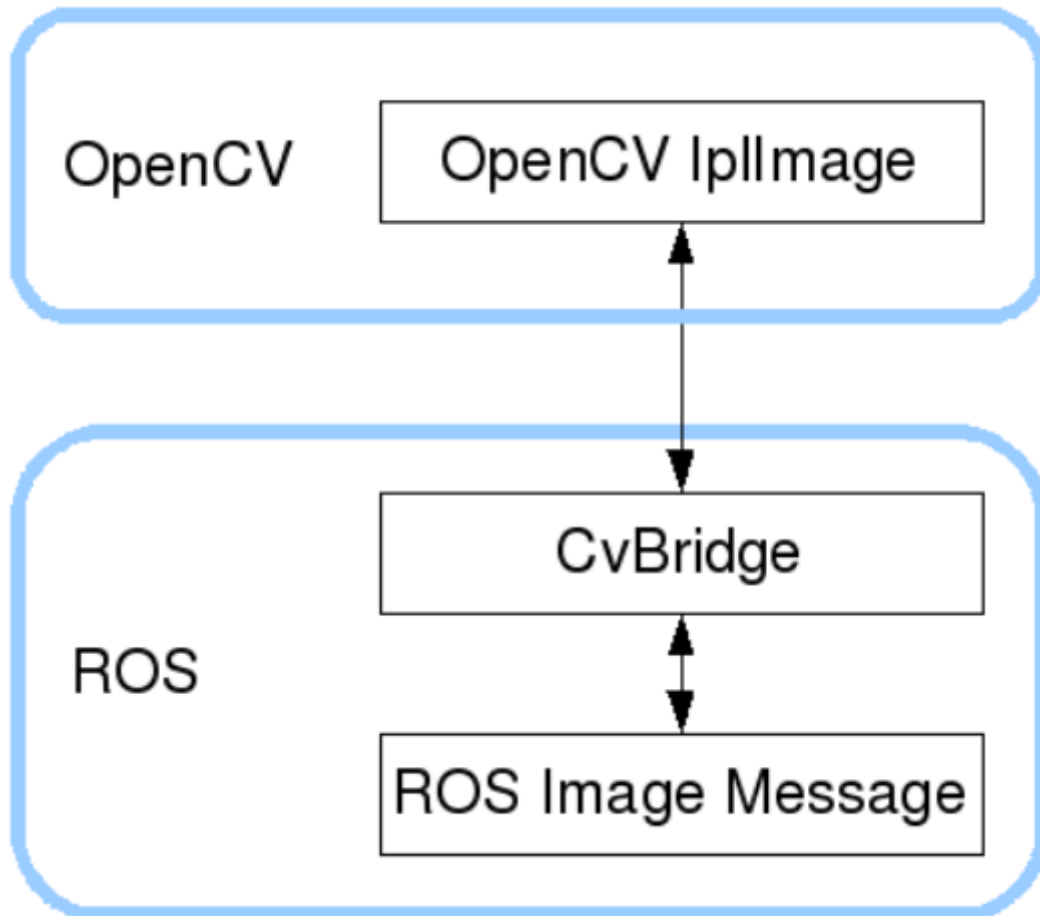


2. ROS+opencv application

This lesson takes an Astra camera as an example, normal cameras are similar.

ROS passes the image in its own sensor_msgs/Image message format, which can't be used for direct image processing, but the provided [CvBridge] can perfectly convert and be converted image data format. CvBridge] is a ROS library, equivalent to the bridge between ROS and Opencv.

Opencv and ROS image data conversion is shown below:



This lesson shows how to use CvBridge for data conversion with three examples.

1. Astra camera

Before driving the depth camera, you need to be able to recognize the astra camera device in the host; when entering the docker container, you need to mount this astra device in order to be in the docker container and recognize the camera. Matching host has been built environment, do not need additional configuration, if the new host, you need to add the rules file. The method is simple, copy the /etc/udev.rules.d/56-orbbecusb.rules file under the host machine to the /etc/udev.rules.d directory of the new environment, and then reboot once.

1.1.1 Starting the camera

To start the Astraproplus camera for example, after entering the docker container, terminal input, the

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

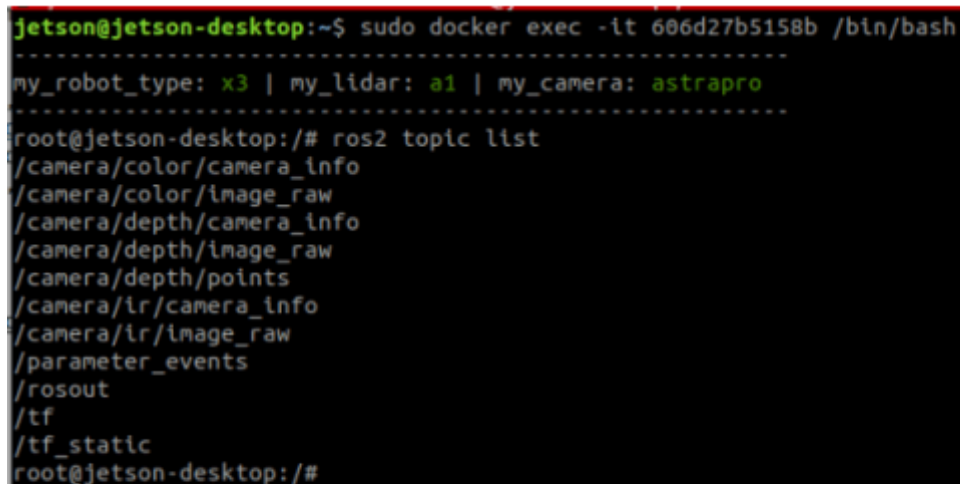
The corresponding camera models are activated as shown in the following table.

launch file	Camera model
ros2 launch astra_camera astra_pro.launch.xml	Astrapro
ros2 launch astra_camera astro_pro_plus.launch.xml	Astraproplus
ros2 launch astra_camera astra.launch.xml	Astramini

1.1.2 Viewing Camera Topics

docker terminal input.

```
ros2 topic list
```



```
jetson@jetson-desktop:~$ sudo docker exec -it 606d27b5158b /bin/bash
-----
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
-----
root@jetson-desktop:/# 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
root@jetson-desktop:/#
```

The main thing to look at is the image data topic, here only parse the RGB color image and Depth depth image topic information, with the following commands to view the respective data information, docker terminal input, the

```
#查看RGB图像话题数据内容
#View RGB image topic data content
ros2 topic echo /camera/color/image_raw
#查看Depth图像话题数据内容
#View Depth image topic data content
ros2 topic echo /camera/depth/image_raw
```

Let's take a frame of RGB color image information and see.

```
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
```

Here illustrates the basic information of the image, an important value, **encoding**, here the value is **rgb8**, this value indicates that the encoding format of this frame is rgb8, this in the back to do the data conversion needs to be referred to.

Similarly, the following is the depth image of a certain frame image data information.

```
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
```

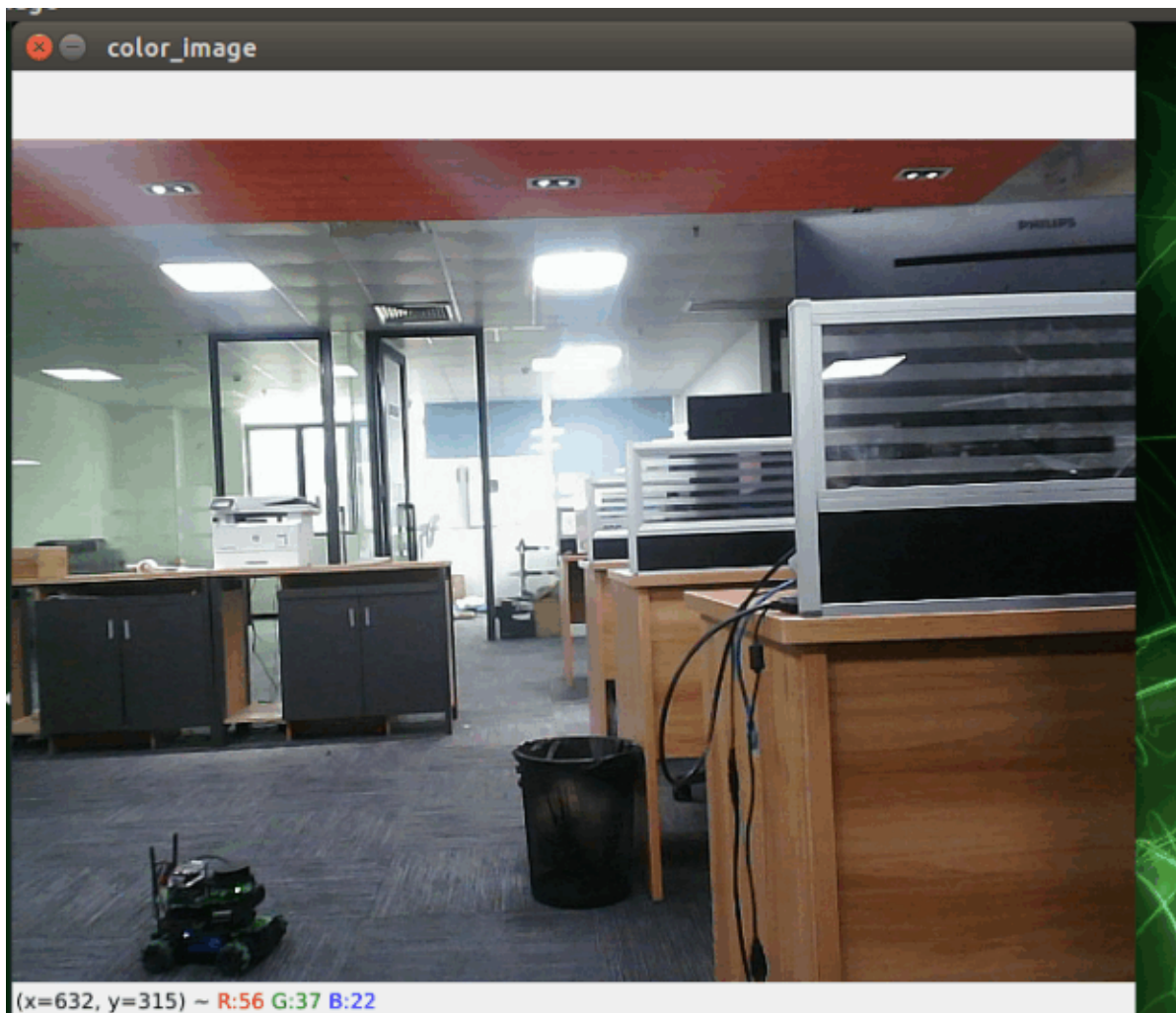
The value of encoding is **16UC1**.

2. Subscribe to the RGB image topic information and display the RGB image

2.1. Run the command

docker terminal enter.

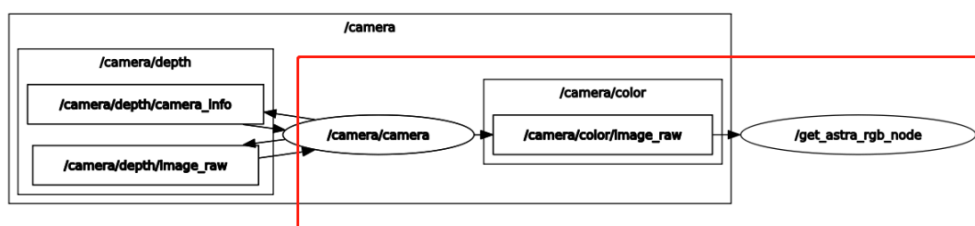
```
#RGB彩色图像显示节点 #RGB color image display node
ros2 run yahboomcar_visual astra_rgb_image
#运行Astrapro相机 #Running the Astrapro camera
ros2 launch astra_camera astra_pro_plus.launch.xml
```



2.2. View node communication graphs

docker terminal enter, the

```
ros2 run rqt_graph rqt_graph
```



2.3. Core Code Analysis

The code reference path.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/yahboomcar_visual/astra_rgb_image.py
```

From 2.2, the /get_astra_rgb_node node subscribes to the topic /camera/color/image_raw, and then converts the topic data to car image data for publishing through data conversion, with the following code.

```
#导入opencv库以及cv_bridge库
# Import the opencv library and the cv_bridge library.
import cv2 as cv
from cv_bridge import CvBridge
#创建CvBridge对象 #Creating a CvBridge object
self.bridge = CvBridge()
#定义一个订阅者订阅深度相机节点发布的RGB彩色图像话题数据
#Define a subscriber to subscribe to RGB color image topic data published by a
depth camera node
self.sub_img
=self.create_subscription(Image, '/camera/color/image_raw', self.handleTopic, 100)
#msg转换成图像数据，这里的bgr8是图像编码格式
#msg to image data, here bgr8 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
```

3. Subscribe to Depth image topic information and display Depth images

3.1 Run the command

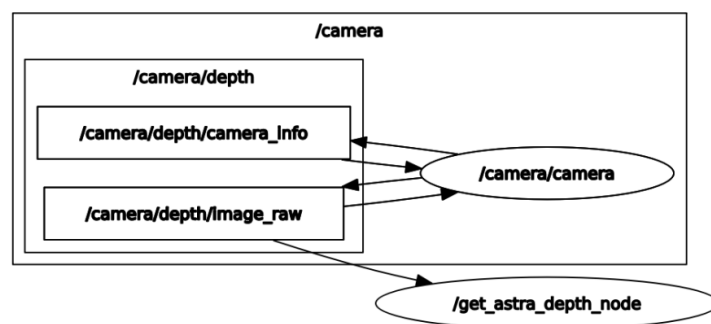
```
#RGB彩色图像显示节点 #RGB color image display node
ros2 run yahboomcar_visual astra_depth_image
#运行Astrapro相机 #Running the Astrapro camera
ros2 launch astra_camera astro_pro_.launch.xml
```



3.2. View node communication graphs

docker terminal by typing.

```
ros2 run rqt_graph rqt_graph
```



3.3 Core Code Analysis

The code reference path.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/yahboomcar_visual/astra_depth_image.py
```

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

```
#导入opencv库以及cv_bridge库
# Import the opencv library and the cv_bridge library.
import cv2 as cv
from cv_bridge import CvBridge
#创建CvBridge对象 #Creating a CvBridge object
self.bridge = CvBridge()
#定义一个订阅者订阅深度相机节点发布的Depth深度图像话题数据
#Define a subscriber to subscribe to Depth depth image topic data published by a
depth camera node
self.sub_img
=self.create_subscription(Image, '/camera/depth/image_raw', self.handleTopic, 10)
#msg转换成图像数据，这里的32FC1是图像编码格式
#msg converted to image data, here 32FC1 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "32FC1")
```

4. Subscribe to the image data and then publish the converted image data

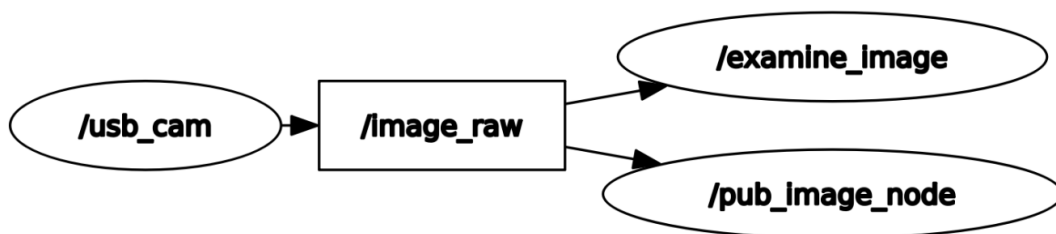
4.1. Run the command

```
#运行发布图像话题数据节点
#run publish image topic data node
ros2 run yahboomcar_visual pub_image
#运行usb摄像头话题节点
#run usb camera topic node
ros2 launch usb_cam demo_launch.py
```

4.2. View node communication graphs

docker terminal by typing.

```
ros2 run rqt_graph rqt_graph
```



4.3 Viewing topic data

First, query what image topics are published, docker terminal input, the

```
ros2 topic list
```

```

root@jetson-desktop:/# ros2 topic list
/camera_info
/image
/image_raw
/image_raw/compressed
/image_raw/compressedDepth
/image_raw/theora
/parameter_events
/rosout
root@jetson-desktop:/# 

```

where /image is the data of the topic we posted, print the following command to see the content of the data of this topic.

```
ros2 topic echo /image
```

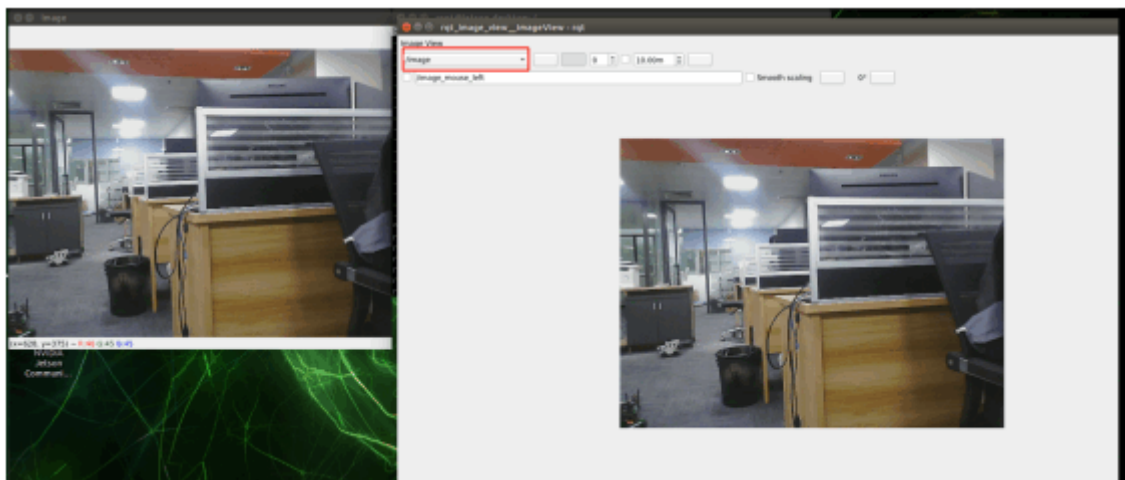
```

---
header:
  stamp:
    sec: 0
    nanosec: 0
  frame_id: ''
height: 480
width: 640
encoding: bgr8
is_bigendian: 0
step: 1920
data:
- 95
- 86
- 122
- 90
- 81
- 117
- 96
- 83

```

The image can be viewed with the rqt_image_view tool, the

```
ros2 run rqt_image_view rqt_image_view
```



Once opened, select the topic name/image in the upper left corner to view the image.

4.4 Core Code Analysis

Code path.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/yahboomcar_visual/pub_image.py
```

The implementation steps are roughly the same as the previous two, the program first subscribes to the /image_raw topic data, and then converts it to image data, but here it also does the conversion, converts the image data to topic data, and then publishes it, i.e., image topic data->image data->image topic data.

```
#导入opencv库以及cv_bridge库
# Import the opencv library and the cv_bridge library.
import cv2 as cv
from cv_bridge import CvBridge
#创建CvBridge对象 #Creating a CvBridge object
self.bridge = CvBridge()
#定义一个订阅者订阅usb图像话题数据
#define a subscriber to subscribe to usb image topic data
self.sub_img = self.create_subscription(Image, '/image_raw', self.handleTopic, 500)
#定义了图像话题数据发布者
#defined image topic data publisher
self.pub_img = self.create_publisher(Image, '/image', 500)
#msg转换成图像数据imgmsg_to_cv2, 这里的bgr8是图像编码格式
#msg converted to image data imgmsg_to_cv2, where bgr8 is the image encoding
format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
#图像数据转换的图像话题数据(cv2_to_imgmsg)然后发布出去
#Image data converted image topic data (cv2_to_imgmsg) and then publish it
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
self.pub_img.publish(msg)
```