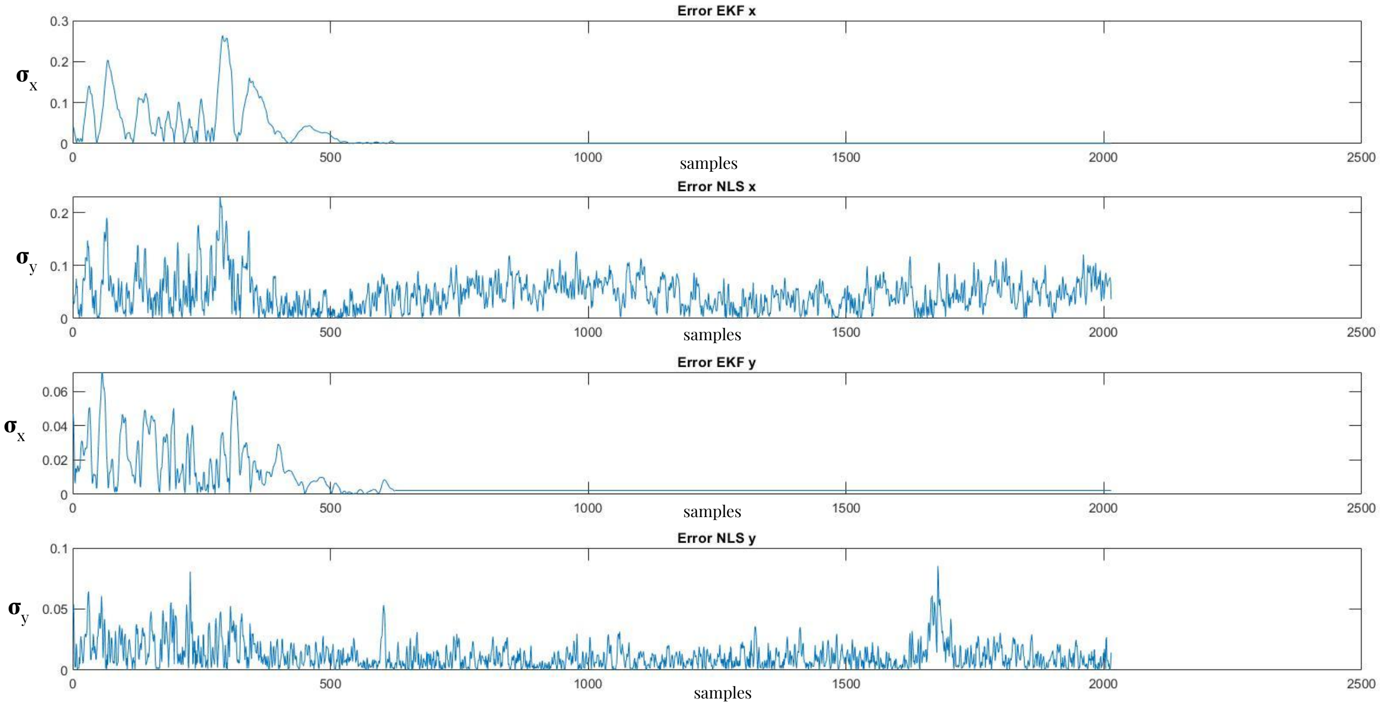
WNLS

σ_x	4.54 cm
σ_y	1.01 cm
σ_H	4.89 cm

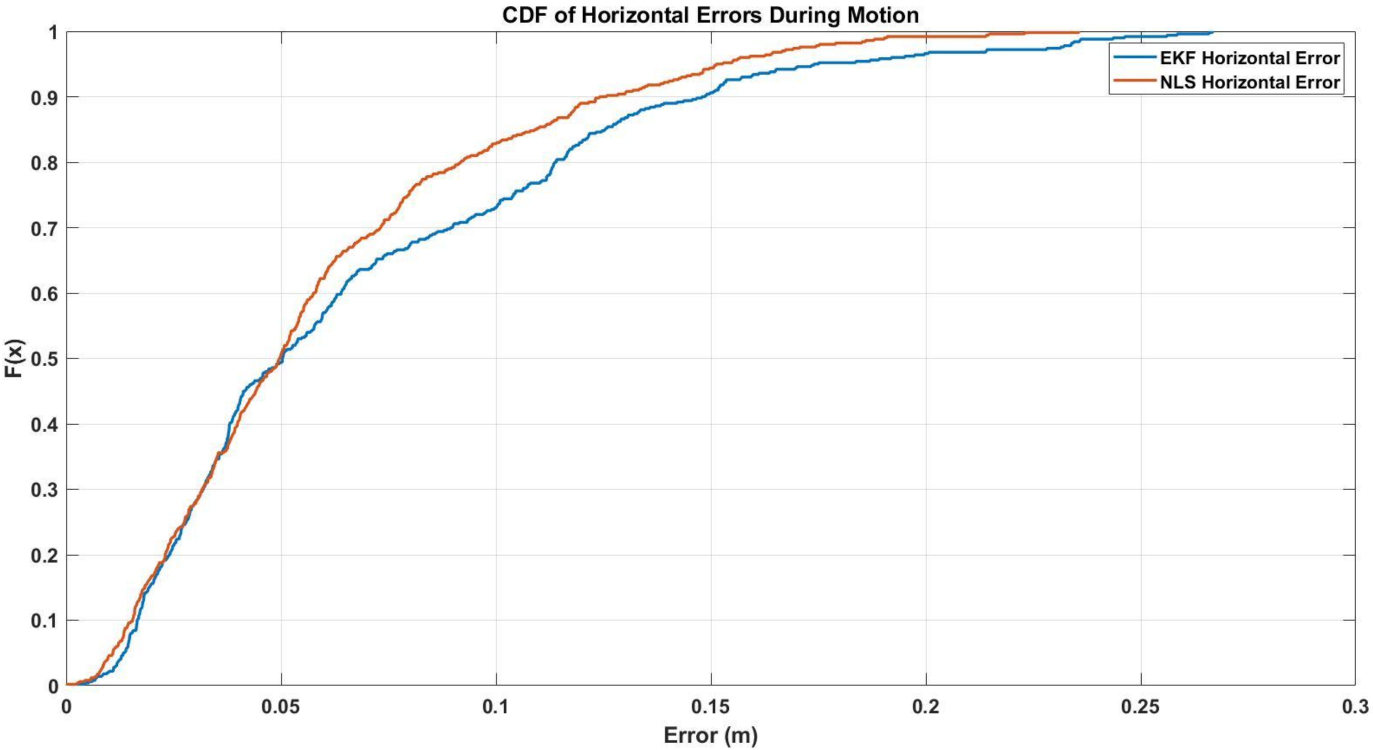
EKF(NCV)

σ_x	0.34 cm
σ_y	0.55 cm
σ_H	0.6466 cm

Results: Performance



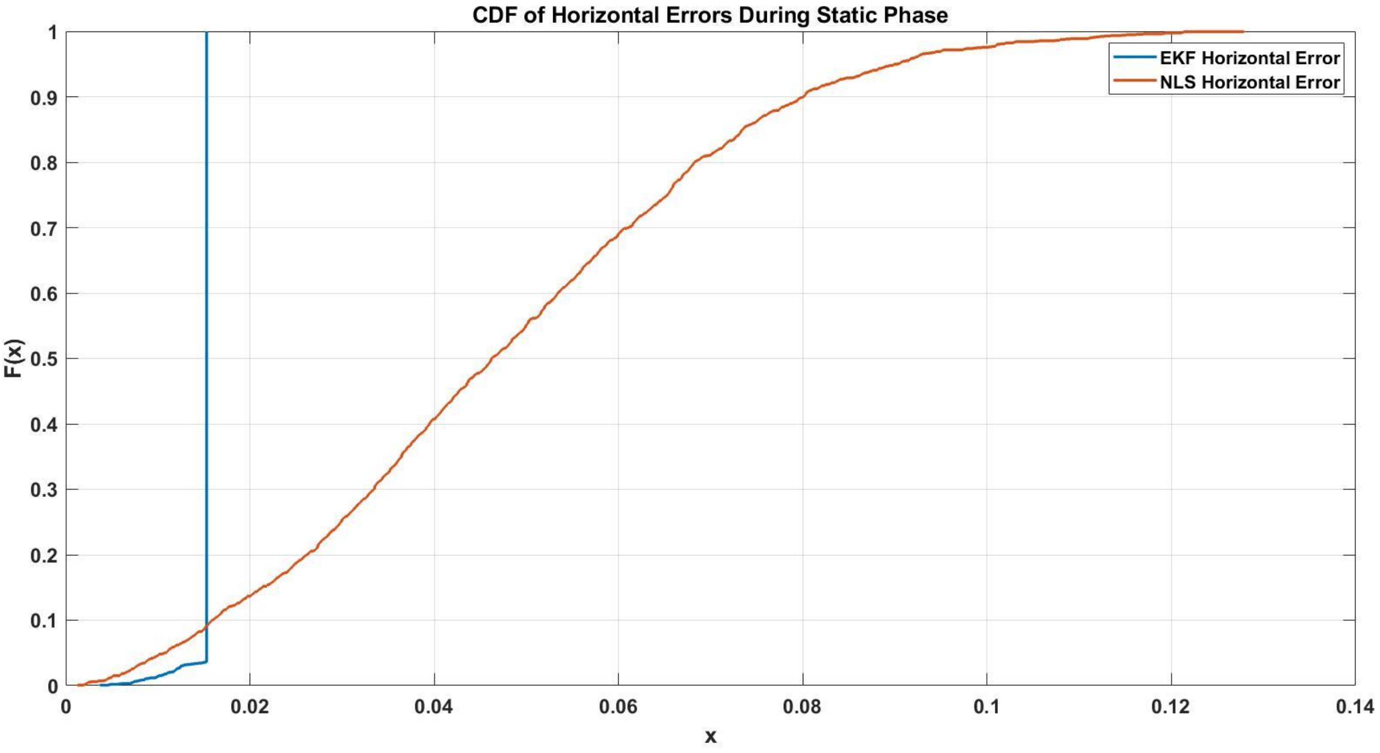
Results: Performance



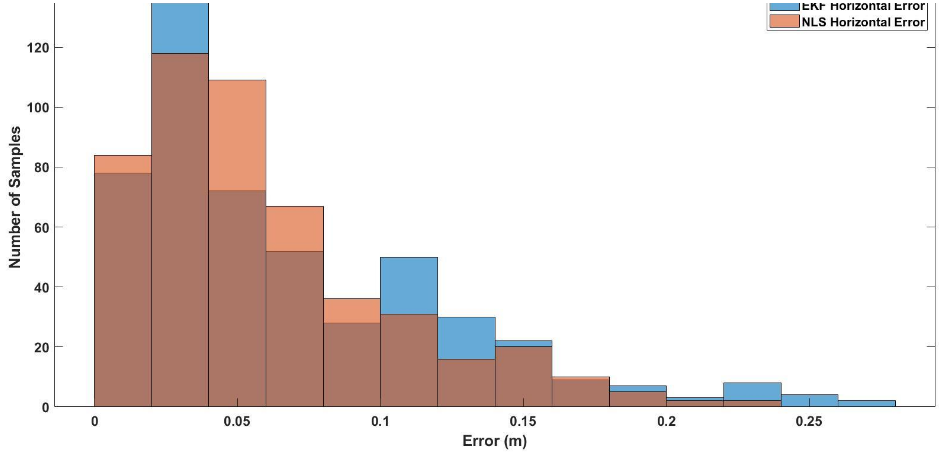
NLS CEP_{95} = 15.99cm
EKF CEP_{95} = 16.95cm

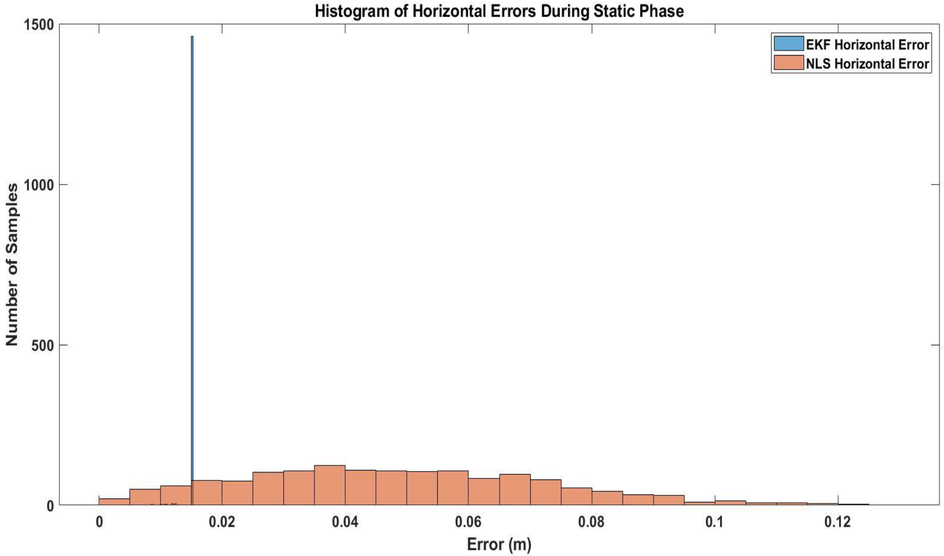
NLS CEP_{50} = 5 cm
EKF CEP_{50} = 5 cm

Results: Performance

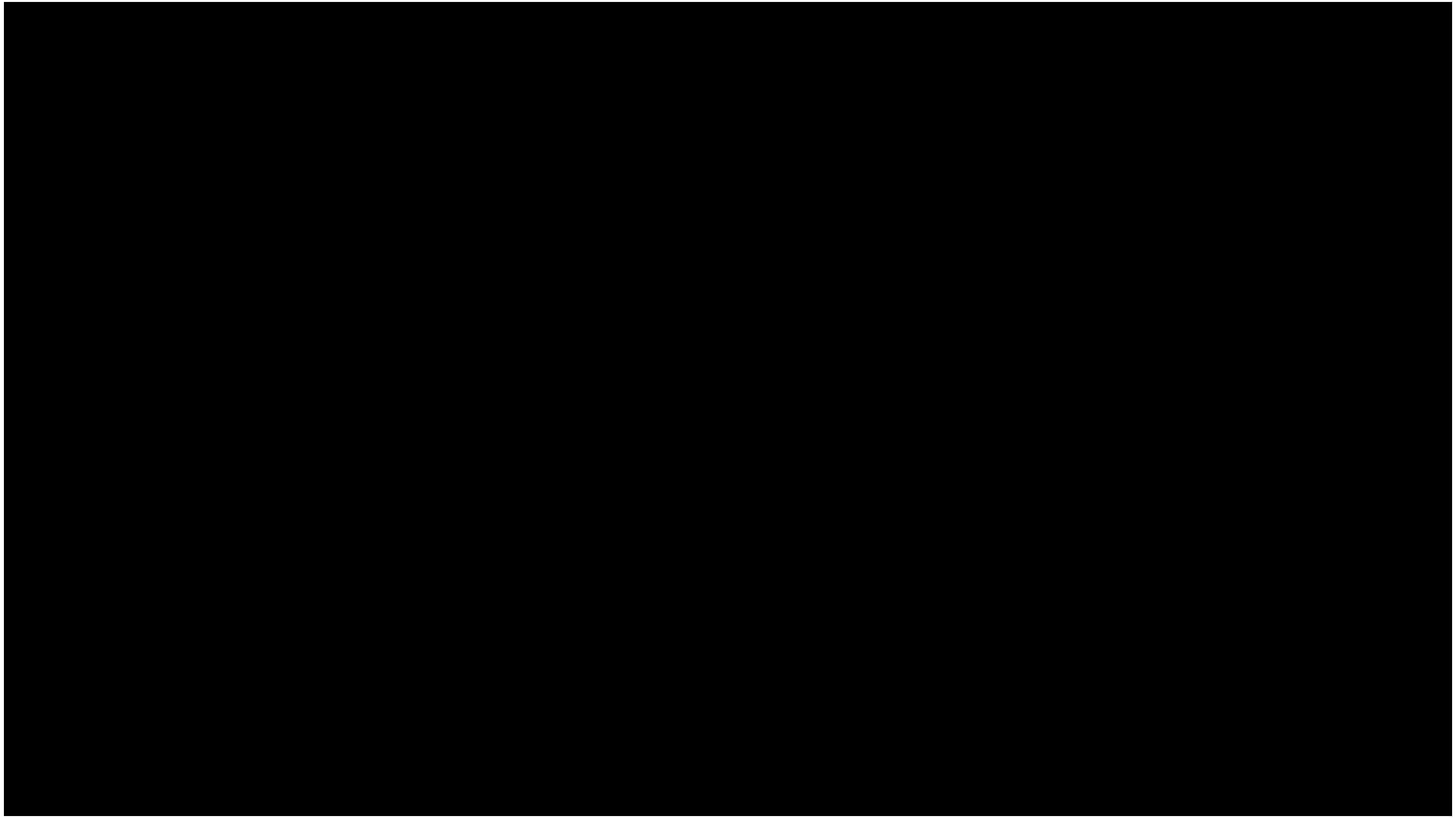


Results: Performance





Results: Animated Trajectory



Thank you for listening!