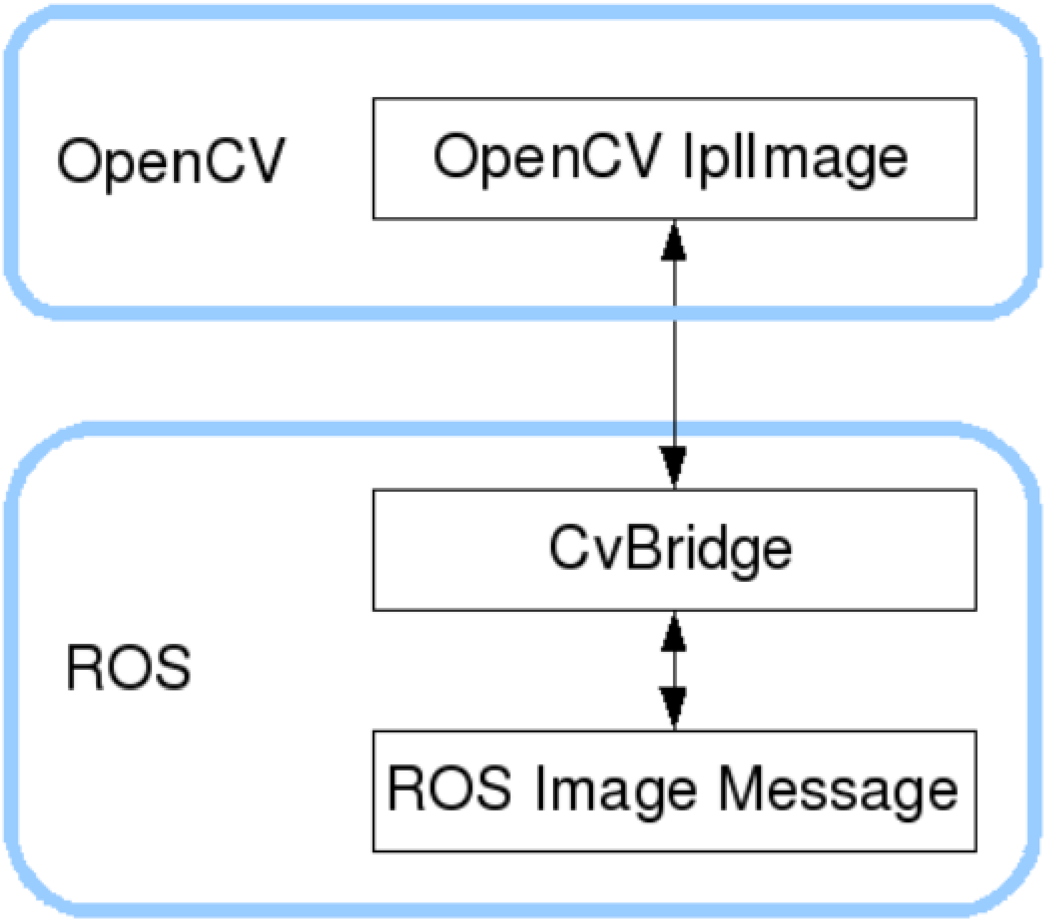
2、ROS+opencv application

This lesson uses a usb no-drive camera as an example

ROS with its own “sensor_ The msgs/Image” message format transmits images, which cannot be directly processed, but the provided 【CvBridge】can perfectly convert and convert image data formats. 【CvBridge】 is an ROS library that serves as a bridge between ROS and Opencv.

The conversion of Opencv and ROS image data is shown in the following figure：



2.1、Subscribe to image data and publish the converted image data

Run Command

```
#Run and publish image topic data nodes
ros2 run yahboomcar_visual pub_image

# Choose one of the following nodes
# Run USB camera topic node (with image display)
ros2 launch usb_cam demo_launch.py

# Run USB camera topic node (without image display)
ros2 launch usb_cam Demo_launch.py
```

Related points to note:

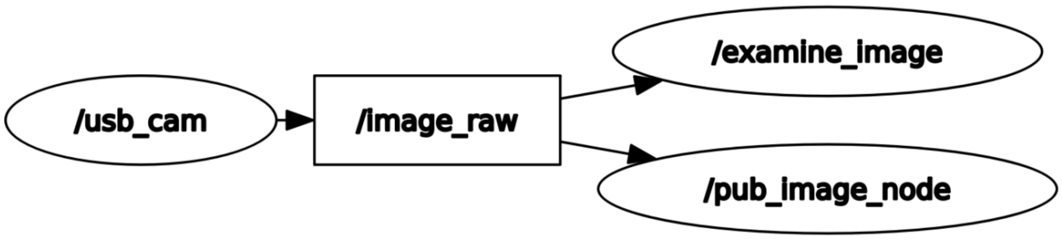
1、Before entering the docker container, make sure whether the usb no-drive camera device is connected, and add the camera device in the startup script. For the specific tutorial, please see [docker Tutorial ——5、Entering docker Container]

2、Due to the performance of some motherboards is not very good, running the usb camera topic node GUI display will be very card (such as jetson nano), can run the node without image display, and then use the rgt_image_view tool in the following tutorial to view the image

View the node communication diagram

Docker terminal input

```
ros2 run rqt_graph rqt_graph
```

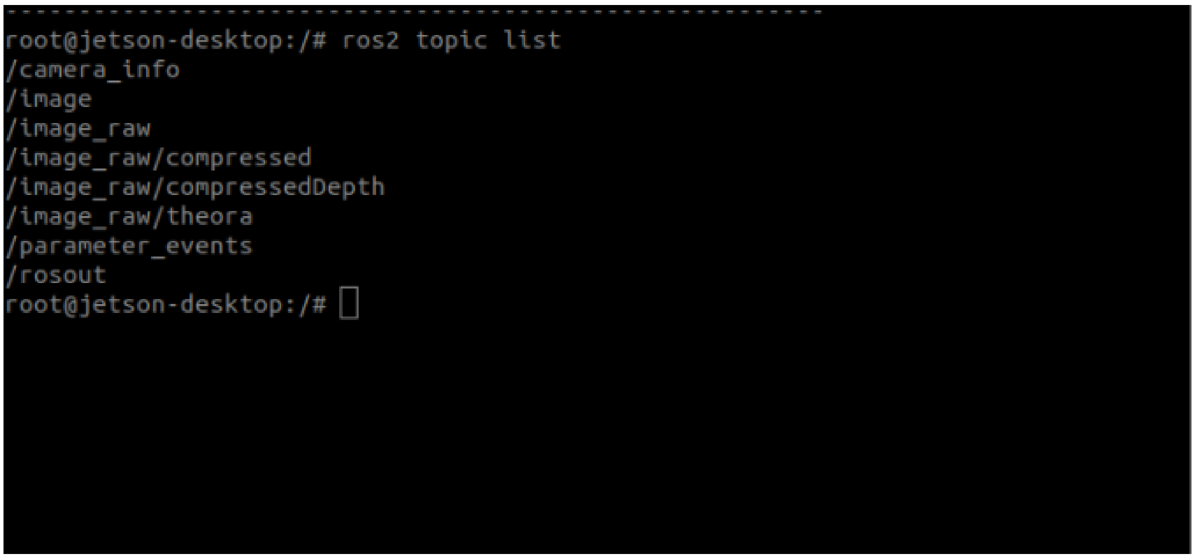


When you run the node directive without image display, the /examine_Image node is not displayed in rgt_graph

View topic list

Check which image topics are published, enter in docker terminal

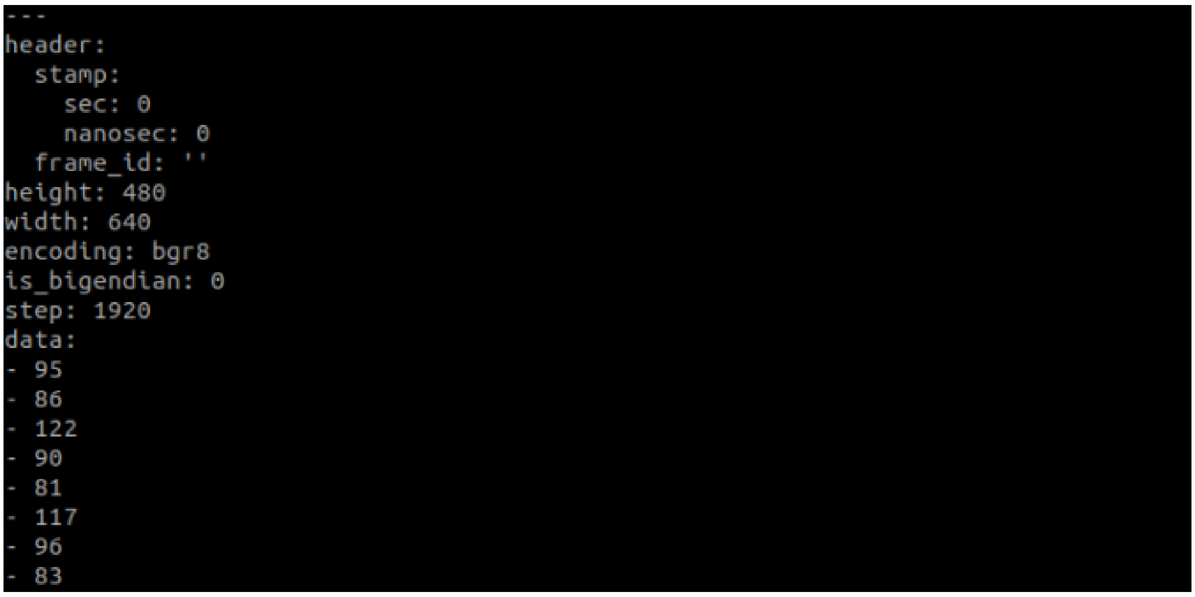
```
ros2 topic list
```



```
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:/# 
```

The/image is the topic data we have published. Use the following command to print and view the data content of this topic

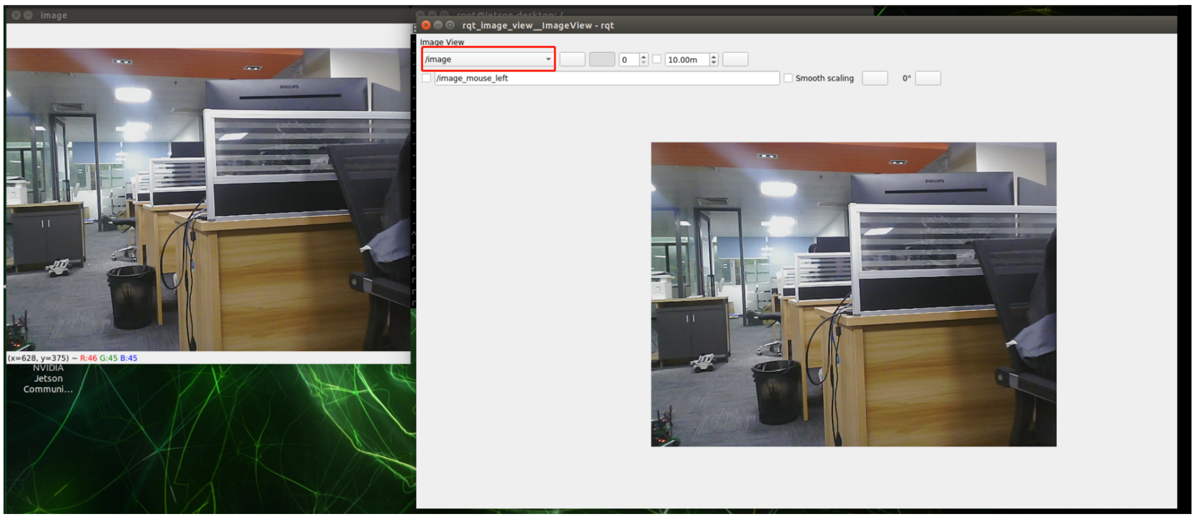
```
ros2 topic echo /image
```



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

You can use 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.

notes: Sometimes the first loading after selecting the topic is not out, you can end the process and re-enter the command to enter the interface to display the image

2.2、Core code parsing

code path

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/yahboomcar_visual/p
ub_image.py
```

The implementation steps are roughly the same as the previous two, and the program first subscribed to/image_ Raw's topic data is then converted into image data, but here we also perform lattice transformation to convert image data into topic data and publish it, which is image topic data ->image data ->image topic data.

```
#Import opecv library and cv_ Bridge Library
import cv2 as cv
from cv_bridge import CvBridge
#Creating CvBridge Objects
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)
#Defined image topic data publisher
self.pub_img = self.create_publisher(Image,'/image',500)
#Convert msg to image data imgmsg_ To_ CV2, where bgr8 is the image encoding
format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
#The image topic data (cv2uto_imgmsg) converted from image data is then
published
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
self.pub_img.publish(msg)
```