Cooperative Navigation Using an Unscented Kalman Filter (UKF)

Steven Dourmashkin
Graduate Research Assistant

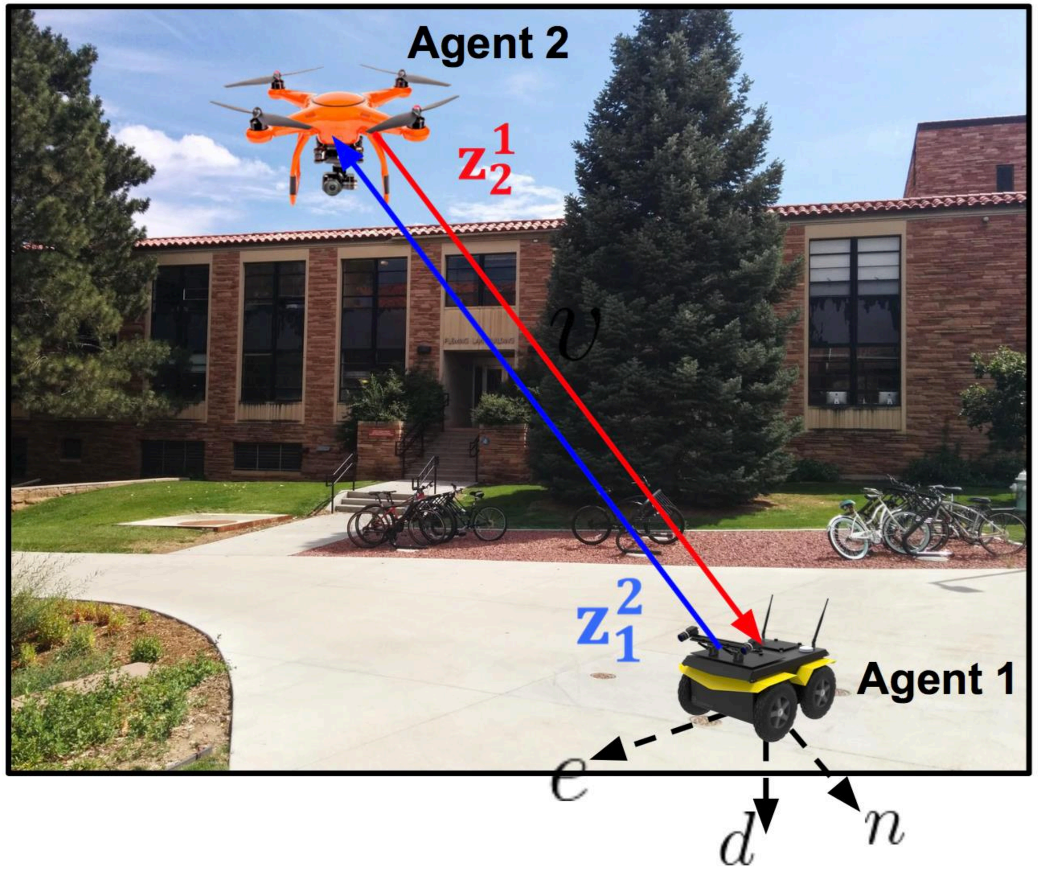
10/16/17

Cooperative Human-Robot Intelligence Laboratory
Ann and H.J. Smead Aerospace Engineering Sciences
University of Colorado at Boulder

COHRINT

Overview

- **Local Nav. Filter:**
  - Disco drone uses inertial measurements (IMU: 3-axis accelerometer + 3-axis gyro) combined with external measurements (GPS, 3-axis magnetometer, barometer, and pitot tube) to estimate ownship states.
- **Tracking Filter:**
  - Range measurements (e.g., 3-axis relative position) to other agents (e.g., Jackal UGV) used to track states of those agents.
- **Cooperative Filter:**
  - Measurements and/or states shared between agents to improve states of entire team.



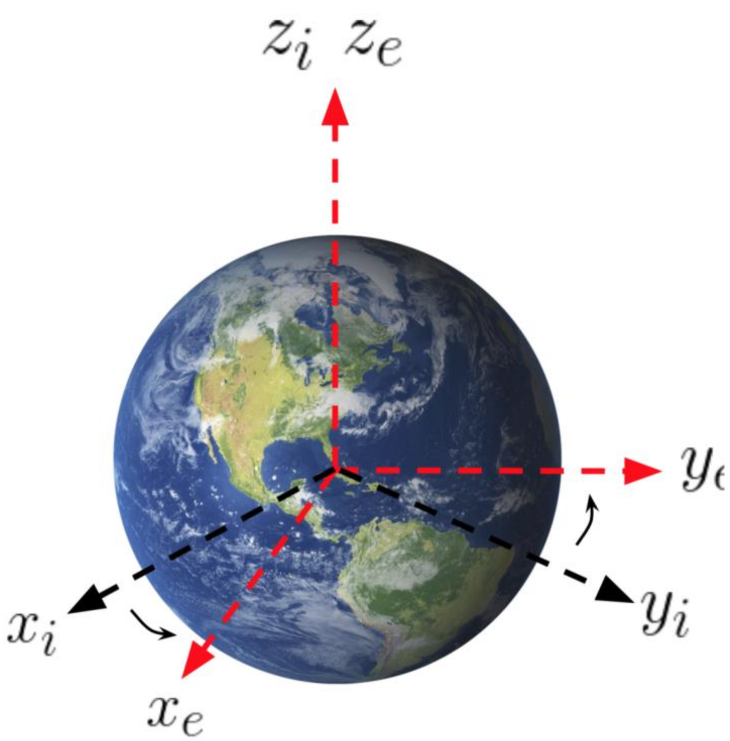
Local Nav. + Tracking Filter

COHRINT

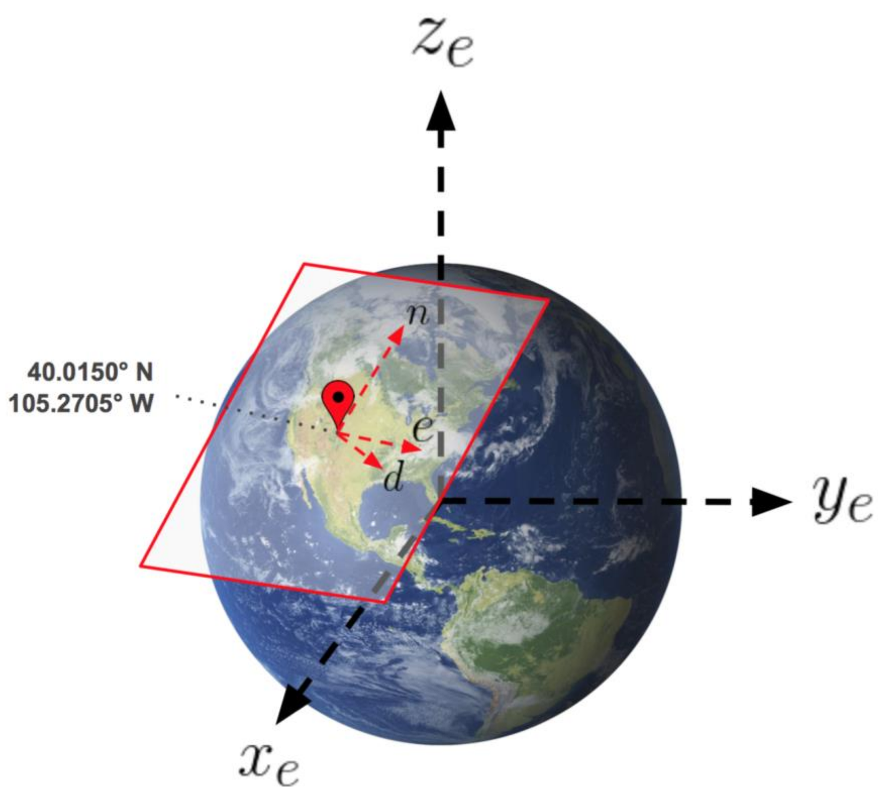
Coordinate Systems

Navigation frame

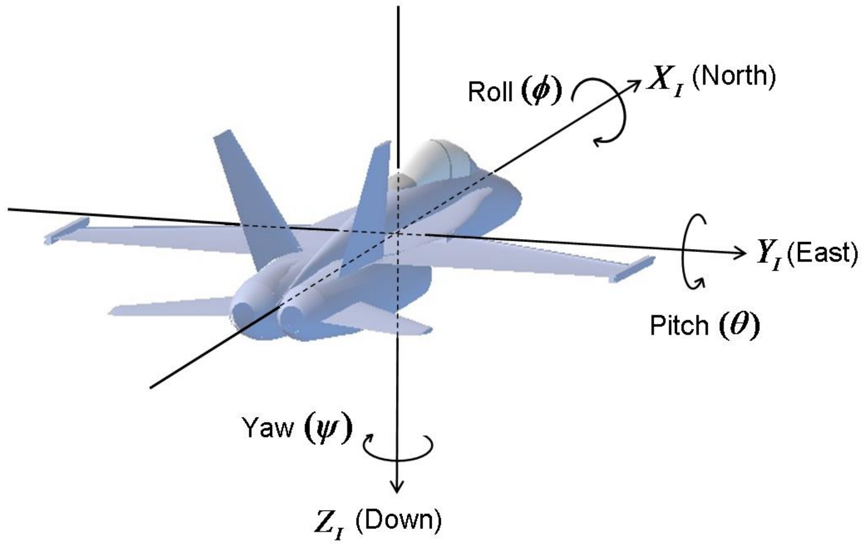
ECI & ECEF



ECEF & Tangential frame
(North-East-Down)

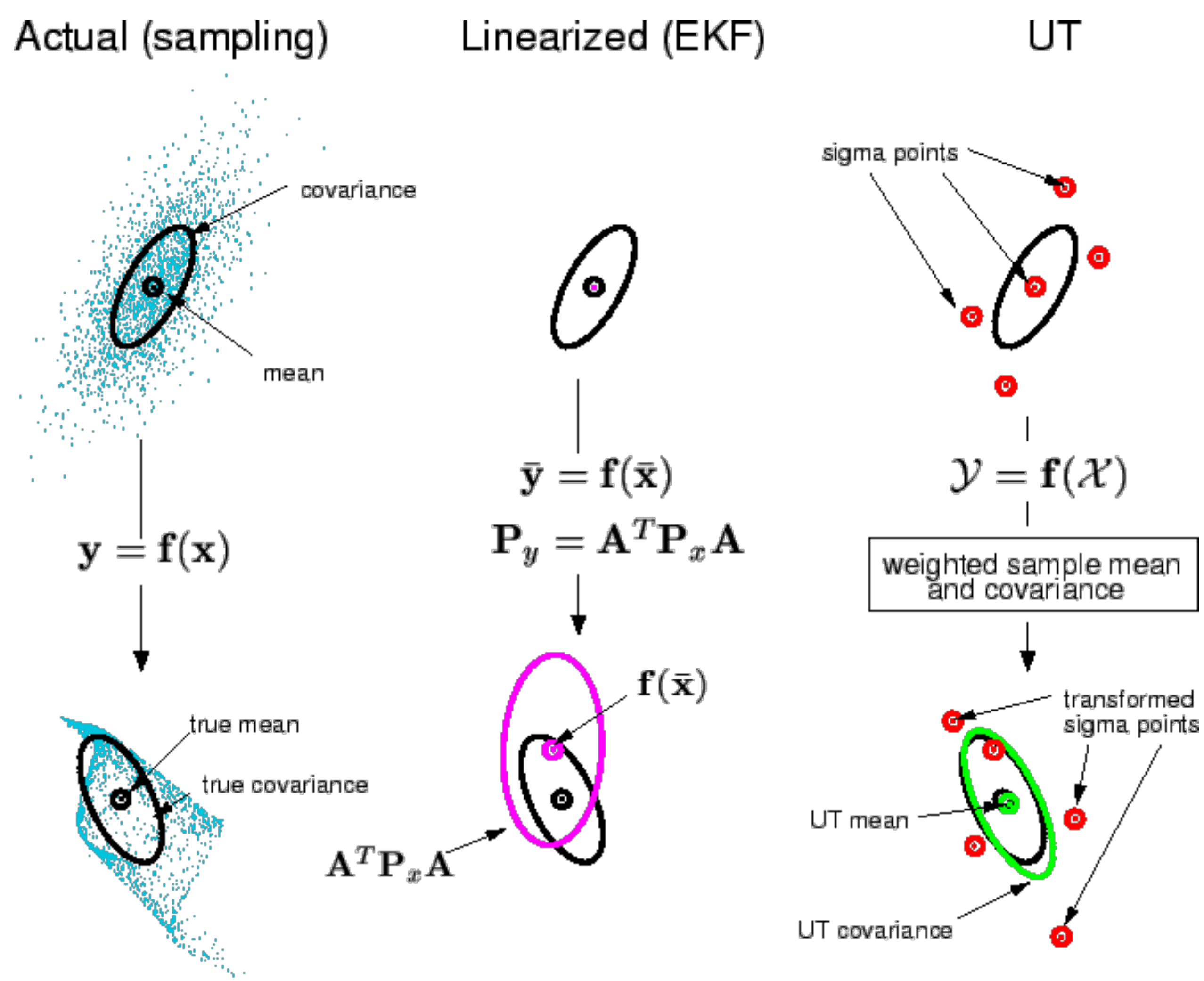


Body Frame



Approach

- **Use Unscented Kalman Filter (UKF)**
  - Deterministic sampling approach to capture propagation of estimated mean and covariance through non-linear dynamics and measurements
  - No linearization required → more desirable than EKF, which requires linearization about estimates at each iteration and can in turn diverge
  - Tunable – parameters can be chosen based on known prior distributions and degree of nonlinearity.



https://www.seas.harvard.edu/courses/cs281/papers/unscented.pdf

UKF Tutorial

- **Part 1:** Calculating Sigma Points and corresponding weights

Estimated state mean

Estimated state cov.

Dimensionality of state vector

$$\chi_0 = \bar{x}$$

$$\chi_i = \bar{x} + \left(\sqrt{(n+\kappa)P_{xx}}\right)_i \qquad i = 1, \cdots, n$$

$$\chi_i = \bar{x} - \left(\sqrt{(n+\kappa)P_{xx}}\right)_i \qquad i = n+1, \cdots, 2n$$

$$\kappa = \alpha^2(n+\lambda) - n$$

sigma points

$$W_0^m = \kappa/(n+\kappa)$$

$$W_0^c = \kappa/(n+\kappa) + (a - \alpha^2 + \beta)$$

$$W_i^c = W_i^m = 1/[2(\kappa+n)] \qquad i = 1, \cdots, 2n$$

corresponding weights

α : influences how far sigma points are away from mean $(0 \leq \alpha \leq 1)$

κ : secondary parameter that influences how far sigma points are away from mean $(\kappa \geq 0)$

β : incorporate any prior knowledge about the distribution $(\beta = 2$ optimal for Gaussians$)$

UKF Tutorial

- **Part 2:** Setting up process model

Ownship process model: based on nav frame IMU dynamics

$$\begin{bmatrix} \dot{p}_j \\ \dot{v}_j \\ \dot{q}_j \\ \dot{b}_{aj} \\ \dot{b}_{\omega j} \end{bmatrix} = \begin{bmatrix} v_j \\ R_b^n \overbrace{(a^b - b_a - n_a)}^{\text{Estimated true acceleration}} + g^n - (2\Omega_{ie}^n + \Omega_{en}^n) v_e^n \\ \underbrace{(\omega_{ib}^b - b_\omega - n_\omega)}_{\text{Estimated true turn rate}} q_j - R_n^b(\omega_{ie}^n + \omega_{en}^n) q_j \\ W_{b_a} \\ W_{b_\omega} \end{bmatrix}$$

Tracked states process model: nearly-constant velocity (NCV)

$$\begin{bmatrix} \dot{p}_{j'} \\ \dot{v}_{j'} \end{bmatrix} = \begin{bmatrix} v_{j'} \\ W_{ncv} \end{bmatrix}$$

$$x = \begin{bmatrix} x_j \\ x_{j'} \end{bmatrix} = \begin{bmatrix} p_j \\ v_j \\ q_j \\ b_{aj} \\ b_{\omega j} \\ p_{j'} \\ v_{j'} \end{bmatrix} \xrightarrow{\textbf{Calculate sigma points}} \mathcal{Y}_i = f(\mathcal{X}_i) = \begin{bmatrix} f_{own}(\mathcal{X}_j) \\ f_{track}(\mathcal{X}_{j'}) \end{bmatrix}$$

UKF Tutorial

- **Part 3:** Propagation of sigma points through process model

1) Initialize sigma points

Process noise states

Measurement noise states

$$\hat{x}_0^a = \begin{pmatrix} \hat{x}_0^T & 0 & 0 \end{pmatrix}^T$$

$$P_0^a = \begin{pmatrix} P_0 & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{pmatrix}$$

$$\chi_{k-1}^a = \begin{pmatrix} \hat{x}_{k-1}^a & \hat{x}_{k-1}^a \pm \sqrt{(n+\kappa)P_{k-1}^a} \end{pmatrix}$$

2) Propagate sigma points

$$\bar{\chi}_k^x = f(\chi_{k-1}^x, \chi_{k-1}^w)$$

$$\bar{x}_k = \sum_{i=0}^{2N} W_i^m \bar{\chi}_{i,k}^x$$

$$\bar{P}_k = \sum_{i=0}^{2N} W_i^c [\bar{\chi}_{i,k}^x - \hat{x}_{k-1}][\bar{\chi}_{i,k}^x - \hat{x}_{k-1}]^T$$

UKF Tutorial

- **Part 4:** Calculate expected measurements for each propagated sigma points

Nonlinear measurement functions

$$\bar{Y}_k = h(\bar{\chi}_k^x, \bar{\chi}_k^v)$$

$$\bar{y}_k = \sum_{i=0}^{2N} W_i^m \bar{Y}_k$$

GPS measurement, converted into NED frame

$$y_{gps} = \begin{bmatrix} y_{gps,n} \\ y_{gps,e} \\ y_{gps,d} \end{bmatrix} = \begin{bmatrix} x_n + \nu_n \\ x_e + \nu_e \\ x_d + \nu_d \end{bmatrix}$$

Barometer calibration constant

$$y_{baro} = -x_d + c_{baro} + \nu_{baro}$$

Forward earth-relative velocity

Wind speed

$$y_{pitot} = (R_n^b v)_1 + v_{wind} + \nu_{pitot}$$

yaw, calculated from quaternion (see appendix)

$$y_{mag} = \psi + \nu_{mag}$$

$$y_{range} = (x_{j'} - x_j) + \nu_{range}$$

UKF Tutorial

- **Part 5:** Perform measurement update based on actual measurements

$$P_{y_k y_k} = \sum_{i=0}^{2N} W_i^c [\bar{Y}_{i,k} - \bar{y}_k][\bar{Y}_{i,k-1} - \bar{y}_k]^T$$

$$P_{x_k y_k} = \sum_{i=0}^{2N} W_i^c [\bar{\chi}_{i,k} - \bar{x}_k][\bar{Y}_{i,k} - \bar{y}_k]^T$$

$$K = P_{y_k y_k} P_{x_k y_k}$$

$$\hat{x}_k = \hat{x}_{k-1} + K(z_k - \bar{y}_k)$$

$$P_k = P_{k-1} - K P_{y_k y_k} K^T$$

Cooperative Filter

COHRINT

Problem Setup

- Two Dubins agents moving with constant command velocity and turn rate, receiving GPS measurements and Cartesian range measurements of other agent (for tracking).

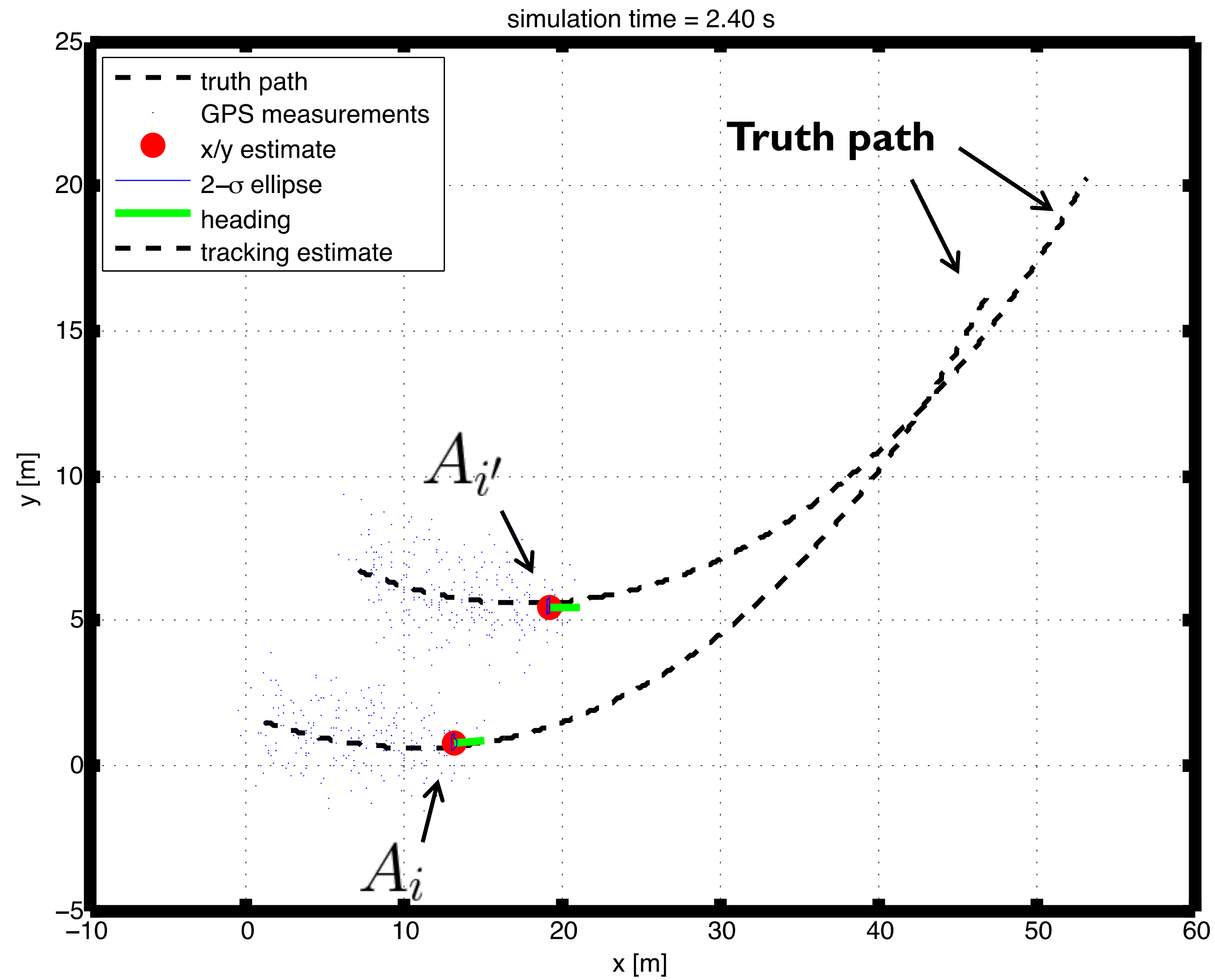
$$\hat{\mathbf{x}}^i = \begin{bmatrix} x^i \\ y^i \\ \theta^i \\ \hline x^{i'} \\ y^{i'} \\ \theta^{i'} \\ \hline v^{i'} \\ \omega^{i'} \end{bmatrix} \begin{array}{l} \textbf{Ownship pose} \\ \\ \textbf{Target pose} \\ \\ \textbf{Target inputs} \end{array}$$

$$\dot{x}^i = v_c(1+\eta_v)\cos\theta^i$$
$$\dot{y}^i = v_c(1+\eta_v)\sin\theta^i$$
$$\dot{\theta}^i = \omega_c(1+\eta_\omega)$$

$$\dot{x}^{i'} = v^{i'}(1+\eta_v)\cos\theta^{i'}$$
$$\dot{y}^{i'} = v^{i'}(1+\eta_v)\sin\theta^{i'}$$
$$\dot{\theta}^{i'} = \omega^{i'}(1+\eta_\omega)$$

$$\left.\begin{aligned} \dot{v}^{i'} &= \eta_{\dot{v}} \\ \dot{\omega}^{i'} &= \eta_{\dot{\omega}} \end{aligned}\right\}$$

***Nearly-Constant Velocity & Turn* (NCVT)**



DDF Approach #2: Factorized Covariance Intersection (FCI)

- First, partial state vanilla CI for this problem…

$$\hat{\mathbf{x}}^i = \left[\begin{array}{c} x^i \\ y^i \\ \theta^i \\ \hdashline x^{i'} \\ y^{i'} \\ \theta^{i'} \\ \hdashline v^{i'} \\ \omega^{i'} \end{array}\right] = \left[\begin{array}{c} \mathbf{a}^i \\ \hdashline \mathbf{b}^{i'} \\ \hdashline \mathbf{c}^{i'} \end{array}\right], \hat{\Sigma}^i = \begin{bmatrix} \mathbf{A}^i & \mathbf{D}^{ii'} & \mathbf{E}^{ii'} \\ (\mathbf{D}^{ii'})^T & \mathbf{B}^{i'} & \mathbf{F}^{i'i'} \\ (\mathbf{E}^{ii'})^T & \mathbf{F}^{i'i'} & \mathbf{C}^{i'} \end{bmatrix}$$

$$\hat{\mathbf{x}}^{i'} = \left[\begin{array}{c} x^{i'} \\ y^{i'} \\ \theta^{i'} \\ \hdashline x^{i} \\ y^{i} \\ \theta^{i} \\ \hdashline v^{i} \\ \omega^{i} \end{array}\right] = \left[\begin{array}{c} \mathbf{a}^{i'} \\ \hdashline \mathbf{b}^{i} \\ \hdashline \mathbf{c}^{i} \end{array}\right], \hat{\Sigma}^{i'} = \begin{bmatrix} \mathbf{A}^{i'} & \mathbf{D}^{i'i} & \mathbf{E}^{i'i} \\ (\mathbf{D}^{i'i})^T & \mathbf{B}^{i} & \mathbf{F}^{ii} \\ (\mathbf{E}^{i'i})^T & \mathbf{F}^{ii} & \mathbf{C}^{i} \end{bmatrix}$$

$\mathbf{a}$: ownship pose (shared)

$\mathbf{b}$: target pose (shared)

$\mathbf{c}$: target inputs (not shared)

Partial State CI

- For agent *i* (and analogously agent *i'*)…

$$\hat{\mathbf{x}}^i = \begin{bmatrix} x^i \\ y^i \\ \theta^i \\ x^{i'} \\ y^{i'} \\ \theta^{i'} \\ v^{i'} \\ \omega^{i'} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^i \\ \mathbf{b}^{i'} \\ \mathbf{c}^{i'} \end{bmatrix}, \hat{\Sigma}^i = \begin{bmatrix} \mathbf{A}^i & \mathbf{D}^{ii'} & \mathbf{E}^{ii'} \\ (\mathbf{D}^{ii'})^T & \mathbf{B}^{i'} & \mathbf{F}^{i'i'} \\ (\mathbf{E}^{ii'})^T & \mathbf{F}^{i'i'} & \mathbf{C}^{i'} \end{bmatrix} \longrightarrow$$

$v^{i'}, \omega^{i'}$ (not shared)

<u>Marginal Estimates</u>

$${}_m\hat{\mathbf{x}}^i = \begin{bmatrix} \mathbf{a}^i \\ \mathbf{b}^{i'} \end{bmatrix}, {}_m\hat{\Sigma}^i = \begin{bmatrix} \mathbf{A}^i & \mathbf{D}^{ii'} \\ (\mathbf{D}^{ii'})^T & \mathbf{B}^{i'} \end{bmatrix}$$

$${}^R_m\hat{\mathbf{x}}^{i'} = \begin{bmatrix} \mathbf{b}^i \\ \mathbf{a}^{i'} \end{bmatrix}, {}^R_m\hat{\Sigma}^{i'} = \begin{bmatrix} \mathbf{B}^i & \mathbf{D}^{ii'} \\ (\mathbf{D}^{ii'})^T & \mathbf{A}^{i'} \end{bmatrix}$$

Partial State CI

- Performing CI over marginal estimates…

$$({}_m\hat{\Sigma}^i_{CI})^{-1} = \omega({}_m\hat{\Sigma}^i)^{-1} + (1-\omega)({}^R_m\hat{\Sigma}^{i'})^{-1}$$

$$({}_m\hat{\Sigma}^i_{CI})^{-1}({}_m\hat{\mathbf{x}}^i_{CI}) = \omega({}_m\hat{\Sigma}^i)^{-1}({}_m\hat{\mathbf{x}}^i) + (1-\omega)({}^R_m\hat{\Sigma}^{i'})^{-1}({}^R_m\hat{\mathbf{x}}^{i'}))$$

Must combine estimates in information space

- … then adding new information to full estimates for agent *i*…

$$(\mathbf{G}^i)^{-1} = ({}_m\hat{\Sigma}^i_{CI})^{-1} - ({}_m\hat{\Sigma}^i)^{-1}$$

$$(\mathbf{G}^i)^{-1}\mathbf{g}^i = ({}_m\hat{\Sigma}^i_{CI})^{-1}({}_m\hat{\mathbf{x}}^i_{CI}) - ({}_m\hat{\Sigma}^i)^{-1}({}_m\hat{\mathbf{x}}^i)$$

$$\longrightarrow$$

$$(\hat{\Sigma}^i_{CI})^{-1} = (\hat{\Sigma}^i)^{-1} + \begin{bmatrix} (\mathbf{G}^i)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

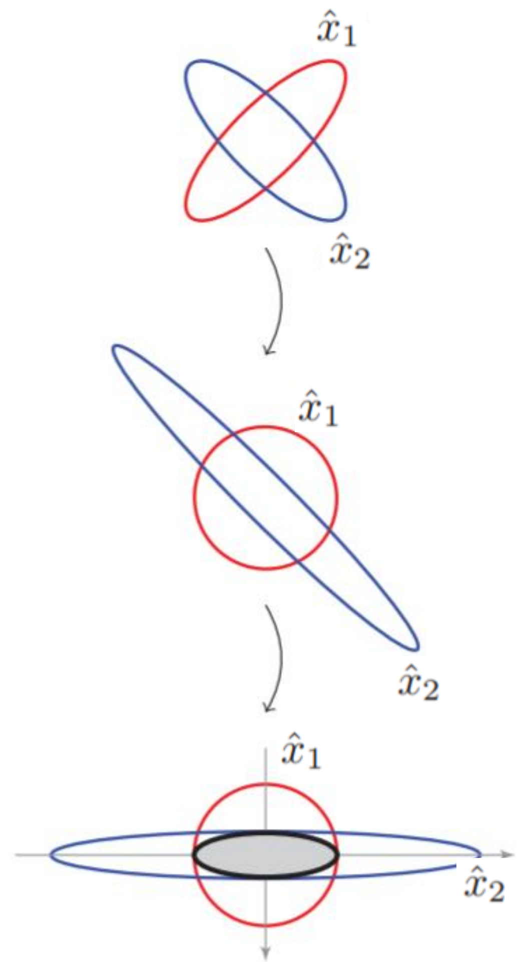
$$(\hat{\Sigma}^i_{CI})^{-1}\hat{\mathbf{x}}^i_{CI} = (\hat{\Sigma}^i)^{-1}\hat{\mathbf{x}}^i + \begin{bmatrix} (\mathbf{G}^i)^{-1}\mathbf{g}^i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Disco Cooperative Nav Approach

- Use Partial State Safe Fusion

$$x^j = \begin{bmatrix} x_j \\ x_{j'} \end{bmatrix} = \begin{bmatrix} p_j \\ v_j \\ q_j \\ b_{aj} \\ b_{\omega j} \\ p_{j'} \\ v_{j'} \end{bmatrix} \longrightarrow x_1 = \begin{bmatrix} p_j \\ v_j \\ p_{j'} \\ v_{j'} \end{bmatrix}$$

$$x^{j'} = \begin{bmatrix} x_{j'} \\ x_j \end{bmatrix} = \begin{bmatrix} p_{j'} \\ v_{j'} \\ q_{j'} \\ b_{aj'} \\ b_{\omega j'} \\ p_j \\ v_j \end{bmatrix} \longrightarrow x_2 = \begin{bmatrix} p_{j'} \\ v_{j'} \\ p_j \\ v_j \end{bmatrix}$$



Algorithm 1 Safe Fusion [9]

Given two possibly correlated estimates of x, $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$ such that $P_1 = \operatorname{cov}(\hat{\boldsymbol{x}}_1)$, and $P_2 = \operatorname{cov}(\hat{\boldsymbol{x}}_2)$:

1) Compute U_1 and D_1, using an SVD of the positive definite matrix P_1, such that

$$P_1 = U_1 D_1 U_1^T. \tag{6}$$

2) Similarly, derive U_2 and D_2 using an SVD, such that

$$D_1^{-1/2} U_1^T P_2 U_1 D_1^{-1/2} = U_2 D_2 U_2^T. \tag{7}$$

3) Let

$$T = U_2^T D_1^{-1/2} U_1^T \tag{8a}$$

$$\hat{\hat{\boldsymbol{x}}}_1 = T\hat{\boldsymbol{x}}_1 \qquad \hat{\hat{\boldsymbol{x}}}_2 = T\hat{\boldsymbol{x}}_2, \tag{8b}$$

where by construction $\operatorname{cov}(\hat{\hat{\boldsymbol{x}}}_1) = I$ and $\operatorname{cov}(\hat{\hat{\boldsymbol{x}}}_2) = D_2$.

4) Select the most informative source for each component $i = 1, 2, \ldots, \dim(x)$, let

$$[\hat{\hat{\boldsymbol{x}}}]_i = [\hat{\hat{\boldsymbol{x}}}_1]_i, \quad [D]_{ii} = 1 \quad \text{if} \quad [D_2]_{ii} \geq 1, \tag{9a}$$

$$[\hat{\hat{\boldsymbol{x}}}]_i = [\hat{\hat{\boldsymbol{x}}}_2]_i, \quad [D]_{ii} = D_2^{ii} \quad \text{if} \quad [D_2]_{ii} < 1. \tag{9b}$$

5) The final estimate given by

$$\hat{\boldsymbol{x}}_f = T^{-1}\hat{\hat{\boldsymbol{x}}} \tag{10a}$$

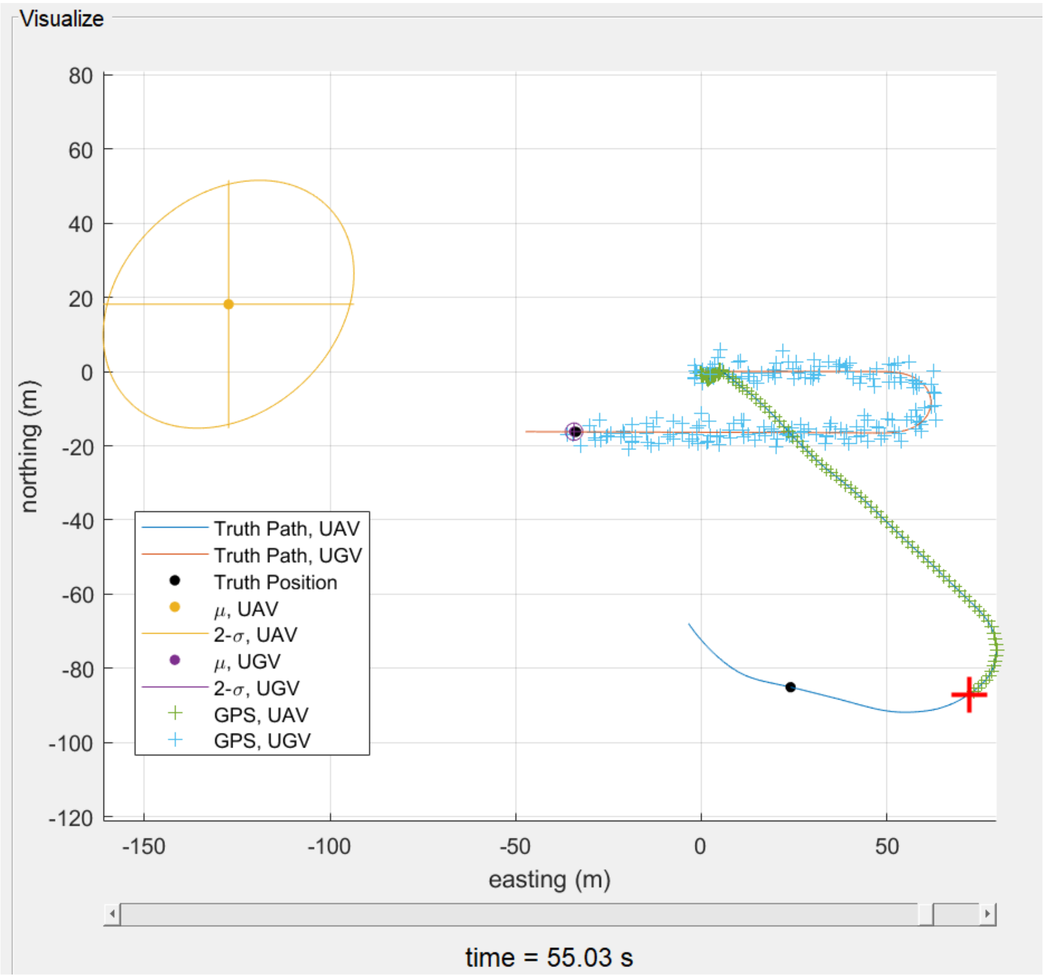
$$P_f = T^{-1} D^{-1} T^{-T}. \tag{10b}$$

Results

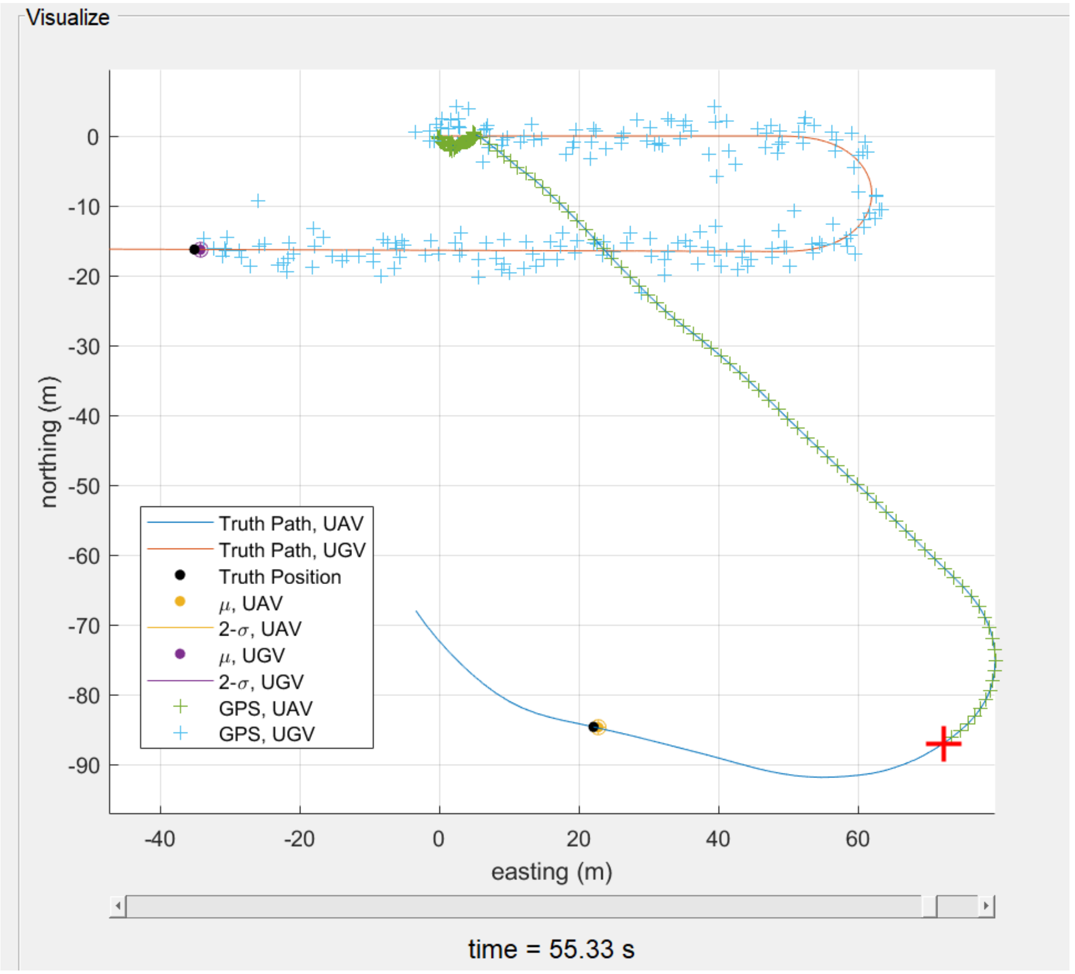
COHRINT

Results using Disco flight data and simulated UGV

Nav. filter only



Nav. + Tracking filters with Safe Fusion



Appendix

COHRINT

Quaternion Math - representation

$$\mathbf{q}_i^b = \begin{pmatrix} a & b & c & d \end{pmatrix}^T \longrightarrow \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \cos(0.5\theta) \\ v_x \sin(0.5\theta) \\ v_y \sin(0.5\theta) \\ v_z \sin(0.5\theta) \end{pmatrix}$$

$\begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T \longrightarrow$ Axis of rotation

$\theta \longrightarrow$ Angle of rotation about that axis

http://www.chrobotics.com/library/understanding-quaternions

Quaternion Math - transformations

$$\mathbf{q}_i^b = \begin{pmatrix} a & b & c & d \end{pmatrix}^T$$

$$R_i^b(\mathbf{q}_i^b) = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{pmatrix}$$

Quaternion to rotation mtarix

$$\phi = \arctan\left(\frac{2(ab+cd)}{a^2-b^2-c^2+d^2}\right),$$

$$\theta = -\arcsin(2(bd-ac)), \text{ and}$$

$$\psi = \arctan\left(\frac{2(ad+bc)}{a^2+b^2-c^2-d^2}\right).$$

Quaternion to Euler angles

http://www.chrobotics.com/library/understanding-quaternions

Quaternion Math – rate of change

$$\omega = (\omega_x, \omega_y, \omega_z)^T$$ → Turn rates about body axis

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}_B \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

https://www.princeton.edu/~stengel/MAE331Lecture9.pdf

Interpreting Disco Measurements – Axes Transformations

Measure gravity along each axis → $R_a^b = \mathbf{I}$

Measure turn rates about each axis → $R_\omega^b = \mathbf{I}$

Measure direction of North along each axis → $R_m^b = \mathbf{I}$