

ROS+OpenCV Basics

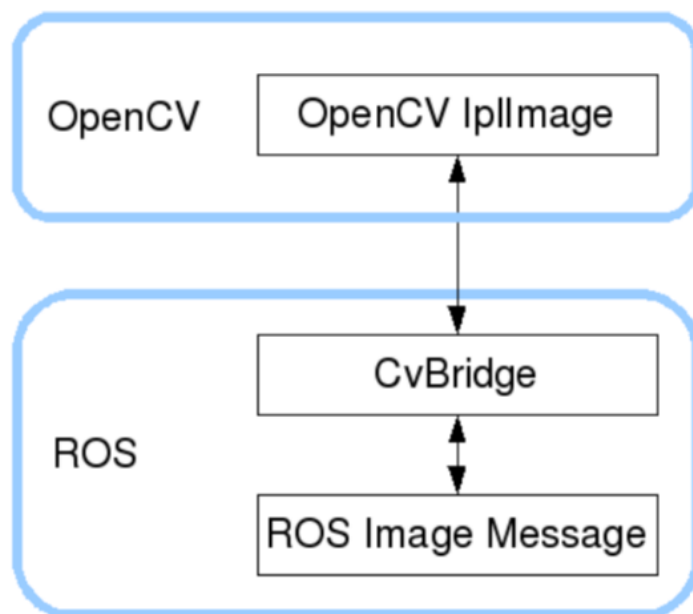
ROS+OpenCV Basics

- 3.1、 Program Function Description
- 3.2, Program code reference path
- 3.3, Start the camera
- 3.4, subscribe to RGB image topic information and publish converted image data
 - 3.4.1, node startup
 - 3.4.2, View the node communication graph
 - 3.4.3, core source code analysis
- 3.5, Subscribe to Depth image topic information and publish the converted image data
 - 3.5.1, Node startup
 - 3.5.2, View the node communication graph
 - 3.5.3, Core source code analysis

3.1、 Program Function Description

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.

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



The following takes the Astra pro camera as an example, using two cases to show how to use CvBridge for data conversion and use the ROSboard web page to display topic data.

3.2, Program code reference path

After SSH connects to the car, the source code of this function is located at,

```
#RGB image example
/home/sunrise/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow/astra_rgb_image.py

#Depth image example
/home/sunrise/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow/astra_depth_image.py
```

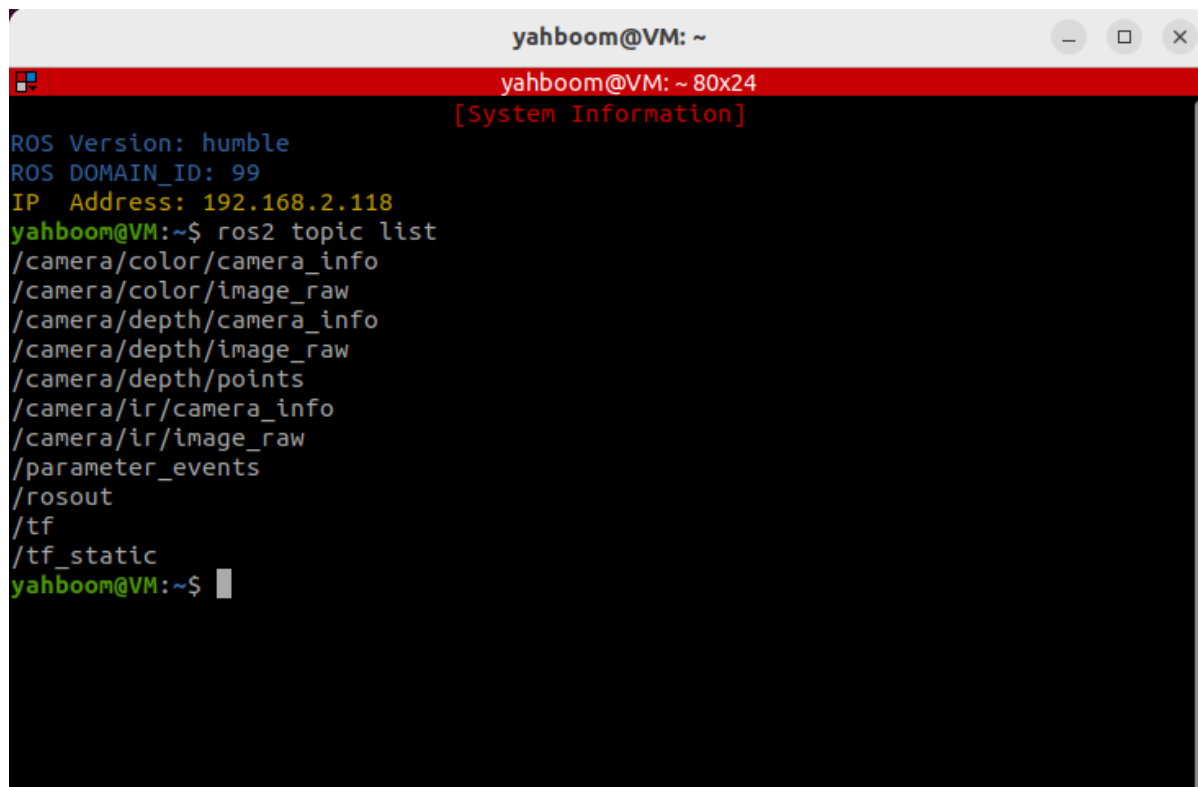
3.3, Start the camera

After SSH connects to the car, enter in the terminal,

```
ros2 launch astra_camera astro_pro_plus.launch.xml
```

View the topic through the following command, enter in the virtual machine terminal,

```
ros2 topic list
```



```
yahboom@VM: ~
yahboom@VM: ~ 80x24
[System Information]
ROS Version: humble
ROS DOMAIN_ID: 99
IP Address: 192.168.2.118
yahboom@VM:~$ ros2 topic list
/camera/color/camera_info
/camera/color/image_raw
/camera/depth/camera_info
/camera/depth/image_raw
/camera/depth/points
/camera/ir/camera_info
/camera/ir/image_raw
/parameter_events
/rosout
/tf
/tf_static
yahboom@VM:~$
```

The main thing to look at is the image data topic. Here, only the RGB color image and Depth depth image topic information are parsed. Use the following command to view the respective data information. Enter the virtual machine terminal,

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

You can intercept a frame of RGB color image information,

```
header:
  stamp:
    sec: 1731566477
    nanosec: 277631610
  frame_id: camera_color_optical_frame
height: 480
width: 640
encoding: rgb8
is_bigendian: 0
step: 1920
data:
- 254
- 254
- 254
- 254
- 254
- 254
- 254
```

Here is the basic information of the image, an important value, **encoding**, the value here is **rgb8**, indicating that the encoding format of this frame of image is rgb8, which needs to be referred to when doing data conversion later.

Input in the virtual machine terminal,

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

Intercept the image data information of a certain frame of the depth image,

```
header:
  stamp:
    sec: 1731566524
    nanosec: 765189853
  frame_id: camera_depth_optical_frame
height: 480
width: 640
encoding: 16UC1
is_bigendian: 0
step: 1280
data:
- 0
- 0
- 0
- 0
```

The encoding value here is **16UC1**.

3.4, subscribe to RGB image topic information and publish converted image data

3.4.1, node startup

After SSH connects to the car and starts the camera, input in the terminal: You need to start the camera first and then enter the following command

```
#RGB color image display node
ros2 run yahboomcar_imshow astra_rgb_image
```

Input in the car terminal,

#Start ROSboard

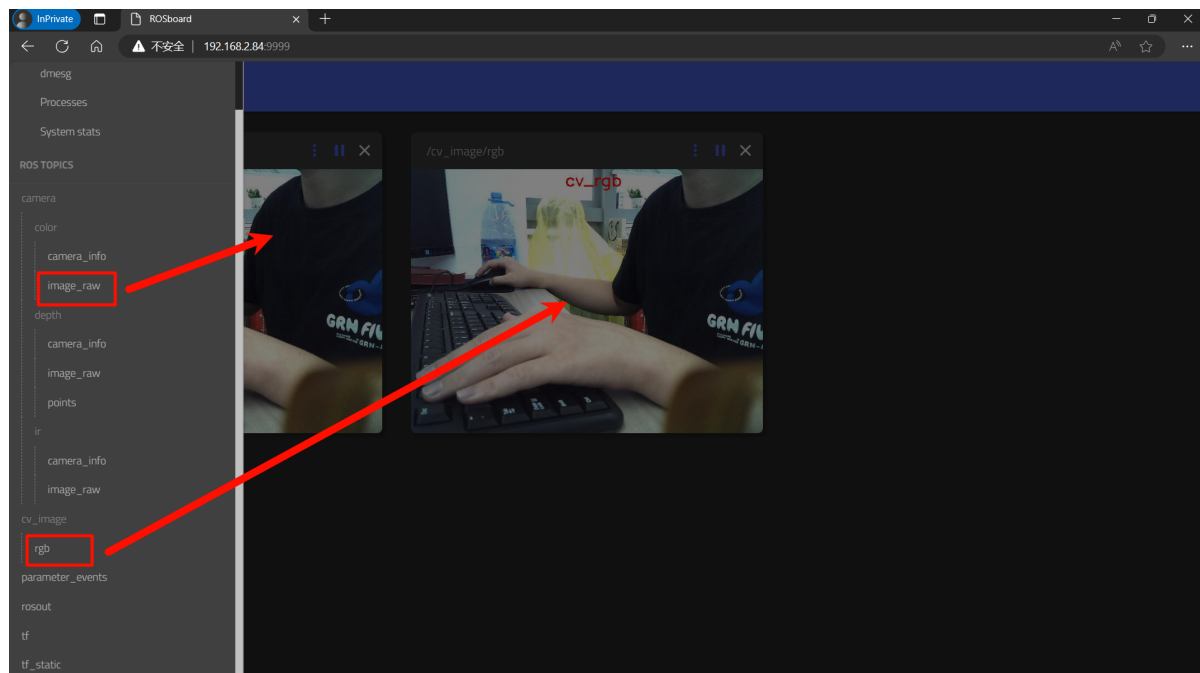
```
ros2 run rosboard rosboard_node
```

Open the browser on the PC (note that the computer and the Sunrise Network must be in the same LAN), enter the URL: car IP:9999, for example, my car IP is 192.168.2.84, enter the URL in the browser on the virtual machine to open the ROSboard web page:

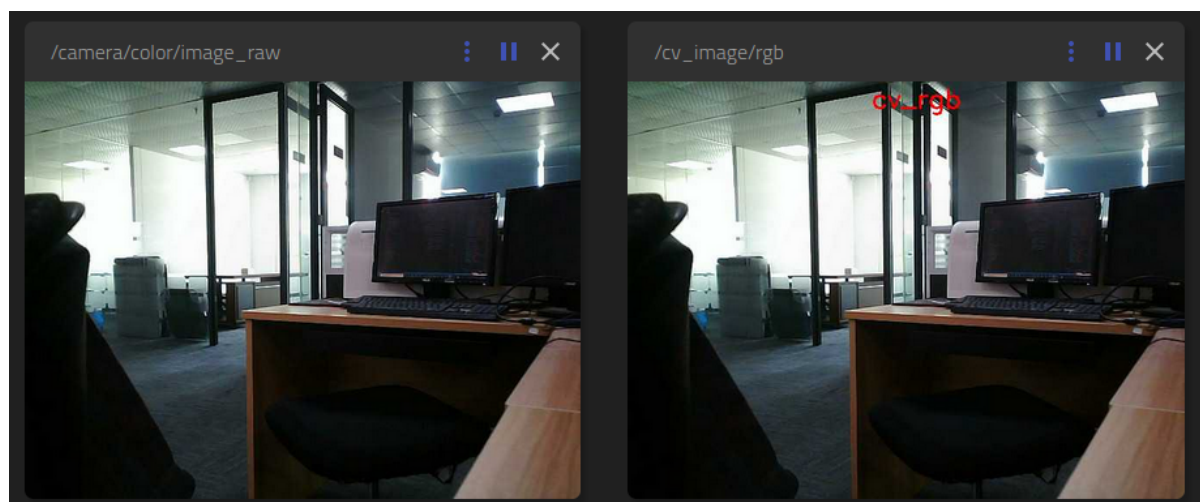
192.168.2.84:9999

 image-20241114144750775

Select the /camera/color/image_raw and /cv_image/rgb topics, and you can see the following screen,



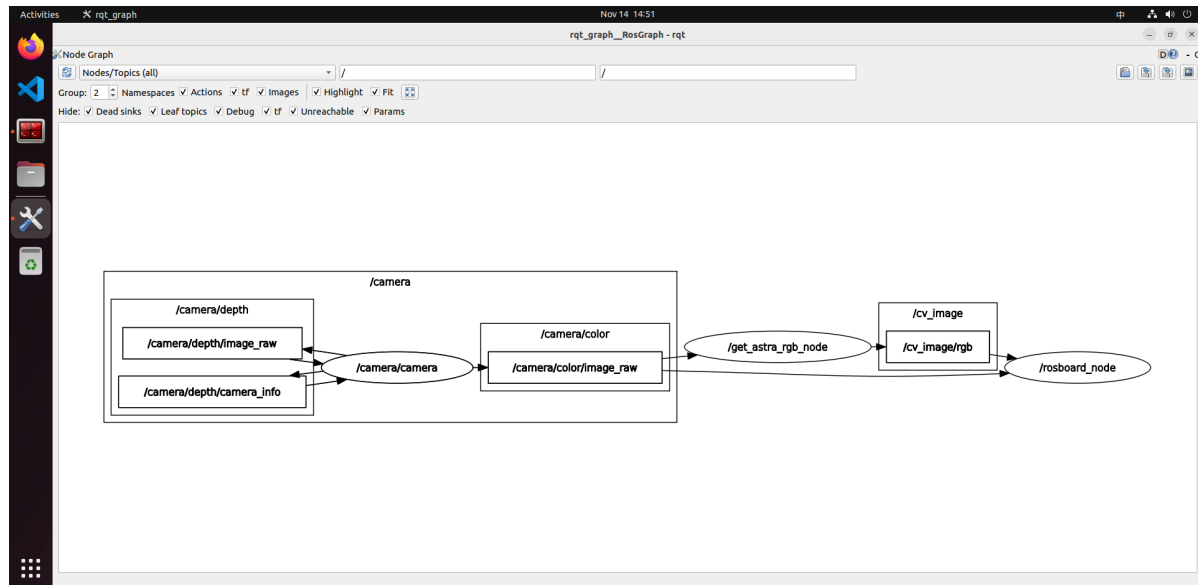
The two images are the same and synchronized, except that the /cv_image/rgb image has the "cv_rgb" character.



3.4.2, View the node communication graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



3.4.3, core source code analysis

Code reference path,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow/astra_rgb_image.py
```

From 3.2.2, we can see that the `/get_astra_rgb_node` node subscribes to the topic of `/camera/color/image_raw`, and then converts the ROS topic data into opencv image data through `CvBridge`. After being processed by opencv, the processed image data is converted into ROS topic data through `CvBridge` and published, that is, ROS topic data->opencv image data->ROS topic data. 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, 1)
# Define a publisher to publish processed image topic data
self.pub_img = self.create_publisher(Image, '/cv_image/rgb', 10)

# Convert msg to cv image data, where bgr8 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
# Standardize input image size
frame = cv.resize(frame, (640, 480))
# Add characters to the image
```

```
cv.putText(frame, "cv_rgb", (280, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255),
2)
# Convert the processed cv image data to msg data
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
# Publish the converted image
self.pub_img.publish(msg)
```

3.5, Subscribe to Depth image topic information and publish the converted image data

3.5.1, Node startup

After SSH connects to the car and starts the camera, enter the terminal: You need to start the camera first and then enter the following command

```
#Depth image display node
ros2 run yahboomcar_imshow astra_depth_image
```

Car terminal input,

```
#Start ROSboard
ros2 run rosboard rosboard_node
```

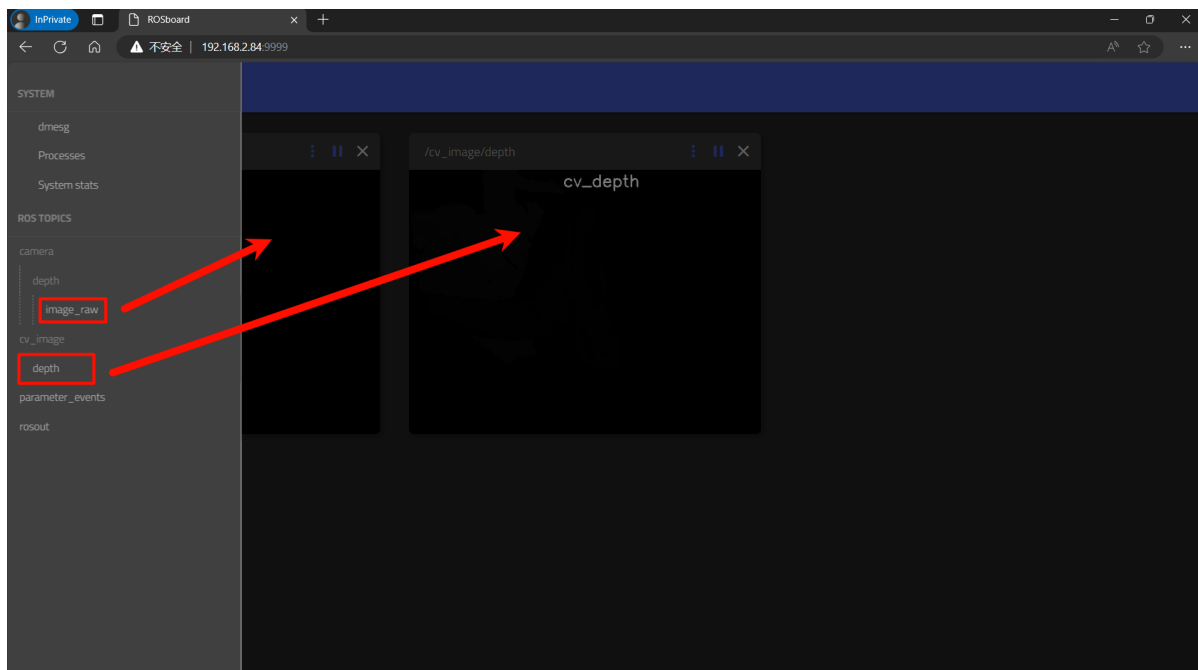
Open the browser on the PC (note that the computer and the Sunrise Network must be in the same LAN), enter the URL: car IP:9999, for example, my car IP is 192.168.2.84, enter the URL in the browser on the virtual machine to open the ROSboard web page:

```
192.168.2.84:9999
```

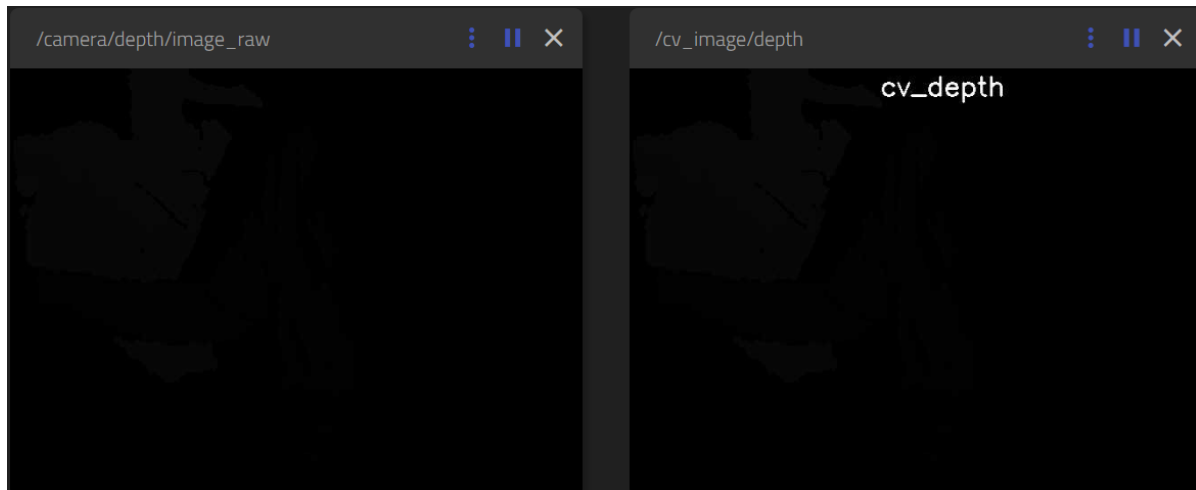


image-20241114144750775

Select /camera/depth/image_raw and /cv_image/depth topics, and you can see the following screen,



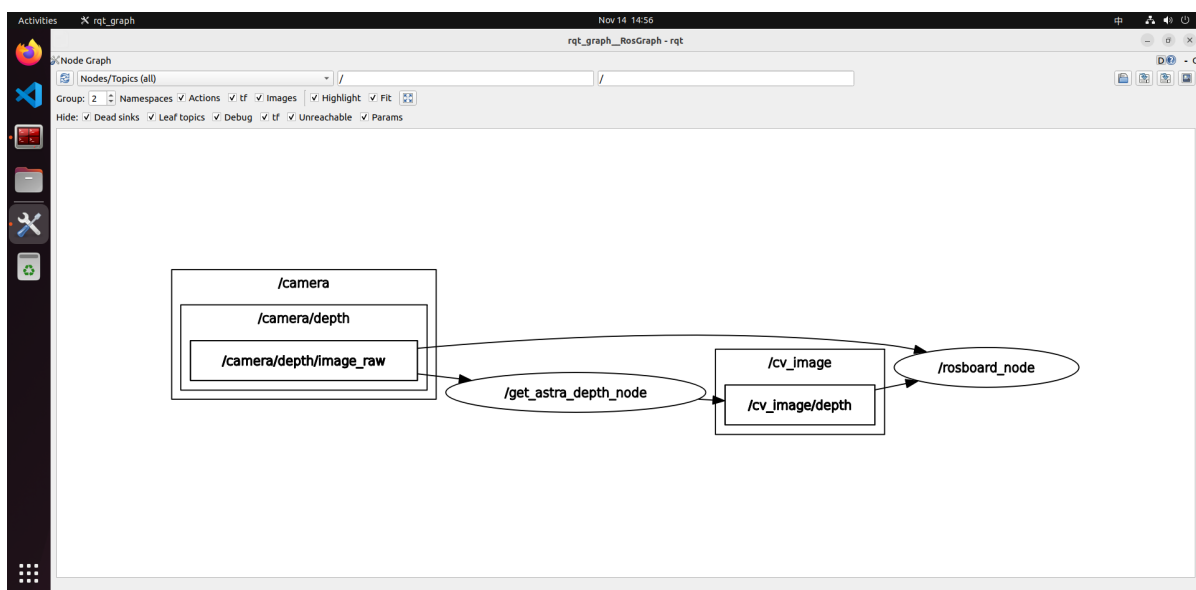
The two images are the same and synchronized, except that the /cv_image/depth image has the "cv_depth" character.



3.5.2, View the node communication graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



3.5.3, Core source code analysis

Code reference path,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow/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 completes the conversion of ROS topic data->opencv image data->ROS topic data through CvBridge. The code is as follows,

```
# Import opencv 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, 1)
# Define a publisher to publish processed image topic data
self.pub_img = self.create_publisher(Image, '/cv_image/depth', 10)

# Convert msg to cv image data, where 16UC1 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "16UC1")
# Standardize input image size
frame = cv.resize(frame, (640, 480))
# Add text to the image
cv.putText(frame, "cv_depth", (280, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (65536,
65536, 65536), 2)
# Convert the processed cv image data to msg data
msg = self.bridge.cv2_to_imgmsg(frame, "16UC1")
# Publish the converted image
self.pub_img.publish(msg)
```