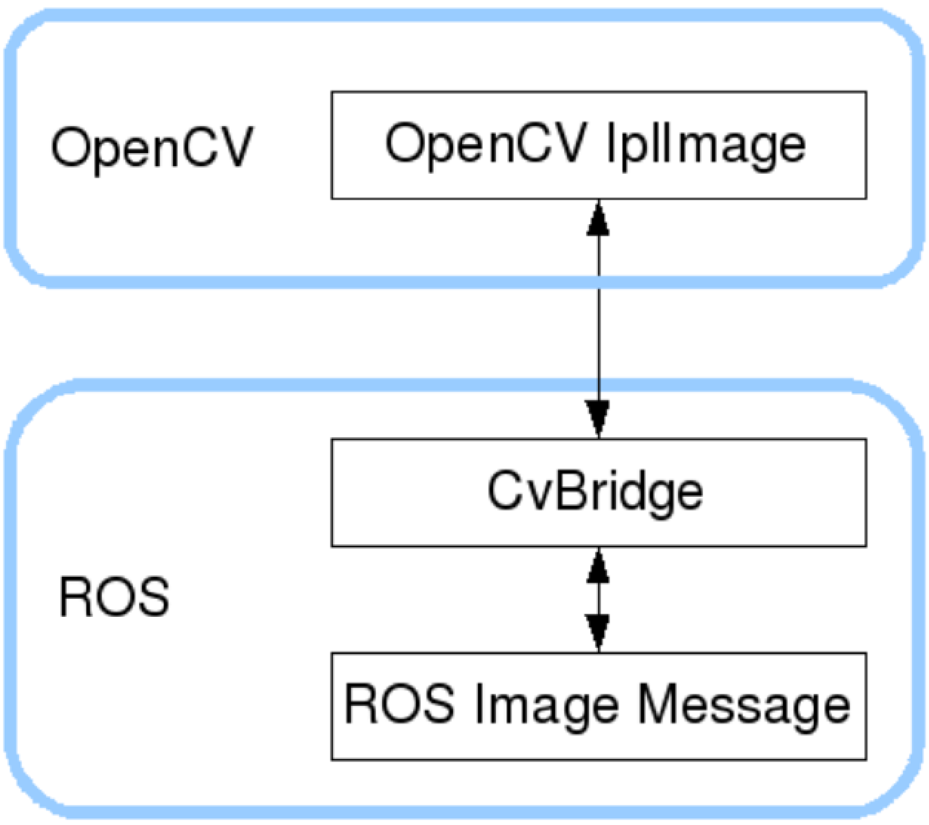
6.ROS+Opencv basics

6.1、Overview

ROS has already integrated Opencv3.0 and above during the installation process, so there is almost no need to think too much about the installation configuration.ROS transmits images in its own sensor_msgs/Image message format and cannot directly perform image processing, but the provided [CvBridge] can perfectly convert and be converted image data formats. [CvBridge] is a ROS library, equivalent to the bridge between ROS and Opencv.

Opencv and ROS image data conversion is shown in the figure below:



Although the installation configuration does not require too much consideration, the usage environment still needs to be configured, mainly the two files [package.xml] and [CMakeLists.txt]. This function package not only uses [CvBridge], but also requires [Opencv] and [PCL], so they are configured together.

- package.xml

  Add the following:

```
<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>
<exec_depend>image_transport</exec_depend>
```

【cv_bridge】： Image conversion dependency package.

- CMakeLists.txt

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

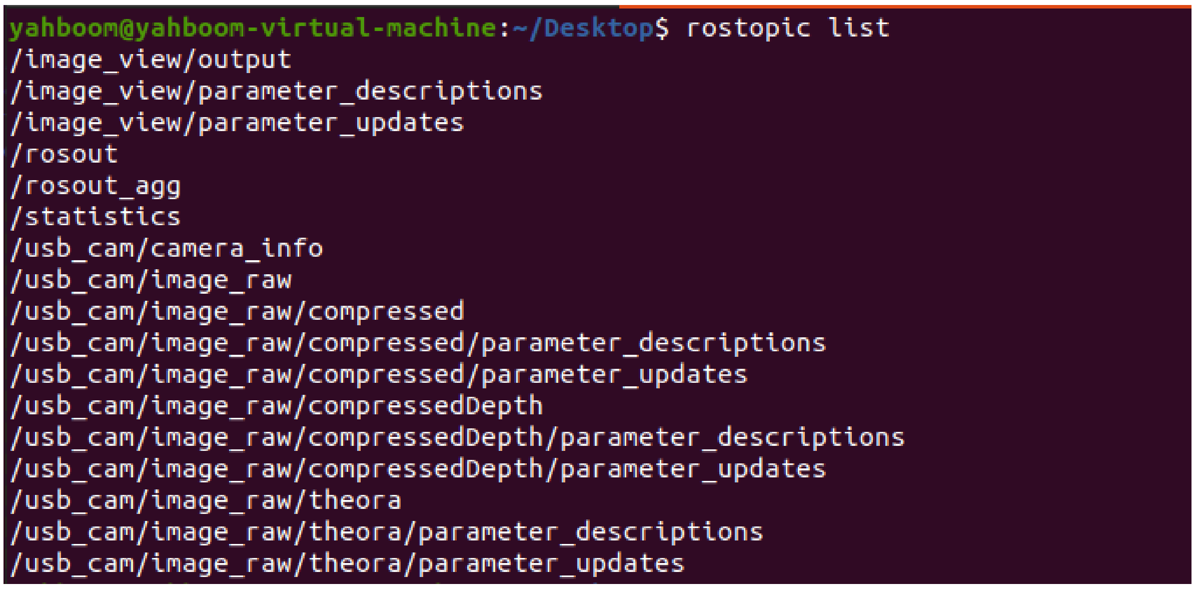
6.2 Start USB camera

Terminal input,

```
roslaunch usb_cam usb_cam-test.launch
```

View topics

```
rostopic list
```

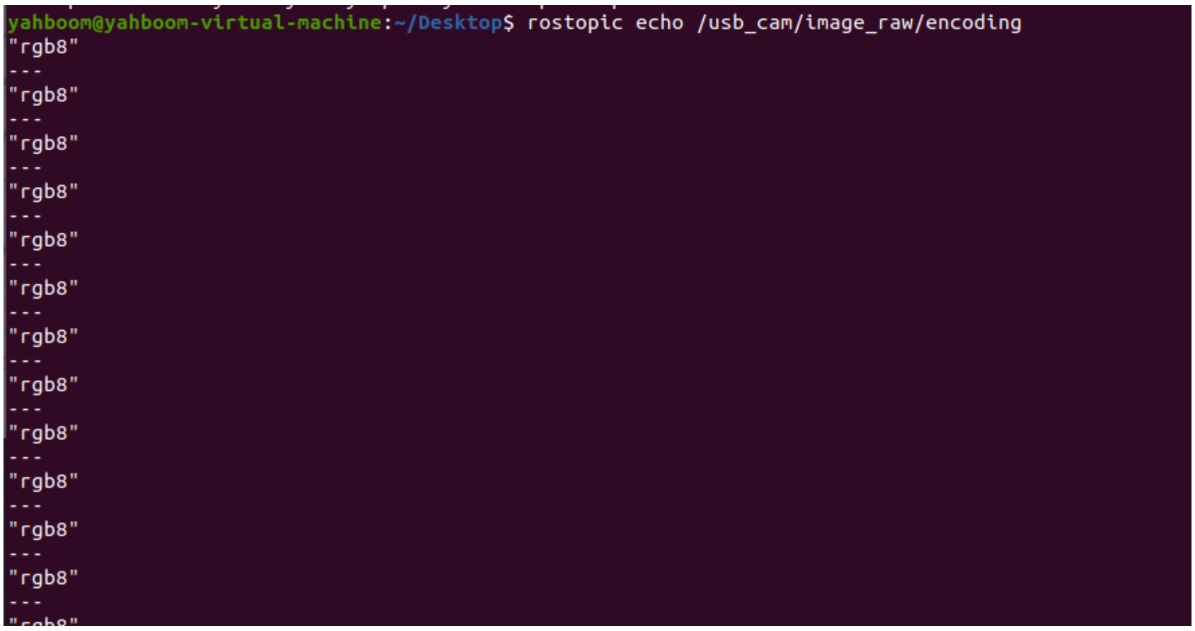


```
yahboom@yahboom-virtual-machine:~/Desktop$ rostopic list
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/rosout
/rosout_agg
/statistics
/usb_cam/camera_info
/usb_cam/image_raw
/usb_cam/image_raw/compressed
/usb_cam/image_raw/compressed/parameter_descriptions
/usb_cam/image_raw/compressed/parameter_updates
/usb_cam/image_raw/compressedDepth
/usb_cam/image_raw/compressedDepth/parameter_descriptions
/usb_cam/image_raw/compressedDepth/parameter_updates
/usb_cam/image_raw/theora
/usb_cam/image_raw/theora/parameter_descriptions
/usb_cam/image_raw/theora/parameter_updates
```

Commonly used ones are /usb_cam/image_raw and /usb_cam/image_raw/compressed. The former is a normal image, and the latter is a compressed image.

Check the encoding format of the topic: rostopic echo +[topic]+encoding, for example,

```
rostopic echo /usb_cam/image_raw/encoding
```



```
yahboom@yahboom-virtual-machine:~/Desktop$ rostopic echo /usb_cam/image_raw/encoding
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
"rgb8"
---
```

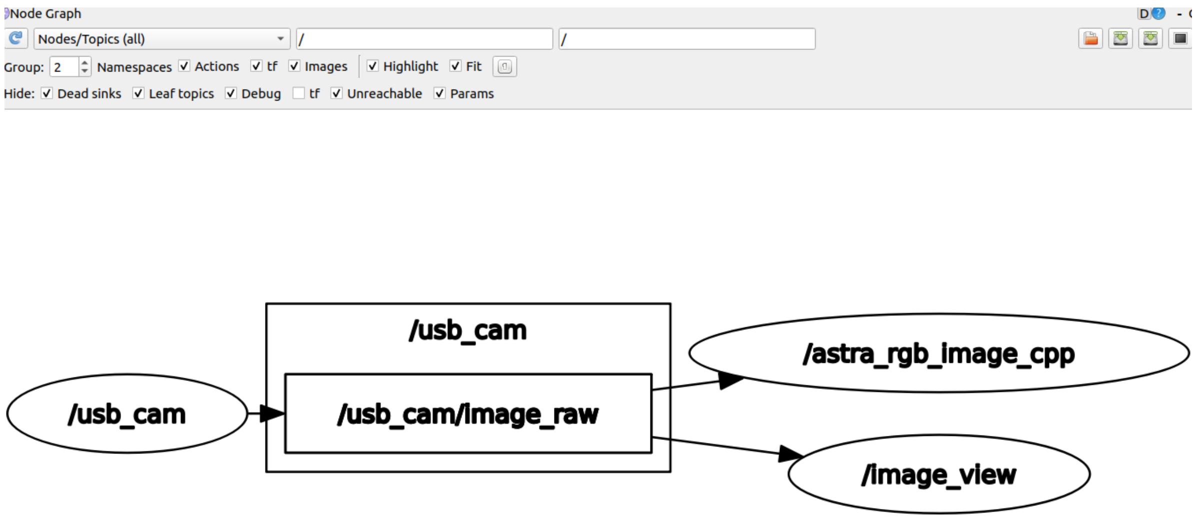
6.3 Start the color picture subscription node

```
roslaunch usb_cam usb_cam-test.launch
rosrun dofbot_visual get_rgb_image
```

6.4 View the node graph

Terminal input,

```
rqt_graph
```



6.5 Source code

Source code location：/home/yahboom/dofbot_ws/src/dofbot_visual/src/get_rgb_image.cpp

```
//
// Created by yahboom on 2021/4/29.
```

```
//

#include "ros/ros.h"
#include <sensor_msgs/Image.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <opencv2/highgui/highgui.hpp>

using namespace std;
using namespace cv;

void RGB_Callback(const sensor_msgs::ImageConstPtr &msg) {
    cv_bridge::CvImagePtr cv_ptr;
    try {
        cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
        imshow("color_image", cv_ptr->image);
        waitKey(1);
    } catch (cv_bridge::Exception &e) {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }
}

int main(int argc, char **argv) {
    //ROS node initialization
    // The ROS node is initialized
    ros::init(argc, argv, "astra_rgb_image_cpp");
    //Create node handle
    // Create a node handle
    ros::NodeHandle n;
    //Create a receiver.
    // Create a receiver.
    ros::Subscriber subscriber = n.subscribe<sensor_msgs::Image>
("/usb_cam/image_raw", 10, RGB_Callback);
    //Delay according to cycle frequency
    // According to the cycle frequency delay
    ros::spin();
    return 0;
}
```