Lab 8: Localization

Learning Outcomes

- understand localization
- implement a particle filter

Section 1: Requirements and Design

In this lab, you will implement the a Particle Filter to localize Jet on a map. To setup the lab, draw a thick black line on a large posterboard or sheet of paper (the diameter should be about 1m). You will also need several boxes of approximately the same size - shoe boxes work very well.

Decide where the origin or start of the circle is (this can be an arbitrary location). Then setup two boxes, one placed directly across from the origin and another setup 90 degrees from the origin. These boxes correspond to the ones in the map parameter within `launch/lab8_localize.launch`. You are free to choose other layouts if you prefer, but you must update the map parameter in the launch file.

Section 2: Installing the Lab

To perform this lab, you will need to get the lab8_localization template into your catkin workspace. First ensure that your Jet has internet access by connecting it using WiFi or ethernet. Next ssh into Jet and enter the following command:

```
wget http://instructor-url/lab8_localization/lab8_localization-code.zip
```

Where the url should be replaced by the URL provided by your instructor. Now unzip the lab:

```
unzip lab8_localization-code.zip -d ~/catkin_ws/src/jetlabs/lab8_localization
```

Delete the zip file:

```
rm lab8_localization-code.zip
```

To build the code, use the following command when you are in `~/catkin_ws/`:

```
catkin_make --pkg lab8_localization && source devel/setup.sh
```

To setup the system, execute the following

```
roslaunch lab8_localization lab8_setup.launch
```

To run the localization, execute the following

```
roslaunch lab8_localization lab8_localize.launch
```

Section 3: Line Following

Before we begin localizing, Jet must follow the black line in a circle. Place Jet on the line pointing in a clockwise direction. Tilt the camera down to face the line. Start the setup launch file:

```
roslaunch lab8_localization lab8_setup.launch
```

On your laptop or desktop, open the segmented line image:

```
rosrun image_view image_view image:=/user/image1
```

Also open rqt_reconfigure:

```
rosrun rqt_reconfigure rqt_reconfigure
```

Adjust the values for the hue, saturation, and value until only the line appears in white on the image feed. You can then begin following the line by entering:

```
rostopic pub /go_follow std_msgs/Empty --once
```

To stop following the line invoke this command:

```
rostopic pub /stop_follow std_msgs/Emtpy --once
```

You can use rqt_reconfigure to change the Kp, Ki, and Kd values for the PID control as well, although you should be able to use the same values from lab 7. Once you are satisfied with the parameters, you can write them explicity in the `lab8_setup.launch` file with the format:

```
<param name="hue_lower" value="xxx"/>
```

Now you need to track how many encoder ticks are registered by the left wheel for each revolution of the circle. You can do this by subscribing to the `arduino/encoder_left_value`. Once you have this measurement, add update the `course_length` value in the `lab8_localize.launch`.

Section 4: Particle Filter

There are three aspects of the Particle Filter that you will have to implement: - moveUpdate - sensorUpdate - resample

The goal is to get the robot to stop at the origin of the circle (a predetermined position on the circle) after placing the robot at a random spot on the circle. As the robot moves around the circle it will update its particles until they converge on a single estimate. Once the robot has high certainty about its location it should stop when it reaches what it believes is the origin.

moveUpdate

In the `moveUpdate` method, all of the particles should be shifted by the amount that the robot moved since its last update. The particles are all one-dimensional values between 0 and MAP_SIZE (8). You must convert the change in encoder value into the units of the particles (0.0 - 8.0). If any particle ends up greater than the MAP_SIZE, it should wrap around.

Since the encoder values will not be perfect representations of the movement, you will need to add some noise to the updates. You can do this with

```
move_error_distribution(generator)
```

The standard deviation of the Gaussian Noise generated by the move_error_distribution is controlled by the `move_std_dev` parameter in `lab8_localize.launch`. Increasing the standard deviation will spread out the particles more, while decreasing the standard deviation will keep the particles closer to the motion estimate.

sensorUpdate

In the `sensorUpdate` method, you should give a high weight to the particles that match the sonar reading and a low weight to ones that do not. If the sonar detects an object, then all particles that are located at a point in the map that has an object should have high weights.

The parameter `hit_weight_ratio` in `lab8_localize.launch` defines the ratio between the particles that are match the sonar data and ones that do not. Even if a particle is not a hit, it still receives a non-zero weight.

To make the resampling step easier, you should normalize all of the weights so that the sum of the weights is equal to 1.0.

resample

The `resample` method is used to create a new vector of particles that contain old particles according to their weight. An old particle could appear zero, one, or many times in the new particles. Assign the vector of new particles to the particles variable. The new set of particles should have the same number of particles as the previous set.

Section 5: Analysis and Stopping

The `analyzeState` method has already been written for you. It publishes the mean particle location as well as the standard deviation of the particles. The method returns true when the particles have converged the mean particle value is close to the origin. You can change the `stop_threshold` parameter in `lab8_localize.launch` to adjust the certainty that the robot must have before stopping.

To troubleshoot your particle filter, you may consider publishing the particles and their weights at certain intervals. Since you will have many particles it may be beneficial to write the particles and weights to a file for analyzing. You can test the moveUpdate and senseUpdate methods independantly by running just `lab8_localize.launch` without the line following. Then publish to the `arduino/encoder_left_value` and `arduino/sonar_3` topics. You can modify the `loop` function to perform updates less frequently (like once every 30 seconds), so you can see how the particles change more clearly.

Section 5: Challenge

A famous problem in localization is the Stolen Robot Problem. In this scenario, the robot could be picked up and moved to a different location. Can you solve this problem. To do it, you will need to stop the line following with the `stop_follow` topic and then move the robot to a new spot. You can resume the line following by publishing to `go_follow`. Your code will need to handle particle deprivation because there might be no particles in the place where you deposit the robot. The easiest way to avoid particle deprivation is to randomly allocate new particles throughout the map space. The Stolen Robot Problem is a difficult task, but you will gain a deep understanding of particle filters if you are able to solve it.