








# QR code creation and recognition

## 1、QR code

### 1.1、Introduction of QR code

QR code is a kind 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 make its content quickly decoded. QR code not only has large information capacity, high reliability and low cost, but also can represent various text information such as Chinese characters and images. It has strong confidentiality and anti-counterfeiting and is very convenient to use. What's more, the QR code technology is open source.

### 1.2、The structure of QR code

picture	Parse
	<b>Positioning</b> markings indicate the direction of the QR code.
	<b>Alignment</b> markings If the QR code is large, these additional elements help with positioning.
	<b>pattern</b> With these lines, the scanner can identify how big the matrix is.
	<b>Version information</b> (Version information) here specifies the version number of the QR code in use. There are currently 40 different version numbers of the QR code. Version numbers for the sales industry are usually 1-7.
	<b>Format</b> information Format patterns contain information about fault tolerance and data mask patterns and make scanning codes easier.
	<b>Data</b> and error correction keys These modes hold the actual data.
	<b>Quiet</b> zone This zone is very important for the scanner, its role is to separate itself from the surrounding.

### 1.3、Features of QR code

The data values in the QR code contain duplicate information (redundant values). Therefore, even up to 30% of the QR code structure 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, 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.

## 1.4、 QR code creation and recognition

### 1) 、 Source path:

```
/root/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode
```

### 2) 、 Install

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

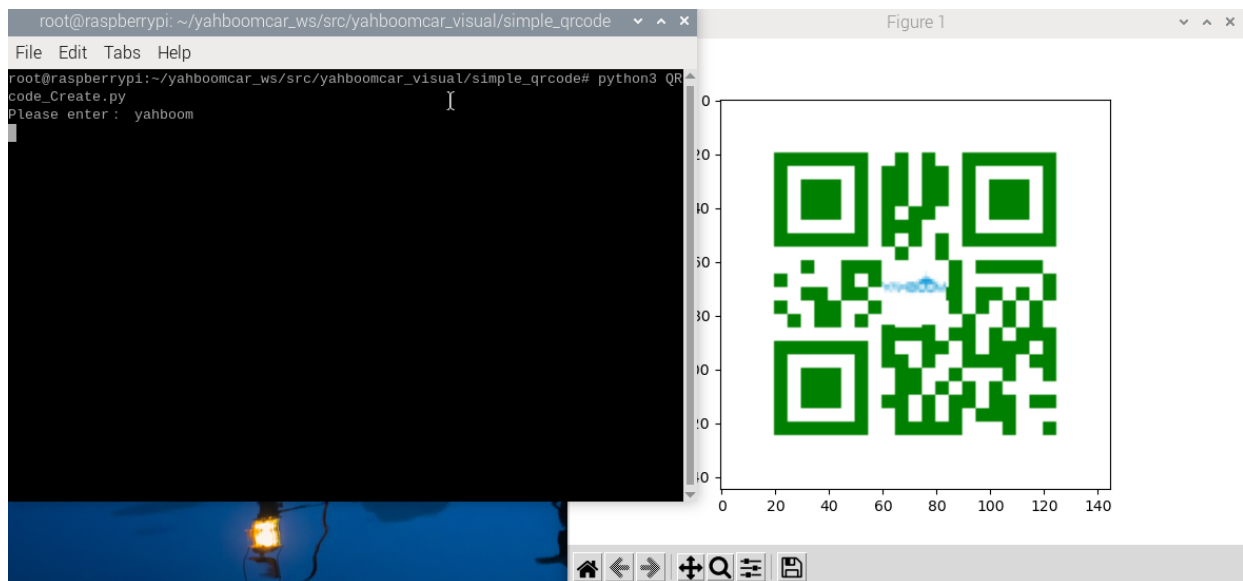
The factory docker image has been installed.

### 3) 、 Create Qrcode\_Create.py

Enter docker, open the terminal and enter ,

```
cd /root/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode/  
python3 Qrcode_Create.py
```

After the program is run, you will be prompted to enter the generated content, and press Enter to confirm the content. Here we take creating the "yahboom" string as the content as an example.



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: An integer with a value from 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 this automatically, set the value to None and use the fit argument.

```
error_correction: Controls the error correction function of QR codes. Possible
values are the following 4 constants.
```

```
        ERROR_CORRECT_L: About 7% or less of errors can be corrected
```

```
        ERROR_CORRECT_M(Default): Approximately 15% or less of errors can be
corrected.
```

```
        ROR_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 cells included in the border (the distance between the
QR code and the image border) (the default is 4, which is the minimum value
specified by relevant standards)'''
```

```
#Add logo to qrcode QR code
```

```
my_file = Path(logo_path)
```

```
if my_file.is_file(): img = add_logo(img, logo_path)
```

```
#adding data
```

```
qr.add_data(data)
```

```
# fill data
```

```
qr.make(fit=True)
```

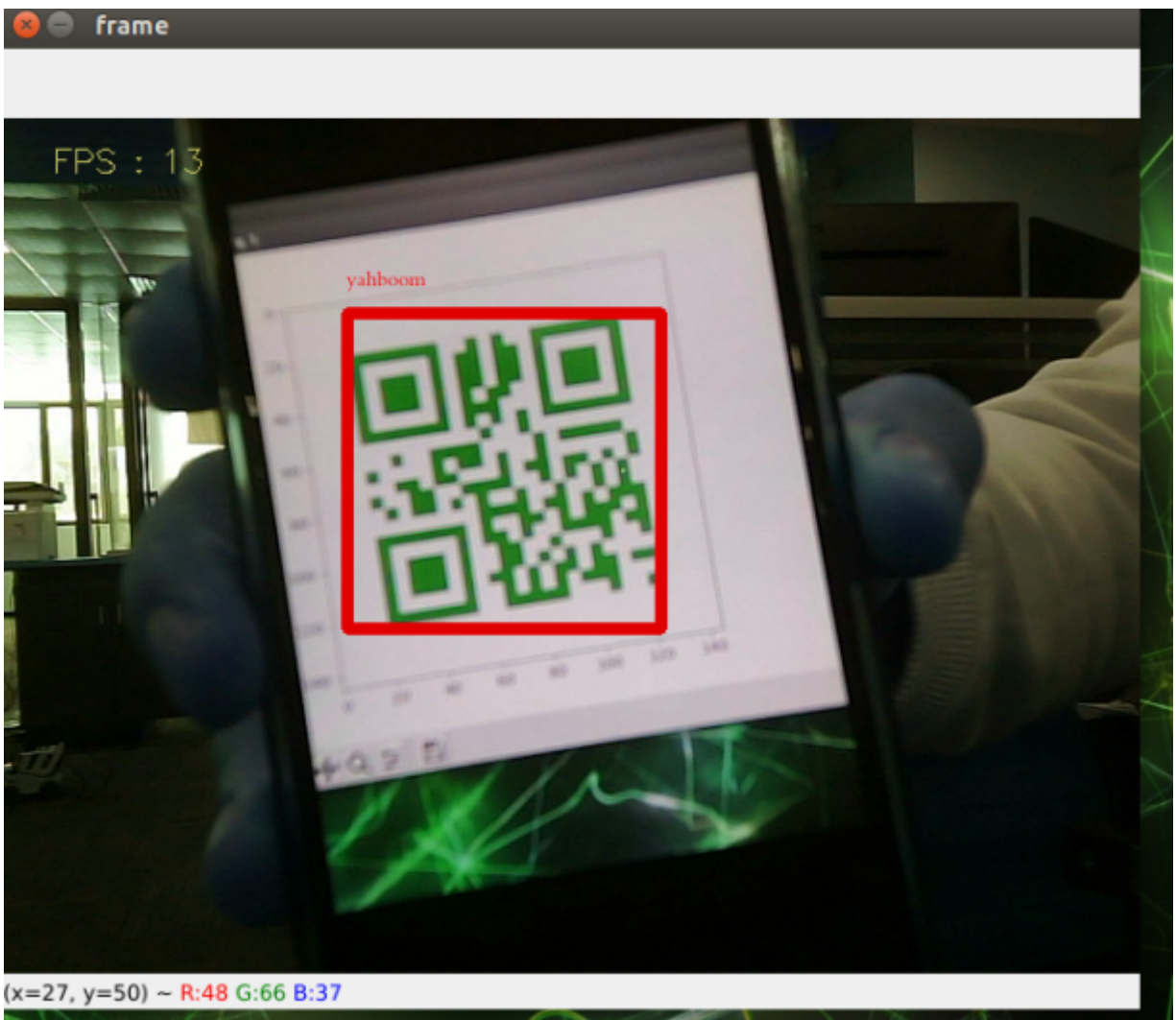
```
# generate images
```

```
img = qr.make_image(fill_color="green", back_color="white")
```

#### 4) 、 Identify QRcode\_Parsing.py

Enter docker, open the terminal and enter

```
cd /root/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode/
python3 QRcode_Parsing.py
```



After the program is run, we place the QR code in front of the camera. The program will recognize the content of the QR code, mark it on the picture and print out the recognized content on the terminal.

Source code analysis

```
def decodeDisplay(image, font_path):
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    # The output Chinese characters need to be converted into Unicode encoding form
    # first.
    barcodes = pyzbar.decode(gray)
    for barcode in barcodes:
        # Extract the position of the bounding box of the QR code
        (x, y, w, h) = barcode.rect
        # Draw the bounding box of the barcode in the image
        cv.rectangle(image, (x, y), (x + w, y + h), (225, 0, 0), 5)
        encoding = 'UTF-8'
        # To draw it, you need to convert it into a string first
        barcodeData = barcode.data.decode(encoding)
        barcodeType = barcode.type
        # Plot data and types on the image
        piling = Image.fromarray(image)
        # Create a brush
        draw = ImageDraw.Draw(piling)
```

```
# Parameter 1: font file path, parameter 2: font size
fontStyle = ImageFont.truetype(font_path, 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 image to cv2 image
image = cv.cvtColor(np.array(pilimg), cv.COLOR_RGB2BGR)
# Print barcode data and barcode type to terminal
print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
return image
```