Sensor Data Fusion

Exercise 10

Prof. Dr.-Ing. Marcus Baum
M.Sc. Kolja Thormann

www.fusion.informatik.uni-goettingen.de

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

DATA
FUSION Lab

Today

- Lecture review
- Questions
- Homework 9 solution
- Example - EKF issues
- Example - Multiplicative noise with EKF
- Problem 10 - Radar measurements with EKF
- Homework 10 presentation

Questions

What is are disadvantages of the EKF?

- Might diverge for severe nonlinearities
- Need to compute Jacobians

Questions

Can you explain the white noise velocity model?

- State p_k
- Velocity as white noise $\dot{p}_k = w_k \sim \mathcal{N}(0, \sigma_w^2)$
- $p_{k+1} = p_k + T w_k$ with $T = t_{k+1} - t_k$

Questions

What is the assumption of the noise during sampling periods in the white noise acceleration model?

The noise is constant during sampling periods.

Homework 9: solution

Assume a robot in 2D-space at position $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}}$ moving with velocity v_1 in x_1 direction and v_2 in x_2 direction. Its state is defined as $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & v_1 & v_2 \end{bmatrix}^{\mathrm{T}}$.

a) Formulate a motion model for the robot, assuming independent zero-mean Gaussian noise on each state element. Write a function implementing the motion model.

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \nu, \text{ with } \nu \sim \mathcal{N}(0, \mathbf{Q}) \text{ and } \mathbf{F} = \begin{bmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) In each time step, a sensor measures the robot's position. Formulate a measurement equation assuming independent zero-mean Gaussian noise and implement it as well.

$$y = \mathbf{H}\mathbf{x}_k + \mu, \text{ with } \mu \sim \mathcal{N}(0, \mathbf{R}) \text{ and } \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Homework 9: solution

c) Use the functions from a) and b) to implement a simulation using initial state $\hat{\mathbf{x}}_{\text{init}} = \begin{bmatrix} 0\text{m} & 0\text{m} & 2\text{m}\,\text{s}^{-1} & 2\text{m}\,\text{s}^{-1} \end{bmatrix}$ (for each run, draw the true state from the prior), 10 time steps of length 1s, and covariances for the initial state $\mathbf{C}_{\text{init}}$, the transition noise $\mathbf{Q}$ and the measurement noise $\mathbf{R}$ as

$$\mathbf{C}_{\text{init}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} .$$

Homework 9: solution

d) Now based on a) and b), implement a predict and an update function for a Kalman filter. Use the Kalman filter to track the robot simulated with your function from c).

$$\mathbf{x}_{k+1|k} = \mathbf{F}\mathbf{x}_{k|k}$$
$$\mathbf{C}_{k+1|k} = \mathbf{F}\mathbf{C}_{k|k}\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$$

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + \underbrace{\mathbf{C}_{k+1|k}\mathbf{H}^{\mathrm{T}}(\overbrace{\mathbf{H}\mathbf{C}_{k+1|k}\mathbf{H}^{\mathrm{T}} + \mathbf{R}}^{\mathbf{S}})^{-1}}_{\mathbf{K}}(y - \mathbf{H}\mathbf{x}_{k+1|k})$$
$$\mathbf{C}_{k+1|k+1} = \mathbf{C}_{k+1|k} - \mathbf{K}\mathbf{S}\mathbf{K}^{\mathrm{T}}$$

Homework 9: solution

e) Write a function which calculates the root mean square error (RMSE) of $n = 100$ simulation runs with the error as the Euclidean norm at the last time step.

f) Finally, assume a worse sensor with noise covariance

$$\mathbf{R}_2 = \begin{bmatrix} 2.0 & 0 \\ 0 & 0.2 \end{bmatrix} .$$

Calculate the RMSE as in e). To deal with the noise, add a second sensor. Assume the measurements to be independent of each other and update the simulation and Kalman filter accordingly and observe the difference in RMSE. Now, instead of using the same type of sensor, use the sensor with $\mathbf{R}_2$ along a second sensor with noise covariance

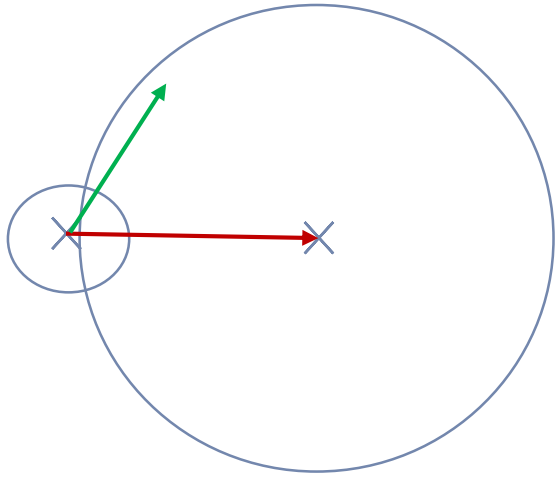
$$\mathbf{R}_3 = \begin{bmatrix} 0.2 & 0 \\ 0 & 2.0 \end{bmatrix} .$$

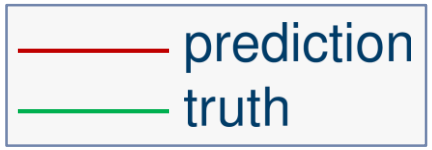
Example – EKF issues

Consider the state $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & v_1 & v_2 \end{bmatrix}^{\mathrm{T}}$ consisting of position and velocity vectors. It uses the discrete linear motion model $\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \nu$ to move t seconds into the future with

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .$$

Due to the shape of the transition matrix, the uncertainty in the velocities influences the corresponding position uncertainty. Looking at the following scenario with high velocity uncertainty in both directions (as they are both completely wrong), we see that the covariance handles the true position being far off.





Example – EKF issues

Now consider the state $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & v & \alpha \end{bmatrix}^{\mathrm{T}}$ consisting of position and polar velocity with speed v and orientation α. It moves with the discrete non-linear motion model

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) = \begin{bmatrix} x_1 + v\cos(\alpha) & x_2 + v\sin(\alpha) & v & \alpha \end{bmatrix}^{\mathrm{T}} + \nu \ .$$

The EKF would linearize using Jacobian

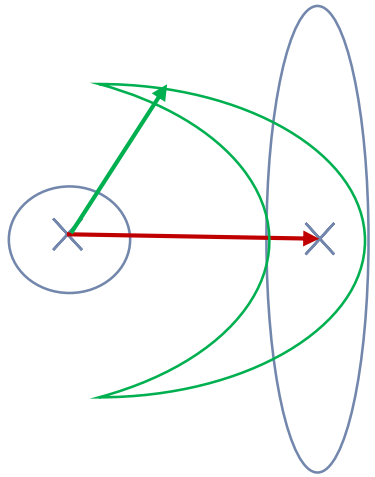
$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \cos(\alpha) & -v\sin(\alpha) \\ 0 & 1 & \sin(\alpha) & v\cos(\alpha) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ .$$

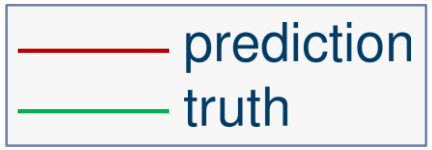
For the same scenario as before, we would have a low variance in v, but a high in α. Linearizing at the current estimate with $\alpha = 0$, we get

$$\mathbf{F} = \alpha = 0 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ .$$

Example – EKF issues

Linearized at this point, the x_1 uncertainty does not grow much due to the certain v. x_2 will get a larger uncertainty, but the resulting covariance does not cover the true position well. The reason is how the linearization handles the angular uncertainty of the velocity vector. The true uncertainty (green banana shape) would cover the true position well.





Example – Multiplicative noise with EKF

Assume the zero-mean Gaussian noise ν in the motion model is not additive, but multiplicative, resulting in a transition function

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \nu) \ .$$

The EKF is able to deal with this setting as well, applying the same linearization technique. Given the random variables are distributed with $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \mathbf{C}_k)$ and $\nu \sim \mathcal{N}(0, \mathbf{Q})$, we get a covariance update

$$\mathbf{C}_{k+1} = \mathbf{F}\mathbf{C}_k\mathbf{F}^{\mathrm{T}} + \mathbf{L}\mathbf{Q}\mathbf{L}^{\mathrm{T}} \ ,$$

with

$$\mathbf{F} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k}, \ \mathbf{L} = \left.\frac{\partial f}{\partial \nu}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k} \ .$$

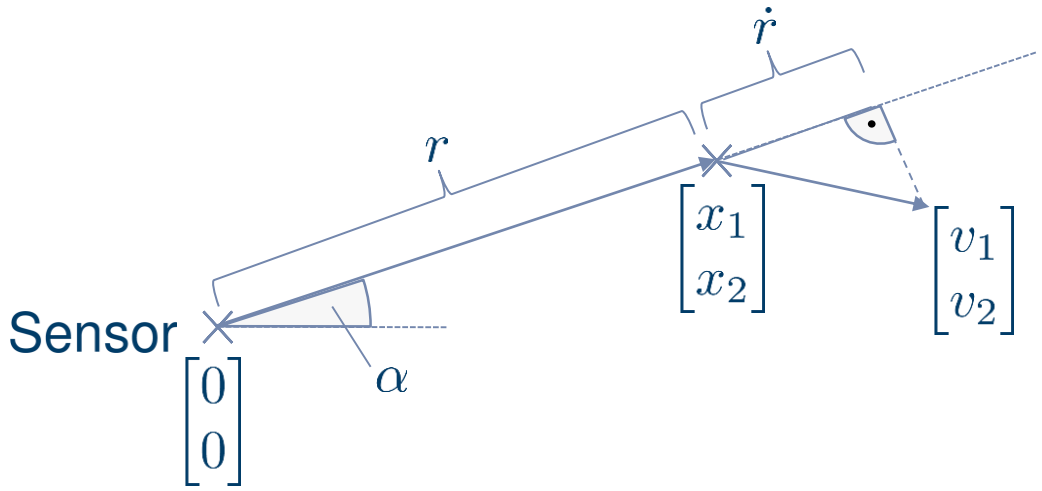
The same can be applied for multiplicative noise in the measurement model.

Problem 10 – Radar measurements with EKF

Consider a point target $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & v_1 & v_2 \end{bmatrix}$ consisting of 2D position and velocity. The target is measured via radar (located at $\begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$), giving us measurements

$$y = \begin{bmatrix} r \\ \alpha \\ \dot{r} \end{bmatrix} \quad ,$$

consisting of range r, angle α (relative to the x_1-axis), and range rate $\dot{r}$ with the latter being a projection of the velocity vector onto the measurement direction $\dot{r} = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \end{bmatrix} \cdot \begin{bmatrix} v_1 & v_2 \end{bmatrix}^{\mathrm{T}}$. Formulate the measurement equation and calculate the Jacobian necessary for the EKF measurement update.



Hint:

$$\frac{\partial}{\partial x}\mathrm{atan2}(y,x) = \frac{-y}{x^2+y^2}$$
$$\frac{\partial}{\partial y}\mathrm{atan2}(y,x) = \frac{x}{x^2+y^2}$$

Problem 10: solution

$$h(\mathbf{x}) = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ \text{atan2}(x_2, x_1) \\ (v_1 x_1 + v_2 x_2)/\sqrt{x_1^2 + x_2^2} \end{bmatrix}$$

$$H = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} & 0 & 0 \\ \frac{-x_2}{x_1^2+x_2^2} & \frac{x_1}{x_1^2+x_2^2} & 0 & 0 \\ \frac{-x_1(v_1x_1+v_2x_2)}{\sqrt{x_1^2+x_2^2}^3} + \frac{v_1}{\sqrt{x_1^2+x_2^2}} & \frac{-x_2(v_1x_1+v_2x_2)}{\sqrt{x_1^2+x_2^2}^3} + \frac{v_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \end{bmatrix}$$

Homework 10

Consider the same motion model as in the last homework:

Assume a robot in 2D-space at position $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ moving with velocity v_1 in x_1 direction and v_2 in x_2 direction. Its state is defined as $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & v_1 & v_2 \end{bmatrix}^T$.

a) This time, instead of measuring the position directly, we measure the distance to a landmark P_i. Formulate the measurement equation and write a function implementing it.

   For now, we will assume only a single landmark is used. In that case, the measurement noise will be a 1×1 matrix.

b) To use the measurement in the Kalman filter, we utilize the EKF formulas. Calculate the Jacobian of the measurement function from a) and implement the EKF measurement update.

   Again, we assume that for now only one landmark exists, and $\mathbf{R}$ will be a 1x1 matrix.

c) Your implementation will be used to track the robot, getting measurements from a single landmark $P_1 = \begin{bmatrix} 5 & 0 \end{bmatrix}^T$.

The measured distance is noise corrupted with Gaussian noise having zero mean and variance 0.1.

For the initial state and the process covariance, we will use

$$\mathbf{C}_{\text{init}} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0.02 \end{bmatrix}$$

For this, you will first need to implement the motion model of the robot.

Homework 10

d) To improve your estimate, use a second landmark $P_2 = \begin{bmatrix} 0 & 5 \end{bmatrix}^T$. Formulate a stacked measurement equation and calculate the corresponding Jacobian.

You will need to implement new versions of your `measurement_model` and your `update` functions.

Hints:

- since we are now dealing with a multi-dimensional scenario, you should consider using `np.random.multivariate_normal` instead of `np.random.normal`.
- make sure you calculate the distances between x and P_i across the correct axis, since P is now multi-dimensional too.