

Localization Project: Where am I?

Christoph Doerr

Abstract—This study shows two models of robots created using URDF in a ROS Gazebo/Rviz setup. These robots are used to navigate through a given map to reach a defined goal pose. The robots are set up with a camera and a laser range finder (LIDAR). To process these sensor data, the ROS navigation stack is implemented with an AMCL(Adaptive Monte Carlo Localisation) ROS package. The parameters of the navigation package are tuned within this study to increase the performance of the robots at the localization/navigation task.

Index Terms—Robot, IEEEtran, Udacity, \LaTeX , Localization, Navigation, ROS, Kalman Filter.

1 INTRODUCTION

IN mobile robotics, the problem of localization consists of determining the robots pose (position and orientation) within a environment using sensor data and control inputs. In real world, sensor data as well as control inputs are not accurate and uncertain, which means that localization techniques have to take into account the uncertainty of control commands and sensor inputs. One option to perform the localization tasks including these uncertainties offers the AMCL(Adaptive Monte Carlo Localization) algorithm. As part of the project, this AMCL algorithm is implemented and tuned within two different robots. One robot description was provided and the other one was created from scratch. The base source code of this project can be found at <https://github.com/udacity/RoboND-Localization-Project> and the implementation of the localization task can be found at <https://github.com/chrisy-d1989/udacityRSENano/tree/master/RoboND-Localization-Project>.

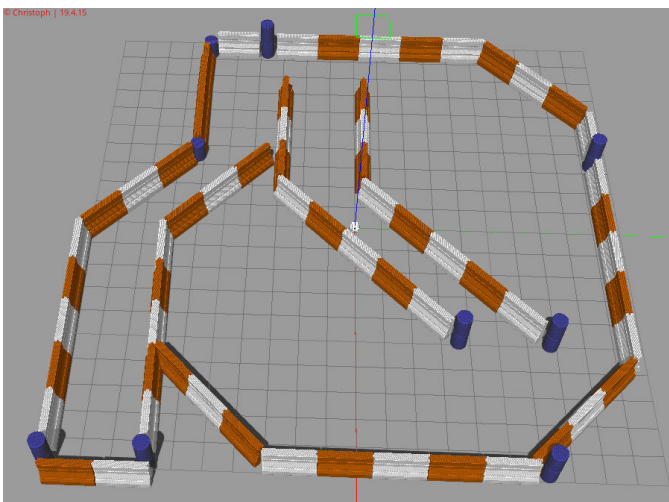


Fig. 1. Project Map

2 BACKGROUND / FORMULATION

Localization is a fundamental task in mobile robotics. To perform this task, robots are equipped with different sen-

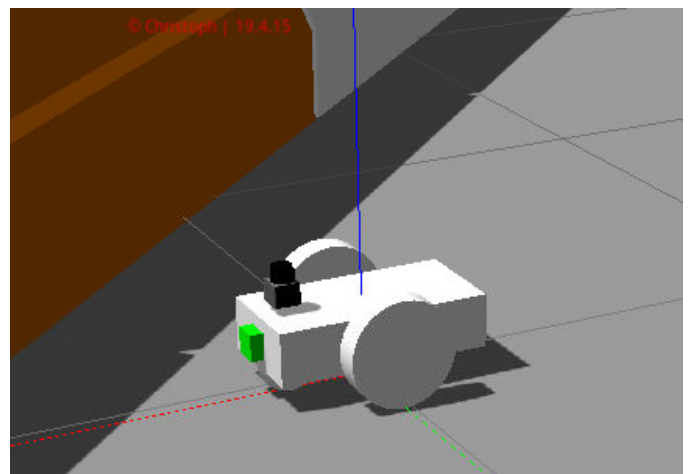


Fig. 2. Udacity Robot Model

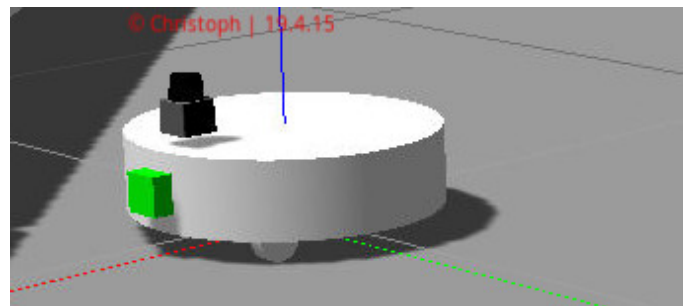


Fig. 3. Cleaning Robot Model

sors to detect their surrounding environment. Furthermore, control inputs are used to further specify its pose. However, both sensor data and control inputs are noisy and uncertain. Therefore, the localization algorithms have to perform within these uncertainties and calculate a approximate pose of the robot. The two most common used localization algorithms are the Extended Kalman-Filter and Monte Carlo Localization.

2.1 Kalman Filters

The Kalman-Filter is a very popular solution in control theory. Its advantages range from taking in noisy measurements in real time to providing accurate prediction of actual variables such as poses. The Extended Kalman-Filter (EKF) addresses the limitations of the Kalman-Filter of linear motion and measurement models as well as the assumption of a unimodal Gaussian distributed state space with linearizing the nonlinear motion and measurement function. However, the mathematics is complex and the computational costs are very high.

2.2 Particle Filters

Particle Filters work by distributing particles uniformly throughout the known map. These particles get weighted based on sensor data, that is received by the robot. These particles get resampled by their likelihood of the particle pose to the robot pose. As the robot moves through the map, the particles converge to the estimated robot pose. The Monte Carlo Filter is a type of particle filter that uses Monte Carlo simulation for its location likelihood calculation [1].

2.3 Comparison

Particle Filters presents many advantages against the EKF:

- 1) easier to implement
- 2) easier to control computational memory and resolution
- 3) unrestricted to Gaussian probability distributions and linear models of measurement and actuation

3 DATA ACQUISITION

The simulation is done in a ROS Gazebo/Rviz environment as seen in figure 1 and the localization task was performed with two different robot models, which were built in URDF format. The first robot model for the project is called `udacity_bot` and the other one is called `cleaning_bot`. The differences will be discussed in the upcoming sections.

3.1 Model Desings

The model of `udacity_bot` is shown in figure 2 and the self designed cleaning robot is shown in figure 3. Both robots are equipped with a front facing camera mounted to the front and a Hokuyo LIDAR attached to the top. The robot models can be found in the `udacity_bot.xacro` and `cleaning.xacro` as well as the `udacity_bot.gazebo` and `cleaning.gazebo` files. The main difference between the robots is their main body, where the body of the `cleaning_bot` should simulate a vacuum robot.

3.2 Packages Used

The packages used for the localization task are "amcl" and "move_base". "amcl" performs a AMCL algorithm the generate the robots pose in the provided map and "move_base" created local and global costmaps, as well as calculates local and global trajectories to the goal pose. The project is performed in the `jackal_race` map provided by Udacity.

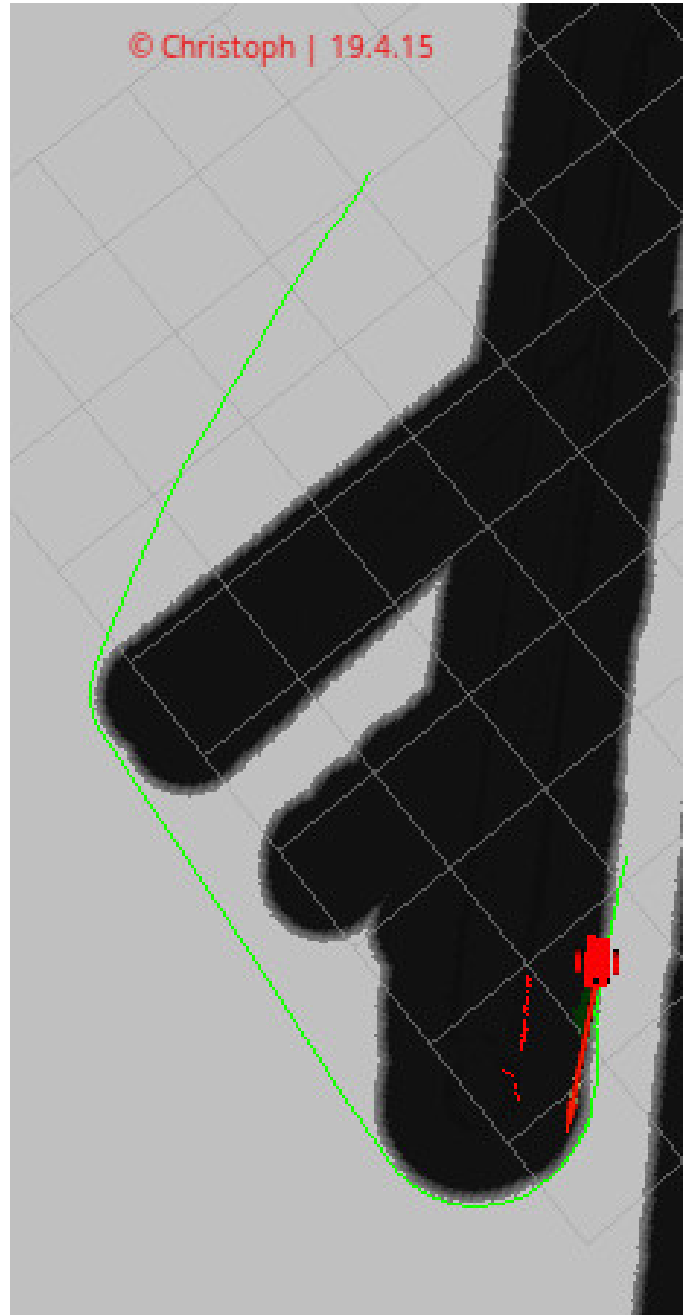


Fig. 4. Udacity robot on the way to the goal

3.3 Parameter Tuning

The main task, besides setting up the navigation stack for the localization, is to tune the parameters of the AMCL as well as the `move_base` node to improve the localization performance. The following list shows the tuned parameters in the two nodes:

- the min and max used particles of the particle filter were set to 20 and 200 respectively. The initial values of 100 and 5000 came with to high computational costs for the simulation.
- The `transform_tolerance` was set to 0.2
- `odom_model_type`.
- The `obstacle_range` and `raytrace_range` were set to 5.0 and 8.0 respectively. The `obstacle_range` is the de-

fault distance from the robot at which an obstacle will be inserted into the cost map and the `raytrace_range` is the range at which to raytrace out obstacles from the map.

- Both `update_frequency` and `publish_frequency` were changed to 5Hz
- The `yaw_goal_tolerance` was set to 0.05 radians and the `xy_goal_tolerance` was set to 0.05 meters to achieve the end goal.

Figure 4 shows the `udacity_bot` on the way to its goal. The red dots show the LIDAR points on the wall of the right side of the robot. The green line visualizes the global planned path and the red arrow shows the current pose of the robot. Figure 5 shows the `cleaning_bot` with distributed particle poses (green arrows) as well as the LIDAR points in red. Figure 4 and 5 are both taken in the Rviz environment.

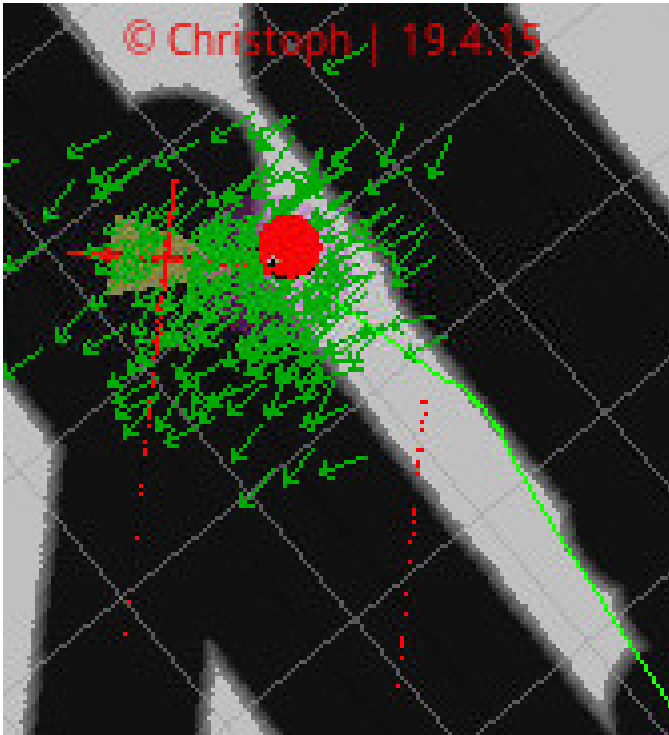


Fig. 5. Cleaning Robot with Particle Poses

4 RESULTS

4.1 Localization Results

Both robots were able to perform the task of navigating to a specific place in the given map. The `udacity_bot` reached the goal after ≈ 2 minutes, whereas the `cleaning_bot` reached the goal after ≈ 15 minutes. The `cleaning_robot` started wondering around for a bit before getting to the main path of the global planned path. Figure 6 and figure 7 visualize both robots in their final goal pose in the Gazebo environment.

5 DISCUSSION

Both robots could localize themselves in the given map and reach the given goal. The `udacity_bot` performed better and

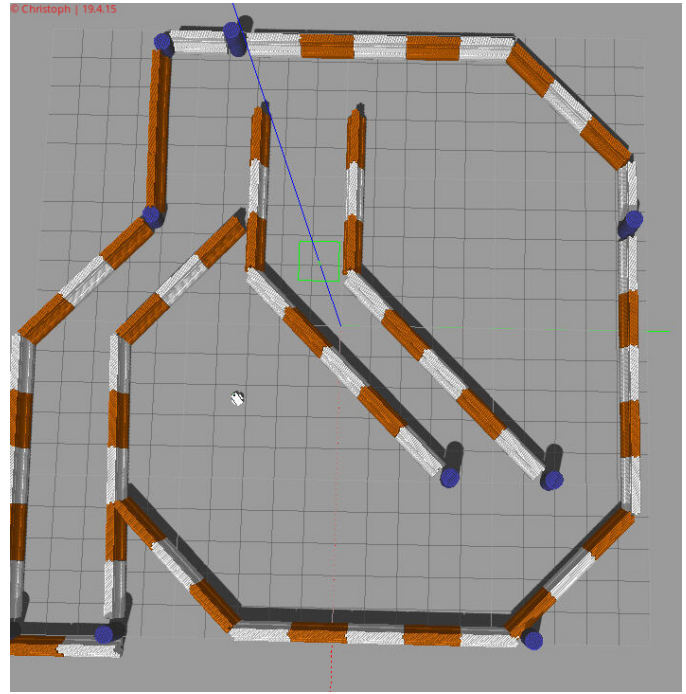


Fig. 6. Reached Goal by Udacity Robot

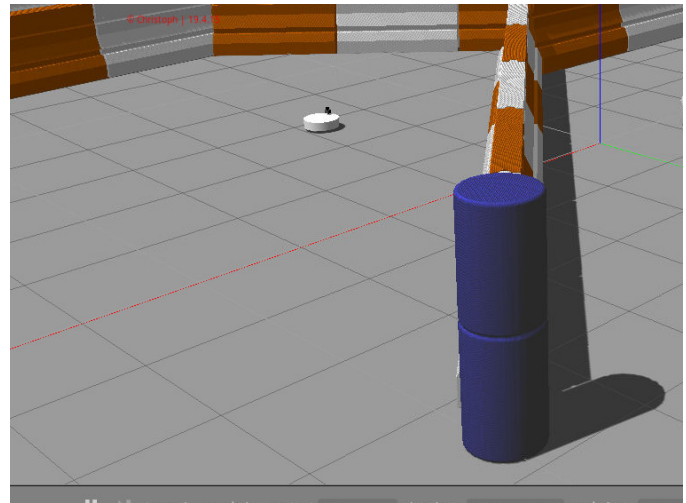


Fig. 7. Reached Goal by Cleaning Robot

reached the goal faster compared to the `cleaning_bot`, which needed five times longer than the `udacity_bot`. The robots performed well in the described task, but they wouldn't work well in the kidnapped robot problem, because the AMCL does not work well in these problems. Both AMCL and MCL would work well in any industry domain, where clear barriers guide the path of the robot. Furthermore, the ground needs to be flat and free of obstacles. For example, industry domains could be vacuum or lawnmower robots.

6 CONCLUSION / FUTURE WORK

To improve the current work, further tuning of parameters of the path planner would be necessary. Trajectory scoring parameters like `pdist_scale` or `gdist_scale` could be implemented. Furthermore, an increase of computational power

would be increase the accuracy of the particle filter with a higher number of particles.

REFERENCES

- [1] Udacity, *Udacity's Robotics Softwareengineer Nanodegree, Localization Lessons*. Udacity, 2019.