Extended Kalman Filter localization

David FILLIAT - Goran FREHSE
ENSTA Paris

December 16, 2020

1 Introduction

In this practical work, we will work on robot localization using an Extended Kalman Filter (EKF). For this, we will use the python code available on the course Moodle. The provided code makes it possible to simulate a robot moving on a given trajectory in an environment made up of point landmarks. It implements a simple extended Kalman filtering method using the perception of the direction and distance of these point landmarks. The provided code requires the installation of the `numpy`[1] and `matplotlib`[2] python packages. **The provided code is incomplete and will not work directly.**

Upload your report as a pdf file that includes your answers to the questions and the code you wrote on the Moodle.

2 Models

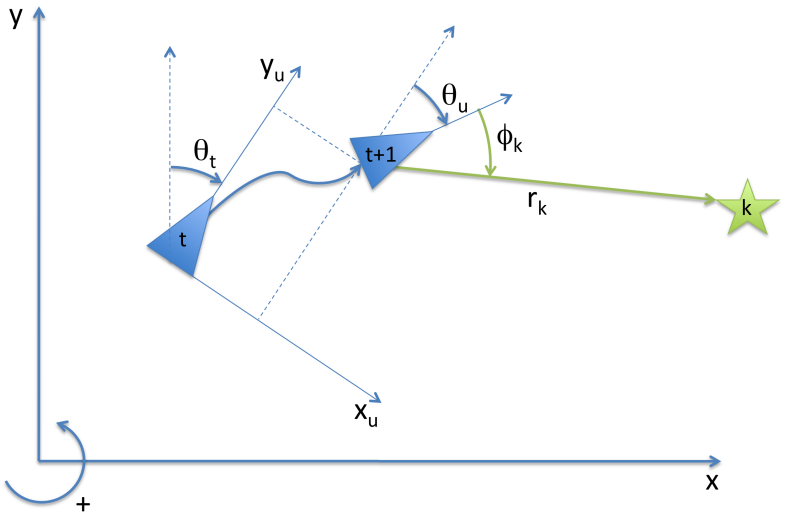

Figure 1: Notations for the motion and observation models used in the code.

The modeled robot (Fig. 1) moves on a plane and perceives the direction and the distance of point landmarks located on this same plane. Its state is represented by a column vector containing its position and its orientation in a global coordinate system:

$$X_t = [x_t, y_t, \theta_t]^T$$

The motion of the robot between t et $t+1$ is measured through odometry and given by its position at time $t+1$ in the robot frame at time t (Fig. 1):

$$U_t = [x_u, y_u, \theta_u]^T$$

[1] https://numpy.org/
[2] https://matplotlib.org/

The evolution model is then :

$$X_{t+1} = f(X_t, U_t) = \begin{bmatrix} x_t + x_u cos(\theta_t) - y_u sin(\theta_t) \\ y_t + x_u sin(\theta_t) + y_u cos(\theta_t) \\ \theta_t + \theta_u \end{bmatrix}$$

with a gaussian noise given by the covariance matrix Q.

The perceptions are the distance and direction of a landmark k supposed to be perfectly identifiable (Fig. 1):

$$Y_t = \left[r_t^k, \phi_t^k \right]^T$$

The observation model is therefore:

$$Y_t^* = h^k(X_t^*) = \begin{bmatrix} \sqrt{(x_k - x_t)^2 + (y_k - y_t)^2} \\ atan2(\frac{y_k - y_t}{x_k - x_t}) - \theta_t \end{bmatrix}$$

where x_k et y_k are the known coordinates of the landmark in the global coordinate system. This model is corrupted by a Gaussian noise of covariance matrix P_Y.

3 Questions

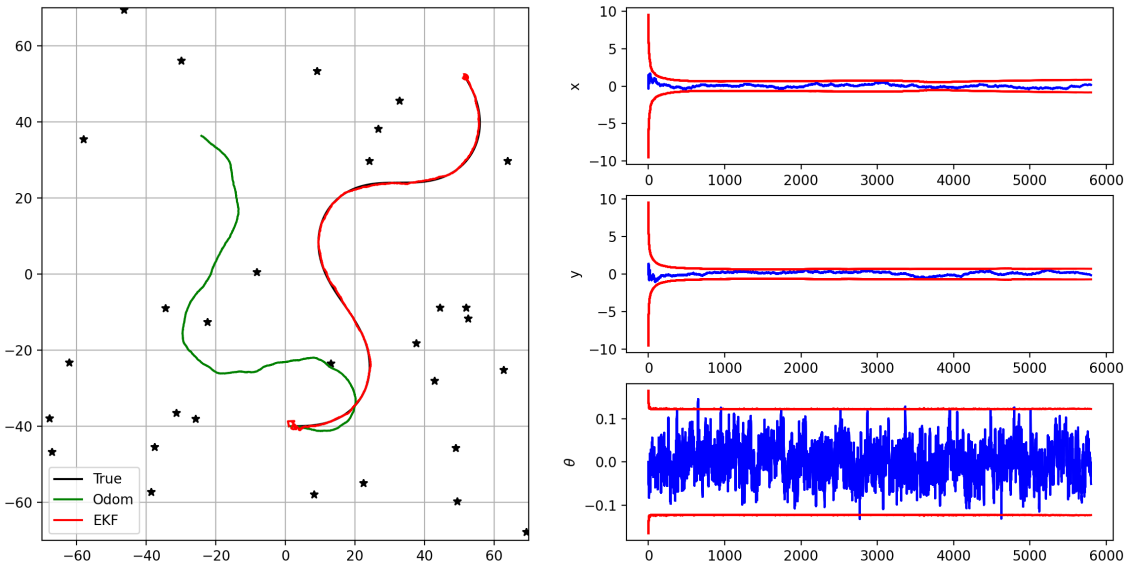

Figure 2: Example of script result

Question 1 : Jacobians Calculate the Jacobians A, B and H and write the corresponding functions in the python code (functions `get_obs_jac(xPred, iFeature, Map)`, `A(x, u)` and `B(x, u)` starting line 136). Then run the `EKFLocalization.py` script and observe the results, which should be similar to Fig. 2 if your Jacobians are correct. What do the three curves displayed on the right represent?

Question 2 : Sensor failure Implement a sensor failure with the function `get_observation(k)` returning the value `z = None` for a given duration (for example `k>2500 and k<3500`). What happens for the kalman filter during this time ?

Question 3 : Model errors The covariance matrices used in the filter (QEst, PYEst) faithfully reflect the noise of the robot's sensors (QTrue and PYTrue which are used for the simulator). Investigate the behavior of the filter when the noise used by the filter is grossly over or underestimated. What happens in particular

when the noise used on the perceptions is very low and the noise on odometry very high? What happens when the estimated noise on the distance of the landmarks is large and the noise on the direction of the landmarks is correct?

Question 4 : Partial observability One of the interesting aspects of the Kalman filter is the possibility of having only partial observations. This is already the case with the implemented version of the filter (the state is of dimension 3 while the observations are of dimension 2). It is however possible to use even less information, for example to use only the distance or the direction of the landmarks. Modify the model and the code (calculation of Innov, S, W and xEst around line 225) to use only the distance or the direction of the landmarks (advice: you must modify the dimensions of H, Py and Innov). What can we observe about the performance of the filter when you use the distance ? and when you use the direction ?