Course of

Robot Programming with ROS 2

Day 3

4. Hands On!!!

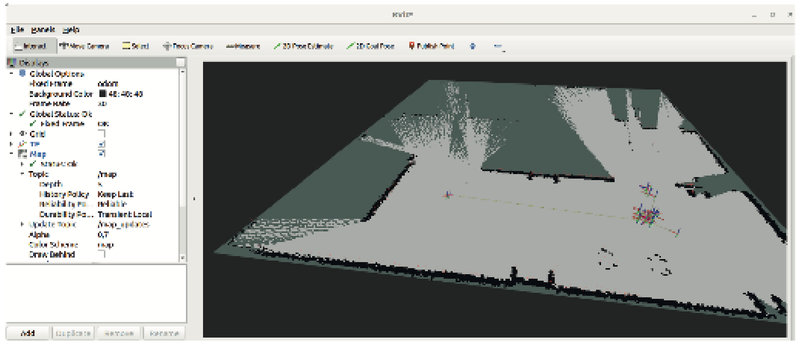
ikerlan

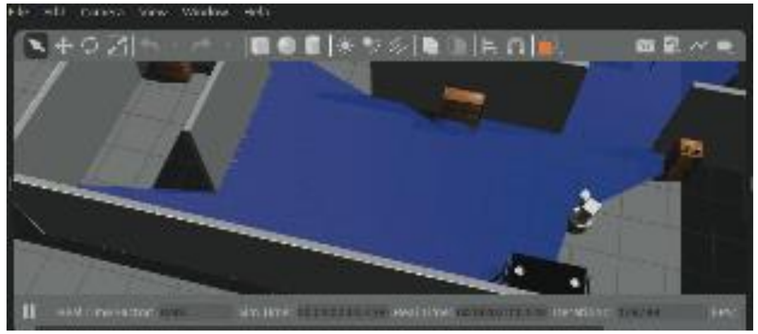
Universidad Rey Juan Carlos

Intelligent Robotics Lab

Create a map of the environment using SLAM

Make the robot navigate using the map





1 Creating a map using SLAM - Simulation

1. Launch Tiago simulator

```
$ ros2 launch tiago_simulator simulation.launch.py
```

2. Launch rviz with sim_time true to visualize the mapping process

```
$ rviz2 --ros-args -p use_sim_time:=true
```

1 Creating a map using SLAM - Simulation

3. Create a folder params, and then create the file mapper_params_online_async.yaml

params/mapper_params_online_async.yaml

```
# ROS Parameters
odom_frame: odom
map_frame: map
base_frame: base_footprint
scan_topic: /scan_raw
mode: mapping #localization
```

4. Launch the SLAM node. It will publish in /map the map while it is building it

```
$ ros2 launch slam_toolbox online_async_launch.py slam_params_file:=[Full path
to ws]/ros2_ws/src/ikerlan/tiago/params/mapper_params_online_async.yaml
use_sim_time:=true
```

Intelligent Robotics Lab

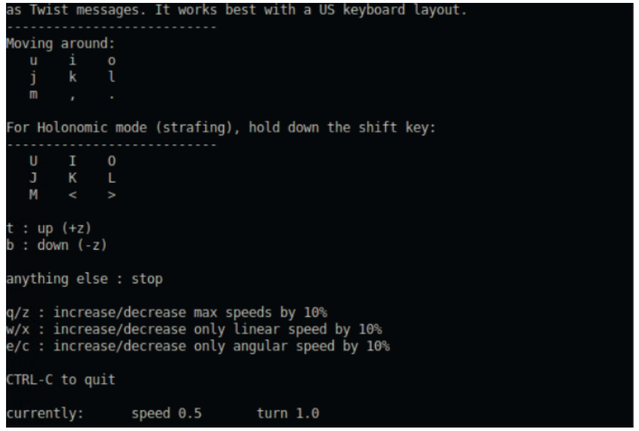
1 Creating a map using SLAM - Simulation

5. Launch the map server. This node will subscribe to /map, and it will save the map to disk when requested

```
$ ros2 launch nav2_map_server map_saver_server.launch.py
```

6. Teleoperate the robot in order to explore the environment. Open Rviz2 and check how the map is built

```
$ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args --remap
/cmd_vel:=/key vel -p use_sim_time:=true
```



Intelligent Robotics Lab

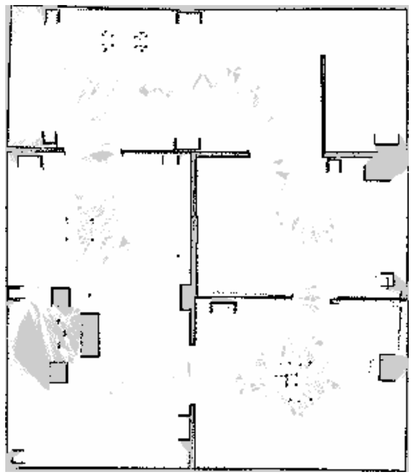
1 Creating a map using SLAM - Simulation

7. When the map is completed, ask the map server to sabe the map to disk

```
$ ros2 run nav2 map_server map_saver_cli --ros-args -p use_sim_time:=true
```

At this point, two files have been created:

1. PGM image file: It can be modified if you need to do any fix

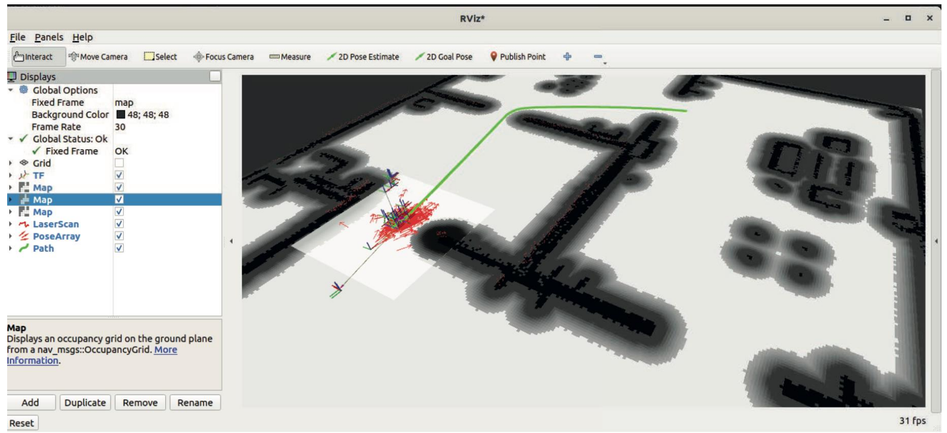


2. YAML file: contains enough information to interpret the image as a map.

2 Make the robot navigate with the map – Simulation

1. Open Rviz2 and display:

- Frame: map
- TF: To display the robot
- Map: Display the topic map
- LaserScan: To see how it matches with obstacels
- It is interesting to display the AMCL particles. Each one is a hypothesis about the robot´s position



2. Use the “2D Goal Pose” button to command a goal position to the robot

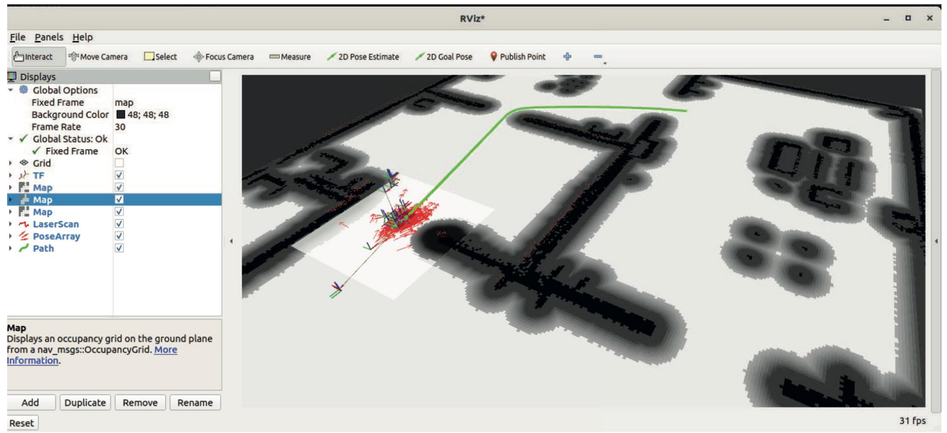
2 Make the robot navigate with the map – Simulation

3. Launch Tiago simulator

```
$ ros2 launch tiago_simulator simulation.launch.py
```

4. Launch navigation

```
$ ros2 launch tiago_simulator navigation.launch.py
```



5. Use the “2D Goal Pose” button to command a goal position to the robot

1 Creating a map using SLAM – Real robot

1. Launch Robot driver

```
$ ros2 launch kobuki kobuki.launch.py
```

2. Launch rviz with sim_time false to visualize the mapping process

```
$ rviz2 --ros-args -p use_sim_time:=false
```

1 Creating a map using SLAM - Real Robot

3. Create a folder params, and then create the file mapper_params_online_async.yaml

params/mapper_params_online_async.yaml

```
# ROS Parameters
odom_frame: odom
map_frame: map
base_frame: base_footprint
scan_topic: /scan_raw
mode: mapping #localization
```

4. Launch the SLAM node. It will publish in /map the map while it is building it

```
$ ros2 launch slam_toolbox online_async_launch.py params file:=[Full path
to ws]/ros2_ws/src/ikerlan/kobuki/params/mapper_params_online_async.yaml
use_sim_time:=false
```

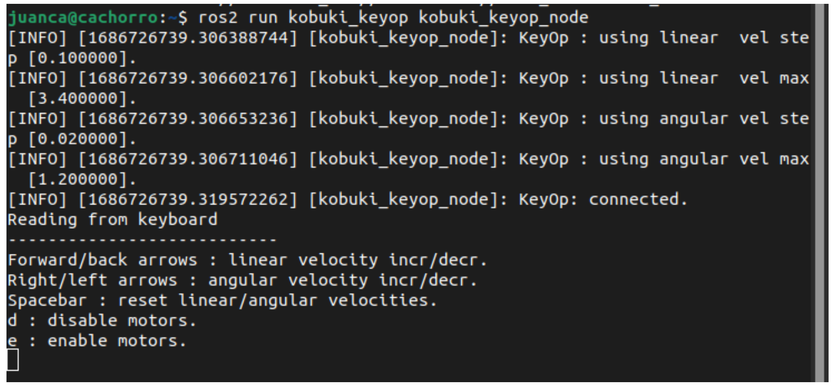
1 Creating a map using SLAM - Real Robot

5. Launch the map server. This node will subscribe to /map, and it will save the map to disk when requested

```
$ ros2 launch nav2_map_server map_saver_server.launch.py
```

6. Teleoperate the robot in order to explore the environment. Open Rviz2 and check how the map is built

```
$ros2 run kobuki_keyop kobuki_keyop_node
```



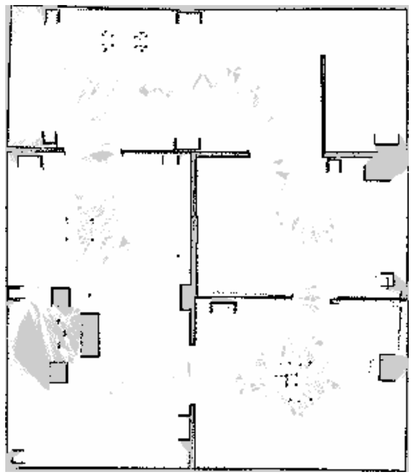
1 Creating a map using SLAM - Real Robot

7. When the map is completed, ask the map server to sabe the map to disk

```
$ ros2 run nav2 map_server map_saver_cli --ros-args -p use_sim_time:=false
```

At this point, two files have been created:

1. PGM image file: It can be modified if you need to do any fix



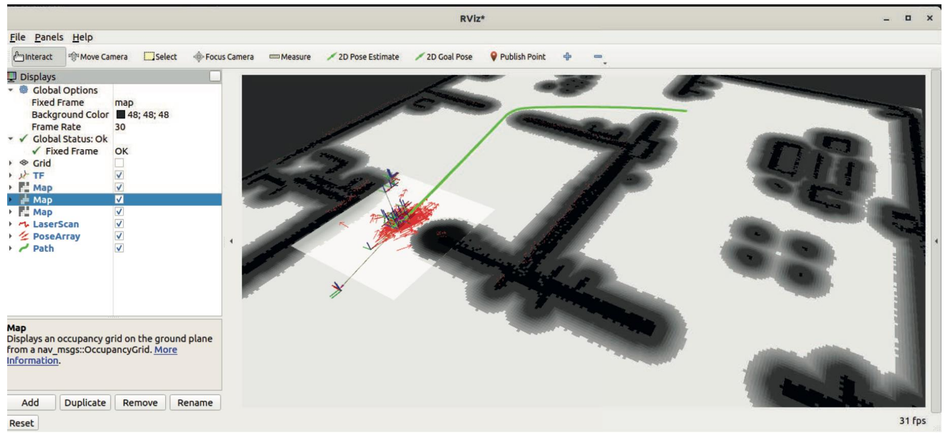
2. YAML file: contains enough information to interpret the image as a map.

2 Make the robot navigate with the map – Real Robot

1. Open Rviz2 and display:

```
$ rviz2
```

- Frame: map
- TF: To display the robot
- Map: Display the topic map
- LaserScan: To see how it matches with obstacels
- It is interesting to display the AMCL particles. Each one is a hypothesis about the robot´s position

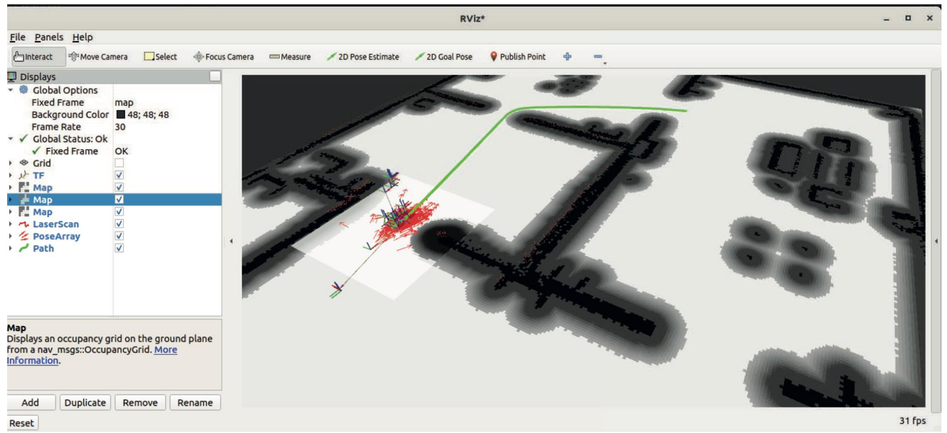


2. Use the “2D Goal Pose” button to command a goal position to the robot

2 Make the robot navigate with the map – Real Robot

3. Launch navigation

```
$ ros2 launch kobuki navigation.launch.py
```



2. Use the “2D Goal Pose” button to command a goal position to the robot