

Project 3: Robot Localization Using Particle Filters

Assigned: Oct 28, 2025

Deliverable 1 (Motion and Sensor Models) due: Nov 04, 23:59:59

Deliverable 2 (Particle Filter) due: Nov 11, 23:59:59

Note: Project 3 has a four-day late submission period for each deliverable.

Late submissions will lose 25% points per day of that specific deliverable.

In the project, you will enable an autonomous wheeled robot to localize itself in a given environment through a particle filter algorithm. You will utilize Gazebo in ROS to simulate a Turtlebot robot. The robot is equipped with a 2D LiDAR to perform sensing, and the robot is controlled using linear and angular velocity commands. A simulated home environment will be used for developing and evaluating your robot localization solution. Students are required to program this project using Python in ROS Noetic running on Ubuntu 20.04 LTS (i.e., the same development environment used in the previous projects). In addition, students must write a report following the format of standard IEEE robotics conferences using L^AT_EX.

Please **START EARLY!**

1 The Robot Localization Problem

Robot localization is the process of identifying where an autonomous robot is posed (including both position and orientation) with respect to its environment using a known map. Localization is one of the fundamental capabilities required by an autonomous robot since the knowledge of the robot's own pose provides essential state information for many downstream capabilities, such as path planning and navigation.

Successful localization should allow the robot to identify its location on a given map and gradually improve localization accuracy. A number of methods were used to address robot localization, such as odometry and place recognition with different pros and cons. In this project, students are required to solve the problem by defining motion and sensor models, and implementing a particle filter method to perform robot localization.

2 Gazebo Simulation of Turtlebot for Particle Filters

This project requires to use a simulated **Turtlebot** robot as illustrated in Fig. 1. For localization, the 360° 2D LiDAR on top of the Turtlebot robot will be used. Students are required to at least use the **house environment** (Fig. 2) provided by the Gazebo simulation. Additional maps can be used, but not required.

Starter Code: You may use the starter code for this project in the GitLab repository. The instruction of setting up the house environment is also included in the starter code:

<https://gitlab.com/peng-lab/particle-filter-ros2>.

Particle Filter launch file is included in the directory: `particle-filter-ros2/launch/`:

1. **Particle Filter launch file** spawns robot and sets up the house environment, as well as executing your code on particle filter for robot localization:

```
$ particle_filter.launch
```

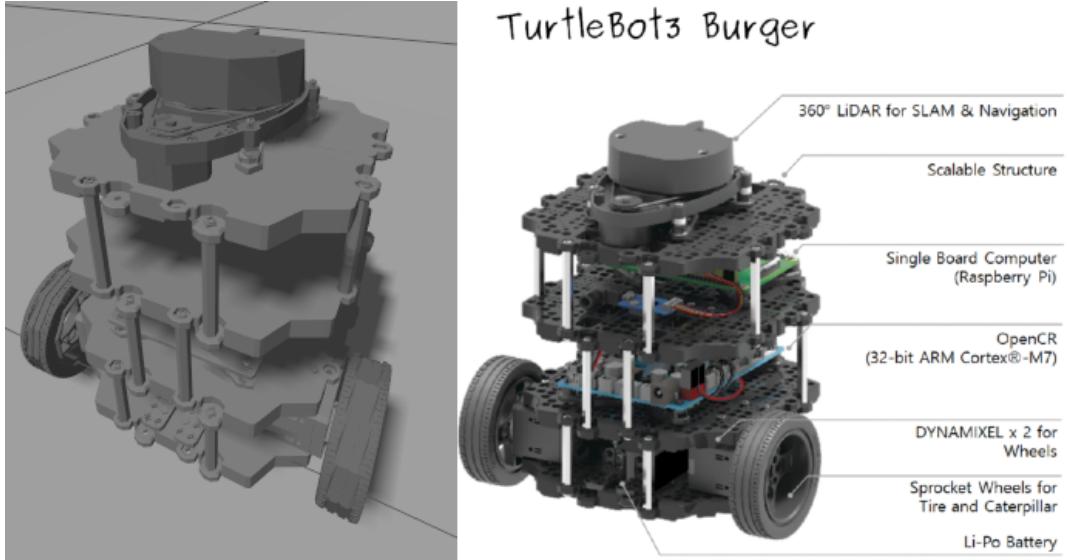


Figure 1: The turtlebot robot and its simulation in Gazebo.

3 Deliverable 1: Motion and Sensor Models

In the first part of the project, the main objective is to implement motion and sensor models. This deliverable has three tasks:

Task 1: Create a Map: You will use the mapping package and manual control to create a map of the house environment. You have also practiced building a map in Project 1.

You must save the map for future use, and submit a png file of the map (e.g., Fig. 2) as a part of Deliverable 1. It is okay to take a screenshot of the map for submission.

Please note that, before using the map, you may need to project the built map onto the simulated environment, i.e., each point on the map must accurately correspond to its respective location in the Gazebo simulation.

Task 2: Implement Motion Model: You will implement the motion model based on wheel odometry as we discussed in the lecture. Before implementing this model, please make sure you go through the lecture slides and the Chapter 6 of the textbook “Probabilistic Robotics” to understand the mathematical formulation of the motion model.

The odometry messages of Turtlebot are published on the topic `/odom`. More details can be found at:

https://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html

Task 3: Implement Sensor Model: You will implement the likelihood field model as your sensor model based on the LiDAR data. Before implementing the sensor model, please make sure to go through the lecture slides and the Chapter 5 of the textbook “Probabilistic Robotics.” At the minimum, your likelihood field model must include measurement noise. You may also implement the full model that includes measurement noise, maximum range, and randomness. The LiDAR data is published on the topic `/scan`. You may find more details from the above link.

You must compute the distance matrix as a lookup table (either applying Gaussian smoothing or not is okay), save it as a png file, and submit it in Deliverable 1. An example of this distance matrix or likelihood field

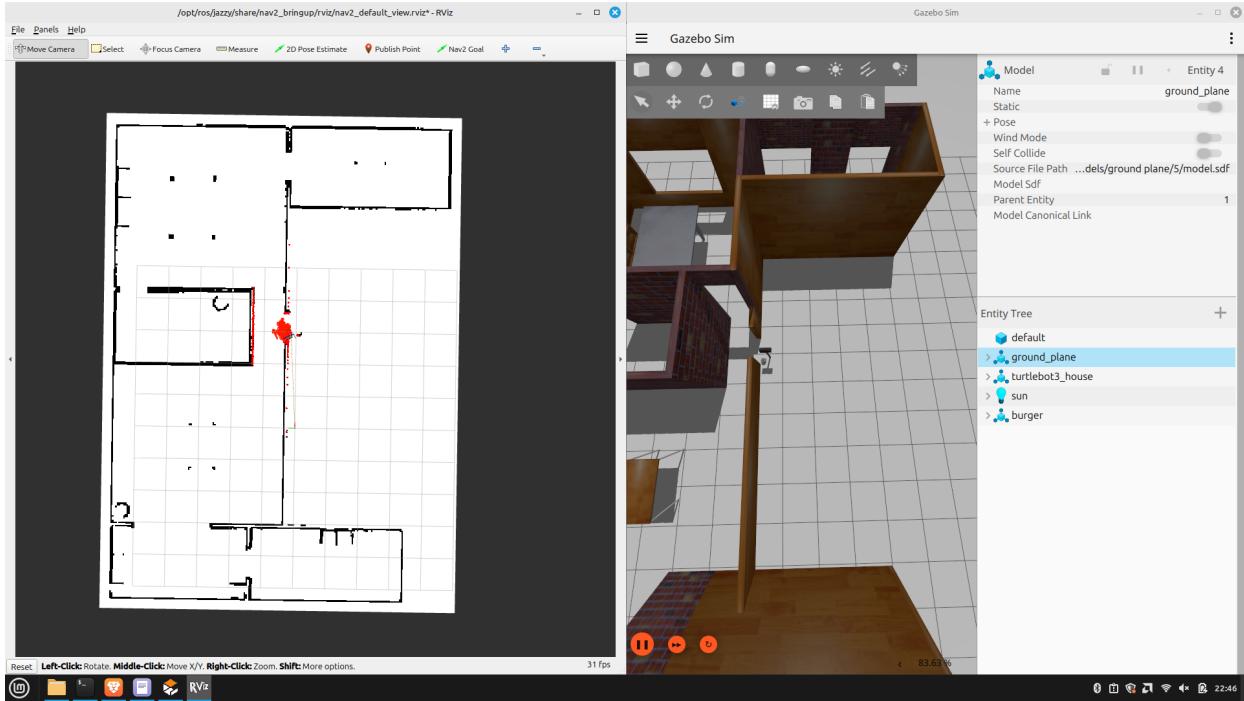


Figure 2: Gazebo simulation of the house environment, and its map built by GMapping for robot localization.

(from the lecture) is illustrated in Fig. 3.

What to Submit for Deliverable 1:

Students must submit a single tarball, named *P3D1_firstname_lastname.tar* (or *.tar.gz*), to the portal named P3-D1 in Canvas. The tarball must contain the following **three items**:

- Your **ROS package** (that must include all necessary package files, including Python scripts, launch files, package.xml, CMakeLists.txt, map files, etc.). The launch file must automatically show the robot in the environment in Gazebo (as demonstrated in Fig. 2). The package.xml file must provide sufficient information of your submitted package, and clearly describe how to use the launch file to run the demonstration; it also clearly state whether you use a starter package or program the project from scratch.
- A png file of your created metric map of the environment, with a format similar to the example in the left image of Fig. 3.
- A png file of your calculated likelihood field (or distance matrix), with a format similar to the example in the right image of Fig. 3.

4 Deliverable 2: Particle Filters

In Deliverable 2, the main objective is to implement a particle filter algorithm to perform robot localization. In the problem of robot localization, assuming a known map, a particle filter uses a motion model to predict the particles (i.e., hypothesis of robot poses) using odometry given a user-defined control, a sensor model to compute particle weights using sensor observations (e.g., LiDAR data) in order to correct the particles, and a resampling technique to convert a weight-based particle set to a frequency/density-based particle set. As the robot moves around in the environment and the particle filter continues repeating the prediction, correction



Figure 3: A metric map and the respective likelihood field (or distance matrix).

and resampling procedures, the particle filter is more likely to find the true pose of the robot.

Specifically, given a known map of an environment, the particle filter initializes a set of particles with random poses (i.e., locations and orientations) in the map. Then, the particle filter repeats the following steps:

1. Move the robot given a user-defined control, and capture the movement of the robot from odometry.
2. Predict the pose (location and orientation) of each particle based on the robot's movement using the motion model (the odometry-based motion model).
3. Correct the pose of each particle by using the sensor model (the likelihood field) to compute a weight for each particle from the LiDAR observations.
4. Resample with replacement a new set of particles using importance sampling.

What to Submit for Deliverable 2:

Students must submit a single tarball, named *P3D2_firstname_lastname.tar* (or .tar.gz), to the portal named P3-D2 in Canvas. The tarball must contain the following **two items**:

- Your **ROS package** (that must include all necessary package files, including Python scripts, launch files, package.xml, CMakeLists.txt, map files, etc.). The launch file must automatically show a working particle filter for robot localization in the Gazebo environment (as demonstrated in Fig. 2). The package.xml file must provide sufficient information of your submitted package, and clearly describe how to use the launch file to run the demonstration; it also clearly state whether you use a starter package or program the project from scratch.
- A demonstration video to show how your particle enables robot localization, including the initialization phase and iterations of the particle filter steps. Your demo video must illustrate how the moves in the environment and how the particles are updated, similar to the following example (blue dots are LiDAR readings): [\[Demo Link\]](#).

5 Grading

Your grade will be based upon the quality of your package implementations, the demo demonstrations, and the documentation of your findings in the report. Project 3 will be graded as follows (for a total of 10 points):

- Deliverable 1: 2.5/10 – You should have a ROS package with the motion model and the sensor model (Cannot be validated). You must submit a png file of the map and a png file of the likelihood field (or distance matrix) together with the ROS package.
- Deliverable 2: 7.5/10 – You should have a working implementation of the required robot localization algorithm, meaning that your code of the required algorithm is implemented as a ROS package, runs without crashing in ROS and Gazebo, and performs robot localization in the given map. You should also create and submit a video demo to demonstrate that the robot is able to localize itself using your designed particle filter.