3. QR code creation and recognition

1. QR code

1.1. Introduction to QR code

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 the QR code can allow its content to be decoded quickly. QR code not only has large information capacity, high reliability and low cost, but can also 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 important is that the QR code technology is open source.

1.2. Characteristics of QR code

The data values in the QR code contain repeated information (redundant values). Therefore, even if up to 30% of the QR code structure is destroyed, the readability of the QR code is not affected. The storage space of QR code is up to 7089 bits or 4296 characters, including punctuation marks and special characters, which can be written into QR code. In addition to numbers and characters, words and phrases (such as web addresses) can also be encoded. As more data is added to the QR code, the code size increases and the code structure becomes more complex.

1.3. QR code creation and recognition

1), source code path

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode
```

2), installation package (the factory docker image has been installed)

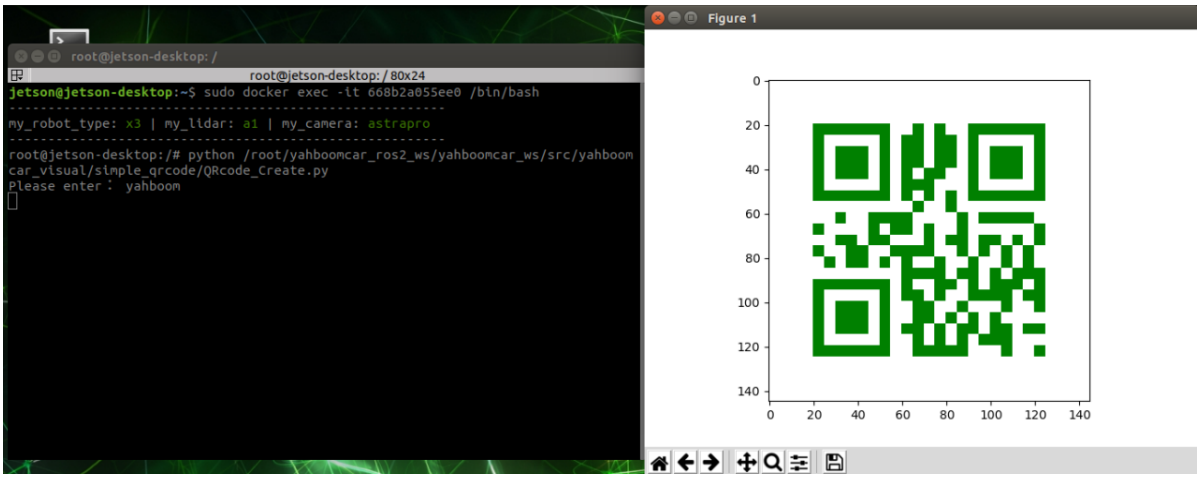
```
python3 -m pip install qrcode pyzbar
sudo apt-get install libzbar-dev
```

3) Create QRcode_Create.py

Enter docker, open the terminal and enter,

```
cd /root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode/
pythonQRcode_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,



Take out your phone and try to scan the QR code that appears on the right. The scan result will be the characters yahboom.

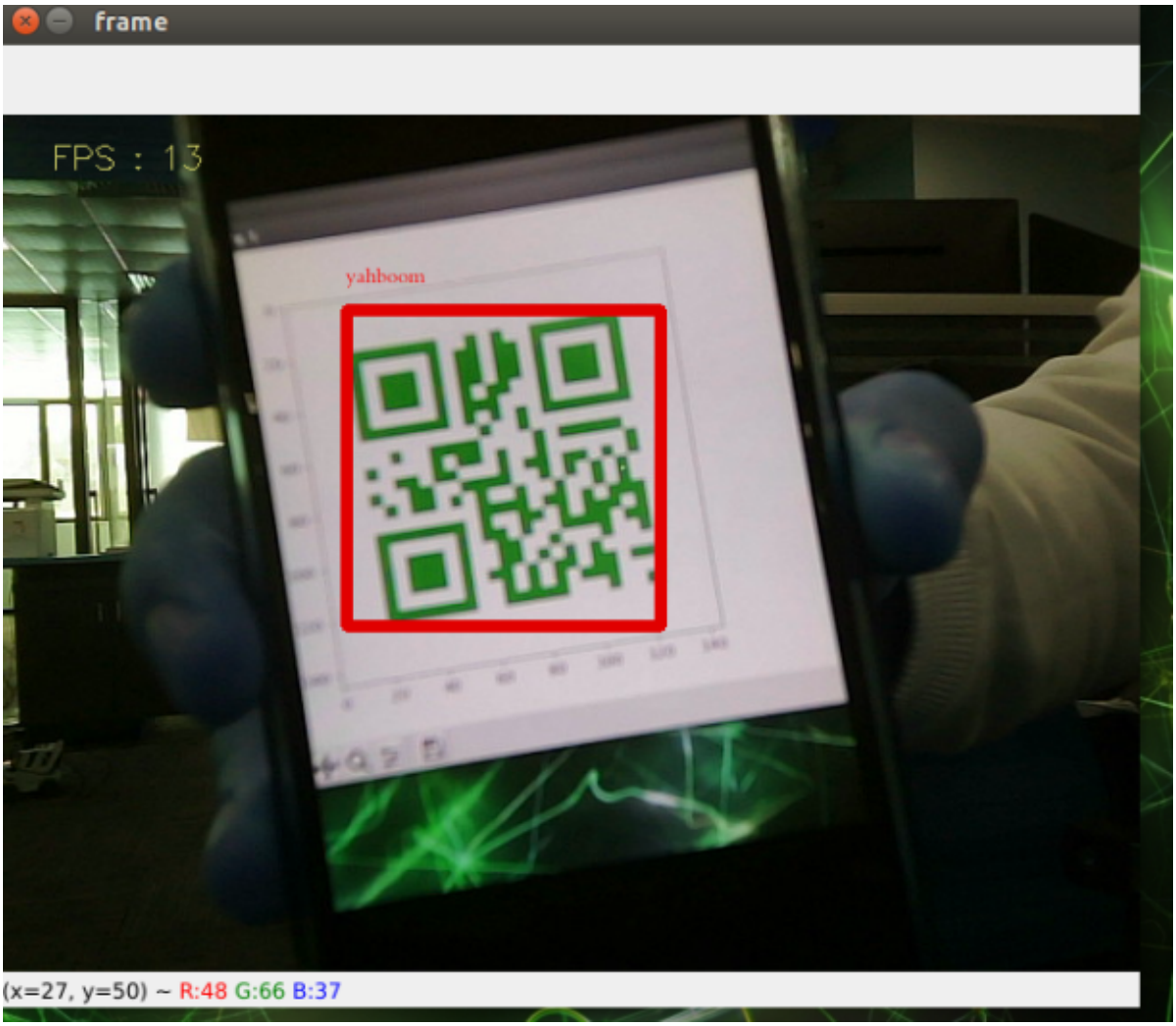
Source code analysis,

```
#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 1~40, controlling the size of the QR code
(the minimum value is 1, which is a 12×12 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 the QR code. Possible
values are 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.
    ROR_CORRECT_H: About 30% or less of errors can be corrected.
box_size: Controls 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)'''
#qrcode QR 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 pictures
# 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_ros2_ws/yahboomcar_ws/src/yahboomcar_visual/simple_qrcode/
pythonQRcode_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)
# You need to convert the output Chinese characters into Unicode encoding 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
     # Draw the data and types on the image
```

```
        pilimg = Image.fromarray(image)
        #Create brush
        draw = ImageDraw.Draw(pilimg)
        # 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 the terminal
        print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))
        return image
```