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

picture	Parse
	**Positioning** markings indicate the direction of the QR code.
	Alignment markings If the QR code is large, these additional elements help with positioning.
	**pattern** With these lines, the scanner can identify how big the matrix is.
	**Version information** (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.
	**Format** information Format patterns contain information about fault tolerance and data mask patterns and make scanning codes easier.
	**Data** and error correction keys These modes hold the actual data.
	**Quiet** zone This zone is very important for the scanner, its role is to separate itself from the surrounding.

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

```
~/ascam_ros2_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 ~/ascam_ros2_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
```

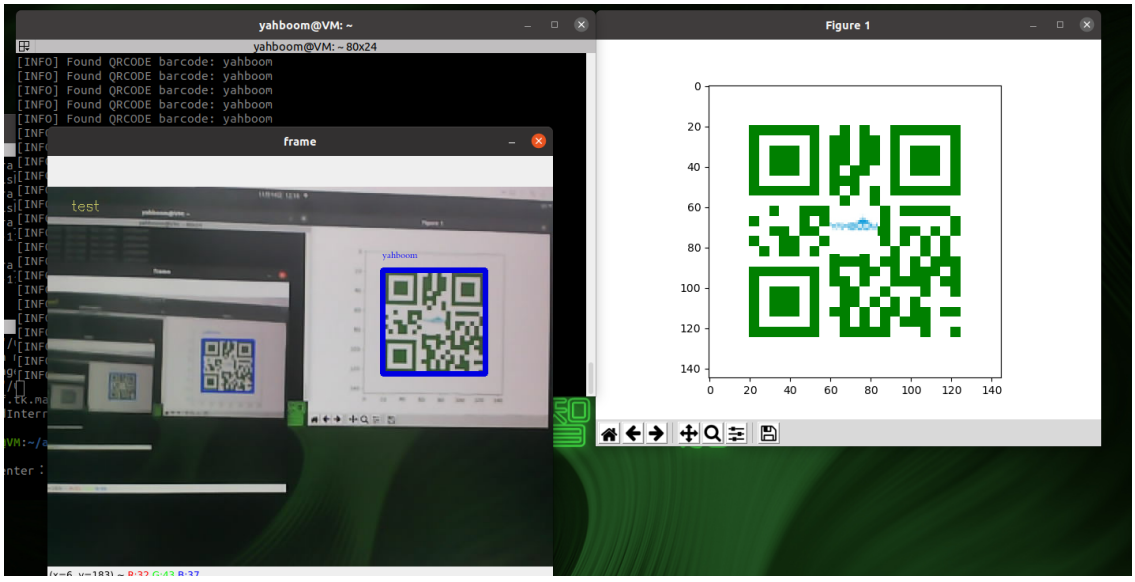
```
'''version: An integer with a value of 1 to 40, which controls the size of the QR code (the minimum value is 1, which is a 12×12 matrix).
If you want the program to determine it automatically, set the value to None and use the fit parameter.
error_correction: Controls the error correction function of the QR code. The following 4 constants can be used.
ERROR_CORRECT_L: About 7% or less 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

```
~/ascam_ros2_ws/src/yahboomcar_visual/yahboomcar_visual/QRcode_Parsing.py
```

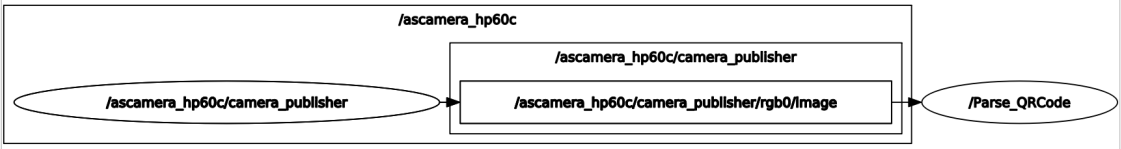
  - Start

```
#Camera startup
ros2 launch ascamera hp60c.launch.py
ros2 run yahboomcar_visual QRcode_Parsing
```



  - View node communication graph

```
ros2 run rqt_graph rqt_graph
```



- Core source code analysis

```
self.sub_img =
self.create_subscription(CamImage,'/ascamera_hp60c/camera_publisher/rgb0/ima
ge',self.handleTopic,100)
frame = self.decodeDisplay(frame)
def decodeDisplay(self,image):
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    # 需要先把输出的中文字符转换成Unicode编码形式
    # The output Chinese characters need to be converted to Unicode encoding
first
    barcodes = pyzbar.decode(gray)
    for barcode in barcodes:
        # 提取二维码的边界框的位置
        # Extract the position of the boundary box of the TWO-DIMENSIONAL
code
        # 画出图像中条形码的边界框
        # 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), (225, 0, 0), 5)
        encoding = 'UTF-8'
        # 画出来，就需要先将它转换成字符串
        # to draw it, you need to convert it to a string
        barcodeData = barcode.data.decode(encoding)
        barcodeType = barcode.type
        # 绘出图像上数据和类型
        # Draw the data and type on the image
        pilimg = Image.fromarray(image)
        # 创建画笔
        # create brush
        draw = ImageDraw.Draw(pilimg)  # 图片上打印  Print on picture
        # 参数1：字体文件路径，参数2：字体大小
        # parameter 1: font file path, parameter 2: font size
        fontStyle =
ImageFont.truetype("/home/yahboom/ascam_ros2_ws/src/yahboomcar_visual/yahboo
mcar_visual/Block_Simplified.TTF", size=12, encoding=encoding)
        # # 参数1：打印坐标，参数2：文本，参数3：字体颜色，参数4：字体
        # 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图片转cv2 图片
        # PIL picture to CV2 picture
```

```
        image = np.array(pilimg)
        # 向终端打印条形码数据和条形码类型
        # Print barcode data and barcode type to terminal
        print("[INFO] Found {} barcode: {}".format(barcodeType,
barcodeData))
        return image
```