thws Technische Hochschule Würzburg-Schweinfurt

Faculty of Computer Science and Business Information Systems
5172080: Fundamentals of Mobile Robotics

Prof. Dr. Pascal Meißner
pascal.meissner@thws.de
+49 (0)931 35118353
Franz-Horn-Str. 2 (room K.3.12)

Computer Practical / Laboratory 5

Topic: Feature-based FastSLAM

On Monday 12:10-13:10, Week 2 of the calendar year,
you will complete the following tasks:

T 1 – Simultaneous Localization and Mapping

The FastSLAM approach is a Rao-Blackwellized particle filter for SLAM (Simultaneous Localization And Mapping). The pose of the robot in the environment is represented by a particle filter. In addition, 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 Extended Kalman filters (EKFs) that estimate the mean positions and covariances of landmarks.

In this computer practical, you are to implement one step of the landmark-based FastSLAM algorithm as detailed in the supporting PDF file "fastslam_algorithm.pdf". Assume the feature correspondences are known. A code skeleton in Python with the work flow of the algorithm is provided for you. A visualization of the state of FastSLAM is also provided by the framework.

The following folders are included in the lab_5_framework.zip archive:

data This folder contains files representing the world definition (i.e. landmarks) and sensor readings used by FastSLAM.

code This folder contains the FastSLAM framework with a stub for you to complete.

You can run the FastSLAM framework in the terminal: `python feature_based_fast_slam.py`. It will not work properly until you completed the blanks in the code.

1.a

Familiarize yourself with the existing code skeleton. What is the value of the robot pose that the particle filter is initialized with? Why does it make sense to choose this value? Is sampling from the motion model in `sample_odometry_motion_model` or the resampling procedure in `resampling` significantly different from those in Monte Carlo localization (see "Computer Practical / Laboratory 4")?

1.b

Fill in the `sensor_update` function and comment on the meaning of each line. `sensor_update` 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 measurements is given by a diagonal matrix

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

When updating the map of a particle, each measurement can belong either to a known landmark or to a new one. If a measurement belongs to a new landmark, a 2×2 EKF must be initialized according to lines 7 to 9 on slide 1 of the supporting PDF file. When initializing the mean position, you should take inspiration from "Tutorial 4 - Q 2". Use functions from `numpy.linalg` to initialize the landmark covariance. A function `measurement_prediction_and_jacobian` is provided to you in the framework so that you don't need to implement a measurement model.
If, on the other hand, a measurement belongs to a known landmark, the mean position and covariance of the landmark must be updated according to the EKF Correction step (see lines 12 to 17 on slide 3 of the supporting PDF file). Use the provided function `measurement_prediction_and_jacobian`. To correctly compute the difference between measured and expected bearing, you should use the `angle_diff` function also available in the framework. After the Correction step, the likelihood of the measurement must be computed according to line 18 on slide 3 of the supporting PDF file. Use the function `scipy.stats.multivariate_normal.pdf` to this end. Use this likelihood to update the importance weight of the particle at last.
Please also answer this question: What is the value of the best particle's estimate of the robot pose and how many landmarks have been observered by the end of time stamp 0?

1.c

Use the `plot_fast_slam_state` function to evaluate your completed FastSLAM implementation on the data provided in the folder `data`. Comment on how the uncertainties in the best particle's map, i.e. its Kalman filters, evolve between time stamps 0 and 7. Put this in relation to the main effects of the EKF Correction step.

1.d

How does the best particle's map of the environment change between time stamps 7 and 8? What is the reason for the differences in the dimensions of the error ellipses in the Kalman filters at time stamp 8? What uncertainties account for the largest error ellipse at this time stamp?

1.e

Comment on how the particle filter's uncertainty in the robot pose evolves from time stamp 0, to time stamp 100, to time stamp 250. In comparison, how does the uncertainty evolve between time stamps 250 and 300, and what is the name of the event that this is due to?

Implementation Tips

We used dictionaries to read in the sensor and landmark data. Dictionaries provide an easier way to access data structs based on single or multiple keys. The `read_sensor_data` and `read_world_data` functions in the `read_files.py` file read the data from the files and create a dictionary for each of them with time stamps as the primary keys.

For accessing 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']
```

For accessing the landmark positions from the landmarks dictionary, you can use
```
position_x = landmarks[id][0]
position_y = landmarks[id][1]
```