Student: Manuel Rinaldi

Course: Robotics Software Engineer (Udacity nanodegree)

Home Service Robot

Table of Contents

Overall presentation

The project is divided in 5 scripts. In the first 4 scripts we have tested the components separately and in the last script we have put everything together in the project Home Service Robot.

The home service robot project simulates a robot capable of picking objects from one point in the map and drop them off in another point.

To execute these steps the robot needs first to create a map. Usually creating a map is a process of moving the robot following the walls and mapping all the location. This can be done by using the SLAM algorithm. Once the map is created then using the AMCL package we can set goals to the robot and expect he will reach the goal creating a graph of the path it will take. The robot uses the Dijkstra's algorithm, a variant of the Uniform Cost Search algorithm, to plan its trajectory from start to goal position.

I have created my own assets package to contains the scripts, world and map for all the following steps.

Package overview

- gmapping: With the gmapping_demo.launch file, you can easily perform SLAM and build a map of the environment with a robot equipped with laser range finder sensors or RGB-D cameras.
- turtlebot_teleop: With the keyboard_teleop.launch file, you can manually control a robot using keyboard commands.
- turtlebot_rviz_launchers: With the view_navigation.launch file, you can load a preconfigured RVIZ workspace. You will save a lot of time by launching this file, because it will automatically load the robot model, trajectories, and map for you.
- turtlebot_gazebo: With the turtlebot_world.launch you can deploy a turtlebot in a gazebo environment by linking the world file to it.
- pgm_map_creator: With this package you can create a pgm map using and exist world in gazebo.
- add_markers: With this package I have drawn the virtual objects in the RVIZ.
- pick_objects: With this package I have sent goals to the robot using the MoveBaseClient.
- assets: With this package I have organized the map, world and scripts files.

Let us go through the steps one by one.

Test Slam

In this step we create the map. We used the gmapping package (which implements the SLAM algorithm) from ROS and the turtlebot robot as robot model. The office world was created by me using AutoCAD, then exported the drawing in jpeg and used it in gazebo and with the building editor.

In Figure 1 we can see the gazebo office world and the robot in one of the rooms.

Figure 1 Gazebo Office World in test slam

In figure 2 below we can see that moving the robot is creating the map. RVIZ listen to the map topics and display it in the main aera.

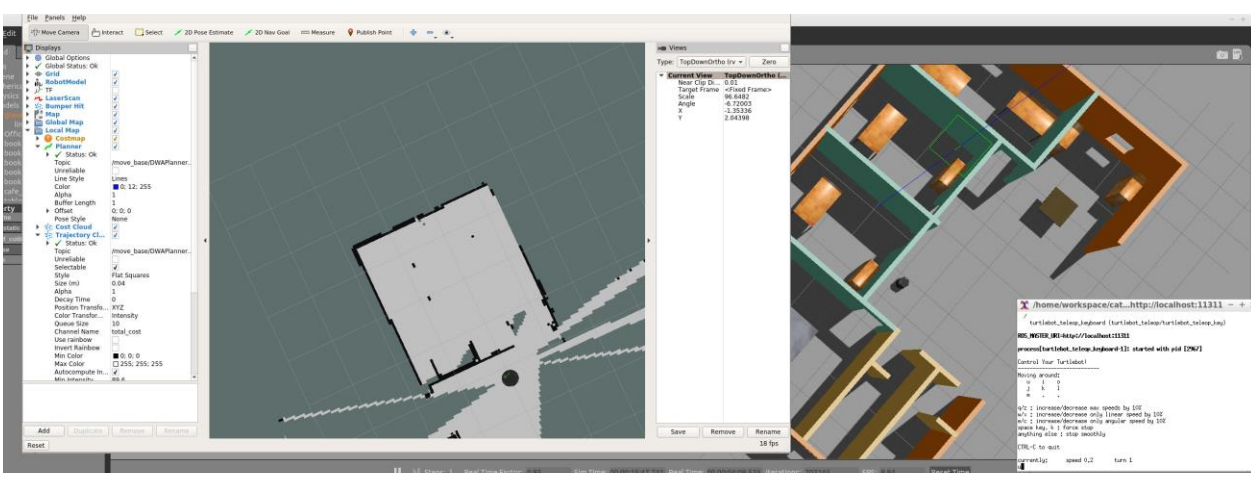

Figure 2 RVIZ And Gazebo test slam

Once we have a map, we can use it to navigate in it. For the purpose of this project I have used the pgm_map_creator package in order to extract a map from the world. It uses a libcollision_map_creator.so gazebo plugin to check the collision in the world and based on them it draws lines in the map.

Test navigation

Once we have the map, we use the AMCL package which is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach, which uses a particle filter to track the pose of a robot against a known map.

For the purpose of this test we used the RVIZ 2D nav goal button to set a goal for the robot and check that moves to the right position.

We can notice that when RVIZ opens we can see a lot of green vectors. These vectors are created by the particle filter randomly (RVIZ reads them by using the particlecloud topic) and then after some iterations, the filter will keep alive only those that are most likely to be in the robot position.

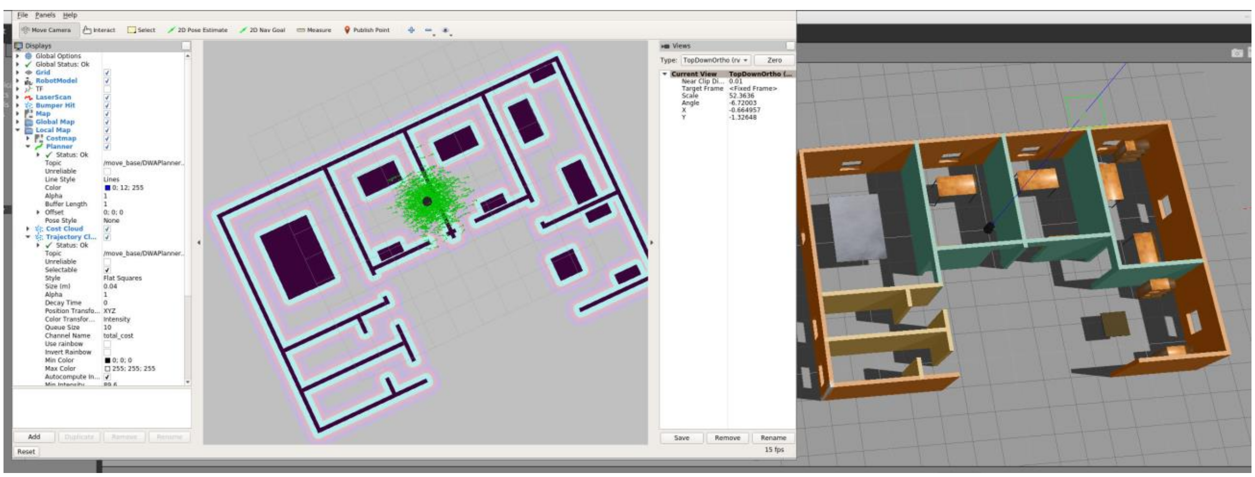

Figure 3 RVIZ particles and map

We can see that once the goal is set, the robot starts to move toward the goal. The walls are obstacles and they have a tolerance of the robot width.

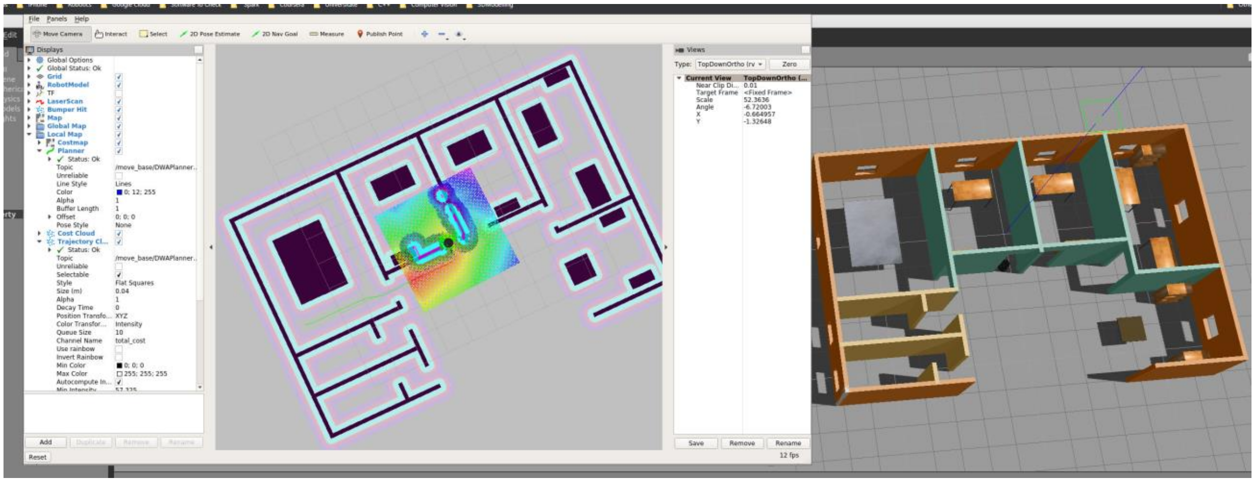

Figure 4 RVIZ robot moving toward the goal

Pick objects

We have now a robot capable to move in a world. We can create our own package which can send goals to the robot using the MoveBaseClient of the ROS packages.

In figure 1 we have sent a goal of (x: -5.86, y: 1.89) as pickup location. Then after 5 seconds we have sent the location (x: 5.36, y: 1.19) as drop off location. This can be seen in figure 2.

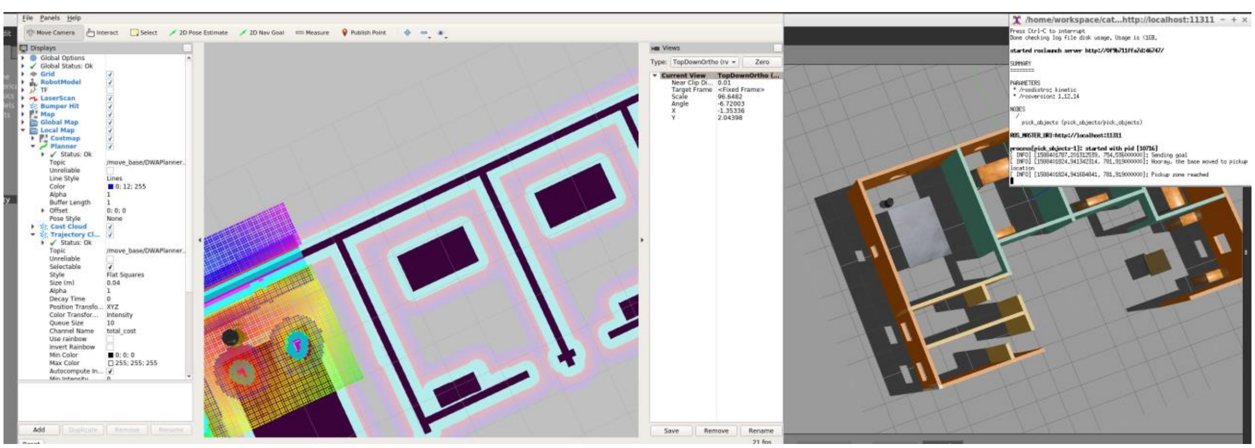

Figure 5 Pickup location

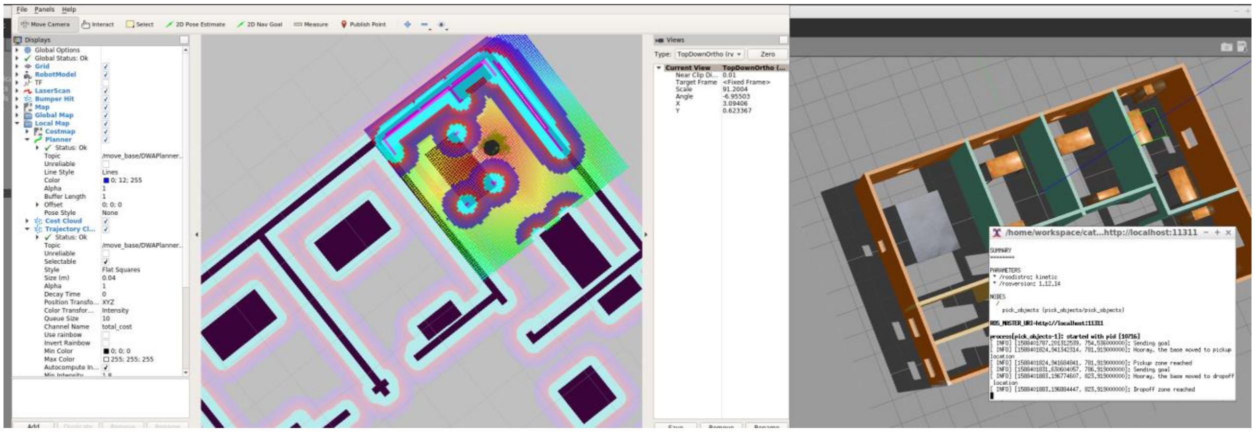

Figure 6 Dropoff location

Add markers

When we simulate robots in a virtual environment sometimes, we may need to add markers in RVIZ to simulate virtual objects. We can do that by publishing in a new topic that I have called visualization_marker which can be used in RVIZ as marker topic.

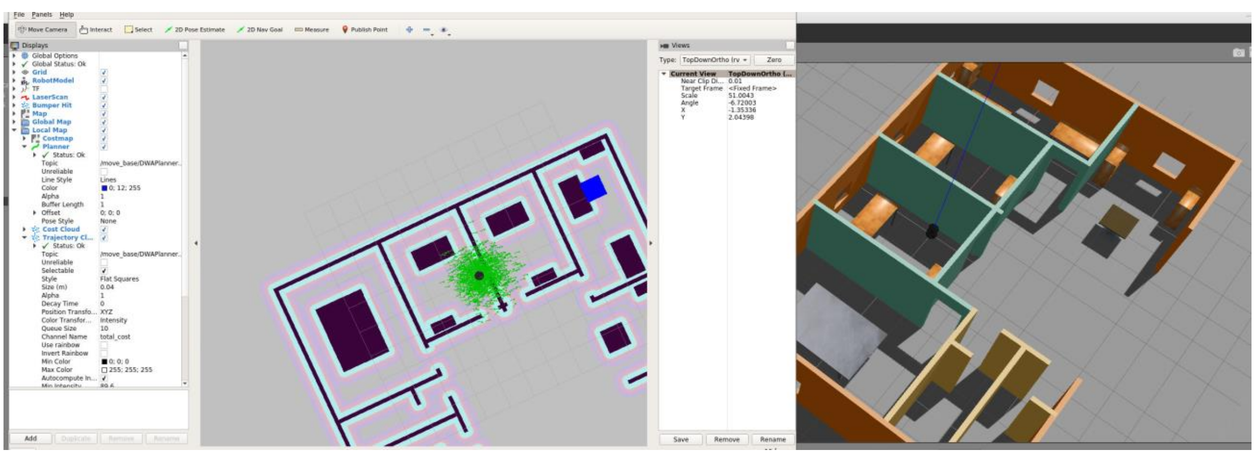

Figure 7 A marker displayed in RVZI

Home service robot

In the final project we have used all this packages in order to create a real simulation of a home service robot capable to navigate to a goal and pickup an object then navigate to another goal and drop off the object. We have used two custom packages and the AMCL package to achieve this goal.

The first step in figure 8 is to send the robot using the pickup object package to the marker created by the add_markers package. This add_markers package was modified to listen to the odom topic which is published with the actual position of the robot. The position is published with two vectors, one for the position and one for the orientation. So, using the position we can check that the robot reached the pickup or the drop off location.

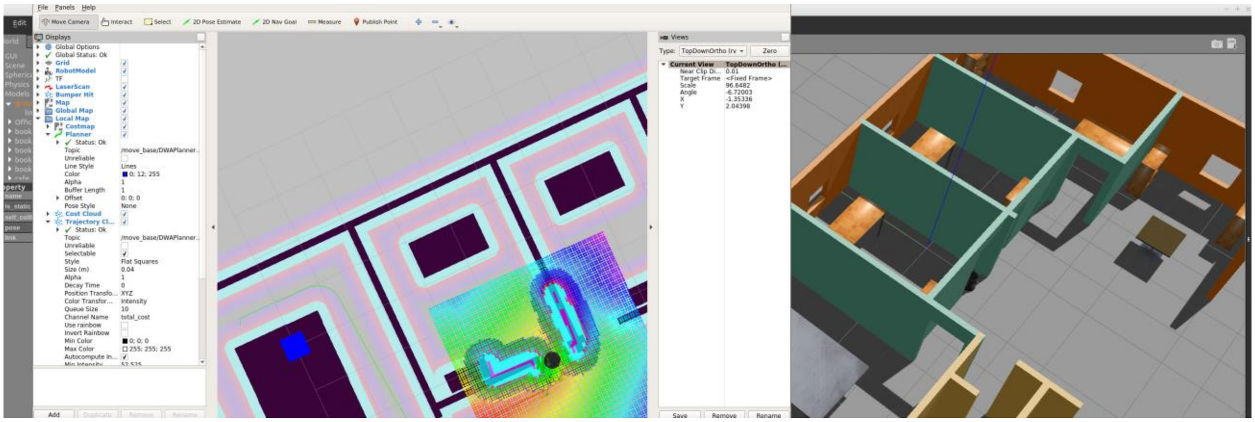

Figure 8 Robot is moving toward pickup location

In figure 9 the robot reached the pick-up location and we removed the marker.

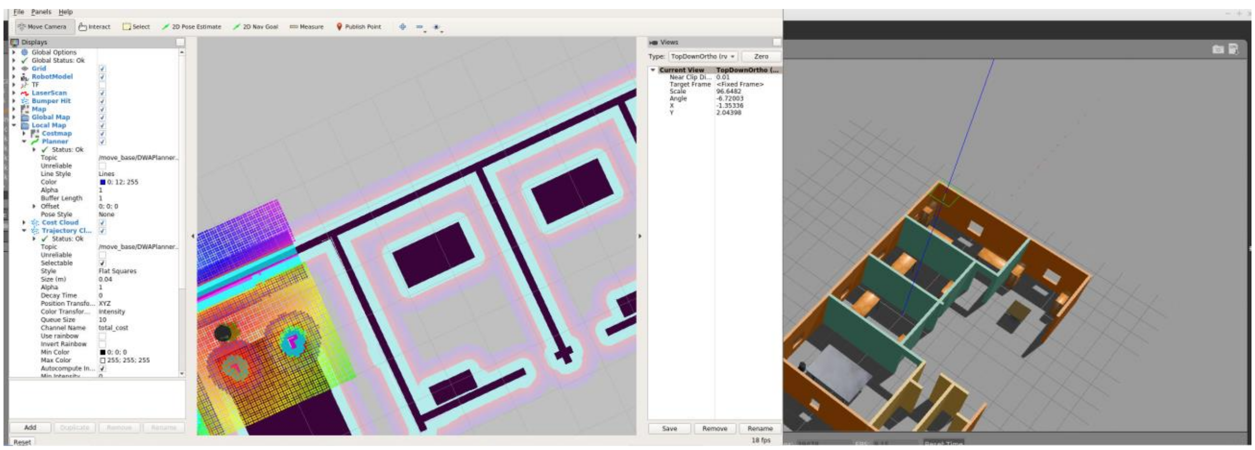

Figure 9 Robot is moving toward drop off location

After 5 seconds we have sent the robot to the drop off location as can be seen in figure 10.

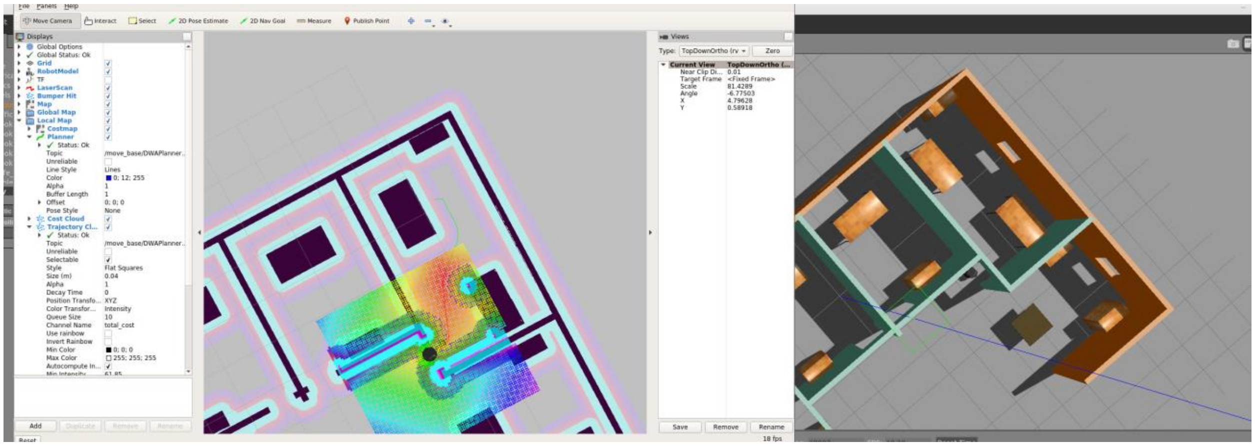

Figure 10 Robot has reached pickup location

When the robot reached the drop off location the add_markers, listening to the odom topic, displayed the marker at the drop off location.

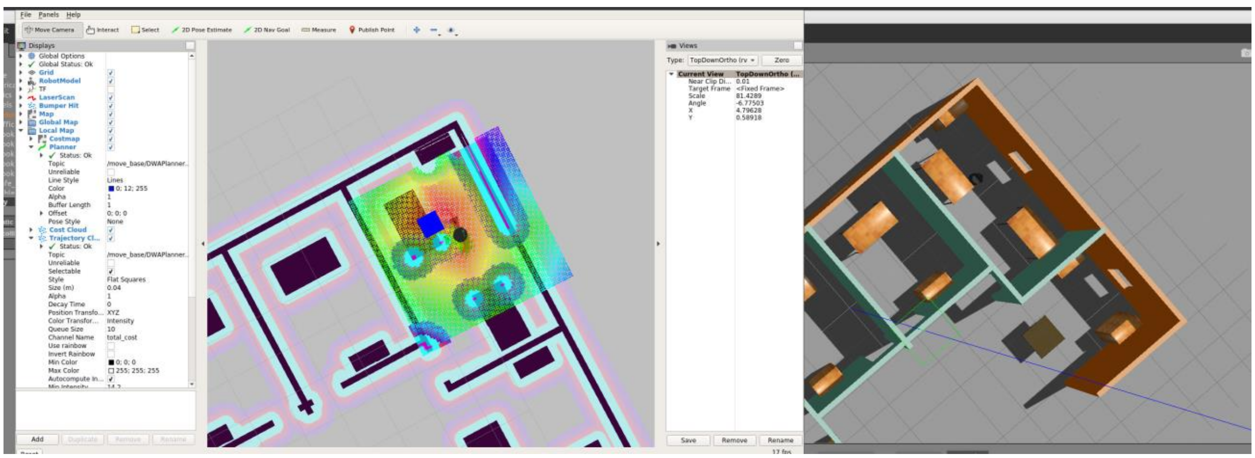

Figure 11 Robot has reached drop off location

Table of figures