Integrating RGBD Camera to TB3 using ROS – A step-by-step Guide.

Part 1: Static Map (Using LiDAR sensor to create maps and navigation)

1) Run the roscore command from the remote PC.

```
jakelcj@jakelcj-Latitude-3420:~$ roscore
... logging to /home/jakelcj/.ros/log/866a71c4-4a2f-11ef-afba-0d80916fbd09/roslaunch-jakelcj-La
titude-3420-9169.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.142:32879/
ros_comm version 1.16.0
```

Diagram 1.0

2) **Connect the remote PC to the Raspberry PI with its IP address.**

    1) The default password for Turtlebot3 is “turtlebot”.

```
jakelcj@jakelcj-Latitude-3420:~$ ssh ubuntu@192.168.0.141
ubuntu@192.168.0.141's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-1104-raspi aarch64)
```

Diagram 2.0

3) **Bring up basic packages to start TurtleBot3 applications.**

```
ubuntu@ubuntu:~$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
... logging to /home/ubuntu/.ros/log/b3fa95ba-4a39-11ef-afba-0d80916fbd09/roslaunch-ubuntu-1389
.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.141:42395/
```

Diagram 3.0

1) When successfully done, the terminal will print these messages.

```
SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0
 * /turtlebot3_core/baud: 115200
 * /turtlebot3_core/port: /dev/ttyACM0
 * /turtlebot3_core/tf_prefix:
 * /turtlebot3_lds/frame_id: base_scan

NODES
  /
    turtlebot3_core (rosserial_python/serial_node.py)
    turtlebot3_diagnostics (turtlebot3_bringup/turtlebot3_diagnostics)
    turtlebot3_lds (ld08_driver/ld08_driver)

ROS_MASTER_URI=http://192.168.0.142:11311

process[turtlebot3_core-1]: started with pid [1406]
process[turtlebot3_lds-2]: started with pid [1407]
process[turtlebot3_diagnostics-3]: started with pid [1408]
/dev/ttyUSB0    CP2102 USB to UART Bridge Controller
/dev/ttyACM0    OpenCR Virtual ComPort in FS Mode
FOUND LiDAR_LD08 @port :/dev/ttyUSB0
[INFO] [1721880218.579298]: ROS Serial Python Node
[INFO] [1721880218.639821]: Connecting to /dev/ttyACM0 at 115200 baud
[INFO] [1721880220.763215]: Requesting topics...
[INFO] [1721880220.824019]: Note: publish buffer size is 1024 bytes
[INFO] [1721880220.832980]: Setup publisher on sensor_state [turtlebot3_msgs/SensorState]
[INFO] [1721880220.972804]: Setup publisher on firmware_version [turtlebot3_msgs/VersionInfo]
[INFO] [1721880221.110848]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1721880221.131893]: Setup publisher on cmd_vel_rc100 [geometry_msgs/Twist]
[INFO] [1721880221.176739]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1721880221.203059]: Setup publisher on joint_states [sensor_msgs/JointState]
[INFO] [1721880221.247974]: Setup publisher on battery_state [sensor_msgs/BatteryState]
[INFO] [1721880221.269213]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]
[INFO] [1721880221.336492]: Setup publisher on /tf [tf/tfMessage]
[INFO] [1721880221.370921]: Note: subscribe buffer size is 1024 bytes
[INFO] [1721880221.383085]: Setup subscriber on cmd_vel [geometry_msgs/Twist]
[INFO] [1721880221.431927]: Setup subscriber on sound [turtlebot3_msgs/Sound]
```

Diagram 3.1

4) **Launch SLAM Node:**

1) Launch a SLAM package (default SLAM method will be gmapping) to generate a 2D occupancy grid map.

```
jakelcj@jakelcj-Latitude-3420:~$ export TURTLEBOT3_MODEL=burger
jakelcj@jakelcj-Latitude-3420:~$ roslaunch turtlebot3_slam turtlebot3_slam.launch
... logging to /home/jakelcj/.ros/log/b3fa95ba-4a39-11ef-afba-0d80916fbd09/roslaunch-jakelcj-Latitude-3420-14554.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: in-order processing became default in ROS Melodic. You can drop the option.
started roslaunch server http://192.168.0.142:44343/
```

Diagram 4.0

2) 3D visualization tool such as rviz will automatically open displaying the current surrounding around the TB3.

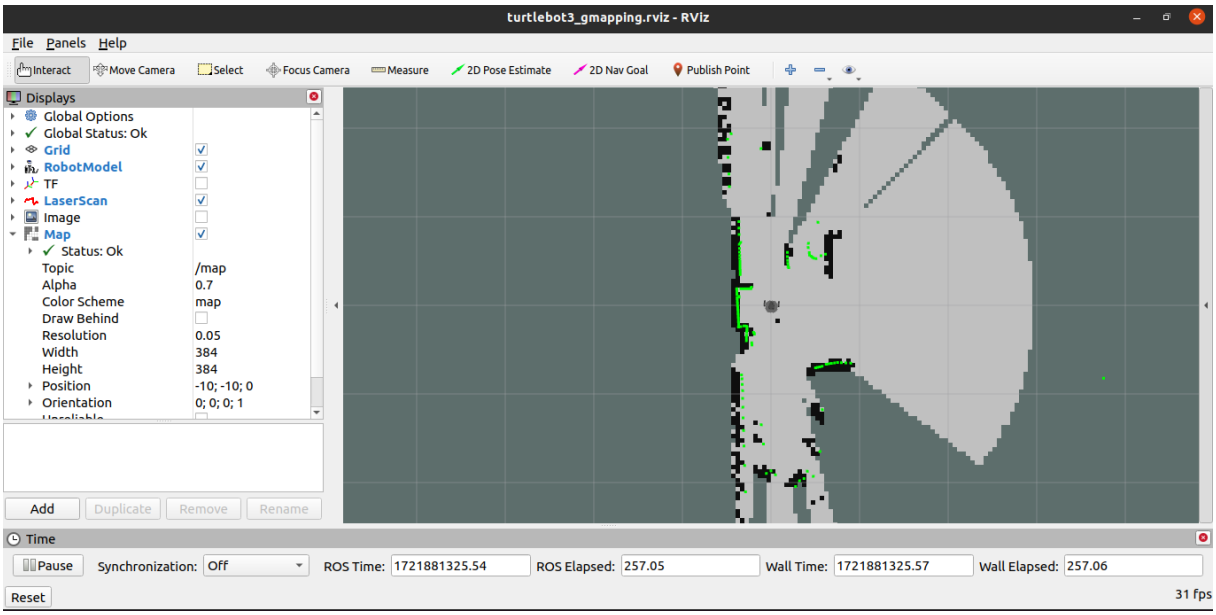


Diagram 4.1

5) **Run the teleoperation node from the Remote PC.**

```
jakelcj@jakelcj-Latitude-3420:~$ export TURTLEBOT3_MODEL=burger
jakelcj@jakelcj-Latitude-3420:~$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
... logging to /home/jakelcj/.ros/log/b3fa95ba-4a39-11ef-afba-0d80916fbd09/roslaunch-jakelcj-Latitude-3420-15007.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.0.142:41329/
```

Diagram 5.0

1) When successfully done, a virtual keyboard analog will appear to control the TurtleBot.

```
ROS_MASTER_URI=http://192.168.0.142:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [15021]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit
```

Diagram 5.1

6) **Collect information by exploring using the TurtleBot3.**

7) **Save the Static Map**:

1) Once the mapping is complete, use the map_saver node from the map_server package to save the generated map as a .pgm and .yaml file for future use.

```
jakelcj@jakelcj-Latitude-3420:~$ rosrun map_server map_saver -f ~/map
[ INFO] [1721887120.127019239]: Waiting for the map
[ INFO] [1721887120.386785739]: Received a 384 X 384 map @ 0.050 m/pix
[ INFO] [1721887120.386830820]: Writing map occupancy data to /home/jakelcj/map.pgm
[ INFO] [1721887120.393611323]: Writing map occupancy data to /home/jakelcj/map.yaml
[ INFO] [1721887120.393860736]: Done
```

Diagram 7.0

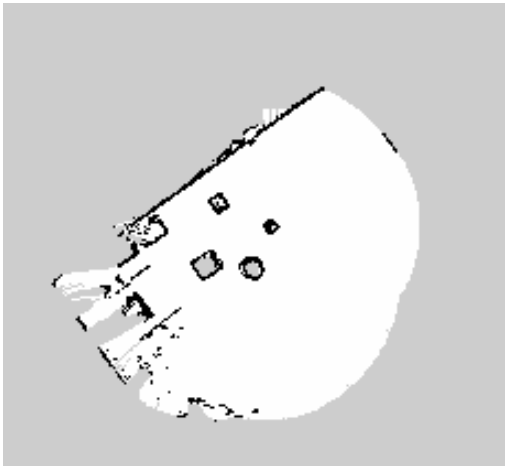


Diagram 7.1

map.yaml

C: > Users > jacob > Downloads > map.yaml

```yaml
image: /home/jakelcj/map.pgm
resolution: 0.050000
origin: [-10.000000, -10.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

Diagram 7.2

8) **Autonomous Navigation of a Known Map with TurtleBot**:

1) Test the navigation stack by loading the static map and observing how the TurtleBot3 navigates within the mapped area.

2) Use RViz to visualize the map and verify its accuracy.

Steps:

1) Launch the Navigation on RViz with the map saved using SLAM.

```
jakelcj@jakelcj-Latitude-3420:~$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch m
ap_file:=$HOME/map.yaml
```

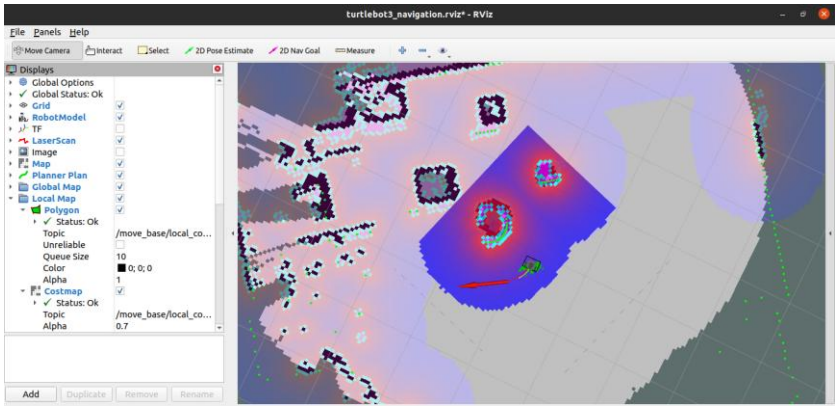
2) Once the map is opened, check whether the robot is correctly placed and whether the surroundings detected matched the map (There should be red light around the robot representing the objects).

    - If in case the robot did not detect any object near it, check the time synchronization between the Remote PC and the robot.

3) To move the robot to the correct location, perform the Initial Pose Estimation by selecting the 2D Pose Estimation function and dragging the green arrow toward the direction where the robot is facing.

4) You can also launch keyboard teleoperation node to precisely locate the robot on the map. Don't forget to terminate the keyboard to prevent different cmd_vel values from being published from multiple nodes during Navigation.

5) Lastly, set the navigation goal by selecting the 2D Nav Goal function and dragging the red arrow toward the direction where you want the robot to go, and the robot will find the shortest path to reach the destination.



Part 2: Dynamic Map (Integrating the use of Depth Camera for 3D Vision)

Initial step: Please make sure the Raspberry PI is installed with the correct ROS version (Noetic for ROS1).

Setup and initial procedure

Connect the Camera and the Raspberry Pi

1) Connect the camera to the raspberry Pi via USB cable.
2) Ensure that the camera is successfully detected by the device by using the command 'lsusb'.

Install necessary dependencies and setting up the procedures.

1) Once verified, go to https://github.com/luxonis/depthai-ros and scroll until you see the link to the newest documentation.
2) It will bring you to luxonis page which specifically to setup DepthAI ROS.
3) You will need to install from source by just following the steps given.
    - If you are using Raspberry Pi, before you try to build your workspace by using the command 'catkin_make', use the command 'export MAKEFLAGS="-j4" 'to specify the number of parallel jobs so that the Raspberry Pi won't crash.
4) Check if all the required files are properly built and all directory is available.

Configure Network Settings between Master and Client.

1) You will need to append the environment variable settings directly into the ~/.bashrc file of those devices.
    - Set ROS Master URI and Hostname:
        - For the Laptop:

```
echo 'export ROS_MASTER_URI=http://192.168.1.100:11311' >> ~/.bashrc
echo 'export ROS_HOSTNAME=192.168.1.200' >> ~/.bashrc
```

Replace 192.168.1.100 with the IP address of your ROS master and 192.168.1.200 with the IP address of your laptop.

- For the Raspberry Pi:

```
echo 'export ROS_MASTER_URI=http://192.168.1.100:11311' >> ~/.bashrc
echo 'export ROS_HOSTNAME=192.168.1.200' >> ~/.bashrc
```

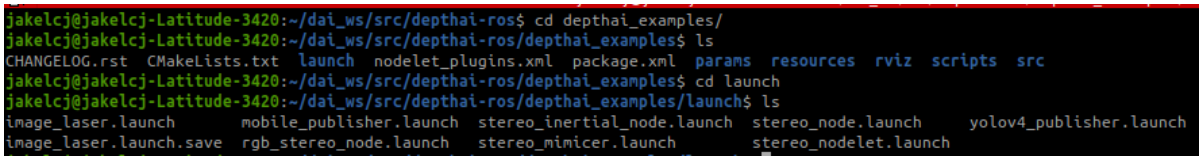
Replace 192.168.1.100 with the IP address of your ROS master and 192.168.1.200 with the IP address of your Raspberry Pi.

2) Once done, apply the changes immediately by using the command 'source ~/.bashrc'.

Camera Startup

Launch the DepthAI camera node

1) You will need to source your ROS environment first by using the command 'source ~/dai_ws_ws/devel/setup.bash'.
2) Next, find the launch file you wanted by diving into multiple directories.



3) Run the roscore command from the remote PC.
4) Launch the node by using the command 'roslaunch' (ROS1) and 'ros2 launch (ROS2).

Create a Static Map using SLAM

Please refer to the previous example.

First Layer (Converting the images to laser scan)

Build the Node from Source.

1) Now you will need to create clone the ‘depthimage_to_laserscan’ repository from the ROS GitHub.

```
jakelcj@jakelcj-Latitude-3420:~$ cd dai_ws/src
jakelcj@jakelcj-Latitude-3420:~/dai_ws/src$ git clone https://github.com/ros-perception/depthimage_to_laserscan.git
```

2) Then build the package again.

```
catkin_make
source devel/setup.bash
```

3) Check if the ‘depthimage_to_laserscan’ node is available in the launch file.

```
jakelcj@jakelcj-Latitude-3420:~$ cd dai_ws/src/depthimage_to_laserscan/launch/
jakelcj@jakelcj-Latitude-3420:~/dai_ws/src/depthimage_to_laserscan/launch$ ls
depthimage_to_laserscan.launch
jakelcj@jakelcj-Latitude-3420:~/dai_ws/src/depthimage_to_laserscan/launch$
```

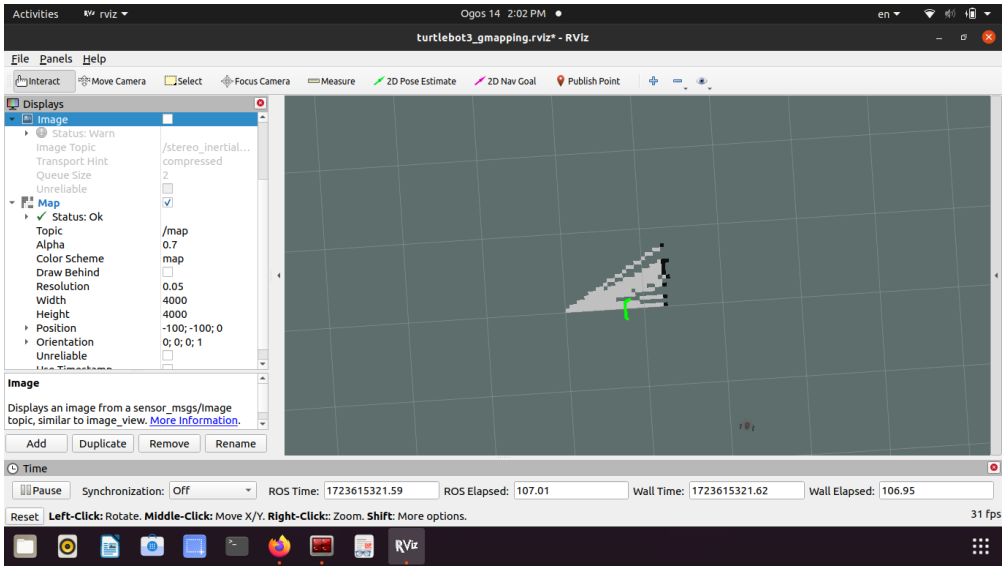
4) Open gmapping file in SLAM directory to edit the file to launch the ‘depthimage_to_laserscan’ functionality.

```
jakelcj@jakelcj-Latitude-3420: /opt/ros/noetic/share/turtlebot3_slam/launch 95x39
 GNU nano 4.8                    turtlebot3_gmapping.launch
<launch>

   <!-- Publish Static Transform between base_link and map -->
   <node pkg="tf" type="static_transform_publisher" name="base_to_map_transform" args="2.2 -1>

<!-- Depth Image to Laser Scan Node -->
   <node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_las>
       <param name="scan_height" value="1" />
       <param name="scan_time" value="3.0" />
       <param name="range_min" value="0.1" />
       <param name="range_max" value="2.5" />
       <param name="output_frame_id" value="map" />
       <remap from="image" to="/stereo_inertial_publisher/stereo/depth" />
       <remap from="camera_info" to="/stereo_inertial_publisher/color/camera_info" />
       <remap from="scan" to="/laser_scan"/>
   </node>
```

5) To use the ‘depthimage_to_laserscan’ node for mapping in SLAM, remap from ‘scan’ to ‘/laser_scan’

```
<!-- Gmapping -->
<node pkg="gmapping" type="slam_gmapping" name="turtlebot3_slam_gmapping" output="screen">
  <param name="base_frame" value="$(arg set_base_frame)"/>
  <param name="odom_frame" value="$(arg set_odom_frame)"/>
  <param name="map_frame" value="$(arg set_map_frame)"/>
  <remap from="scan" to="/laser_scan"/>
</node>
```

6) Now bringup the turtlebot and launch the ‘turtlebot3_slam.launch’ file to view the output.



Merging the laser from the camera and LIDaR sensor.

1) Clone the Repository.

```
jakelcj@jakelcj-Latitude-3420:~/dai_ws/src$ git clone https://github.com/ira-laser-tools/ira_laser_tools.git
```

2) Build the package and source your workspace.

```
catkin_make
source devel/setup.bash
```

3) Edit the ‘Laserscan_multi_merger.launch’ file.

```
<launch>
    <!-- Launch the existing multi-merger node if needed -->
    <node pkg="ira_laser_tools" name="laserscan_multi_merger" type="laserscan_multi_merger" >
        <param name="destination_frame" value="base_footprint"/>
        <param name="laserscan_topics" value="/laser_scan /scan" />
        <param name="angle_min" value="-2.0"/>
        <param name="angle_max" value="50.0"/>
        <param name="angle_increment" value="0.0058"/>
        <param name="scan_time" value="0.05"/>
        <param name="range_min" value="0.00"/>
        <param name="range_max" value="60.0"/>
    </node>

</launch>
```

4) Edit the ‘turtlebot3_gmapping. launch’ file by adding the the following node to subscribe to the merged topic.

```
  <!-- Gmapping -->
  <node pkg="gmapping" type="slam_gmapping" name="turtlebot3_slam_gmapping" output="screen">
    <param name="base_frame" value="$(arg set_base_frame)"/>
    <param name="odom_frame" value="$(arg set_odom_frame)"/>
    <param name="map_frame" value="$(arg set_map_frame)"/>
    <remap from="scan" to="/scan_multi"/>
  </node>
```

5) Launch the ‘scan_merger.py’ file by using ‘python3’ command.

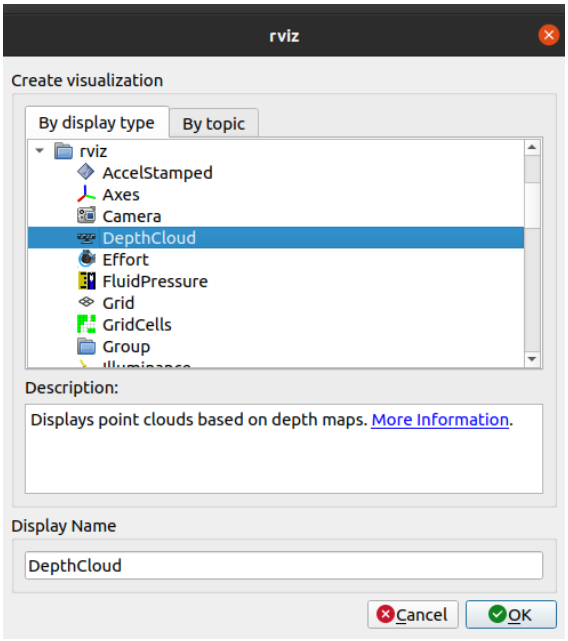
```
jakelcj@jakelcj-Latitude-3420:~/dai_ws/src/ira_laser_tools/launch$ roslaunch laserscan_multi_merger.launch
```

6) View the laser in the rviz

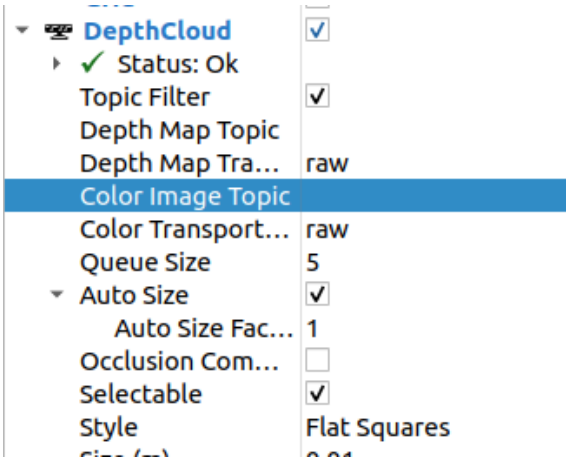
Second Layer (Displaying the 3D images onto the map)

Subscribe to the camera node.

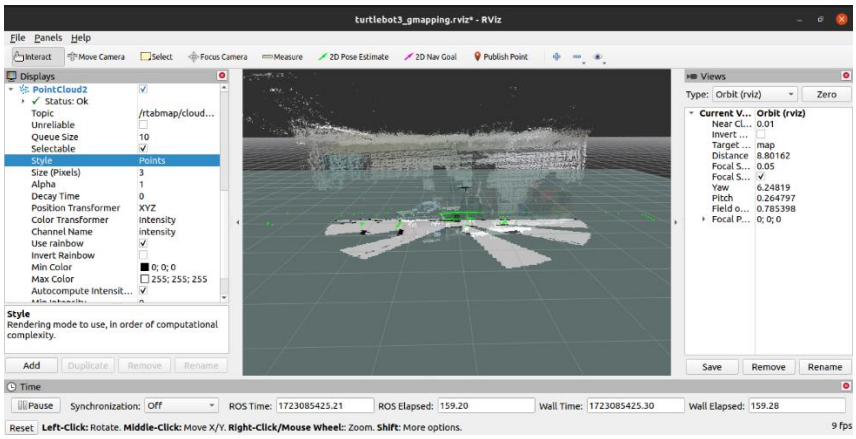
1) In the rviz click the button ‘add’ and add either DepthCloud or Pointcloud2 topic.



2) Click on the topic and choose the right image topic to subscribe to.



3) The image will be displayed on the map.



Troubleshooting the incorrect position of the 3D images and the camera laser_scan.

1) Open the camera launch file and change the x,y,z,roll,pitch, and yaw of the camera.

```
<arg name="cam_pos_x" default="2.2" /> <!-- Example position relative to base_link -->
<arg name="cam_pos_y" default="0.83" />
<arg name="cam_pos_z" default="1.17" />
<arg name="cam_roll" default="0.0" />
<arg name="cam_pitch" default="0.5" />
<arg name="cam_yaw" default="78.7" />
```

Troubleshooting the incorrect position of the combined laser scan.

1) Open the ‘laserscan_multi_merger.launch’ and change the configuration to synchronize with the actual placement.

```
<param name="angle_min" value="-2.0"/>
<param name="angle_max" value="50.0"/>
<param name="angle_increment" value="0.0058"/>
<param name="scan_time" value="0.05"/>
<param name="range_min" value="0.00"/>
<param name="range_max" value="60.0"/>
```