

## 3. ROS+OpenCV Foundation

---

### 3. ROS+OpenCV Foundation

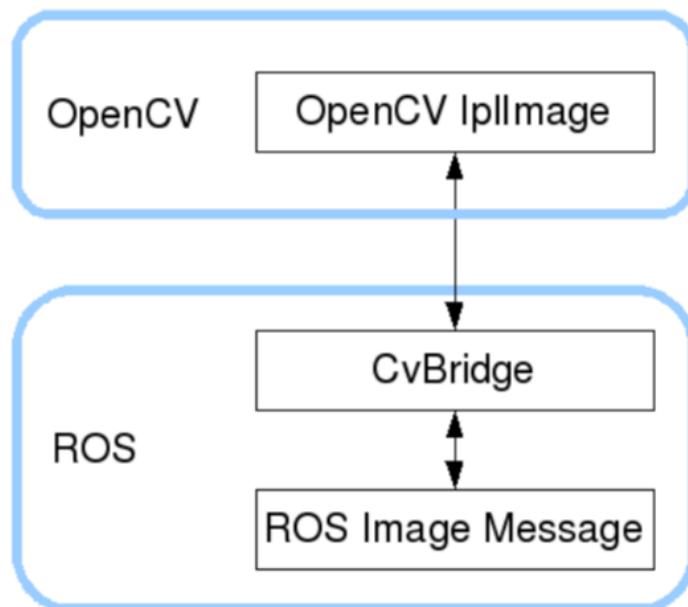
- 3.1 Description of program function
- 3.2 Application code reference path
- 3.3 Start the camera
- 3.4 Subscribe to RGB image topics and publish the converted image data
  - 3.4.1. Node Startup
  - 3.4.2. View the node communication graph
  - 3.4.3. Core source code parsing
- 3.5 Subscribe to Depth image topics and publish the transformed image data
  - 3.5.1. Node Startup
  - 3.5.2. View the node communication graph
  - 3.5.3. Core Source Code Resolution

### 3.1 Description of program function

---

ROS delivers images in its own sensor\_msgs/Image message format, which cannot be directly processed, but provides [CvBridge] that can perfectly convert and be converted image data formats. [CvBridge] is a ROS library that acts as 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 examples to show how to use CvBridge for data conversion, and using ROSboard web side to display topic data.

### 3.2 Application code reference path

---

After SSH connection car, the location of the function source code is located at,

#RGB图像示例#RGB image example

```
/userdata/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow  
/astra_rgb_image.py
```

#深度图像示例# Depth image example

```
/userdata/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow  
/astra_depth_image.py
```

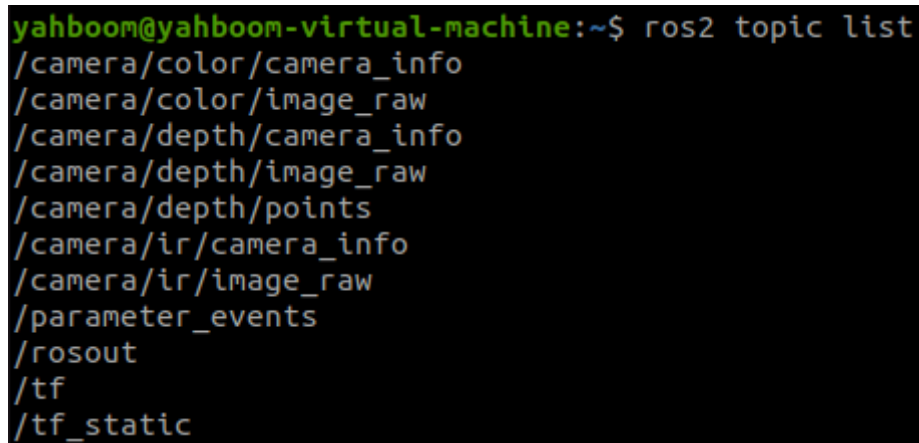
## 3.3 Start the camera

After SSH connects to the car, terminal input,

```
ros2 launch astra_camera astra_pro.launch.xml
```

To view the topic, run the following command on the VM terminal:

```
ros2 topic list
```



```
yahboom@yahboom-virtual-machine:~$ 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
```

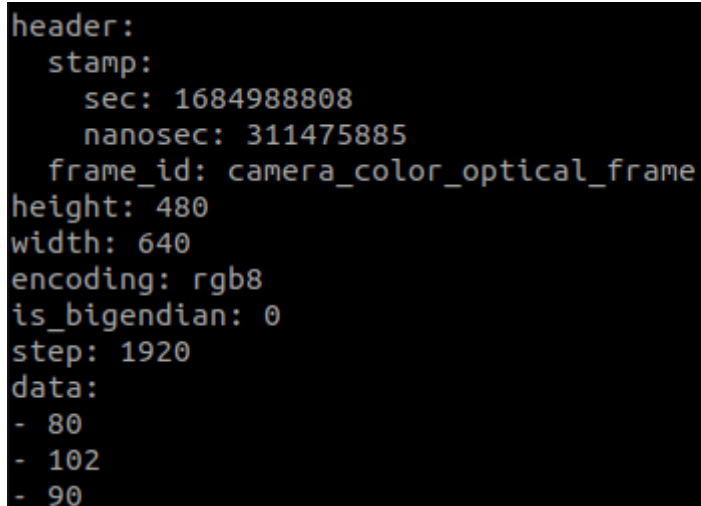
Mainly look at the image data topic, here only analyze the RGB color image and Depth image topic information, use the following command to view the respective data information, virtual machine terminal input,

#查看RGB图像话题数据内容

# View the RGB image topic data content

```
ros2 topic echo /camera/color/image_raw
```

Can capture a frame of RGB color image information,



```
header:  
  stamp:  
    sec: 1684988808  
    nanosec: 311475885  
  frame_id: camera_color_optical_frame  
height: 480  
width: 640  
encoding: rgb8  
is_bigendian: 0  
step: 1920  
data:  
- 80  
- 102  
- 90
```

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

Virtual machine terminal input,

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

Intercept depth image a frame of image data information,

```
header:
  stamp:
    sec: 1684994835
    nanosec: 994188364
  frame_id: camera_depth_optical_frame
height: 480
width: 640
encoding: 16UC1
is_bigendian: 0
step: 1280
data:
- 0
- 0
- 0
```

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

## 3.4 Subscribe to RGB image topics and publish the converted image data

### 3.4.1. Node Startup

After SSH connects to the car and starts the camera, terminal type,

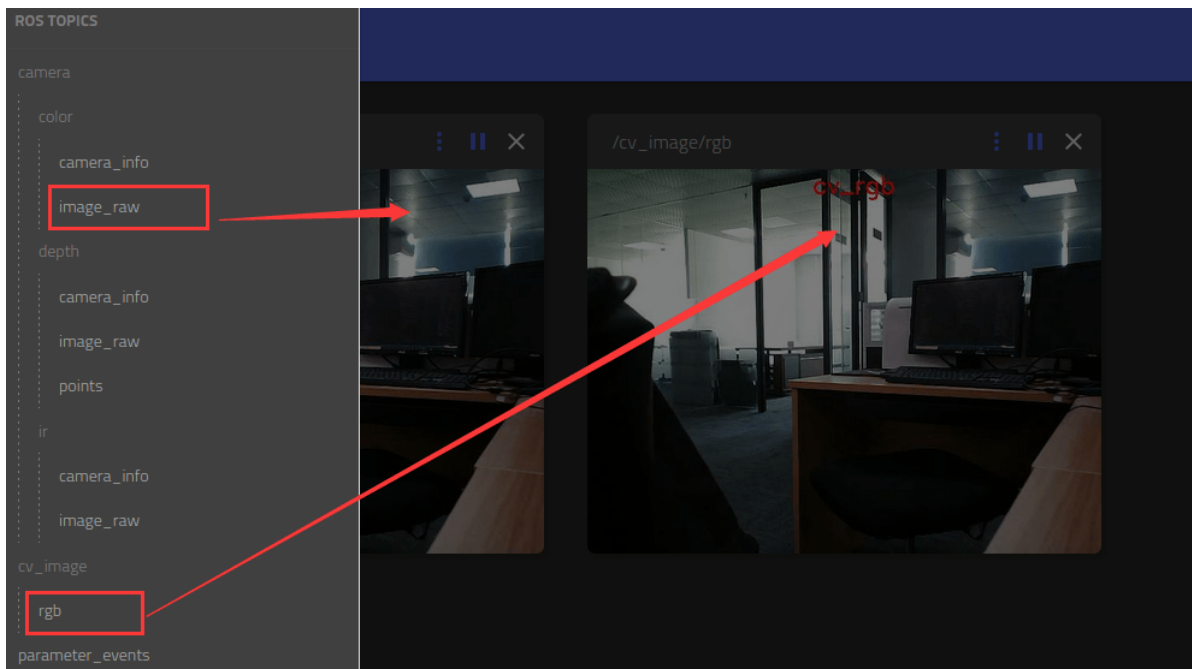
```
#RGB彩色图像显示节点
#RGB color image display node
ros2 run yahboomcar_imshow astra_rgb_image
```

Car terminal input,

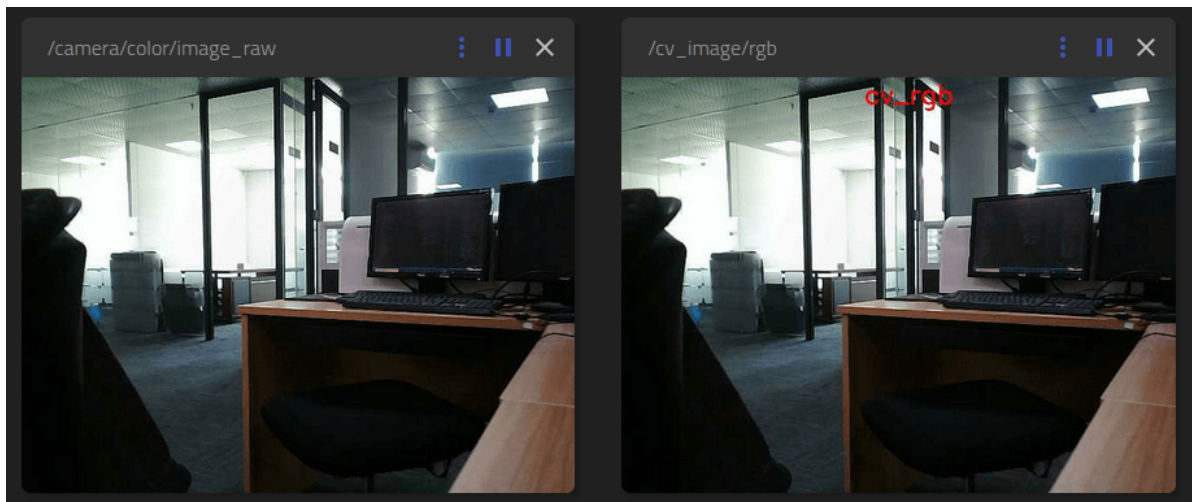
```
#启动ROSboard
# Start ROSboard
ros2 run rosboard rosboard_node
```

Open the browser on the PC side (note that the computer and the Rising Sun network must be in the same LAN), enter the URL: car IP:9999, for example, my car IP is 192.168.2.67, enter the URL in the browser on the virtual machine side to open the ROSboard webpage:

```
192.168.2.67:9999
```



Select /camera/color/image\_raw and /cv\_image/rgb topics, you can see the following screen,

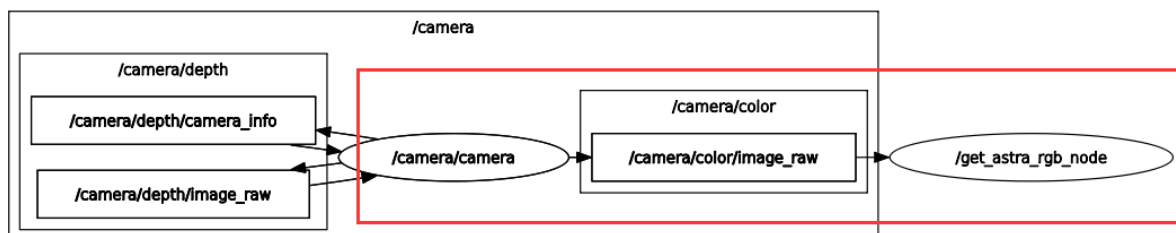


The two images are the same and keep in sync, only the /cv\_image/rgb image has more "cv\_rgb" characters.

### 3.4.2. View the node communication graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



### 3.4.3. Core source code parsing

Code reference path,

```
/userdata/yahboomcar_ws/src/yahboomcar_astra/yahboomcar_imshow/yahboomcar_imshow  
/astra_rgb_image.py
```

As can be seen from 3.2.2, /get\_astra\_rgb\_node subscribes to the topic of /camera/color/image\_raw, and then converts ROS topic data into opencv image data through CvBridge. After opencv processing, Then the processed image data is converted into ROS topic data through CvBridge and released, that is, ROS topic data ->opencv image data ->ROS topic data. The code is as follows,

```
# 导入opencv库以及cv_bridge库  
# Import opencv library and cv_bridge library  
import cv2 as cv  
from cv_bridge import CvBridge  
  
# 创建CvBridge对象  
# Create CvBridge objects  
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 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)  
# 定义一个发布者发布经处理后的图像话题数据  
# Define a publisher to publish processed image topic data  
self.pub_img = self.create_publisher(Image, '/cv_image/rgb', 10)  
  
# msg转换成cv图像数据，这里的bgr8是图像编码格式  
# msg converts to cv image data, where bgr8 is the image encoding format  
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")  
# 规范输入图像大小  
# Standardize the input image size  
frame = cv.resize(frame, (640, 480))  
# 为图像添加字符  
# Adds characters to the image  
cv.putText(frame, "cv_rgb", (280, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255),  
2)  
# 将处理后的cv图像数据转换成msg数据  
# Convert the processed cv image data into msg data  
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")  
# 发布转换后的图像  
# Post the converted image  
self.pub_img.publish(msg)
```

## 3.5 Subscribe to Depth image topics and publish the transformed image data

### 3.5.1. Node Startup

After SSH connects to the car and starts the camera, terminal type,

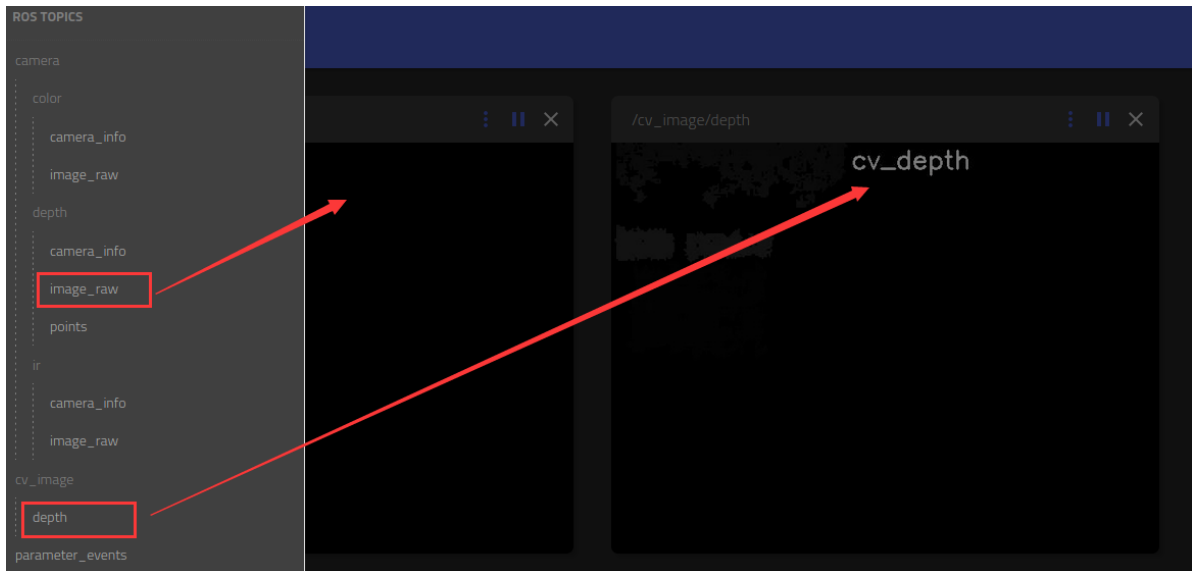
```
#深度图像显示节点  
# Depth image display node  
ros2 run yahboomcar_imshow astra_depth_image
```

Car terminal input,

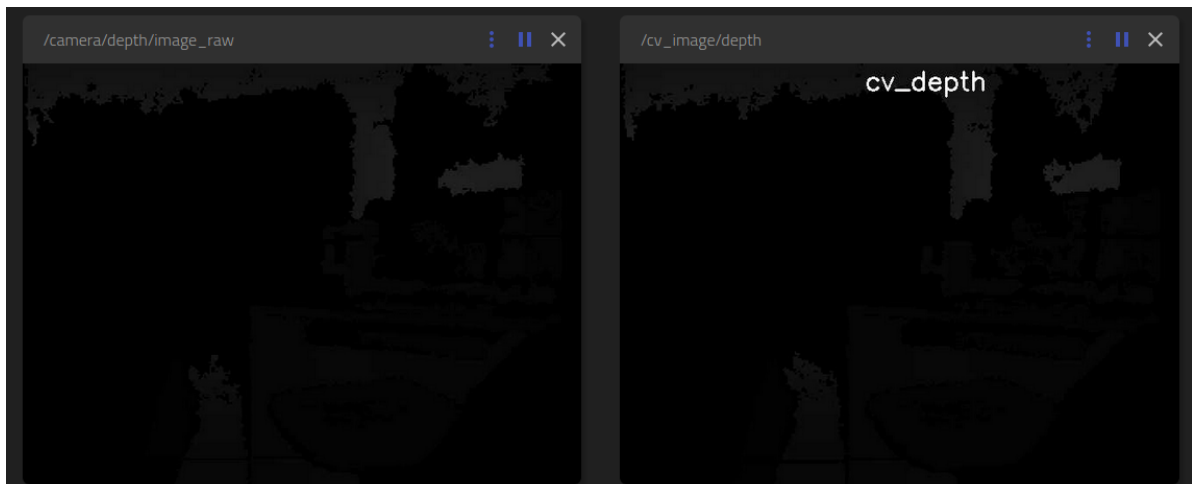
```
#启动ROSboard  
# Start ROSboard  
ros2 run rosboard rosboard_node
```

Open the browser on the PC side (note that the computer and the Rising Sun network must be in the same LAN), enter the URL: car IP:9999, for example, my car IP is 192.168.2.67, enter the URL in the browser on the virtual machine side to open the ROSboard webpage:

192.168.2.67:9999



Select the /camera/depth/image\_raw and /cv\_image/depth topics to see the following screen,

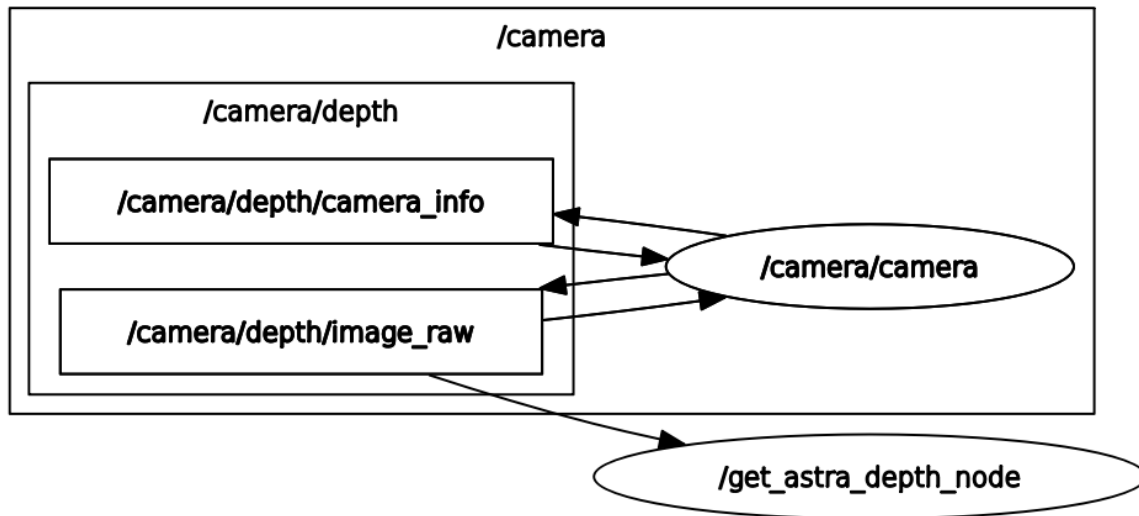


The two images are the same and synchronized, except that the /cv\_image/depth image has the "cv\_depth" character.

### 3.5.2. View the node communication graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



### 3.5.3. Core Source Code Resolution

Code reference path,

```
/userdata/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, subscribe to the topic data of /camera/depth/image\_raw published by the depth camera node, and then complete the conversion of ROS topic data ->opencv image data ->ROS topic data through CvBridge. The code is as follows,

```
# 导入opencv库以及cv_bridge库
# Import opencv library and cv_bridge library
import cv2 as cv
from cv_bridge import CvBridge

# 创建CvBridge对象
# Create CvBridge objects
self.bridge = CvBridge()
# 定义一个订阅者订阅深度相机节点发布的深度图像话题数据
# 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)
# 定义一个发布者发布经处理后的图像话题数据
# Define a publisher to publish processed image topic data
self.pub_img = self.create_publisher(Image, '/cv_image/depth', 10)

# msg转换成cv图像数据，这里的16UC1是图像编码格式
# msg converts to cv image data, where 16UC1 is the image encoding format
frame = self.bridge.imgmsg_to_cv2(msg, "16UC1")
# 规范输入图像大小
# Standardize the input image size
```

```
frame = cv.resize(frame, (640, 480))
# 为图像添加字符
# Adds characters to the image
cv.putText(frame, "cv_depth", (280, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (65536,
65536, 65536), 2)
# 将处理后的cv图像数据转换成msg数据
# Convert the processed cv image data into msg data
msg = self.bridge.cv2_to_imgmsg(frame, "16UC1")
# 发布转换后的图像
# Post the converted image
self.pub_img.publish(msg)
```