MAE 145: Intro Robot Planning and Estimation
Extra Credit Problem 2

Due on March 18, 12:00pm

Notice:

For this assigment, update your amended particle filter code needed to run the filter. All extra credit programming problems (all combined) in the course will count up to an additional 5% of the grade in the course.

Particle Filter implementation for a wheeled robot:

In the following you will implement a complete particle filter. A code skeleton with the particle filter work flow is provided for you. A visualization of the particle filter state is also provided by the framework. You can find the following folders in "Homework > Extra Credit > particle filter":

data This folder contains files representing the world definition 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 particle_filter.py`. It will only work properly once you filled in the blanks in the code. Do the following:

1. Complete the code blank in the sample motion model function by implementing the odometry motion model and sampling from it. The function samples new particle positions based on the old positions, the odometry measurements $\delta_{\text{rot1}}, \delta_{\text{trans}}$ and δ_{rot2} and the motion noise. The motion noise parameters are:
$$[\alpha_1, \alpha_2, \alpha_3, \alpha_4] = [0.1, 0.05, 0.05]$$
(Note: this was part of a previous homework assignment)

2. Complete the function `eval_sensor_model`. This function implements the measurement update step of a particle filter, using a range-only sensor. It takes as input landmarks positions and landmark observations. It returns a list of weights for the particle set. Instead of computing a probability, it is sufficient to compute the likelihood $p(z|x, l)$. The standard deviation of the Gaussian zero-mean measurement noise is $\sigma_r = 0.2$.

3. Complete the function `resample_particles` by implementing stochastic universal sampling. The function takes as an input a set of particles and the corresponding weights, and returns a sampled set of particles.

Some implementation tips: To read in the sensor and landmark data, we have used dictionaries. Dictionaries provide an easier way to access data structures 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]`

To access the positions of the landmarks from `landmarks` dictionary, you can use

```
position x = landmarks[id][0]
position y = landmarks[id][1]
```