Simultaneous Localization and Mapping using Extended Kalman Filter

David FILLIAT - Goran FREHSE
ENSTA Paris

January 2, 2023

1 Introduction

In this practical work, we will study a Simultaneous Localization and Mapping (SLAM) method that builds a map of an unknown environment using an Extended Kalman Filter (EKF). For this, we will use the python code available on the course Moodle. The provided code is modified from the Python Robotics library[1] and makes it possible to simulate a robot moving on a given trajectory in an environment made up of punctual landmarks. It also implements a simple extended Kalman filtering method using the perception of the direction and distance of these landmarks. It requires the installation of the `numpy`[2] and `matplotlib`[3] python packages.

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

2 Code overview

The state vector of the Kalman filter (variable `xEst` in the code) contains the position of the robot and the position of all the currently known landmarks:

$$xEst = \begin{bmatrix} x \\ y \\ \theta \\ x_{a1} \\ y_{a1} \\ \vdots \\ x_{aN} \\ y_{aN} \end{bmatrix}$$

The associated covariance matrix is in the variable `PEst`.

The motion model is using commands on the translational and rotational speed ($u = (v, \omega)$):

$$f(xEst, u) = \begin{bmatrix} x + v.dt.cos(\theta) \\ y + v.dt.sin(\theta) \\ \theta + \omega.dt \end{bmatrix}$$

The observation model gives the direction and distance of the landmarks from the robot position:

$$h(xEst) = \begin{bmatrix} \sqrt{(x - x_{ai})^2 + (y - y_{ai})^2} \\ atan2(\frac{y_{ai} - y}{x_{ai} - x}) - \theta \end{bmatrix}$$

Most of the parameters that you need to change are at the beginning of the file :

- `Q_Sim` and `Py_Sim` are the noises used by the robot simulator, corresponding to the noise you would find on a real robot

[1] https://github.com/AtsushiSakai/PythonRobotics
[2] https://numpy.org/
[3] https://matplotlib.org/

- `Q` and `Py` are the noises used by the Extended Kalman Filter. In real scenarios they are based on an estimation of the real system noise, but here they can be set according to the simulation noise.
- `MAX_RANGE` is the maximum sensing distance of the sensor. Landmarks farther than this distance are ignored.
- `KNOWN_DATA_ASSCIATION` switch between known data association (using landmark id) or association to nearest neighbor computed using mahalanobis distance.
- `M_DIST_TH` is the threshold on the Mahalanobis distance between a real observation and an estimated observation from the map to recognize a landmark if the `KNOWN_DATA_ASSOCIATION` parameter is 0.

The environment (i.e. list of landmarks) is defined in the beginning the `main()` function, in the variable `Landmarks`, and the trajectory is defined by setting fixed controls in the `calc_input()` function.

3 Influence of the environment

For this question, use the default parameters of the provided code. By default, the data association is assumed to be known, ie for each perceived landmark, the corresponding landmark in the map is identified without ambiguities. In particular, this makes it possible to properly manage loop closures, even when the error in the map is very severe.

Question 1 : Modify the number and position of landmarks and the robot trajectory and explain what you observe (on the map quality and the evolution of errors, in particular around the time when the loops are closed) in the following situations :

- a short loop and a dense map with many landmarks inside the robot perception radius
- a long loop and a dense map with many landmarks all along the loop
- a long loop and a sparse map with only few landmarks near the start position

Question 2 : Answer the same question when the data association is performed using the Mahalanobis distance (`KNOWN_DATA_ASSOCIATION` = 0). You may have to tune the `M_DIST_TH` parameter depending on your environment.

4 Probabilistic models

For the this question, keep the configuration with unknown data association (`KNOWN_DATA_ASSOCIATION` = 0) and an environment with a large loop and a sparse map.

Question 3 : Change the estimated noise values `Q` and `Py` so that they are (1) smaller, (2) equal or (3) larger than the values used for simulation (`Q_Sim` and `Py_Sim`). What happens in each case for the filter performance, the filter consistency and the map quality? What seems to be the best configuration ?