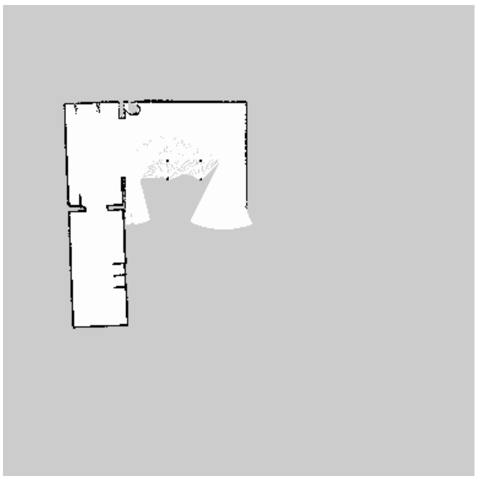
Laboratory of Industrial Robotics M
Simulation Exercises with ROS and TurtleBot3 burger
Fabio Lisurici 903858 Dragos Froicu 903618

Report

Part 1- Autonoumous Exploration and SLAM

The first step for solving the different exercises is to download and configure the TurtleBot3 packages which can be found at the following link: http://emanual.robotis.com/. Afterwards, in order to solve the first and the second exercise the file test.launch has been created in the turtlebot3_gazebo package. The first thing that this file does is to set up the gazebo house scenario. At the beginning the model of the robot and its configuration is set up, then the house_world is loaded in Gazebo with its parameters and then robot URDF is spawned.



The second part of this file concerns of the autonoumous exploration of the Turtlebot3. Here the turtlebot3_drive has been run. This node already exits in the robot packages and it checks the data from the lidar. Based on the frontal and sides distances of the robot with respect to the objects this node decides what velocity input to publish on the /cmd_vel topic in order to avoid them. The last part of the test.launch file deals with the mapping problem.

In order to solve this problem the gmapping ROS package has been used. This package provides laser-based SLAM and it creates an occupancy grid map. In order to lunch the test file the following terminal command can be used:

```
roslaunch turtlebot3_gazebo test.launch
```

As output Gazebo and Rviz will be run and the robot will start moving autonomously. In Rviz the current map can be seen. After a while, when the map is sufficiently large it can be saved by using the following command:

```
rosrun map_server map_saver -f ~/map
```

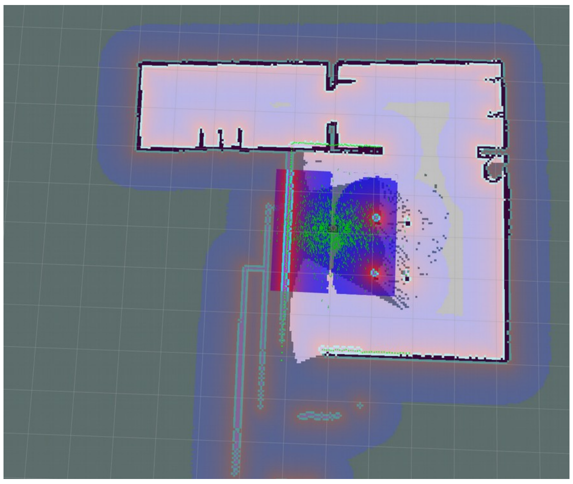
This map will be used further by the navigation stack.

Part 2 – Alignment and Navigation

In order to navigate the robot needs a map of the world, a starting location, and a goal location. Once the map has been saved the navigation system can be used. The navigation stack can be launched by using the navigation. launch file. This file will set up the house scenario again, and it will run the move_base node and the amcl localization system. The move_base node provides the implementation of a ROS action that, given a goal on the map, will attempt to reach it. Amcl is a probabilistic localization system for moving a robot in a 2D. It implements the adaptive Monte Carlo localization approach, which uses a particle filter to track the pose of a robot against a known map.
In order to lunch the navigation stack the following terminal command can be used:

roslaunch turtlebot3_navigation navigation.launch map_file:=$HOME/map.yaml

At this point in order to be able to use the navigation system it needs to be aligned with the map.



The initial pose of the robot needs to be set up. In order to do this the python node alignement.py has been created. This node creates a subscriber and a publisher. The subscriber listens on the topic /odom the current position and orientation of the base_footprint with respect to the map and publishes these information on the topic /initialpose. In order to run this node the following terminal command can be used:

rosrun navigation alignment.py

The alignment can be done also by using rviz. You can do this by clicking on the "2D Pose Esti-mate" button, then clicking, holding, and dragging in the rviz window. In this case Rviz does the same thing, it publishes a message of type geometry_msgs /PoseWithCovarianceStamped on the /initialpose topic.
Once the alignment has been set up, we can start interacting with the navigation stack by using the navigation.py node.

rosrun navigation navigation.py

The navigation stack works in the following way:

- a goal of type MoveBaseGoal is sent to the navigation stack by using an action call. This goal specifies a position and an orientation in the map frame(a sequence of positions and orientations can be also sent).
- A path-planning algorithm is used to plan a feasible path to the goal (global planner).
- The path is passed to the local planner that drives the robot to the goal avoiding obstacles by using the lidar sensor.
- The action terminates when the robot gets sufficiently close to the goal.

In order to communicate with the navigation stack the navigation.py has been created. This node has a sequence of goals, it calls the move_base action and then it waits for it to terminate.

Part3 – Navigation plugin

The ROS navigation stack has a quite complex structure. It contains a local and a global planner and static and dynamic cost maps which are used by the path planning algorithms. They store and maintain information in the form of occupancy grid about the obstacles in the environment and where the robot should navigate. In this section the modification of the navigation stack is covered. A new global planner can be added in the navigation stack as a navigation plugin in order to replace the default one. The integration has two main steps: writing the planner class and deploying it as a plugin so that it can be used by the move_base package. Because of little experience in C++ a new planner hasn't been implemented from scratch. In this part the default global planner has been changed by using the simpler CarrotPlanner, already implemented in ROS. In order to use the new planner the move_base launch file need to be modified by simply adding the new following line:

```
<param name="base_global_planner" value="carrot_planner/CarrotPlanner" />
```

Some tests have been performed by using the new global planner.

Sources

Quigley, Morgan, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

Koubâa, Anis, ed. *Robot Operating System (ROS)*. Verlag: Springer, 2017.

Newman, Wyatt. *A Systematic Approach to Learning Robot Programming with ROS*. CRC Press, 2017.

https://hotblackrobotics.github.io/en/blog/2018/01/29/action-client-py/
https://www.theconstructsim.com/read-laserscan-data/
http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
http://wiki.ros.org/ROS/Tutorials
http://www2.ece.ohio-state.edu/~zhang/RoboticsClass/index.html