1.QR code

Orin board users can directly run the program by opening the terminal and entering the tutorial command. Jetson-Nano motherboard users need to first enter a Docker container and then enter the tutorial command in the Docker container to start the program.

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 QR code structure

Picture	Analyze
	**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 large the matrix is.
	**Version information**:Here specifies the version number of the QR code being used. Currently there are 40 different version numbers of QR codes. Version numbers used in the sales industry are usually 1-7.
	**Format information**:Format patterns contain information about fault tolerance and data masking patterns and make scanning code easier.
	**Data and error correction keys**:These modes save actual data.
	**Quiet zone**: This area is very important for the scanner. Its function is to separate itself from the surroundings.

1.2.1 Features 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.

1.2.2 QR code creation and recognition

Source code path：

```
#Jetson-Nano users need to enter the Docker container to view this.
~/dofbot_pro_ws/src/dofbot_pro_vision/dofbot_pro_vision/
```

Install

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

- Create

Create qrcode object

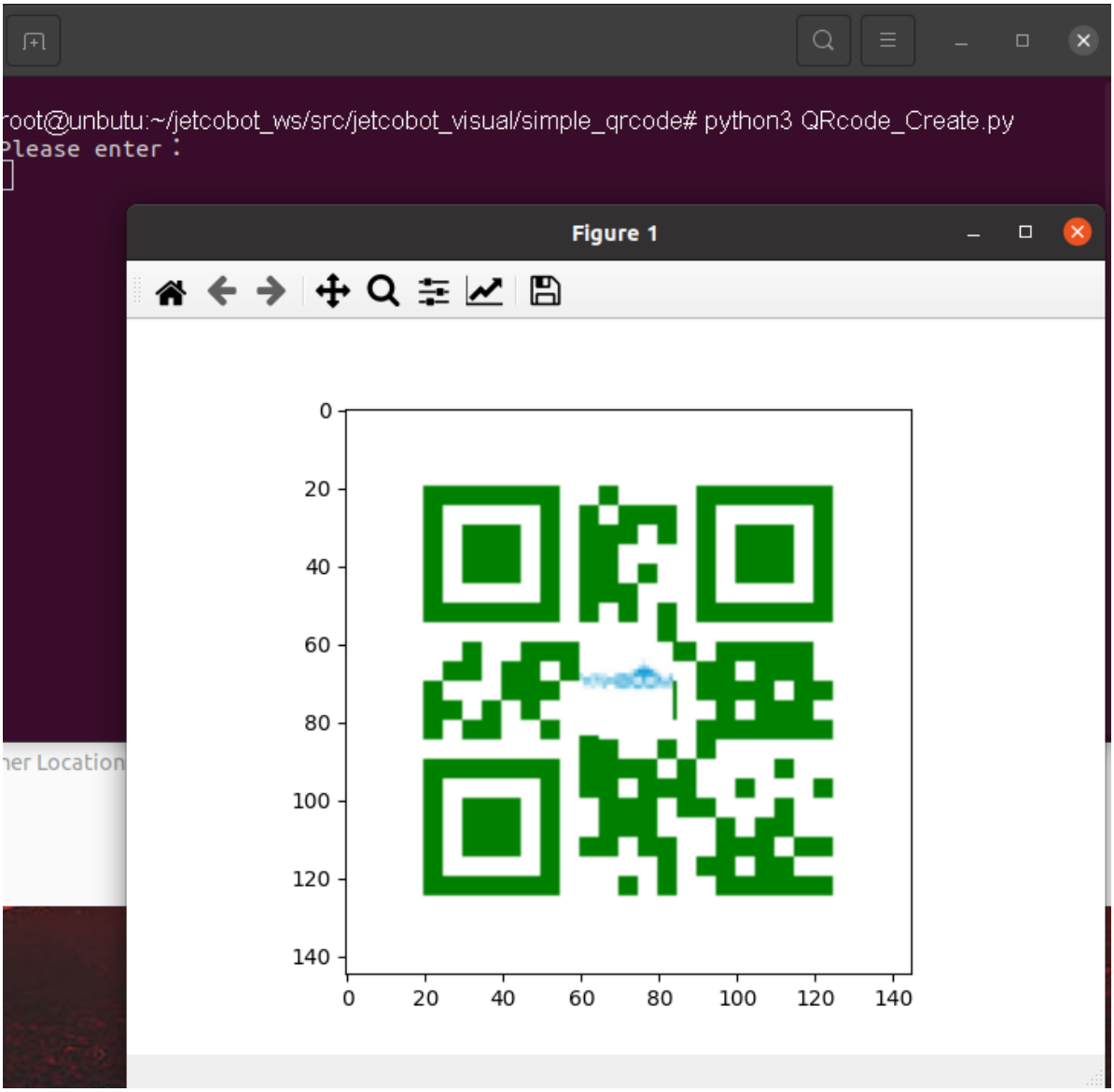
```
    '''
    Parameter meaning:
    version: an integer with a value from 1 to 40, controlling the size of the QR
code (the minimum value is 1, which is a 12×12 matrix).
            If you want the program to determine this automatically, set the
value to None and use the fit argument.
    error_correction: Controls the error correction function of the QR code.
Possible values are the following 4 constants.
    ERROR_CORRECT_L: About 7% or less of the 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)
    '''
    qr = qrcode.QRCode( version=1,
error_correction=qrcode.constants.ERROR_CORRECT_H, box_size=5, border=4,)
```

qrcode 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)
```

Note: When using Chinese, you need to add Chinese characters

```
ros2 run dofbot_pro_vision create_qrcode
```



- Recognize

```
def decodeDisplay(image, font_path):
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    # It is necessary to convert the output Chinese characters 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
        # 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
```

- Effect demonstration

```
#If using a Jetson-Nano motherboard, you need to start the camera. The startup
command is `ros2 launch orbbec_camera dabai_dcw2.launch.py`
ros2 run dofbot_pro_vision parse_qrcode
```