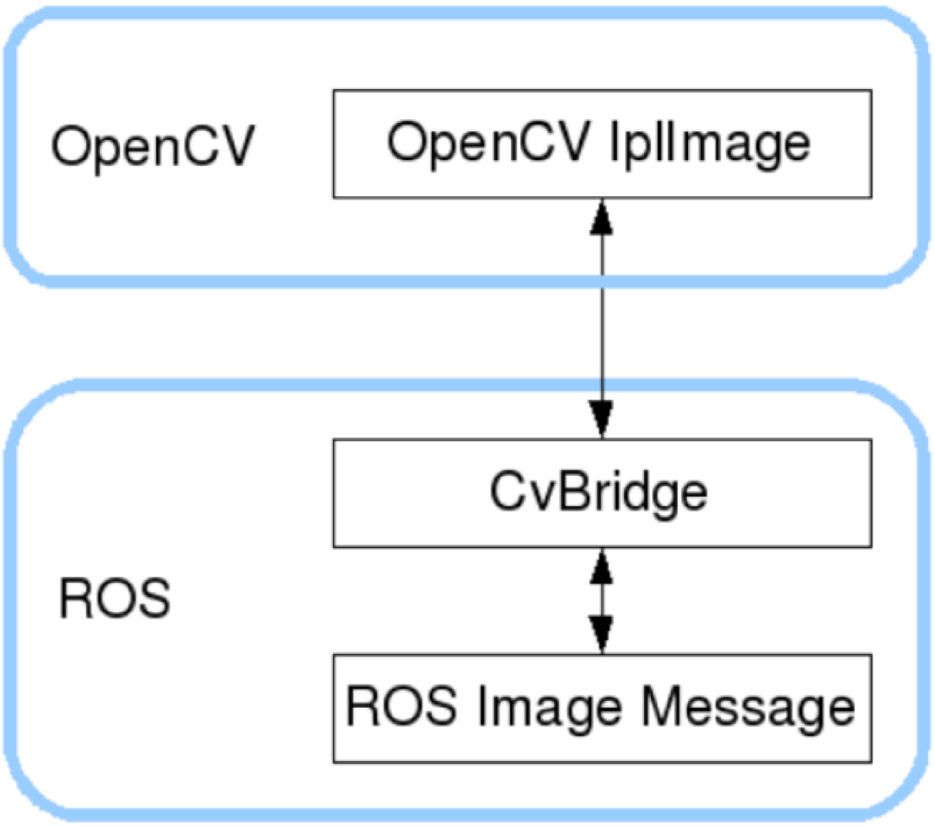
1. ROS+opencv application

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

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



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

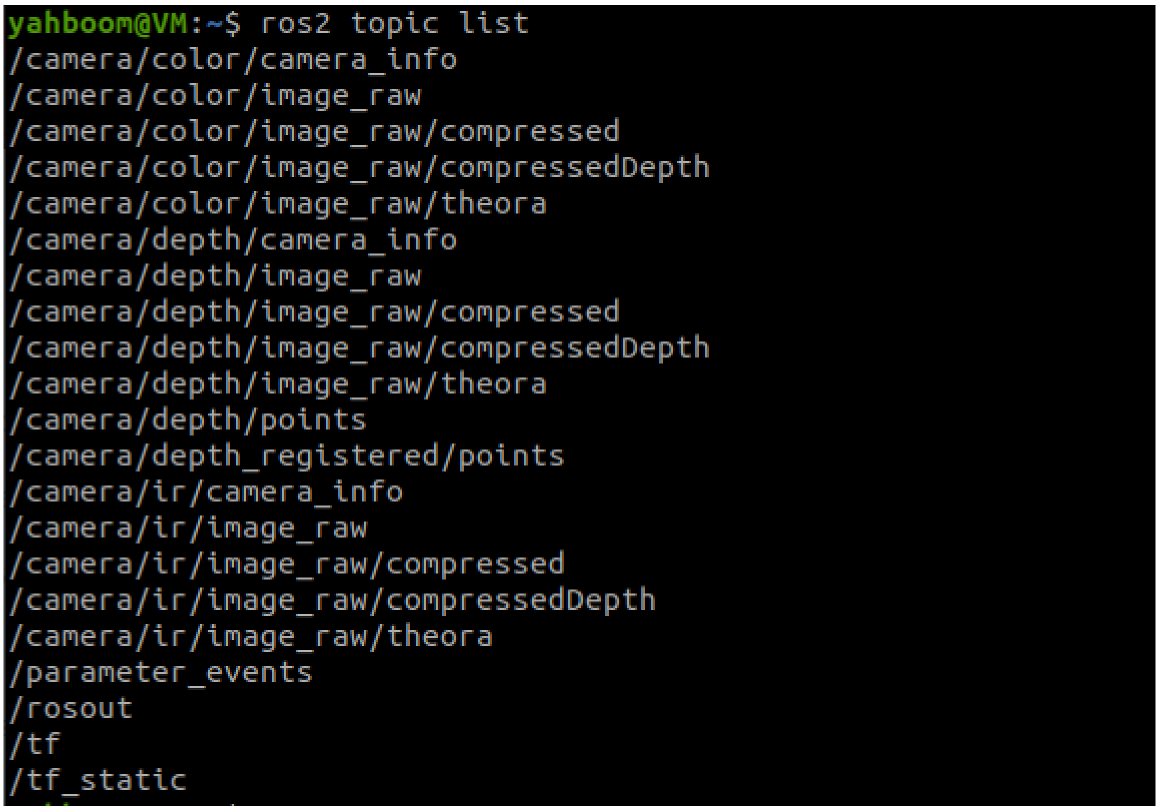
1.1. Camera topic data

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

```
#astrapropluscamera
ros2 launch orbbec_camera astra.launch.xml
#gemini2camera
ros2 launch orbbec_camera gemini2.launch.py
```

Then, view the topic data list through the following command,

```
ros2 topic list
```

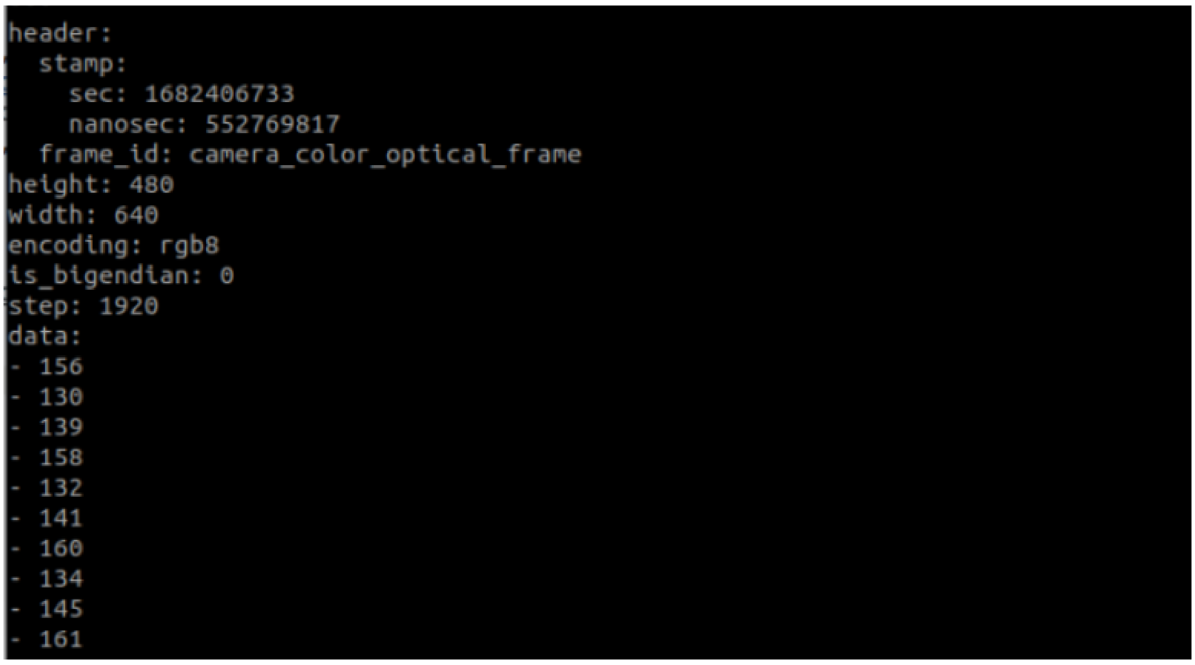


```
yahboom@VM:~$ ros2 topic list
/camera/color/camera_info
/camera/color/image_raw
/camera/color/image_raw/compressed
/camera/color/image_raw/compressedDepth
/camera/color/image_raw/theora
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/image_raw/compressed
/camera/depth/image_raw/compressedDepth
/camera/depth/image_raw/theora
/camera/depth/points
/camera/depth_registered/points
/camera/ir/camera_info
/camera/ir/image_raw
/camera/ir/image_raw/compressed
/camera/ir/image_raw/compressedDepth
/camera/ir/image_raw/theora
/parameter_events
/rosout
/tf
/tf_static
```

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

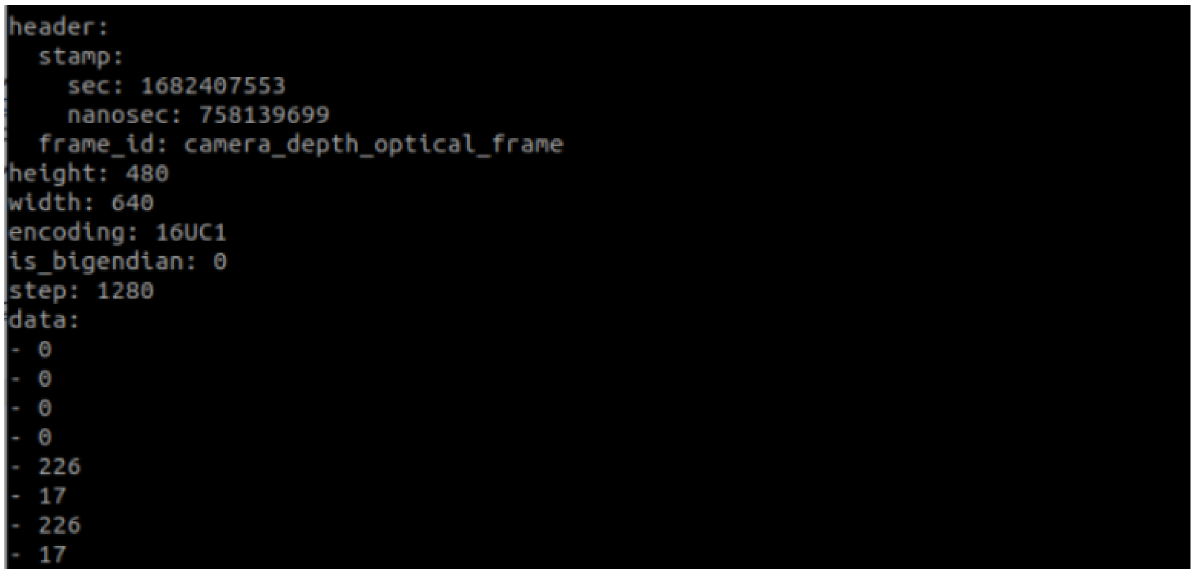
```
#View RGB image topic data content
ros2 topic echo /camera/color/image_raw
#View Depth image topic data content
ros2 topic echo /camera/depth/image_raw
```

Color image:



```
header:
  stamp:
    sec: 1682406733
    nanosec: 552769817
  frame_id: camera_color_optical_frame
height: 480
width: 640
encoding: rgb8
is_bigendian: 0
step: 1920
data:
- 156
- 130
- 139
- 158
- 132
- 141
- 160
- 134
- 145
- 161
```

Depth map:

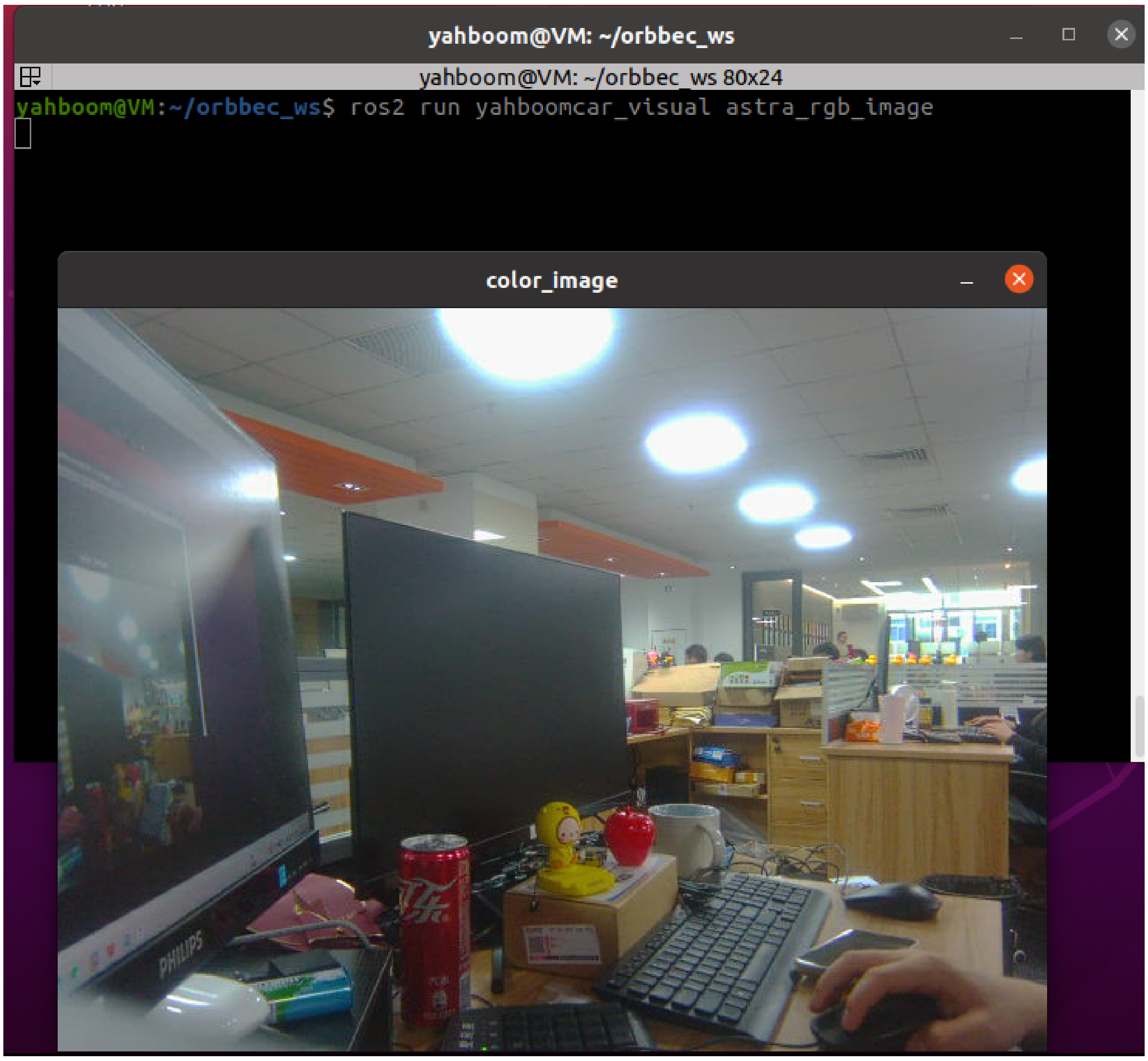

```
header:
  stamp:
    sec: 1682407553
    nanosec: 758139699
  frame_id: camera_depth_optical_frame
height: 480
width: 640
encoding: 16UC1
is_bigendian: 0
step: 1280
data:
- 0
- 0
- 0
- 0
- 226
- 17
- 226
- 17
```

There is an important value here: **encoding**, which represents the encoding format of the image and 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

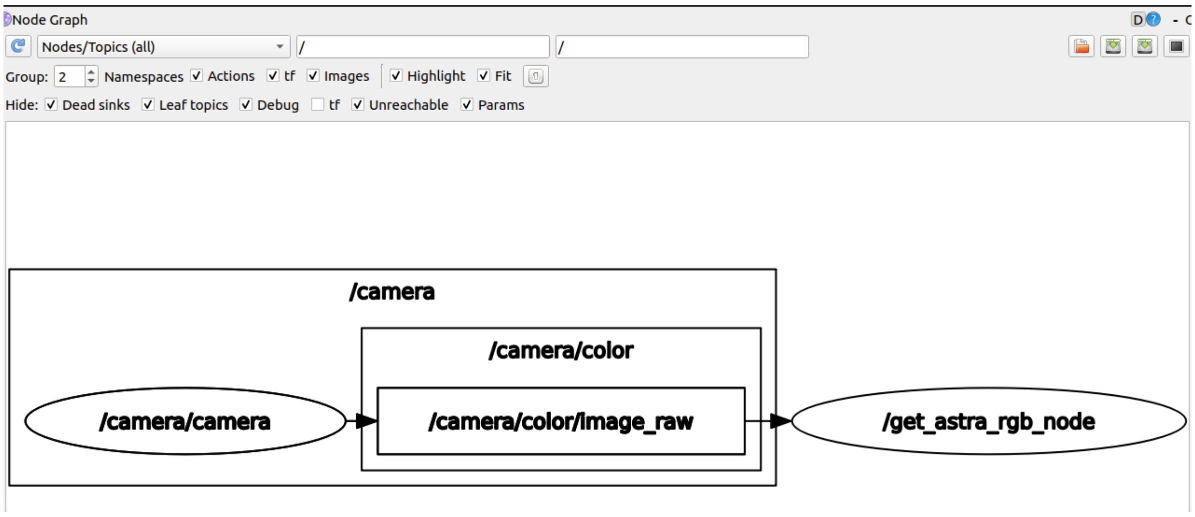
```
ros2 launch orbbec_camera gemini2.launch.py
ros2 run yahboomcar_visual astra_rgb_image
```



```
yahboom@VM:~/orbbec_ws$ ros2 run yahboomcar_visual astra_rgb_image
```

View topic communication between nodes, terminal input,

```
ros2 run rqt_graph rqt_graph
```



Group: 2 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit
Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug tf ✓ Unreachable ✓ Params

1.2.2. Core code analysis

Code reference path:

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

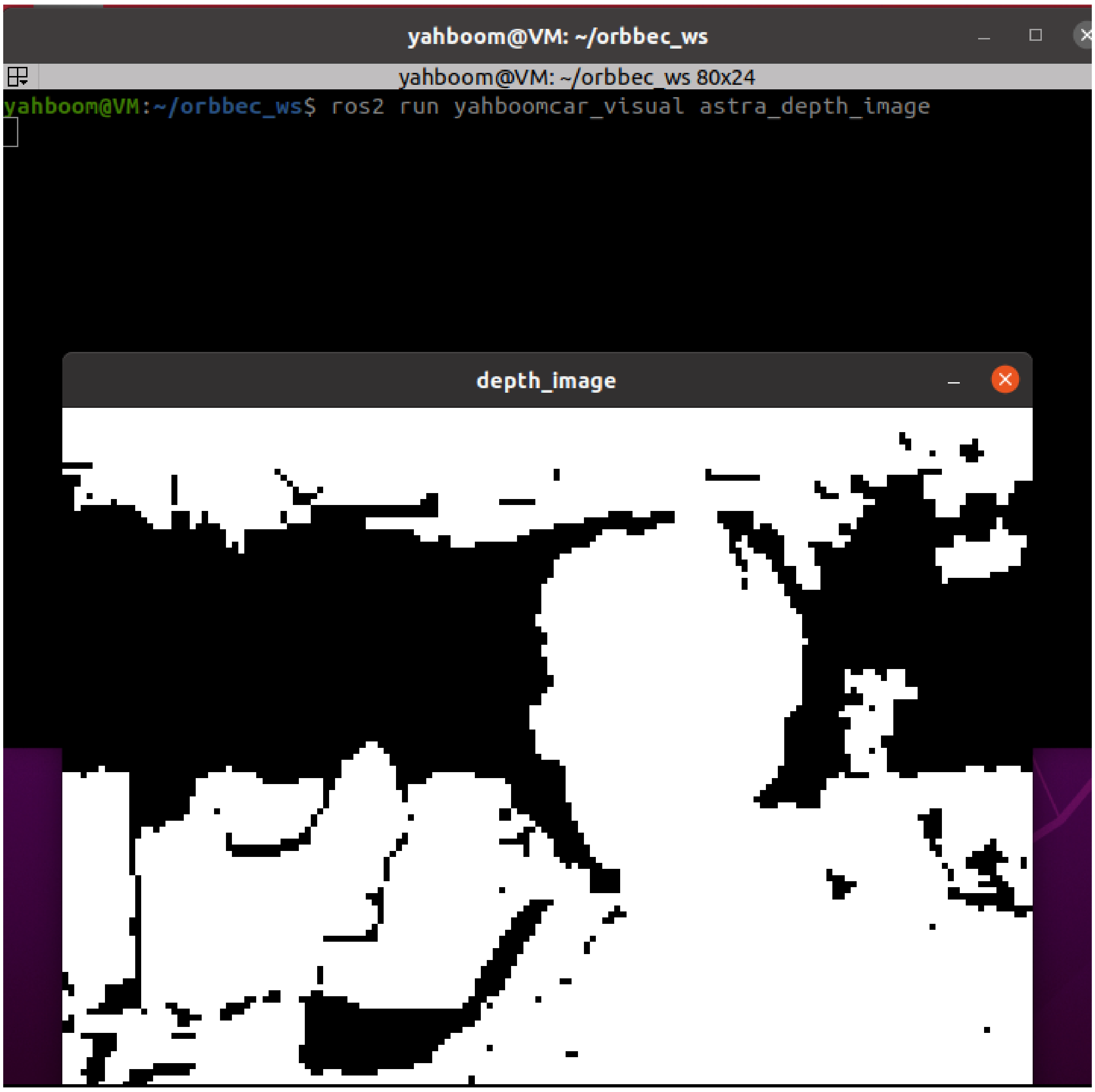
The /get_astra_rgb_node node subscribes to the topic of /camera/color/image_raw, and then uses data conversion to convert the topic data into image data and publish it. The code is as follows,

```
#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 published 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 depth image topic information and display depth images

1.3.1. Run command

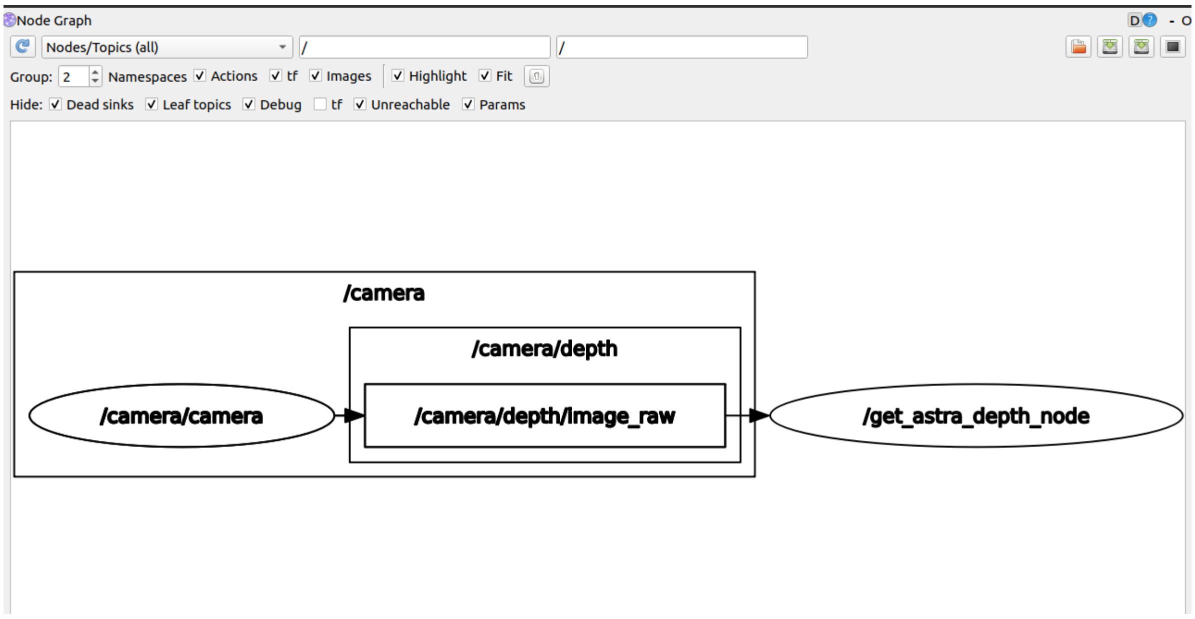
```
ros2 launch orbbec_camera gemini2.launch.py
ros2 run yahboomcar_visual astra_depth_image
```



```
yahboom@VM:~/orbbec_ws$ ros2 run yahboomcar_visual astra_depth_image
```

View 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 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 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")
```