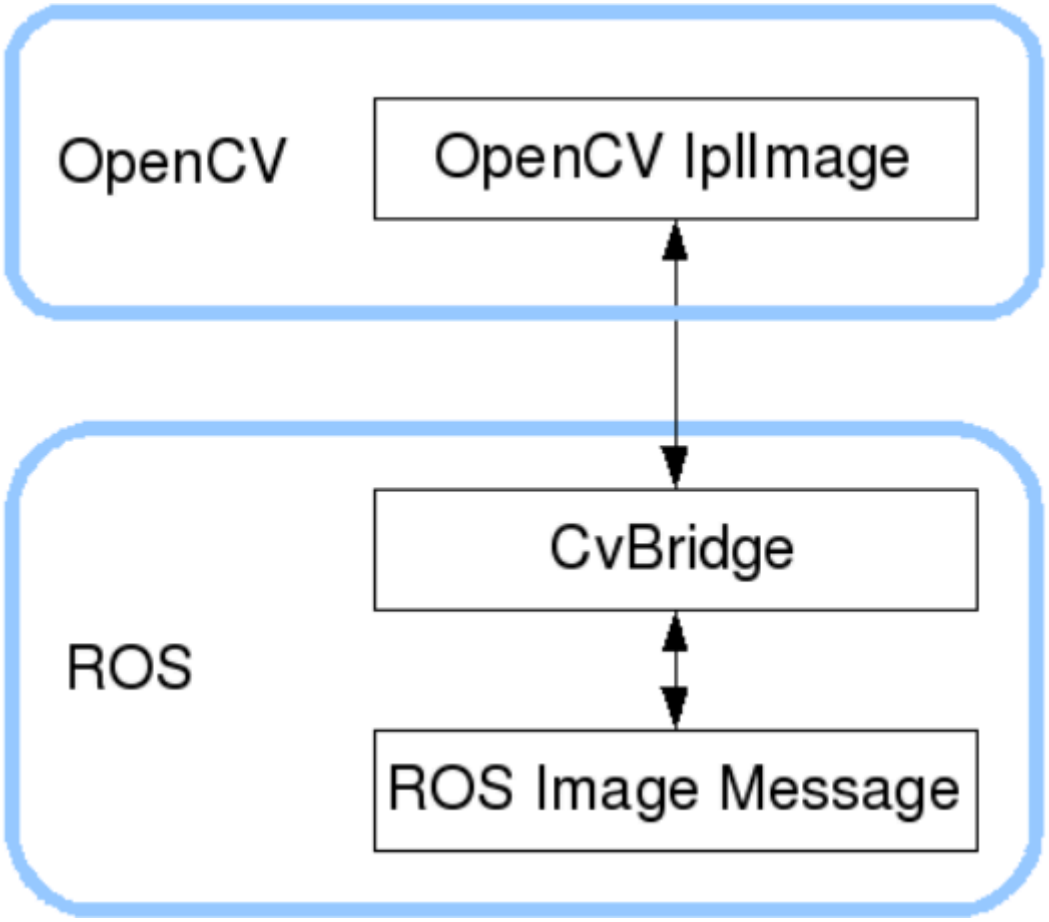
2. ROS+opencv application

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:



This lesson uses a case to show how to use CvBridge for data conversion.

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

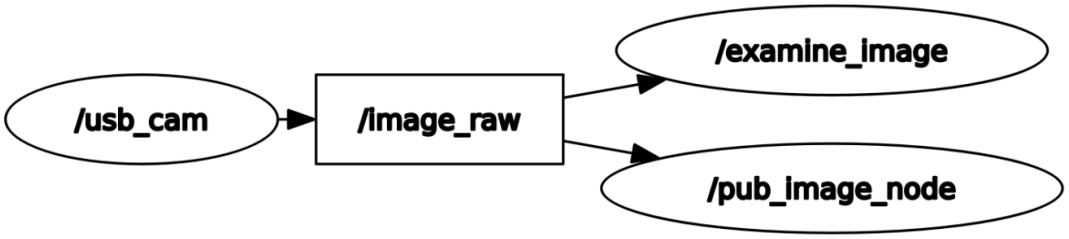
1.1. Run command

```
#Run publish image topic data node
ros2 run jetcobot_visual pub_image
#Run USB camera topic node
ros2 launch usb_cam camera.launch.py
```

1.2. View node communication diagram

Terminal input,

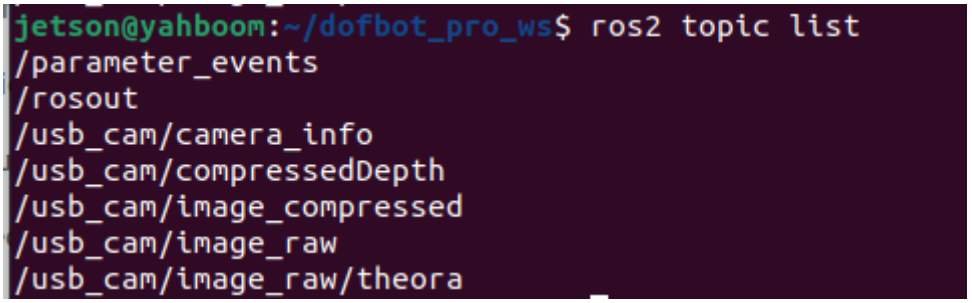
```
ros2 run rqt_graph rqt_graph
```



4.3. View topic data

First check which image topics are published, terminal input,

```
ros2 topic list
```

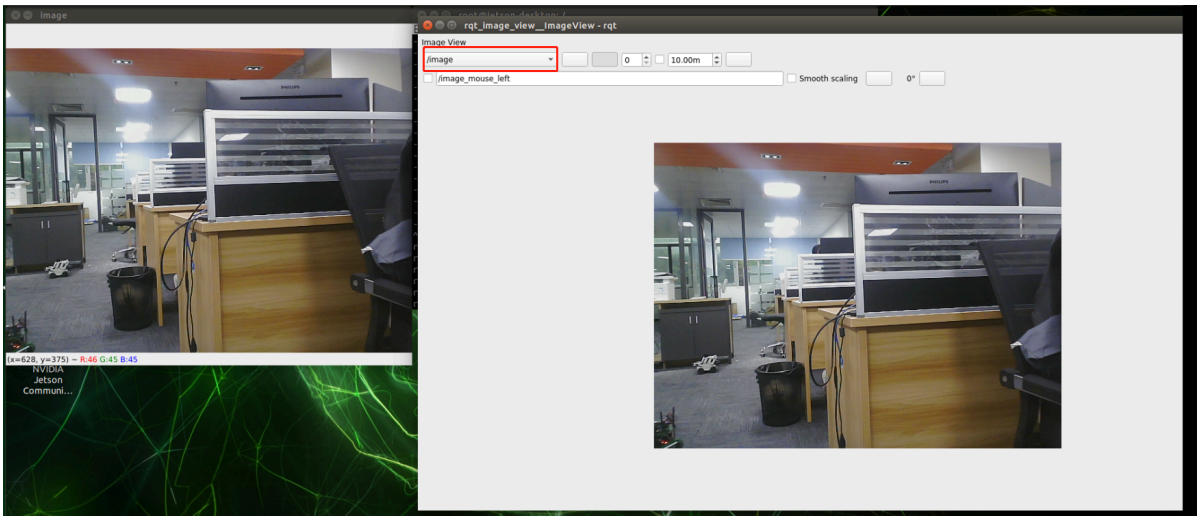


The /image is the topic data we published. Use the following command to print the data content of this topic.

```
ros2 topic echo /image
```

You can use the rqt_image_view tool to view the image.

```
ros2 run rqt_image_view rqt_image_view
```



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

4.4, core code analysis

Code path,

```
~/jetcobot_ws/src/jetcobot_visual/jetcobot_visual
```

The implementation steps are roughly the same as the previous two. The program first subscribes to the topic data of /usb_cam/image_raw and then converts it into image data. However, format conversion is also performed here to convert the image data into topic data and then publish it, that 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
#Create CvBridge object
self.bridge = CvBridge()
#Define a subscriber to subscribe to USB image topic data
self.sub_img =
self.create_subscription(Image,'/usb_cam/image_raw',self.handleTopic,500)
#Define the publisher of image topic data
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")
#Image data converted to image topic data (cv2_to_imgmsg) and then published
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
self.pub_img.publish(msg)
```