CSCI 445 Lab 8 & 9 — Particle Filter

Prof. Culbertson, University of Southern California Fall 2021

In the next two labs you will implement a particle filter to allow your robot to localize itself in a known environment. As your first milestone, you will remote-control your robot and test the particle filter. In the second step you will let the robot autonomously explore the area until it has localized itself.

1 Prerequisites

- Review particle filters (i.e. Lecture 14).
- Complete pre-lab.
- Bring your laptop.

2 Getting Started

Open `Lab8.ttt` in CoppeliaSim and run `python3 run.py --sim lab8` as usual. Verify that you see in CoppeliaSim two particles (drawn as red triangles). If you press one of the buttons in the bottom right corner in CoppeliaSim (e.g. "Move Forward"), the robot should move and you should see a message on your command prompt.

3 Implement Particle Filter

Create a new file `particle_filter.py` with a new class `ParticleFilter`. As discussed in the lecture, you need to provide at least three functions:

1. Movement: This is called if a new movement command is issued (i.e. move forward, turn left, turn right).
2. Sensing: This is called if you get a new sensor reading.
3. Estimation: This gives you the current best estimate of the robot location.

Implement your particle filter, but keep the code as generic as possible. Any parameters (like the number of particles or variance estimates) should be passed in rather than hard-coded.

Hints:

- First, write a class for a single particle that contains the states you will estimate.
- To avoid numerical instabilities, you will need to work with the logarithm of probabilities rather than probabilities directly. In that case, multiplications become additions: $\log(ab) = \log a + \log b$.
- Assuming that you imported numpy using `import numpy as np`, here are some useful functions:
  - `np.random.uniform`: Generates random number(s) uniformly in the desired range.
  - `np.random.normal`: Generates random number(s) according to a Gaussian/normal distribution with given mean and variance.
  - `np.random.choice`: Chooses n objects from a given array of objects with specified probabilities.
  - `np.average`: Computes the (optionally weighted) average of the input array.

  You can find details about calling these functions in the NumPy reference: https://docs.scipy.org/doc/numpy-1.15.0/reference/index.html
- Assuming that you imported scipy using `import scipy`, here are some useful functions:
  - `scipy.stats.norm.pdf`: Probability density function of the Guassian/normal distribution.
  - `scipy.misc.logsumexp`: Computes $\log\left(\sum_{i=1}^{n} e^{a_i}\right)$ where a is an n-dimensional input array of probabilities.

You can find details about calling these functions in the SciPy reference: https://docs.scipy.org/doc/scipy/reference/

4 Particle Filter with Manual Motion Control

4.1 Simulation

Complete `lab8.py` so your particle filter is updated whenever your robot moves or makes a sensor reading and the particles displayed in CoppeliaSim are updated whenever your particles change. Whenever your robot moves, you need to call the movement function of your particle filter to move the particles as well. Whenever you get a sensor reading, you should execute the sensing step of the particle filter. Whenever your particles change, make sure that they get updated in CoppeliaSim.

Find reasonable parameters for the movement and sonar variances and test your implementation in simulation. How much do you need to move and sense until the robot knows where it is?

4.2 Robot

Make sure that your code works on the robot as well. For that, we will use CoppeliaSim as a visualization tool rather than as a simulation tool. (In the previous step we actually did both: CoppeliaSim was used for simulation and visualization at the same time. It is common in robotics to split those tasks in order to make it easier to switch between simulation and physical experiments.)

To connect the robot to CoppeliaSim, open `lab10.ttt` in CoppeliaSim. Press the "Toggle real-time mode" button in the CoppeliaSim toolbar to enable real-time mode. In this mode, your simulation time will exactly match the wall-clock time. Next, start the simulation manually by pressing the "Start/resume simulation" button on the CoppeliaSim toolbar.

In your code, instantiate `virtual_create` by providing the IP address of your laptop. It should start with 192.168 if you are connected to the CSCI445 network. Run your code on the robot. It should move when you press the appropriate buttons in CoppeliaSim, and the particles should be displayed in the simulator.

Hint: You can find your IP address from the command line using `ipconfig` in Windows or `ifconfig` in Mac OS or Linux.

5 Particle Filter with Automatic Motion Control

5.1 Simulation

So far you had to guide your robot to localize itself by pressing buttons in CoppeliaSim. Now, your robot should execute actions (such as movement or sensing) autonomously and stop once it is confident that it has found its location. Implement this behavior and test it in simulation.

Hints:

- You can give your robot a series of actions to execute in order to complete this task. It does not need to make complex decisions about where to go next. However, it should avoid taking any action that will cause it to run into a wall.
- Your robot should stop when it is confident about its location. How can you determine its confidence from the particle filter?

5.2 Robot

We will place your Create2 randomly inside a real maze-like structure like the one in `lab8.ttt`. Your robot will need to localize itself autonomously.

6 Timeline

You will have all of Labs 8 and 9 to finish the assignment and show all your results to the TAs. We recommend that you get all parts of the particle filter implemented during Lab 8 and begin connecting the particle filter with the simulation. If you do not complete the simulation during Lab 8, you should work on it during the week and bring a fully working simulation to Lab 9. This will ensure that you have enough time to deal with any robot-related issues and retuning your filter parameters during Lab 9.