Multirobot System
Name: Iesus René Dávila Aguilar

Contenido

Descripción

El sistema multirobot desarrollado en ROS Melodic posee una plataforma estable y funcional diseñada para la exploración autónoma o asistida de entornos que se requieran navegar para diferentes tareas de investigación donde no intervenga la mano humana. Todos los robots dentro sistema son de tipo ROVER donde cada robot móvil posee una navegación autónoma de manera independiente por lo que permite una navegación mas especial para entornos muy grandes donde un solo robot puede ser de poca utilidad.

El sistema cumple la viabilidad de poder incorporar de manera muy sencilla más robots sin la necesidad de reestructurar todo el código desarrollado, además, se cumple con el detalle de que el sistema puede soportar desde 1 robot hasta N robots donde la única limitante sería el poder computacional para incorporar N ROVERs.

Funcionamientos generales

Funcionamiento para 1 robot (sin navegación autónoma)

1. Ejecutar la simulación: roslaunch main gazebo_house.launch

   Hay más mundos donde para ejecutarlos son de la siguiente manera:
   - roslaunch main gazebo_empty.launch
   - roslaunch main gazebo_world.launch
   - roslaunch main gazebo_agriculture.launch

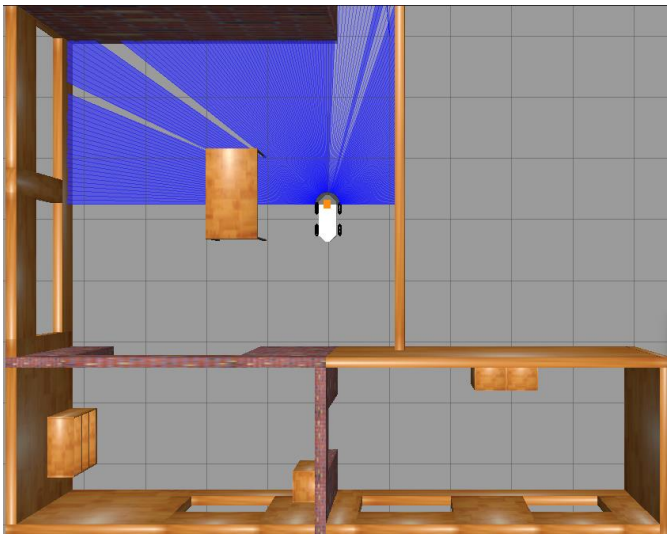

Figure 1 Simulación sin navegación autónoma - 1 robot

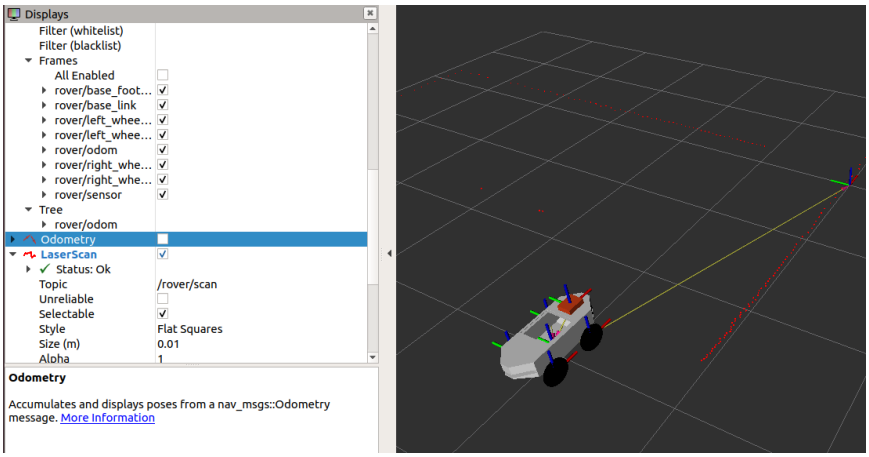

Figure 2 RViz sin navegación autónoma - 1 robot

2. Ejecutar la teleoperación: rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=rover/cmd_vel

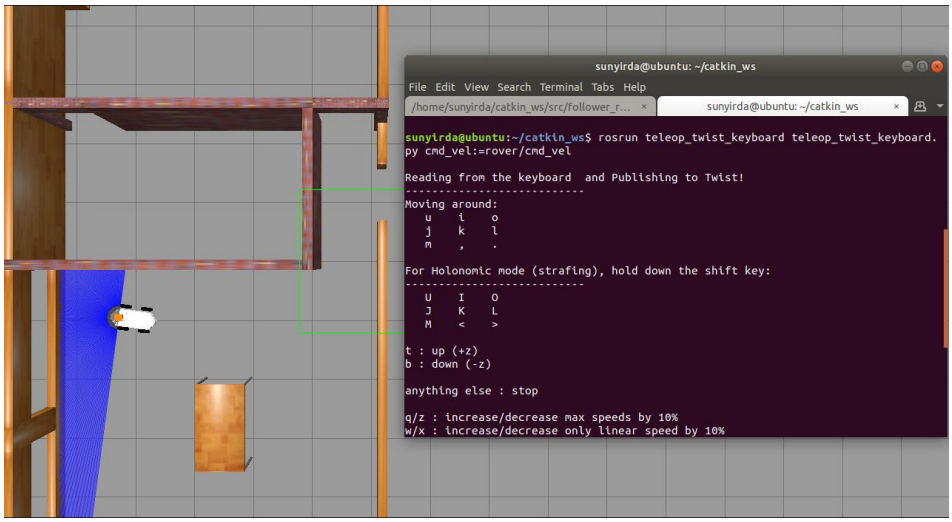

Figure 3 Teleoperación de ROVER

Multirobot System
Name: Iesus René Dávila Aguilar

Funcionamiento para 1 robot (navegación autónoma)

1. Ejecutar la simulación: roslaunch main gazebo_house.launch navigation:=true

   Hay más mundos donde para ejecutarlos son de la siguiente manera:
   - roslaunch main gazebo_empty.launch navigation:=true
   - roslaunch main gazebo_world.launch navigation:=true
   - roslaunch main gazebo_agriculture.launch navigation:=true

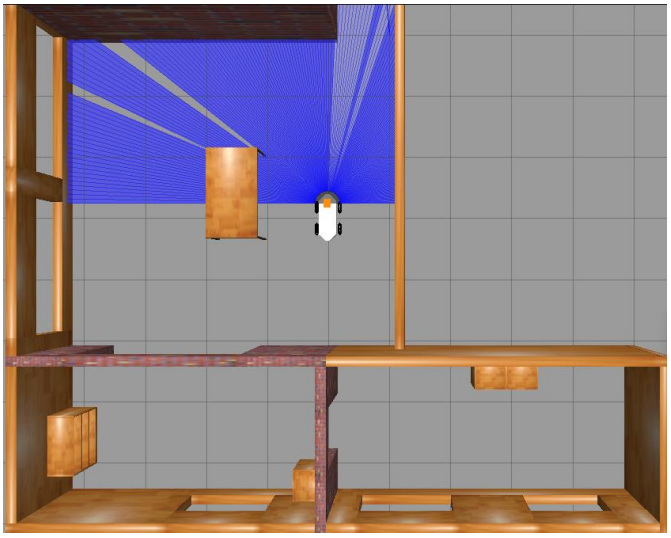

Figure 4 Simulación con navegación autónoma - 1 robot

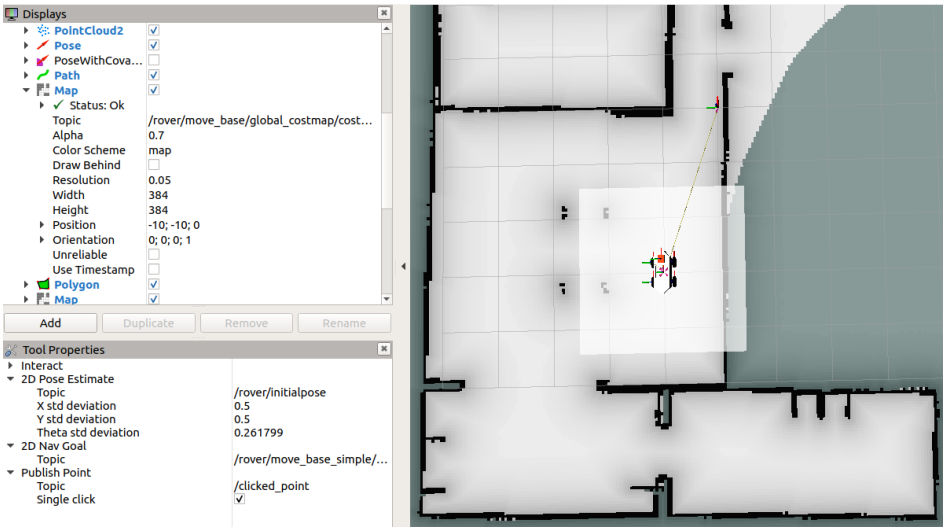

Figure 5 RViz con navegación autónoma - 1 robot

2. Enviar un punto de meta mediante RViz: Seleccionar la opción “2D Nav Goal” (flecha roja) y seleccionar el punto hacia donde se desee mover el robot (flecha azul).

   Toda navegación autónoma mediante mapas requiere establecer una posicion inicial de movimiento, en este caso el sistema al ejecutarse se encarga de definir dicha posicion inicial.

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.

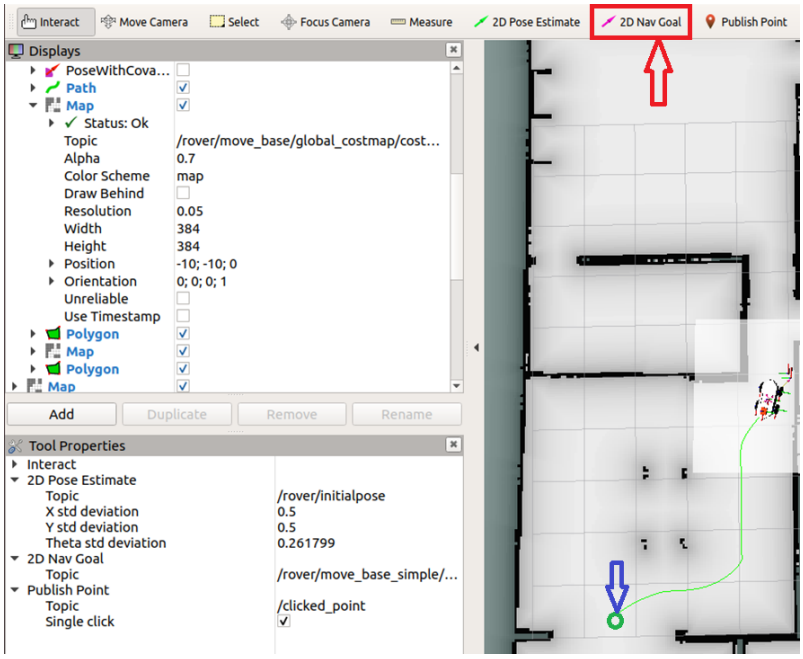

Figure 6 Enviar un punto de meta.

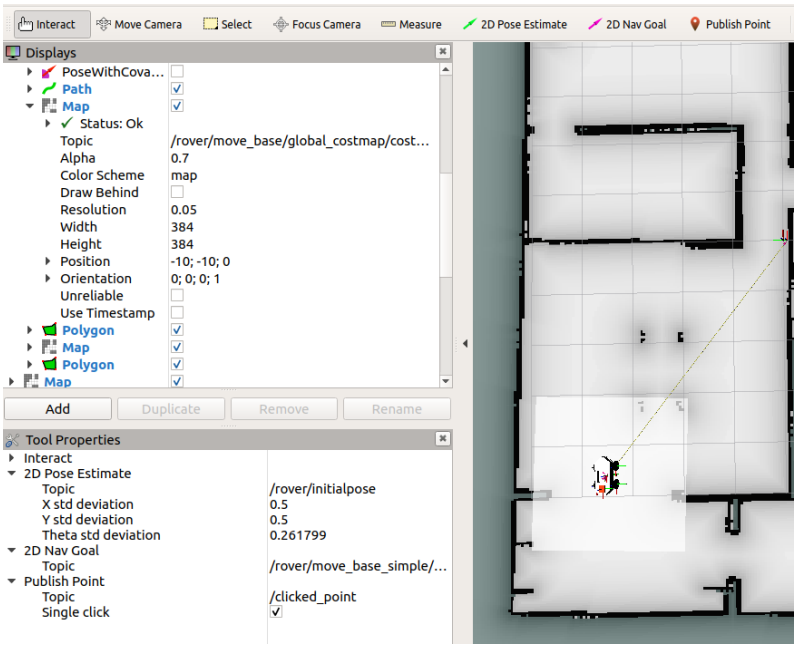

Figure 7 Robot en el punto de meta.

3. Ejecución de teleoperación (en caso de necesitar): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=rover/cmd_vel

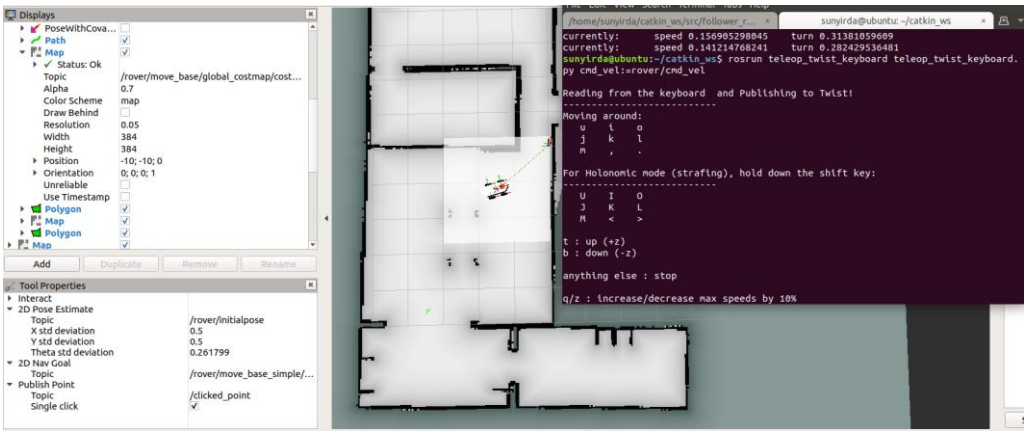

Figure 8 Teleoperación con activación de navegación autonoma.

Funcionamiento para multirobots (sin navegación autónoma)

1. Ejecutar la simulación: roslaunch main multi_rover_house.launch

   Hay más mundos donde para ejecutarlos son de la siguiente manera:
   - roslaunch main multi_rover_empty.launch
   - roslaunch main multi_rover_world.launch

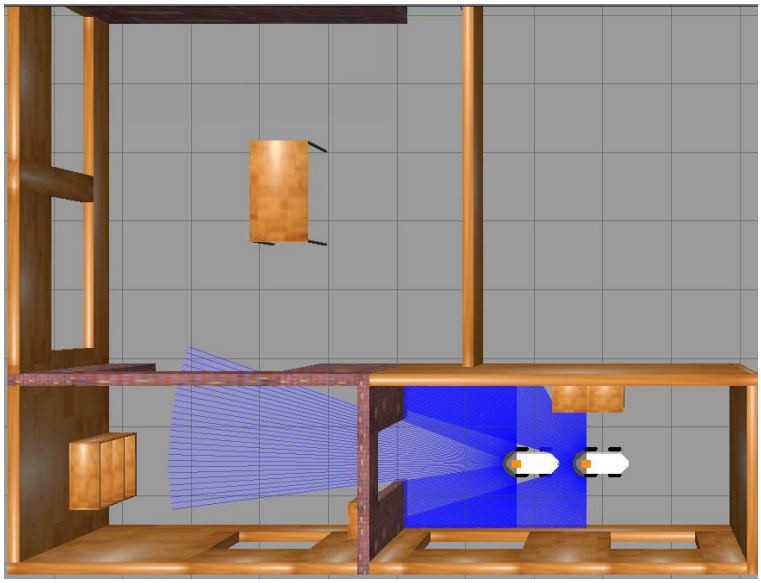

Figure 9 Simulación multirobot sin navegación autónoma

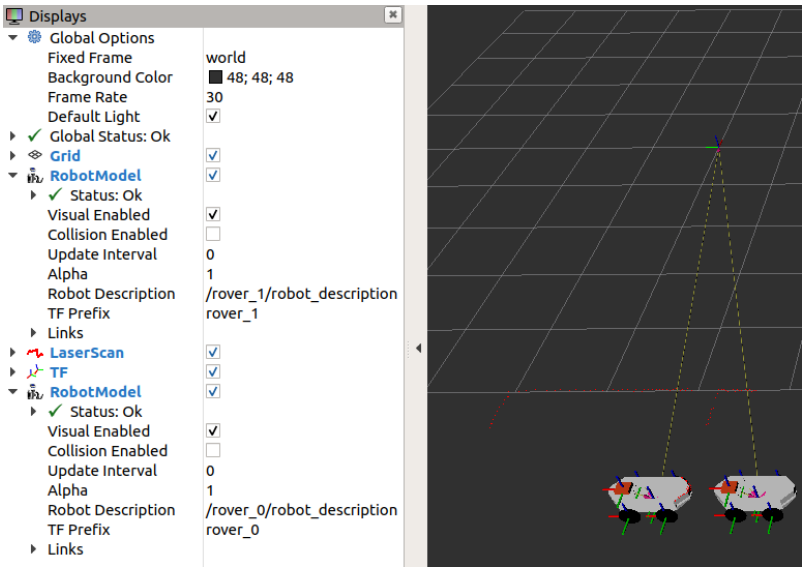

Figure 10 RViz multirobot sin navegación autónoma

2. Ejecutar la teleoperación (ejemplo con el robot llamado rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel

   En el sistema multirobot cada topic depende del robot, por lo que si tenemos N robots entonces tendremos N topics cmd_vel con el prefijo del nombre de su robot. En este caso al tener dos robots tendremos los siguientes topics:

   - /rover_0/cmd_vel
   - /rover_1/cmd_vel

   Nota importante: Lo mismo sucede para los otros topics como para el robot_description, odom, etc.

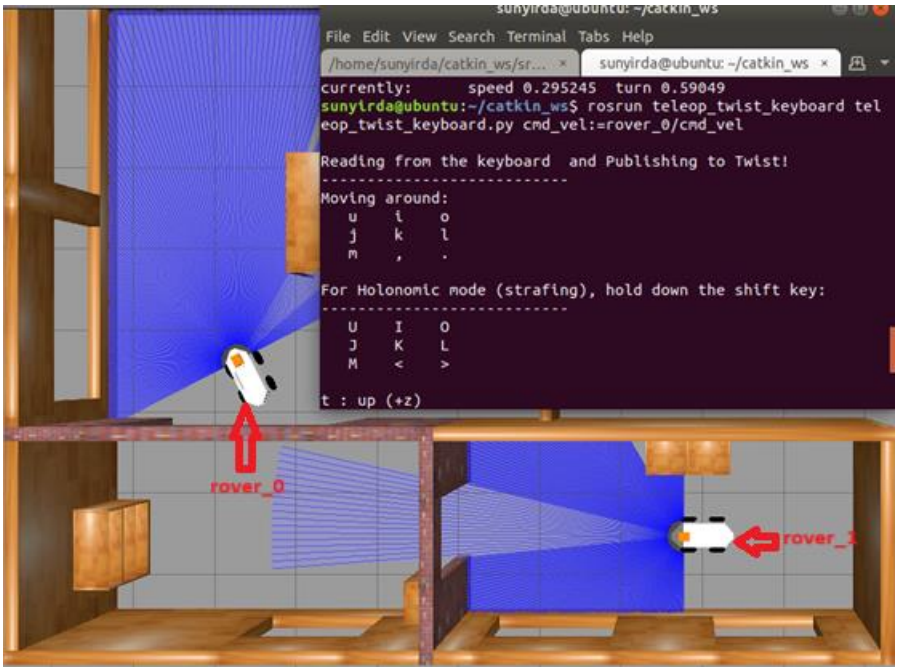

Figure 11 Teleoperación de rover_0

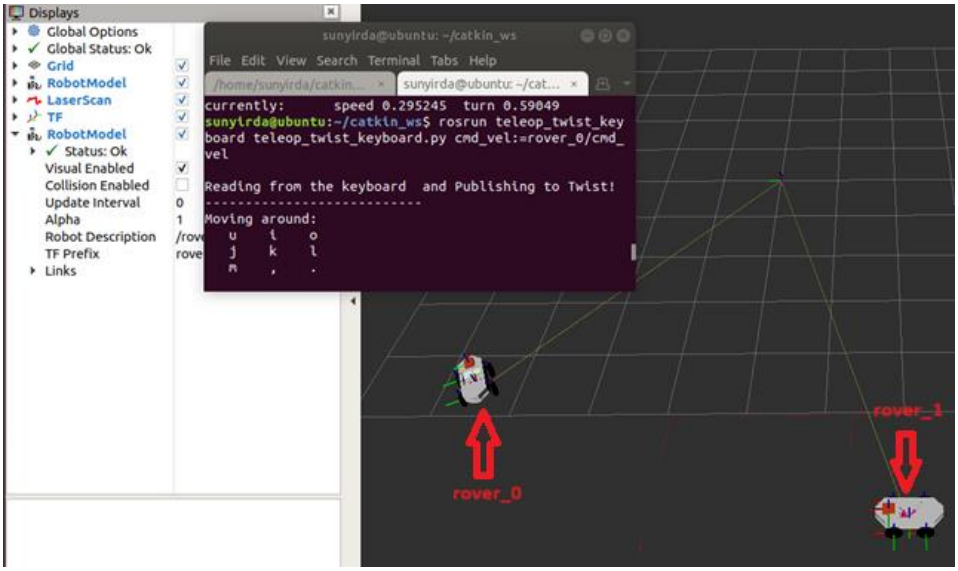

Figure 12 Teleoperación de rover_0 - Visualizacion en RViz

Funcionamiento para multirobots (navegación autónoma)

1. Ejecutar la simulación: roslaunch main multi_rover_house.launch navigation:=true

   Hay más mundos donde para ejecutarlos son de la siguiente manera:
   - roslaunch main multi_rover empty.launch navigation:=true
   - roslaunch main multi_rover launch navigation:=true

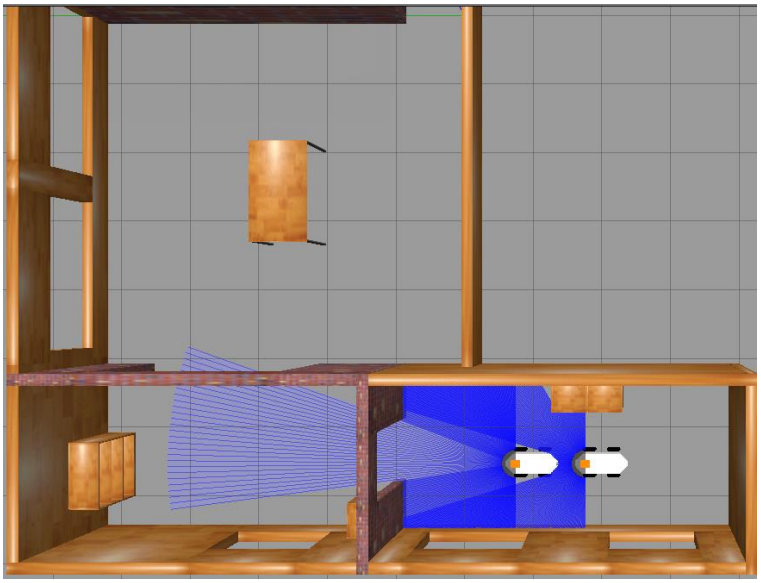

Figure 13 Simulación multirobot con navegación autónoma

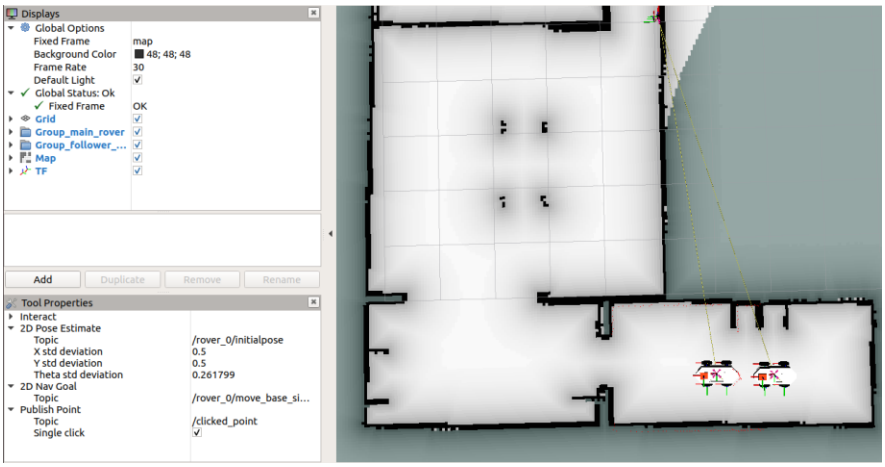

Figure 14 RViz multirobot con navegación autónoma

2. Enviar un punto de meta mediante RViz (ejemplo con el robot llamado rover_0): Seleccionar la opción "2D Nav Goal" (flecha roja) y seleccionar el punto hacia donde se desee mover el robot (flecha azul).

   Toda navegación autónoma mediante mapas requiere establecer una posicion inicial de movimiento, en este caso el sistema al ejecutarse se encarga de definir dicha posicion inicial.

   Se puede enviar el punto de meta mediante los actions que ofrece ROS, el uso de RViz es una vista mas intuitiva para mostrar el correcto funcionamiento.

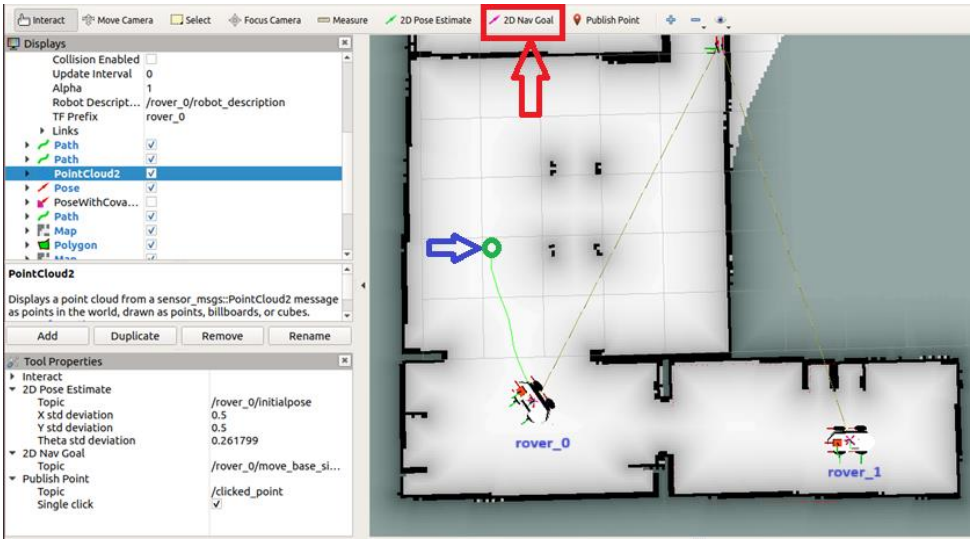

Figure 15 Enviar un punto de meta a rover_0.

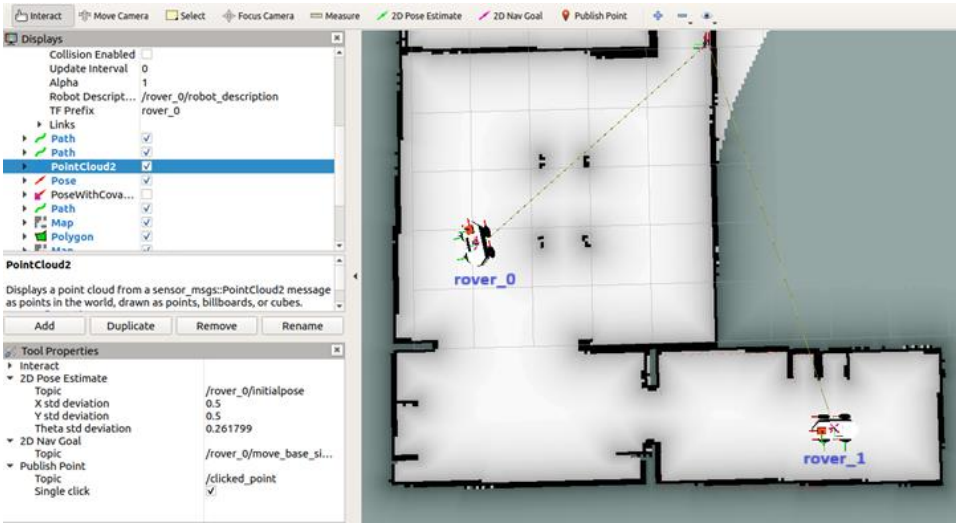

Figure 16 Robot rover_0 en el punto de meta.

3. Enviar un punto de meta mediante RViz a cualquier robot.

   Por defecto se tiene fijado en RViz para usar el boton de “2D Nav Goal” solo para rover_0, pero en caso de que se requiera pasar un punto de meta a otro robot haciendo uso de RViz se tienen que realizar el siguiente proceso.

   a. Cambiar el topic de la posicion inicial del robot (ejemplo con el robot llamado rover_1): /NAME_ROBOT/initialpose

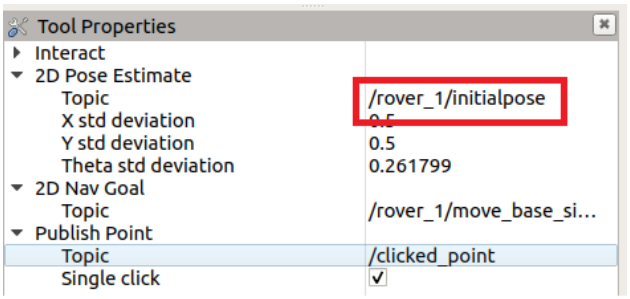

Figure 17 Actualizar el topic de 2D Pose Estimate.

   b. Cambiar el topic de “2D Nav Goal” (ejemplo con el robot llamado rover_1): /NAME_ROBOT/ move_base_simple/goal

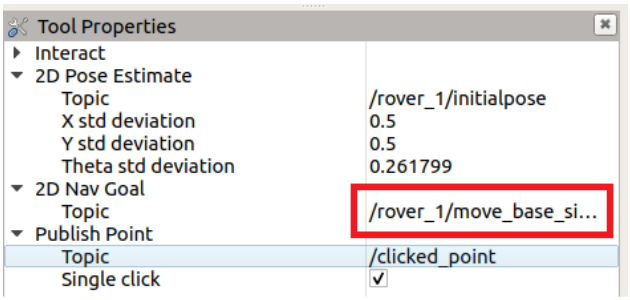

Figure 18 Actualizar el topic de 2D Nav Goal.

c. Pulsar el boton de “2D Nav Goal” tal como se explica en el paso 2

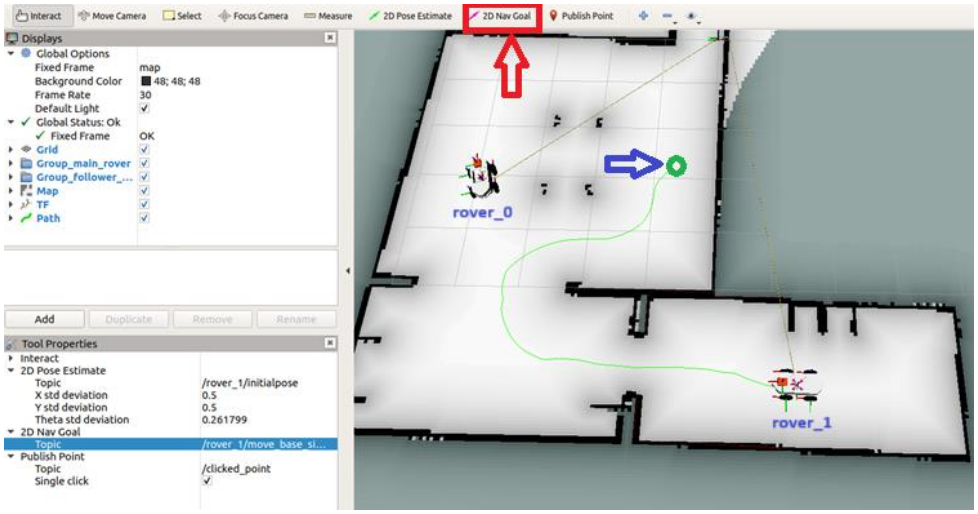

Figure 19 Enviar un punto de meta a rover_1.

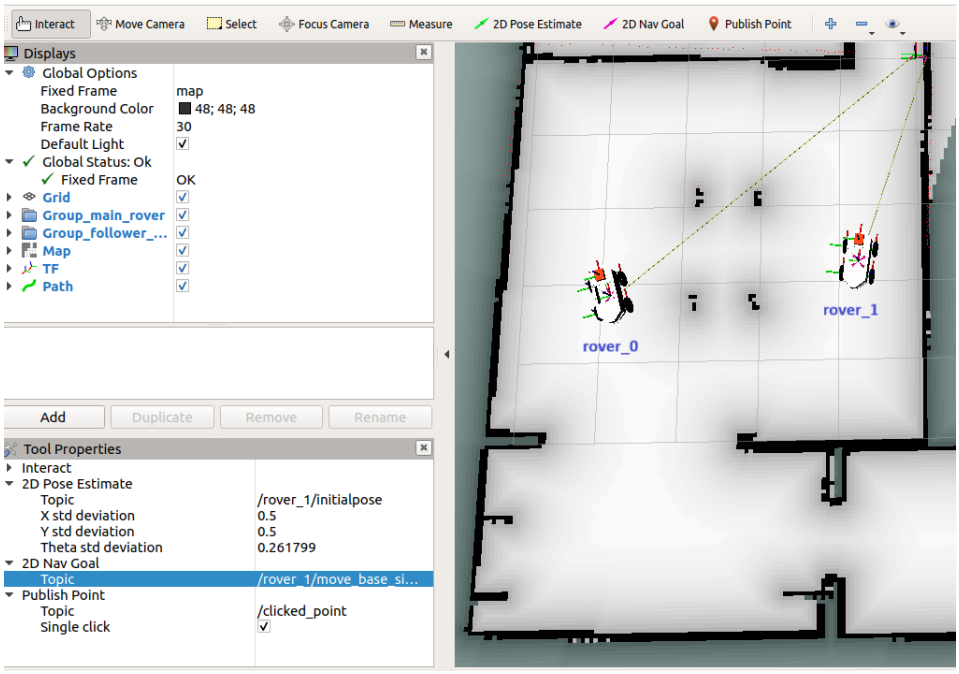

Figure 20 Robot rover_1 en el punto de meta.

Añadir un mundo nuevo o más robots

Añadir nuevo mundo

Para fines prácticos se utilizará un mundo diseñado por otras personas, este procedimiento es para añadir un nuevo mundo, no para diseñar un nuevo mundo.

Todos estos pasos son dentro del paquete llamado “main”.

1. Dentro de la carpeta models crear una carpeta con el nombre de cómo se llamará nuestro modelo del mundo.

Aquí se colocará el diseño en SDF, el archivo de configuración del modelo y la carpeta de diseño.

2. Dentro de la carpeta worlds importar el archivo con extensión world que hemos diseñado o importado de personas externas.

   Recomendación: El modelo del mundo y el mundo como tal se sugiere que contengan el mismo nombre.

3. En caso de que requiera un mundo para un robot simple dirigirse al paso 4, si desea un mundo para múltiples robots dirigirse al paso 5.
4. En la carpeta launch crear un nuevo launch: gazebo_world.launch

   Copiar y pegar el launch de gazebo_house.launch y cambiar las siguientes líneas:

Línea anterior	Línea nueva
<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" />	Posición X de origen.
<arg name="rover_y_pos" value=" 1.0" />	Posición Y de origen.
<arg name="rover_z_pos" value=" 0.0" />	Posición Z de origen.
<arg name="rover_yaw" value=" 0.0" />	Orientación YAW de origen.
<arg name="initial_pose_x" value="-3.0" />	Posición X de origen. (Sirve para el topic de initialpose de la navegación autonoma)
<arg name="initial_pose_y" value=" 1.0" />	Posición Y de origen. (Sirve para el topic de initialpose de la navegación autonoma)

Tabla 1 Launch para mundo nuevo - 1 robot

5. En la carpeta launch crear un nuevo launch: multi_robot_world.launch

   Copiar y pegar el launch de gazebo_house.launch y cambiar las siguientes líneas:

Línea anterior	Línea nueva
<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"/>	Nombre del primer ROVER.
<arg name="second_rover" default="rover_1"/>	Nombre del segundo ROVER.
<arg name="first_rover_x_pos" default="-0.5"/>	Posición X de origen para el primer ROVER.
<arg name="first_rover_y_pos" default=" 0.5"/>	Posición Y de origen para el primer ROVER.
<arg name="first_rover_z_pos" default=" 0.0"/>	Posición Z de origen para el primer ROVER.
<arg name="first_rover_yaw" default=" 0.0"/>	Orientación YAW de origen para el primer ROVER.
<arg name="second_rover_x_pos" default="-1.5"/>	Posición X de origen para el segundo ROVER.
<arg name="second_rover_y_pos" default=" 0.5"/>	Posición Y de origen para el segundo ROVER.
<arg name="second_rover_z_pos" default=" 0.0"/>	Posición Z de origen para el segundo ROVER.
<arg name="second_rover_yaw" default=" 0.0"/>	Orientación YAW de origen para el segundo ROVER.

Tabla 2 Launch para mundo nuevo – multirobot

6. Dirigirse a su workspace y ejecutar: catkin_make

Añadir un nuevo ROVER

Todos estos pasos son dentro del paquete llamado “main”.

1. Dirigirse al launch llamado spawn_multi_rover.launch y agregar las siguientes líneas:

Línea nueva	Explicación
<arg name="third_rover" default="rover_2"/>	Nombre del nuevo robot
<arg name="third_rover_x_pos" default="-6.0"/>	Posición X de origen.
<arg name="third_rover_y_pos" default=" 1.0"/>	Posición Y de origen.
<arg name="third_rover_z_pos" default=" 0.0"/>	Posición Z de origen.
<arg name="third_rover_yaw" default=" 0.0"/>	Orientación YAW de origen.

Tabla 3 Modificación de spawn multi ROVER.

2. En el mismo archivo archivo launch agregar un grupo nuevo:

   Están etiquetas llamadas <group> permiten llevar una mejor estructura cuando se trabajan con multirobots y se requiere una correcta definición de las transformaciones, topics, services, etc.

```
<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. Crear un launch parecido al multi_rover_house.launch, pero en lugar de colocar 2 robots hacerlo para colocar 3 robots.
4. Dirigirse a su workspace y ejecutar: catkin_make