Assignment 3

Localization and Mapping

Deadline: Thursday, December 13, 11:59pm
Perfect score: 100 points

Assignment Instructions:

Teams: Assignments should be completed by teams of students. Up to three members are allowed for CS460 students and up to two members for CS560 students. Since the assignment has different requirements for the undergraduate and graduate version of the course, we will not allow mixed teams in terms of CS460 and CS560. No additional credit will be given for students working in smaller than the allowed teams. You are strongly encouraged to form a team of the maximum size. Please inform the TAs (email: comprobfall2018/AT/gmail.com) as soon as possible about the members of your team so they can update the scoring spreadsheet. Failure to inform the TAs on time will result in your assignment not being graded.

Submission Rules: Submit your reports electronically as a PDF document through Sakai (`sakai.rutgers.edu`). For your reports, do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. You also need to submit a compressed file via Sakai, which contains your results and/or code as requested by the assignment. *Each team of students should submit only a single copy of their solutions and indicate all team members on their submission*. Failure to follow these rules will result in a lower grade in the assignment.

Program Demonstrations: You will need to demonstrate your program to the TAs and grader on a date after the deadline. The schedule of the demonstrations will be coordinated by the TAs. During the demonstration you have to use the files submitted on Sakai and execute it either on your laptop or an available machine, where the TAs and grader will be located (you probably need to coordinate this ahead of time). You will be asked to describe the architecture of your implementation and algorithmic aspects of the project. You need to make sure that you are able to complete the demonstration and answer the TAs' questions within the allotted 12 minutes of time for each team. If your program is not directly running on the computer you are using and you have to spend time to configure your computer, this counts against your allotted time.

Late Submissions: No late submissions are allowed. You will be awarded 0 points for late assignments.

Extra Credit for LaTeX: You will receive 6% extra credit points if you submit your answers as a typeset PDF (i.e., using LaTeX). Resources on how to use LaTeX are available on the course's website. There will be a 3% bonus for electronically prepared answers (e.g., on MS Word, etc.) that are not typeset. Have in mind that these bonuses are computed as percentage of your original grade, i.e., if you were to receive 50 points and you have typesetted your report using LaTeX then you get 3 points bonus. If you want to submit a handwritten report, scan it and submit a PDF via Sakai. We will not accept hardcopies. If you choose to submit scanned handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable.

Precision: Try to be precise in your description and thorough in your evaluation. Have in mind that you are trying to convince skeptical evaluators (i.e., computer scientists...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common code, notes or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Unless explicitly allowed by the assignment, do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or the university (the standards are available through the course's website). Failure to follow these rules may result in failure in the course.

Course's website:
`https://robotics.cs.rutgers.edu/pracsys/courses/intro-to-computational-robotics/`

Particle Filter for Robot Localization

1 Setup

For this part of the assignment you are asked to estimate the path followed by a Turtlebot that has moved in a maze-like environment given odometry and proximity sensor data. The Turtlebot is both rotating and translating while it is moving, so its pose exists in the $SE(2)$ space.

Parametrized noise is introduced in the odometry data to simulate realistic noisy wheel encoders. This means that the robot's pose cannot be accurately predicted just from the available odometry data. The robot, however, has also access to a map of the environment. It can use the map as well as the sensing data, which correspond to distances from the closest obstacle along different directions, in order to come up with an improved pose estimate.

The objective is to solve the problem in an incremental fashion following the principles of Bayesian filtering. This means that a new robot pose is computed given the previous pose estimate, as well as the latest odometry and observation measurement. You are not required, however, to solve the localization problem in real-time. Instead, you will have access to files representing the odometry and scans that correspond to an entire trajectory. You can then execute a particle filter for estimating the robot's pose corresponding to each pair of odometry measurement and scans.

1.1 Software Infrastructure and Available Input

The corresponding Gazebo-based software is available under the following URL: `https://github.com/rajanya/comprobfall2018-hw3`.

The purpose of this package is to allow the teleoperation of a Turtlebot robot in one of 7 different maze-like environment, similar to Assignment 1. During the robot's (noisy) motion, the software is generating noisy laser scan data. Furthermore, it also generates ground truth robot pose data, which can be used to evaluate the accuracy of the robot localization procedure (but not used by the algorithm).

We provide a ROS service `turtlebot_control`, which uses the following service file `turtlebot_control/TurtleBotControl`:

```
geometry_msgs/Point point
std_msgs/Bool return_ground_truth
--
gazebo_msgs/ModelState ground_truth
turtlebot_ctrl/TurtleBotScan scan_data
```

A request to the `turtlebot_control` service requires you to specify two values:

- Two `Float32` msgs that specify the new heading of the robot, and the distance that it must move along that heading. For example, if the robot is currrently at $(x, y, \theta) = (0, 0, 0)$, and the command you send it is $(\phi, d) = (1.57, 1)$, the robot will turn to $(0, 0, 1.57)$, and then drive for a distance of 1m to arrive at the state $(0, 1, 1.57)$. We will return a noisy reading of the control actually executed sampled from two Gaussians with standard deviations `rotation_noise` and `translation_noise` respectively. For example, we may return $(\phi', d') = (1.55, 1.02)$, which you can use as the control inputs for the various particles.
- A `Bool` msg that specifies whether Gazebo must provide you the ground truth location of the robot in the world. You may set this to `True` during data gathering and for debugging purposes, but your localization algorithm **must** set this value to `False`.

A response from the `turtlebot_control` service returns the following information after the robot's motion has been executed:

- A `ModelState` msg that specifies the ground truth location of the TurtleBot (if you had requested it).
- A `TurtleBotScan` msg that gives you the laser scan data. Gazebo simulates a laser scanner for the TurtleBot, that returns the distances to obstacles within the range of $-30°$ to $30°$, in increments of approximately $1.125°$ (for a total

of 54 range values - to which a Gaussian noise with standard deviation `scan_noise` is added). The range of the laser scanner is approximately between 0.45m-10m, so any landmark that is located outside this range will return a range value of `nan`.

You will need to set three ROS parameters: `scan_noise`, `rotation_noise` and `translation_noise`. These correspond to the standard deviations of the laser scan data, and the model of the rotational and translational motion respectively. By default, these are set to 0.1. It is reasonable to set these values in the ranges 0-0.125, 0-0.2, and 0-0.15 respectively, though it is recommended that you experiment with other values to test the robustness of your implementation.

You will receive example scan data and ground truth files (as well as the commands that were used to execute them) that have been generated by this software package, which you can use in order to test your algorithm. This means that the implementation of your solution can proceed independently from any Gazebo issues. But you should use this software in order to generate additional scan data and ground truth files in order to experiment with the effects of different noise levels on the localization algorithm's performance.

For each one of the environments, you will also have access to a text file that you can use to define a map of the world and predict the expected scans at different robot locations. These files are located inside `turtlebot_maps/map_x.txt`. Each of these files contains the following information, each separated by a $---$.

- The co-ordinates of the endpoints of the world. These co-ordinates define the walls that surround the world. The robot should not attempt to move beyond these walls.
- The co-ordinates of the endpoints of the different polygonal obstacles that are present in the world. The robot must not collide with any of these obstacles.

1.2 Questions and Deliverable

Given the map of the corresponding world, the robot's initial location, the odometry and the scan data, you are asked to provide your own implementation of the Sequential Importance Resampling algorithm for a particle filter approach to the localization problem.

In particular, build a software package that is able to load the map of the world, the scan data and visualize the robot's location (2D - could be just images) over time. Report on the accuracy of the approach against the ground truth data.

Run experiments for different levels of noise by using the available software to generate a variety of scan data examples and corresponding ground truth. Provide graphs that show the performance of the method (in terms of accuracy) as the noise level increases, as well as for different values in terms of the number of particles used (e.g., 100, 500, 1000, 2000, 5000, etc. - try at least 3 different values).

CS560 - Required compoment (extra credit for CS460): Perform experiments for an unknown initial robot location.