KALMAN FILTER

Chun-Kit Law

Ali Demir

Alireza Ahmadi

15 January 2018

universitätbonn

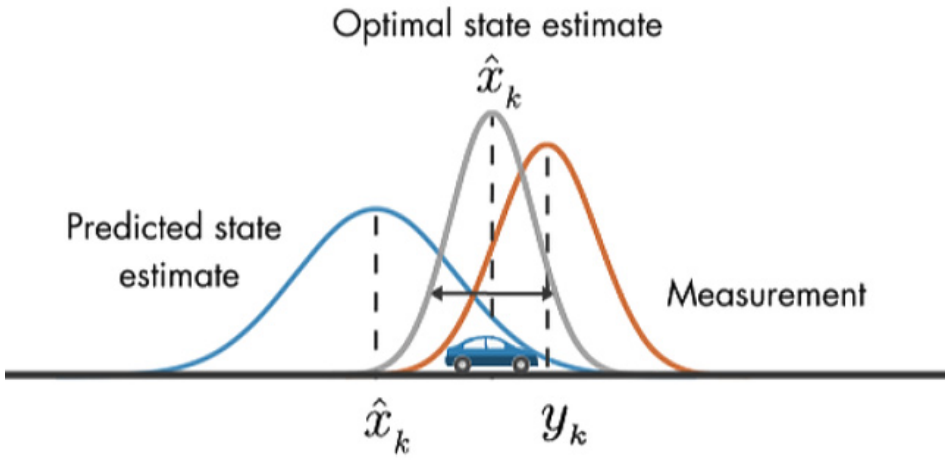
Task Description

- Determine of the trajectory of a moving platform by using a Kalman filter combining the prediction of the system model and the measurements of GPS, the accelerometer and the gyroscope

- Compare the results of the Kalman filter with the results of the Strapdown integration from the previous exercise

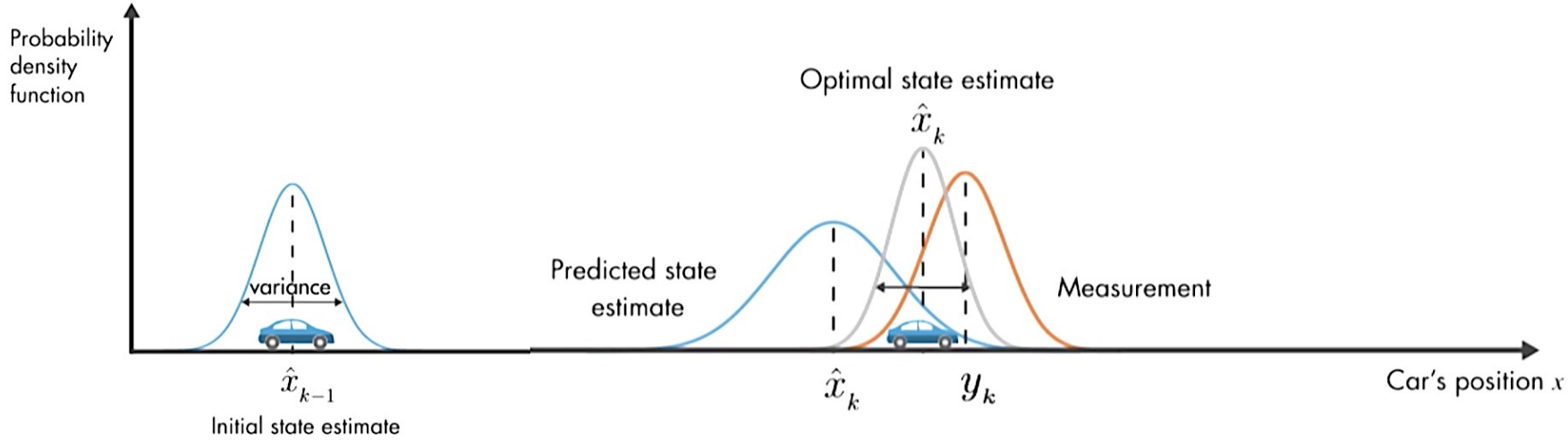
Idea

- **Kalman filter** is an **application** of **Bayesian estimation** technique.
- It obtains **optimal estimates** using the **deterministic** and **stochastic** properties of the **system model** and **measurements** to execute **recursive state estimation**.
- A **recursive algorithm**: The current estimate is updated by using the previous best estimates as inputs
- **System model** provides a **prediction** of the **current state**
- **Measurements** are used to **correct** the **predicted state**



idea

- The **predicted result** and the **measurements** are combined by assigning **weights** to the prediction and the measurements
- The **new estimate** is the **weighted mean** of the **predicted state** and the measurements.
- **Combination** of these two **independent estimates** of a particular variable to get the mean value is the core of the Kalman filtering process.



universitätbonn

Requirements of Kalman Filter

- System model **linear** in the previous state and the control parameters
- Measurement model **linear** in current state

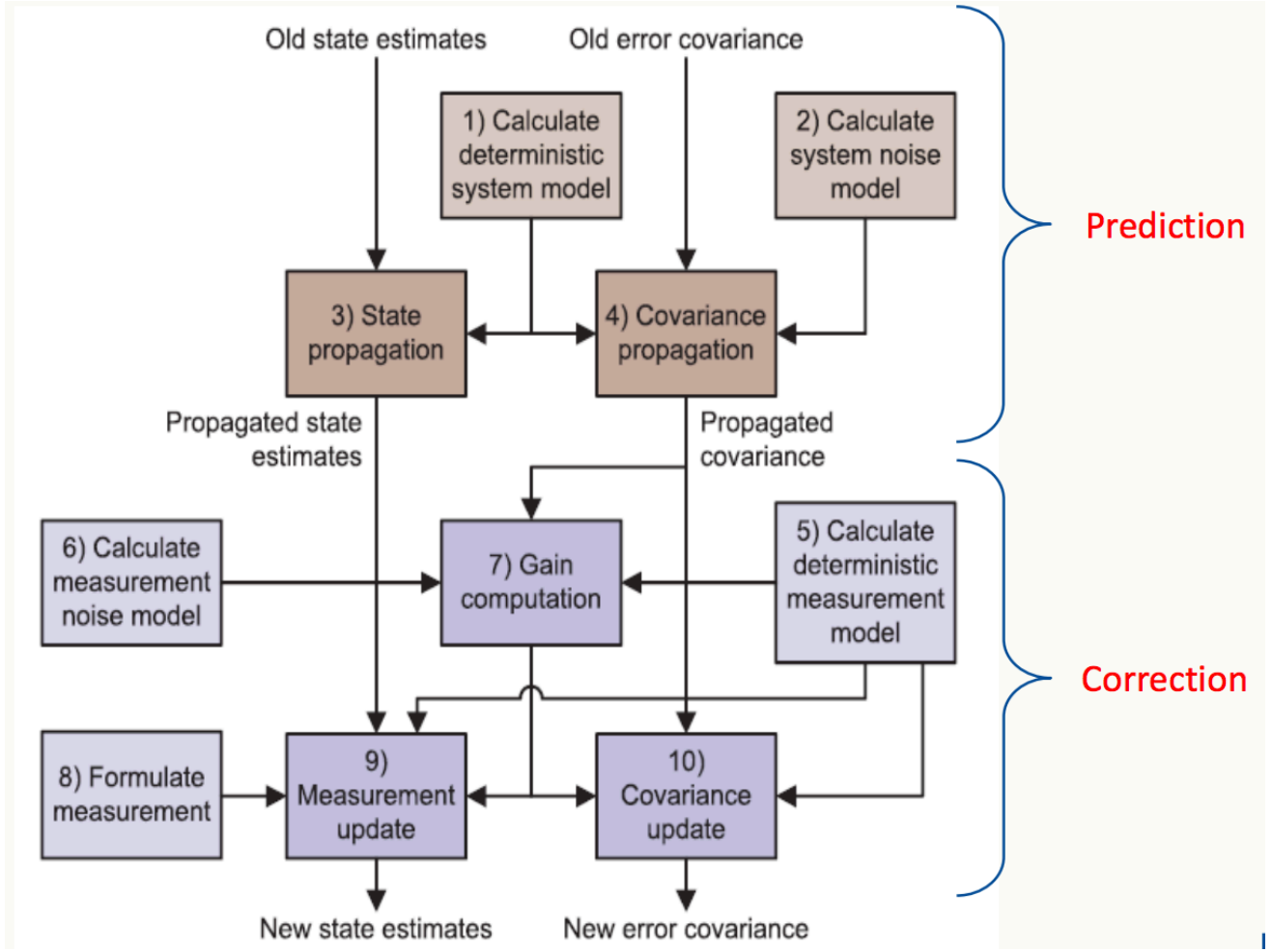
In the case of **extended Kalman Filter**,

the requirement on **linearity** on the system model and the measurement model may be **relaxed**.

An additional requirement:

Linearise the non-linear system model and/or the non-linear measurement model

Flow Diagram



universitätbonn

Predefined Parameters

Input parameters which need to be predefined:

X_0^- : **Initial state of the system (deterministic)**

P_0^- : **Covariance matrix of the initial state (stochastic)**

Parameters which need to be predefined for the algorithm:

A : Matrix which describes the evolution of the state without controls or noise (deterministic)

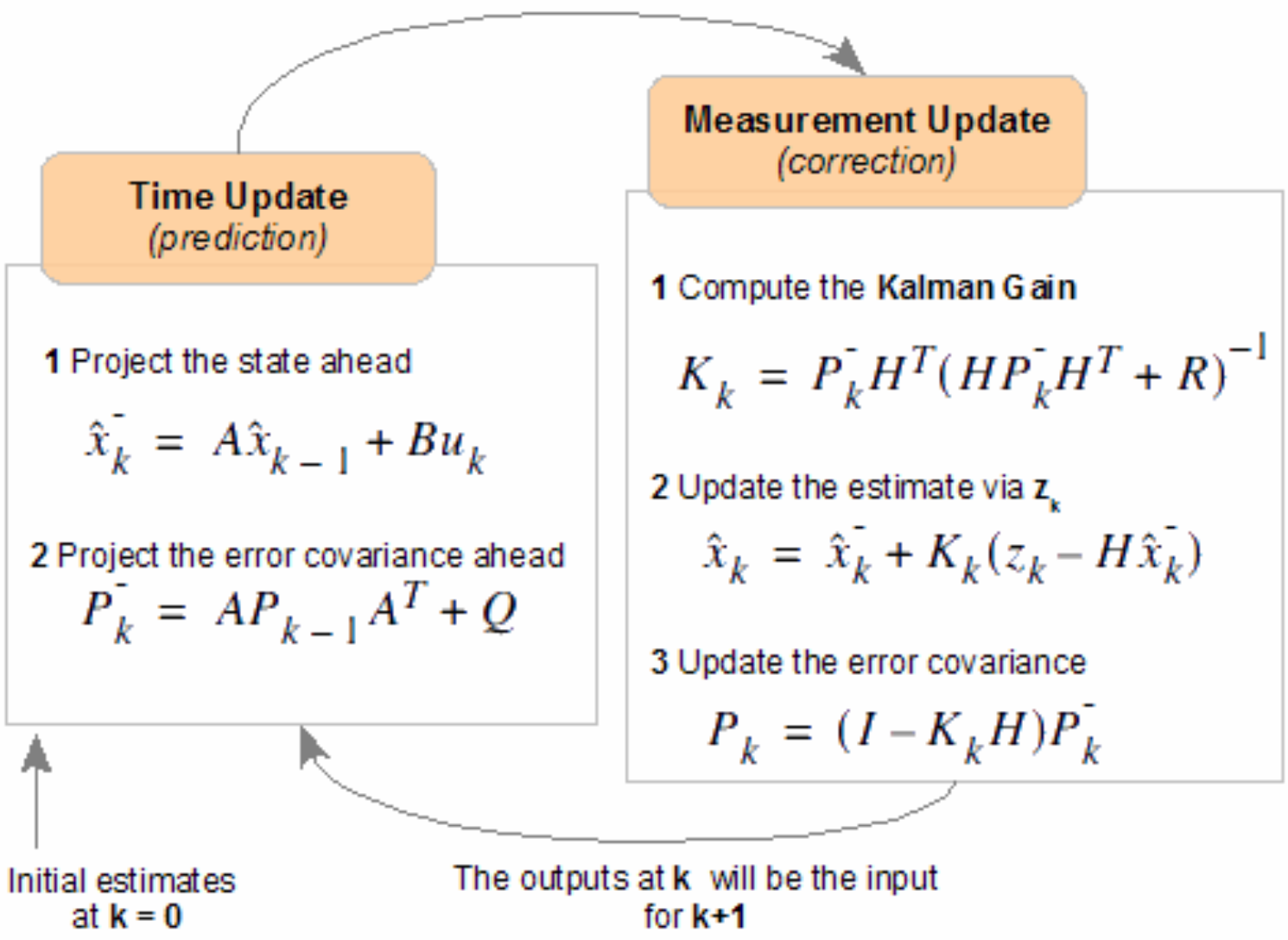
B : Matrix that describes how the control evolves the state (deterministic)

H : Matrix that describes how to map the state to an observation (deterministic)

Q : **System noise model (stochastic)**

R : **Measurement noise model (stochastic)**

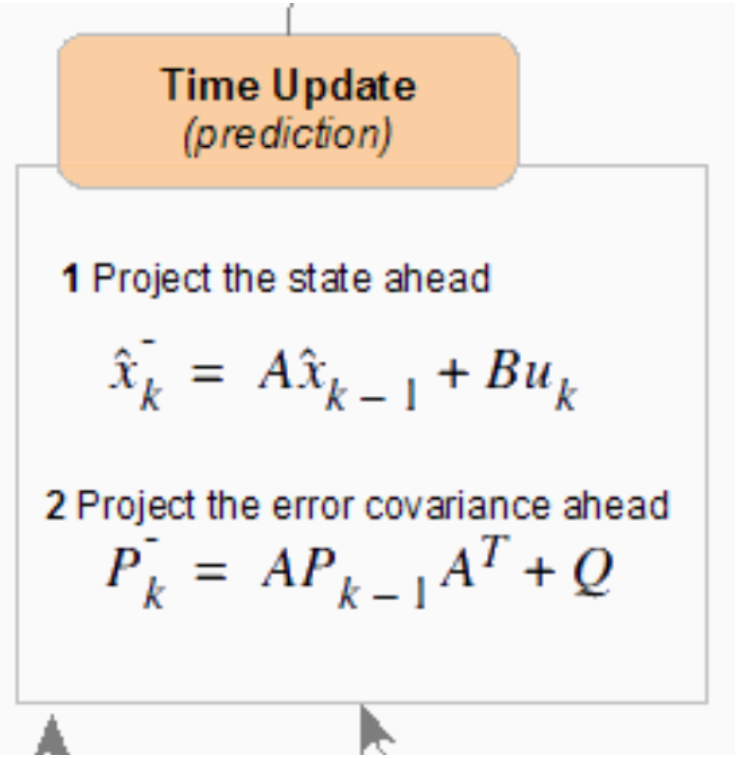
Computation Details



Two key steps:
1)Prediction stage
2)Correction stage

Prediction stage

- $\hat{x}_{k-1}$: input state
- P_{k-1}: Covariance matrix of the input state
- $\hat{x}_k^-$: Predicted state
- P_k^-: Covariance matrix of the predicted state
- u_t: Control
- Q: System noise
- A: Matrix that describes how the state evolves from time k-1 to k without controls or noise
- B: Matrix that describes how the control changes the state from time k-1 to k



Correction stage

- $\hat{x}_k^+$: Output state after combining the prediction with the measurements
- P_k^+: Covariance matrix of the output state
- z_k: Measurement vector at time k
- H: Matrix that describes how to map the state to an observation .
- K_k: Kalman gain
- R: Measurement noise

Measurement Update
(correction)

1 Compute the **Kalman Gain**

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

2 Update the estimate via z_k

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-)$$

3 Update the error covariance

$$P_k = (I - K_k H) P_k^-$$

Influence of the predefined stochastic settings

- **Kalman gain** assigns the **weights** to the prediction and the measurements
- The **criteria** for assigning the **weights** is the **level** of **precision**
- This is achieved by comparing the noise matrices **R** and **Q** in the two steps
- Visualise **R** and **Q** as scalars
- Big **R** but small **Q** implies the prediction is not precise but the measurements are
- High weight on **Q** but small weight on **R**

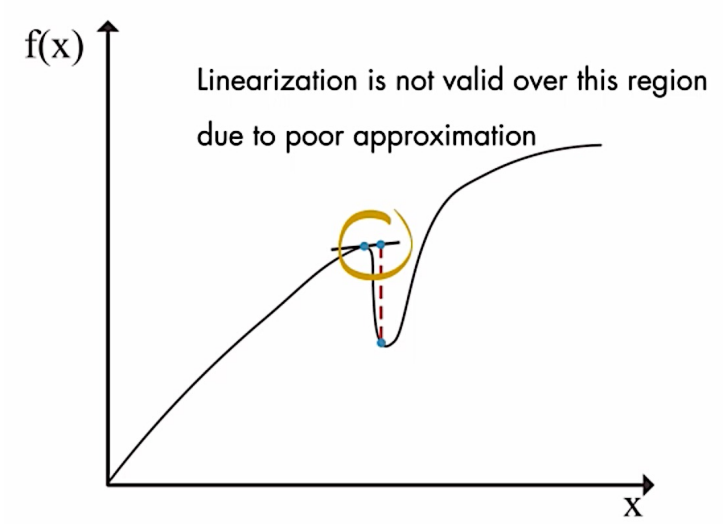
Kalman Filter (a real-time algorithm)

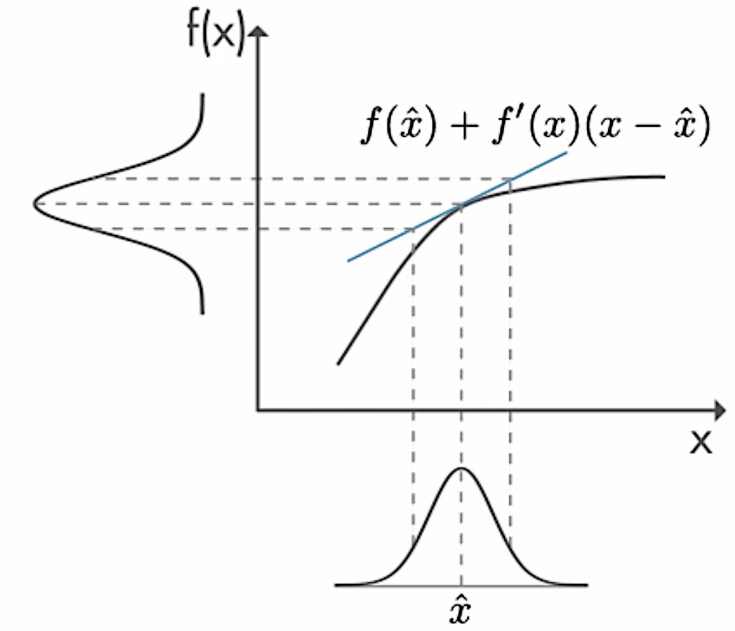
- Number of iterations: n

  where n is the total number of time steps we set for the inputs
- Runtime~O(n)
- Total **runtime** does not grow faster with the input size
- The runtime of each of the later steps remains bounded
- Its Possible for the **inputs** and **outputs** to be processed **simultaneously**

Extended Kalman filter

- Relax the condition on the linearity of the motion model and the sensor model
- Requires the application of linearisation of the non-linear models for the update of the covariance matrixes
- The rest is similar to Kalman filter





Extended kalman filter

Prediction

$$\begin{aligned} \mathbf{x}_k^- &= f(\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \end{aligned}$$

Correction

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \end{aligned}$$

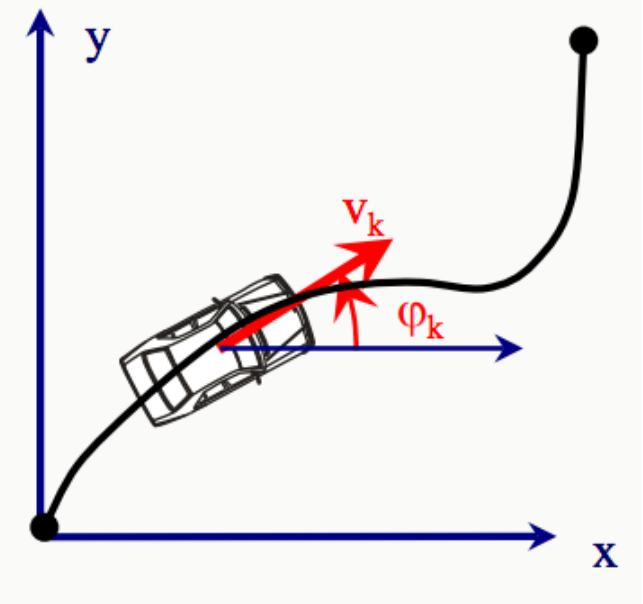
$$\mathbf{\Phi} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}, \mathbf{u}=\mathbf{u}_k}, \quad \mathbf{G} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{w}}\bigg|_{\dots} \quad \mathbf{H}_k = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}_k^-}$$

universitätbonn

Simplified 2D motion model

Assumptions:

- Constant acceleration
- Constant angular rate



Motion Model:

$$\begin{bmatrix} x_k \\ y_k \\ \phi_k \\ \omega_k \\ v_k \\ a_k \end{bmatrix} = f(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{W}) = \begin{bmatrix} x_{k-1} + v_{k-1}\Delta t \cos(\phi_{k-1}) \\ y_{k-1} + v_{k-1}\Delta t \sin(\phi_{k-1}) \\ \phi_{k-1} + \omega_{k-1}\Delta t \\ \omega_{k-1} + W_\omega \Delta t \\ v_{k-1} + a_{k-1}\Delta t \\ a_{k-1} + W_a \Delta t \end{bmatrix}$$

W_ω
W_a : System noise

Prediction Stage:

1) State vector

Prediction

$$\begin{aligned} \mathbf{x}_k^- &= f(\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \end{aligned}$$

The state vector is updated by taking the expectation value of the prediction of the motion model.

$$E[W_\omega] = 0$$

$$E[W_a] = 0$$

$$\begin{bmatrix} x_k \\ y_k \\ \phi_k \\ \omega_k \\ v_k \\ a_k \end{bmatrix} = E[f(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{W})] = \begin{bmatrix} x_{k-1} + v_{k-1}\Delta t \cos(\phi_{k-1}) \\ y_{k-1} + v_{k-1}\Delta t \sin(\phi_{k-1}) \\ \phi_{k-1} + \omega_{k-1}\Delta t \\ \omega_{k-1} \\ v_{k-1} + a_{k-1}\Delta t \\ a_{k-1} \end{bmatrix}$$

Prediction Stage:

2) Covariance Matrix

Prediction

$$\begin{aligned} \mathbf{x}_k^- &= f(\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \end{aligned}$$

Transition Matrix — Obtained by linearisation with respect to the state vector

$$\Phi = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}, \mathbf{u}=\mathbf{u}_k}$$

$$= \begin{bmatrix} 1 & 0 & -\sin(\varphi_{k-1})v_{k-1}\Delta t & 0 & \cos(\varphi_{k-1})\Delta t & 0 \\ 0 & 1 & \cos(\varphi_{k-1})v_{k-1}\Delta t & 0 & \sin(\varphi_{k-1})\Delta t & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & & 1 \end{bmatrix}_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}}$$

universitätbonn

Prediction Stage:

2) Covariance Matrix

Prediction

$$\begin{aligned} \mathbf{x}_k^- &= f(\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \end{aligned}$$

Matrix G — Obtained by linearisation with respect to the noise vector

$$\mathbf{G} = \frac{\partial f}{\partial \mathbf{W}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & 0 \\ 0 & \Delta t \end{bmatrix}$$

Prediction Stage:

2) Covariance Matrix

$$\begin{aligned} \mathbf{x}_k^- &= f(\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \end{aligned}$$

Prediction

System noise — predefined

$$\mathbf{Q} = \begin{bmatrix} \sigma_{\dot{\varphi}}^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}$$

Covariance matrix of the predicted state vector:

$$\mathbf{P}_k^- = \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T$$

Model does not match reality…

- Measurements have high precisions — Low variance
- BUT inaccurate — with large systematic bias
- Kalman filter assigns weight to different stages based on their relative levels of precision
- The output state will deviate significantly from the true state
- Because we put a lot of weight on the wrong model

Measurement model

Types of Measurements:

- GPS measurements of the position
- Non-gravitational acceleration/specific force measurements with the accelerometer
- Angular rate measurements with the gyroscope

Put these in a measurement vector **z**

$$z_k = \begin{bmatrix} x_{\text{gps},k} \\ y_{\text{gps},k} \\ a_{\text{acc}},k \\ \omega_{\text{gyro}},k \end{bmatrix}$$

Correction Stage

Assume the true state is known,
the measurement and the true state are related by

$$z_k = h(x_k)$$

As the measurement model is linear,

$$z_k = h(x_k) = Hx_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k$$

Correction Stage

1) Computing the Kalman Gain

Measurement noise matrix:

$$R = \begin{bmatrix} \sigma_{\text{gps},x} & 0 & 0 & 0 \\ 0 & \sigma_{\text{gps},y} & 0 & 0 \\ 0 & 0 & \sigma_{\text{acc}} & 0 \\ 0 & 0 & 0 & \sigma_{\text{gyro}} \end{bmatrix}$$

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H})\mathbf{P}_k^- \end{aligned}$$

Kalman Gain

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1}$$

Correction Stage

2) Updating the state with the measurements

$$
\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1} \\
\mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \\
\mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-
\end{aligned}
$$

Since the measurement model is linear,

$$
h(x_k^-) = Hx_k^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{x}_k^- & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}
$$

universitätbonn

Correction Stage

2) Updating the state with the measurements

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-))$$

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \end{aligned}$$

Note that:

- In the event of very large measurement noise R,
- Kalman gain tends to 0
- The Kalman output of the state is simply the predicted state
- All the weight is on the prediction step.

Correction Stage

3) Updating the covariance matrix of the state

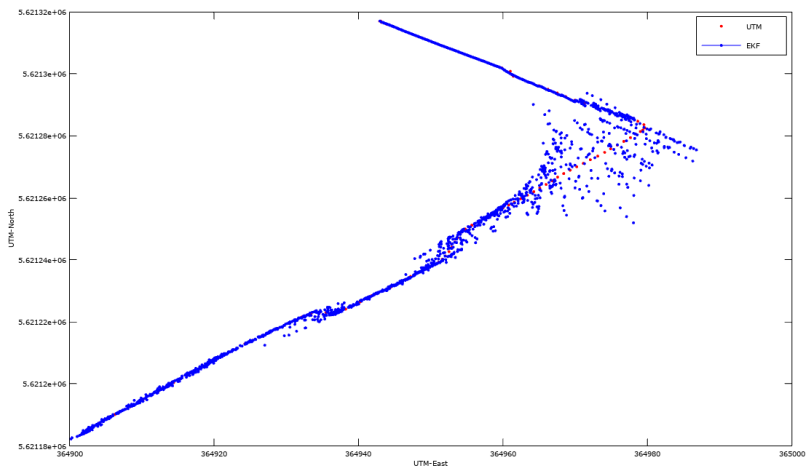
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

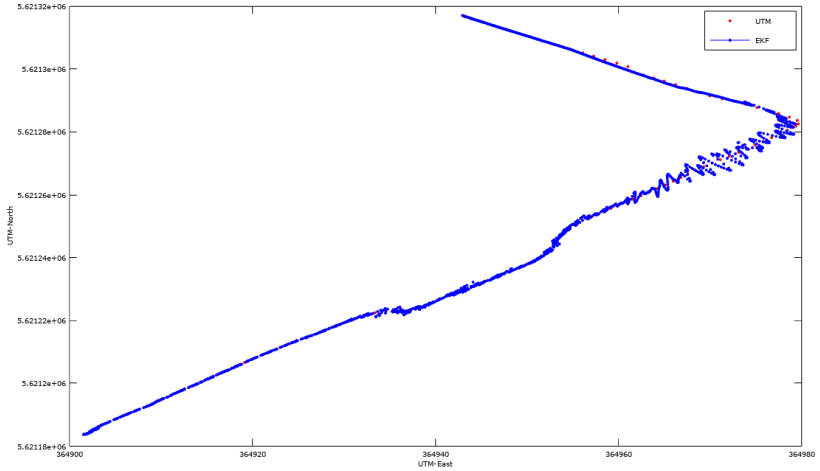
$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \\ \mathbf{x}_k &= \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_k^-)) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-\end{aligned}$$

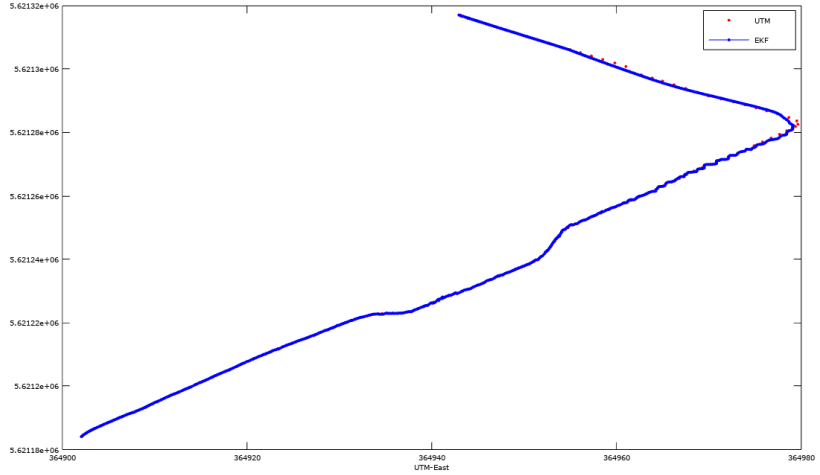
Filter Tuning

Turning process adjusts a **trade-off** between trusting to **predictions** and **measurements**.

- This process has to be done through adjusting measurement and system noise models **R** and **Q.**
- In most of the cases **R** cold be set based on sensor specifications. (**it better to assume a bit more than values in datasheets**)
- elements of **Q** have to set in an stochastic form (**they should be more than one**), depending on how much you think, the system model is precis or in could be affected by misalignment respect to real measurements and error sources.



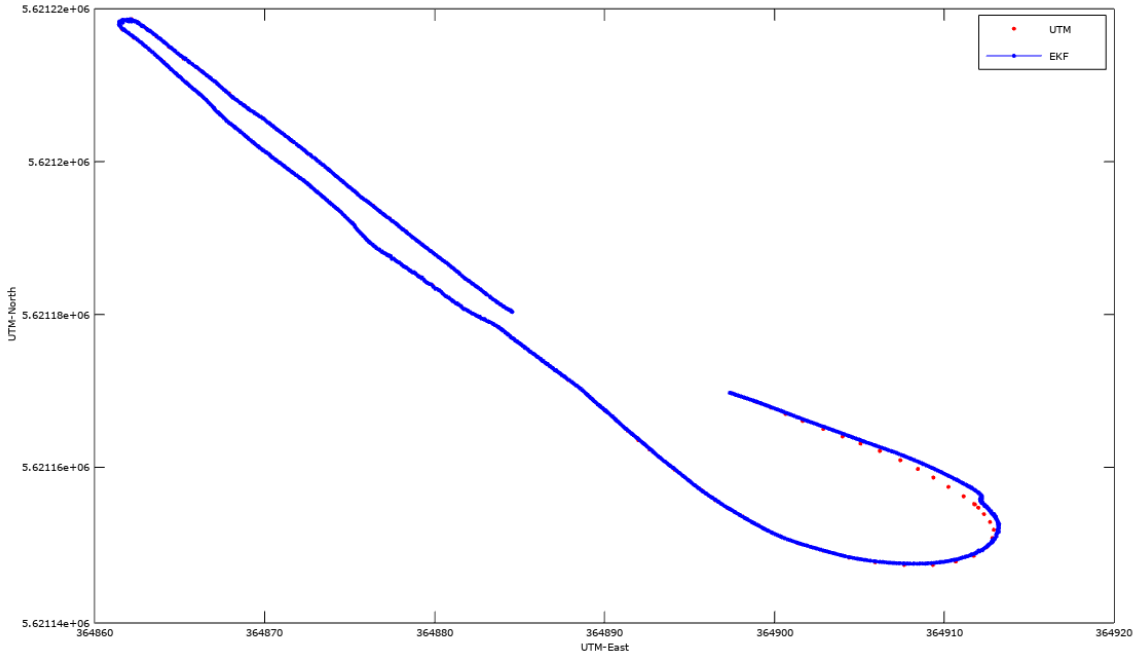




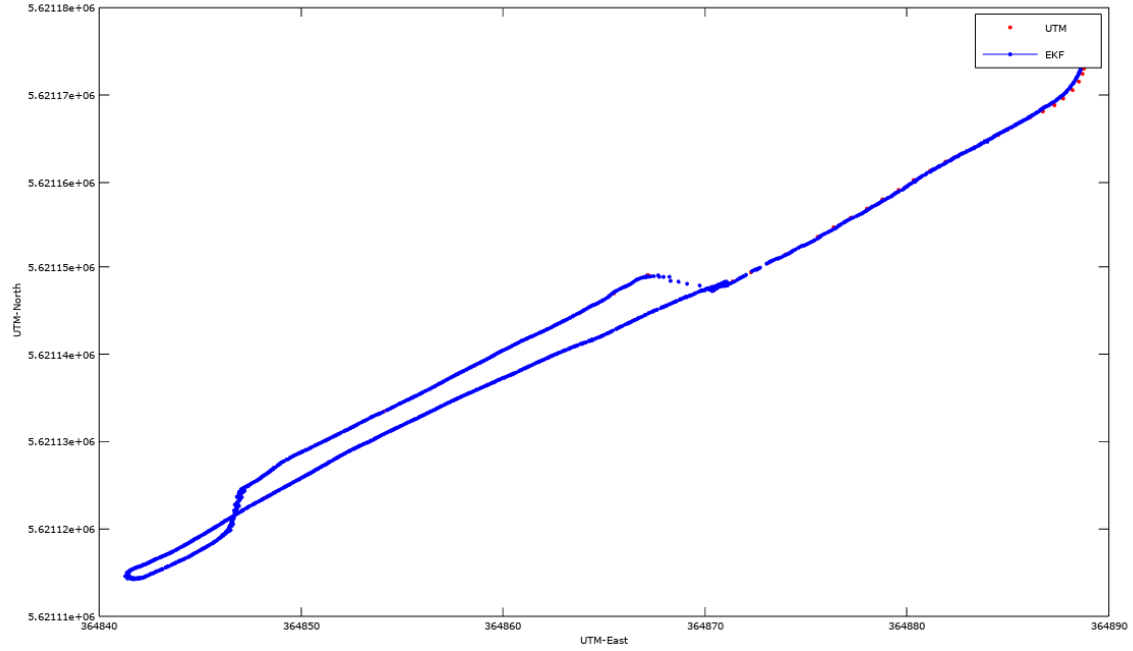
Filter tuning

Tactical- and Aviation-Grade INS Characteristics		
Sensor grade	**Tactical**	**Aviation**
Initial position error SD	10 m	10 m
Initial velocity error SD	0.1 m s^{-1}	0.01 m s^{-1}
Initial (roll and pitch) attitude error SD	1 mrad	0.1 mrad
Accelerometer bias SD	0.01 m s^{-2} (1 mg)	0.001 m s^{-2} (0.1 mg)
Gyro bias SD	5×10^{-5} rad s^{-1} (10° h^{-1})	5×10^{-8} rad s^{-1} (0.01° h^{-1})
Accelerometer noise PSD	10^{-6} m^2 s^{-3} (100°g/$\sqrt{Hz}$)^2	10^{-7} m^2 s^{-3} (32°g/$\sqrt{Hz}$)^2
Gyro noise PSD	10^{-9} rad^2 s^{-1} (0.1°/$\sqrt{h}$)^2	10^{-12} rad^2 s^{-1} (0.003°/$\sqrt{h}$)^2

Results IMAR0000-IMAR0001

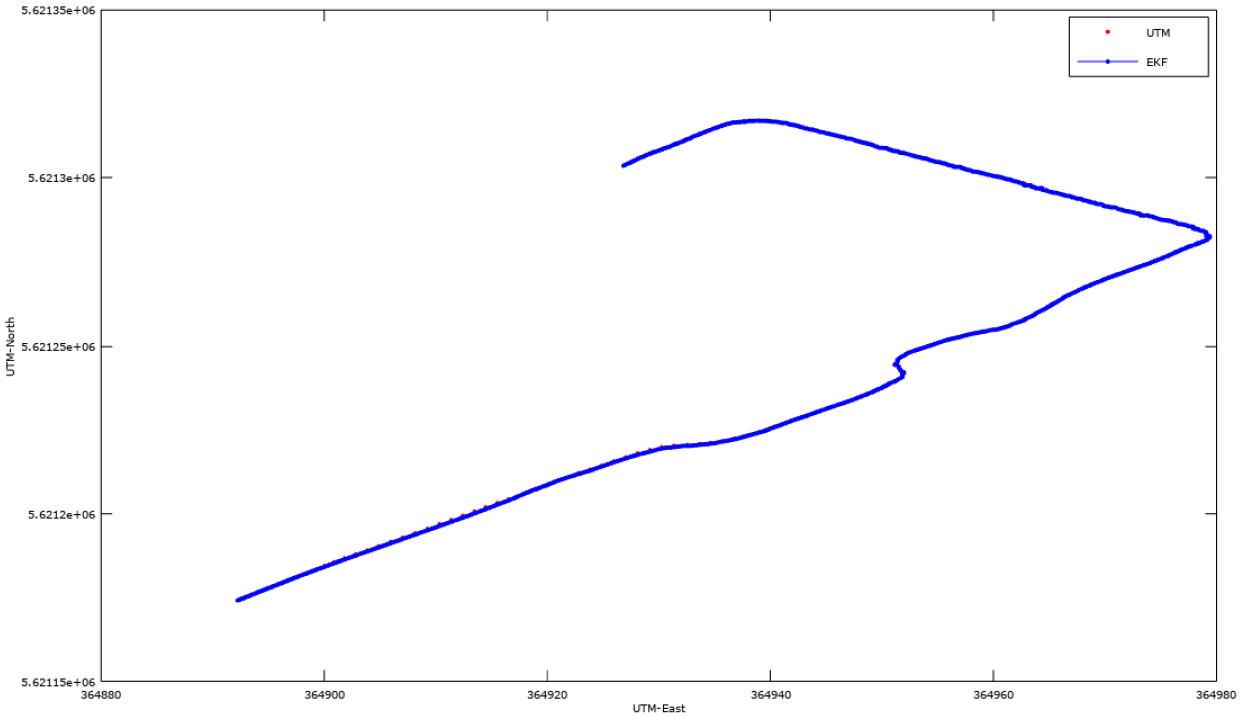


IMAR0000

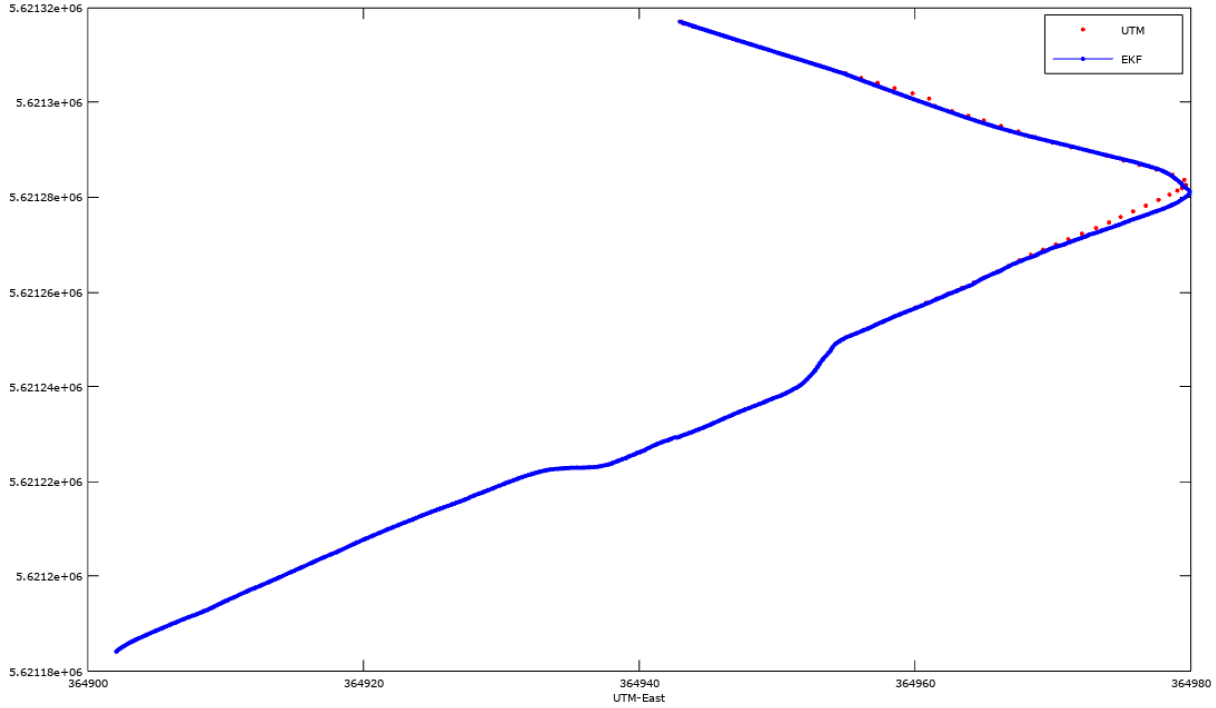


IMAR0001

Results IMAR0002-IMAR0003



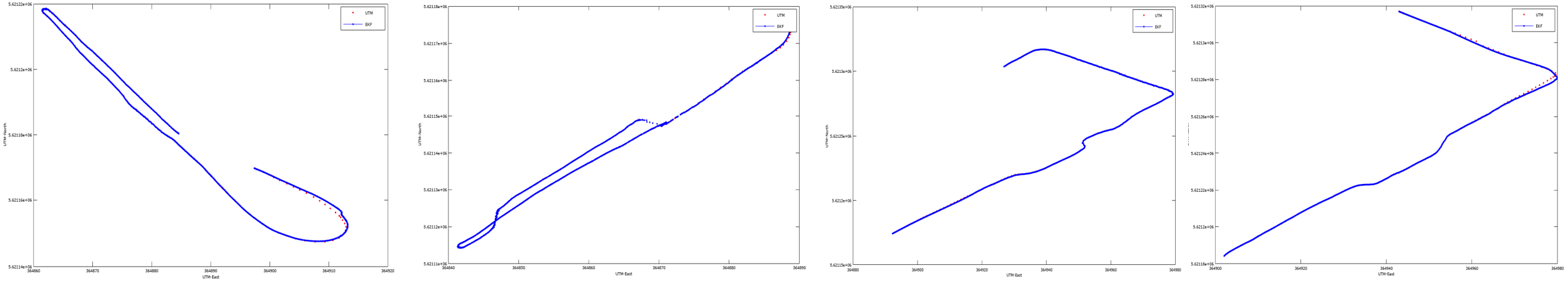
IMAR0002

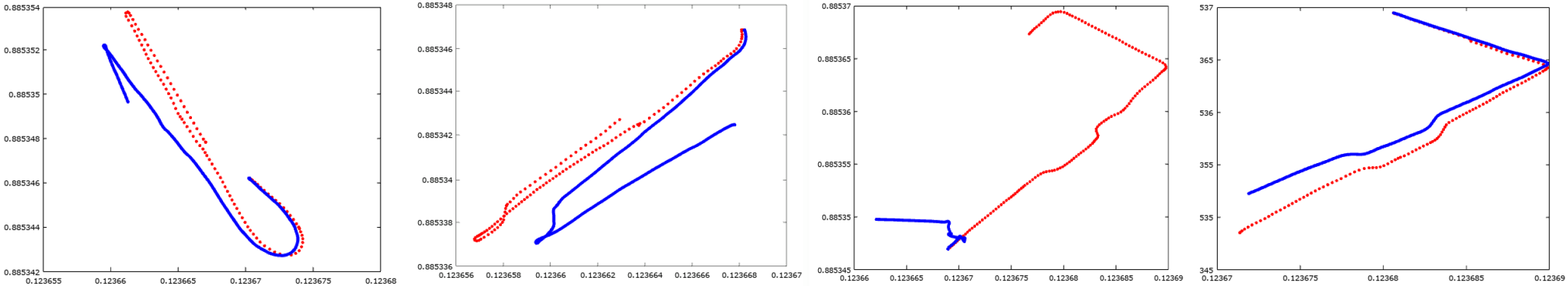


IMAR0003

Comparison of result with INS approach

EKF Results





INS Results

Comparison

The results from the **strapdown** integration **starts to deviate** from the measurements by GPS positioning **very quickly.**

The **strapdown** integration was **unable** to **track the trajectory** accurately when there is a **sharp turn.**

Despite some observable deviations, the computation by **Kalman filter agrees** with the **GPS** measurements **very well.**

Manages to **follow** the trajectory without **losing** its **accuracy** in the event of a **sharp turn.**

Why combine the two?

- Predictions based on the motion model can be output at extremely small time intervals
- However, the **system error accumulates** and the **updated state loses** its **precision** very quickly

- **Measurements** does not suffer from this **growth** of **error**.
- However, the measurements can only be made in much larger time intervals

By combining the prediction of the motion model and the measurements,

we can have data concerning the state at a very small time interval while limiting the growth in error