Intelligent Robotic Systems

Exercise 2: Particle Filter

Lecturer: Armin Biess, Ph.D.

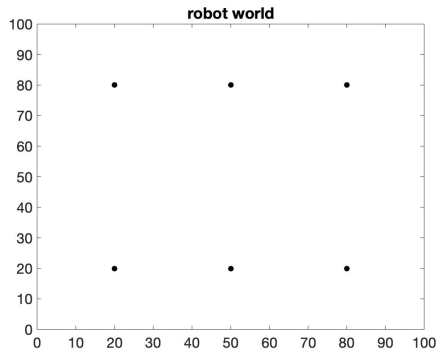
Due date: 2.1.2020

Monte Carlo Localization of a mobile robot using landmarks [100 pts]

Consider a planar robot with three DOFs $\boldsymbol{x} = (x, y, \theta)$ operating in a world of size 100×100. The world includes six landmarks at

$$\boldsymbol{m} = \{(20,20),(50,20),(80,20),(80,80),(50,80),(20,80)\}\,,$$

as shown in the figure



The robot can take two motor commands, a turn movement command $u_1, (u_1 \in [0, 2\pi))$ and a forward movement command u_2 $(u_2 > 0)$. The deterministic motion model of the robot is given by

$$\begin{aligned}
\theta' &= \theta + u_1\,,\\
x' &= x + u_2 \cos\theta'\,,\\
y' &= y + u_2 \sin\theta'\,,
\end{aligned}$$

For the implementation of the MCL algorithm two Matlab classes are provided. The `cWorld` class defines the world of the robot and includes a `plot` function. The `cRobot` class initializes a robot with an arbitrary pose in the world. This class includes the following functions:

- `set`: sets new pose of the robot
- `print`: prints the pose of the robot to the Matlab prompt
- `plot`: plots the robot in the world [different colors and different plot styles - `'robot'` and `'particle'` - can be chosen]
- `set_noise`: sets the noise parameters (forward_noise, turn_noise, sense_noise_range, sense_noise_bearing)

a.)[10 pts] Experiment with the `cRobot` class by plotting three robots into the world with the following poses (x, y, θ): $\{(45, 45, 0), (50, 60, \pi/2), (70, 30, 3\pi/4)\}$ and by printing these poses to the Matlab prompt. Initialize the world and robot by typing at the Matlab prompt (and similar for Python):

$$\begin{aligned} >> &\quad \text{myworld} = \text{cWorld}(); \\ >> &\quad \text{myworld.plot}; \\ >> &\quad \text{myrobot} = \text{cRobot}(); \end{aligned}$$

b.)[10 pts] Add a function `move` to your robot class, which takes as input the two motor commands and outputs the new pose. Assume that the real motor commands are noisy and corrupted with Gaussian noise of zero means and standard deviations $\sigma_1 = 0.1$ (rad) (turn_noise) and $\sigma_2 = 6$ (forward_noise), respectively.
c.)[10 pts] Add a function `sense` to your robot class that outputs the distances and bearing to the six landmarks. Assume that the measurements are corrupted by Gaussian noise with zero mean and standard deviation $\sigma_{range} = 5$ (sensor_noise_range) and $\sigma_{bearing} = 0.3$ (sensor_noise_bearing)
d.)[10 pts] Add a function `measurement_probability` to your robot class that takes the measurement from part c.) as input and outputs the probability for this measurement to be observed when being in pose $\boldsymbol{x}$.
e.) [5 pts] The robot performs now the following five actions:

1. `move` by $u_1 = 0$, $u_2 = 60$,
2. `move` by $u_1 = \pi/3$, $u_2 = 30$,
3. `move` by $u_1 = \pi/4$, $u_2 = 30$,
4. `move` by $u_1 = \pi/4$, $u_2 = 20$,
5. `move` by $u_1 = \pi/4$, $u_2 = 40$,

starting from pose $\boldsymbol{x} = (10, 15, 0)$ at time $t = 0$. Plot the robot poses and the robot path (as a dotted line) resulting from the action commands assuming that the robot follows the action commands exactly, i.e., there is *no* noise in the system.
f.)[5 pts] Plot the true robot poses and robot path (as a solid line) into the same figure. The true poses and the true robot path result from the noisy motor commands (noise: zero means and standard deviations σ_1 and σ_2) (Make sure to store the true pose because it is needed for the particle filter).
g.)[50 pts] Implement a particle filter with $N = 1000$ particles to estimate the poses and the path of the robot by taking measurements of the landmarks. Start by initializing the particles at the initial pose $\boldsymbol{x} = (10, 15, 0)$. After each motor command the robot takes a measurement from its true pose and updates its belief. Plot the particles before the measurement update (in black), i.e., before resampling, and after the measurement update (in gray), i.e., after resampling, using the `'particle'` plot-style of the `plot` function in the `cRobot` class. Determine the estimated poses and estimated path by averaging over all particles at each time step. Plot the estimated path (as a dash-dotted line) and compare it with the commanded and true path.

Remarks:

1. It is assumed that the world is cyclic, i.e., if the robot exits on one side it will re-enter on the opposing side.
2. It is assumed that the problem of correspondences between measurement and number of landmark is solved, i.e., the i-th measurement corresponds to the i-th landmark.
3. **IMPORTANT**: Seed your random number generator so that I can repeat your implementation.