50.047 – Mobile Robotics
DUE: Apr 6, 2025, at 11:59 pm

Overview: In this assignment you will implement a particle filter for Monte Carlo localization in simulation and object detection on the turtlebot.

1 Monte Carlo Localization (15 pts)

In this exercise you are going to implement Monte Carlo Localization (i.e. localization in a known occupancy grid map, using particle filters), as discussed in class. Your robot is going to start by being completely lost in the environment, so particles are going to be spread out uniformly at random in the known world. After many LiDAR measurements, the robot's pose is going to be constrained and the particles are going to converge to a small cluster. The main mechanism for survival of the fittest among particles will be: which particles are more likely to have generated the laser scans that the robot is observing?

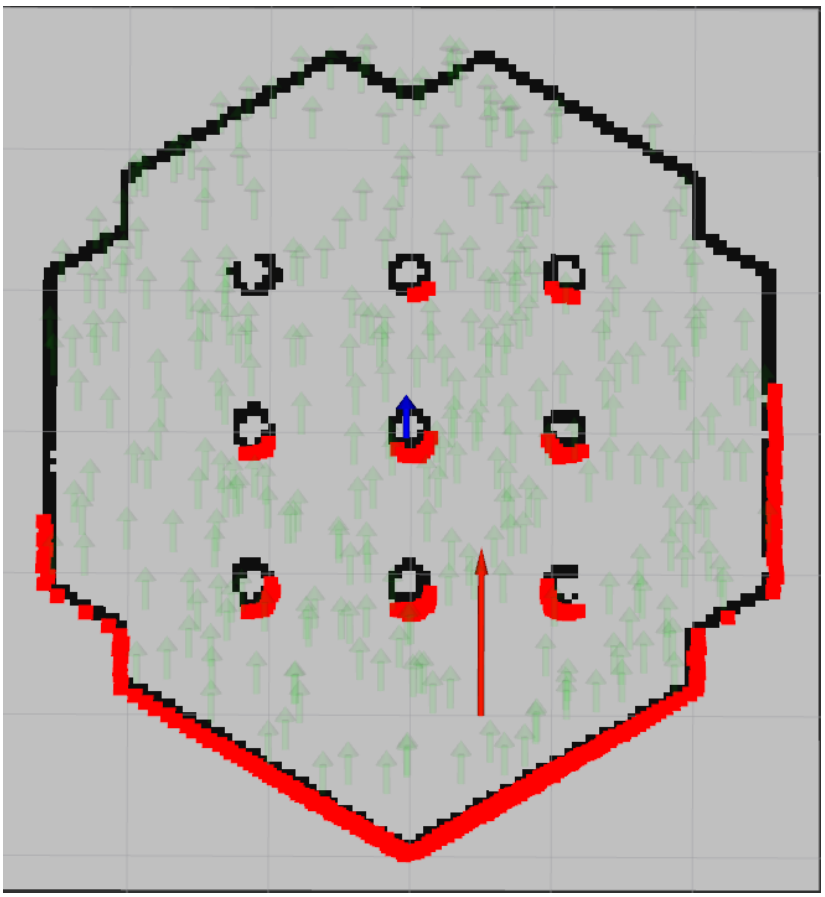

Figure 1: Uniform initialization of particles in Monte Carlo Localization, within the boundaries of a workspace. Your task is to make the particles cluster around the robot, as it makes LiDAR measurements of the environment. The environment consists of the black square and circular-shaped obstacles, shown above.

Starter code

On eDimension portal you can get the starter code. The functionality that you need to implement is marked using TODO comments in the file `~/mobile_robot_ws/src/HW4/estimation_and_vision_assignment/`

`python/monte_carlo_localization.py`.
Copy the files from edimension into the `~/mobile_robot_ws/src` folder previously created in HW1. Then run the following commands:

```
cd ~/mobile_robot_ws
# Build the new package
colcon build --symlink-install
# Source your install files
source install/local_setup.bash
```

The dependencies required should have already been installed from HW1, if working on a new computer, get the `package_install.bash` file from HW1 and run it to ensure you have the right dependencies.

Once installed, to run the starter code, execute the following commands on different terminals. What is shown on rviz2 should be similar to what is seen in Figure 1. The starter code for the `monte_carlo_localization.py` script will fail and only after filling in the TODOs, should the green arrows show up. The green arrows represent the particles and should be initialized with prior knowledge of the robot's heading (i.e. it should be facing forward):

```
# Startup Rviz2
rviz2 -d ~/mobile_robot_ws/src/HW4/estimation_and_vision_assignment/resources\
/viz.rviz

# In a different terminal, startup the map republisher
cd ~/mobile_robot_ws/src/HW4/estimation_and_vision_assignment\
/python
python3 republish_map.py

# In a different terminal, run the ros2 bag file
cd ~/mobile_robot_ws/src/HW4/estimation_and_vision_assignment\
/laser_odom_map_control
ros2 bag play laser_odom_map_control.db3

# In a different terminal, launch the monte carlo localization script
ros2 launch estimation_and_vision_assignment monte_carlo_localization.launch.py

# NOTE: Only if the map does not show on rviz2, then run the following command
# This will start a static tf publisher which lets rviz2 start
# reading the tf data from the ros2 bag file
ros2 launch estimation_and_vision_assignment ground_truth_pub.launch.py
```

After implementing the monte carlo localization algorithm, try running it with 2000 particles and without the initial heading prior (i.e. randomizing the initial orientation of the particles).

Specifically, in the `monte_carlo_localization.py` script, comment out line 103: `theta = np.random.uniform(0, 0.01)` and uncomment line 104: `theta = np.random.uniform(0, 2*pi)`. Then, change `num_particles = 300` to textttnum_particles = 2000 at the bottom of the script.
[Bonus 1 Point] Part B: Describe what was the output of your algorithm with this new setting and discuss at least 2 limitations of the particle filter algorithm. Write this in a word document or text file and submit it along with your code.

What you need to submit: In addition to your code, a video recording of the rviz visualization demonstrating your particles converging from beginning to end for the default settings is also required. Your video should be named `mcl_firstname_lastname.mp4/avi/ogg`. Submit your answer for Part B along with the code.

Example: An example video of how the localization algorithm should run would also be provided in edimension to give you an idea of how it should look it.

2 Object detection and open loop control

This is a group question which would require you to work together and use the 1/10th self-driving car. Using what was discussed in the object detection tutorial, program the car to move towards a bag and stop within 1m distance of it. You may choose to start the car from any position as long as it is at least 1.2m away from the bag.
The following is a suggested method for doing this.

1. Position the car to be directly in front of the bag at 1.2m distance away.
2. Similar to the object detection tutorial, write a ros2 node which subscribes to the ZED2 node and obtains information about the object class and distance to the object.
3. Once the distance is detected, give a forward velocity command until the distance is less than 1.0m to the bag.
4. Set the velocity to be 0.0 once it is within 1.0m range of the bag.

What you need to submit: In addition to your code, a video recording of the 1/10th car moving towards and stopping in front of the bag is required. Your video should be named `obj_det_<group_num>.mp4/avi/ogg`.
Example: An example video of how the car could move is given on edimension.

3 How to submit

Submit all your work in a file called `HW4_firstname_lastname_studentid.zip` that should contain 2 other folders. (1) Your extensions to the provided starter code for Monte Carlo Localization which should be named `estimation_and_vision_assignment` and provide a README file on how to run it, (2) Your object detection code implemented on the car, share all files necessary to run the code on the car and provide a README file on how to run it. Submissions will be done on e-dimension.