








## 3、QR code creation and recognition

### 1、QR code

#### 1.1、QR Code Introduction

QR code is a type of two-dimensional barcode. It not only has large information capacity, high reliability, and low cost, but it can also express a variety of text information such as Chinese characters and images. It has strong confidentiality and anti-counterfeiting and is very convenient to use.

#### 1.2、Structure of QR code

Pciture	Parsing
	Positioning markings: Indicate the direction of the QR code.
	Alignment markings: If the QR code is large, these additional elements help positioning.
	Timing pattern: Through these lines, the scanner can identify the size of the matrix
	Version information) The version number of the QR code being used. There are currently 40 different version numbers of the QR code. Version numbers used in the sales industry are usually 1-7.
	Format information: The format mode contains information about fault tolerance and data mask mode, and makes it easier to scan the code.
	Data and error correction keys : These modes save actual data.
	Quiet zone: This area is very important to the scanner, and its role is to separate itself from the surroundings.

#### 1.3、Features of QR codes

The data value in the QR code contains repeated information (redundant value)..Therefore, even up to 30% of the structure of the QR code is destroyed without affecting the readability of the QR code. The storage space of the QR code is up to 7089 bits or 4296 characters, including punctuation marks 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 a QR code, the code size increases and the code structure becomes more complex.

## 1.4、QR code creation and recognition

### 1) 、 Create a QR code

- Code path

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

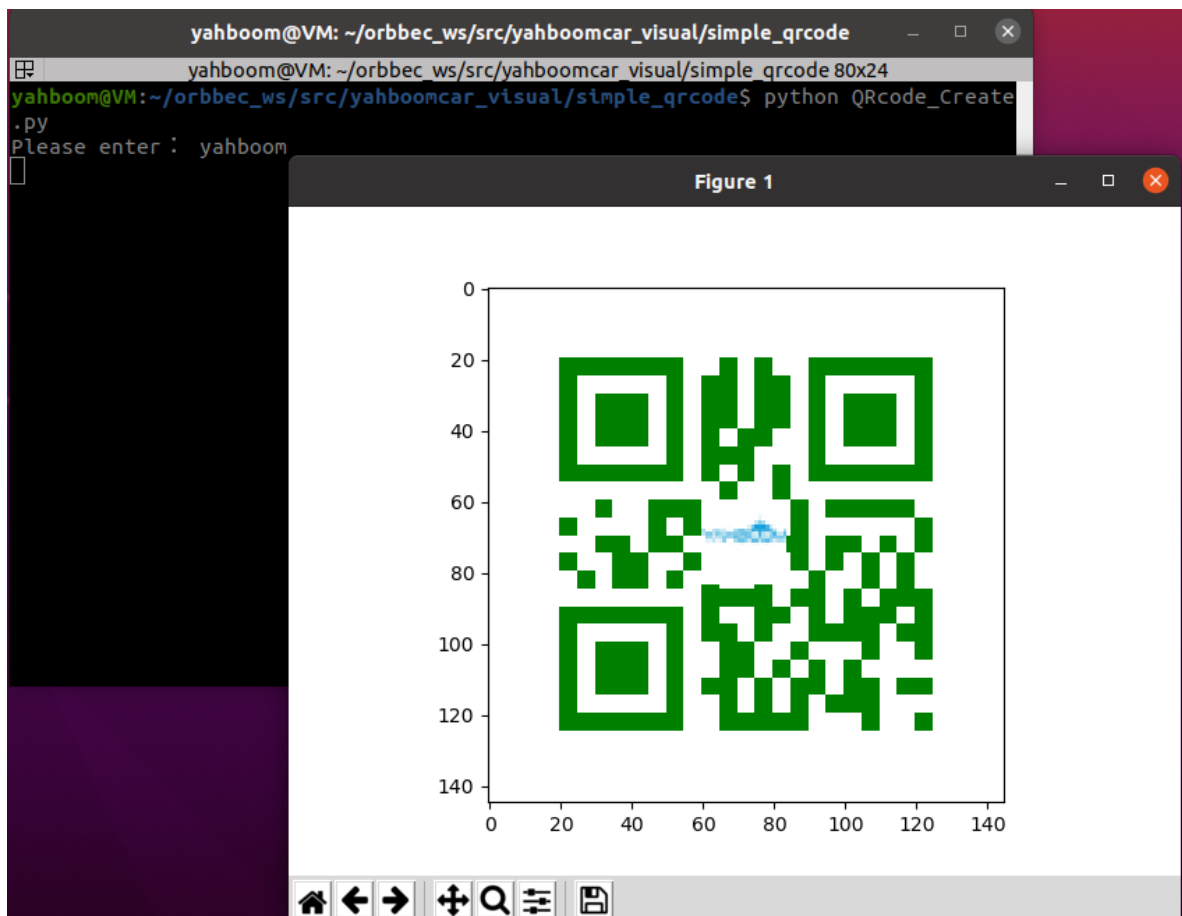
- Install

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

- start up

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

After the program runs, it will prompt you to input the generated content, press Enter to confirm the content. Here we take creating "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,)
#The meaning of each parameter
'''version: The value is an integer from 1 to 40, which controls the size of the
QR code (the minimum value is 1, which is a 12x12 matrix).
If you want the program to determine this automatically, set the value to None
and use the fit parameter.
error_correction: Controls the error correction function for QR codes. It can
take the following 4 constants.
ERROR_CORRECT_L: About 7% or less of errors can be corrected.
ERROR_CORRECT_M (default): About 15% or less of errors can be corrected.
ERROR_CORRECT_H: About 30% or less of errors can be corrected.
box_size: Control the number of pixels contained in each small grid in the QR
code.
border: Control 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 stipulated by relevant standards)'''
#qrcode two-dimensional code to add logo
my_file = Path(logo_path)
if my_file.is_file(): img = add_logo(img, logo_path)
#adding data
qr.add_data(data)
#Data input
# fill data
qr.make(fit=True)
# generate image
# generate images
img = qr.make_image(fill_color="green", back_color="white")

```

## 2) 、 Identify QR code

- source path

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

- start up

```

ros2 launch orbbec_camera gemini2.launch.xml
ros2 run yahboomcar_visual QRcode_Parsing

```



```

#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 parsing image function in the callback function
frame = self.decodeDisplay(frame)
#parsing images
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
        barcodeData = barcode.data.decode(encoding)
        barcodeType = barcode.type

        # Draw the data and type on the image
        piling = Image.fromarray(image)

        # create brush
        draw = ImageDraw.Draw(piling) # 图片上打印 Print on picture

        # parameter 1: font file path, parameter 2: font size
        fontStyle =
        ImageFont.truetype("/home/yahboom/orbbec_ws/src/yahboomcar_visual/yahboomcar_vis
        ual/Block_Simplified.TTF", size=12, encoding=encoding)

        # 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
        image = np.array(piling)

        # Print barcode data and barcode type to terminal
        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
        return image

```