在进行机器人ROS开发时，我们常常会添加摄像头进行图像采集，并且对于采集的图像会进行一些处理，例如添加某些图案或者对图像进行识别，这个时候就需要对相机进行标定，然后使用OpenCV对图像进行处理，再由CvBridge工具发送OpenCV处理后的图像给ROS，最后完成ROS中使用处理后的图像。

下面将通过一个示例来阐释ROS中进行图像处理这一个过程。

## 1. 启动摄像头

        要进行图像处理，首先需要获取到图像，因此需要启动摄像头。首先运行以下命令找到系统中存在的视频设备：

ls /dev/ | grey video

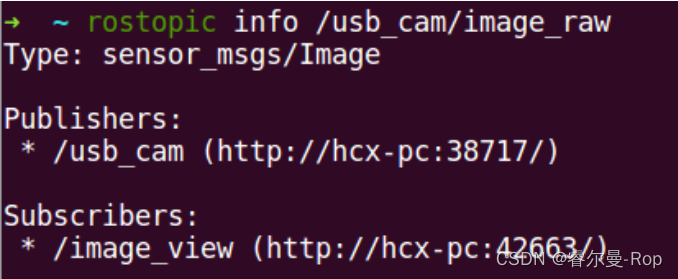
        若有输出，则说明USB相机可以连接上。

        在ROS中启动USB摄像头采集图像，一般需要安装驱动。使用以下命令安装：

sudo apt-get install ros-melodic-usb-cam  
sudo apt-get install ros-melodic-image-view

        安装成功后，可以查看图像话题信息。终端输入

rostopic info /usb\_cam/image\_raw

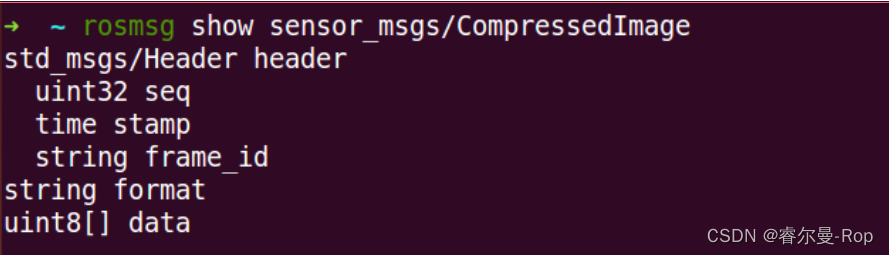


        若是类似realsense、Astra此类厂家摄像头，则参照其官网的驱动安装教程，然后启动相机，达到同样的图像采集效果。

        图像话题的消息类型为sensor\_msgs/Image类型，这是ROS定义的一种摄像头原始数据消息类型。查看图像消息：

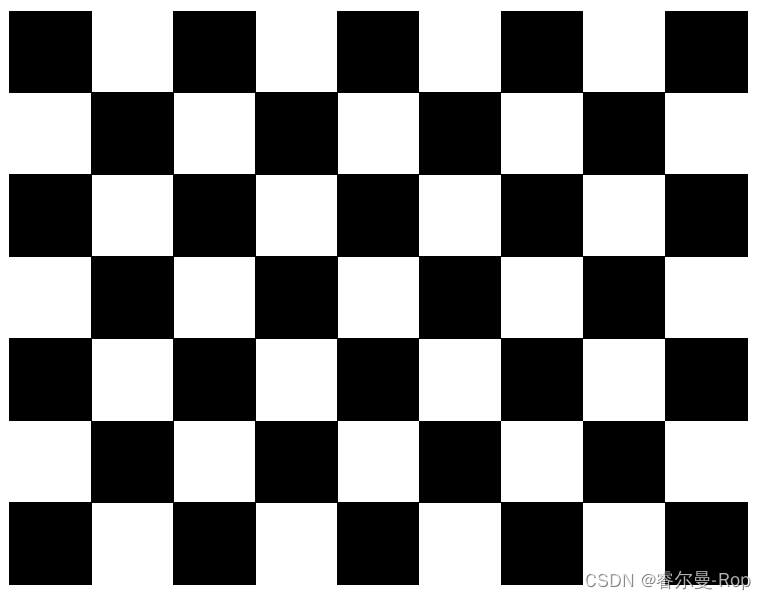
rosmsg show sensor\_msgs/Image

         1080\*720分辨率的摄像头产生一帧图像数据大小为：3\*1080\*720=2764800字节。实际应用中需要将图像压缩，ROS压缩图像消息类型为：sensor\_msgs/CompressedImage；



## 2. 摄像头标定

        摄像头这种精密仪器对光学器件的要求较高，由于摄像头的内部与外部的一些原因，产生的物体图像会发生畸变，为避免数据源造成的误差，需要对摄像头进行标定。一般使用如下方格标定板标定相机参数，因此需要提前将标定纸打印出来。



 然后安装标定包

sudo apt-get install ros-melodic-camera-calibration

启动摄像头

roslaunch robot\_vision usb\_cam.launch

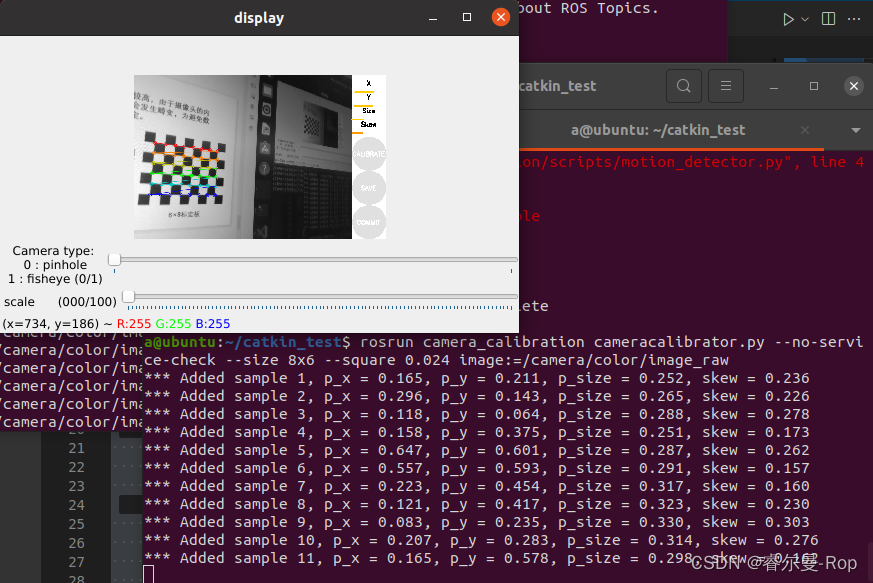
启动标定包

rosrun camera\_calibration cameracalibrator.py --no-service-check --size 8x6 --square 0.024 image:=/usb\_cam/image\_raw camera:=/usb\_cam

size：标定板内部角点个数；

square：每个棋盘格边长，单位（米）；

image和camera：设置摄像头发布的图像话题



 标定参数含义：

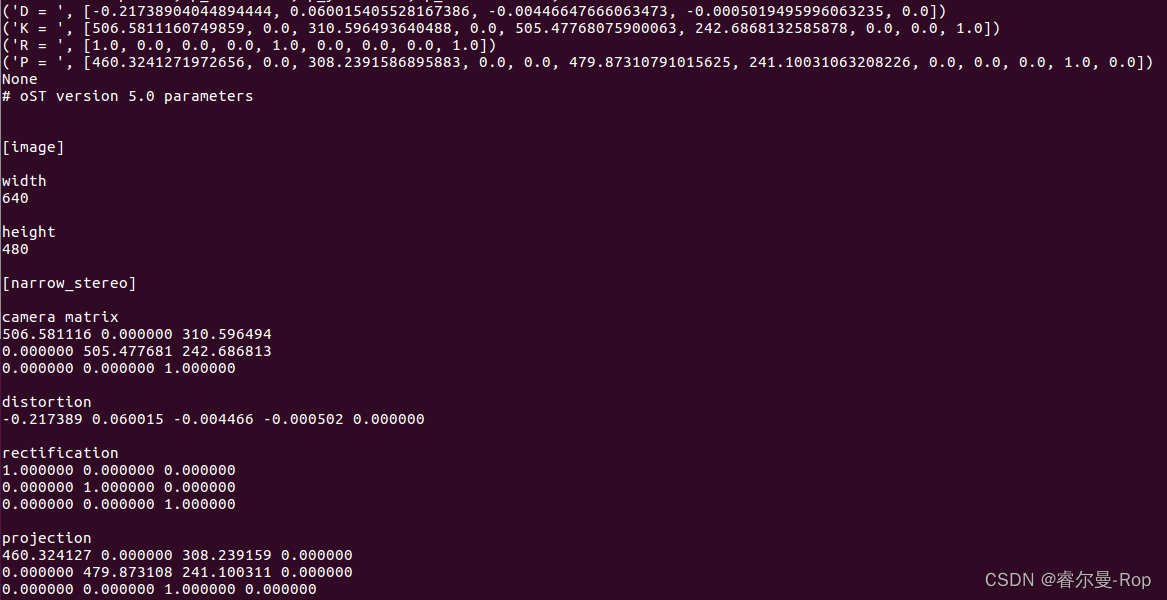
X：标定板左右移动；

Y：标定板上下移动；

Size：标定板前后移动；

Skew：标定板倾斜转动。

最后终端会显示标定结果。



## 3. OpenCV安装

        OpenCV的安装有两种方式，一是源码安装，二是通过apt-get安装，源码安装比较繁琐，且需要用户自己设置的地方较多，适合用户自定义操作。因此在学习过程中，我们采用apt-get安装降低繁琐程度。使用下面命令安装：

sudo apt-get install ros-melodic-vision-OpenCV libOpenCV-dev python-OpenCV

## 4. CvBridge图像

        准备好了摄像头与OpenCV，我们就可以通过cvbridge建立ROS和OpenCV的传输，通过一个例程来演示这一过程。首先，建立一个python文件，实现接收ROS发的图像信息，在图像上画一个圆后，返回给ROS。返回的话题名称是cv\_bridge\_image。

        在工作空间的src下新建robot\_vision功能包：

catkin\_create\_pkg robot\_vision sensor\_msgs cv\_bridge roscpp std\_msgs image\_transport

        在robot\_vision功能包下新建script文件夹，并新建python文件：

mkidr script   
cd script  
touch cv\_bridge\_test.py

        打开py文件，在里面添加如下内容：

#!/usr/bin/env python3 #这里决定于当前系统的python版本  
# -\*- coding: utf-8 -\*-  
import rospy  
import cv2  
from cv\_bridge import CvBridge, CvBridgeError  
from sensor\_msgs.msg import Image  
   
class image\_converter:  
    def \_\_init\_\_(self):      
        # 创建cv\_bridge，声明图像的发布者和订阅者  
        self.image\_pub = rospy.Publisher("cv\_bridge\_image", Image, queue\_size=1)  
        self.bridge = CvBridge()  
        self.image\_sub = rospy.Subscriber("/camera/color/image\_raw", Image, self.callback)  
   
    def callback(self,data):  
        # 使用cv\_bridge将ROS的图像数据转换成OpenCV的图像格式  
        try:  
            cv\_image = self.bridge.imgmsg\_to\_cv2(data, "bgr8")  
        except CvBridgeError as e:  
            print(e)  
   
        # 在OpenCV的显示窗口中绘制一个圆，作为标记  
        (rows,cols,channels) = cv\_image.shape  
        if cols > 60 and rows > 60 :  
            cv2.circle(cv\_image, (60, 60), 30, (0,0,255), -1)  
   
        # 显示OpenCV格式的图像  
        cv2.imshow("Image window", cv\_image)  
        cv2.waitKey(3)  
   
        # 再将OpenCV格式额数据转换成ros image格式的数据发布  
        try:  
            self.image\_pub.publish(self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8"))  
        except CvBridgeError as e:  
            print(e)  
   
if \_\_name\_\_ == '\_\_main\_\_':  
    try:  
        # 初始化ros节点  
        rospy.init\_node("cv\_bridge\_test")  
        rospy.loginfo("Starting cv\_bridge\_test node")  
        image\_converter()  
        rospy.spin()  
    except KeyboardInterrupt:  
        print("Shutting down cv\_bridge\_test node.")  
        cv2.destroyAllWindows()

在script文件夹下打开终端，输入

sudo chmod +x \*.py

然后编译功能包，编译通过后运行

roslaunch robot\_vision usb\_cam.launch

启动cv\_bridge节点

rosrun robot\_vision cv\_bridge\_test.py

启动rqt显示图像

rqt\_image\_view

        得到如下图像，可以看到图像中被添加了红点。左图为通过cv\_bridge将ROS图像转换成OpenCV图像后的显示效果，右图是再次通过cv\_bridge转换成ROS图像后的效果。注意左上角的红点，这是通过OpenCV生成的，左右图像完全一致。

