Probabilistic Robotics
KF - Laboratory exercise

This document will guide you through the 3rd practical work related with the Localization subject.

1. The goal

The goal of this practical exercise is to implement a KF Localization algorithm to localize a one-dimensional Mobile robot being moved in a hallway. During the exercise different motion models as well as different sensor measurements are used. You will program all the code in Matlab.

2. Work to do

During this lab session you must complete the m-file containing the KF Localization algorithm. These are the steps you must follow:

Previous to the lab sessions:

Read the complete lab document and the m-file and try to understand how it works. Determine the values for the matrices in the prediction model (A, B and Q) and the measurement model (H and R) for the 1st session example.

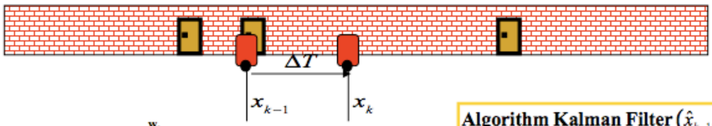
1st Lab session:

Odometry Motion Model with Odometry and Position Fixes – The simulated robot motion must move the robot in an almost constant velocity ($\dot{x}_k = \dot{x}_{k-1} + \Delta T w_k$). Use an odometry motion model to predict the robot position and absolute robot position fixes for the update. Complete the provided m-file.

- **Simulation data:**
  - **Simulation acceleration model noise:** σ_{sa}=1 cm/s^2
  - **Simulated Sonar noise (sonar position)**: σ_{ss}=20 cm
  - **Simulated Odometry noise:** σ_{so}=2 cm
  - **Sonar model noise:** σ_s= 2σ_{ss}
  - **Odometry model noise:** σ_o=2σ_{so}
- **Graphics to be plotted:**
  - Estimation Error + 2σ bound (position & velocity)
  - Histogram of the estimation error
- **Experiment with:**
  - In the current implementation, new odometry and position measurements are obtained at every time step. What would happen if position fixes were available at a lower frequency? (i.e. a new measurement arrives each k steps)
  - Test different periods of arrival for the absolute fixes.
  - What happens if no absolute fixes are used at all?

2nd Lab session:

Constant Velocity Motion Model with Position Fixes – The simulated robot motion must move the robot in an almost constant velocity ($\dot{x}_k = \dot{x}_{k-1} + \Delta T w_k$). Use a constant velocity model with acceleration noise to predict the robot position and absolute robot position fixes for the update. Complete the provided m-file.



$$\overbrace{\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}}^{\mathbf{x}_k} = \overbrace{\begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}}^{\mathbf{A}_k} \overbrace{\begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix}}^{\mathbf{x}_{k-1}} + \overbrace{\begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix}}^{\mathbf{w}_k} w_k \qquad \mathbf{B}_k = \mathbf{u}_k = \mathbf{0}$$

Algorithm Kalman Filter $(\hat{x}_{k-1}, P_{k-1}, u_k, z_k)$

$\hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k$

$P_k^- = A_k P_{k-1} A_k^T + Q_k$

$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$

$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H_k \hat{x}_k^-\right)$

$P_k = \left(I - K_k H_k\right) P_k^-$

return $\left(\hat{x}_k, P_k\right)$

$$z_k = \overbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}^{\mathbf{H}_k} \overbrace{\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}}^{\mathbf{x}_k} + \overbrace{v_k}^{\mathbf{v}_k}$$

$$\mathrm{Q}_k = \mathrm{Cov}\left(\mathbf{w}_k\right) = E\left[\mathbf{w}_k \mathbf{w}_k^T\right] = \begin{bmatrix} \frac{\Delta T^2}{2} \\ \Delta T \end{bmatrix} w_k w_k^T \begin{bmatrix} \frac{\Delta T^2}{2} & \Delta T \end{bmatrix} = \begin{bmatrix} \frac{\Delta T^4}{4} & \frac{\Delta T^3}{2} \\ \frac{\Delta T^3}{2} & \Delta T^2 \end{bmatrix} \sigma_w^2$$

$$R_k = Cov\left(\mathbf{v}_k\right) = E\left[v_k v_k^T\right] = \sigma_v^2$$

- **Simulation data:**
  - **Simulation acceleration model noise:** σ_{sa}=1 cm/s^2
  - **Simulated Sonar noise (sonar position)**: σ_{ss}=20 cm
  - **Simulated Odometry noise:** σ_{so}=2 cm
  - **Sonar model noise:** σ_s= 2σ_{ss}
  - **Odometry model noise:** σ_o=2σ_{so}
- **Graphics to be plotted:**
  - Estimation Error + 2σ bound (position & velocity)
  - Histogram of the estimation error
- **Experiment with:**
  - What happens if $\sigma_s > \sigma_{ss}$? What is the meaning? How does the filter perform?
  - And in the case of $\sigma_s < \sigma_{ss}$? What is the meaning? How does the filter perform?

After the lab sessions:

Write a report explaining your solution. Include the m-files of the two implementations.

Note: Two sessions are allocated for this lab.