4、 ROS+Opencv basic course

This lesson takes the Astra camera as an example, the ordinary camera is similar.

4.1、Overview

Wiki：http://wiki.ros.org/cv_bridge/

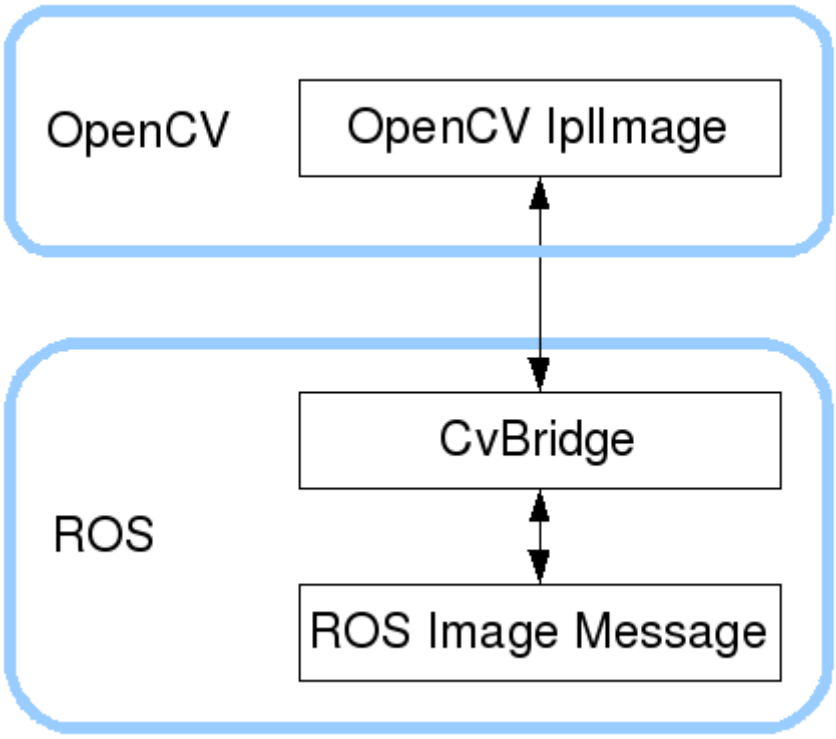
Tutorials：http://wiki.ros.org/cv_bridge/Tutorials

Source code：https://github.com/ros-perception/vision_opencv.git

Function package path：~/transbot_ws/src/transbot_visual

【CvBridge】is a ROS library, equivalent to a bridge between ROS and Opencv. It can perfectly convert and be converted image data format.

Opencv and ROS image data conversion is shown below:



This function package not only needs to use [CvBridge], but also needs [Opencv] and [PCL], so we need to perform the following configuration.

- package.xml

Add following content.

```
    <build_depend>sensor_msgs</build_depend>
    <build_export_depend>sensor_msgs</build_export_depend>
    <exec_depend>sensor_msgs</exec_depend>

    <build_depend>std_msgs</build_depend>
    <build_export_depend>std_msgs</build_export_depend>
    <exec_depend>std_msgs</exec_depend>

    <build_depend>cv_bridge</build_depend>
    <build_export_depend>cv_bridge</build_export_depend>
    <exec_depend>cv_bridge</exec_depend>

    <build_depend>transbot_msgs</build_depend>
    <build_export_depend>transbot_msgs</build_export_depend>
    <exec_depend>transbot_msgs</exec_depend>

    <exec_depend>image_transport</exec_depend>
```

【cv_bridge】：Image conversion dependent package.

【transbot_msgs】：Custom message dependency package.

- CMakeLists.txt

This file has a lot of configuration content, please check the source file for specific content.

4.2、Astra

4.2.1、Start up Astra camera

```
roslaunch astra_camera astraproplus.launch
```

View topic

```
rostopic list
```

Common topics are as follows

Topic name	Data type
/camera/depth/image_raw	sensor_msgs/Image
/camera/depth/image	sensor_msgs/Image
/camera/rgb/image_raw	sensor_msgs/Image

View the encoding format of the topic: rostopic echo +【topic】+encoding, for example

```
rostopic echo /camera/rgb/image_raw/encoding
rostopic echo /camera/depth/image_raw/encoding
```

```
yahboom@Yahboom:~$ rostopic echo /camera/rgb/image_raw/encoding
"bgr8"
---
"bgr8"
---
```

```
yahboom@Yahboom:~$ rostopic echo /camera/depth/image_raw/encoding
"16UC1"
---
"16UC1"
---
```

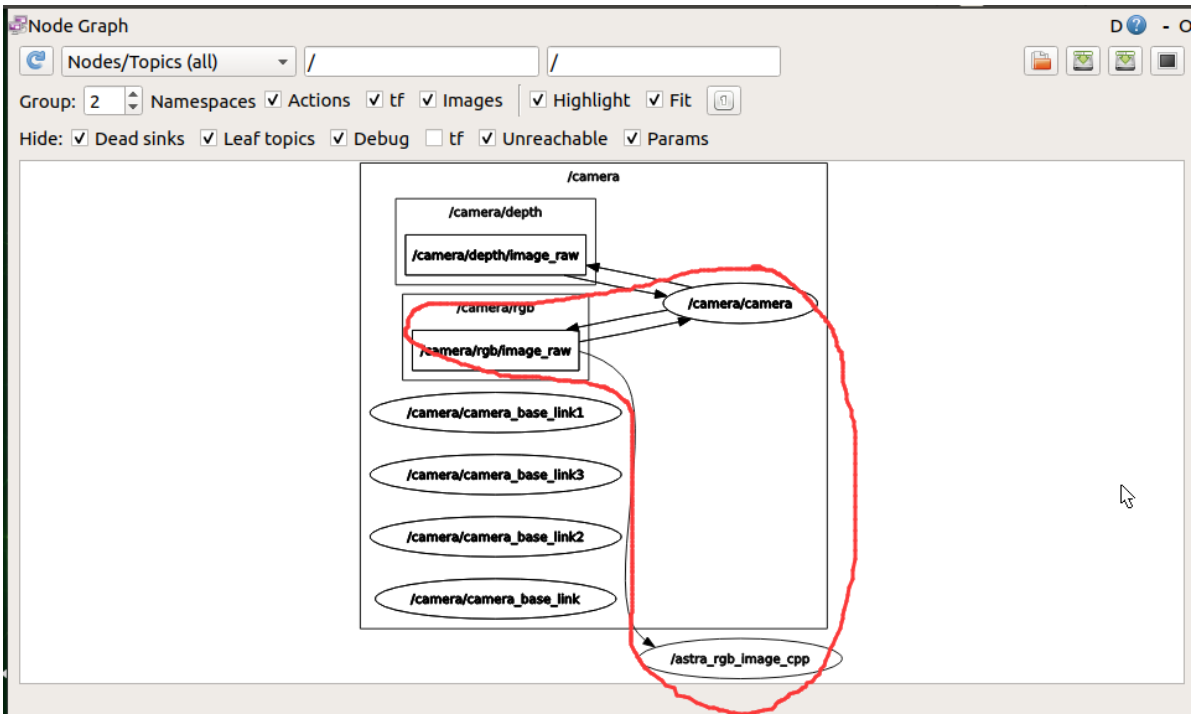
When ROS images are transmitted, factors such as the network, main control running speed, main control running memory, and huge video stream data may cause data packet loss and the topic cannot be obtained. This is normal. This phenomenon can be improved by changing to a better network.

4.2.2、Start the color map subscription node

```
roslaunch transbot_visual astra_get_rgb.launch    # launch
rosrun transbot_visual astra_rgb_image.py         # py
rosrun transbot_visual astra_rgb_image            # C++
```

View node graph

```
rqt_graph
```



【/astra_rgb_Image_cpp】is the node we wrote.

- py code analysis

Create subscribers: The subscribed topic is 【"/camera/rgb/image_raw"】, the data type is 【Image】, and the callback function is 【topic()】

```
sub = rospy.Subscriber("/camera/rgb/image_raw", Image, topic)
```

Use 【CvBridge】 for data conversion to ensure that the encoding format is correct.

```
frame = bridge.imgmsg_to_cv2(msg, "bgr8")
```

- c++ code analysis

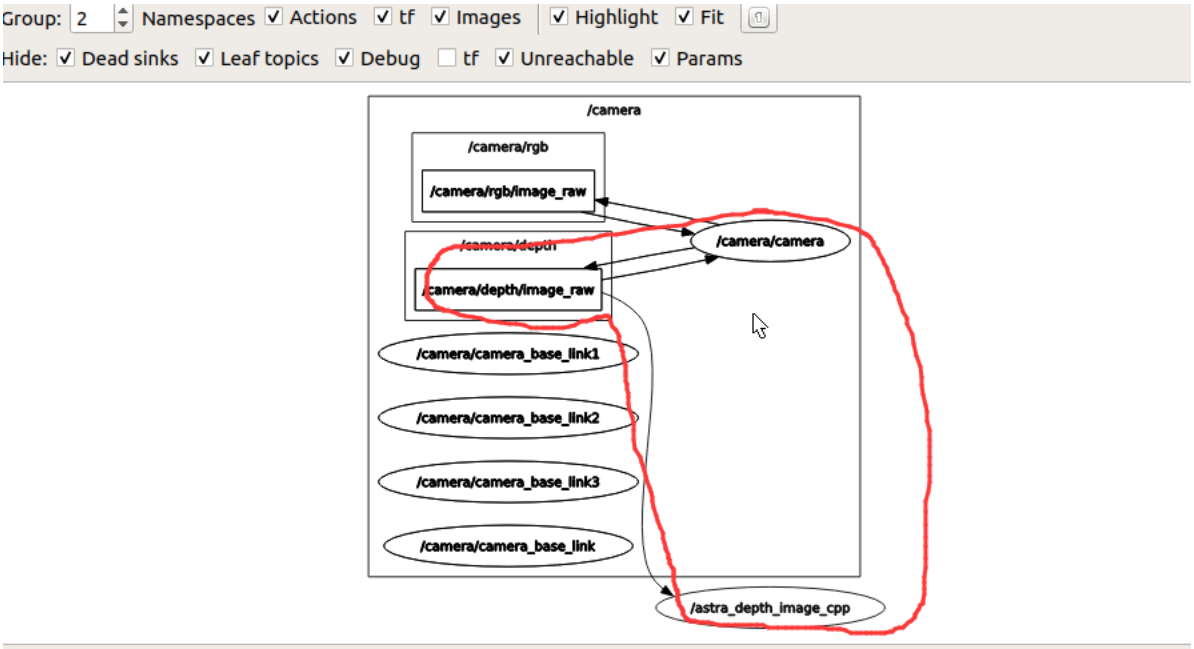
```
//Create a receiver.
ros::Subscriber subscriber = n.subscribe<sensor_msgs::Image>
("/camera/rgb/image_raw", 10, RGB_Callback);
//Create cv_bridge example
cv_bridge::CvImagePtr cv_ptr;
//Data conversion
cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
```

4.2.3、Start the depth map subscription node

```
roslaunch transbot_visual astra_get_depth.launch    # launch
rosrun transbot_visual astra_depth_image.py         # py
rosrun transbot_visual astra_depth_image            # C++
```

View node graph

```
rqt_graph
```



- py code analysis

Create subscribers: The subscribed topic is ["/camera/depth/image_raw"], the data type is [Image], and the callback function is [topic()]

```
sub = rospy.Subscriber("/camera/depth/image_raw", Image, topic)
```

Use 【CvBridge】 for data conversion to ensure that the encoding format is correct.

```
# Encoding format
encoding = ['16UC1', '32FC1']
# Can switch different encoding formats to test the effect
frame = bridge.imgmsg_to_cv2(msg, encoding[1])
```

- c++ code analysis

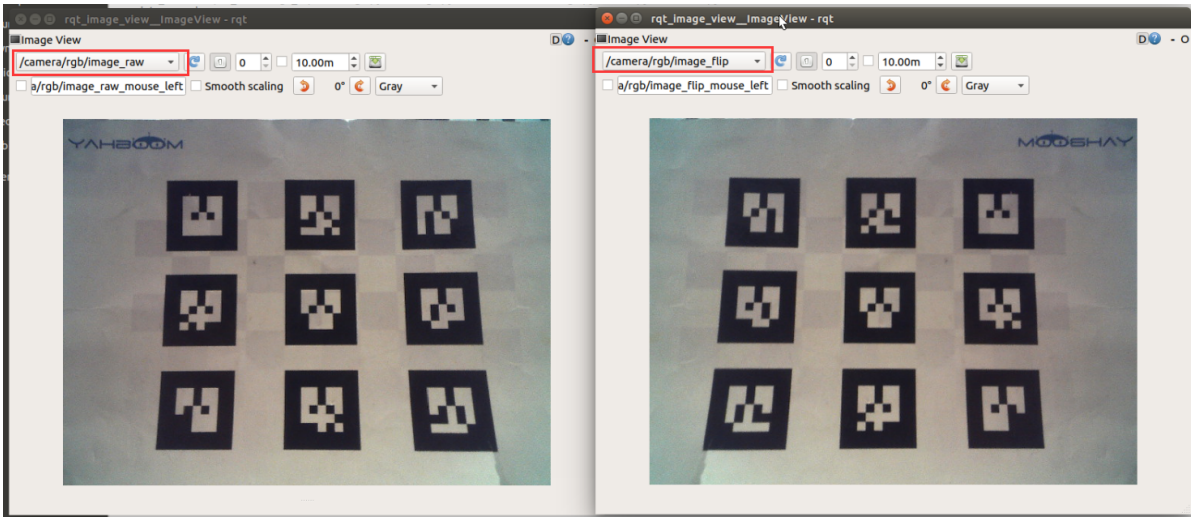
```
//Create a receiver.
ros::Subscriber subscriber = n.subscribe<sensor_msgs::Image>
("/camera/depth/image_raw", 10, depth_Callback);
//Create cv_bridge example
cv_bridge::CvImagePtr cv_ptr;
//Data conversion
cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::TYPE_16UC1);
```

4.2.4、Start color image inversion

```
roslaunch transbot_visual astra_image_flip.launch    # launch
rosrun transbot_visual astra_image_flip.py           # py
```

View node graph

```
rqt_image_view
```



- py code analysis

Two subscribers and two publishers are created here, one for processing general image data and one for processing compressed image data.

1) Create subscriber

The subscribed topic is【"/camera/rgb/image_raw"】, the data type is【Image】, and the callback function is【topic()】。

2) Create publisher

The published topic is【"/camera/rgb/image_flip"】, the data type is【Image】, Queue size【10】。

```
sub_img = rospy.Subscriber("/camera/rgb/image_raw", Image, topic)
pub_img = rospy.Publisher("/camera/rgb/image_flip", Image, queue_size=10)
```

3) Callback function

```
# Normally image transmission processing
def topic(msg):
```

```
    if not isinstance(msg, Image):
        return
    bridge = CvBridge()
    frame = bridge.imgmsg_to_cv2(msg, "bgr8")
    # Opencv processing image
    frame = cv.resize(frame, (640, 480))
    frame = cv.flip(frame, 1)
    # opencv mat ->  ros msg
    msg = bridge.cv2_to_imgmsg(frame, "bgr8")
    # The image is processed and released directly
    pub_img.publish(msg)
```

4.2.5、Publish image

Start up command as following content:

```
roslaunch transbot_visual astra_to_image.launch    # launch
rosrun transbot_visual astra_to_image.py           # py
```