

• 2. QR code creation and recognition

2.1. QR QR code

2.1.1. QR QR code introduction

QR code is a type of two-dimensional barcode. QR comes from the abbreviation of "Quick Response" in English, which means quick response. It comes from the inventor's hope that QR code can allow its content to be decoded quickly. QR code not only has large information capacity, high reliability and low cost, but also can represent a variety of text information such as Chinese characters and images. It has strong confidentiality and anti-counterfeiting properties and is very convenient to use. More importantly, the QR code technology is open source.

2.1.2. QR QR code structure

图片	解析
	定位标识 (Positioning markings) 标明二维码的方向。
	对齐标记 (Alignment markings) 如果二维码很大，这些附加元素帮助定位。
	计算模式 (Timing pattern) 通过这些线，扫描器可以识别矩阵有多大。
	版本信息 (Version information) 这里指定正在使用的QR码的版本号，目前有QR码有40个不同的版本号。用于销售行业的版本号通常为1-7。
	格式信息 (Format information) 格式模式包含关于容错和数据掩码模式的信息，并使得扫描代码更加容易。
	数据和错误校正值 (Data and error correction keys) 这些模式保存实际数据。
	宁静区域 (Quiet zone) 这个区域对于扫描器来说非常重要，它的作用就是将自身与周边的进行分离。

2.1.3. QR QR code characteristics

The data value in the QR code contains repeated information (redundant value). Therefore, even if up to 30% of the QR code structure is destroyed, it does not affect the readability of the QR code. The storage space of the QR code is up to 7089 bits or 4296 characters, including punctuation and special characters, which can be written into the QR code. In

addition to numbers and characters, words and phrases (such as URLs) can also be encoded. As more data is added to the QR code, the code size increases and the code structure becomes more complex.

2.1.4, QR code creation and recognition

1), create QR code

- Source code path

```
~/orbbec_ws/src/yahboomcar_visual/simple_qrcode/QRcode_Create.py
```

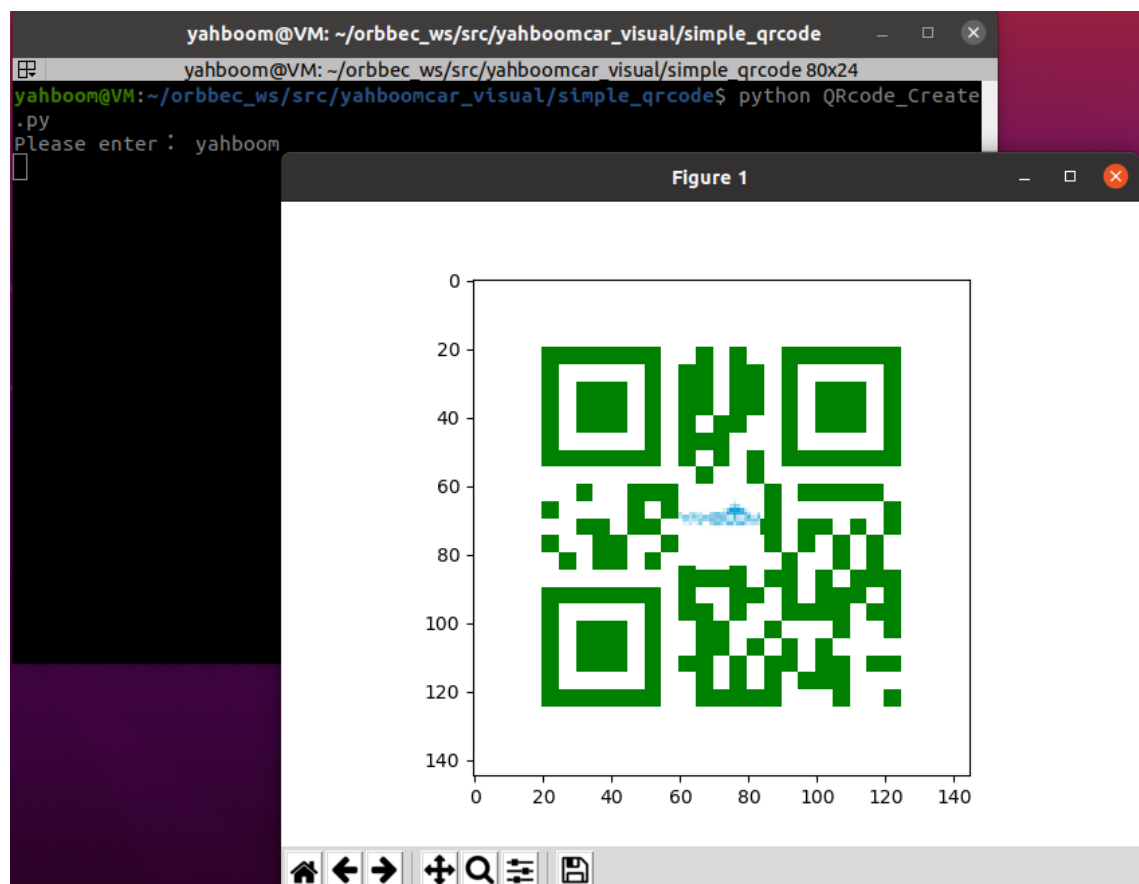
- Installation package

```
python3 -m pip install qrcode pyzbar  
sudo apt-get install libzbar-dev
```

- Start

```
cd ~/orbbec_ws/src/yahboomcar_visual/simple_qrcode  
python QRcode_Create.py
```

After the program runs, it will prompt you to enter the generated content, and press Enter to confirm the content. Here we take the creation of the "yahboom" string as an example,



- Core source code analysis

```
#Create qrcode object  
qr = qrcode.QRCode(  
    version=1,
```

```

error_correction=qrcode.constants.ERROR_CORRECT_H,
box_size=5,
border=4,)
#Meaning of each parameter
'''version: An integer with a value of 1 to 40, controlling the size of the
QR code (the minimum value is 1, which is a 12x12 matrix).
If you want the program to determine it automatically, set the value to None
and use the fit parameter.
error_correction: Controls the error correction function of the QR code. The
following 4 constants can be used.
ERROR_CORRECT_L: About 7% or less of the errors can be corrected.
ERROR_CORRECT_M (default): About 15% or less errors can be corrected.
ERROR_CORRECT_H: About 30% or less errors can be corrected.
box_size: Controls the number of pixels contained in each small grid in the
QR code.
border: controls the number of grids contained in the border (the distance
between the QR code and the image border) (the default is 4, which is the
minimum value specified by the relevant standards)'''
#qrcode QR code adds logo
my_file = Path(logo_path)
if my_file.is_file(): img = add_logo(img, logo_path)
#Add data
qr.add_data(data)
#Fill data
#fill data
qr.make(fit=True)
#Generate images
#generate images
img = qr.make_image(fill_color="green", back_color="white")

```

2), identify QR code

- Source code path

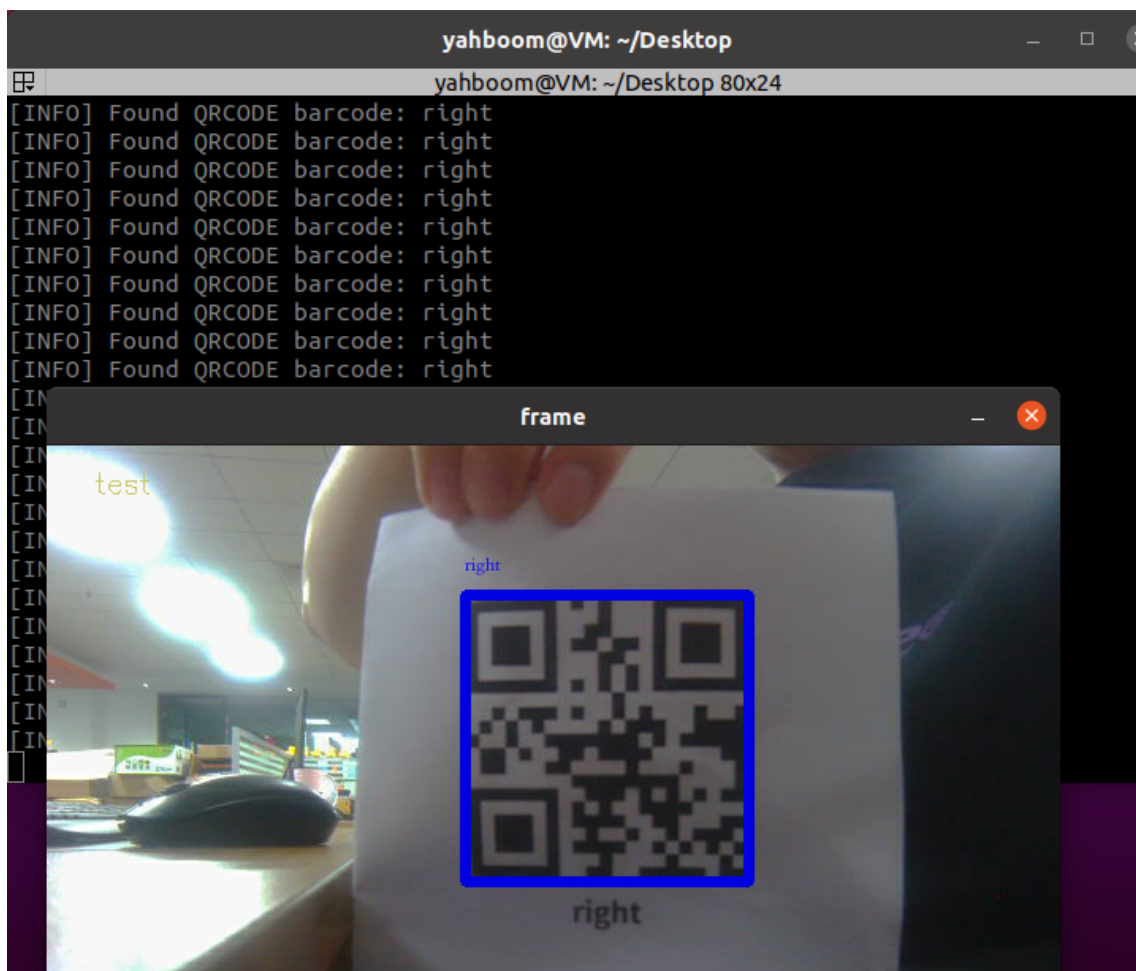
```
~/orbbec_ws/src/yahboomcar_visual/yahboomcar_visual/QRcode_Parsing.py
```

- Start

```

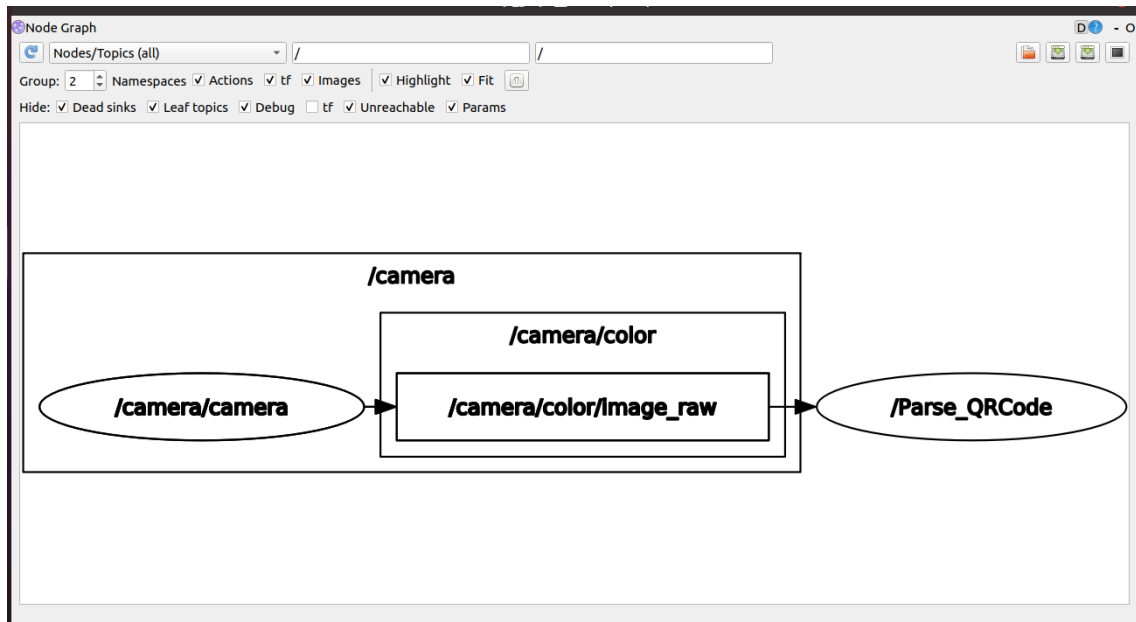
#gemini camera start
ros2 launch astra_camera gemini.launch.xml
ros2 run yahboomcar_visual QRcode_Parsing

```



- View node communication diagram

```
ros2 run rqt_graph rqt_graph
```



- Core source code analysis

```
#Subscribe to color image information
self.sub_img =
self.create_subscription(CamImage, '/camera/color/image_raw', self.handleTopic,
100)
#Pass the image data to the image parsing function in the callback function
```

```

frame = self.decodeDisplay(frame)
# parse the image
def decodeDisplay(self, image):
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    # The output Chinese characters need to be converted to Unicode encoding
    # first
    barcodes = pyzbar.decode(gray)
    for barcode in barcodes:
        # Extract the position of the boundary box of the TWO-DIMENSIONAL code
        # Draw the bounding box for the bar code in the image
        (x, y, w, h) = barcode.rect
        cv.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 5)
        encoding = 'UTF-8'
        # to draw it, you need to convert it to a string first
        # to convert it to a string
        barcodeData = barcode.data.decode(encoding)
        barcodeType = barcode.type
        # Draw the data and type on the image
        pilimg = Image.fromarray(image)
        # create brush
        draw = ImageDraw.Draw(pilimg) # Print on picture
        # parameter 1: font file path, parameter 2: font size
        fontStyle =
        ImageFont.truetype("/home/yahboom/orbbec_ws/src/yahboomcar_visual/yahboomcar_
        visual/Block_Simplified.TTF", size=12, encoding=encoding)
        # # Parameter 1: print coordinates, parameter 2: text, parameter 3: font
        # color, parameter 4: font
        # Parameter 1: print coordinates, parameter 2: text, parameter 3: font color,
        # parameter 4: font
        draw.text((x, y - 25), str(barcode.data, encoding), fill=(255, 0, 0),
        font=fontStyle)
        # # PIL picture to cv2 picture
        # PIL picture to CV2 picture
        image = np.array(pilimg)
        # Print barcode data and barcode type to terminal
        # Print barcode data and barcode type to terminal
        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
        return image

```