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

Computer Practical / Laboratory 4

Topic: Localization - Particle Filter

On Monday 12:10-13:10, Week 48 of the calendar year,
you will complete the following tasks:

T 1 – Particle Filter

In this computer practical, you will implement in Python a complete particle filter. A code skeleton with the work flow of the particle filter is provided for you. A visualization of the state of the particle filter is also provided by the framework. The following folders are included in the lab_4_framework.zip archive:

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

code This folder contains the particle filter framework with stubs for you to complete.

You can run the particle filter in the terminal: `python monte_carlo_localization.py`. It will not work properly until you completed the blanks in the code.

1.a

Fill in the code blank in the `sample_odometry_motion_model` function by implementing the odometry motion model as presented in the lecture "'Probabilistic Motion Models - 2"', slide 13. Comment on the meaning of each line. The function samples new particle positions based on the old positions, the odometry measurements δ_{rot1}, δ_{trans} and δ_{rot2} and the motion noise. This is done by passing each particle of the old set through the motion model in turn. Use the `numpy.random.normal` function for the motion noise and the parameters: $[\alpha_1, \alpha_2, \alpha_3, \alpha_4] = [0.1, 0.1, 0.05, 0.05]$. The function returns the new set of parameters after motion update.
Please also answer the following question: What is the value of the pose of the first particle drawn from the motion model?

1.b

Fill in the `compute_importance_weights` function and comment on the meaning of each line. This function implements the measurement update step of a particle filter as presented in the lecture "'Probabilistic Sensor Models - 2"', slide 25, using a *range-only* sensor. It takes as input positions l_k and observations z_k of landmarks. Note that individual observations are independent given a particle position x_i (see "'Probabilistic Sensor Models - 1"', slide 39). The function returns a list of weights for the particle set. See slide 11 of the lecture "'Localization - Particle Filter - 2"' for the definition of weights w_i. Instead of calculating a probability, it is sufficient to calculate the likelihood $p(z \mid x, l)$. Use the `scipy.stats.norm.pdf` function to evaluate this probability density function. The standard deviation of the normal distribution is $\sigma_r = 0.2$.
Please also answer the following question: What is the value of the weight of the first particle drawn from the motion model?

1.c

Fill in the `resampling` function by implementing stochastic universal sampling as presented in the lecture "'Localization - Particle Filter - 3"', slide 8. Comment on the meaning of each line. The function takes as an input a set of particles and the corresponding weights, and returns a sampled set of particles. Use the `numpy.random.uniform` function to draw random number r.
Please also answer the following question: What is the value of the pose of the first resampled particle?

1.d

Use the `plot_filter_state` function to evaluate your completed particle filter implementation on the data provided in the folder `data`. What is the value of the robot pose estimate for the time stamps 0, 5, and 10?

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]
```