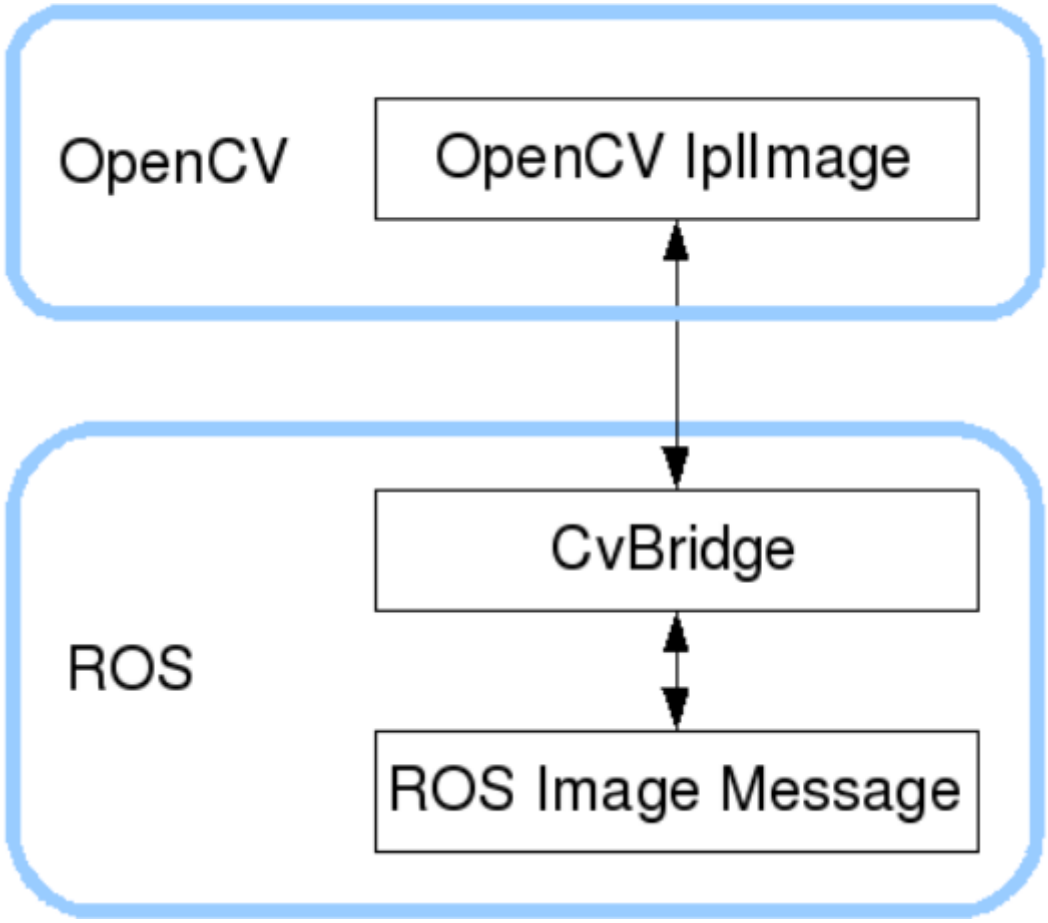
ROS+Opencv Basics

For PI5/JETSON NANO controllers, you must first enter the Docker container. For RDKX5 and Orin boards, this is not necessary.

ROS transmits images using its own sensor_msgs/Image message format, making direct image processing impossible. However, the provided [CvBridge] can seamlessly convert and convert image data formats. [CvBridge] is a ROS library that acts as a bridge between ROS and OpenCV.

The following figure shows image data conversion between OpenCV and ROS:



This lesson uses three examples to demonstrate how to use CvBridge for data conversion.

1. Depth Camera

Before driving the depth camera, the host machine must be able to identify the camera device.

1.1.1. Starting the Camera

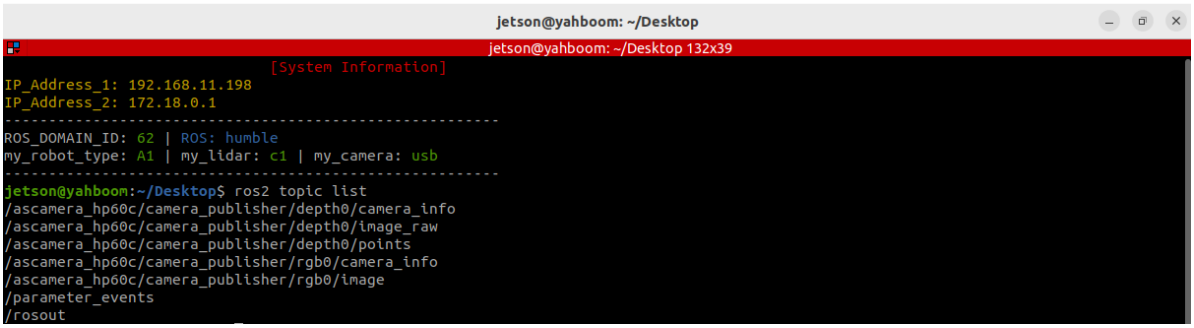
Enter in the terminal:

```
#nuwa camera
ros2 launch ascamera hp60c.launch.py
```

1.1.2. Viewing the Camera Topic

Enter in the terminal:

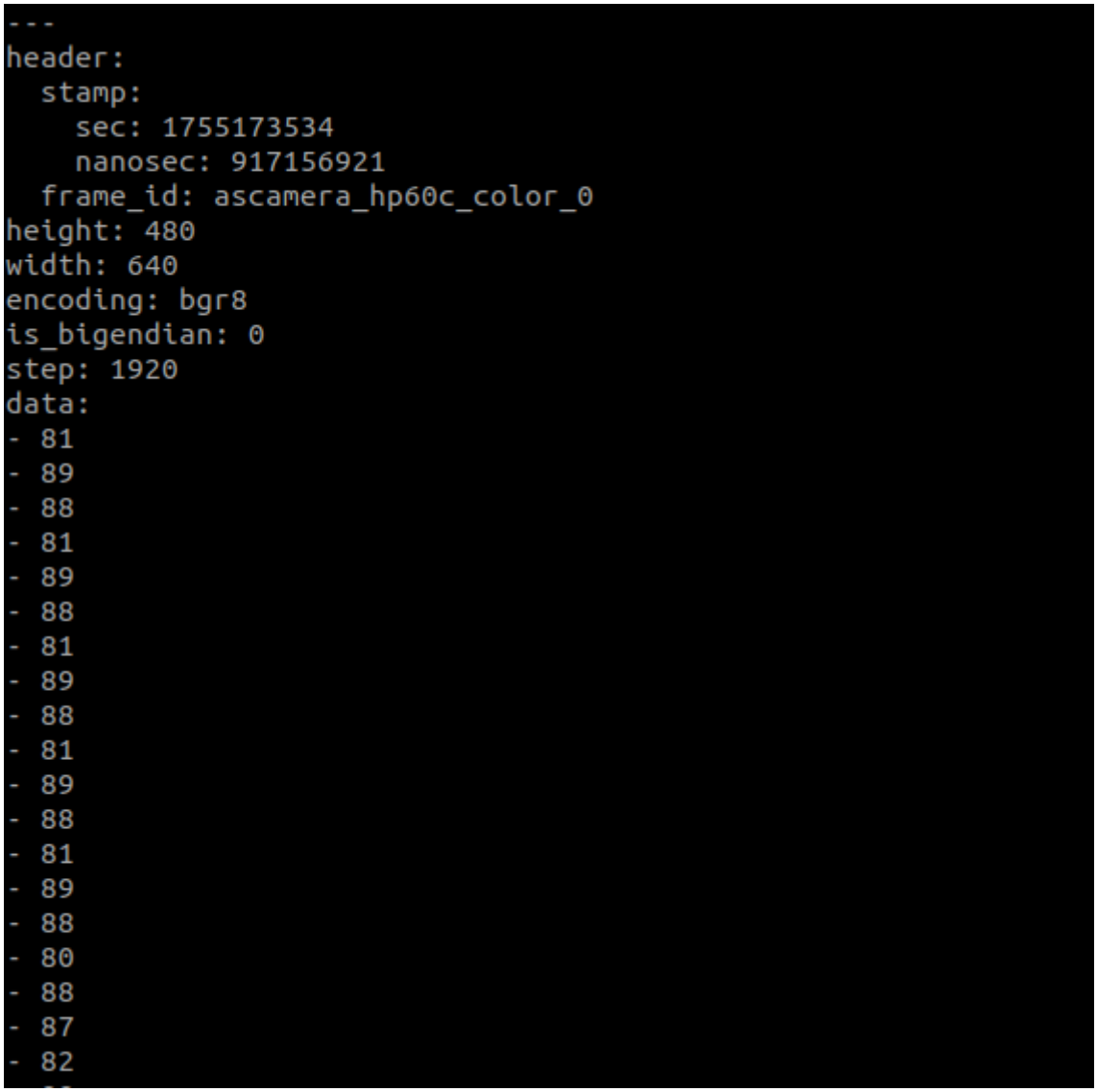
```
ros2 topic list
```



We will primarily focus on the image data topic. Here, we will only parse the RGB color image and depth image topic information. Use the following command to view their respective data information. Enter in the terminal:

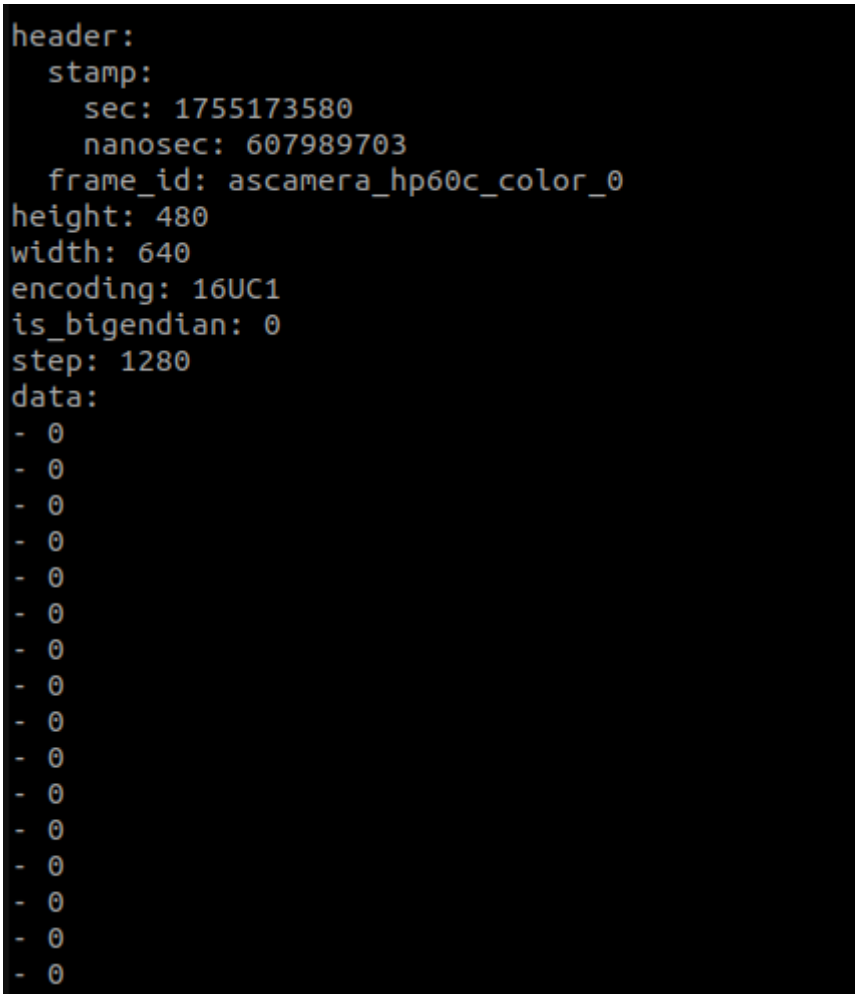
```
#View the RGB image topic data content
ros2 topic echo /ascamera_hp60c/camera_publisher/rgb0/image
# View the depth image topic data content
ros2 topic echo /ascamera_hp60c/camera_publisher/depth0/image_raw
```

First, let's take a look at a frame of RGB color image information.



This shows basic image information. An important value, **encoding**, is **rgb8**. This value indicates that the encoding format of this frame is RGB8. This will be used as a reference when performing data conversion later.

Similarly, below is the image data for a frame of the depth image.



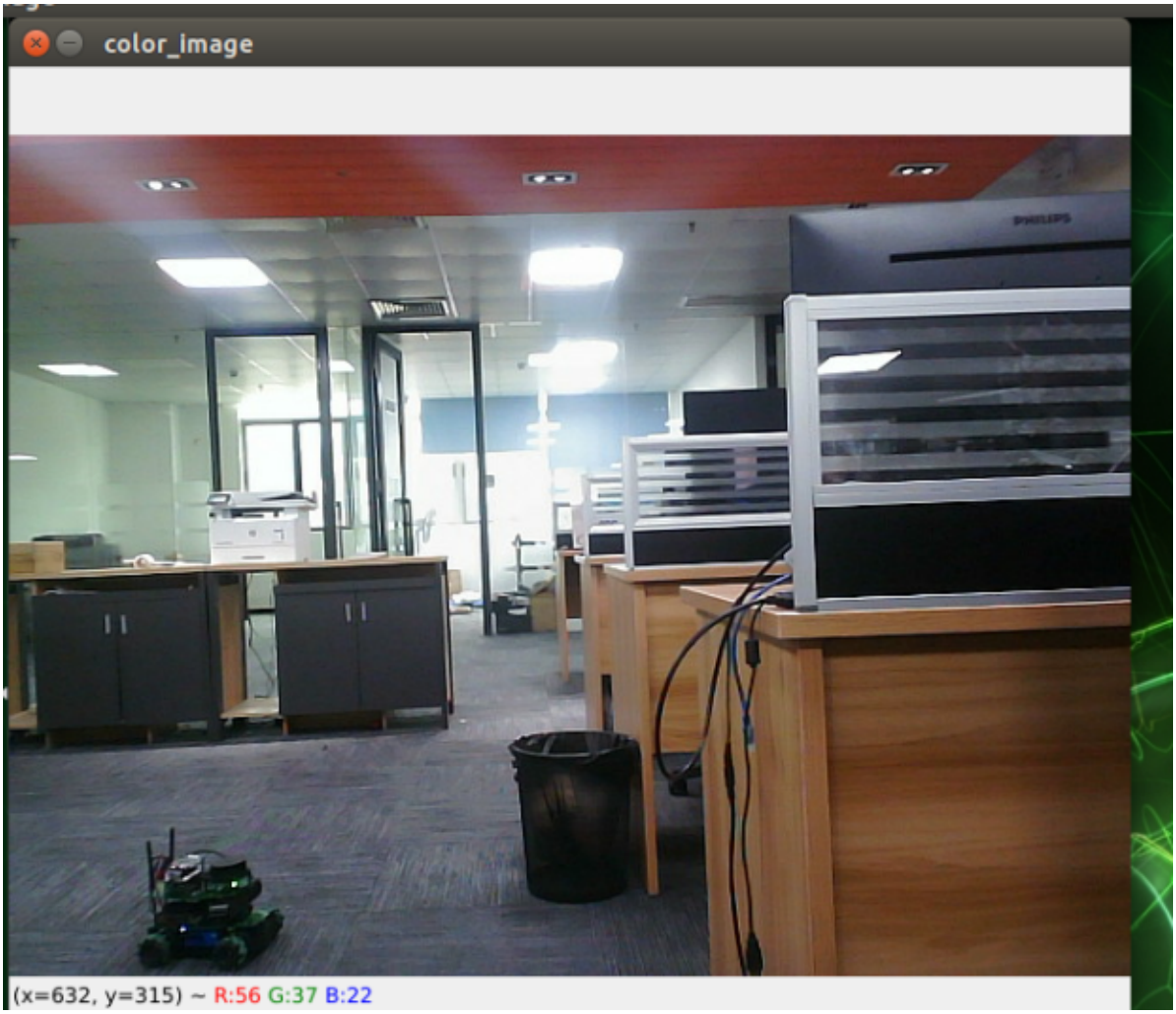
The encoding value here is **16UC1**.

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

2.1. Run the command

Enter the terminal:

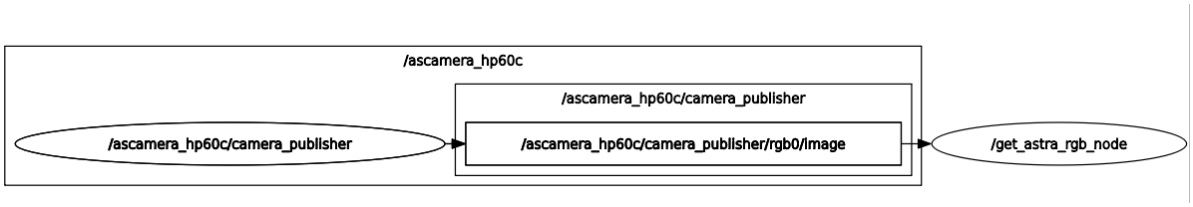
```
#RGB color image display node
ros2 run yahboomcar_vision astra_rgb_image
```



2.2. View the node communication graph

Enter the terminal:

```
ros2 run rqt_graph rqt_graph
```



2.3. Core Code Analysis

Code Reference Path:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_vision/yahboomcar_vision/astra_rgb_image.py
```

As shown in 2.2, the /get_astra_rgb_node node subscribes to the topic /ascamera_hp60c/camera_publisher/rgb0/image. Then, through data conversion, it converts the topic data into image data for publication. The code is as follows:

```
#Import the opecv and cv_bridge libraries
import cv2 as cv
from cv_bridge import CvBridge
#Create a 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,'/ascamera_hp60c/camera_publisher/rgb0/image',sel
f.handleTopic,100)
#Convert msg to image data. Here, bgr8 is the image encoding format.
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
```

3. Subscribe to the Depth image topic and display the Depth image

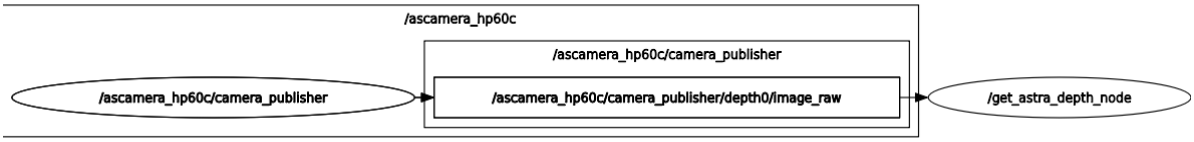
3.1. Run commands

```
#DEPTH depth image display node
ros2 run yahboomcar_vision astra_depth_image
```

3.2. View the node communication graph

In the terminal, enter:

```
ros2 run rqt_graph rqt_graph
```



3.3 Core Code Analysis

Code Reference Path:

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_vision/yahboomcar_vision/astra_depth_image.py
```

The basic implementation process is the same as for RGB color image display. It subscribes to the topic data published by the depth camera node, /ascamera_hp60c/camera_publisher/depth0/image_raw, and then converts the data into image data through data conversion. The code is as follows:

```
#Import the opecv and cv_bridge libraries
import cv2 as cv
from cv_bridge import CvBridge
#Create a CvBridge object
self.bridge = CvBridge()
# Define a subscriber to subscribe to the depth image topic data published by the depth camera node
self.sub_img =
self.create_subscription(Image,'/ascamera_hp60c/camera_publisher/depth0/image_raw',self.handleTopic,10)
# Convert msg to image data. Here, 32FC1 is the image encoding format.
frame = self.bridge.imgmsg_to_cv2(msg, "32FC1")
```

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

4.1. Run Commands

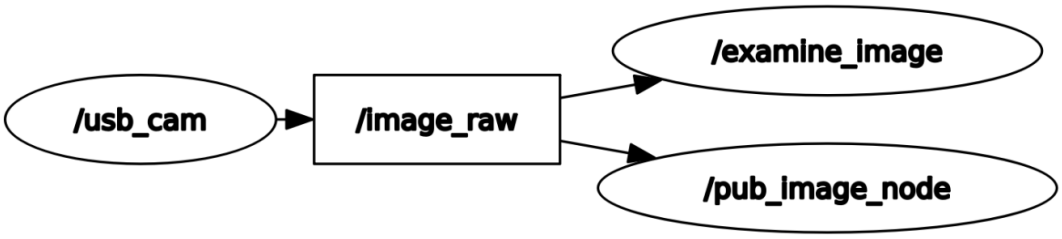
Here, we use a USB camera as an example.

```
# Run the node that publishes image data
ros2 run yahboomcar_vision pub_image
# Run the USB camera topic node
ros2 launch usb_cam camera.launch.py
```

4.2. View the node communication diagram

Terminal input,

```
ros2 run rqt_graph rqt_graph
```



4.3. View Topic Data

First, query which image topics have been published. Enter in the terminal:

```
ros2 topic list
```

```
jetson@yahboom:~$ ros2 topic list
/image
/parameter_events
/rosout
/usb_cam/camera_info
/usb_cam/compressedDepth
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/theora
```

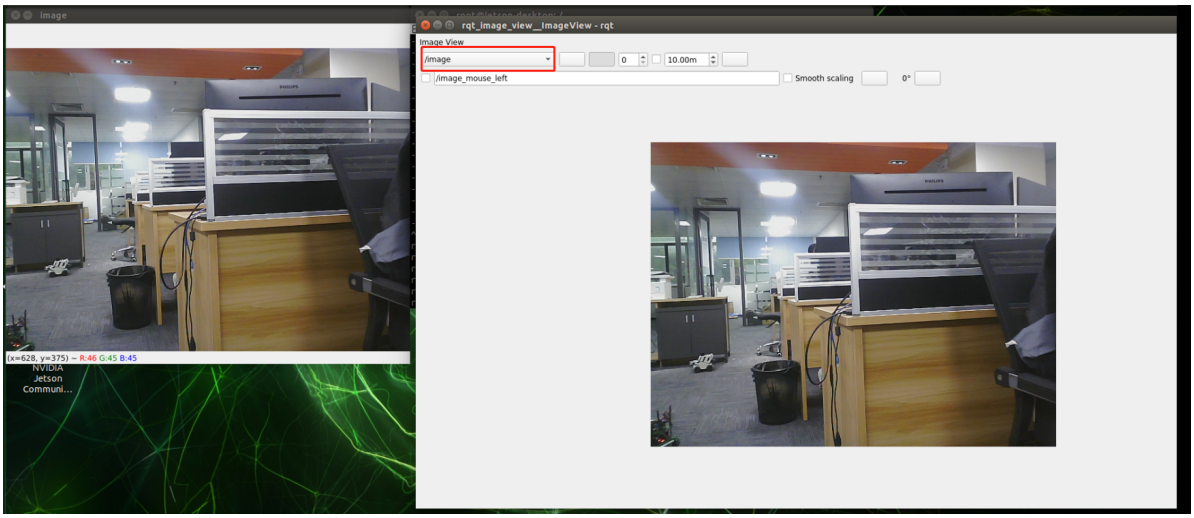
The /image topic is the data we published. Use the following command to print the data content of this topic:

```
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 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

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_vision
```

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, this program also performs a format conversion to convert the image data into topic data and then publishes it. In other words, the process goes from image topic data -> image data -> image topic data.

```
#Import the opecv and cv_bridge libraries
import cv2 as cv
from cv_bridge import CvBridge
#Create a CvBridge object
self.bridge = CvBridge()
#Define a subscriber to subscribe to the USB image topic data
self.sub_img =
self.create_subscription(Image,'/usb_cam/image_raw',self.handleTopic,500)
#Define the publisher of the image topic data
self.pub_img = self.create_publisher(Image,'/image',500)
#Convert msg to image data using imgmsg_to_cv2, where bgr8 is the image encoding
format
frame = self.bridge.imgmsg_to_cv2(msg, "bgr8")
#Convert the image data to the image topic data (cv2_to_imgmsg) and publish it
msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
self.pub_img.publish(msg)
```