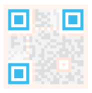
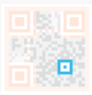
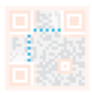
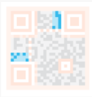
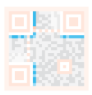
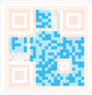
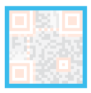
QR code

1.Introduction to QR Code

QR code is a type of two-dimensional barcode. QR is 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.The structure of a QR code

Pic	Analysis
	Positioning markings: Indicate the direction of the QR code.
	Alignment markings: If the QR code is large, these additional elements help with positioning.
	Timing pattern: Through these lines, the scanner can identify how big the matrix is.
	Version information: This specifies the version number of the QR code being used. There are currently 40 different versions of QR codes. Versions used in the sales industry are usually 1-7.
	Format information: The format pattern contains information about error tolerance and data masking patterns and makes scanning the code easier.
	Data and error correction keys: These modes hold the actual data.
	Quiet zone: This area is very important for the scanner, as its function is to separate itself from the surrounding areas.

2.1 Features of QR Code

The data values in QR Code contain repeated information (redundant values). Therefore, even if up to 30% of the QR Code structure is destroyed, it will not affect the readability of the QR Code. The storage space of 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.

2.2 QR code creation and recognition

Code path: ~/jetcobot_ws/src/jetcobot_visual/simple_qrcode

Input command to install.

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

- Create

Create a qrcode object

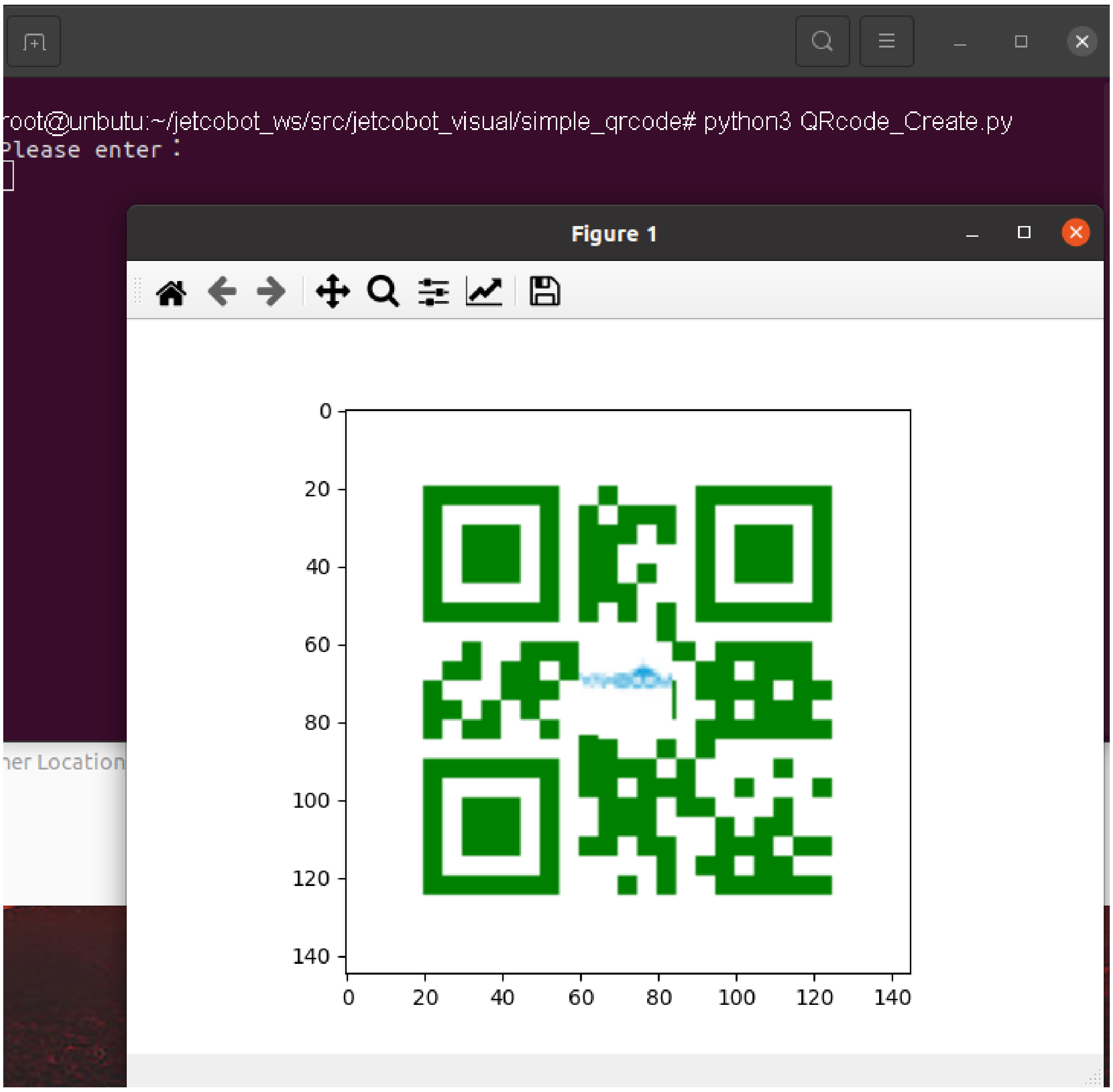
```
    '''
    Parameter meaning:
    version: An integer from 1 to 40 that controls the size of the QR code (the
minimum value is 1, which is a 12×12 matrix).
    If you want the program to determine it automatically, set the value to None
and use the fit parameter.
    error_correction: Controls the error correction function of the QR code. The
following 4 constants are available.
    ERROR_CORRECT_L: About 7% or less errors can be corrected.
    ERROR_CORRECT_M (default): About 15% or less errors can be corrected.
    ROR_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 boundary) (the
    default is 4, which is the minimum value specified by the relevant
standards)
    '''
    qr = qrcode.QRCode( version=1,
error_correction=qrcode.constants.ERROR_CORRECT_H, box_size=5, border=4,)
```

QR code to add logo

```
    # If the logo address exists, add the logo image
    my_file = Path(logo_path)
    if my_file.is_file(): img = add_logo(img, logo_path)
```

```
Just use python3 + py file to execute, then enter the content to be generated and
press Enter to confirm.
```

```
cd ~/jetcobot_ws/src/jetcobot_visual/simple_qrcode
python3 QRcode_Create.py
```



```
root@unbutu:~/jetcobot_ws/src/jetcobot_visual/simple_qrcode# python3 QRcode_Create.py
Please enter：
```

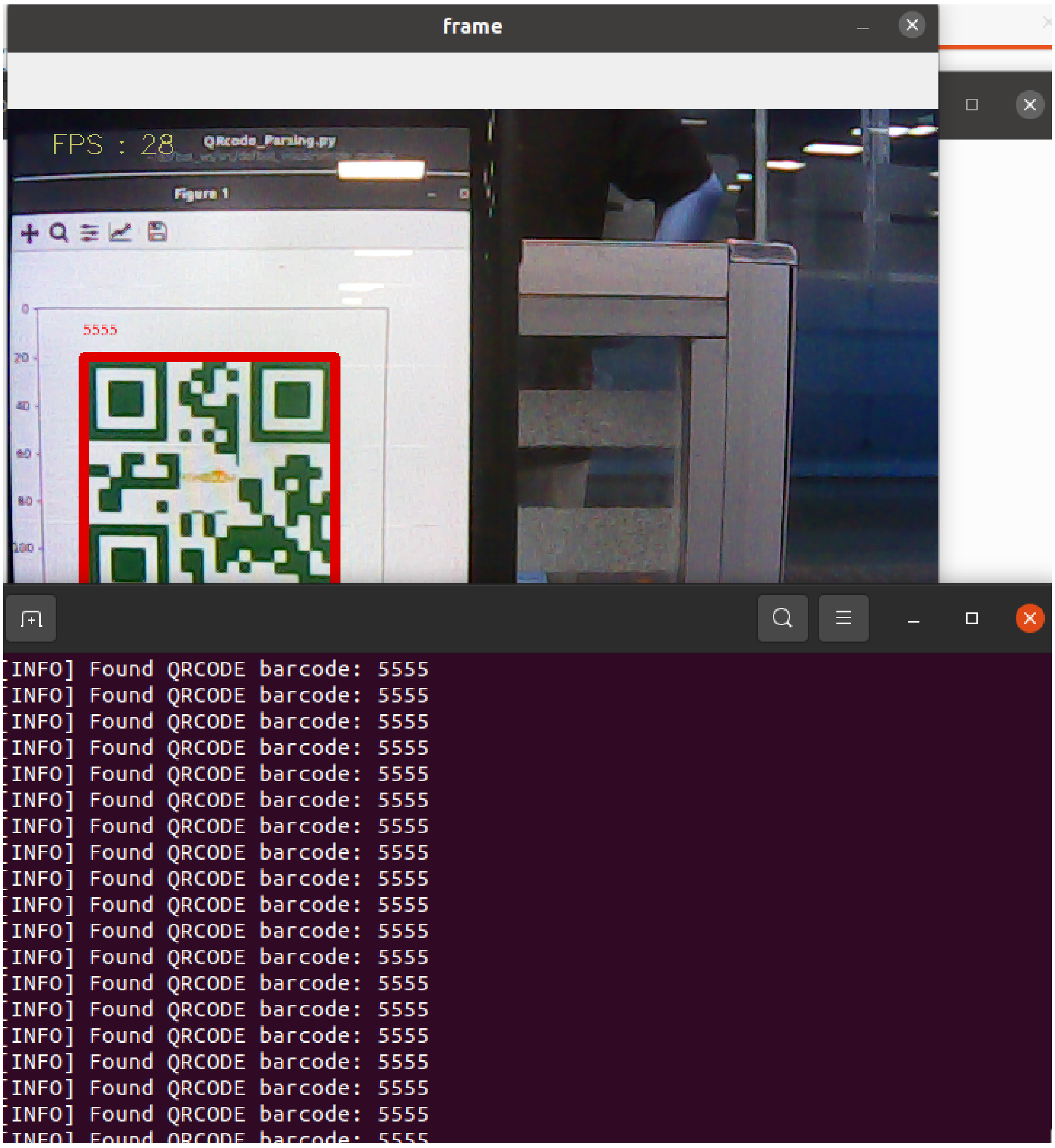
- Identification

```
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
        # Plot data and types on the image
        pilimg = Image.fromarray(image)
        # Create a 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
```

- Effect Demonstration

```
Just use python3 + py file to execute
```

```
cd ~/jetcobot_ws/src/jetcobot_visual/simple_qrcode
python3 QRcode_Parsing.py
```



```
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
[INFO] Found QRCODE barcode: 5555
```