Multirobot System
Name: Iesus René Dávila Aguilar

Content

Description

The multirobot system developed at ROS Melodic has a stable and functional platform designed for autonomous or assisted exploration of environments that require navigation for different research tasks where the human hand does not intervene. All robots within the system are of the ROVER type where each mobile robot has autonomous navigation independently, which allows more special navigation for very large environments where a single robot may be of little use.

The system meets the feasibility of being able to easily incorporate more robots without the need to restructure all the developed code. In addition, it meets the detail that the system can support from 1 robot to N robots where the only limitation would be power. computational to incorporate N ROVERs.

General operations

Operation for 1 robot (without autonomous navigation)

1. Run the simulation: roslaunch main gazebo_house.launch

   There are more worlds where to execute them are as follows:
   - roslaunch main gazebo_empty.launch
   - roslaunch main gazebo_world.launch
   - roslaunch main gazebo_agriculture.launch

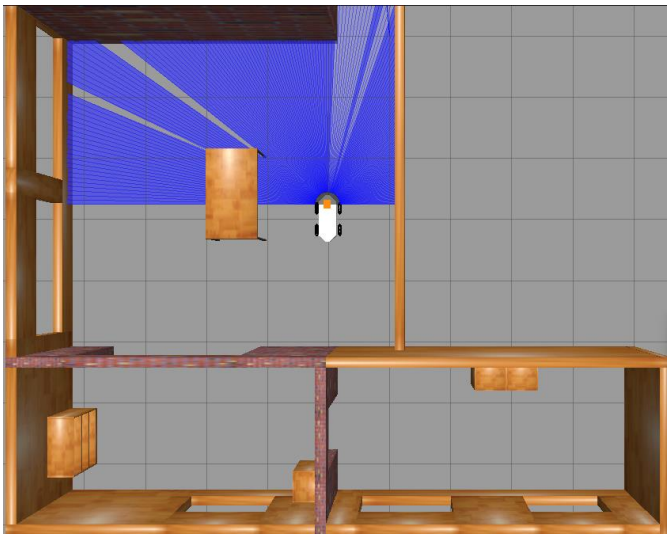

Figure 1 Simulation without autonomous navigation - 1 robot

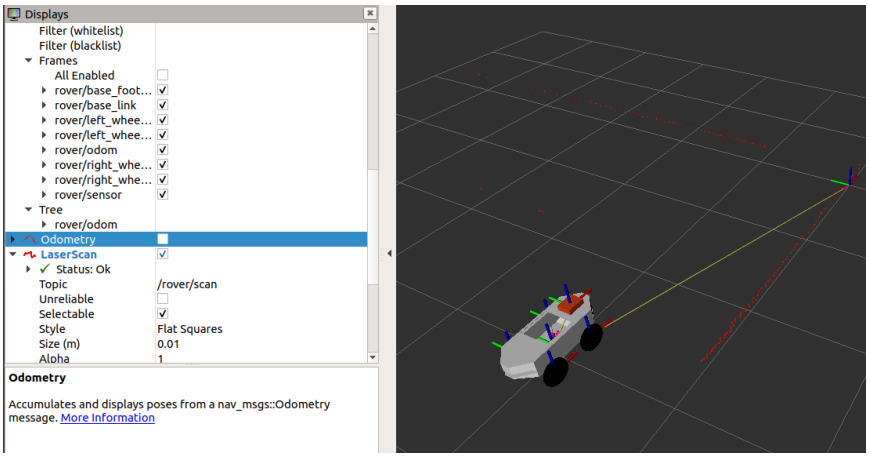

Figure 2 RViz without autonomous navigation - 1 robot

2. Execute teleoperation: rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=rover/cmd_vel

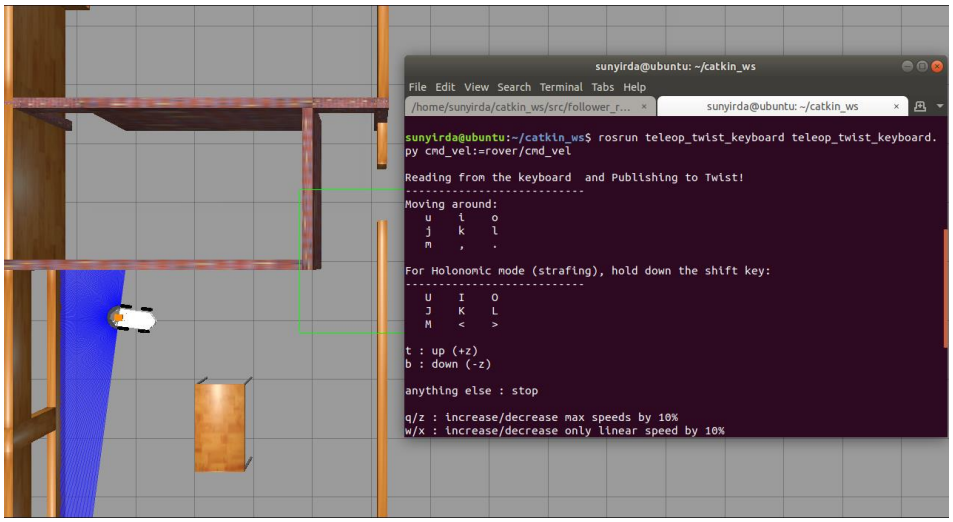

Figure 3 Teleoperación de ROVER

Operation for 1 robot (autonomous navigation)

1. Run the simulation: roslaunch main gazebo_house.launch navigation:=true

   There are more worlds where to execute them are as follows:
   - roslaunch main gazebo_empty.launch navigation:=true
   - roslaunch main gazebo_world.launch navigation:=true
   - roslaunch main gazebo_agriculture.launch navigation:=true

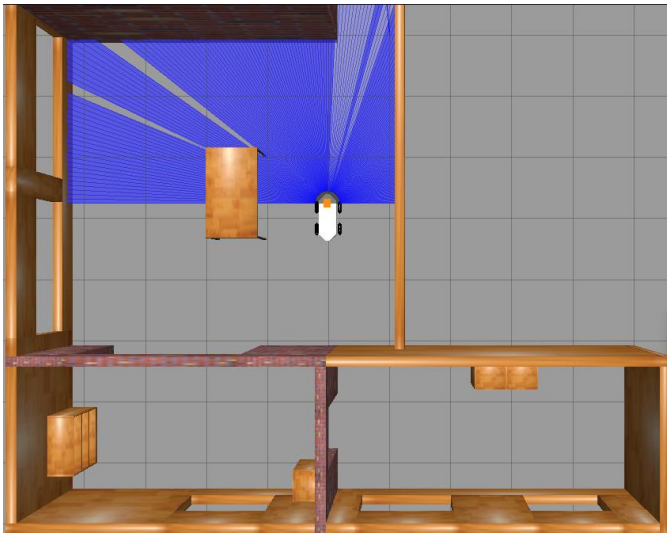

Figure 4 Simulation with autonomous navigation - 1 robot

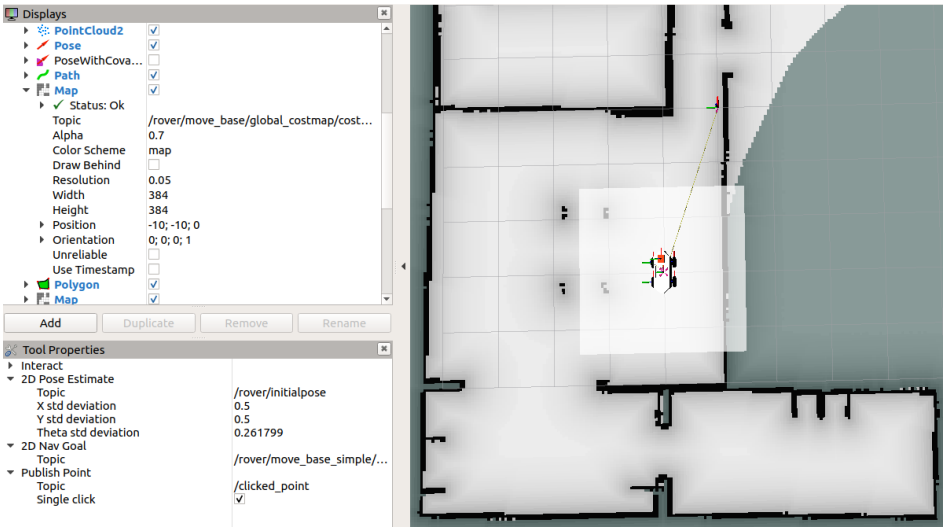

Figure 5 RViz with autonomous navigation - 1 robot

2. Send a goal point using RViz: Select the “2D Nav Goal” option (red arrow) and select the point where you want to move the robot (blue arrow).

   All autonomous navigation using maps requires establishing an initial position of movement, in this case the system when executed is responsible for defining said initial position.
   Se puede enviar el punto de meta mediante los actions que ofrece ROS, el uso de RViz es una vista más intuitiva para mostrar el correcto funcionamiento.

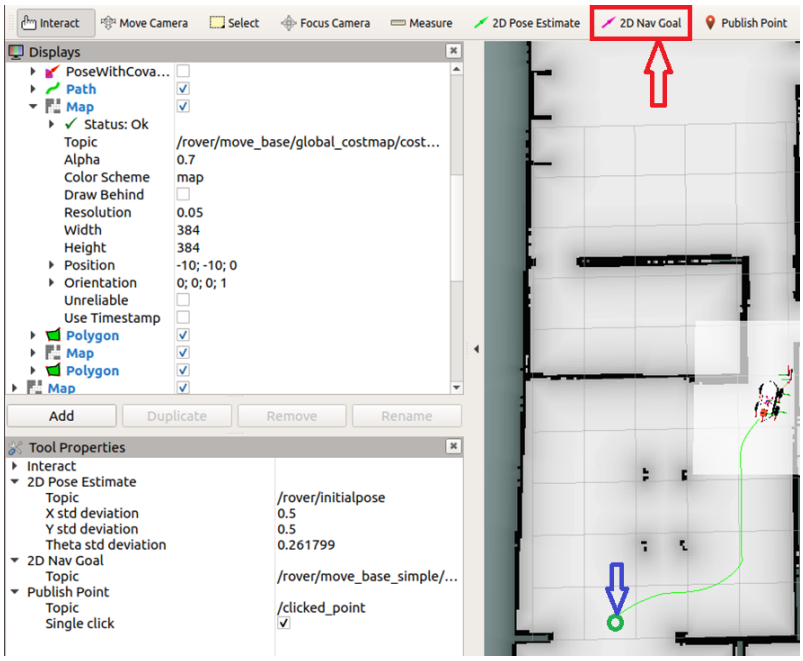

Figure 6 Send a goal point.

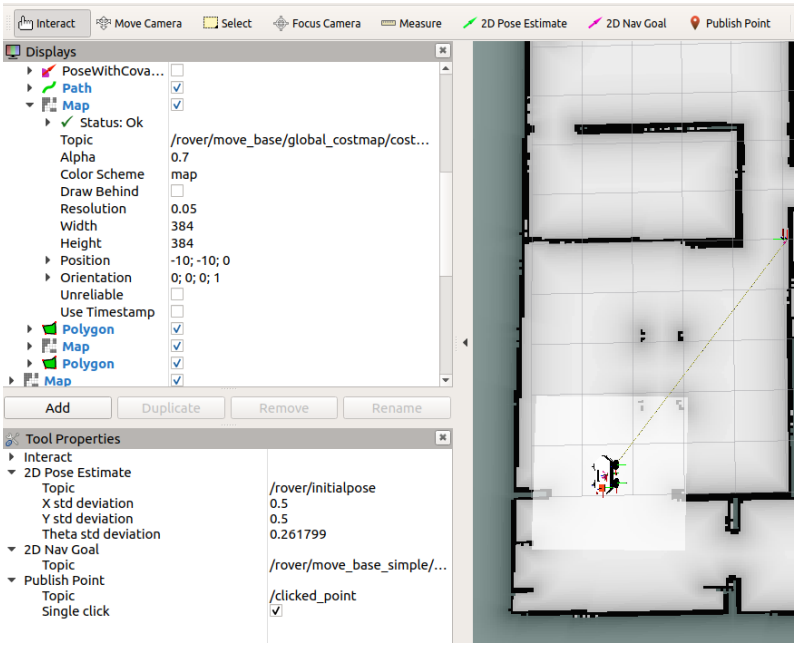

Figure 7 Robot at the finish point.

3. Execution of teleoperation (if necessary): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=rover/cmd_vel

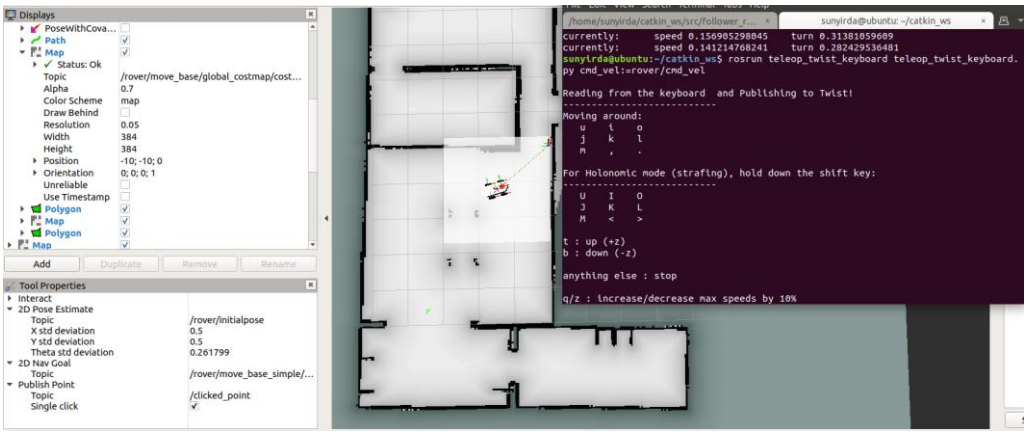

Figure 8 Teleoperation with activation of autonomous navigation.

Operation for multirobots (without autonomous navigation)

1. Run the simulation: roslaunch main multi_rover_house.launch

There are more worlds where to execute them are as follows:

- roslaunch main multi_rover_empty.launch
- roslaunch main multi_rover_world.launch

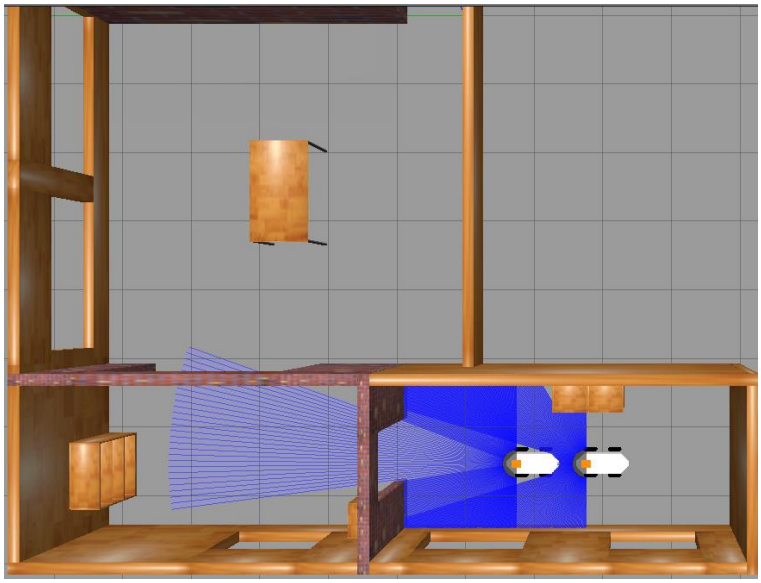

Figure 9 Multirobot simulation without autonomous navigation

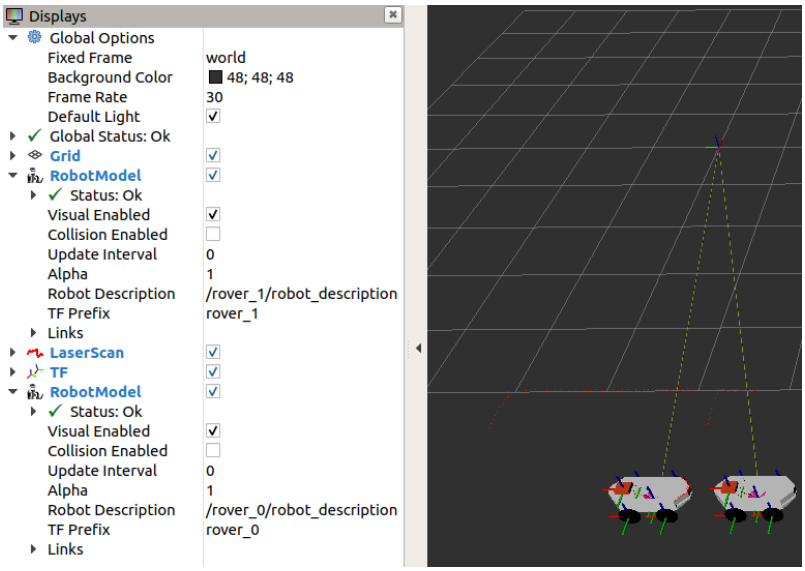

Figure 10 Multirobot RViz without autonomous navigation.

2. Execute the teleoperation (example with the robot named rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel

   In the multirobot system each topic depends on the robot, so if we have N robots then we will have N topics cmd_vel with the prefix of the name of its robot. In this case, by having two robots we will have the following topics:

   - /rover_0/cmd_vel
   - /rover_1/cmd_vel

   Important note: The same happens for the other topics such as robot_description, odom, etc.

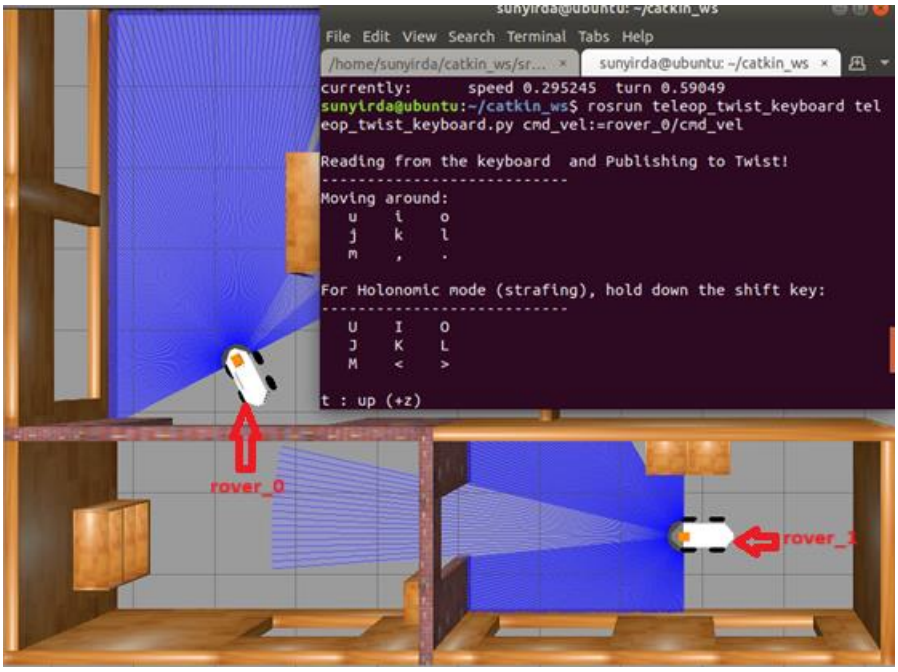

Figure 11 rover_0 teleoperation.

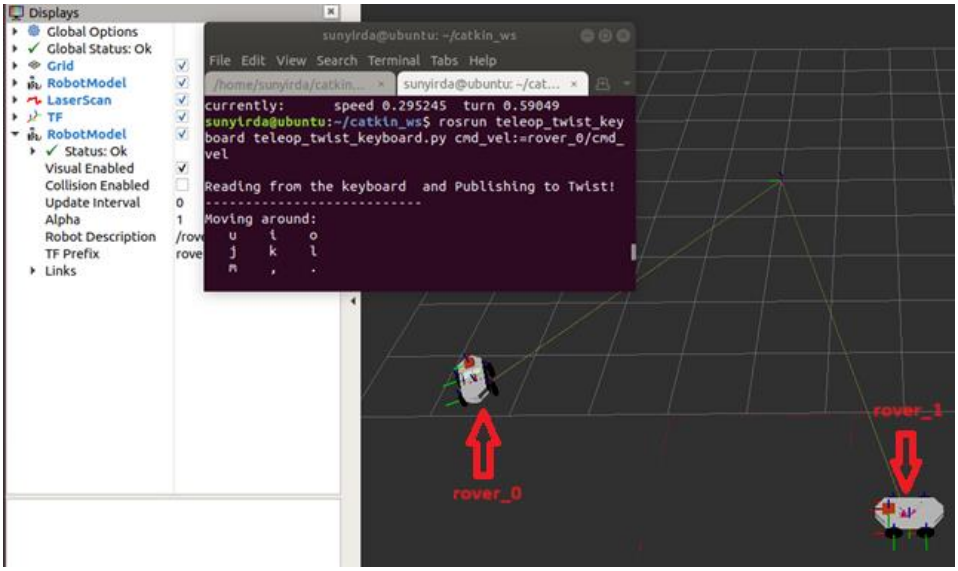

Figure 12 Rover_0 teleoperation - Visualization in RViz.

Operation for multirobots (autonomous navigation)

1. Run the simulation: roslaunch main multi_rover_house.launch navigation:=true

   There are more worlds where to execute them are as follows:
   - roslaunch main multi_rover empty.launch navigation:=true
   - roslaunch main multi_rover launch navigation:=true

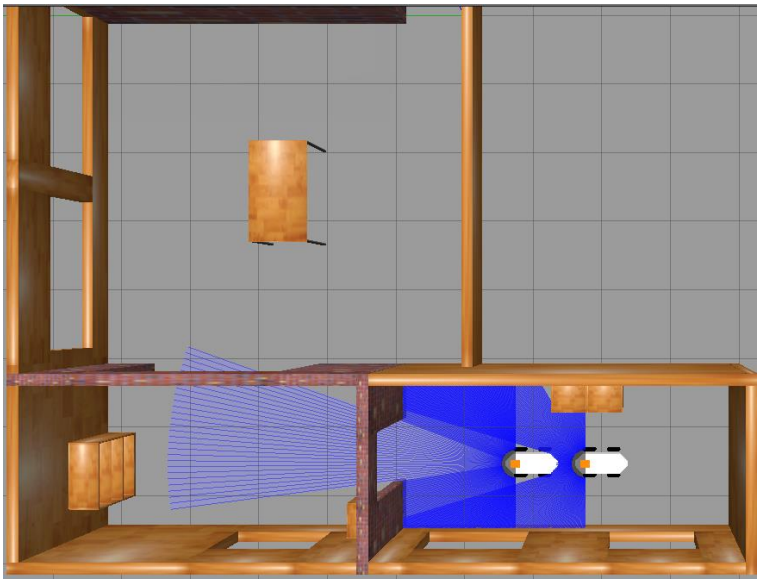

Figure 13 Multi-robot simulation with autonomous navigation.

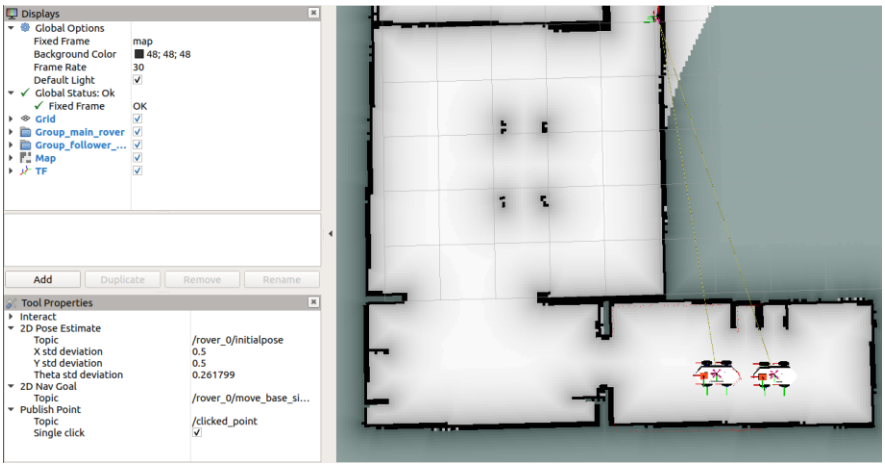

Figure 14 RViz multirobot with autonomous navigation.

2. Send a goal point using RViz (example with the robot called rover_0): Select the "2D Nav Goal" option (red arrow) and select the point where you want to move the robot (blue arrow).

   All autonomous navigation using maps requires establishing an initial position of movement, in this case the system when executed is responsible for defining said initial position.

   The goal point can be sent using the actions offered by ROS, using RViz is a more intuitive view to show correct operation.

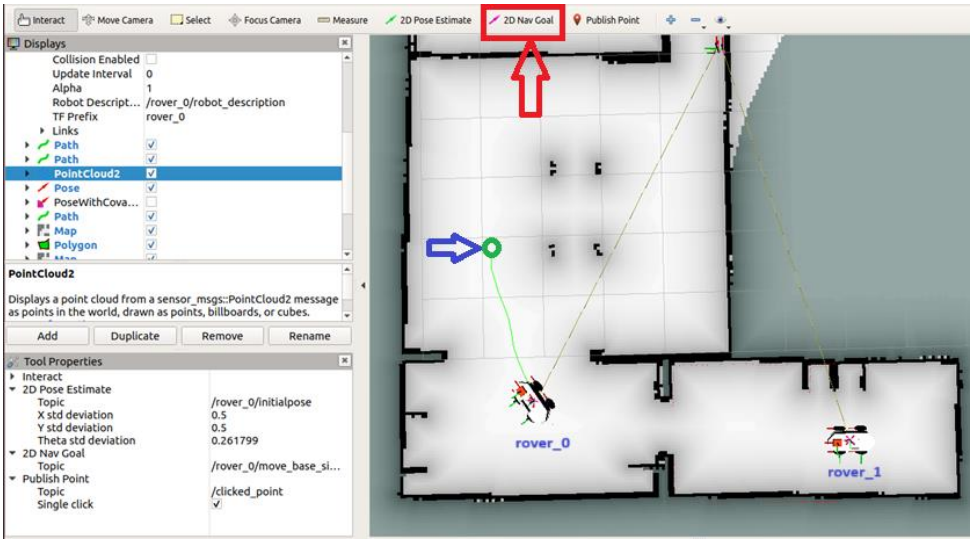

Figure 15 Send a goal point to rover_0.

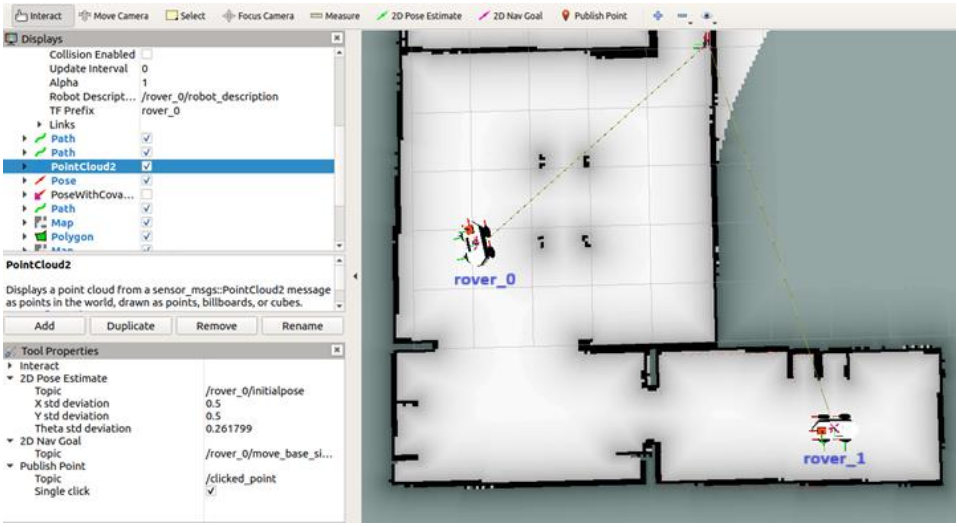

Figure 16 Robot rover_0 at the finish point.

3. Send a goal point using RViz to any robot.

   By default, RViz is set to use the "2D Nav Goal" button only for rover_0, but if it is required to pass a goal point to another robot using RViz, the following process must be carried out.

   a. Change the topic of the robot's initial position (example with the robot named rover_1): /NAME_ROBOT/initialpose

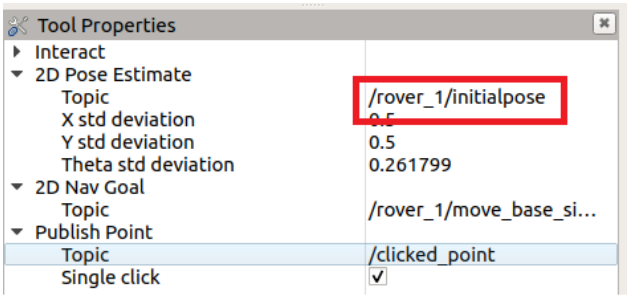

Figure 17 Update the 2D Pose Estimate topic.

   b. Change the topic of "2D Nav Goal" (example with the robot named rover_1): /NAME_ROBOT/ move_base_simple/goal

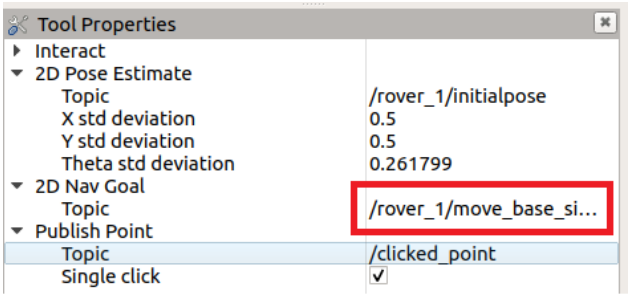

Figure 18 Update the 2D Nav Goal topic.

   c. Press the "2D Nav Goal" button as explained in step 2

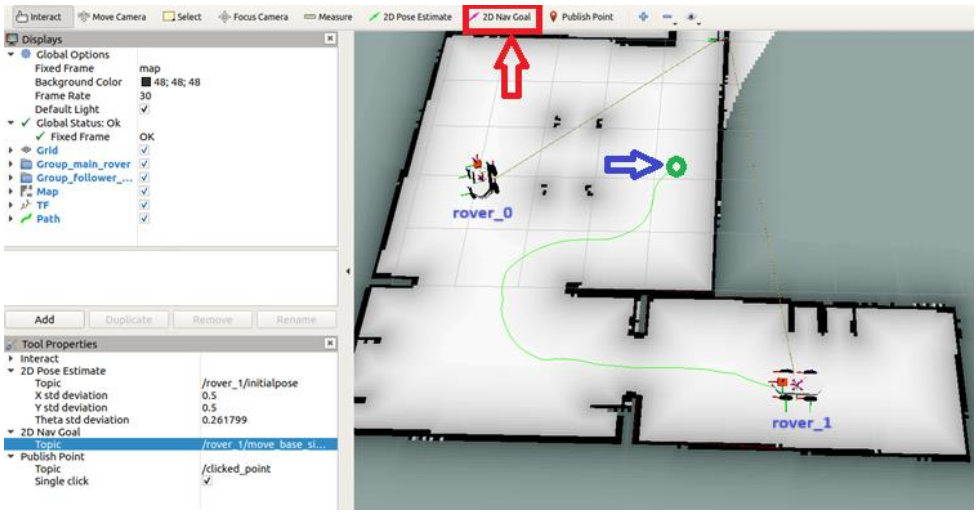

Figure 19 Send a goal point to rover_1.

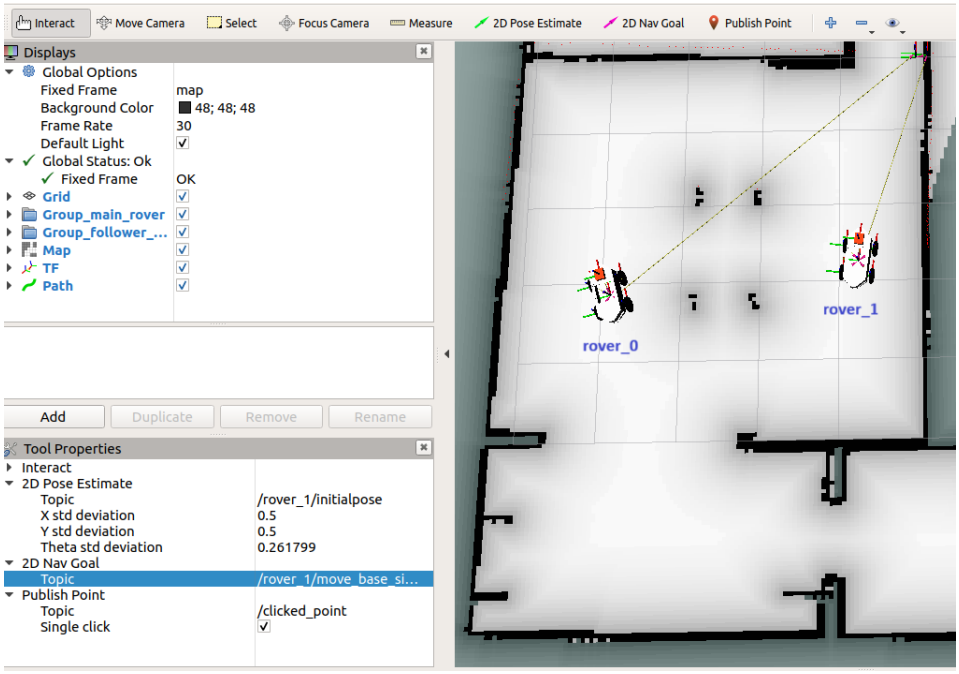

Figure 20 Robot rover_1 at the finish point.

Add a new world or more robots

Add new world

For practical purposes a world designed by other people will be used, this procedure is to add a new world, not to design a new world.

All these steps are within the package called “main”.

1. Inside the model’s folder, create a folder with the name of what our world model will be called.

Here the design will be placed in SDF, the model configuration file and the design folder.

2. Inside the worlds folder, import the file with the world extension that we have designed or imported from external people.

   Recommendation: The world model and the world as such are suggested to contain the same name.

3. If you require a world for a single robot, go to step 4, if you want a world for multiple robots, go to step 5.
4. In the launch folder create a new launch: gazebo_world.launch

   Copy and paste the launch of gazebo_house.launch and change the following lines:

Previous line	New line
<arg name="world_name" value="$(find main)/worlds/house.world"/>	<arg name="world_name" value="$(find main)/worlds/world.world"/>
<arg name="rover_x_pos" value="-3.0" />	Origin X position.
<arg name="rover_y_pos" value=" 1.0" />	Origin Y position.
<arg name="rover_z_pos" value=" 0.0" />	Origin Z position.
<arg name="rover_yaw" value=" 0.0" />	Origin YAW orientation.
<arg name="initial_pose_x" value="-3.0" />	Origin X position. (It is used for the initialpose topic of autonomous navigation)
<arg name="initial_pose_y" value=" 1.0" />	Origin Y position. (It is used for the initialpose topic of autonomous navigation)

Table 1 Launch for new world - 1 robot

5. In the launch folder create a new launch: multi_robot_world.launch

   Copy and paste the launch of gazebo_house.launch and change the following lines:

Previous line	New line
<arg name="world_name" value="$(find main)/worlds/house.world"/>	<arg name="world_name" value="$(find main)/worlds/world.world"/>

<arg name="first_rover" default="rover_0"/>	Name of the first ROVER.
<arg name="second_rover" default="rover_1"/>	Name of the second ROVER.
<arg name="first_rover_x_pos" default="-0.5"/>	Origin X position for the first ROVER.
<arg name="first_rover_y_pos" default=" 0.5"/>	Origin Y position for the first ROVER.
<arg name="first_rover_z_pos" default=" 0.0"/>	Origin Z position for the first ROVER.
<arg name="first_rover_yaw" default=" 0.0"/>	Origin YAW orientation for the first ROVER
<arg name="second_rover_x_pos" default="-1.5"/>	Origin X position for the second ROVER.
<arg name="second_rover_y_pos" default=" 0.5"/>	Origin Y position for the second ROVER.
<arg name="second_rover_z_pos" default=" 0.0"/>	Origin Z position for the second ROVER.
<arg name="second_rover_yaw" default=" 0.0"/>	Origin YAW orientation for the second ROVER.

Table 2 Launch for new world – multirobot

6. Go to your workspace and execute: catkin_make

Add a new ROVER

All these steps are within the package called “main”.

1. Go to the launch called spawn_multi_rover.launch and add the following lines:

Línea nueva	Explicación
<arg name="third_rover" default="rover_2"/>	Name of the new robot
<arg name="third_rover_x_pos" default="-6.0"/>	Origin X position.
<arg name="third_rover_y_pos" default=" 1.0"/>	Origin Y position.
<arg name="third_rover_z_pos" default=" 0.0"/>	Origin Z position.
<arg name="third_rover_yaw" default=" 0.0"/>	Origin YAW orientation.

Table 3 Multi ROVER spawn modification.

2. In the same launch file add a new group:

   These tags called <group> allow for a better structure when working with multirobots and a correct definition of transformations, topics, services, etc. is required.

```
<group ns = "$(arg third_rover)">
      <include file="$(find main)/launch/spawn_rover.launch">
         <arg name="name_rover" value="$(arg third_rover)" />
         <arg name="rover_x_pos" value="$(arg third_rover_x_pos)" />
         <arg name="rover_y_pos" value="$(arg third_rover_y_pos)" />
         <arg name="rover_z_pos" value="$(arg third_rover_z_pos)" />
         <arg name="rover_yaw"   value="$(arg third_rover_yaw)" />
      </include>
```

```
      <param if="$(arg navigation)" name="$(arg
third_rover_x_pos)/amcl/initial_pose_x" value="$(arg third_rover_x_pos)" />g
      <param if="$(arg navigation)" name="$(arg
third_rover_x_pos)/amcl/initial_pose_y" value="$(arg third_rover_y_pos)" />
      <include if="$(arg navigation)" file="$(find
lidar_navigation)/launch/lidar_navigation.launch">
         <arg name="initial_pose_x" value="$(arg third_rover_x_pos)" />
         <arg name="initial_pose_y" value="$(arg third_rover_y_pos)" />
         <arg name="initial_pose_a" value="$(arg third_rover_yaw)"/>
         <arg name="robot_namespace" value="$(arg third_rover)"/>
         <arg name="map_file" value="$(arg map_file)"/>
      </include>

      <node unless="$(arg navigation)" pkg="tf" type="static_transform_publisher"
name="static_transform_publisher" args="0 0 0 0 0 0 world $(arg third_rover)/odom 1"
/>
   </group>
```

3. Create a launch similar to multi_rover_house.launch, but instead of placing 2 robots, do it to place 3 robots.
4. Go to your workspace and execute: catkin_make