Particle Filter Localization

Objectives

Modified from CMSC 20600

Your goal in this project is to gain in-depth knowledge and experience with solving the problem of robot localization using the particle filter algorithm. This problem set is designed to give you the opportunity to learn about probabilistic approaches within robotics and to continue to grow your skills in robot programming. If you find yourself stuck on something, email jason27146913@gmail.com or join the TA office hour.

Learning Goals

- Continue gaining experience with ROS and robot programming
- Gain a hands-on learning experience with robot localization, specifically the particle filter algorithm
- Learn about probabilistic approaches to solving robotics problems

Writeup

Please modify the `README.md` file as your write-up for this project. Please add pictures, Youtube videos, and/or embedded animated gifs to showcase and describe your work. Your writeup should:

- **High-level description** (1 paragraph): At a high-level, describe how you solved the problem of robot localization. What are the main components of your approach?
- **For each of the main steps of the particle filter localization**, please provide the following:
    - **Code location** (1-3 sentences): Please describe where in your code you implemented this step of the particle filter localization.
    - **Functions/code description** (1-3 sentences per function/portion of code): Describe the structure of your

code. For the functions you wrote, describe what each of them does and how they contribute to this step of the particle filter localization.

The main steps are,

1. **Initialization of particle cloud,**
2. **Movement model,**
3. **Measurement model,**
4. **Resampling,**
5. **Incorporation of noise,**
6. **Updating estimated robot pose,**
7. **Optimization of parameters.**

- **Challenges** (1 paragraph): Describe the challenges you faced and how you overcame them.
- **Future work** (1 paragraph): If you had more time, how would you improve your particle filter localization?
- **Takeaways** (at least 2 bullet points with 2-3 sentences per bullet point): What are your key takeaways from this project that would help you/others in future robot programming assignments working in pairs? For each takeaway, provide a few sentences of elaboration.

Code

Our ROS package contains the following files:

```
lab1_particle_filter/launch/navigate_to_goal.launch

lab1_particle_filter/launch/visualize_particle.launch

lab1_particle_filter/map/house_map.pgm

lab1_particle_filter/map/house_map.yaml

lab1_particle_filter/rviz/particle_filter_project.rviz

lab1_particle_filter/scripts/particle_filter.py

lab1_particle_filter/CMakeLists.txt

lab1_particle_filter/package.xml
```

```
lab1_particle_filter/README.md
```

You will write your code within the `particle_filter.py` file we've provided. You may also create other python scripts that may provide helper functions for the main steps of the particle filter localization contained within `particle_filter.py`.

Gif/MP4

At the beginning or end of your writeup please include a gif/mp4 file of one of your most successful particle filter localization runs. In your gif/mp4, (at minimum) please show what's happening in your rviz window. We should see the particles in your particle filter localization (visualized in rviz) converging on the actual location of the robot.

Note that, unfortunately, there are no recording options directly on rviz. If you are running on Windows/MacOS with NoMachine, we recommend that you record your NoMachine's rviz window directly from your base OS since CSIL machines don't come with screen recorders.

If you are on MacOS, QuickTimePlayer would be a great out-of-the-box option for screen recording. On Windows, a few options work great such as OBS Studio, or the Windows 10 builtin Game bar. On Ubuntu, one option would be Simple Screen Recorder

rosbag

Record a run of your particle filter localization in a rosbag. Please record the following topics: `/map`, `/scan`, `/cmd_vel`, `/particle_cloud`, `/estimated_robot_pose`, and any other topics you generate and use in your particle filter localization project. **Please do not record all of the topics, since the camera topics make the rosbags very large.** For ease of use, here's how to record a rosbag:

```
$ rosbag record -O filename.bag topic-names
```

Please refer to the ROS Resources page for further details on how to record a rosbag.

The Particle Filter Localization

The goal of our particle filter localization (i.e., Monte Carlo localization) will be to help a robot answer the question of "where am I"? This problem **assumes that the robot has a map of its environment**, however, the robot either does not know or is unsure of its position and orientation within that environment.

The way that a robot determines its location using a particle filter localization is similar to how people used to find their way around unfamiliar places using physical maps (yes, before Google Maps, before cell phones... aka the dark ages). In order to determine your location on a map, you would look around for landmarks or street signs, and then try to find places on the map that matched what you were seeing in real life.

The particle filter localization makes many guesses (particles) for where it might think the robot could be, all over the map. Then, it compares what it's seeing (using its sensors) with what each guess (each particle) would see. Guesses that see similar things to the robot are more likely to be the true position of the robot. As the robot moves around in its environment, it should become clearer and clearer which guesses are the ones that are most likely to correspond to the actual robot's position.

In more detail, the particle filter localization first initializes a set of particles in random locations and orientations within the map and then iterates over the following steps until the particles have converged to (hopefully) the position of the robot:

1. Capture the movement of the robot from the robot's odometry
2. Update the position and orientation of each of the particles based on the robot's movement
3. Compare the laser scan of the robot with the hypothetical laser scan of each particle, assigning each particle a weight that corresponds to how similar the particle's hypothetical laser scan is to the robot's laser scan
4. Resample with replacement a new set of particles probabilistically according to the particle weights
5. Update your estimate of the robot's location

Running the Code

When testing and running your particle filter code, you'll have the following commands running in different terminals or terminal tabs.

First terminal: run `roscore`.

```
$ roscore
```

Second terminal: run your Gazebo simulator. For this project, we're using the Turtlebot3 house environment.

```
$ roslaunch turtlebot3_gazebo turtlebot3_house.launch
```

Third terminal: launch the launch file that we've constructed that 1) starts up the map server, 2) sets up some important coordinate transformations, and 3) runs rviz with some helpful configurations already in place for you (visualizing the map, particle cloud, robot location). If the map doesn't show up when you run this command, we recommend shutting down all of your terminals (including `roscore`) and starting them all up again in the order we present here.

```
$ roslaunch lab1_particle_filter visualize_particles.launch
```

Fourth terminal: run the Turtlebot3 provided code to teleoperate the robot.

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

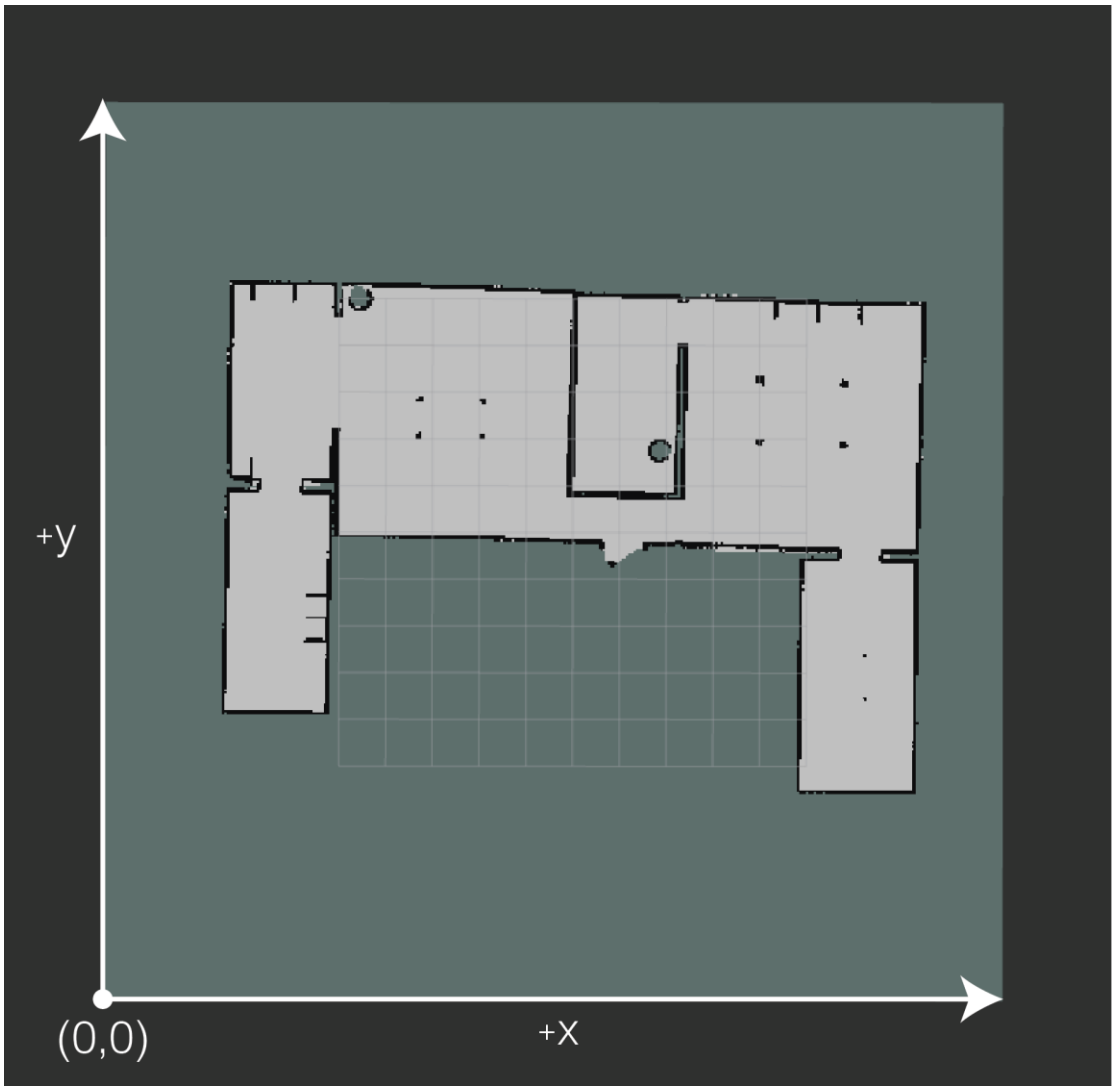
Fifth terminal: run your particle filter code.

```
$ rosrun lab1_particle_filter particle_filter.py
```

Note: It is very important that you execute `roslaunch lab1_particle_filter visualize_particles.launch` **AFTER** you execute `roslaunch turtlebot3_gazebo turtlebot3_house.launch` to ensure that the coordinate transforms work properly.

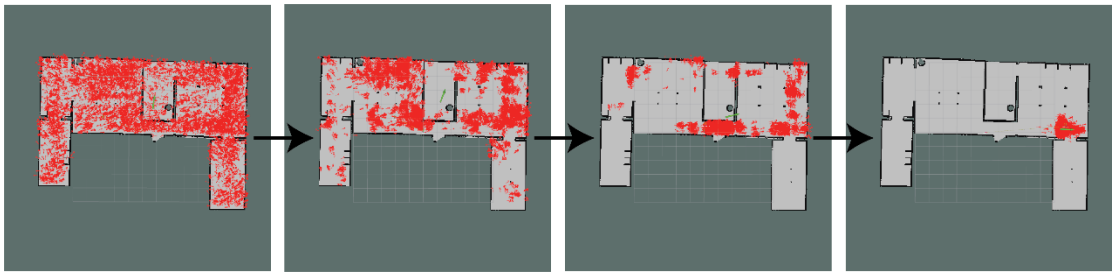
Working with the Map

A large component of step #3 of implementing the particle filter (see "The Particle Filter" section above), is determining the hypothetical laser scan of each particle so that it can be compared to the actual robot's laser scan. In order to calculate the properties of each particle's hypothetical laser scan, you'll need to work with the particle filter's `map` attribute, which is of type nav_msgs/OccupancyGrid. The map data list (`data`) uses row-major ordering and the map `info` contains useful information about the width, height, resolution, and more. The origin of the map is in the bottom left corner of the visualization of the map below.



A View of the Goal

The following progression of screenshots is from my implementation of the particle filter for this project. You can see that at the beginning, the particles are randomly distributed throughout the map. Over time, the particles converge on likely locations of the robot, based on its sensor measurements. And eventually, the particles converge on the true location of the robot.



Here are a few gifs that better portray the progression of the particle clouds,

Demo1
Demo2

Helpful Tips

- If your map isn't coming up in rviz, shut down all of your terminals, and re-run them in the order listed in "Running the Code".
- Start out using a small number of particles in your particle cloud, so you can make sure they're acting as you expect before scaling up the number of particles.
- Depending on the efficiency of your implementation, you may get exceptions regarding Gazebo's time going backward. In such cases, decrease the number of your particles and try running your localization again.
- It is possible that your Localization initially converges to multiple clouds. However, after a longer period of time, your particles should converge into one cloud.
- Both the sensors and motors generate noise within the Gazebo world. We suggest that you counteract Gazebo noise in your particle cloud update step by introducing artificial noise to the particles at each iteration. An easy way to generate artificial noise is to sample it from a normal distribution (Gaussian noise).
- Test your code early and often. It's much easier to incrementally debug your code than to write up the whole thing and then start debugging.

Grading

The Particle Filter Project will be graded as follows:

- 30% Writeup
  - 7% Objectives, High-level description, & gif
  - 15% Main steps
  - 8% Challenges, Future Work, Takeaways
- 10% ROS Bag Recordings
- 60% Code

Submission Deadline and Method

Language : python3

Packages : If you use any package, please mention it in README.md

Deadline : Thu 23 Mar 2023 13:00 (UTC+8)

Delay policy : One Day -> points * 0.7
Two Days -> points * 0.5
After Two Days -> 0 point

Everyone has a chance for a one-day late submission this semester!!!

Submission Method:
Upload a zip file to CGU Teams with the format :

```
m1234567_lab1.zip
    + m1234567_lab1
        - scripts
            + particle_filter.py
        - bags
            + particle_filter.bag
        + README.md
        + demo.mp4/gif
        + other files in the lab1_particle_filter folder
```

**Do not include other files other than those mentioned above*

All file names should be **only zip files (No rar and 7zip)*

*****Incompatible format will not be graded.***

Notice : Zero points for plagiarism!!!
(either from the internet or copy from classmate)