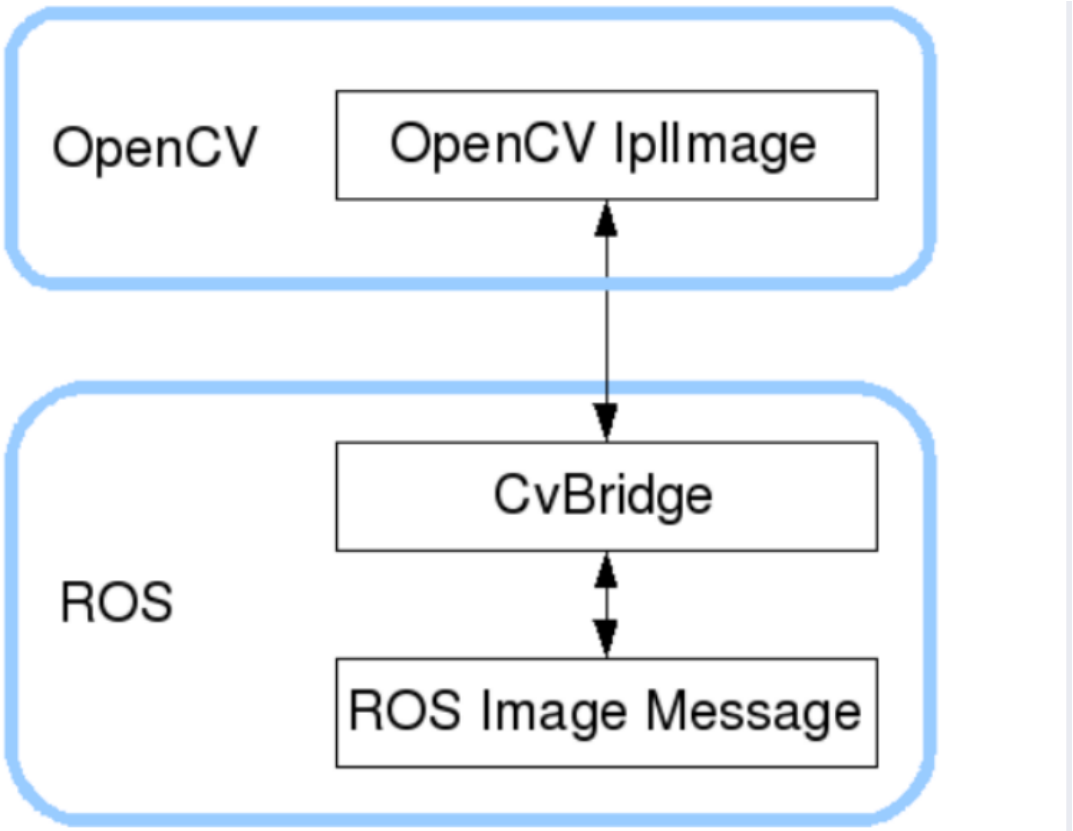
1. ROS+opencv application

ROS transmits images in its own sensor_msgs/Image message format and cannot directly process images, but the provided [CvBridge] can perfectly convert and be converted image data formats. [CvBridge] is a ROS library, which is equivalent to a bridge between ROS and Opencv.

The image data conversion between Opencv and ROS is shown in the figure below:



This lesson uses two cases to show how to use CvBridge for data conversion.

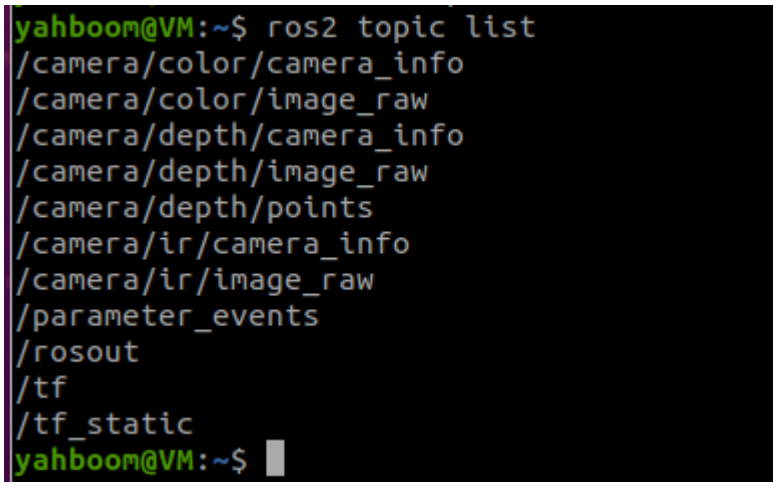
1.1. Camera topic data

In the previous section, we have built the camera driver environment and the color images, depth images and infrared IR images that can be seen. We can first check which topics are published and what the image data is after driving the camera. Enter the following command in the terminal to start the camera,

```
#astraproplus camera
ros2 launch orbbec_camera astra.launch.xml
#gemini camera startup
ros2 launch astra_camera gemini.launch.xml
```

Then, use the following command to view the topic data list,

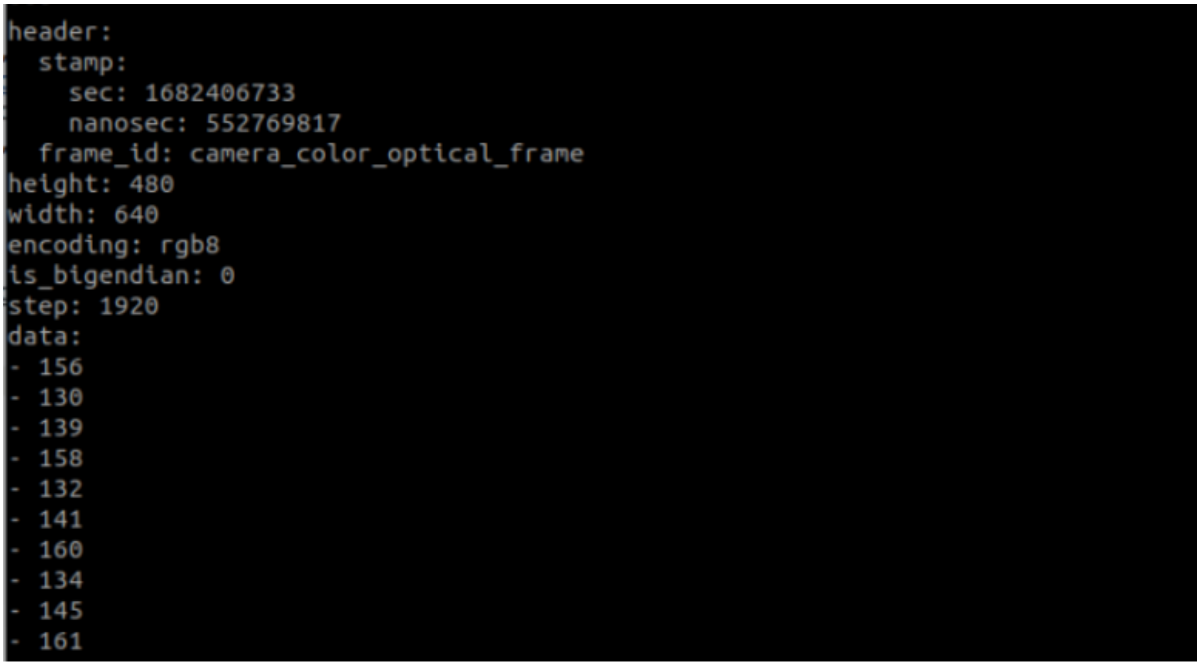
```
ros2 topic list
```



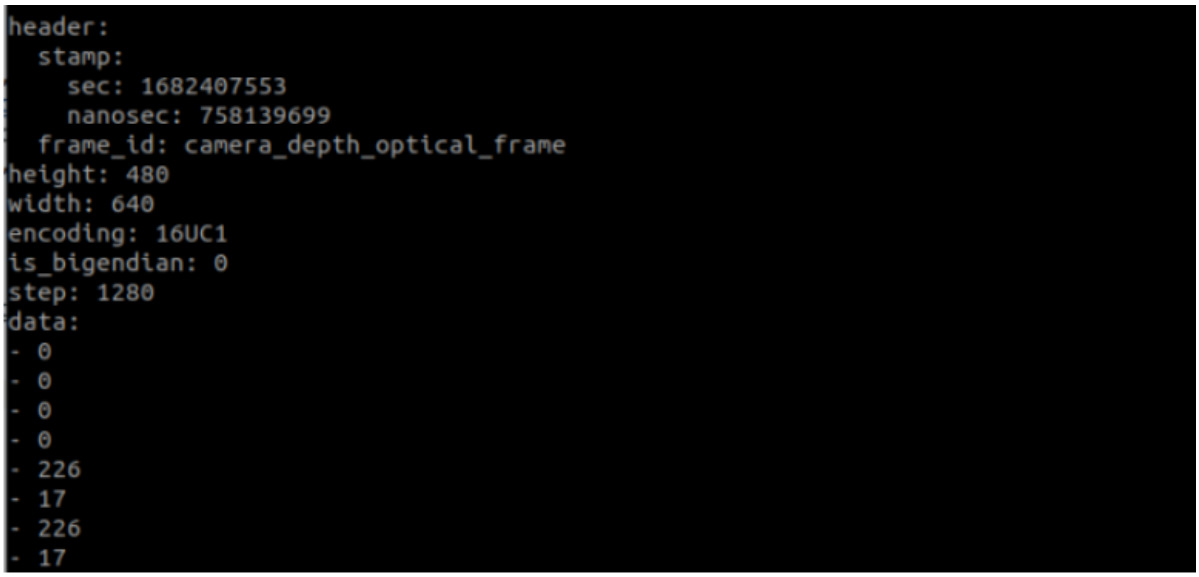
Among them, /camera/color/image_raw and /camera/depth/image_raw are the data of the color image and depth image. You can use the following command to view the data content of a certain frame,

```
#View the data content of the RGB image topic
ros2 topic echo /camera/color/image_raw
#View the data content of the Depth image topic
ros2 topic echo /camera/depth/image_raw
```

Color image:



Depth image:

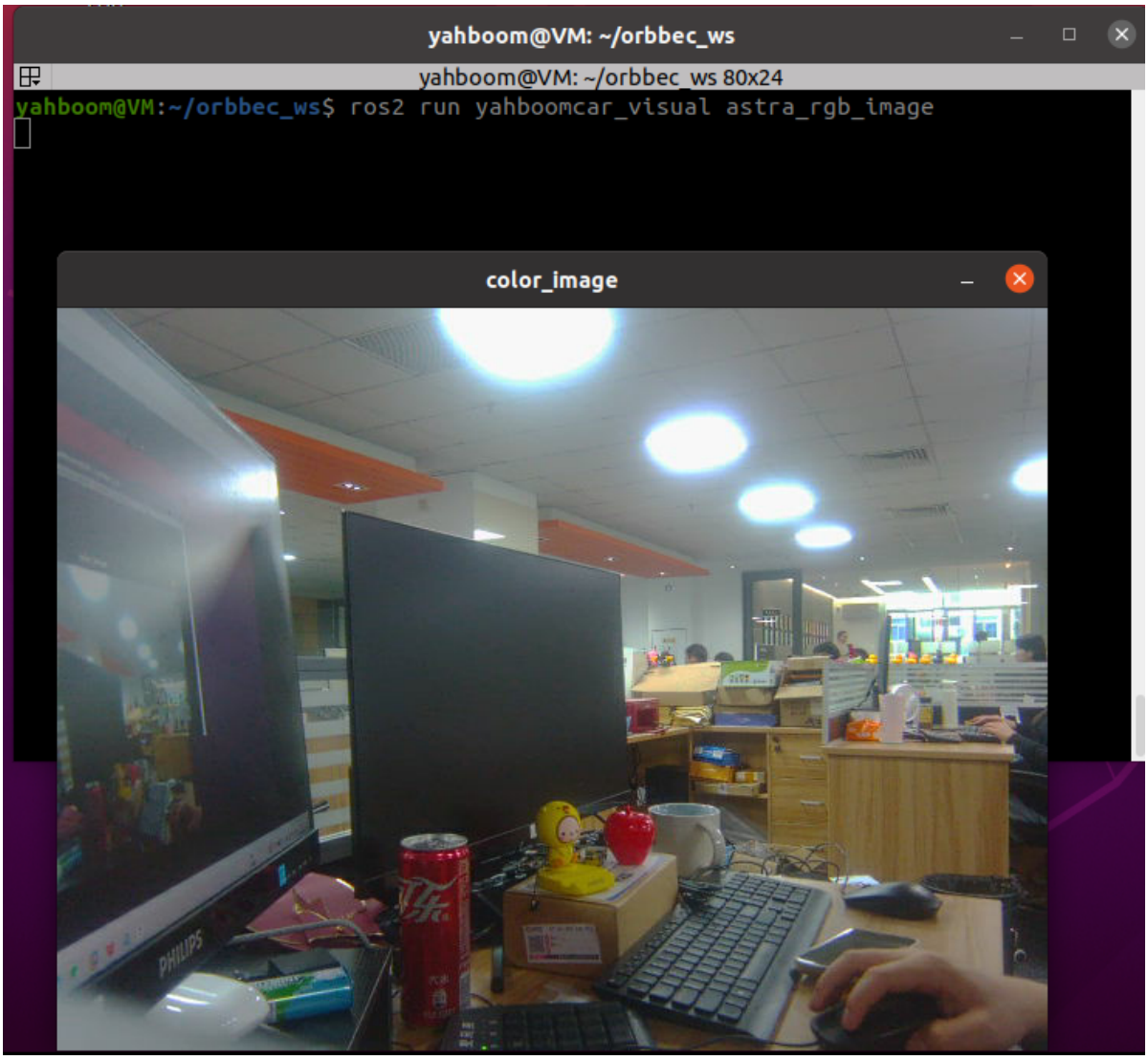


Here is a key value: **encoding**, which indicates the encoding format of the image, which needs to be referenced when converting image data later.

1.2. Subscribe to color image topic data and display color images

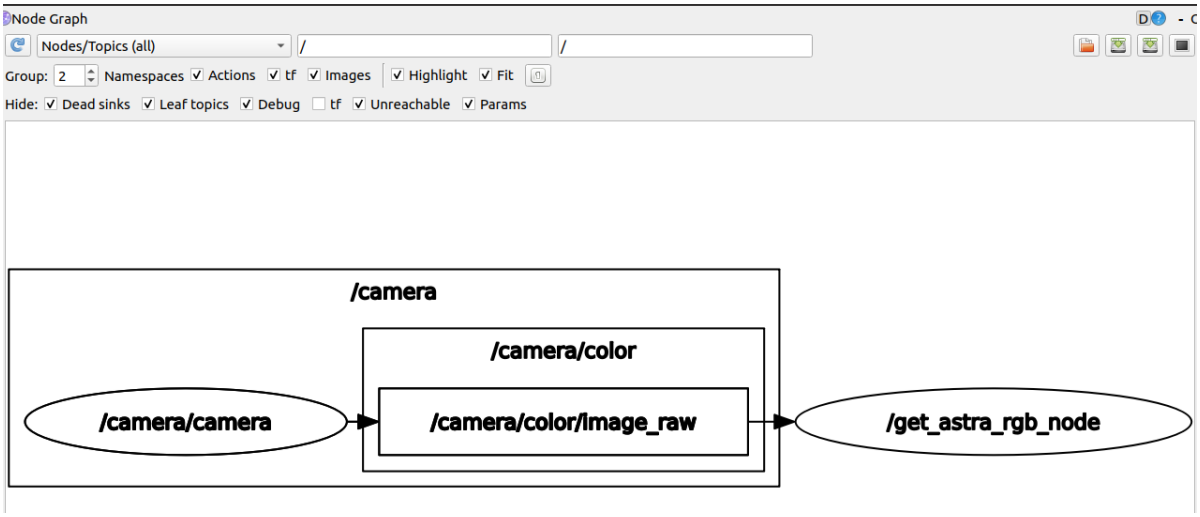
1.2.1. Run command

```
#gemini camera launch
ros2 launch astra_camera gemini.launch.xml
ros2 run yahboomcar_visual astra_rgb_image
```



View the topic communication between nodes, terminal input,

```
ros2 run rqt_graph rqt_graph
```



1.2.2, core code analysis

Code reference path:

```
~/orbbec_ws/src/yahboomcar_visual/yahboomcar_visual/astra_rgb_image.py
```

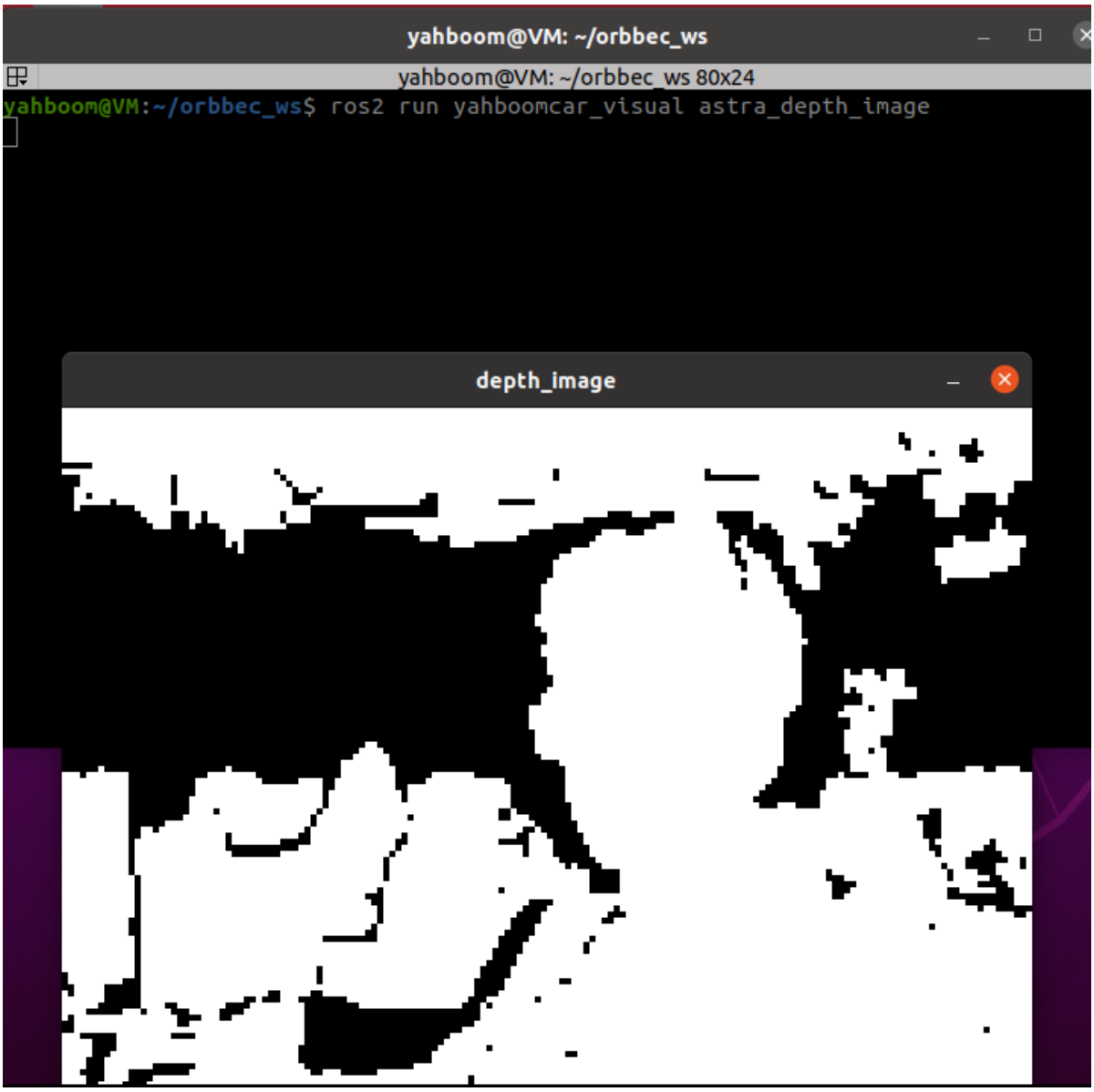
/get_astra_rgb_node node subscribes to the topic of /camera/color/image_raw, and then converts the topic data into car image data through data conversion. The code is as follows,

```
#Import opecv library and cv_bridge library
import cv2 as cv
from cv_bridge import CvBridge
#Create CvBridge object
self.bridge = CvBridge()
#Define a subscriber to subscribe to the RGB color image topic data released by
the depth camera node
self.sub_img
=self.create_subscription(Image,'/camera/color/image_raw',self.handleTopic,100)
#msg is converted into image data, where bgr8 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
```

1.3. Subscribe to the depth image topic information and display the depth image

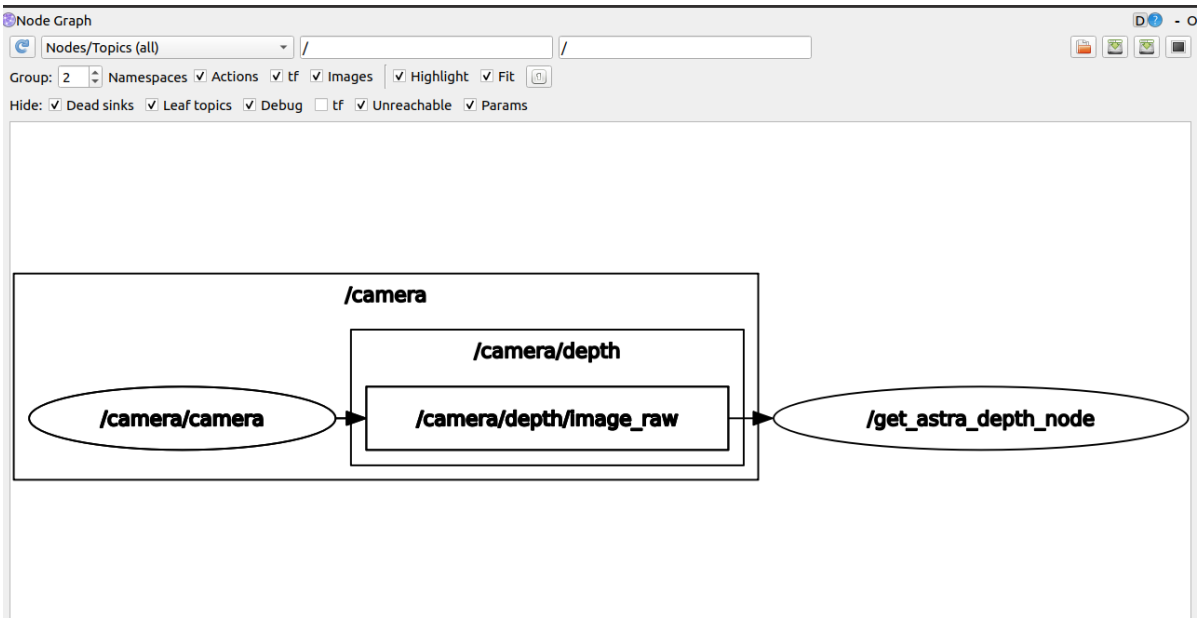
1.3.1. Run the command

```
#gemini camera launch
ros2 launch astra_camera gemini.launch.xml
ros2 run yahboomcar_visual astra_depth_image
```



View the topic communication between nodes, terminal input,

```
ros2 run rqt_graph rqt_graph
```



1.3.2, core code analysis

Code reference path:

```
~/orbbec_ws/src/yahboomcar_visual/yahboomcar_visual/astra_depth_image.py
```

The basic implementation process is the same as the RGB color image display. It subscribes to the topic data of /camera/depth/image_raw published by the depth camera node, and then converts it into image data through data conversion. The code is as follows,

```
#Import opecv library and cv_bridge library
import cv2 as cv
from cv_bridge import CvBridge
#Create CvBridge object
self.bridge = CvBridge()
#Define a subscriber to subscribe to the Depth image topic data published by the
depth camera node
self.sub_img
=self.create_subscription(Image,'/camera/depth/image_raw',self.handleTopic,10)
#msg is converted into image data, where 32FC1 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "32FC1")
```