Home Service Robot Project

Carlos E. Sanoja

Abstract

In this project is presented a home service robot that autonomously map a simulated environment and navigate to pickup and deliver objects.

1. Design a simple environment with the Building Editor tool in Gazebo

For this project were designed several environments to test the algorithms. All of them can be found in the World folder of the repository. The one selected to analyzes is presented in Fig. 1.

Figure 1: Custom Gazebo environment

To accomplish this objective were performed several things:

- Create the world with the editor
- Save the custom map
- Create a package named "folders" to make easy find the file.

2. Robot Model

It is used a turtlebot for this project

3. Catkin Workspace

To program the home service robot, it will be needed to interface it with different ROS packages. Some of these packages are official ROS packages which offer great tools and others are packages that were created.

3.1. Official ROS packages

1. **gmapping**: With the gmapping_demo.launch file, it is possible perform SLAM and build a map of the environment with a robot equipped with laser range finder sensors or RGB-D cameras.
2. **turtlebot_teleop:** With the keyboard_teleop.launch file, can be manually controlled a robot using keyboard commands.
3. **turtlebot_rviz_launchers**: With the view_navigation.launch file, can be loaded a pre-configured rviz workspace.
4. **turtlebot_gazebo**: With the turtlebot_world.launch can be deployed a turtlebot in a gazebo environment by linking the world file to it.

3.2. Created packages

1. **folders**: inside this packages are saved the world and launch files for this project.
2. **wall follower**: store a wall_follower node that will autonomously drive your robot around to perform SLAM.
3. **pick objects:** node that commands your robot to drive to the pickup and drop off zones.
4. **add markers:** node that model the object with a marker in rviz.

4. Mapping

4.1. Manually test SLAM with a robot inside your customized environment

- **xterm -e " roslaunch turtlebot_gazebo turtlebot_world.launch " :** Launch turtlebot in the custom world.

  To successfully launch the world have to be edited the launch file inside the turtlebot world launch. The direction was edited to point to the

custom world inside the World folder. In addition to, is added a delay of 20 seconds to open the next script to ensure gazebo has been opened correctly.

- **roslaunch turtlebot_gazebo gmapping_demo.launch :** Launch gmapping demo. Here is important to note that for this launch was set up the kinect 3D sensor as default for this project.
- **roslaunch turtlebot_teleop keyboard_teleop.launch:** Launch turtlebot teleoperation.
- **roslaunch turtlebot_rviz_launchers view_navigation.launch:** Launch rviz.

Figure 2: Test SLAM teleop

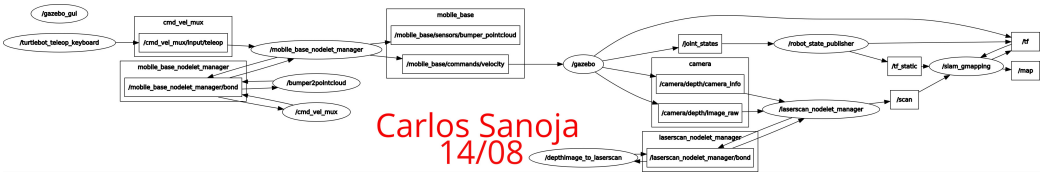

Figure 3: Nodes and Topics list

4.2. Write a wall follower node that will autonomously map your environment

- the node name was changed to wall_follower_NODE

- Inform ROS master that we will be publishing a message of type geometry_msgs::Twist on the robot actuation topic with a publishing queue size of 100. Topic: /cmd_vel_mux/input/navi

- Subscribe to the /scan topic and call the laser_callback function

4.3. write a wall_follower.sh script file and launch it to autonomously perform SLAM.

- **xterm -e " roslaunch turtlebot_gazebo turtlebot_world.launch " :** Launch turtlebot in the custom world.

- **roslaunch turtlebot_gazebo gmapping_demo.launch :** Launch gmapping demo. Here is important to note that for this launch was set up the kinect 3D sensor as default for this project.

  To successfully map the environment was tunned the parameters of the gmapping parameters. Thats because the gmapping parameters values used were the default values. In general, its essential to tune them in order to get a 100% accurate map. When those parameters aren't tunned, the map presents several problems: the size doesn't correspond with real world size and mainly have parts at the boarders that doesn't described correctly the map just as Fig.4.

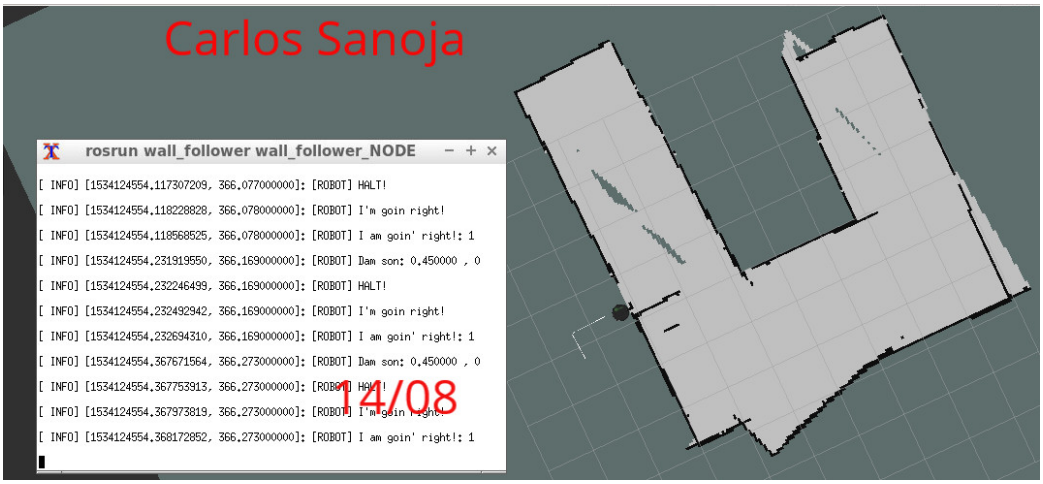

Figure 4: Test wall follower map in initial U map designed for testing

To avoid this issue and as kinect was settle as default senor for this project it was necessary to edit the parameters in the following direction

(see gmapping.launch):

turtlebot_navigation/launch/includes/gmapping/kinect_gmapping.launch.xml

1. particles: 150
2. Minimum score: for considering the outcome of the scan matching good. Can avoid jumping pose estimates in large open spaces when using laser scanners with limited range. Scores go up to 600+, try 50 for example when experiencing jumping estimate issues. it was set to 1250.
3. Linear and angular update: 0.1 and 0.2 correspondingly.
4. maxrange: 8.0

- **roslaunch turtlebot_rviz_launchers view_navigation.launch:** Launch rviz.
- **rosrun wall_follower wall_follower_NODE :** run wall follower

Figure 5: Test wall follower map in custom world

As can be seen in Fig.5 the map was correctly map with high accuracy after a few minutes. In Fig.6 is shown the added nodes and topics to the project. Finally, the map was saved in the World folder under a pgm and yaml format.

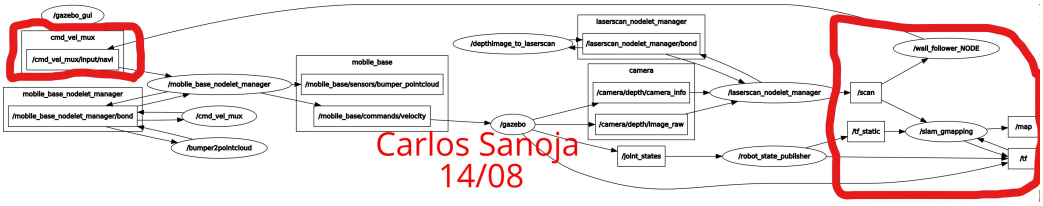

Figure 6: Nodes and Topics list

5. Navigation

5.1. Manually test navigation with your robot inside your environment.

- **roslaunch turtlebot_gazebo turtlebot_world.launch**
- **roslaunch turtlebot_gazebo amcl_demo.launch**: The map file was set to /mymap5.yaml and the default 3D sensor was kinect. also it is launch the move_base node to perform the navigation task.
- **roslaunch turtlebot_rviz_launchers view_navigation.launch**

The task of use the 2D Nav Goal in rviz and command the robot to navigate to two different locations was perform successfully.

Figure 7: Test navigation task

5.2. Write a node that will communicate with the ROS navigation stack and autonomously send two goals for your robot to reach.

The robot has to travel to the desired pickup zone, display a message that it reached its destination, wait 5 seconds, travel to the desired drop off zone, and display a message that it reached the drop off zone. It was provide a template and it was added and edited as follow:

- Node name: pick_objects
- frame id: /map
- set pick and drop goals:

$$pick = [-6.0, -9.25];$$

$$drop = [-1.0, -0.0];$$

- publish over /robot_position when the robot reached the target position. This message is used to indicate when the market must appear or disappear in the next section.

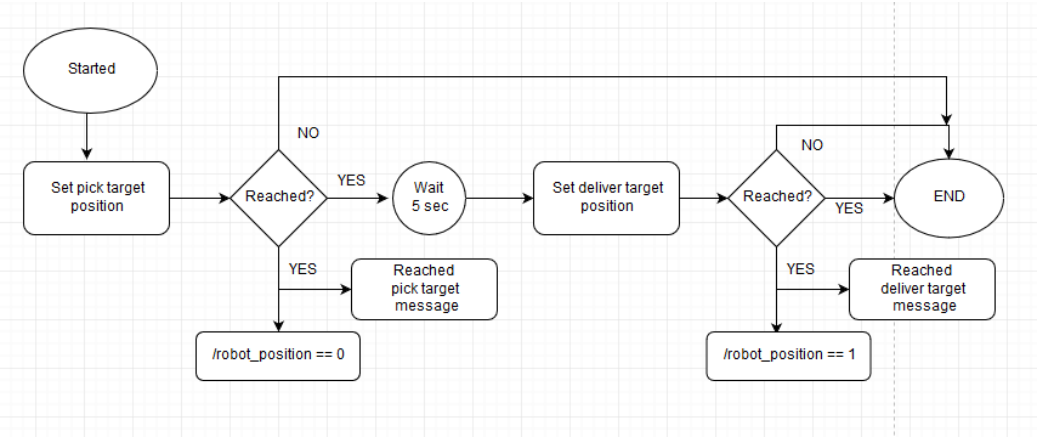

Figure 8: pick and delver algorithm

5.3. write a pick_objects.sh file that will send multiple goals for the robot to reach.

- **roslaunch turtlebot_gazebo turtlebot_world.launch**
- **roslaunch turtlebot_gazebo amcl_demo.launch**: The map file was set to /mymap5.yaml and the default 3D sensor was kinect. also it is launch the move_base node to perform the navigation task.
- **roslaunch turtlebot_rviz_launchers view_navigation.launch**
- **rosrun pick_objects pick_objects**

Finally, the robot perform the operation successfully and reached both target positions (Fig. 9). It was set the position as far as possible to test the algorithm. In Fig.10 is shown how the nodes are added to the project to build and perform the navigation task autonomously.

Figure 9: Test navigation task

6. Markers

6.1. write a add_marker.sh file that will publish a marker to rviz

- **roslaunch turtlebot_gazebo turtlebot_world.launch**

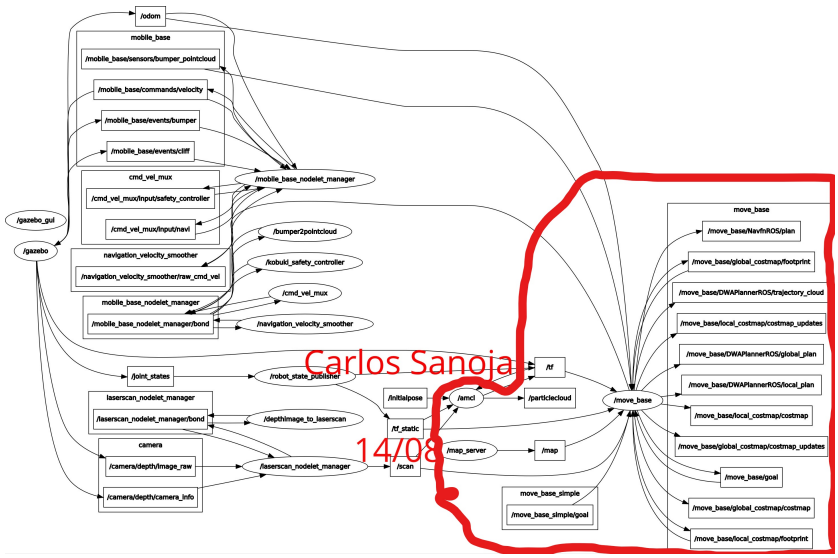

Figure 10: Nodes and Topics list

- **roslaunch turtlebot_gazebo amcl_demo.launch**: The map file was set to /mymap5.yaml and the default 3D sensor was kinect. also it is launch the move_base node to perform the navigation task.
- **roslaunch turtlebot_rviz_launchers view_navigation.launch**
- **rosrun add_markers basic_shapes**

6.2. Model a virtual object with markers in rviz

The final task of this project is to model a virtual object with markers in rviz. The virtual object is the one being picked and delivered by the robot, thus it should first appear in its pickup zone, and then in its drop off zone once the robot reaches it.

To accomplish this task was edited the template and connected with pick_objects node. The marker should initially be published at the pickup zone. After 5 seconds it should be hidden. Then after another 5 seconds it should appear at the drop off zone.

- Node name: add_markers
- Frame id: /map

- set pick and drop goals:

$$pick = [-6.0, -9.25];$$

$$drop = [-1.0, -0.0];$$

- To publish and hide the marker was added a conditional to know which operation perform
- Subscribe to /robot_position to connect with pick_objects node.
- Publish over visualization_marker the markers.

7. Home Service

7.1. write a home_service.sh file that will run all the nodes in this project

- **roslaunch turtlebot_gazebo turtlebot_world.launch**
- **roslaunch turtlebot_gazebo amcl_demo.launch**: The map file was set to /mymap5.yaml and the default 3D sensor was kinect. also it is launch the move_base node to perform the navigation task.
- **roslaunch turtlebot_rviz_launchers view_navigation.launch**
- **rosrun pick_objects pick_objects**
- **rosrun add_markers basic_shapes**

Finally, the entire process can be seen in the following link:

$$https://youtu.be/e1sITd4z_jg$$

8. Conclusion

The entire project was successfully done. The objectives were accomplish and the home service robot completed the pick and deliver task autonomously.

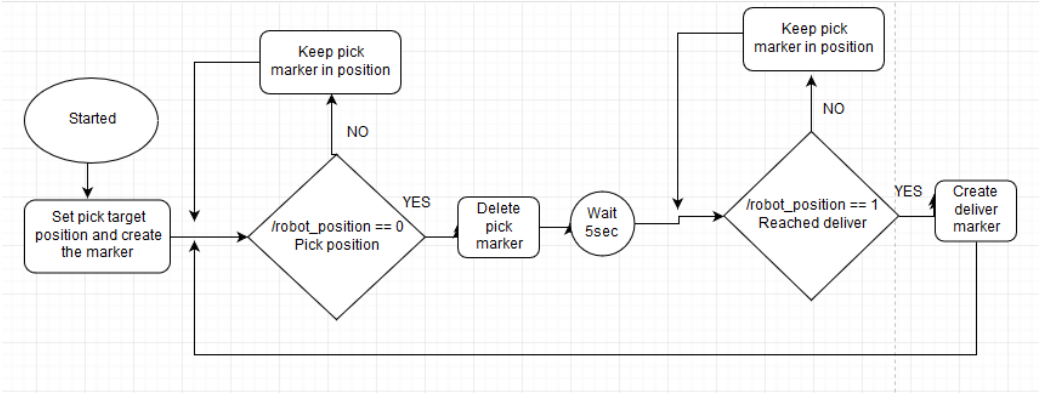

Figure 11: pick and deliver algorithm with markers

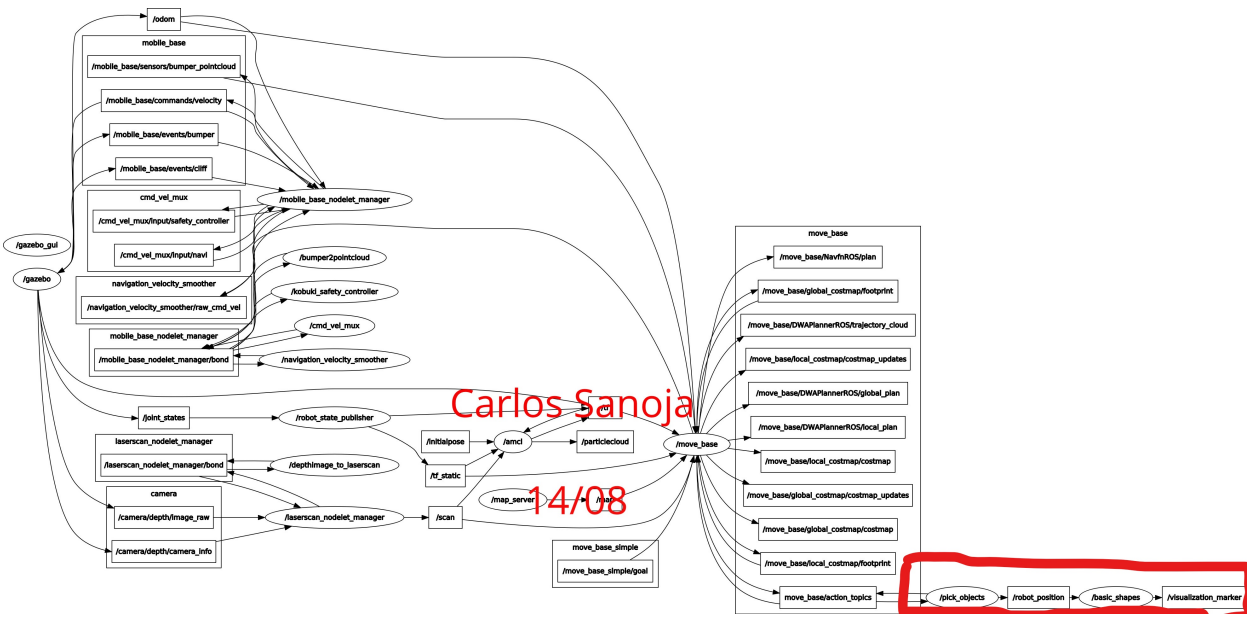

Figure 12: Nodes and Topics list