Slide

https://kasets.art/mT81xr

Robot sensor workshops

Workshop for robot sensor.

robotcitizens.org

Webcam & External camera

How to use webcam and external camera with ROS.

Webcam & External camera

```
$ sudo apt install ros-noetic-usb-cam
```

Webcam & External camera

```
$ rospack profile
```

Webcam & External camera

```
$ roscore
```

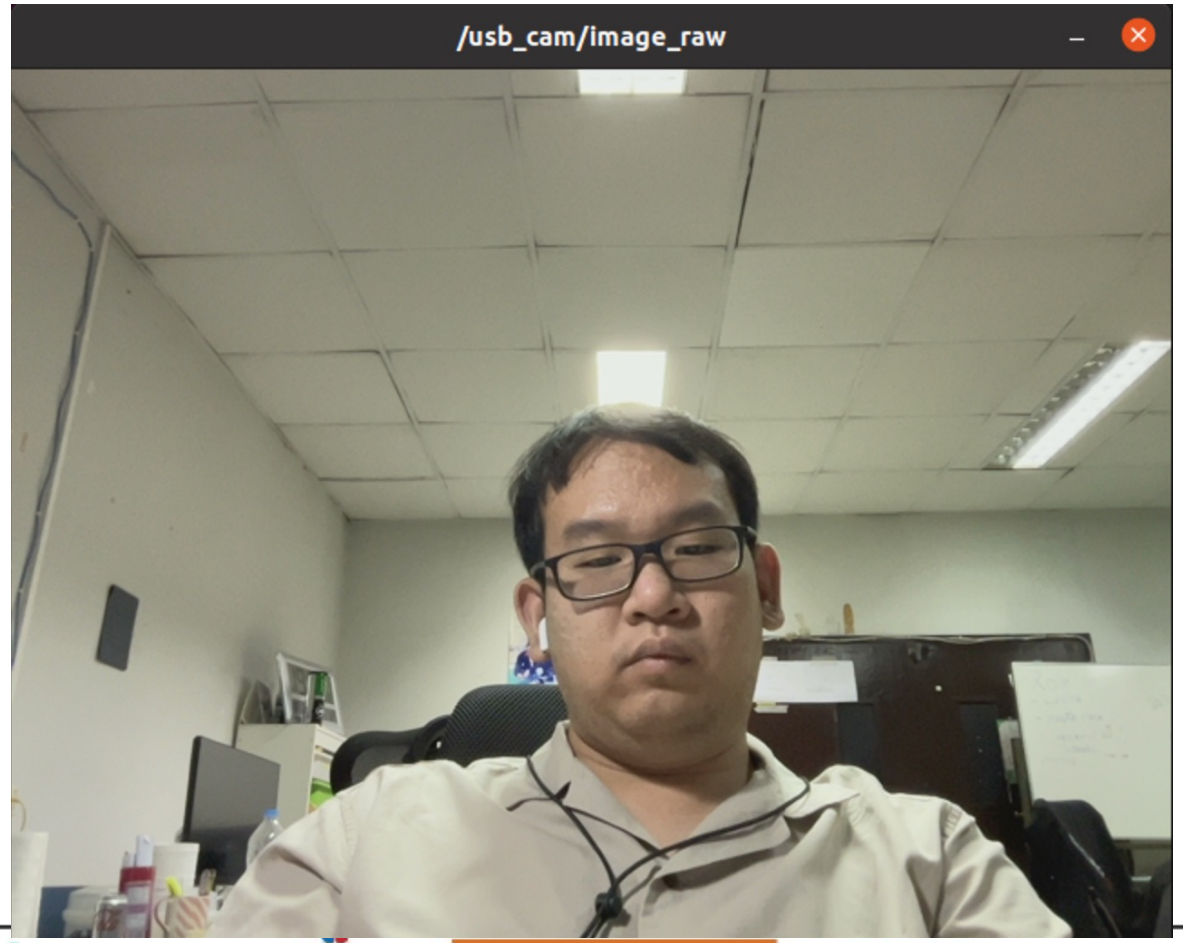
Webcam & External camera

Open new termimal

Webcam & External camera

```
$ roslaunch usb_cam usb_cam-test.launch
```

Webcam & External camera



Webcam & External camera

Create launch file on your package

Webcam & External camera

```
$ roscd your_package/launch
```

Webcam & External camera

```
$ gedit usb-cam.launch
```

Webcam & External camera

code

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Webcam & External camera

```
$ ls /dev/ | grep video
```

```
pat@pat:~$ ls /dev/ | grep video
video0
video1
```

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

1080P

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node>
</launch>
```

Webcam & External camera

```
<launch>
  <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="1920" />
    <param name="image_height" value="1080" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
  </node>
  <!-- <node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="true" />
  </node> -->
</launch>
```

Comment

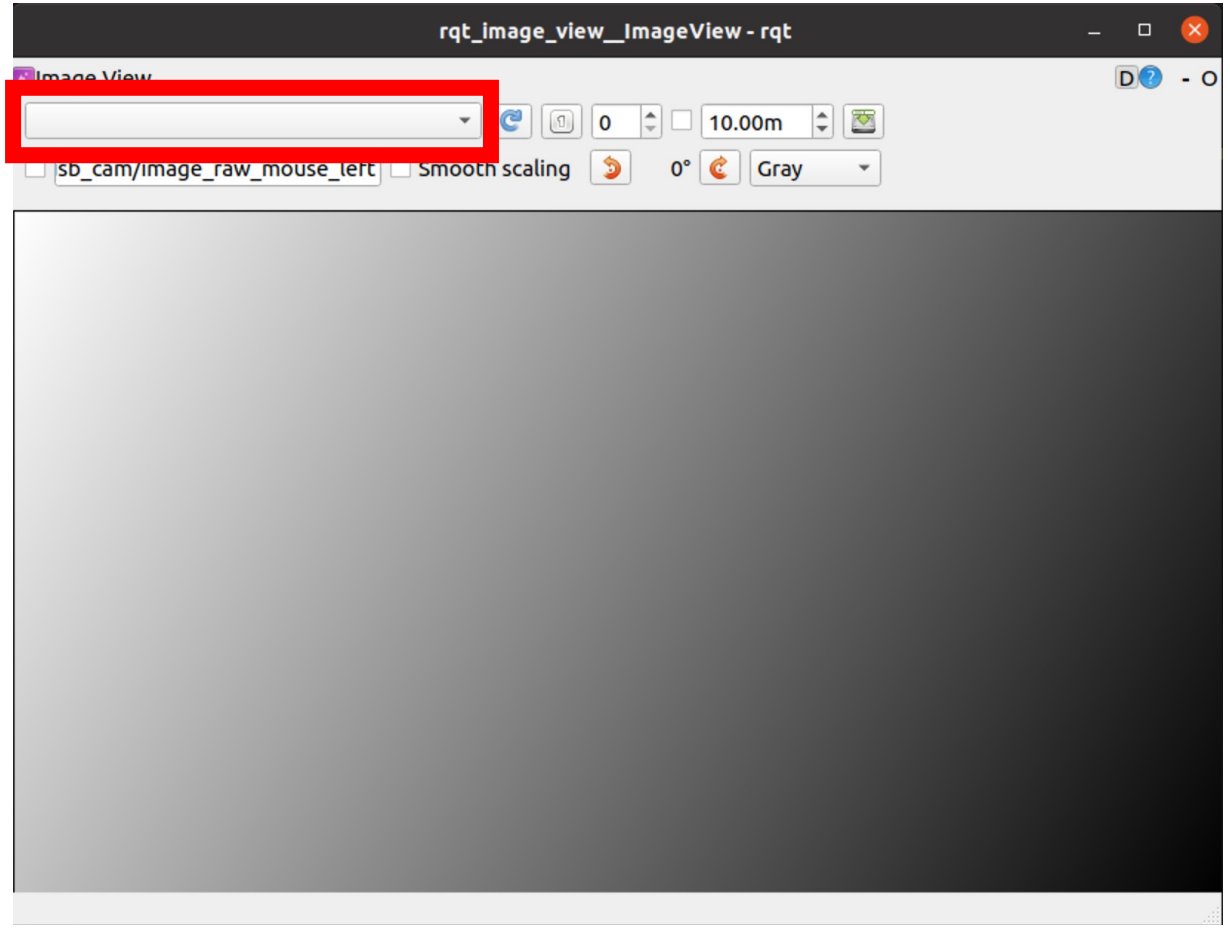
Webcam & External camera

```
$ roslaunch your_package usb_cam.launch
```

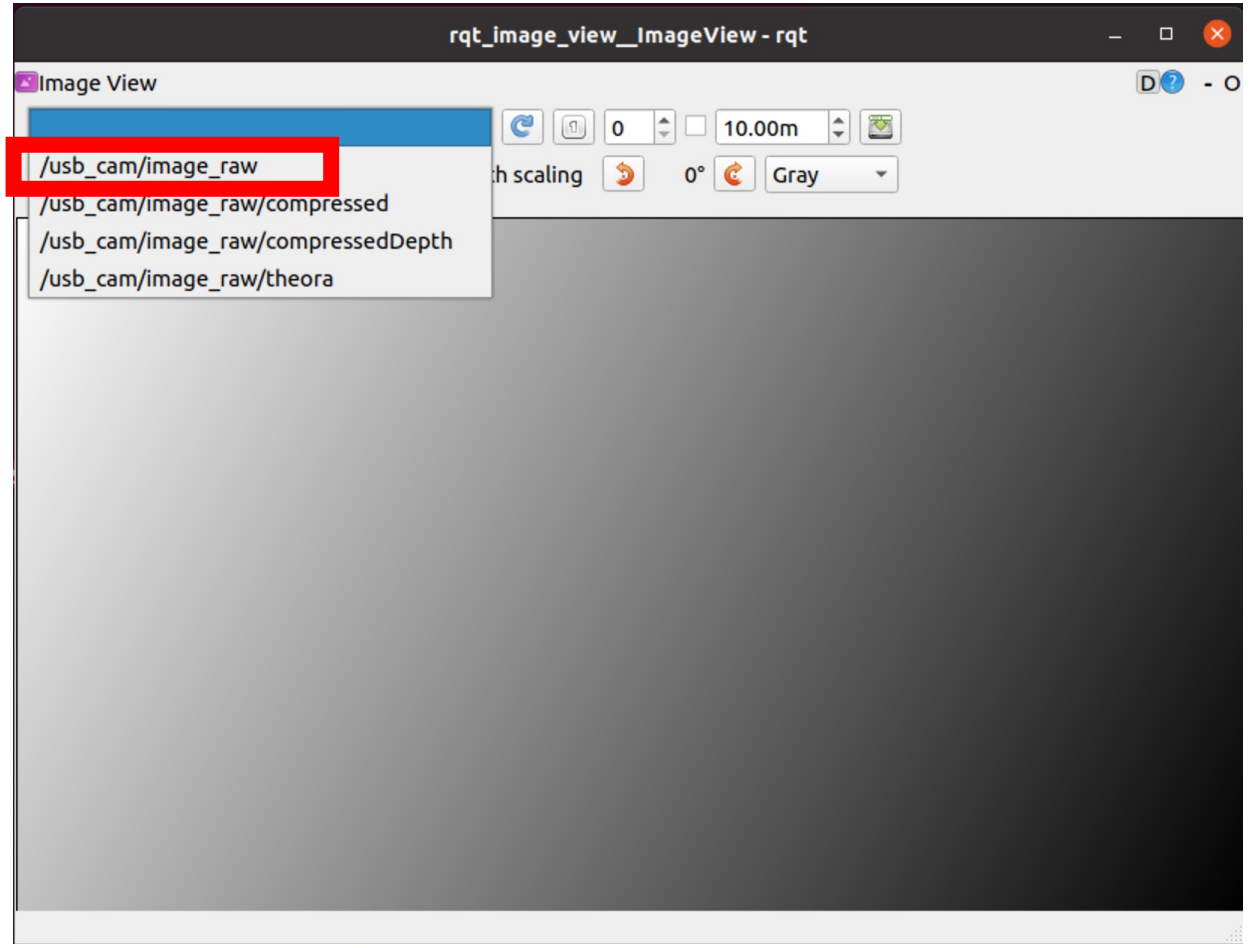
Webcam & External camera

```
$ rqt_image_view
```

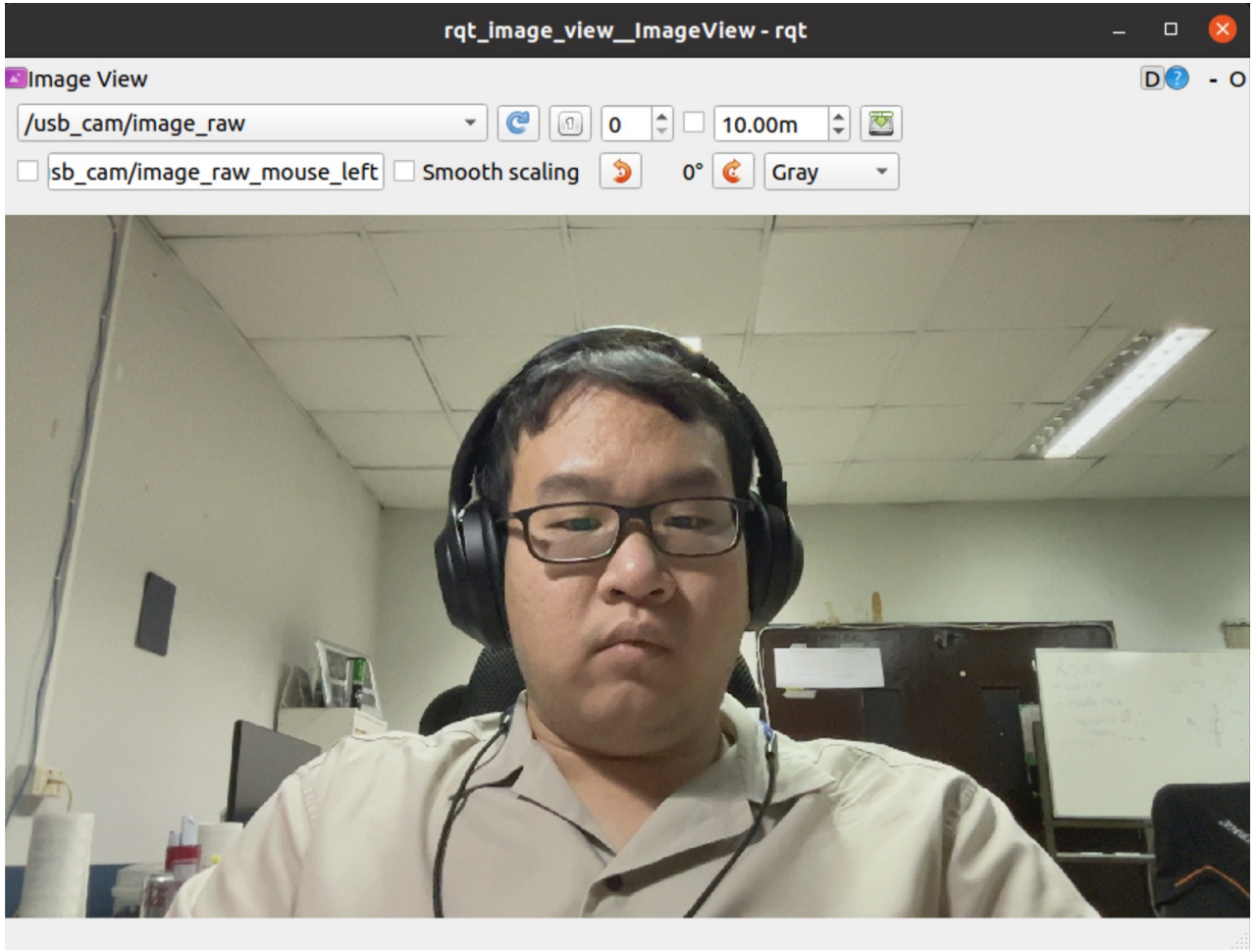
Webcam & External camera



Webcam & External camera



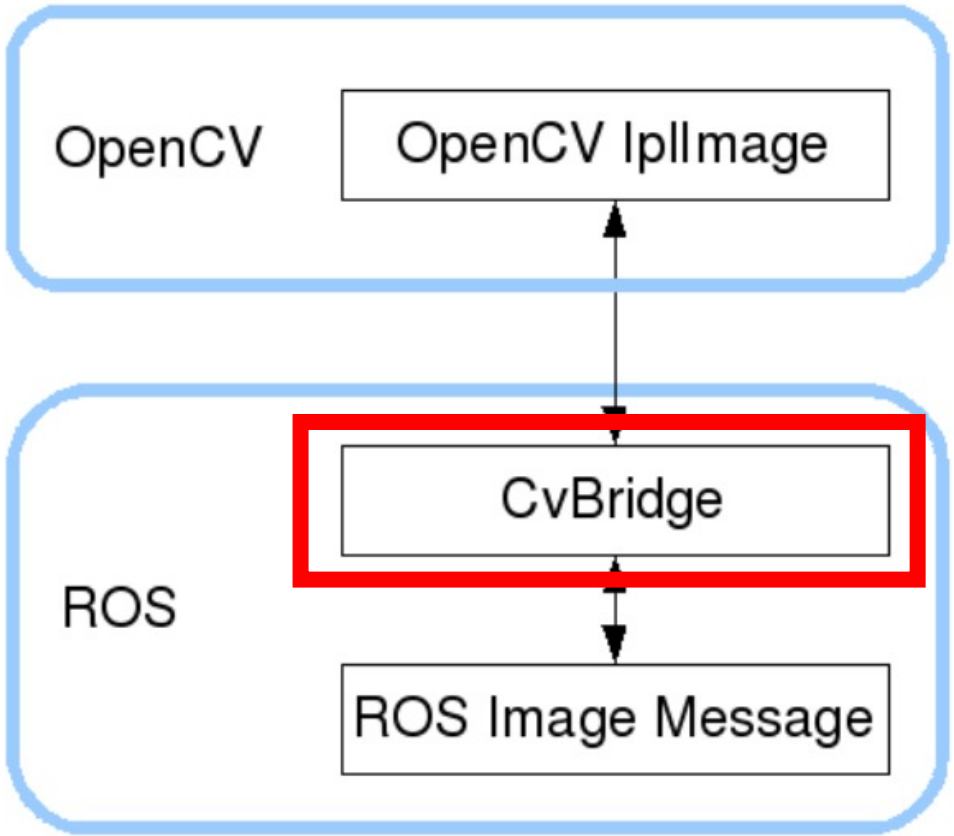
Webcam & External camera



Webcam & External camera

Using with OpenCV

Webcam & External camera



Webcam & External camera

```
$ sudo apt install ros-noetic-cv-bridge
```

Webcam & External camera

If you **don't** have directory ***script***.

```
$ roscd your_package
```

```
$ mkdir script
```

If you **already** have directory ***script***.

```
$ roscd your_package/script
```

Webcam & External camera

```
$ gedit opencv_ros.py
```

Webcam & External camera

code

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        rospy.init_node("opencv_ros", anonymous=True)
        self.img_sub = rospy.Subscriber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.bridge = CvBridge()
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self):
        rospy.init_node("opencv_ros", anonymous=True)
        self.image_sub = rospy.Subscriber("/camera/image_raw", Image, callback = self.image_callback)
        self.bridge = CvBridge()
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        self.img_sub = rospy.Subscriber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        rospy.init_node("opencv_ros", anonymous=True)
        iber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.bridge = CvBridge()

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        rospy.init_node("opencv_ros", anonymous=True)
        self.img_sub = rospy.Subscriber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        rospy.init_node("opencv_ros", anonymous=True)
        self.img_sub = rospy.Subscriber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.bridge = CvBridge()
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```python
#!/usr/bin/env python3
import cv2
import rospy
from cv_bridge import CvBridge
from sensor_msgs.msg import Image


class OpenCVROS(object):
    def __init__(self) -> None:
        rospy.init_node("opencv_ros", anonymous=True)
        self.img_sub = rospy.Subscriber("/usb_cam/image_raw", Image, callback = self.image_callback)
        self.bridge = CvBridge()
        self.image = None

    def image_callback(self, data):
        self.image = self.bridge.imgmsg_to_cv2(data, desired_encoding="bgr8")

    def show_image(self):
        if self.image is None:
            return 0
        cv2.imshow("image", self.image)
        if cv2.waitKey(1) == ord('q'):
            exit()

    def main(self):
        while not rospy.is_shutdown():
            self.show_image()

if __name__ == "__main__":
    opencv_ros = OpenCVROS()
    opencv_ros.main()
```

Webcam & External camera

```
$ roslaunch your_package usb_cam.launch
```

Webcam & External camera

```
$ python3 opencv_ros.py
```

RPLidar A3

How to use RPLiDAR A3 with ROS.

LiDAR – [RPLidar A3]

```
$ roscd
```

```
$ cd ../src
```

```
$ git clone https://github.com/Slamtec/rplidar_ros.git
```

LiDAR – [RPLidar A3]

```
$ cd ../
```

```
$ catkin_make
```

```
$ rospack profile
```

LiDAR – [RPLidar A3]

```
$ sudo usermod -aG dialout $USER
```

```
$ sudo reboot
```

LiDAR – [RPLidar A3]

```
$ export EDITOR="gedit"
```

```
$ rosed rplidar_ros rplidar_a3.launch
```

LiDAR – [RPLidar A3]

Open | rplidar_a3.launch | ~/work_ws/aiforrobot_ws/src/rplidar_ros/launch | Save

```
<launch>
  <node name="rplidarNode"          pkg="rplidar_ros"  type="rplidarNode"
 output="screen">
  <param name="serial_port"         type="string" value="/dev/ttyUSB0"/>
  <param name="serial_baudrate"     type="int"                          -->
  <param name="frame_id"            type="string" value="laser"/>
  <param name="inverted"            type="bool"   value="false"/>
  <param name="angle_compensate"    type="bool"   value="true"/>
  <param name="scan_mode"           type="string" value="Sensitivity"/>
  </node>
</launch>
```

Plain Text | Tab Width: 8 | Ln 1, Col 1 | INS

LiDAR – [RPLidar A3]

```
$ ls /dev | grep USB
```

```
pat@pat:~$ ls /dev/ | grep USB
ttyUSB0
```

LiDAR – [RPLidar A3]

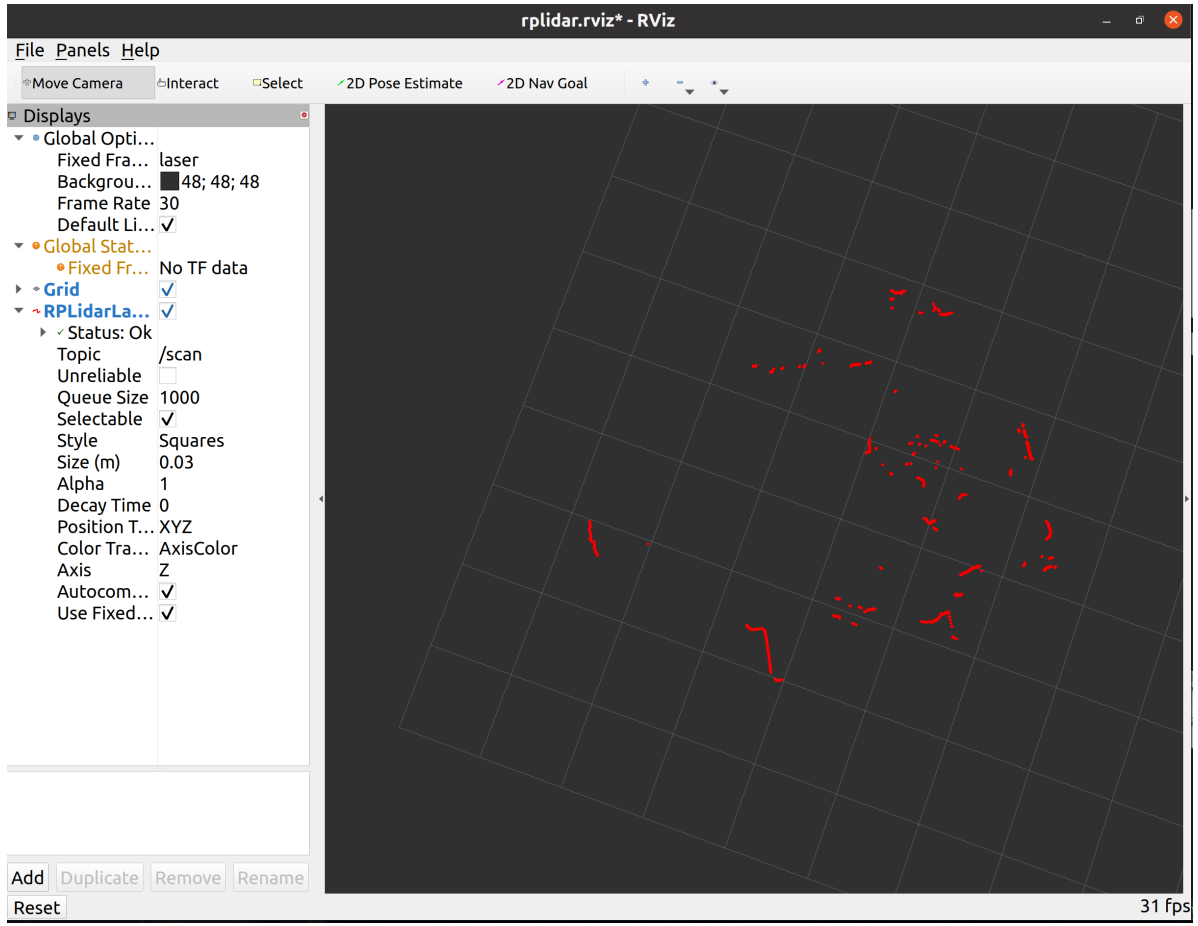
Open new termimal

LiDAR – [RPLidar A3]

```
$ roslaunch rplidar_ros rplidar_a3.launch
```

```
$ roslaunch rplidar_ros view_rplidar_a3.launch
```

LiDAR – [RPLidar A3]



YDLidar – X2

How to use YDLidar X2 with ROS

LiDAR – [YDLidar-X2]

```
$ cd ~/
```

```
$ git clone https://github.com/YDLIDAR/YDLidar-SDK.git
```

LiDAR – [YDLidar-X2]

```
$ cd YDLidar-SDK/
```

```
$ mkdir build
```

```
$ cd build
```

LiDAR – [YDLidar-X2]

```
$ cmake ..
```

```
$ make
```

```
$ sudo make install
```

LiDAR – [YDLidar-X2]

```
$ roscd
```

```
$ cd ../src
```

```
$ git clone https://github.com/YDLIDAR/ydlidar_ros_driver.git
```

LiDAR – [YDLidar-X2]

```
$ cd ../
```

```
$ catkin_make
```

```
$ rospack profile
```

LiDAR – [YDLidar-X2]

```
$ sudo usermod -aG dialout $USER
```

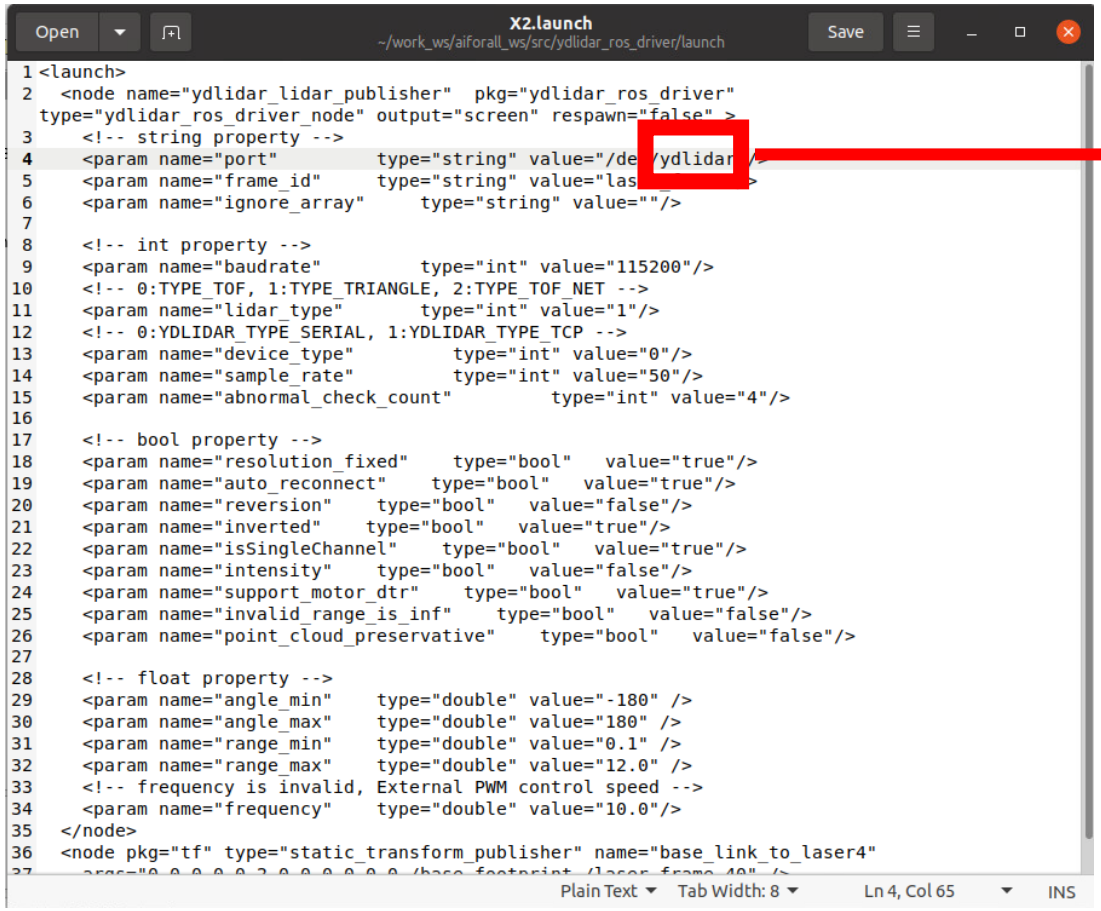
```
$ sudo reboot
```

LiDAR – [YDLidar-X2]

```
$ export EDITOR="gedit"
```

```
$ rosed ydlidar_ros_driver X2.launch
```

LiDAR – [YDLidar-X2]




```
<launch>
  <node name="ydlidar_lidar_publisher"  pkg="ydlidar_ros_driver"
  type="ydlidar_ros_driver_node" output="screen" respawn="false" >
    <!-- string property -->
    <param name="port"         type="string" value="/dev/ydlidar"/>
    <param name="frame_id"     type="string" value="laser_frame"/>
    <param name="ignore_array"     type="string" value=""/>

    <!-- int property -->
    <param name="baudrate"         type="int" value="115200"/>
    <!-- 0:TYPE_TOF, 1:TYPE_TRIANGLE, 2:TYPE_TOF_NET -->
    <param name="lidar_type"       type="int" value="1"/>
    <!-- 0:YDLIDAR_TYPE_SERIAL, 1:YDLIDAR_TYPE_TCP -->
    <param name="device_type"         type="int" value="0"/>
    <param name="sample_rate"         type="int" value="50"/>
    <param name="abnormal_check_count"         type="int" value="4"/>

    <!-- bool property -->
    <param name="resolution_fixed"    type="bool"   value="true"/>
    <param name="auto_reconnect"    type="bool"   value="true"/>
    <param name="reversion"    type="bool"   value="false"/>
    <param name="inverted"    type="bool"   value="true"/>
    <param name="isSingleChannel"    type="bool"   value="true"/>
    <param name="intensity"    type="bool"   value="false"/>
    <param name="support_motor_dtr"    type="bool"   value="true"/>
    <param name="invalid_range_is_inf"    type="bool"   value="false"/>
    <param name="point_cloud_preservative"    type="bool"   value="false"/>

    <!-- float property -->
    <param name="angle_min"    type="double" value="-180" />
    <param name="angle_max"    type="double" value="180" />
    <param name="range_min"    type="double" value="0.1" />
    <param name="range_max"    type="double" value="12.0" />
    <!-- frequency is invalid, External PWM control speed -->
    <param name="frequency"    type="double" value="10.0"/>
  </node>
  <node pkg="tf" type="static_transform_publisher" name="base_link_to_laser4"
```

USB Interface

LiDAR – [YDLidar-X2]

```
$ ls /dev | grep USB
```

```
pat@pat:~$ ls /dev/ | grep USB
ttyUSB0
```

LiDAR – [YDLidar-X2]

Open | X2.launch | ~/work_ws/aiforall_ws/src/ydlidar_ros_driver/launch | Save

```
1 <launch>
2   <node name="ydlidar_lidar_publisher"  pkg="ydlidar_ros_driver"
  type="ydlidar_ros_driver_node" output="screen" respawn="false" >
3     <!-- string property -->
4     <param name="port"         type="string" value="/dev/ttyUSB0"/>
5     <param name="frame_id"     type="string" value="laser_frame"/>
6     <param name="ignore_array"     type="string" value=""/>
7
8     <!-- int property -->
9     <param name="baudrate"         type="int" value="115200"/>
10    <!-- 0:TYPE_TOF, 1:TYPE_TRIANGLE, 2:TYPE_TOF_NET -->
11    <param name="lidar_type"       type="int" value="1"/>
12    <!-- 0:YDLIDAR_TYPE_SERIAL, 1:YDLIDAR_TYPE_TCP -->
```

LiDAR – [YDLidar-X2]

```
$ roslaunch ydlidar_ros_driver X2.launch
```

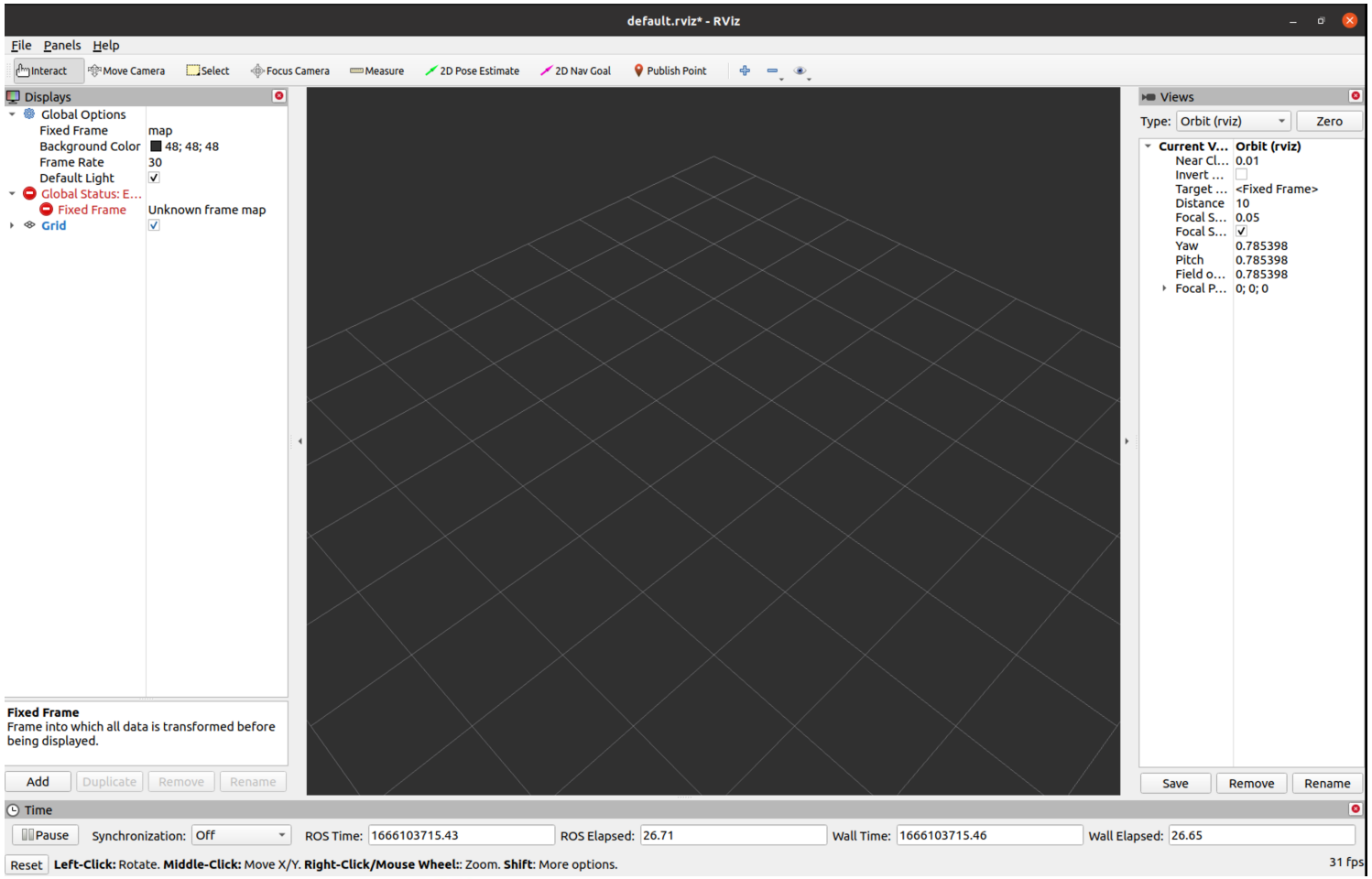
LiDAR – [YDLidar-X2]

Open new termimal

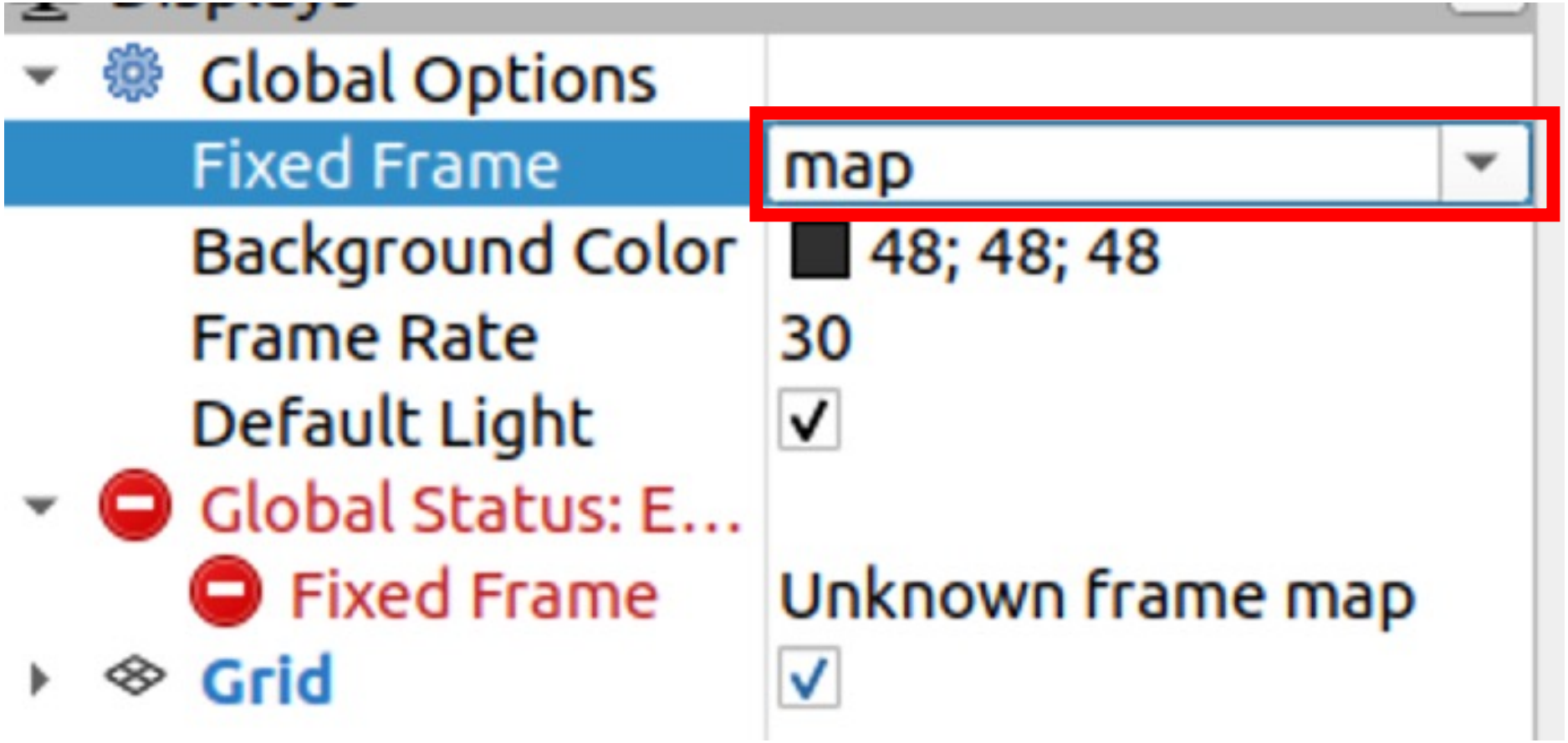
LiDAR – [YDLidar-X2]

```
$ rviz
```

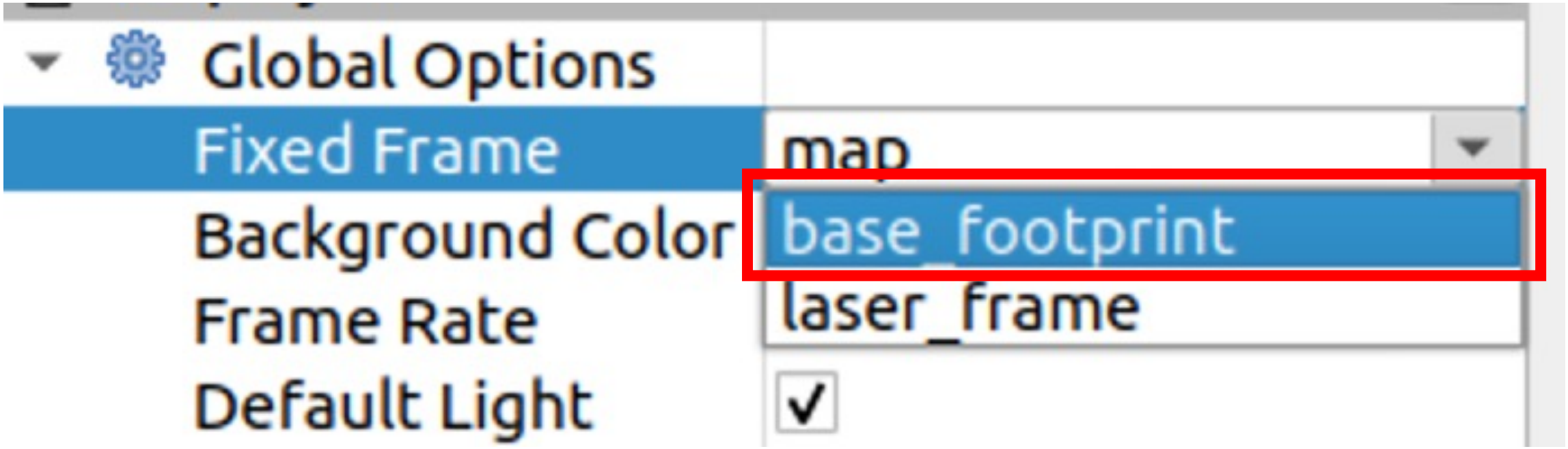
LiDAR – [YDLidar-X2]



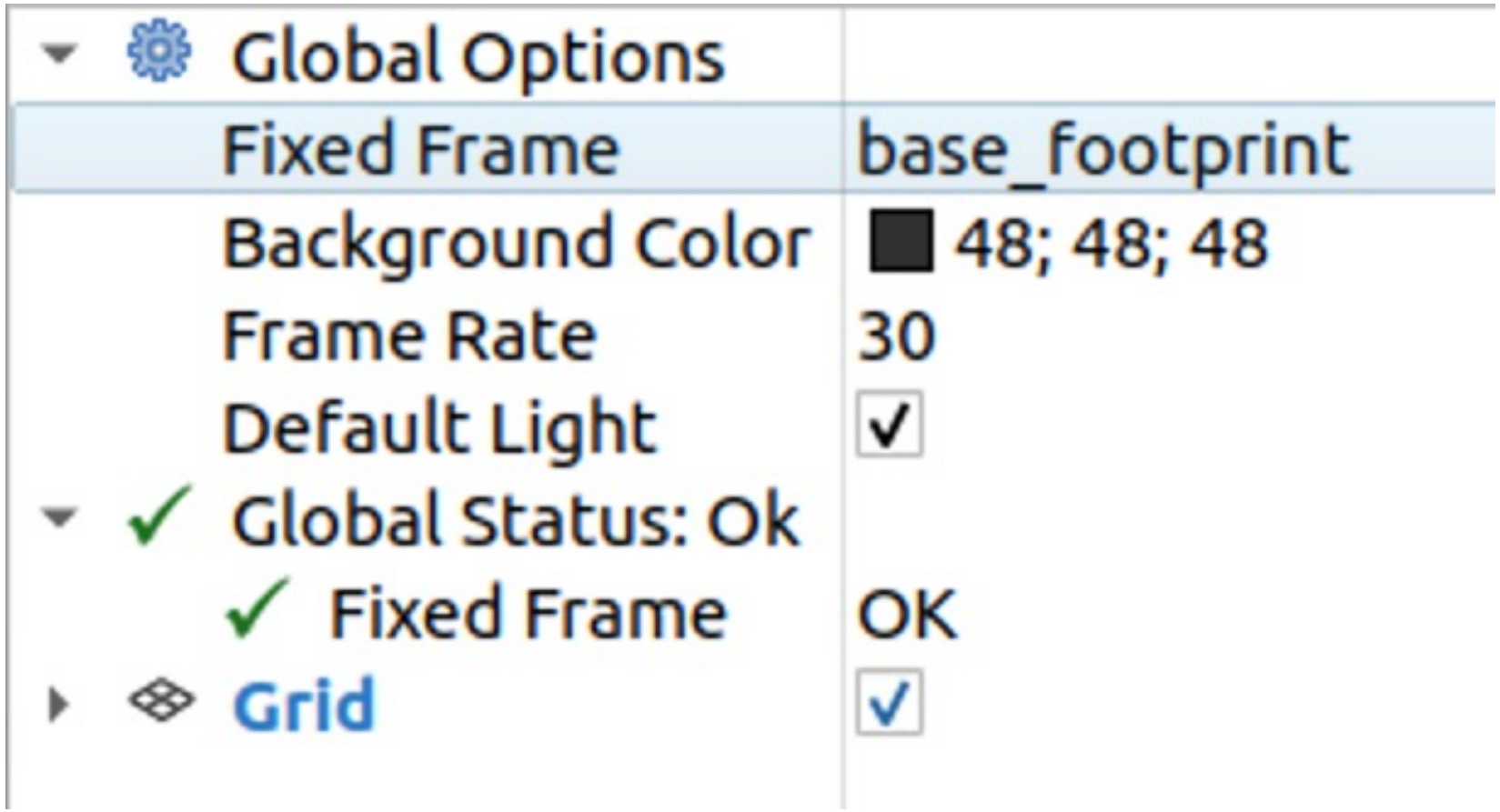
LiDAR – [YDLidar-X2]



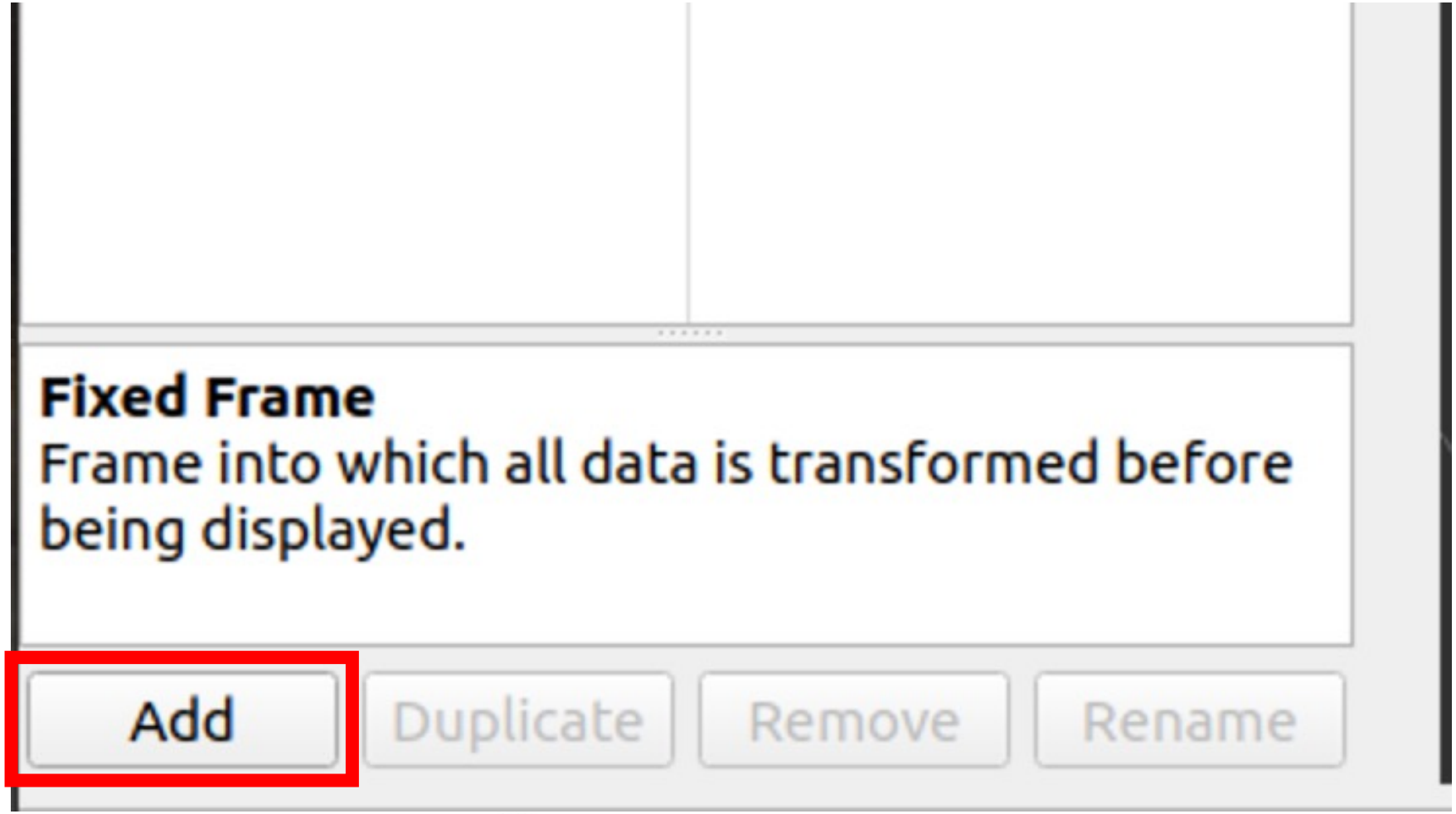
LiDAR – [YDLidar-X2]



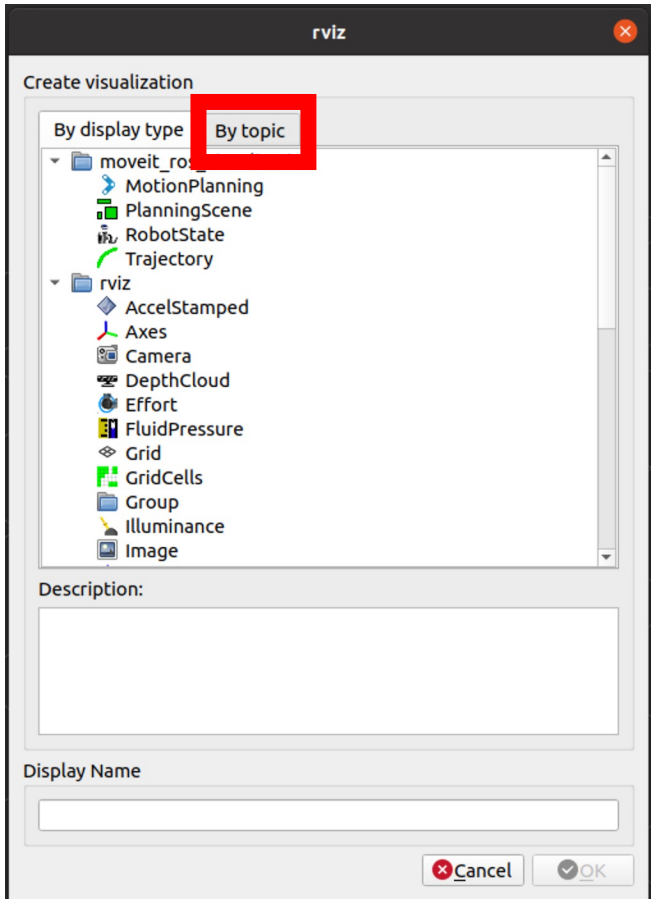
LiDAR – [YDLidar-X2]



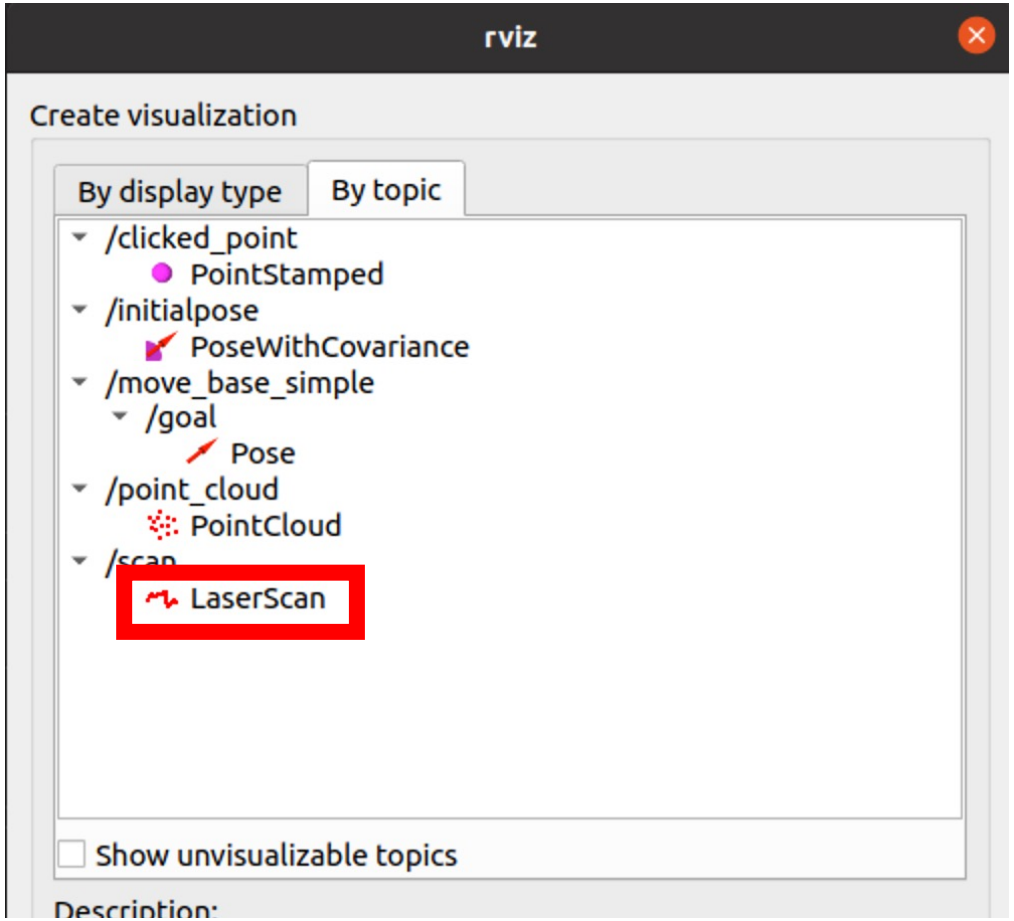
LiDAR – [YDLidar-X2]



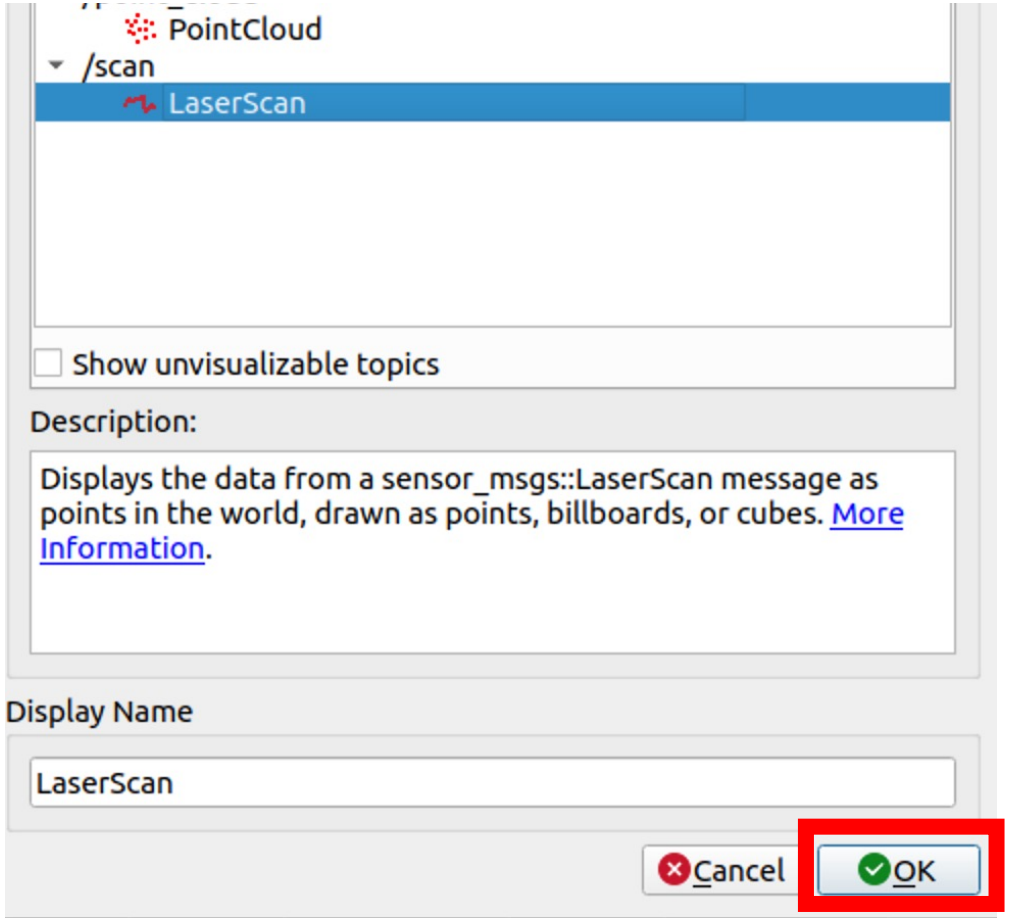
LiDAR – [YDLidar-X2]



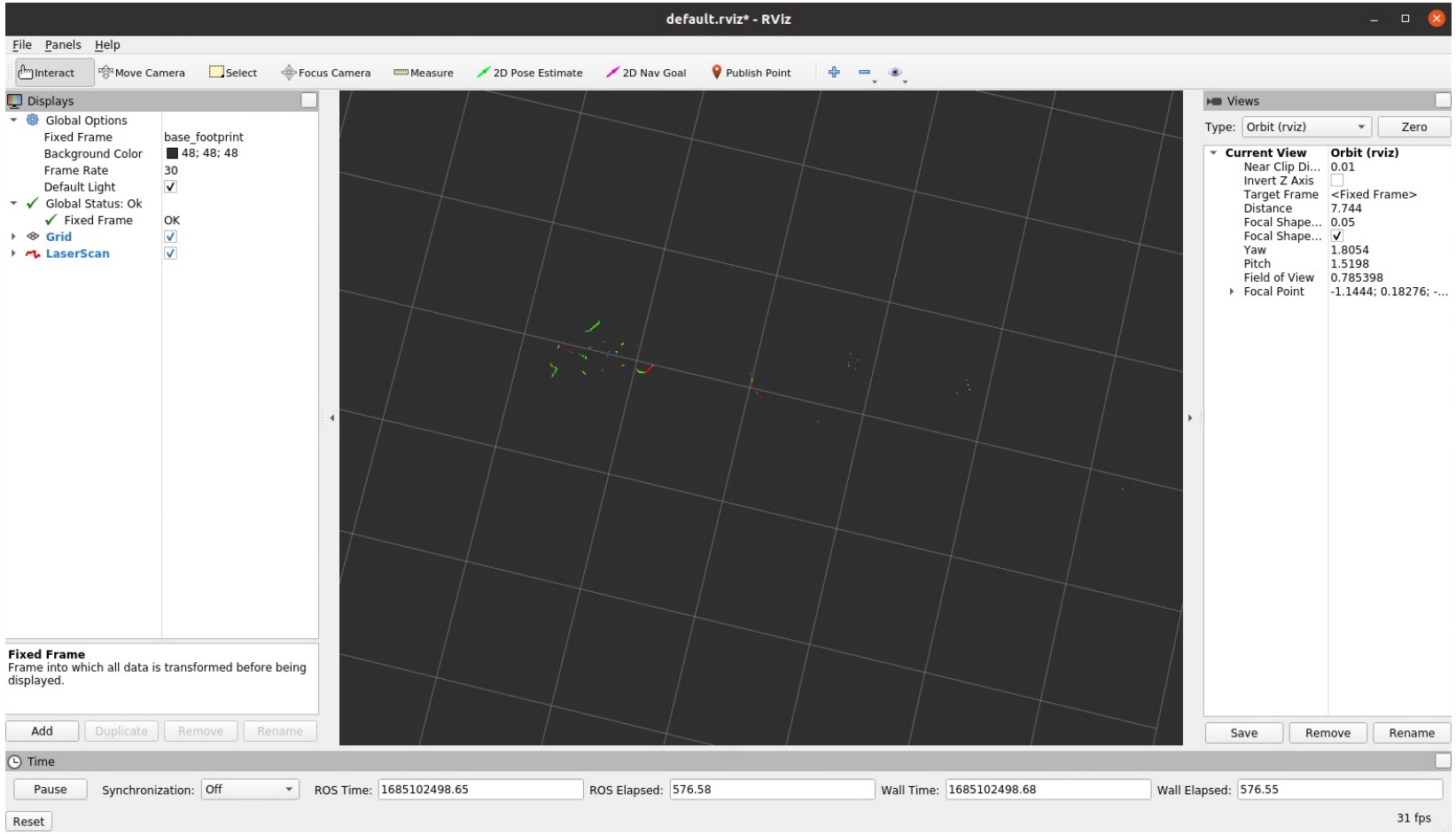
LiDAR – [YDLidar-X2]



LiDAR – [YDLidar-X2]



LiDAR – [YDLidar-X2]



Kinect V.1

How to use kinect with ROS.

Kinect (Depth camera)

```
$ git clone https://github.com/OpenKinect/libfreenect.git
```

```
$ cd libfreenect/
```

```
$ mkdir build && cd build
```

```
$ cmake -L ..
```

Kinect (Depth camera)

```
$ make
```

```
$ sudo make install
```

```
$ sudo ldconfig /usr/local/lib64/
```

Kinect (Depth camera)

```
$ cd ..
```

```
$ python3 src/fwfetcher.py
```

```
pat@pat:~/libfreenect$ python3 src/fwfetcher.py
Downloading SystemUpdate.zip from https://www.xbox.com/system-update-usb
Reading response...
done, saved to SystemUpdate.zip
Extracting $SystemUpdate/FFFE07DF00000001 from system update file...
done.
Handling LIVE/PIRS file.
Writing information file FFFE07DF00000001.txt
Creating and filling content directory FFFE07DF00000001.dir
Moving audios.bin to current folder
Cleaning up
Done!
```

Kinect (Depth camera)

```
$ sudo cp audios.bin /usr/local/share/libfreenect
```

Kinect (Depth camera)

```
$ sudo nano /etc/udev/rules.d/51-kinect.rules
```

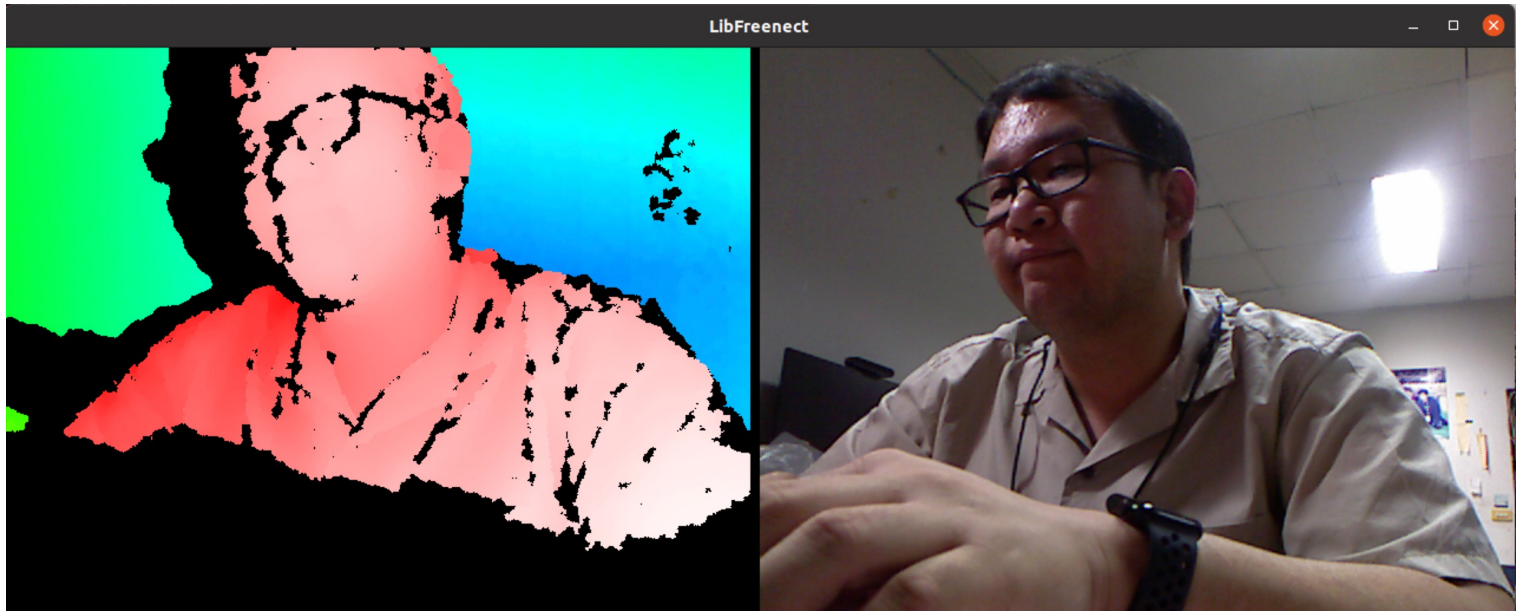
```
"Xbox NUI Motor"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02b0", MODE="0666"
# ATTR{product}=="Xbox NUI Audio"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ad", MODE="0666"
# ATTR{product}=="Xbox NUI Camera"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02ae", MODE="0666"
# ATTR{product}=="Xbox NUI Motor"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02c2", MODE="0666"
# ATTR{product}=="Xbox NUI Motor"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02be", MODE="0666"
# ATTR{product}=="Xbox NUI Motor"
SUBSYSTEM=="usb", ATTR{idVendor}=="045e", ATTR{idProduct}=="02bf", MODE="0666"
```

Kinect (Depth camera)

```
$ sudo reboot
```

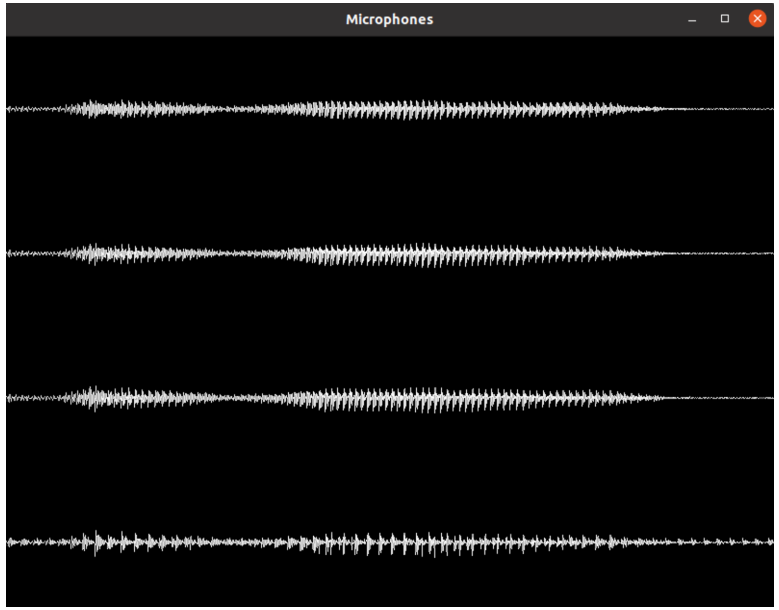
Kinect (Depth camera)

```
$ freenect-glview
```



Kinect (Depth camera)

```
$ freenect-micview
```



Kinect (Depth camera)

```
$ roscd
```

```
$ cd ../src
```

```
$ git clone https://github.com/ros-drivers/freenect_stack.git
```

Kinect (Depth camera)

```
$ cd ../
```

```
$ catkin_make
```

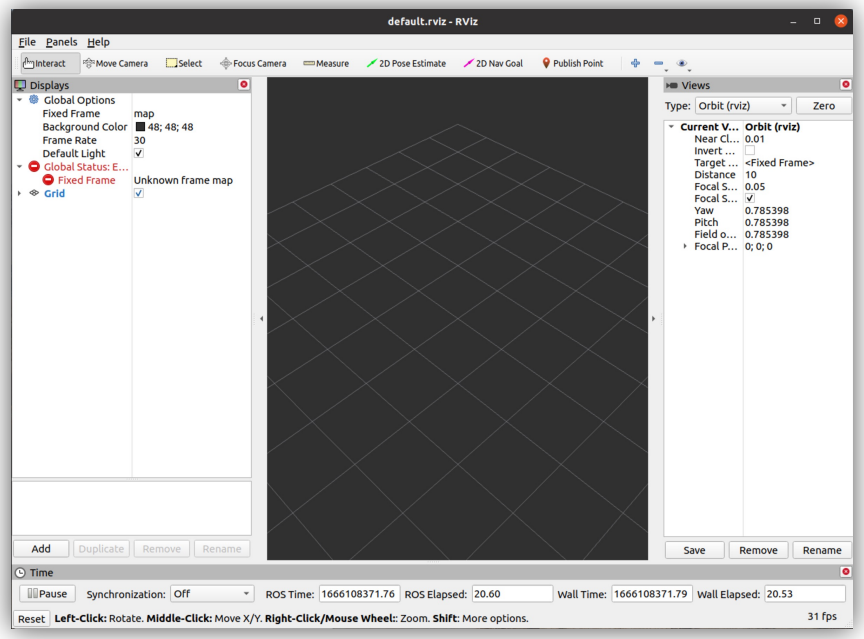
Kinect (Depth camera)

```
$ rospack profile
```

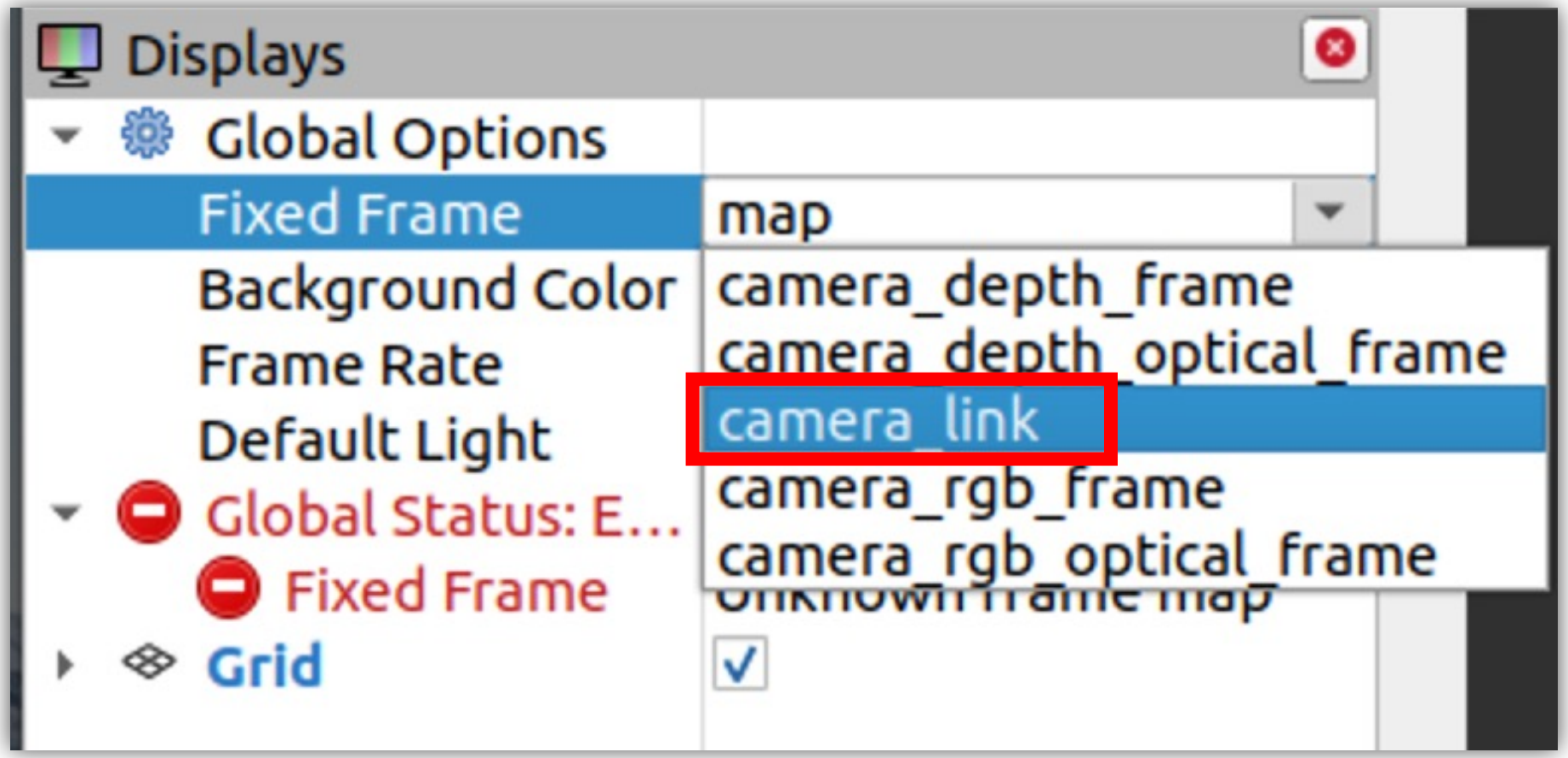
```
$ roslaunch freenect_launch freenect.launch depth_registration:=true
```

Kinect (Depth camera)

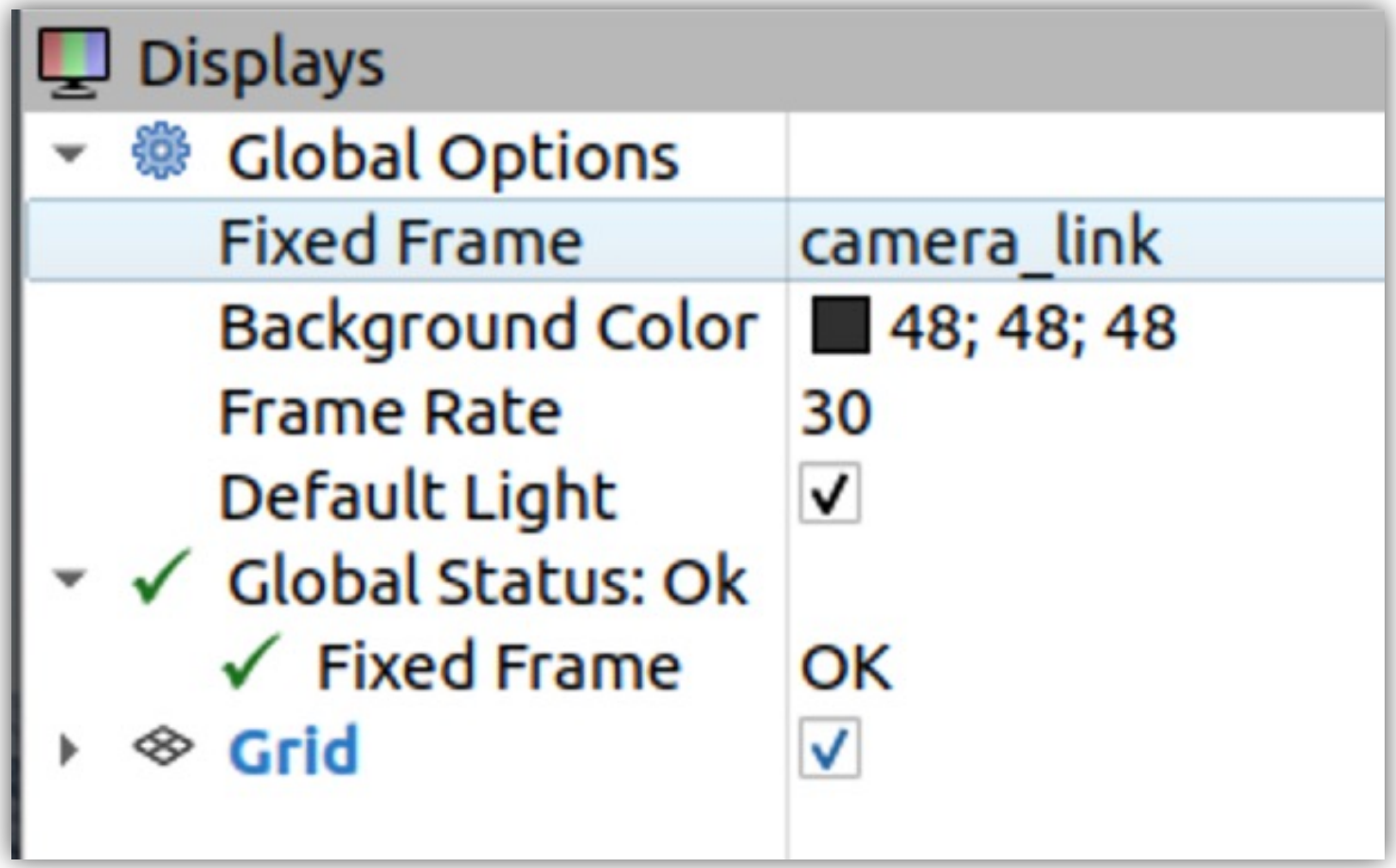
```
$ rviz
```



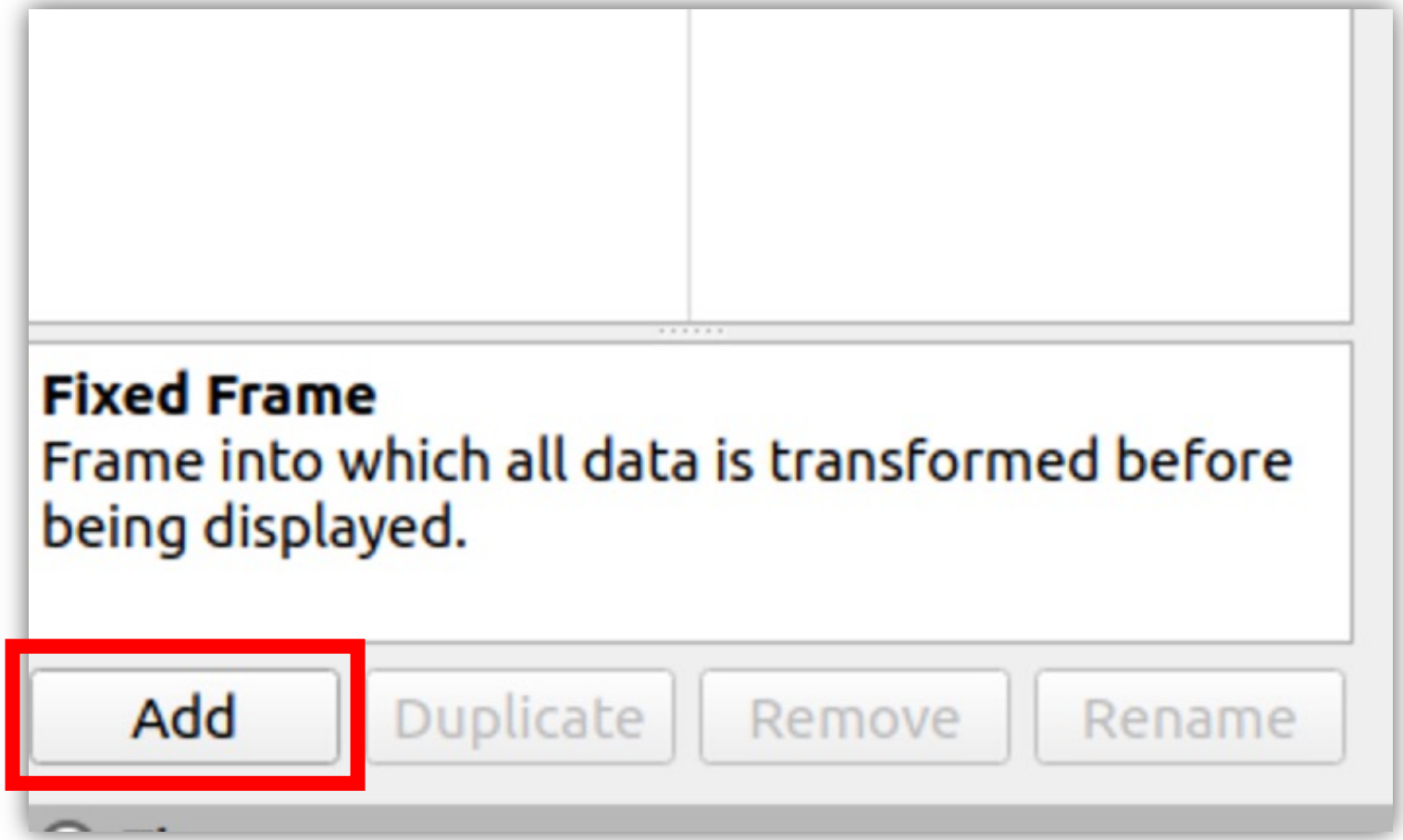
Kinect (Depth camera)



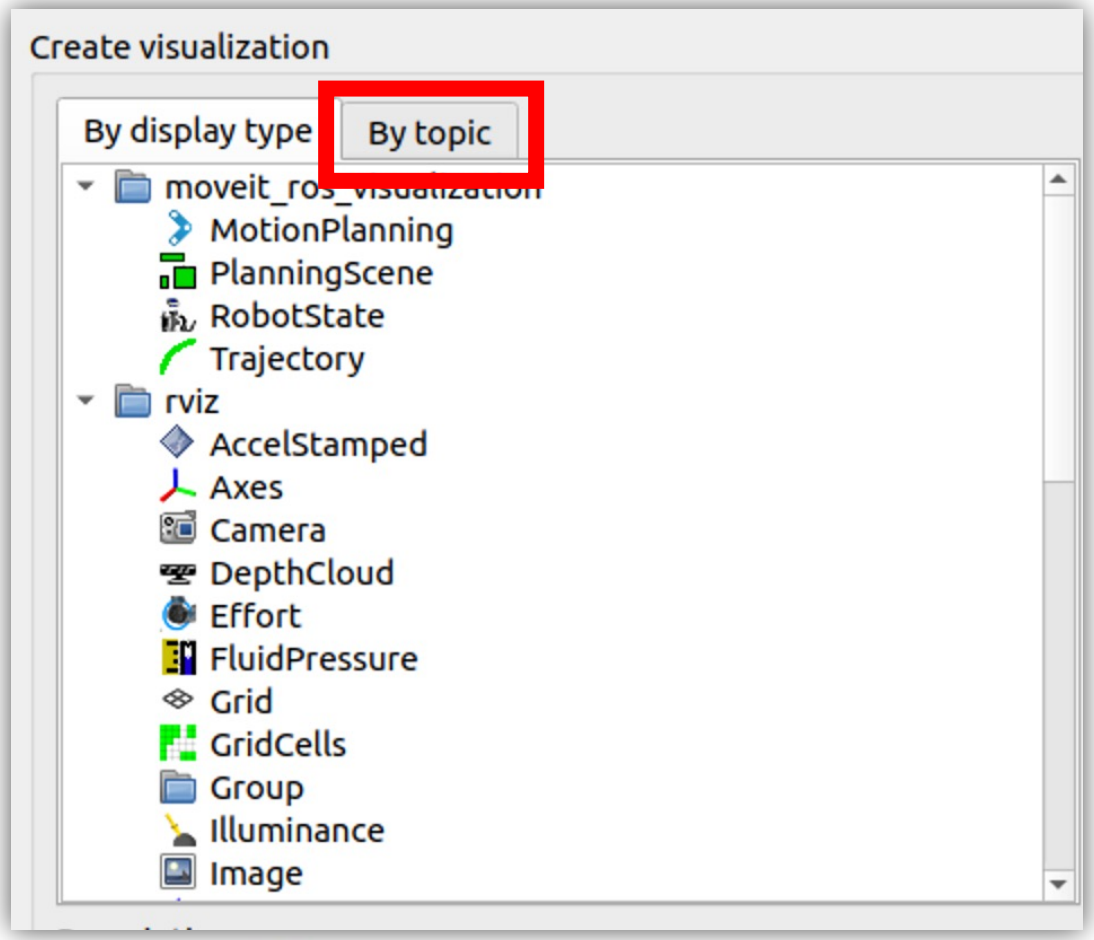
Kinect (Depth camera)



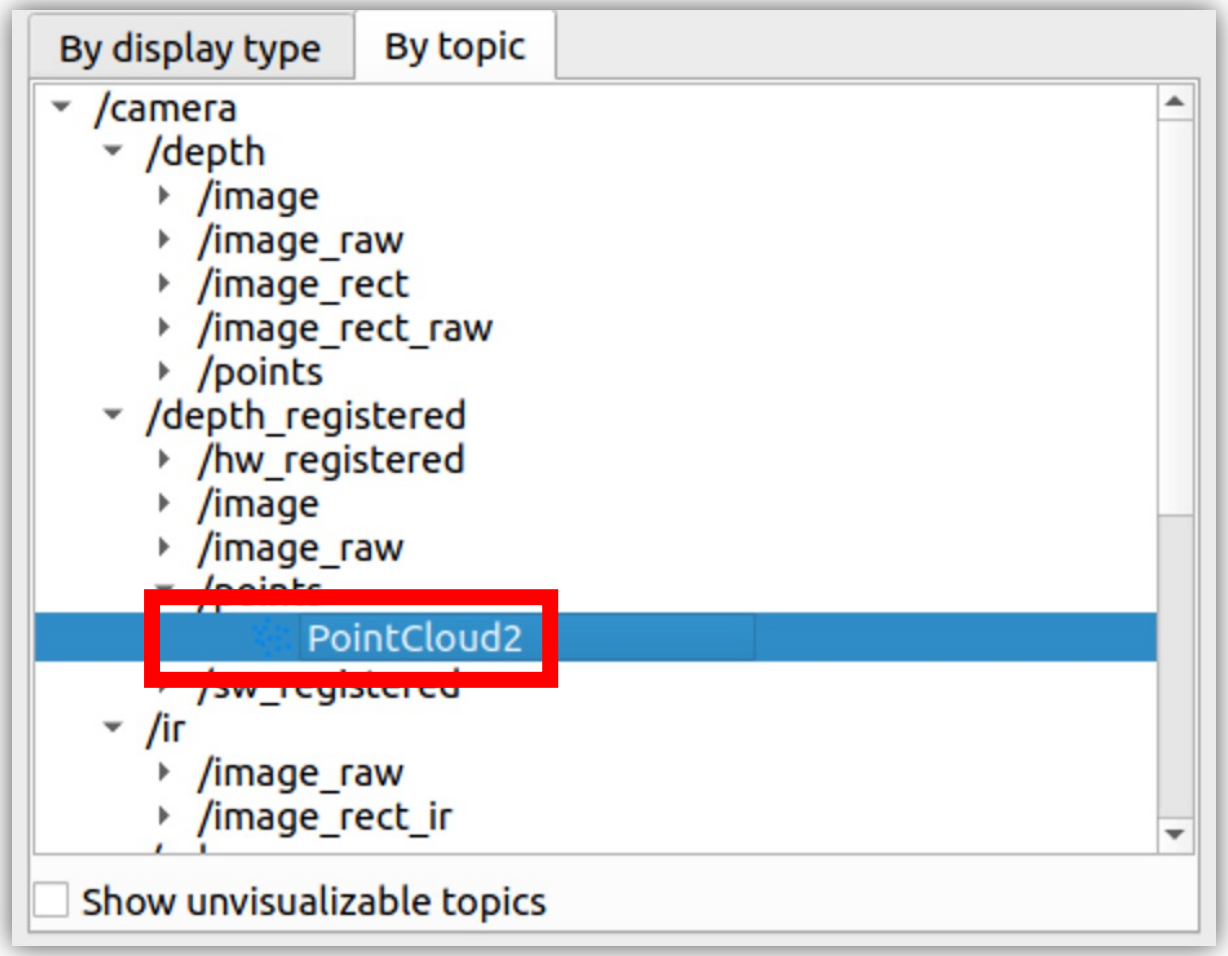
Kinect (Depth camera)



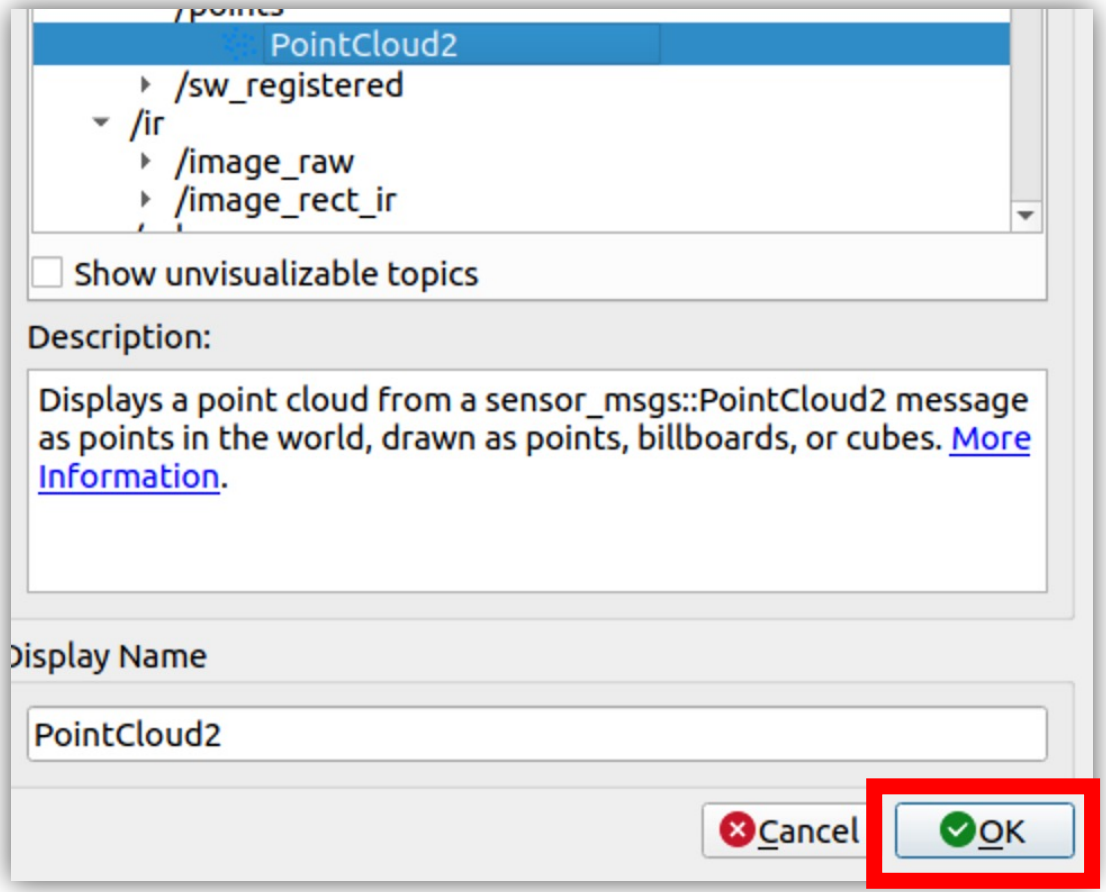
Kinect (Depth camera)



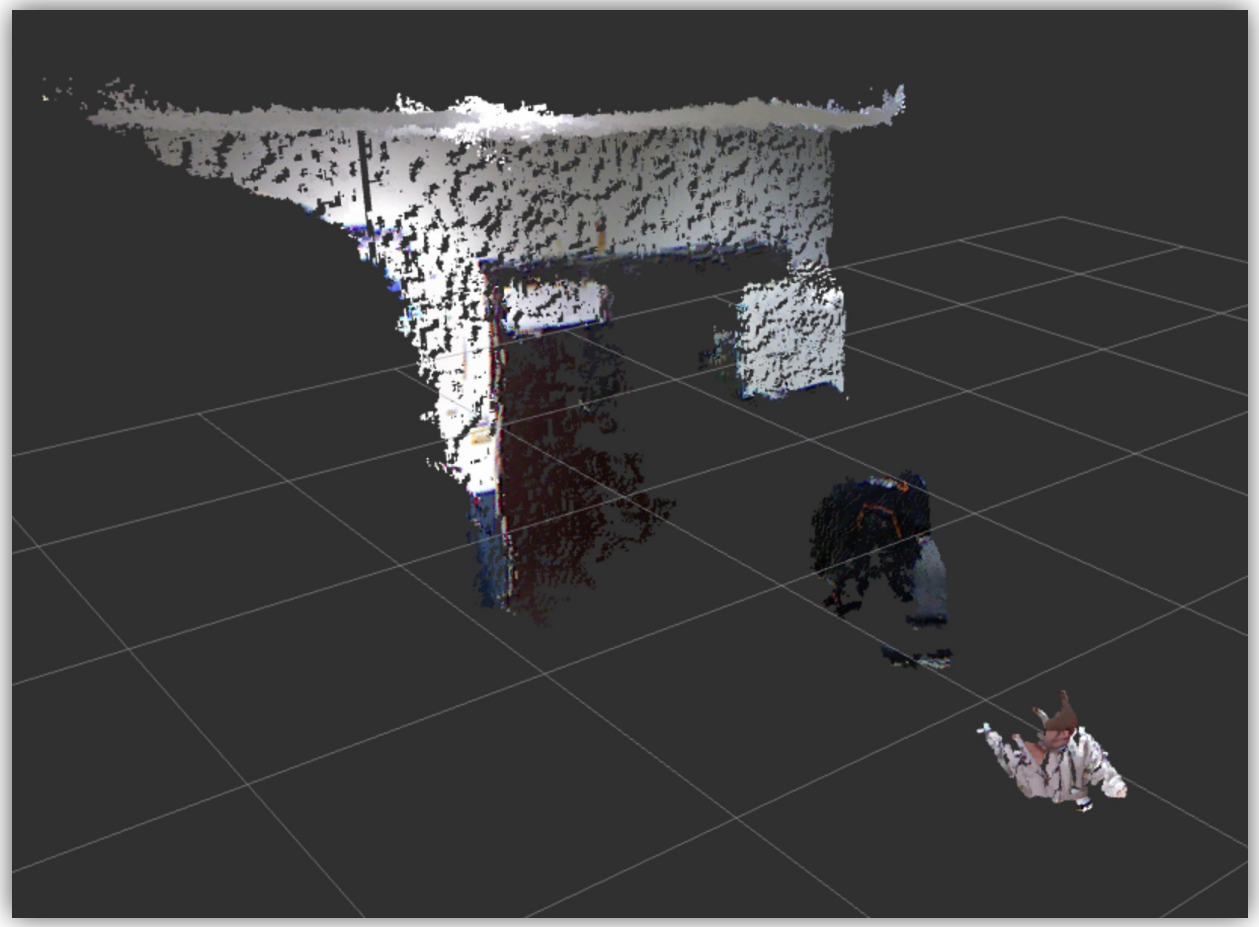
Kinect (Depth camera)



Kinect (Depth camera)

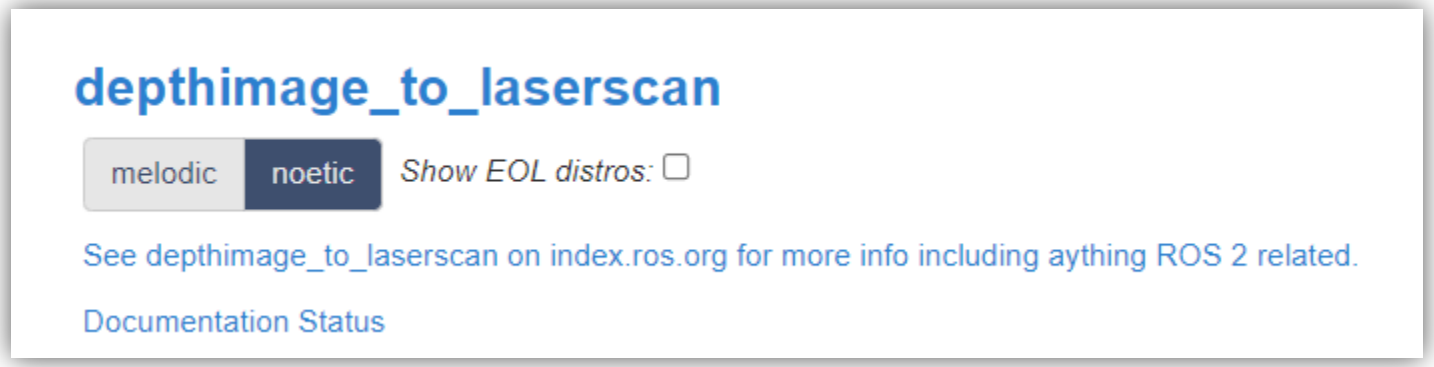


Kinect (Depth camera)



Kinect (Depth camera)

Convert data from Kinect to LaserScan

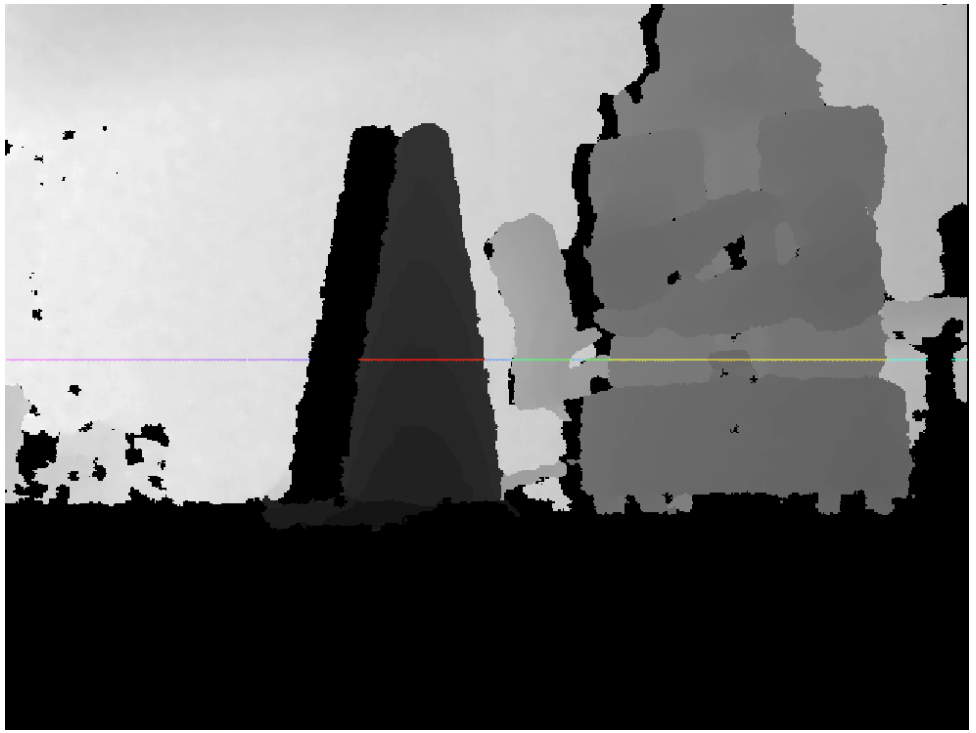



```
$ sudo apt install ros-noetic-depthimage-to-laserscan
```

```
$ sudo apt install ros-noetic-openni-launch
```

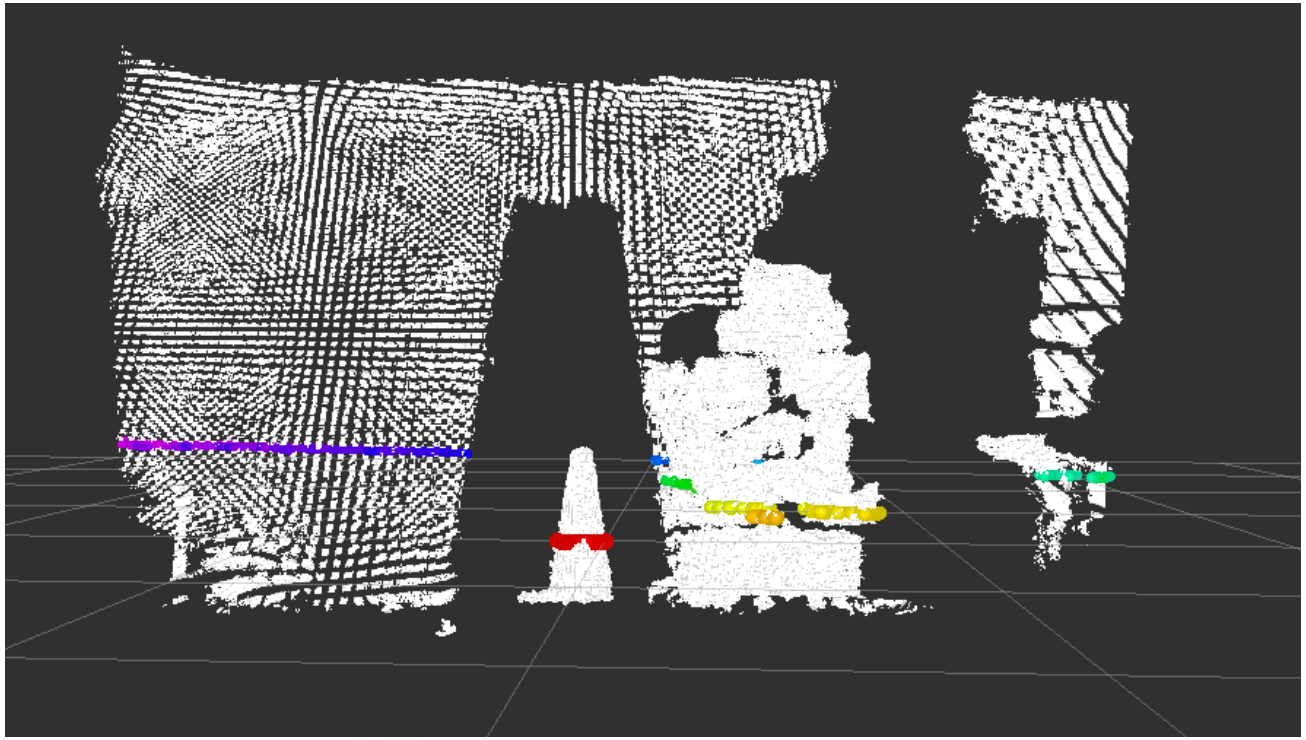
Kinect (Depth camera)

Convert data from Kinect to LaserScan



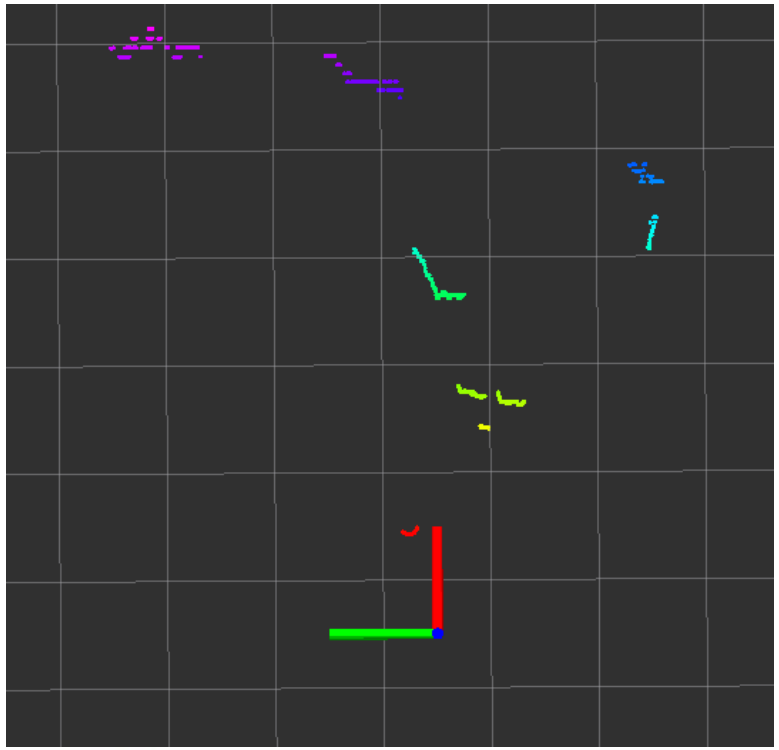
Kinect (Depth camera)

Convert data from Kinect to LaserScan



Kinect (Depth camera)

Convert data from Kinect to LaserScan



Kinect (Depth camera)

Convert data from Kinect to LaserScan

- Create launch directory

```
$ roscd your_package
```

```
$ mkdir launch && cd launch
```

```
$ gedit kinect_depth_process.launch
```

kinect_depth_process.launch

Kinect (Depth camera)

Convert data from Kinect to LaserScan

- Save and exit the gedit
- Launch the launch file

```
$ roslaunch your_package kinect_depth_process.launch
```

- Open RVIZ

```
$ rviz
```

Feature extraction

How to extract the feature from sensor data.

Camera calibration

How to calibrate camera.

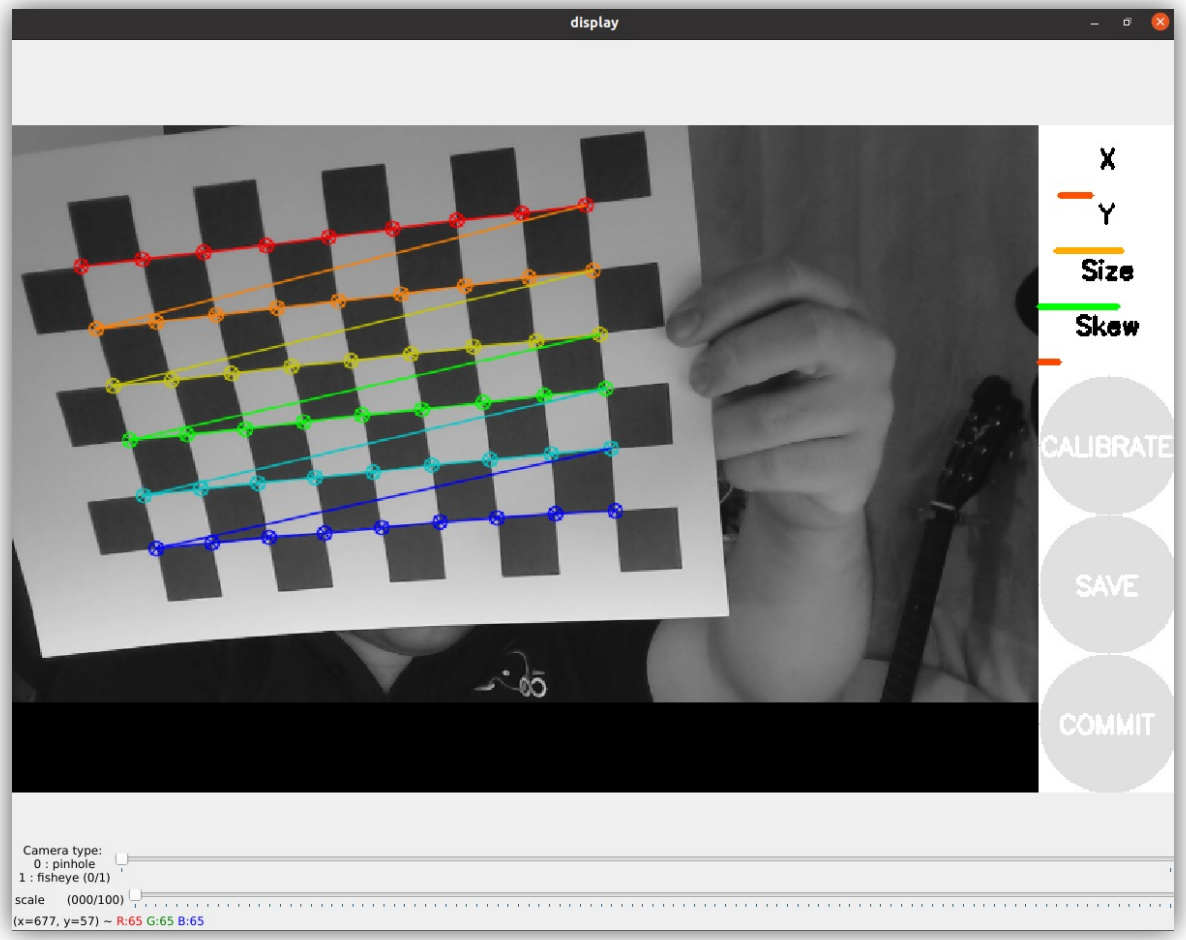
Camera calibration

```
$ sudo apt install ros-noetic-camera-calibration
```

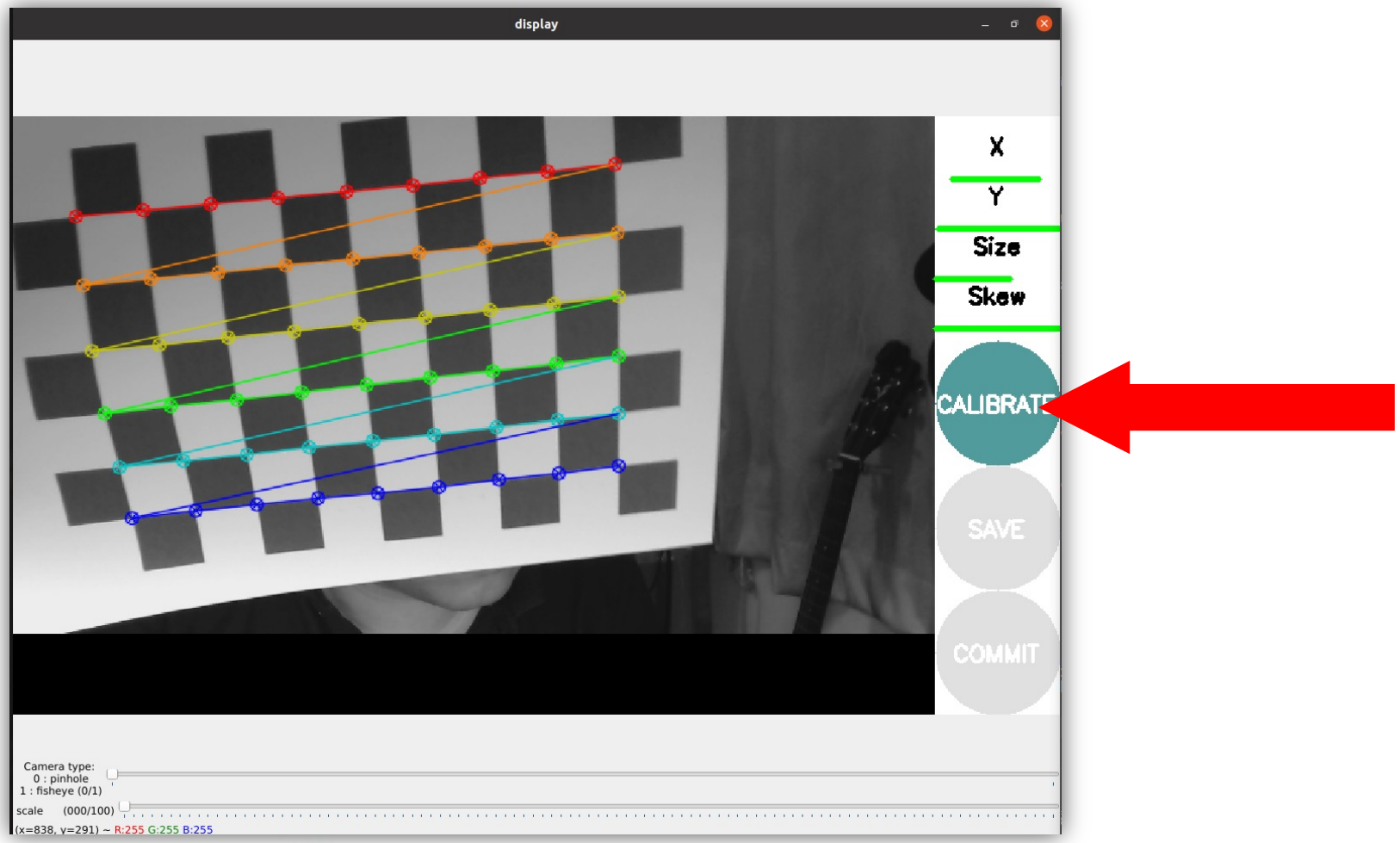
```
$ roslaunch usb_cam usb_cam-test.launch
```

```
$ rosrun camera_calibration cameracalibrator.py --size 9x6 --
square 0.25 image:=/usb_cam/image_raw camera:=/usb_cam
```

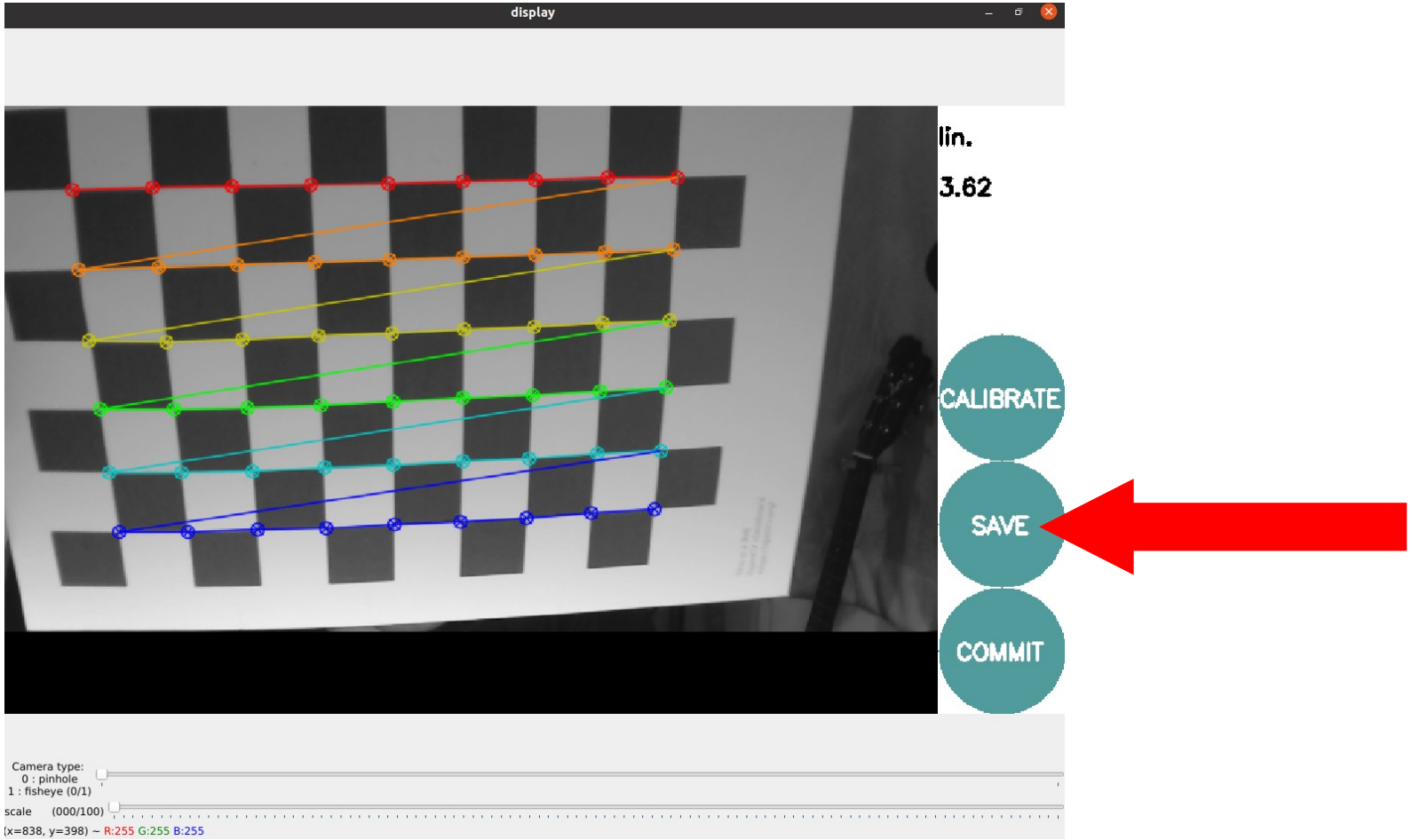
Camera calibration



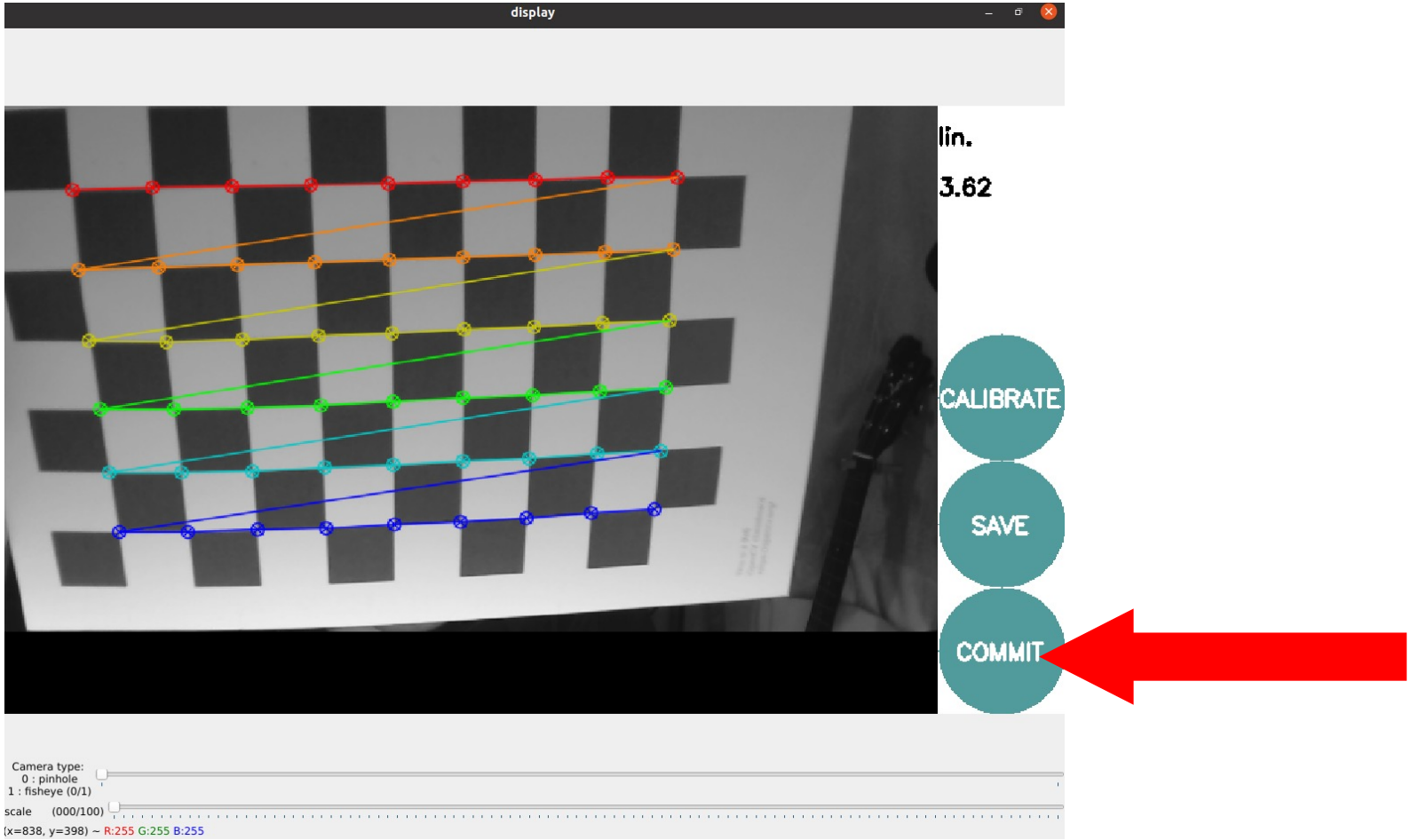
Camera calibration



Camera calibration



Camera calibration



Camera calibration

```
$ cat .ros/camera_info/head_camera.yaml
```

```
nptttn@pat:~$ cat .ros/camera_info/head_camera.yaml
image_width: 1280
image_height: 720
camera_name: head_camera
camera_matrix:
  rows: 3
  cols: 3
  data: [1381.599355165518, 0, 675.4006422618129, 0, 1400.345875470718, 272.7167179460779, 0, 0, 1]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [0.1043850965083147, 0.09886973571653765, 0.002716006997294545, 0.02802969367713559, 0]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1, 0, 0, 0, 1, 0, 0, 0, 1]
projection_matrix:
  rows: 3
  cols: 4
  data: [1428.76318359375, 0, 697.5875599100691, 0, 0, 1474.112670898438, 274.8183526258872, 0, 0, 0, 1, 0]
```

Camera calibration

Ctrl+c in all windows

ORB SLAM

Using monocular camera to make SLAM.

ORB SLAM

```
$ roscd
```

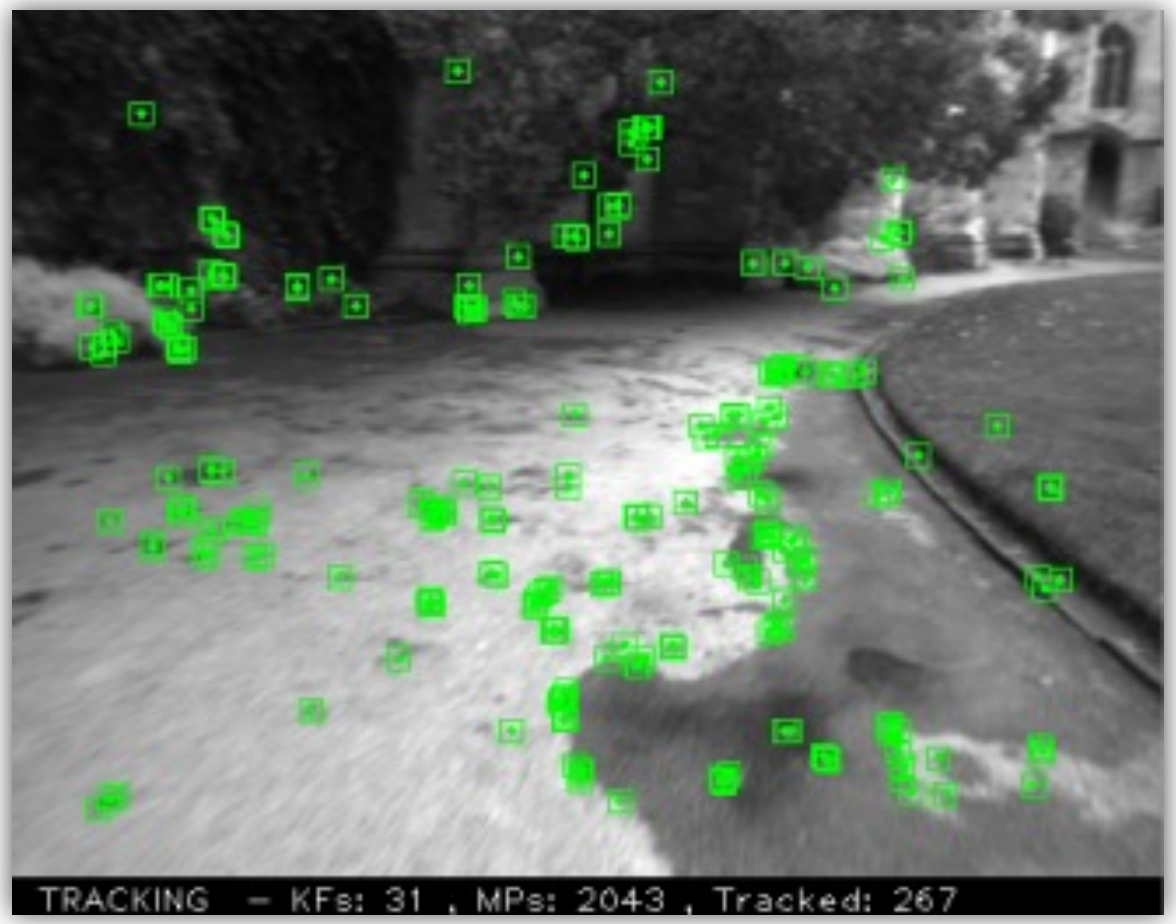
```
$ cd ../src
```

```
$ git clone https://github.com/appliedAI-Initiative/orb_slam_2_ros
```

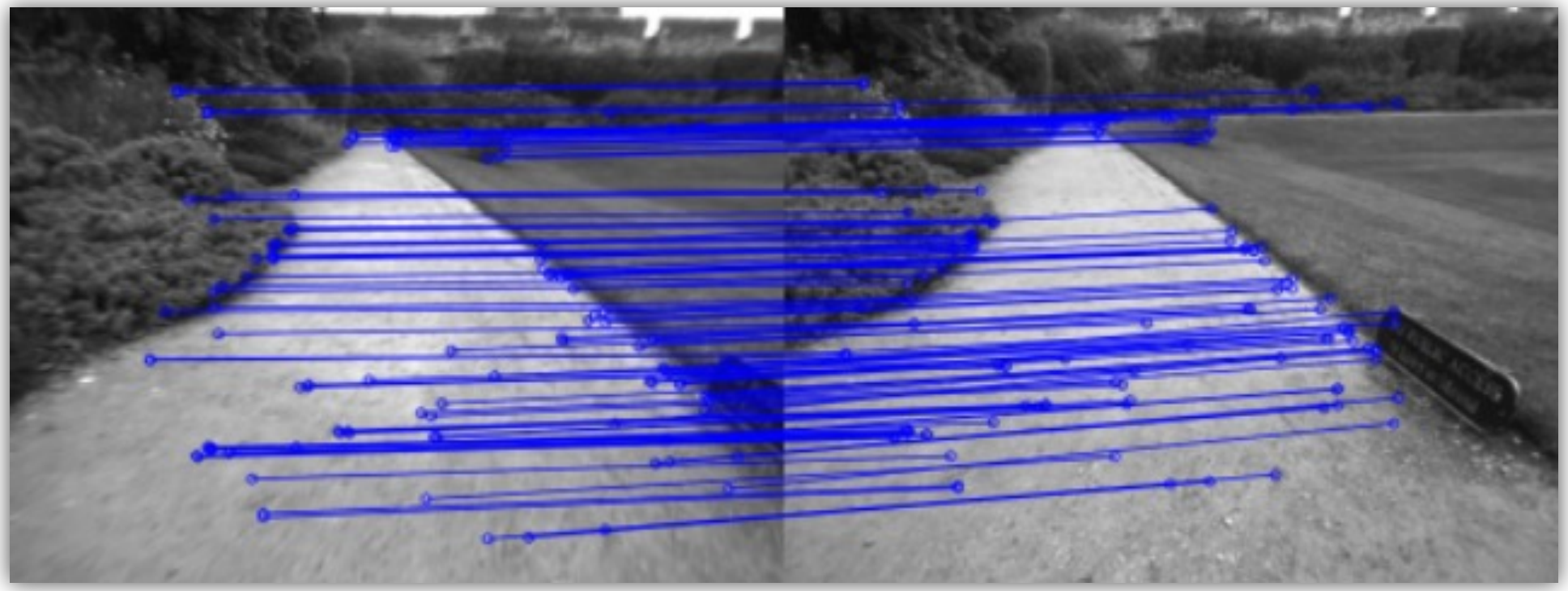
```
$ cd ..
```

```
$ catkin_make
```

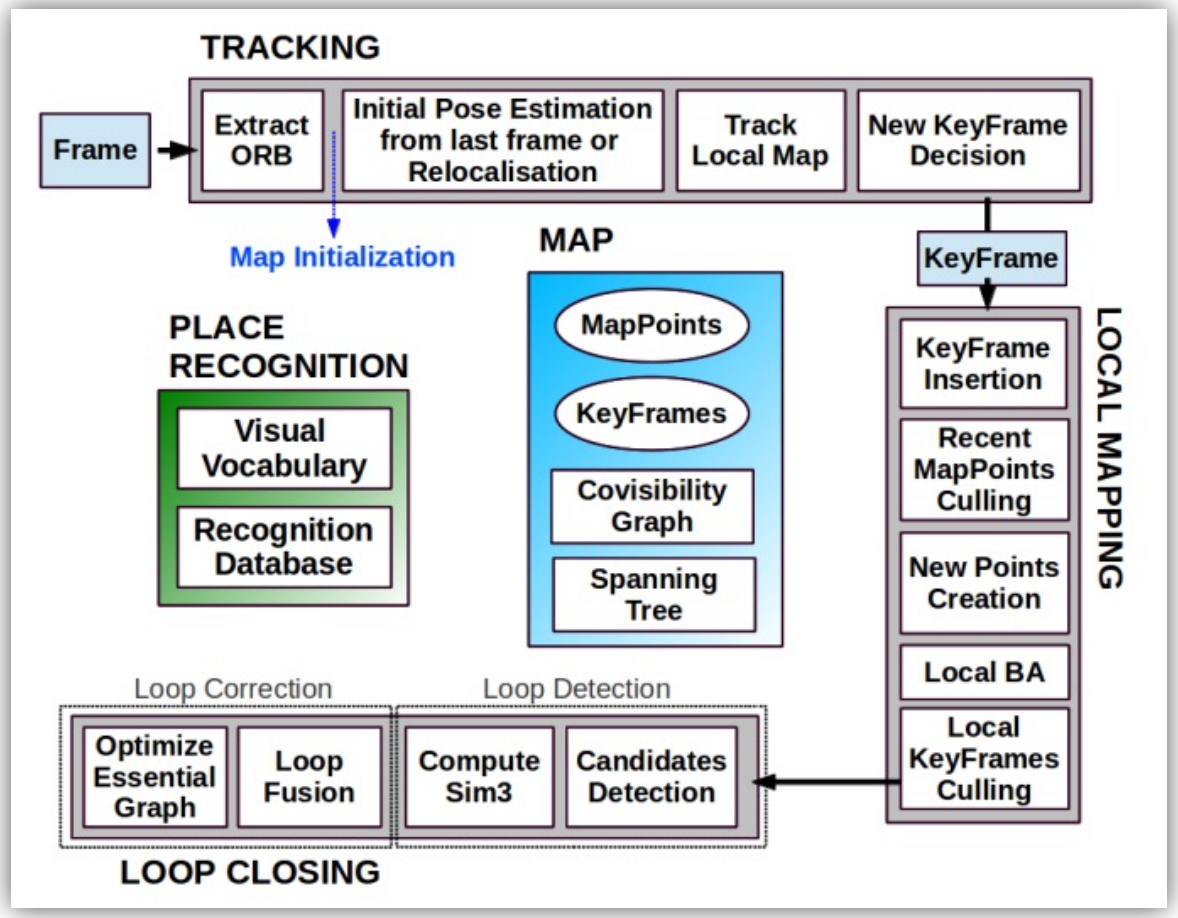
ORB SLAM



ORB SLAM



ORB SLAM


TRACKING
Frame
Extract ORB
Initial Pose Estimation from last frame or Relocalisation
Track Local Map
New KeyFrame Decision
Map Initialization
MAP
KeyFrame
PLACE RECOGNITION
Visual Vocabulary
Recognition Database
MapPoints
KeyFrames
Covisibility Graph
Spanning Tree
KeyFrame Insertion
Recent MapPoints Culling
New Points Creation
Local BA
Local KeyFrames Culling
LOCAL MAPPING
Loop Correction
Loop Detection
Optimize Essential Graph
Loop Fusion
Compute Sim3
Candidates Detection
LOOP CLOSING

ORB SLAM

- Create launch file in your_package

```
$ roscd your_package/launch
```

```
$ gedit test_mono_orb.launch
```

test_mono_orb.launch

- Save and exit

ORB SLAM

- Open USB_CAM

```
$ roslaunch your_package usb-cam.launch
```

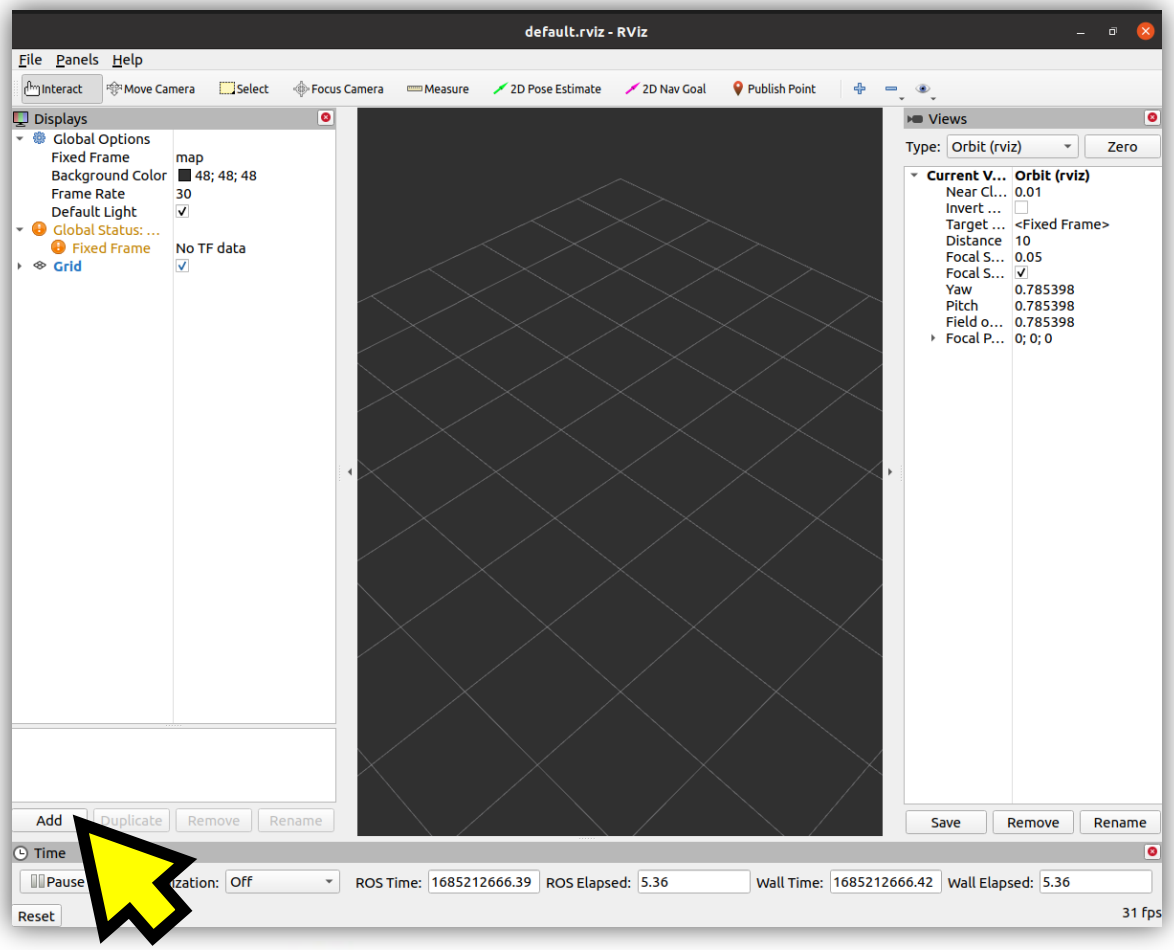
- Open ORB SLAM file

```
$ roslaunch your_package test_mono_orb.launch
```

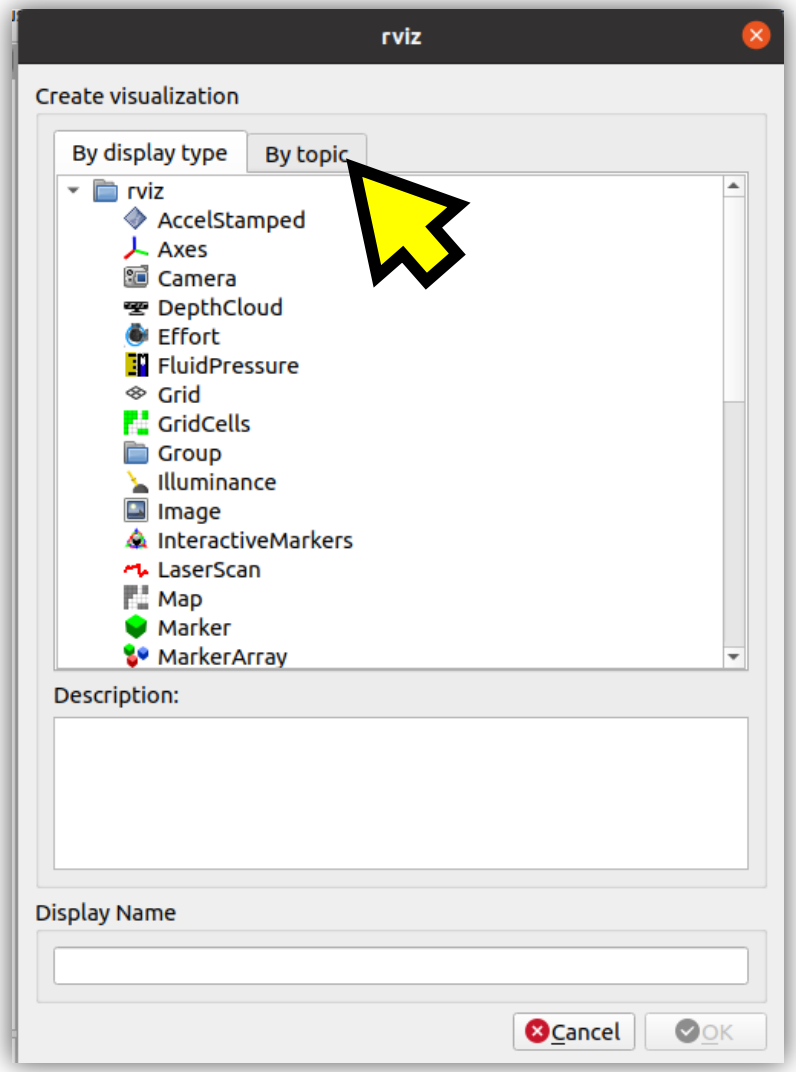
- Open RVIZ

```
$ rviz
```

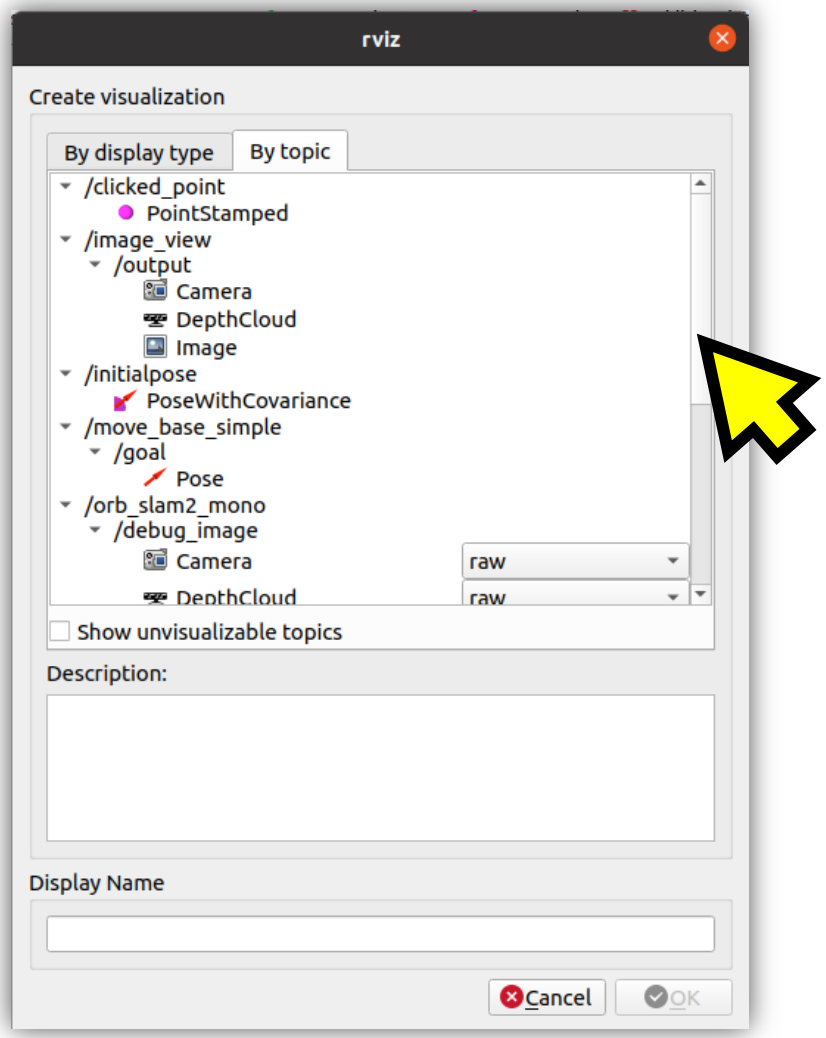
ORB SLAM


default.rviz - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: ...
Fixed Frame No TF data
Grid
Add Duplicate Remove Rename
Views
Type: Orbit (rviz) Zero
Current V... Orbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 10
Focal S... 0.05
Focal S...
Yaw 0.785398
Pitch 0.785398
Field o... 0.785398
Focal P... 0; 0; 0
Save Remove Rename
Time
Pause zation: Off ROS Time: 1685212666.39 ROS Elapsed: 5.36 Wall Time: 1685212666.42 Wall Elapsed: 5.36
Reset
31 fps

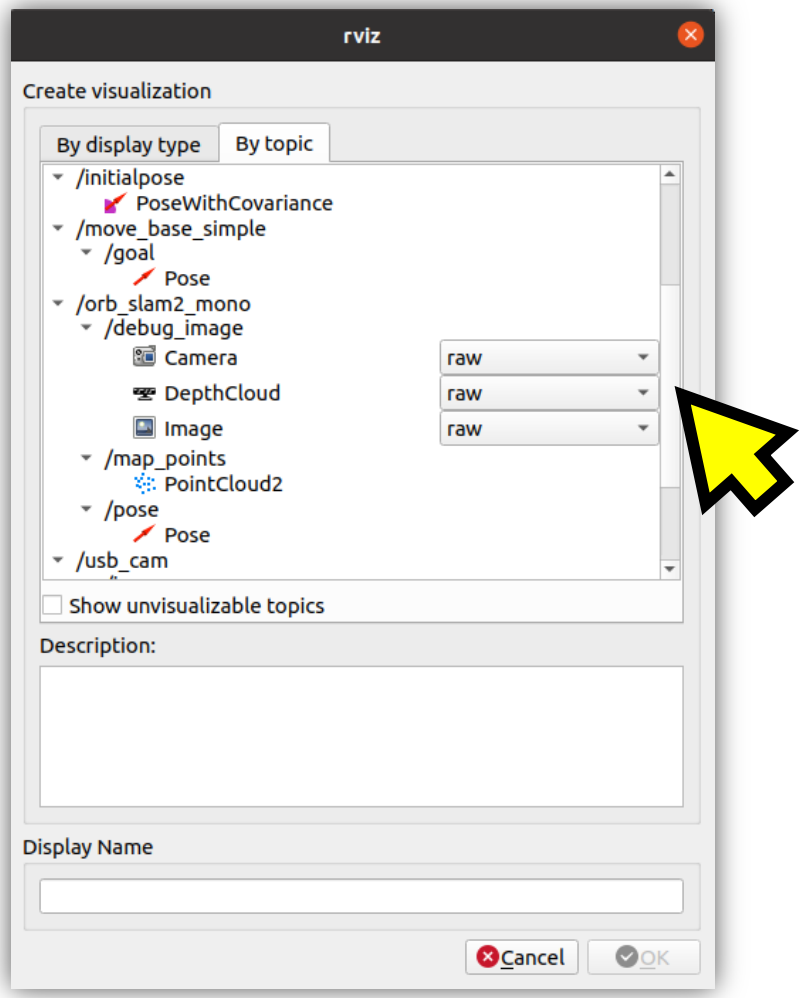
ORB SLAM


rviz
Create visualization
By display type
By topic
rviz
AccelStamped
Axes
Camera
DepthCloud
Effort
FluidPressure
Grid
GridCells
Group
Illuminance
Image
InteractiveMarkers
LaserScan
Map
Marker
MarkerArray
Description:
Display Name
Cancel
OK

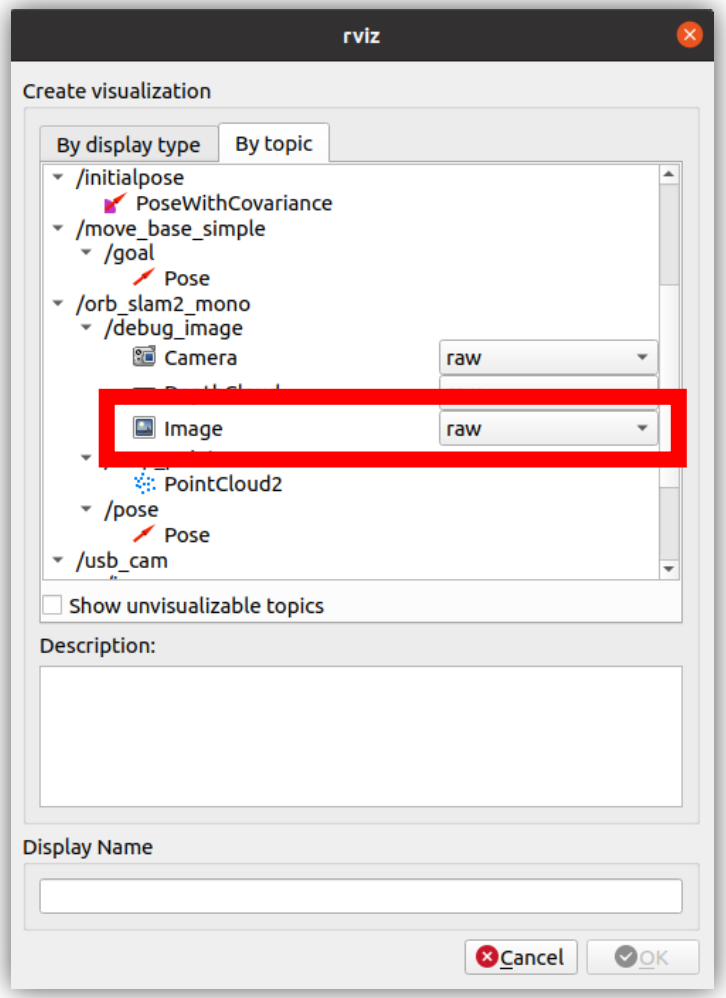
ORB SLAM

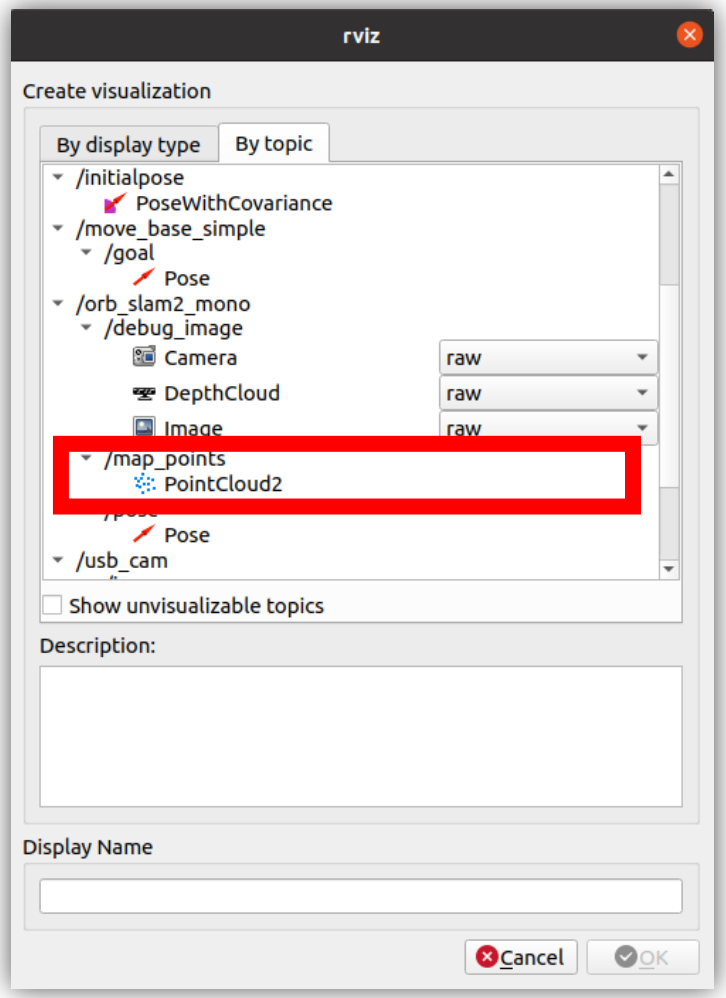

rviz
Create visualization
By display type
By topic
/clicked_point
PointStamped
/image_view
/output
Camera
DepthCloud
Image
/initialpose
PoseWithCovariance
/move_base_simple
/goal
Pose
/orb_slam2_mono
/debug_image
Camera
raw
DepthCloud
raw
Show unvisualizable topics
Description:
Display Name
Cancel
OK

ORB SLAM

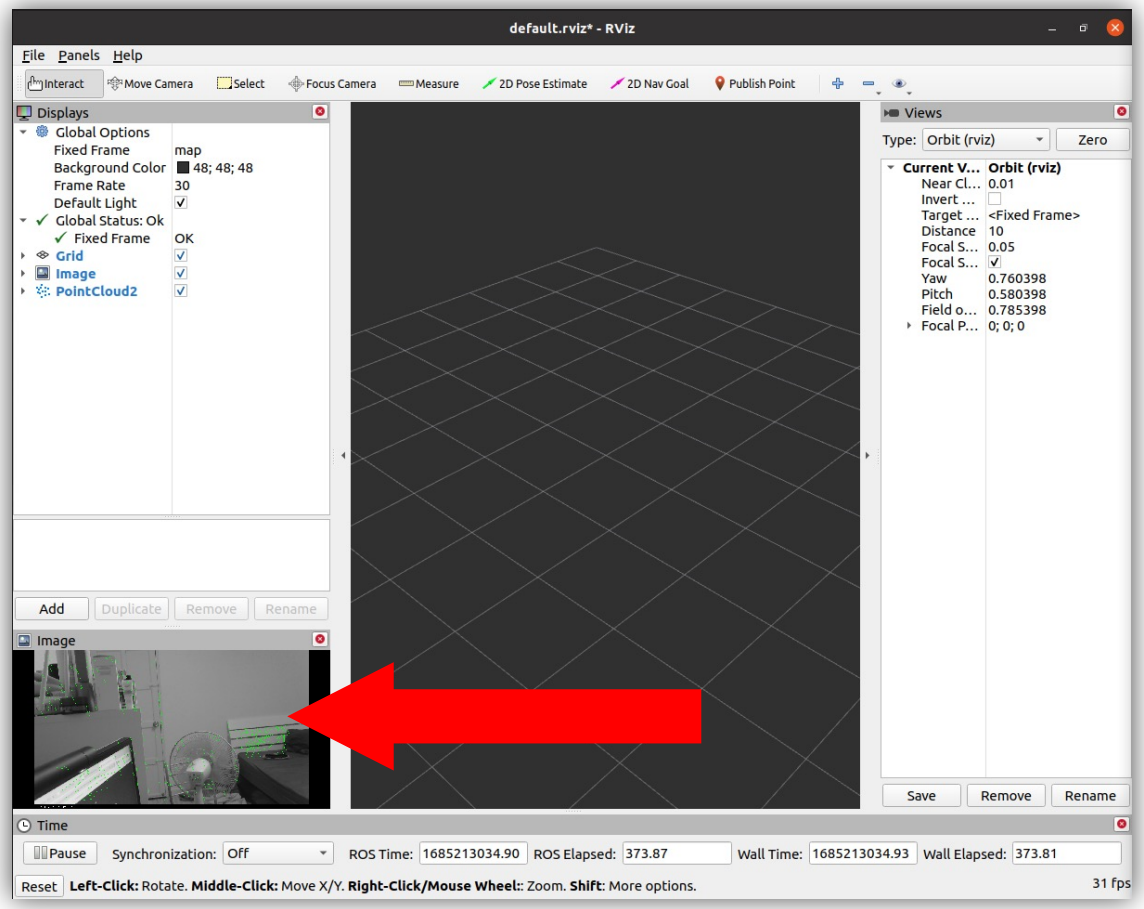

rviz
Create visualization
By display type
By topic
/initialpose
PoseWithCovariance
/move_base_simple
/goal
Pose
/orb_slam2_mono
/debug_image
Camera
raw
DepthCloud
raw
Image
raw
/map_points
PointCloud2
/pose
Pose
/usb_cam
Show unvisualizable topics
Description:
Display Name
Cancel
OK

ORB SLAM


rviz
Create visualization
By display type
By topic
/initialpose
PoseWithCovariance
/move_base_simple
/goal
Pose
/orb_slam2_mono
/debug_image
Camera
raw
Image
raw
PointCloud2
/pose
Pose
/usb_cam
Show unvisualizable topics
Description:
Display Name
Cancel
OK


rviz
Create visualization
By display type
By topic
/initialpose
PoseWithCovariance
/move_base_simple
/goal
Pose
/orb_slam2_mono
/debug_image
Camera
raw
DepthCloud
raw
Image
raw
/map_points
PointCloud2
Pose
/usb_cam
Show unvisualizable topics
Description:
Display Name
Cancel
OK

ORB SLAM


default.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Image
PointCloud2
Add Duplicate Remove Rename
Image
Views
Type: Orbit (rviz) Zero
Current V... Orbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 10
Focal S... 0.05
Focal S...
Yaw 0.760398
Pitch 0.580398
Field o... 0.785398
Focal P... 0; 0; 0
Save Remove Rename
Time
Pause Synchronization: Off ROS Time: 1685213034.90 ROS Elapsed: 373.87 Wall Time: 1685213034.93 Wall Elapsed: 373.81
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

ORB SLAM

```
$ rosservice call /orb_slam2_mono/save_map "name: 'map.bin'"
```

Hector SLAM

How to use hector SLAM with LiDAR

Hector SLAM

```
$ sudo apt install ros-noetic-hector-slam
```

```
$ sudo apt install ros-noetic-hector-slam-launch
```

```
$ sudo apt install ros-noetic-hector-mapping
```

Hector SLAM

- Launch LiDAR
  - If use YDLiDAR X2

```
$ roslaunch ydlidar_ros_driver X2.launch
```

  - If use RPLiDAR A3

```
$ roslaunch rplidar_ros rplidar_a3.launch
```

- Launch hector slam

```
$ roslaunch hector_slam_launch tutorial.launch
```

Hector SLAM

- Edit tutorial.launch

```
$ roscd hector_slam_launch
```

```
$ sudo gedit tutorial.launch
```

- Edit
  - /use_sim_time -> false

Hector SLAM

- Edit for Kinect only!

```
$ roscd hector_mapping
```

```
$ sudo gedit mapping_default.launch
```

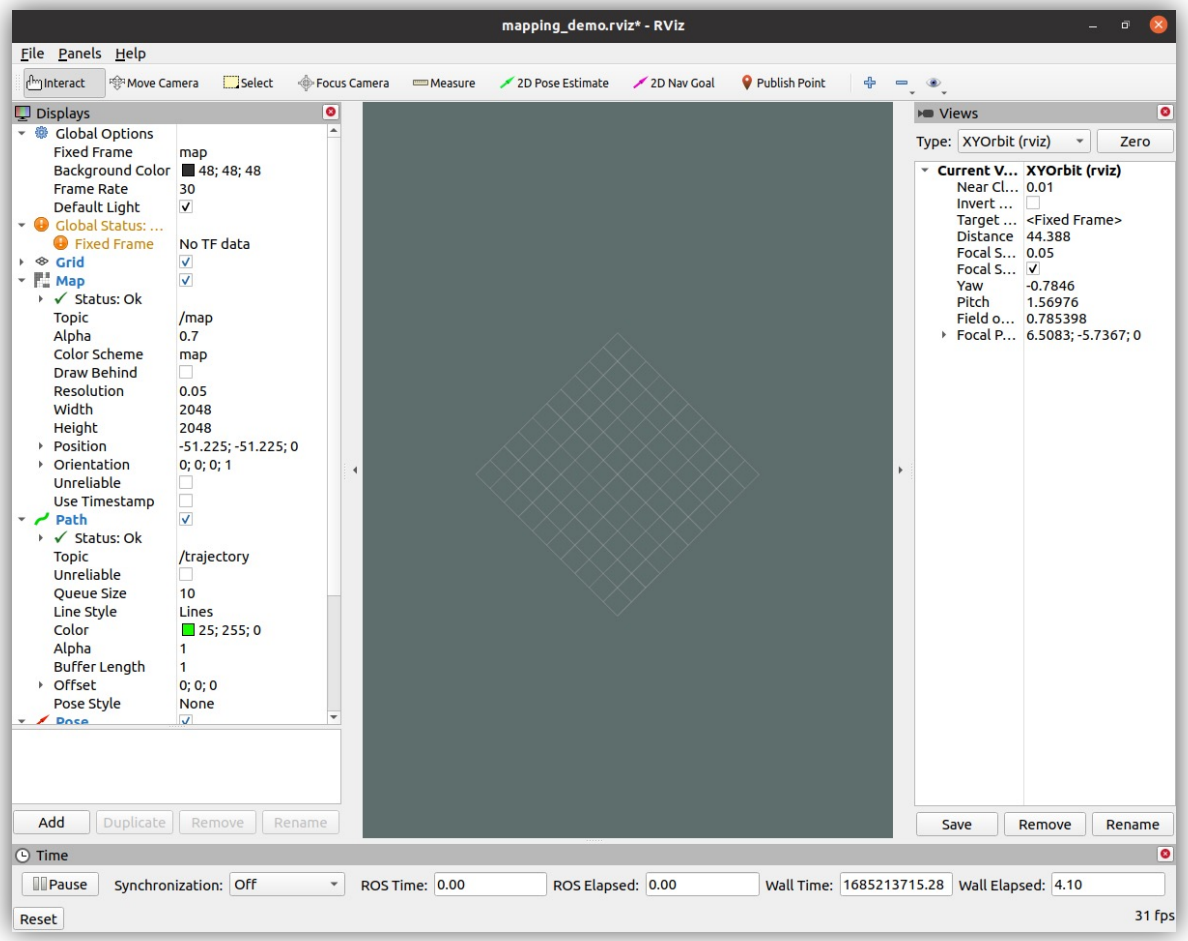
  - <arg name="base_frame" default="base_footprint"/>
    - base_footprint -> camera_link

Hector SLAM

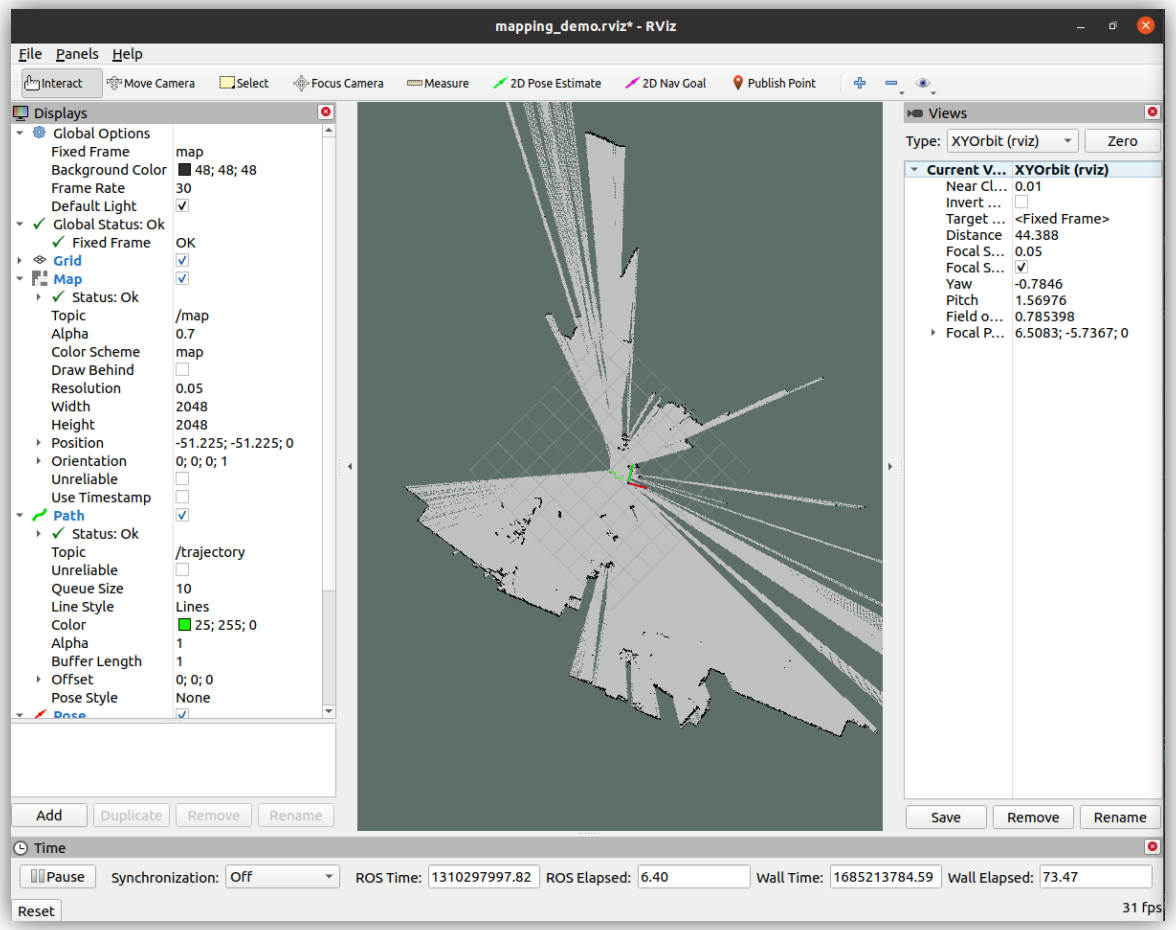
```
$ wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/tu-darmstadt-ros-pkg/Team_Hector_MappingBox_RoboCup_2011_Rescue_Arena.bag
```

```
$ rosbag play Team_Hector_MappingBox_RoboCup_2011_Rescue_Arena.bag --clock
```

Hector SLAM


mapping_demo.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: ...
Fixed Frame No TF data
Grid
Map
Status: Ok
Topic /map
Alpha 0.7
Color Scheme map
Draw Behind
Resolution 0.05
Width 2048
Height 2048
Position -51.225; -51.225; 0
Orientation 0; 0; 0; 1
Unreliable
Use Timestamp
Path
Status: Ok
Topic /trajectory
Unreliable
Queue Size 10
Line Style Lines
Color 25; 255; 0
Alpha 1
Buffer Length 1
Offset 0; 0; 0
Pose Style None
Add
Duplicate
Remove
Rename
Views
Type: XYOrbit (rviz)
Zero
Current V... XYOrbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 44.388
Focal S... 0.05
Focal S...
Yaw -0.7846
Pitch 1.56976
Field o... 0.785398
Focal P... 6.5083; -5.7367; 0
Save
Remove
Rename
Time
Pause
Synchronization: Off
ROS Time: 0.00
ROS Elapsed: 0.00
Wall Time: 1685213715.28
Wall Elapsed: 4.10
Reset
31 fps

Hector SLAM


mapping_demo.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Map
Status: Ok
Topic /map
Alpha 0.7
Color Scheme map
Draw Behind
Resolution 0.05
Width 2048
Height 2048
Position -51.225; -51.225; 0
Orientation 0; 0; 0; 1
Unreliable
Use Timestamp
Path
Status: Ok
Topic /trajectory
Unreliable
Queue Size 10
Line Style Lines
Color 25; 255; 0
Alpha 1
Buffer Length 1
Offset 0; 0; 0
Pose Style None
Add Duplicate Remove Rename
Views
Type: XYOrbit (rviz) Zero
Current V... XYOrbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 44.388
Focal S... 0.05
Focal S...
Yaw -0.7846
Pitch 1.56976
Field o... 0.785398
Focal P... 6.5083; -5.7367; 0
Save Remove Rename
Time
Pause Synchronization: Off ROS Time: 1310297997.82 ROS Elapsed: 6.40 Wall Time: 1685213784.59 Wall Elapsed: 73.47
Reset
31 fps