Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2015
Prof. Dr. Wolfram Burgard
Dr. Gian Diego Tipaldi
Dr. Barbara Frank

Sheet 7

Topic: Extended Kalman Filter

Submission deadline: June 25, 2014

Submit to: `mobilerobotics@informatik.uni-freiburg.de`

General Notice

In this exercise, you will implement an extended Kalman filter (EKF). A framework containing stubs of all the parts of the EKF is provided to you, so you can concentrate on the implementation of the filter itself. The framework for the implementation can be obtained from the website of the course. The tarball contains the following folders:

data This folder contains files representing the world definition and sensor readings used by the filter.

python This folder contains the EKF framework with stubs for you to complete.

To run the EKF, just run in the terminal `python ekf_framework.py sensor.dat world.dat` .

Note: You first have to complete all the functions in order to get the filter working correctly. We are not using any external library for visualizing state of the robot and the landmarks. The stub for plotting the state of the robot is included in the framework. Once you have completed the EKF, you should be able to see a simple matplotlib figure popped up for visualization.

Some implementation tips:

- Turn off the visualization to speed up the computation by commenting the lines in the main for loop
- To read in the sensor and world data, we have used dictionaries. Dictionaries provide an easier way to access data structs based on single or multiple keys. The functions `read_sensor_data` and `read_world_data` read in the data from the files and build a dictionary for each of them with timestamps as the primary keys. To access the sensor data from the data_dict, you can use

  ```
  data_dict[timestamp,'sensor']['id']
  data_dict[timestamp,'sensor']['range']
  data_dict[timestamp,'sensor']['bearing']
  ```

  and for odometry you can access the dictionary as

  ```
  data_dict[timestamp,'odom']['r1']
  ```

`data_dict[timestamp,'odom']['t']`
`data_dict[timestamp,'odom']['r2']`

To access the positions of the landmarks from world_dict, you can simply use

`world_dict[id]`

Exercise 1: Theoretical Considerations

The EKF is an implementation of the Bayes Filter.

(a) The Bayes filter processes three probability density functions, i. e., $p(x_t \mid u_t, x_{t-1})$, $p(z_t \mid x_t)$, and $\text{bel}(x_t)$. State the normal distributions of the EKF which correspond to these probabilities.

(b) Explain in a few sentences all of the components of the EKF, i. e., μ_t, Σ_t, g, G_t, h, H_t, Q_t, R_t, K_t and why they are needed. What are the differences and similarities between the KF and the EKF?

Exercise 2: EKF Prediction Step

We assume a differential drive robot operating on a 2-dimensional plane, i.e., its state is defined by $\langle x, y, \theta \rangle$. Its motion model is defined on slide 10 (Odometry Model) in the chapter Probabilistic Motion Models of the lecture slides.

(a) Derive the Jacobian matrix G_t of the noise-free motion function g. Do not use Python.

(b) Implement the prediction step of the EKF in the function `prediction_step` using your Jacobian G_t. For the noise in the motion model assume

$$Q_t = \begin{pmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.02 \end{pmatrix}.$$

Exercise 3: EKF Correction Step

(a) Derive the Jacobian matrix H_t of the noise-free measurement function h of a range-only sensor. Do not use Python.

(b) Implement the correction step of the EKF in the function `correction_step` using your Jacobian H_t. For the noise in the sensor model assume that R_t is the diagonal square matrix

$$R_t = \begin{pmatrix} 0.5 & 0 & 0 & \dots \\ 0 & 0.5 & 0 & \dots \\ 0 & 0 & 0.5 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{\text{size}(z_t) \times \text{size}(z_t)}.$$

Once you have successfully implemented all the functions, after running the filter script you should see the state of the robot being plotted incrementall with each time stamp.