UNIVERSITÄT BONN HRL

Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik
Abteilung VI
Humanoid Robots Lab

Prof. Dr. Maren Bennewitz
Postanschrift: 53012 Bonn
Sitz: Friedrich-Ebert-Allee 144

Cognitive Robotics

Assignment 8

Due Tuesday, December 19th, before class.

8.1) Implement the landmark-based FastSLAM algorithm as presented in the lecture. Assume known feature correspondences, i.e., each landmark is uniquely identifiable. When a landmark is observed for the first time, initialize the covariance matrix of the EKF with $\Sigma = H^{-1} Q_t (H^{-1})^T$, where the measurement noise is given as $Q_t = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$.

To support this task, we provide a small Python framework (see the course materials website). The framework contains the following:

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

fastslam.py This file contains the FastSLAM framework with stubs for you to complete.

See the course materials webpage for installation instructions. To run the FastSLAM script, double click it or run `python fastslam.py` in a terminal.

Once you have filled in the lines marked with **`TODO:`** in the code, you should see a simple matplotlib figure popping up showing the estimated trajectory, the particles, and the estimated and true landmark positions. The ellipses around the robot's pose and the estimated landmark positions represent the 95% confidence region.

Some implementation tips:

- The Jacobian H of the range-bearing measurement with respect to the landmark position is returned as the second return value of the `measurement_model()` method.
- The function `read_sensor_data()` reads the input data files and provides the data as an array of instances of class `TimeStep`, each containing an odometry command and one or more sensor readings. See the pseudo-code structure on the next page and the comments in the code for how to access the data fields.
- The function `read_world()` reads the ground-truth world file and provides the true landmark positions as a dict mapping the landmark ID to the [x, y] coordinates of the landmark.

- The main prediction-correction-resampling loop is in the method `Plotter.fast_slam()`.

20 points

Structure of the input data as returned by `read_sensor_data()`:

```
data = Array of instances of class TimeStep {
   odom = instance of class Odomtry {
      r1 = initial rotation in radians (Float),
      t = translation in meters (Float),
      r2 = final rotation in radians (Float)
    },
    sensor = instance of class SensorMeasurement {
      landmark_id = Unique ID of the landmark (Integer),
      z_range = measured distance to landmark in meters (Float),
      z_bearing = measured angle to landmark in radians (Float)
    }
}
```