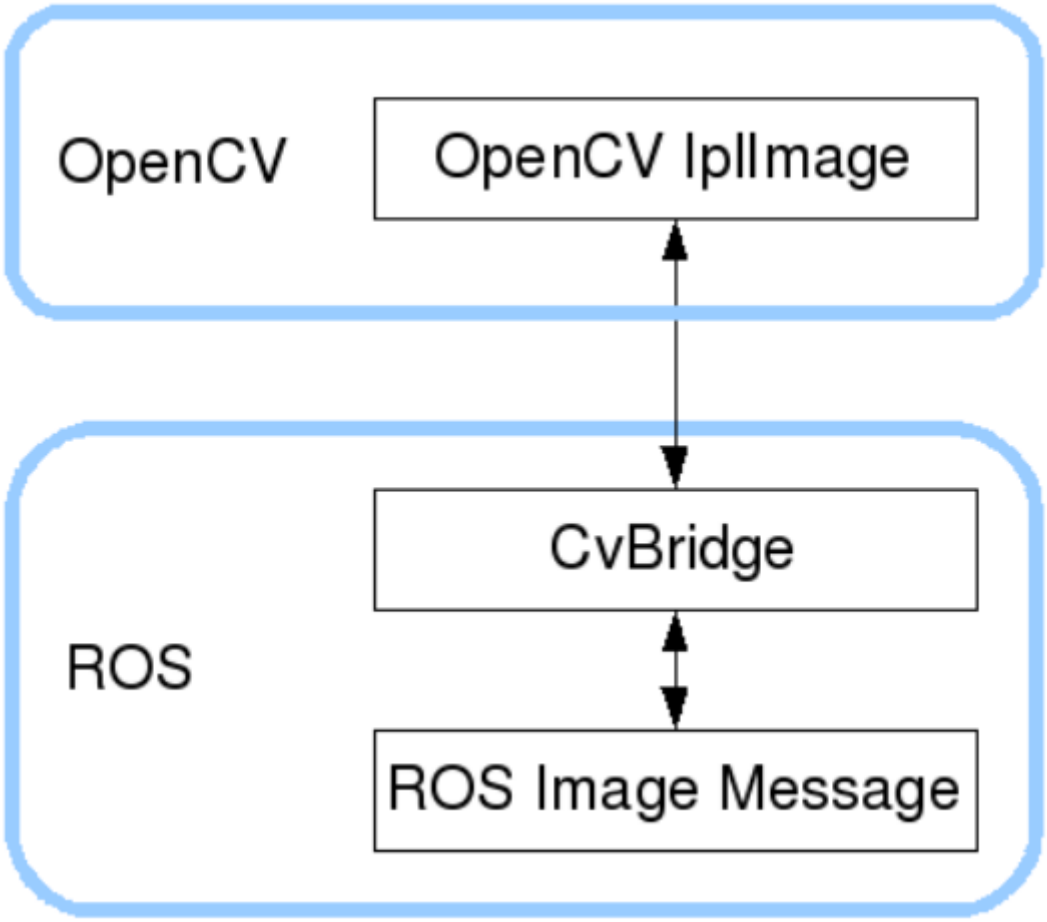
ROS+opencv application

This lesson takes a monocular camera as an example.

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 three cases to show how to use CvBridge for data conversion.

1、Monocular camera

Before driving the camera, the camera device needs to be recognized on the host machine; when entering the docker container, the USB device needs to be mounted to recognize the camera in the docker container. The supporting host has already set up an environment and does not require additional configuration. If it is on a new host, you need to add this to the startup file.

```
-v /dev/video0:/dev/video0
```

```
#!/bin/bash
xhost +
docker run -it --rm \
--privileged=true \
--net=host \
--env="DISPLAY" \
--env="QT_X11_NO_MITSHM=1" \
-v /tem/.X11-unix:/tmp/.X11-unix \
--security-opt apparmor:unconfined \
-v /dev/input:/dev/input \
-v /dev/video0:/dev/video0 \
192.168.2.51:5000/ros-humble:9.0 /bin/bash /root/1.sh
~
~
```

1.1、Start camera

1.1.1、Source code path

```
cd  ~/opt/ros/humble/share/usb_cam/
```

1.1.2、installation steps

Taking starting the monocular camera as an example, you can use the command to directly download the camera driver file of ros2.

```
sudo apt-get install ros-humble-usb-cam
```

The factory docker image has been installed and there is no need to install it again.

1.1.3、Start camera

```
ros2 run usb_cam  usb_cam_node_exe
```

1.2、View camera topics

Enter the following command in docker terminal

```
ros2 topic list
```

```
root@raspberrypi:~# ros2 topic list
/beep
/camera_info
/cmd_vel
/image_raw
/image_raw/compressed
/image_raw/compressedDepth
/image_raw/theora
/imu
/joy
/joy/set_feedback
/odom_raw
/parameter_events
/rosout
/scan
/servo_s1
/servo_s2
root@raspberrypi:~#
```

The main focus is on the topic of image data. Here we only parse RGB color images. Use the following commands to view their respective data information. Enter the following commands in the docke terminal.

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

```
header:
  stamp:
    sec: 1706003975
    nanosec: 527777000
  frame_id: default_cam
height: 480
width: 640
encoding: yuv422_yuy2
is_bigendian: 0
step: 1280
data:
- 86
- 131
- 85
- 129
- 87
- 130
- 86
- 130
- 91
- 128
- 92
- 130
- 94
- 128
- 98
- 130
- 105
- 128
- 107
- 133
- 110
- 127
- 108
- 130
```

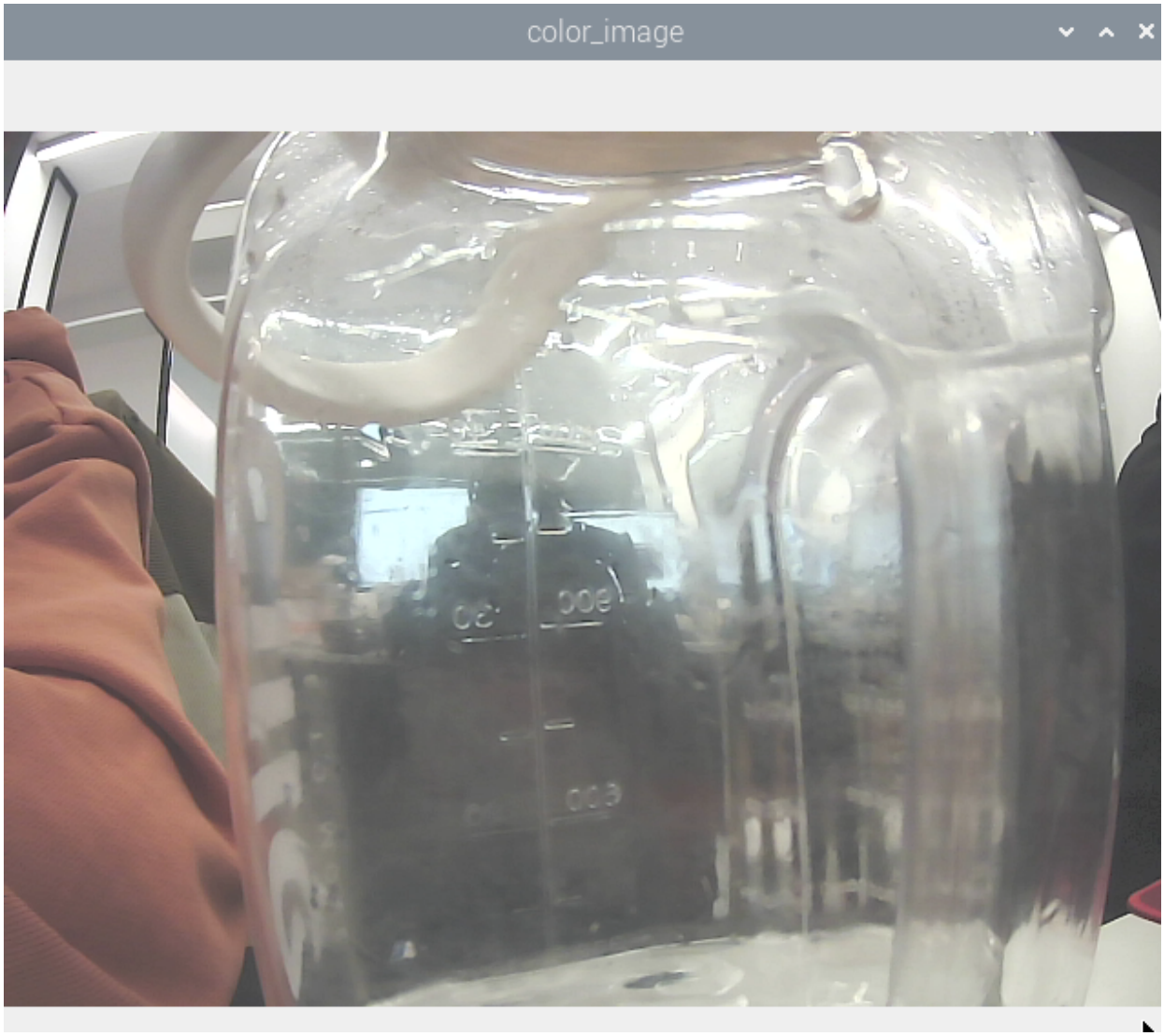
Here is the basic information of the image, an important value, **encoding**, the value here is **rgb8**, this value indicates that the encoding format of this frame of image is yuv422, this will be used for data conversion later. Reference required.

2、Subscribe to RGB image topic information and display RGB images

2.1、Run command

Enter the following commands in the docker terminal.

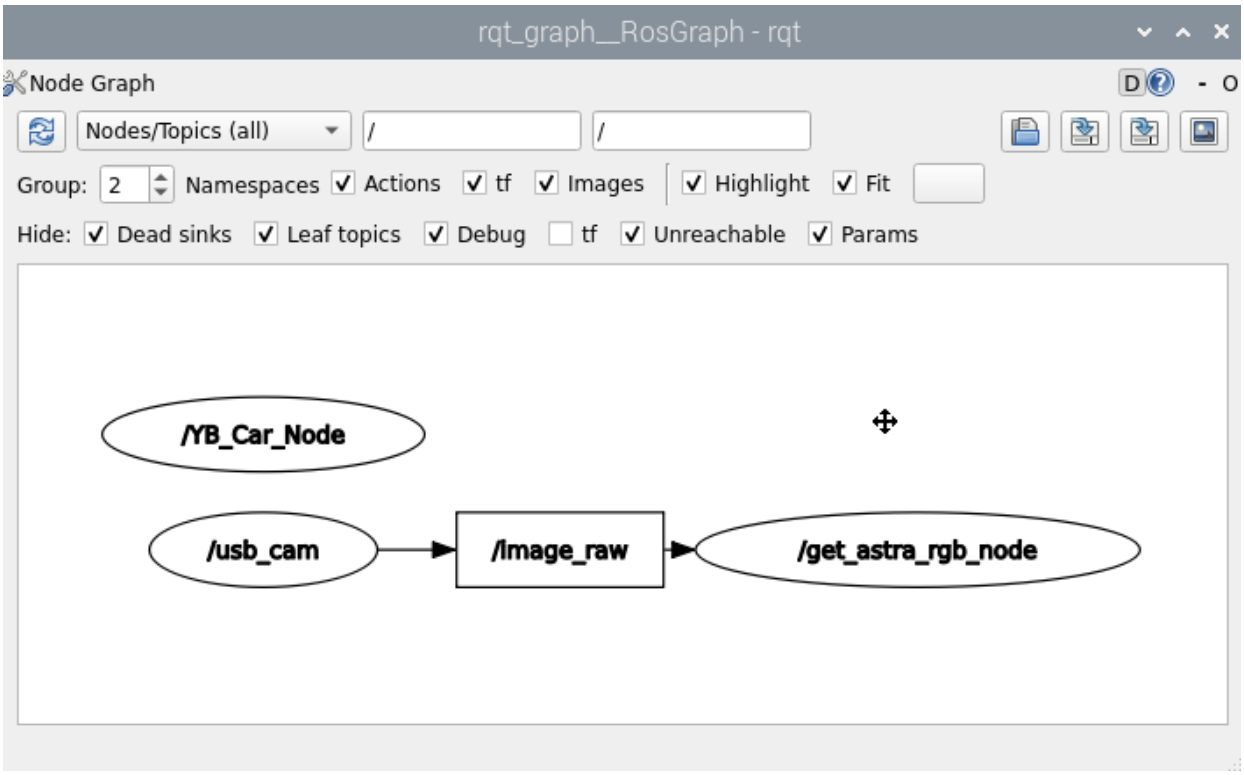
```
ros2 run usb_cam  usb_cam_node_exe
ros2 run yahboomcar_visual astra_rgb_image
```



2.2、View node communication diagram

Enter the following command in the docker terminal.

```
ros2 run rqt_graph rqt_graph
```



2.3、Core code analysis

Code reference path.

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

It can be seen from 2.2 that the /get_astra_rgb_node node subscribes to the topic of /mage_raw, and then through data conversion, the topic data is converted into image data and published. 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,'/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")
```

3、Subscribe to image data and then publish the transformed image data

3.1、Run command

```
#Run the publish image topic data node
ros2 run yahboomcar_visual pub_image
#Run the usb camera topic node
ros2 run usb_cam  usb_cam_node_exe
```

3.2、View node communication diagram

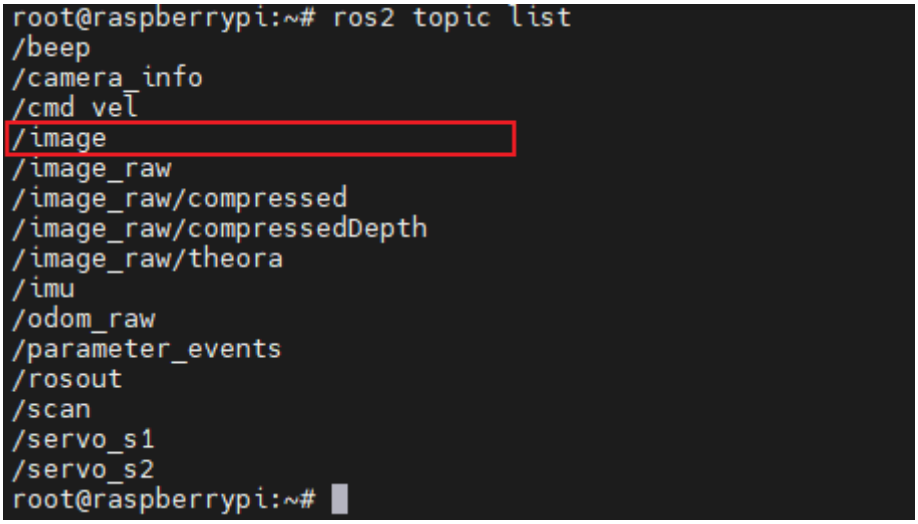
Enter the command in the docker terminal.

```
ros2 run rqt_graph rqt_graph
```

3.3、View topic data

First check which image topics are published, enter the following instructions in the docker terminal.

```
ros2 topoic list
```



The /image is the topic data we published. Use the following command to print and see the data content 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:
- 100
- 98
- 103
- 100
- 98
- 103
- 97
- 99
- 99
- 96
- 98
- 98
```

You can use the rqt_image_view tool to view images.

```
ros2 run rqt_image_view rqt_image_view
```

After opening, select the topic name/image in the upper left corner to view the image.

3.4、Core code analysis

code path

```
/root/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 topic data of /image_raw, and then converts it into image data. However, it also performs lattice conversion here to convert the image data into topic data, and then publishes it, which is the image topic data. ->Image data->Image topic data

```
#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 usb image topic data
self.sub_img = self.create_subscription(Image,'/image_raw',self.handleTopic,500)
#Image topic data publisher defined
self.pub_img = self.create_publisher(Image,'/image',500)
#msg is converted into image data imgmsg_to_cv2, where bgr8 is the image encoding
format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
#The image topic data (cv2_to_imgmsg) converted from the image data is then
published.
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
self.pub_img.publish(msg)
```