QR code identification

1. Introduction

This course mainly uses the camera of raspberry pie to obtain the image of the camera, calls the pyzbar library to analyze the image, and analyzes the position and band information of the two-dimensional code in the image. The QR code used this time is QRcode. You can find a QR code generator on the Internet to generate a QR code as a test.

The two-dimensional code is a black-and-white pattern that records data symbol information in a plane (two-dimensional direction) with a certain geometric pattern. It cleverly uses the concept of "0" and "1" bit streams that constitute the internal logic basis of the computer, and uses several geometric shapes corresponding to binary to represent the character value information. Each code system has its own specific character set, and each character occupies a certain width, It has certain verification function.

2. Code analysis

Create a new 2D code detection task, which is mainly to convert the camera image into a gray image, and then send it to the decodedisplay function for analysis and processing.

```
# 检测二维码  detect qrcode
def Detect_Qrcode_Task():
    ret, frame = image.read()
    image_widget.value = bgr8_to_jpeg(frame)
    t_start = time.time()
    fps = 0
    while True:
        ret, frame = image.read()
        # 转为灰度图像  Convert to grayscale image
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        frame = decodeDisplay(gray, frame)
        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        image_widget.value = bgr8_to_jpeg(frame)
```

The decodedisplay function compares the incoming images. The number of images is transferred to the gray-scale image, and display is the original image. The significance of this is that the gray-scale image recognition result can be written to the original image for display.

```
# 解析图像中的二维码信息  Analyze the qrcode information in the image
def decodeDisplay(image, display):
    barcodes = pyzbar.decode(image)
    for barcode in barcodes:
        # 提取二维码的边界框的位置，画出图像中条形码的边界框
        # Extract the position of the bounding box of the qrcode,
        # and draw the bounding box of the barcode in the image
        (x, y, w, h) = barcode.rect
        cv2.rectangle(display, (x, y), (x + w, y + h), (225, 225, 225), 2)

        # 提取二维码数据为字节对象，转换成字符串
        # The qrcode data is extracted as byte objects and converted into strings
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type

        # 绘出图像上条形码的数据和条形码类型
        # Plot the barcode data and barcode type on the image
        text = "{} ({})".format(barcodeData, barcodeType)
        cv2.putText(display, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (225, 0, 0), 2)

        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
    return display
```

Start a daemon thread, run the camera identification task, and display the camera image.

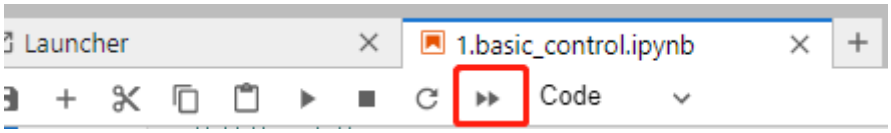
```
# 启动摄像头显示任务  Start the camera display task
thread1 = threading.Thread(target=Detect_Qrcode_Task)
thread1.setDaemon(True)
thread1.start()

box_display = widgets.HBox([image_widget, button_Close_Camera])
display(box_display)
```

1.3 Steps

Open the jupyterLab client and find following code path

```
DOGZILLA/Samples/3_AI_Visual/7.QRCode.ipynb
```

Click the following icon to run all cells, and then pull to the bottom to see the generated controls.



The left side shows the camera image. If the two-dimensional code is placed in front of the camera, the robot dog will recognize the two-dimensional code and display the recognized data on the two-dimensional code.

Finally, click close_camera button turns off the camera.