Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Introduction to Mobile Robotics
Summer term 2023
mobilerobotics@informatik.uni-freiburg.de

Dr. Tim Welschehold
Dr. Daniel Büscher
Dr. Lukas Luft
Adriana Gómez
Chenguang Huang

Sheet 9: Fast SLAM

July 13th, 2023

Exercise: FastSLAM Implementation

FastSLAM is a Rao-Blackwellized particle filter for simultaneous localization and mapping. The pose of the robot in the environment is represented by a particle filter. Furthermore, each particle carries a map of the environment, which it uses for localization. In the case of landmark-based FastSLAM, the map is represented by a Kalman Filter, estimating the mean position and covariance of landmarks.

Implement the landmark-based FastSLAM algorithm as presented in the lecture. Assume known feature correspondences.

To support this task, we provide a detailed listing of the algorithm as a PDF file and a small Python framework on the course website. The framework contains the following folders:

data contains the world definition and sensor readings used by the filter.

code contains the FastSLAM framework with stubs for you to complete.

You can run the fastSLAM framework in the terminal: `python fastslam.py`. It will only work properly once you filled the blanks in the code.

(a) Complete the code blank in the `sample_motion_model` function by implementing the odometry motion model and sampling from it. The function updates the poses of the particles based on the old poses, the odometry measurements δ_{rot1}, δ_{trans} and δ_{rot2} and the motion noise. The motion noise parameters are:

$$[\alpha_1, \alpha_2, \alpha_3, \alpha_4] = [0.1, 0.1, 0.05, 0.05] \tag{1}$$

How is sampling from the motion model different from the standard particle filter for localization (Exercise sheet 7)?

(b) Complete the code blanks in the `eval_sensor_model` function. The function implements the measurement update of the Rao-Blackwellized particle filter, using range and bearing measurements. It takes the particles and landmark observations and updates the map of each particle and calculates its weight w. The noise of the sensor readings is given by a diagonal matrix

$$Q_t = \begin{bmatrix} 1.0 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{2}$$

How is the evaluation of the sensor model different from the standard particle filter for localization (Exercise sheet 7)?

(c) Complete the function `resample_particles` by implementing stochastic universal sampling. The function takes as an input a set of particles which carry their weights, and returns a sampled set of particles.

How does the resampling procedure differ from resampling in the standard particle filter for localization (Exercise sheet 7)?

Some implementation tips:

- To read in the sensor and landmark 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` in the `read_data.py` file read in the data from the files and build a dictionary for each of them with time stamps as the primary keys.

  To access the sensor data from the `sensor_readings` dictionary, you can use

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

  and for odometry you can access the dictionary as

  ```
  sensor_readings[timestamp,'odometry']['r1']
  sensor_readings[timestamp,'odometry']['t']
  sensor_readings[timestamp,'odometry']['r2']
  ```