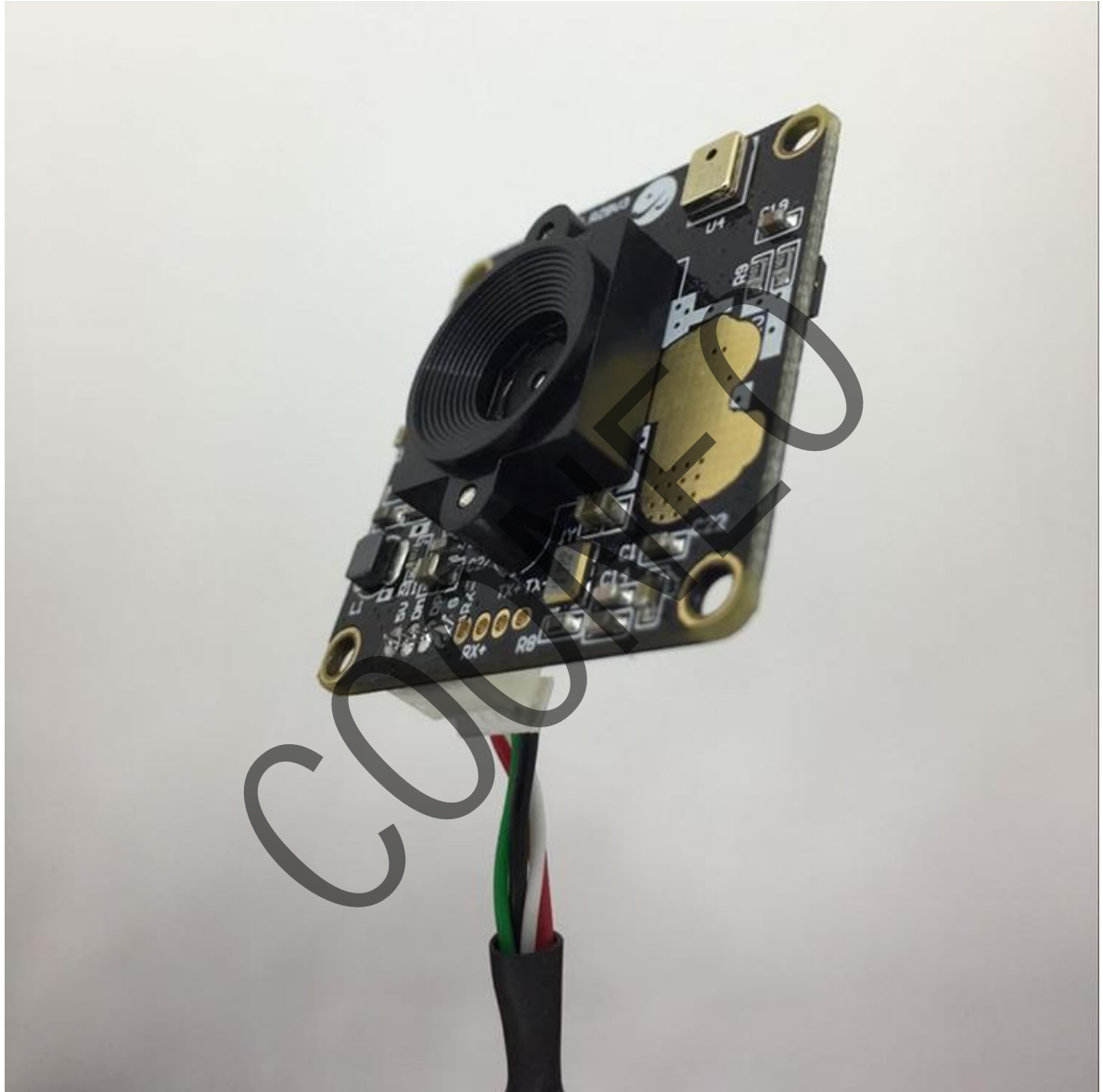


usb_cam的使用与非自动对焦摄像头的校准

一、USB摄像头简介

USB摄像头分自动对焦和手动对焦两种，自动对焦摄像头内部有一个高速对焦马达，其会根据拍摄的视频清晰度来调整摄像头焦距，如下图：



一般情况下，自动对焦摄像头不需要手动校准。但是手动对焦摄像头就需要自行旋转摄像头模组前方的旋钮，调整摄像头采集画面的清晰情况。可以看见下图的镜头下方有很长一段螺纹，这就是手动调节的旋转螺纹。因此在使用的时候需要对其进行校准。



二、usb_cam 功能包的安装与使用

1、安装功能包

```
# cd进入自己的工作空间src目录下，然后下载功能包
git clone https://github.com/ros-drivers/usb_cam.git
# 然后退出到根目录下，catkin_make && source devel/setup.bash 即可运行示例
roslaunch usb_cam usb_cam usb_cam-test.launch
```

2、安装功能包依赖库

```
sudo apt-get install v4l-utils # 否则报错： sudo apt-get install v4l-utils
```

三、手动对焦摄像头的校准

一般情况如果只是拿摄像头作为视频监控功能的话就不用对相机做校准操作，只有在需要调用 **opencv** 功能对相机的视频流做图像处理，例如：物体识别，使用立体相机或双目相机进行测距时才需要对相机进行校准。一般在我们使用 **usb_cam** 来启动相机时，如果没有校准过的话都会收到如下的警告：

```
"[WARN] [1423194481.257752159]: Camera calibration file
/home/xxx/.ros/camera_info/head_camera.yaml not found."
```

对于"unknown control 'focus_auto'"的警告是由于启动的摄像头不具备自动对焦功能，如果启动的摄像头具备自动对焦功能的话，则没有这个警告提示

为了从相机图像信息获得准确的距离信息，则需要多种附加信息，如每台相机各不相同的镜头特性、镜头与图像传感器之间的距离以及扭转角度等信息。这是因为相机是一个将我们生活的三维空间世界投影到二维空间的图像投影设备，所以在投影过程中，由于每台相机的固有特性，这些参数都会不同。并且在相机制作过程中，镜头和图像传感器必须水平组装，但由于细微的偏差，图像中心（image center）与主点（principal point）会有细微的偏离，且图像传感器的倾度也会略有差异。

校准这些部件的过程称为**相机校准（Calibration）**，目的是查找相机的固有参数。摄像机的校准是非常重要的，ROS提供的是利用OpenCV相机校准的校准功能包，接下来我们按照如下步骤操作即可：

1、安装相机校准软件包

```
sudo apt install ros-kinetic-camera-calibration
```

但是在使用该校准软件包时需要启动uvc_camera软件包，这样我们才能使用camera-clibration软件进行校准，最终生成的校准yaml文件是可以在uvc_camera和usb_cam软件包中通用的。安装uvc_camera的命令如下：

```
sudo apt install ros-kinetic-uvc-camera
```

2、查看默认的相机信息

首先启动uvc_camera节点，通过如下命令启动，**注意需要先启动roscore节点后再启动该节点：**

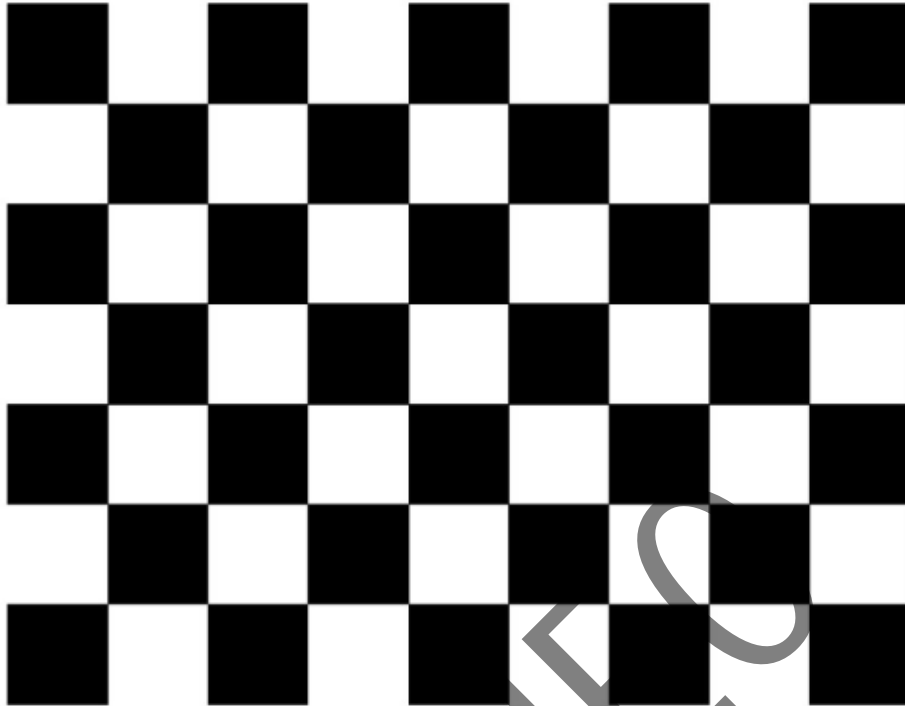
```
roslaunch uvc_camera uvc_camera_node
```

终端反馈的如下图：

```
cooneo@cooneo: ~/mini_ws
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
cooneo@cooneo: ~/mini_ws x roscore http://cooneo:11311/ x
[ WARN] [1595234800.933702476]: Camera calibration file /home/cooneo/.ros/camera_info^
/camera.yaml not found.
opening /dev/video0
pixfmt 0 = 'YUYV' desc = 'YUYV 4:2:2'
discrete: 640x480: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 160x90: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 160x120: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 176x144: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 320x180: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 320x240: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 352x288: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 432x240: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 640x360: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 800x448: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 800x600: 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 864x480: 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 960x720: 1/15 1/10 2/15 1/5
discrete: 1024x576: 1/15 1/10 2/15 1/5
discrete: 1280x720: 1/10 2/15 1/5
discrete: 1600x896: 2/15 1/5
discrete: 1920x1080: 1/5
discrete: 2304x1296: 1/2
discrete: 2304x1536: 1/2
pixfmt 1 = 'MJPG' desc = 'Motion-JPEG'
discrete: 640x480: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 160x90: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 160x120: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 176x144: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 320x180: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 320x240: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 352x288: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 432x240: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 640x360: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 800x448: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 800x600: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 864x480: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 960x720: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 1024x576: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 1280x720: 1/60 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 1600x896: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
discrete: 1920x1080: 1/30 1/24 1/20 1/15 1/10 2/15 1/5
int (Brightness, 0, id = 980900): 0 to 255 (1)
int (Contrast, 0, id = 980901): 0 to 255 (1)
int (Saturation, 0, id = 980902): 0 to 255 (1)
bool (White Balance Temperature, Auto, 0, id = 98090c): 0 to 1 (1)
int (Gain, 0, id = 980913): 0 to 255 (1)
menu (Power Line Frequency, 0, id = 980918): 0 to 2 (1)
0: Disabled
1: 50 Hz
2: 60 Hz
int (White Balance Temperature, 16, id = 98091a): 2000 to 6500 (1)
int (Sharpness, 0, id = 98091b): 0 to 255 (1)
int (Backlight Compensation, 0, id = 98091c): 0 to 1 (1)
menu (Exposure, Auto, 0, id = 9a0901): 0 to 3 (1)
int (Exposure (Absolute), 16, id = 9a0902): 3 to 2047 (1)
bool (Exposure, Auto Priority, 0, id = 9a0903): 0 to 1 (1)
int (Pan (Absolute), 0, id = 9a0908): -36000 to 36000 (3600)
int (Tilt (Absolute), 0, id = 9a0909): -36000 to 36000 (3600)
int (Focus (absolute), 0, id = 9a090a): 0 to 250 (5)
bool (Focus, Auto, 0, id = 9a090c): 0 to 1 (1)
```

3、准备校准用的棋盘

摄像机的校准是以一个由黑白方块组成的棋盘为基准进行的，如下图所示。8×6国际象棋棋盘，并打印出来后将其贴到一个平坦的纸板上。这里用的是A4纸作为参考，8×6棋盘横向有9个方块，所以有8个交叉点，而竖向有7个方块，有6个交叉点，所以它被称为8×6棋盘。



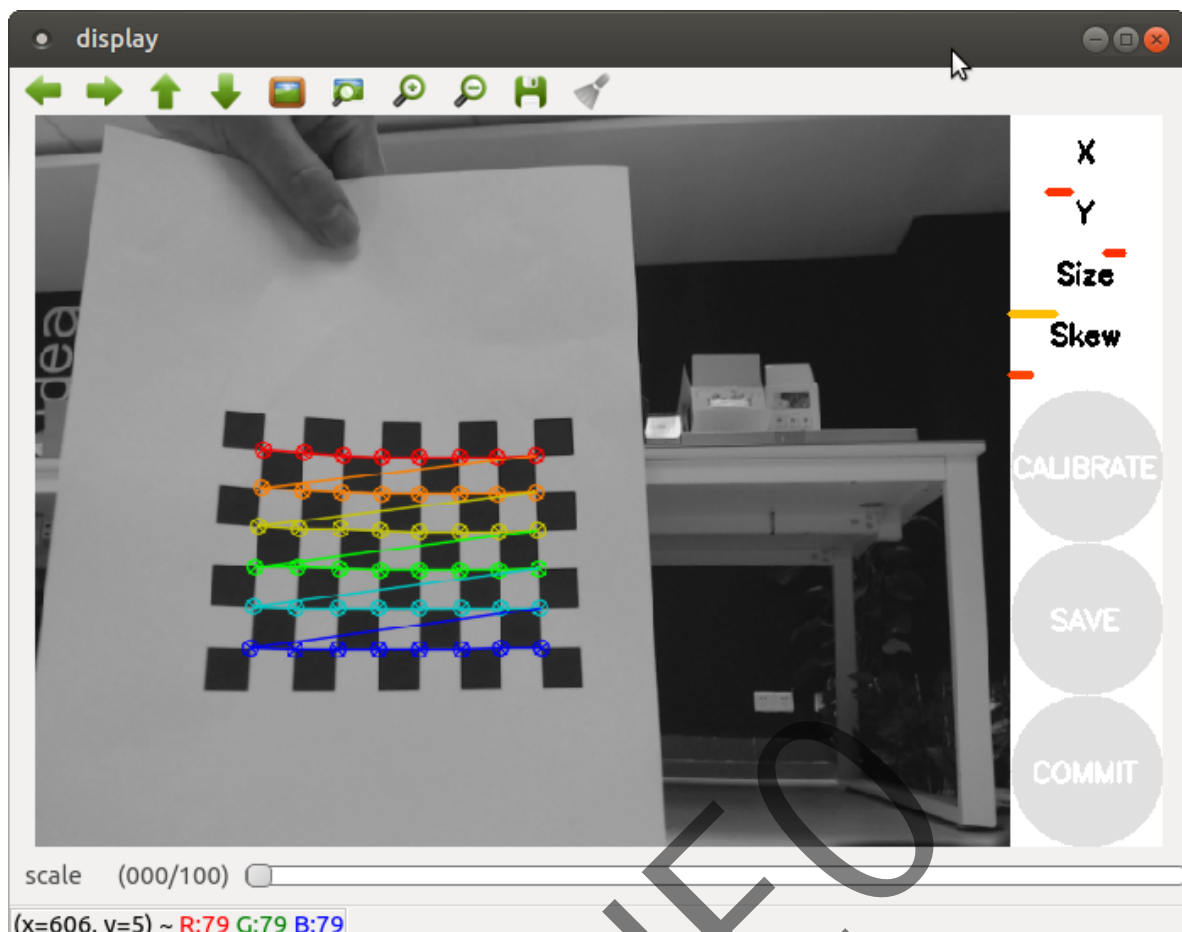
使用如下命令来进行启动校准节点：

```
# 在两个终端中分别运行 如下两条命令，记得事先运行roscore
roslaunch uvc_camera uvc_camera_node
roslaunch camera_calibration cameracalibrator.py --size 8x6 --square 0.015
image:=/image_raw
```

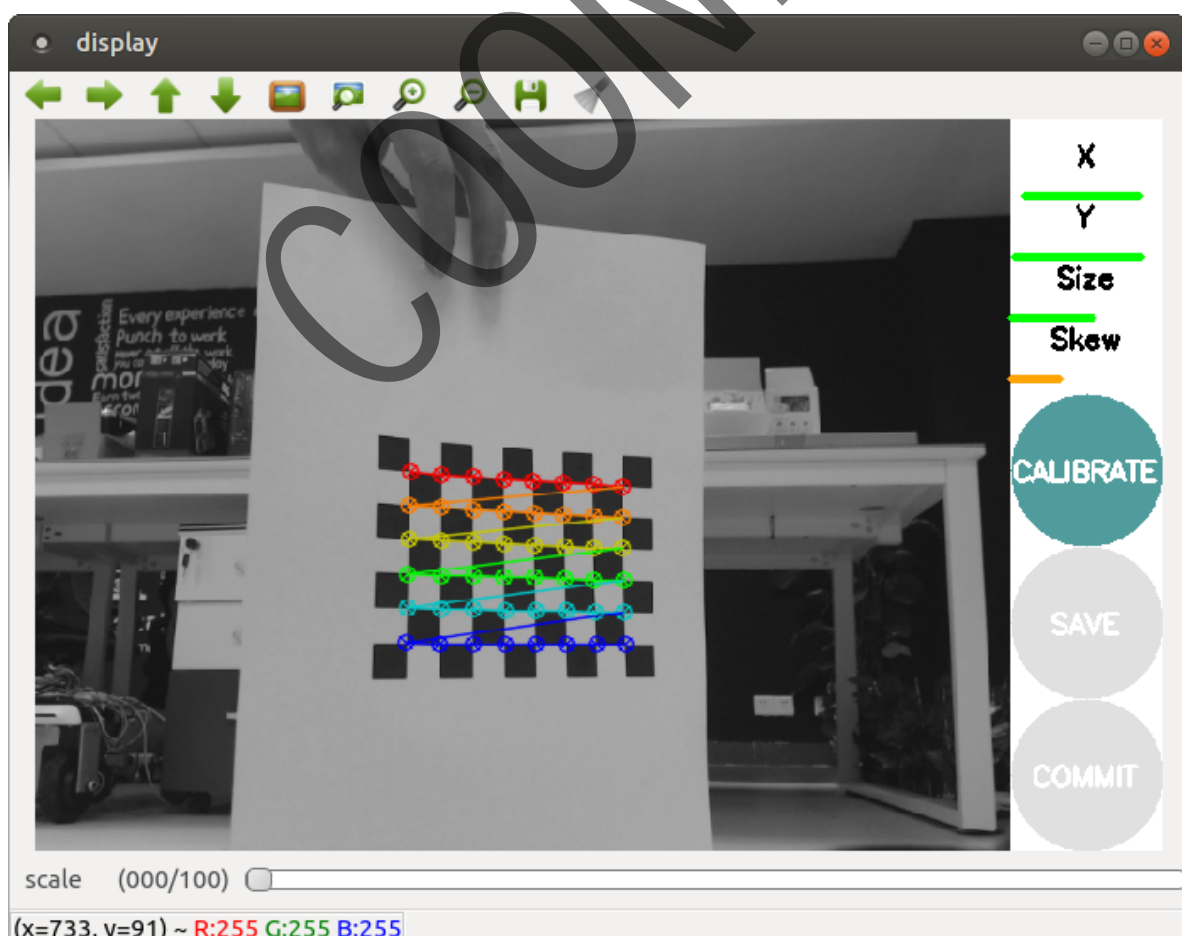
参数介绍：-size是上述棋盘的宽度和高度，-square 0.0255是棋盘的一个小方格的实际尺寸。这个小方块是正方形，但是打印出来的尺寸可能会各不相同。（我们可以使用尺子测量一下每个格子的宽度，我打印出来是15毫米），所以下面命令中写了0.015.一般由于打印的不同大小我们只需要改动square大小，其他参数基本上不用改动。

校准的过程中，需要将棋盘对着相机朝着左/右/上/下/前/后移动，还需要倾斜棋盘，我们需要拿着棋盘不断移动，直到校准所需的所有图像都记录下来之后，CALIBRATE按钮会被激活。

校准开始的GUI画面：



校准完成后的GUI画面：



点击这个按钮后会进行实际的校准计算，这需要大约3到5分钟。此时就可以将校准板放下了，耐心等待计算完成后，点击SAVE按钮保存校准信息，存储的地址显示在执行校准的终端窗口中，是存储在某一个/tmp目录（如“/tmp/calibrationdata.tar.gz”）中。



保存完毕相机校准文件后就可以退出正在运行的两个程序了，之后需要到保存的目录下，解压保存的文件“/tmp/calibrationdata.tar.gz”，将其中的.yaml文件中的内容拷贝到“/home/xxx/.ros/camera_info/head_camera.yaml”中，需要检查拷贝过来的“camera_name”更正为：“head_camera”。

注意：第一次校准摄像头的时候，那么你的ubuntu的“/home/xxx/.ros”目录下应该没有camera_info文件夹，此时需要自己建立该文件夹，及在文件夹中新建head_camera.yaml配置文件。这样在运行usb_cam的launch文件时才能在这个路径下找到配置文件。若更换摄像头，那么就需要按照之上的步骤更换新的配置文件。

此时，再次启动usb_camera功能包时，便不会提示没有校准配置文件了。

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

若提示如下，即成功。



并且此时也可以在rviz中看到图像话题发布。

四、使用rosvbag记录ROS功能包运行时的日志

当我们在对视频中图像信息流进行处理时经常需要不断的来打开摄像头进行测试，但是这样太过麻烦，而且需要一直插着摄像头，此时我们就可以将相机拍摄的视频保存下来，这样我们可以通过不断的向话题中重发消息就可以模拟有相机的情况了，而且我们还可以将该bag包分发给别人进行测试。

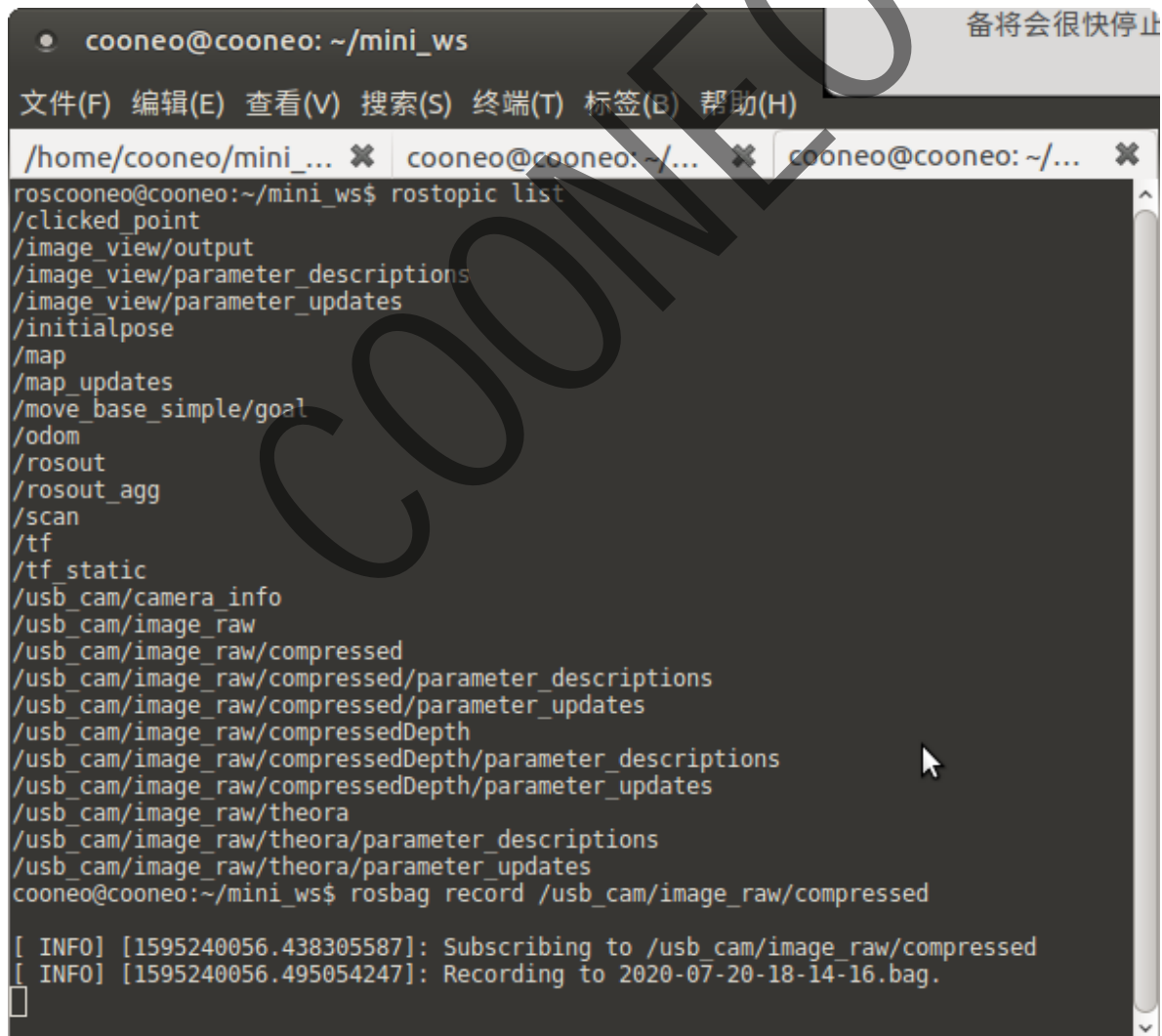
rqt_bag是一个可以将消息进行可视化的GUI工具，ROS日志信息中的rosvbag是基于文本的，对于图像数据类的消息管理就需要用rqt_bag了，它可以进行存储、回放和压缩话题消息，而且所有的命令都是基于按钮制作的，所以很容易使用，就跟常用的视频编辑器一样，可以在时间轴上拖拽查看相机图像，下面来详细介绍如何操作：

- (1) 首先使用roslaunch启动usb_cam节点，使用如下命令启动即可：

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

- (2) 使用rosvbag来记录指定话题的数据，我这里保存的是压缩图像的话题，这样保存的日志文件很小，方便我们进行调试，使用如下命令开始记录话题输出：

```
rosvbag record /usb_cam/image_raw/compressed
```



The image shows a terminal window with the following content:

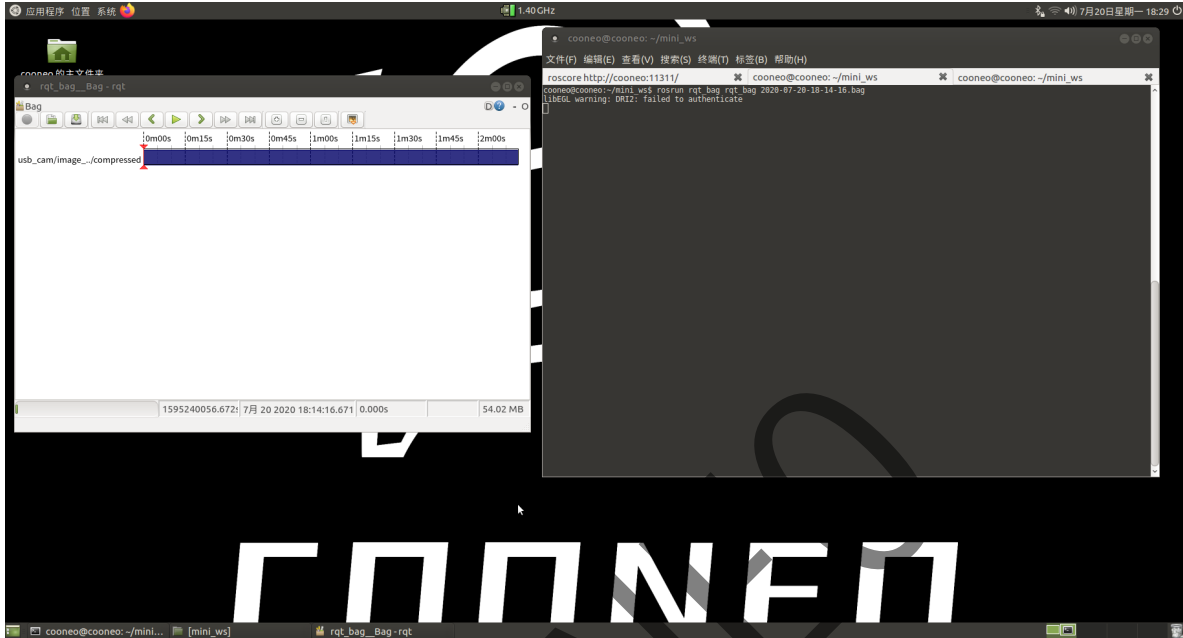
```
cooneo@cooneo: ~/mini_ws
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 标签(B) 帮助(H)
/home/cooneo/mini_... x cooneo@cooneo: ~/... x cooneo@cooneo: ~/... x
rosvcooneo@cooneo:~/mini_ws$ rostopic list
/clicked_point
/image_view/output
/image_view/parameter_descriptions
/image_view/parameter_updates
/initialpose
/map
/map_updates
/move_base_simple/goal
/odom
/rosout
/rosout_agg
/scan
/tf
/tf_static
/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
cooneo@cooneo:~/mini_ws$ rosvbag record /usb_cam/image_raw/compressed

[ INFO] [1595240056.438305587]: Subscribing to /usb_cam/image_raw/compressed
[ INFO] [1595240056.495054247]: Recording to 2020-07-20-18-14-16.bag.
```

从开始记录日志开始相机现在画面内容都会被录制下来，此时我们就可以在相机前进行各种图像录制，当录制完成后只需要ctrl+c来中断record即可。如果没有限制录制的“Topic”以及规定保存的路径，那么此时将录制此rosvmaster所管理下的所有数据，并存放在终端的当前目录下。

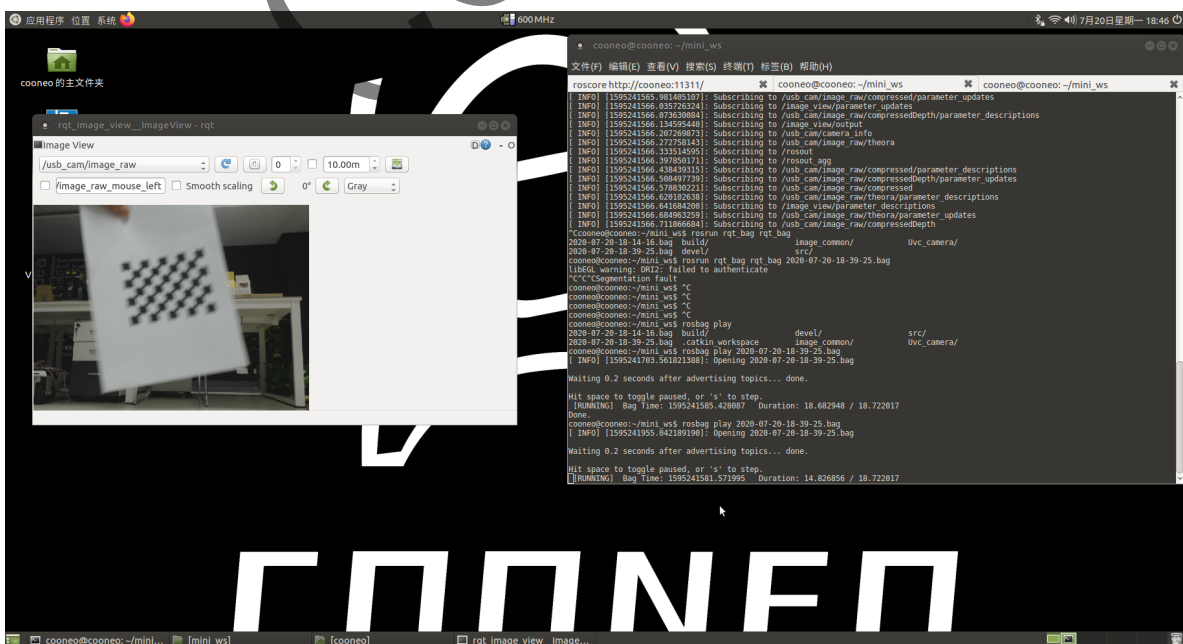
(3) 接下来准备开始数据回放，首先需要将usb_cam的launch给中断了，不然等下的数据回放话题会跟这个原始的有冲突。要想在rqt中加载bag日志信息，只要在中断中输入rqt选中[插件Plugins]->[日志Logging]->[数据包bag]，然后选择左上方的文件夹图标（Load Bag）加载刚才录制的bag文件即可。

```
# 检查是否运行了roscore
# 然后另外开一个终端，进入到刚才自己保存的.bag话题的目录下，并运行：
roslaunch rqt_bag rqt_bag XXXXXXXXXX.bag
```



上面的一步是查看录制的数据播放的过程，下面将展示如何将录制的数据发布出来：

```
# 同样 检查 roscore 是否已启动
roslaunch rqt_image_view rqt_image_view
# 然后可以用rostopic list查看,或者是用:roslaunch rqt_image_view rqt_image_view 查看
rostopic list
roslaunch rqt_image_view rqt_image_view #选择左上角 为对应的话题名称
```



COONEO