3. Image processing and drawing text line segments

3.1. OpenCV image grayscale processing

1. Image grayscale

The process of converting a color image into a grayscale image is the grayscale processing of the image. The color of each pixel in a color image is determined by three components: R, G, and B, and each component can take a value of 0-255, so there can be more than 16 million pixels (256*256*256=1677256) The range of color changes. The grayscale image is a special color image with the same three components of R, G, and B. The variation range of one pixel is 256. Therefore, in digital image processing, images in various formats are generally converted into grayscale. image to make subsequent images less computationally intensive. The description of a grayscale image, like a color image, still reflects the distribution and characteristics of the overall and local chroma and highlight levels of the entire image.

2. Image grayscale processing

Grayscale processing is the process of converting a color image into a grayscale image. The color image is divided into three components: R, G, and B, which display various colors such as red, green, and blue respectively. Grayscale is the process of making the color R, G, and B components equal. Pixels with large grayscale values are brighter (the maximum pixel value is 255, which is white), and vice versa (the lowest pixel is 0, which is black).

The core idea of image grayscale is R = G = B. This value is also called grayscale value.

1. Maximum value method: Make the converted values of R, G, and B equal to the largest of the three values before conversion, that is: R=G=B=max (R, G, B). The grayscale image converted by this method is very bright.
2. Average method: The values of R, G, and B after conversion are the average values of R, G, and B before conversion. That is: R=G=B=(R+G+B)/3. This method produces softer grayscale images.

In OpenCV, use cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY) to grayscale the image

3. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_1.py
```

Raspberry Pi 5

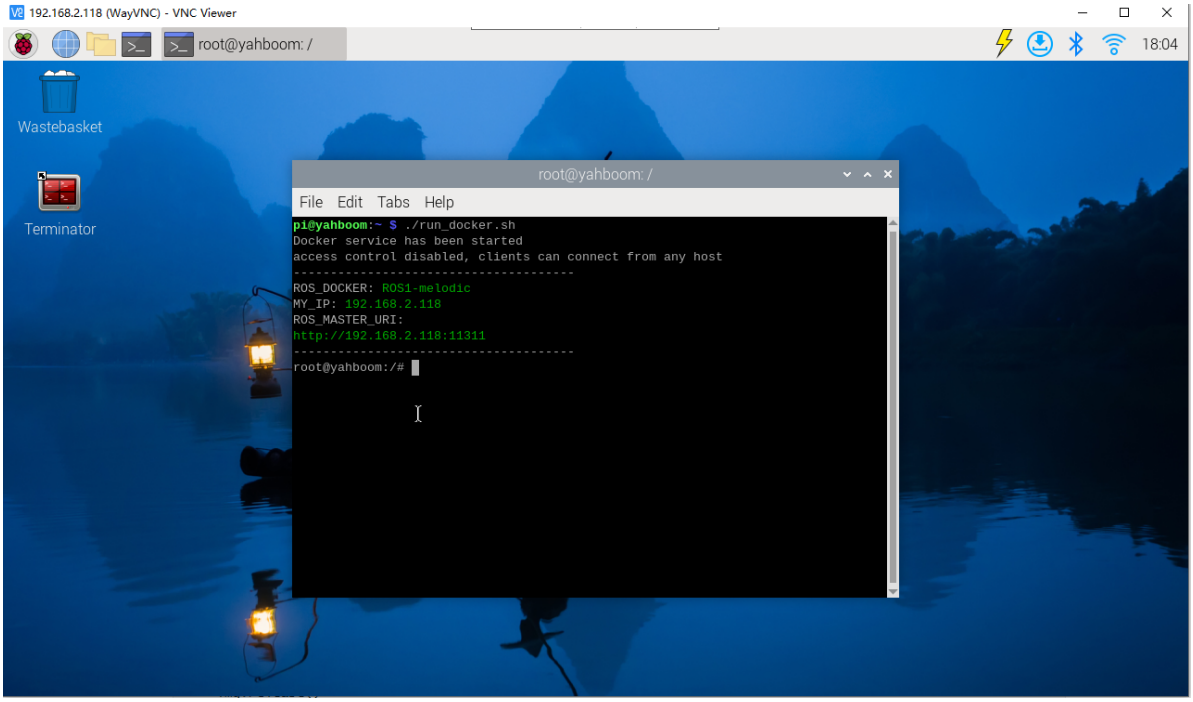
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

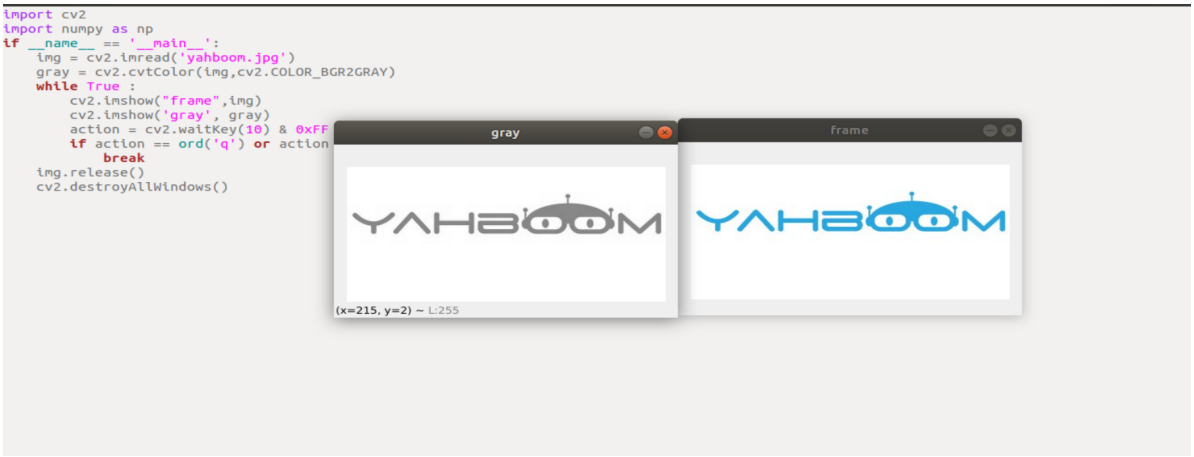
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_1.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
while True :
cv2.imshow("frame",img)
cv2.imshow('gray', gray)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



3.2. OpenCV image binarization processing

1. The core idea of binarization

Set a threshold, and the value greater than the threshold is 0 (black) or 255 (white), so that the image is called a black and white image. The threshold can be fixed or adaptive. The adaptive threshold is generally a comparison of a point pixel with the mean value of regional pixels or the weighted sum of Gaussian distribution at this point, in which a difference value may or may not be set.

2. Python-OpenCV provides a threshold function: cv2.threshold (src, threshold, maxValue, thresholdType)

Parameter meaning:

src: original image

threshold: current threshold

maxVal: maximum threshold, generally 255

thresholdType: Threshold type, generally has the following values

enum ThresholdTypes {
    THRESH_BINARY = 0, #The gray value of pixels greater than the threshold is set to maxValue (for example, the maximum 8-bit gray value is 255), and the gray value of pixels whose gray value is less than the threshold is set to 0.
    THRESH_BINARY_INV = 1, #The gray value of pixels greater than the threshold is set to 0, and the gray value of pixels less than the threshold is set to maxValue.

THRESH_TRUNC = 2, #The gray value of pixels greater than the threshold is set to 0, and the gray value of pixels less than the threshold is set to maxValue.

THRESH_TOZERO = 3, #If the gray value of the pixel is less than the threshold, no change will be made, and for the part greater than the threshold, the gray value will all become 0.

THRESH_TOZERO_INV = 4 #If the gray value of the pixel is greater than the threshold, no change will be made. If the gray value of the pixel is less than the threshold, all the gray values will be changed to 0.

}

return value:

retval: consistent with parameter thresh

dst: result image

Note: Before binarization, we need to grayscale the color image to obtain a grayscale image.

3. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_2.py
```

Raspberry Pi 5

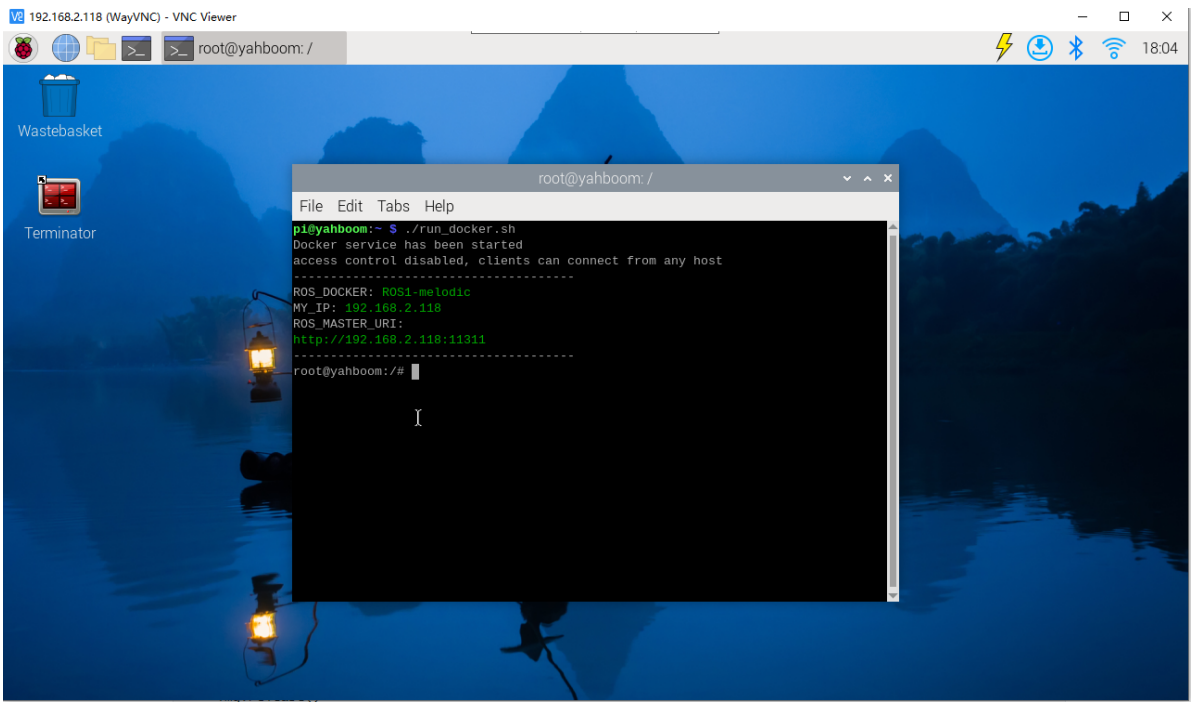
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

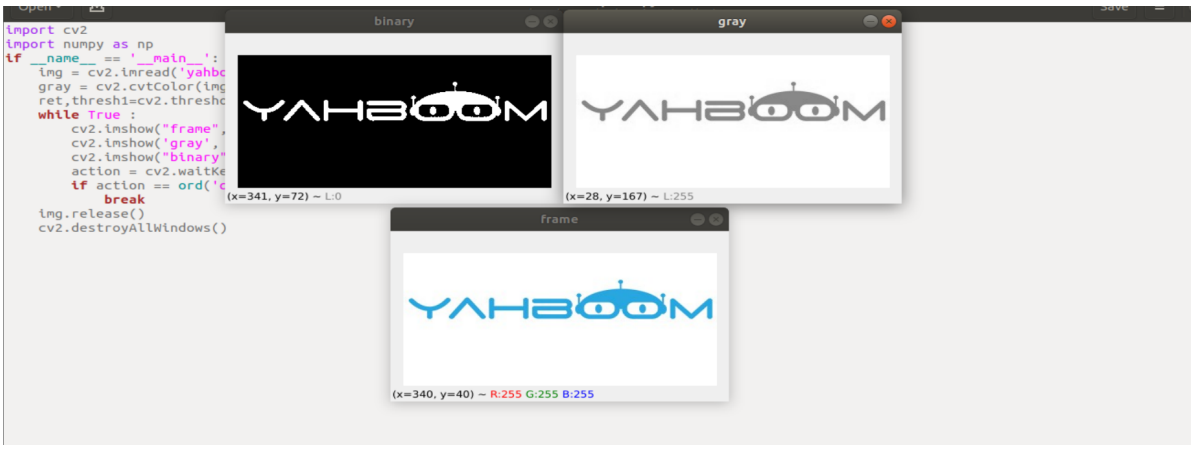
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_2.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret,thresh1=cv2.threshold(gray,180,255,cv2.THRESH_BINARY_INV)
while True :
cv2.imshow("frame",img)
cv2.imshow('gray', gray)
cv2.imshow("binary",thresh1)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



3.3. OpenCV image edge detection

1. The purpose of image edge detection

Significantly reduce the data size of the image while retaining the original image attributes. There are currently many algorithms for edge detection. Although the Canny algorithm is old, it can be said that it is a standard algorithm for edge detection and is still widely used in research.

2. Canny edge detection algorithm

Among currently commonly used edge detection methods, the Canny edge detection algorithm is one of the methods that is strictly defined and can provide good and reliable detection. Because it has the advantages of meeting the three criteria of edge detection and being simple to implement, it has become one of the most popular algorithms for edge detection.

The Canny edge detection algorithm can be divided into the following 5 steps:

(1). Use Gaussian filter to smooth the image and filter out noise

(2). Calculate the gradient intensity and direction of each pixel in the image

(3). Apply non-maximum suppression to eliminate spurious responses caused by edge detection

(4). Apply double-threshold detection to determine real and potential edges

(5). Finally complete edge detection by suppressing isolated weak edges.

3. How do we implement it in OpenCV? It's very simple, divided into three steps

(1). Image grayscale: gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

(2). Gaussian filtering (noise reduction processing) image: GaussianBlur (src, ksize, sigmaX [, dst [, sigmaY [, borderType]]]) -> dst

Parameter meaning:

src: input image, usually a grayscale image

ksize: Gaussian kernel size

sigmaX: Gaussian kernel standard deviation in the X direction

sigmaY: Gaussian kernel standard deviation in Y direction

dst: processed image

(3). The Canny method is used to process the image: edges=cv2.Canny(image, threshold1, threshold2[, apertureSize[, L2gradient]])

```
Parameter meaning:
```

edges: calculated edge image

image: The calculated edge image, usually the image obtained after Gaussian processing

threshold1: the first threshold in the process

threshold2: the second threshold during processing

apertureSize: aperture size of Sobel operator

L2gradient: The flag for calculating the gradient magnitude of the image. The default value is False. If True, the calculation is performed using the more accurate L2 norm (that is, the sum of the squares of the derivatives in both directions and then the square root), otherwise the L1 norm is used (directly adding the absolute values of the derivatives in both directions).

4. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_3.py
```

Raspberry Pi 5

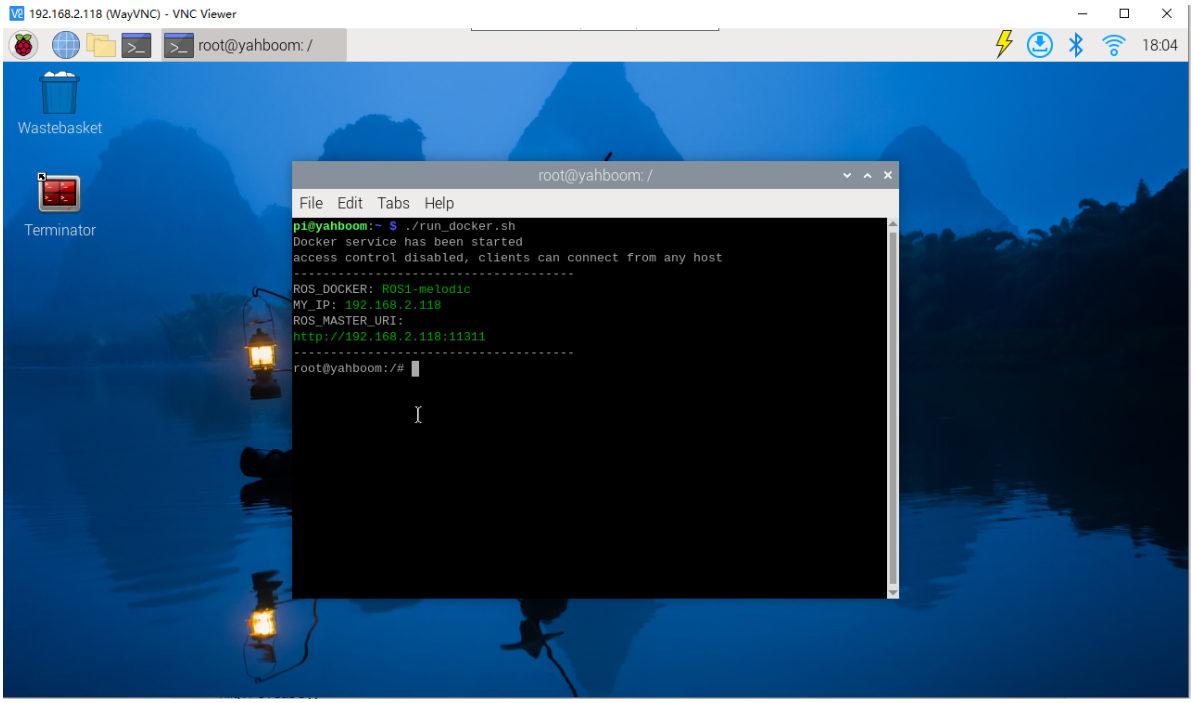
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

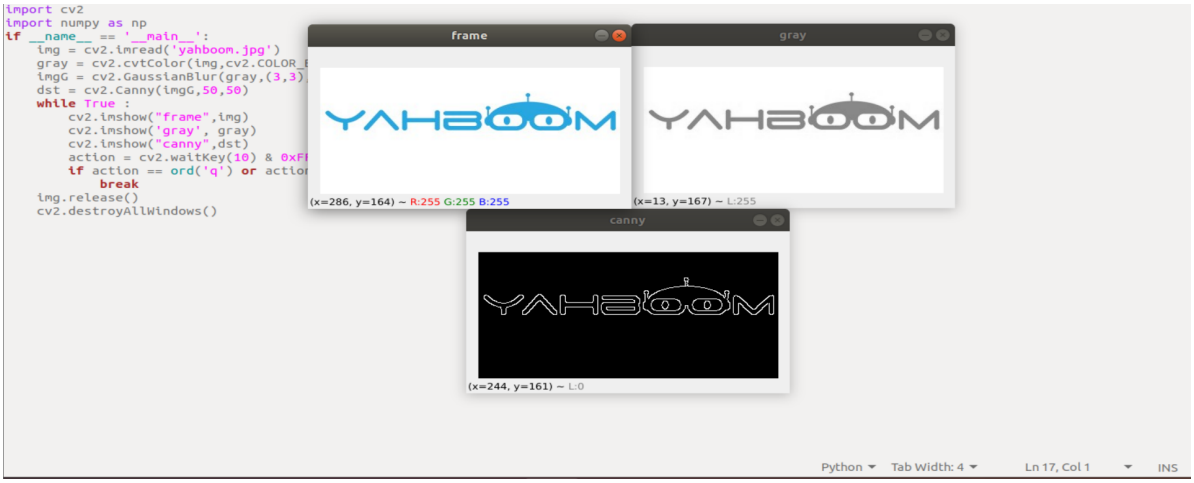
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_3.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgG = cv2.GaussianBlur(gray,(3,3),0)
dst = cv2.Canny(imgG,50,50)
while True :
cv2.imshow("frame",img)
cv2.imshow('gray', gray)
cv2.imshow("canny",dst)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



3.4. OpenCV line segment drawing

1. When using OpenCV to process images, we sometimes need to draw line segments, rectangles, etc. on the image. Used in OpenCV

The cv2.line(dst, pt1, pt2, color, thickness=None, lineType=None, shift=None) function draws line segments.

Parameter meaning:

dst: output image.

pt1, pt2: required parameters. The coordinate points of the line segment represent the starting point and the ending point respectively.

color: required parameter. Used to set the color of line segments

thickness: optional parameter. Used to set the width of the line segment

lineType: optional parameter. Used to set the type of line segment, optional 8 (8 adjacent connecting lines - default), 4 (4 adjacent connecting lines) and cv2.LINE_AA is anti-aliasing

2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_4.py
```

Raspberry Pi 5

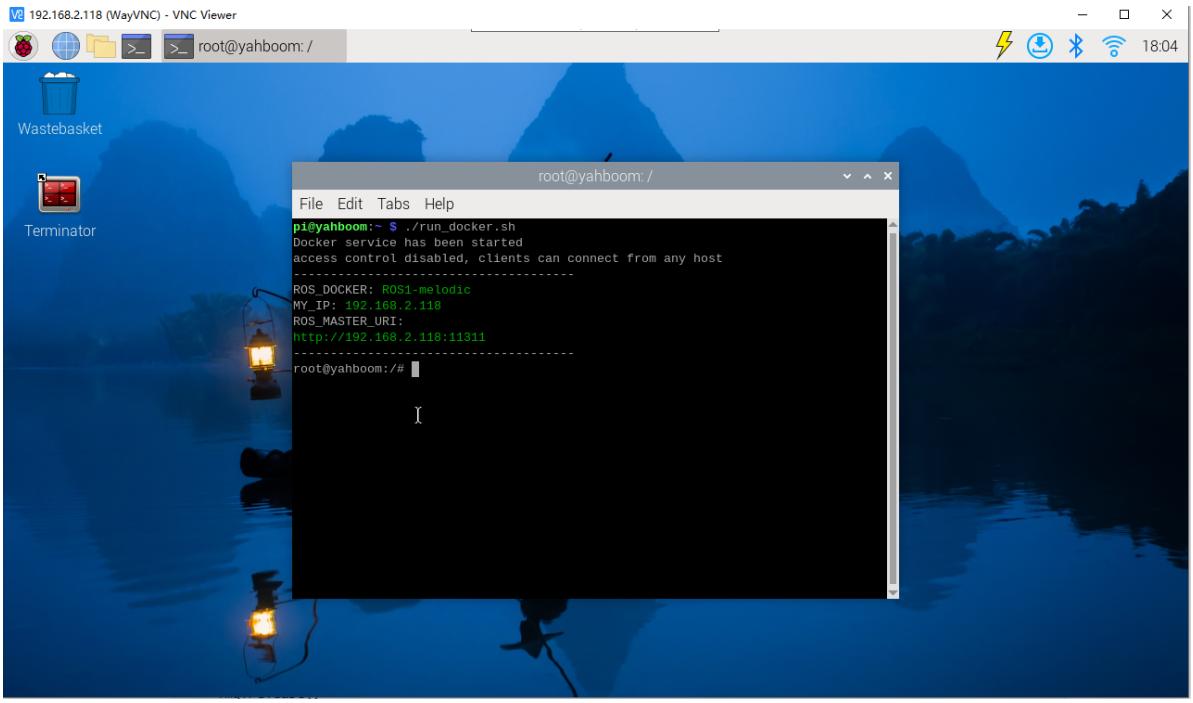
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

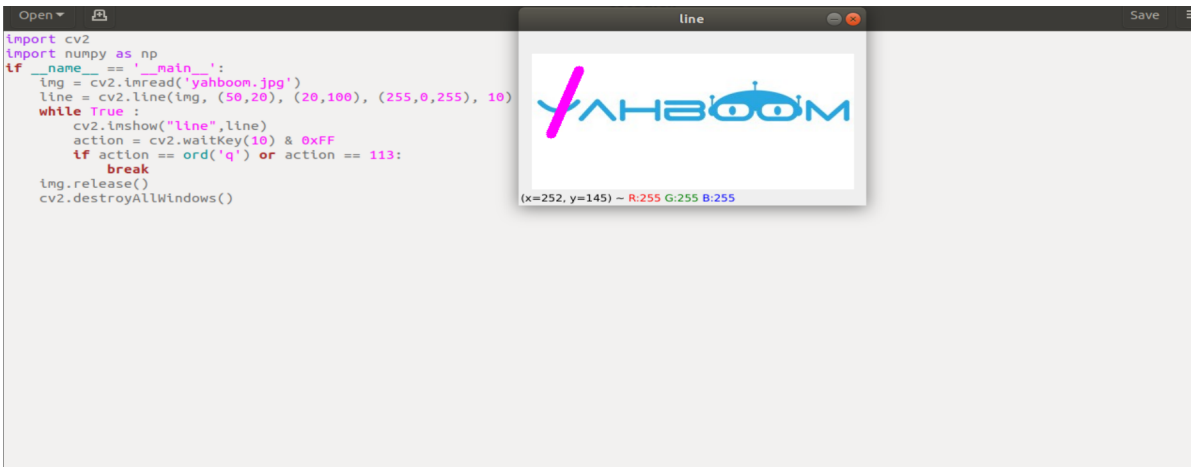
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_4.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
line = cv2.line(img, (50,20), (20,100), (255,0,255), 10)
while True :
cv2.imshow("line",line)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



3.5. OpenCV draws rectangle

1. In OpenCV, drawing a rectangle is used

cv2.rectangle(img, pt1, pt2, color, thickness=None, lineType=None, shift=None)

Parameter meaning:

img: canvas or carrier image

pt1, pt2: required parameters. The vertices of a rectangle represent the vertices and diagonal vertices respectively, that is, the upper left corner and the lower right corner of the rectangle (these two vertices can determine a unique rectangle), which can be understood as diagonals.

color: required parameter. Used to set the color of the rectangle

thickness: optional parameter. Used to set the width of the rectangle side. When the value is a negative number, it means filling the rectangle.

lineType: optional parameter. Used to set the type of line segment, optional 8 (8 adjacent connecting lines - default), 4 (4 adjacent connecting lines) and cv2.LINE_AA is anti-aliasing

2. Code and effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_5.py
```

Raspberry Pi 5

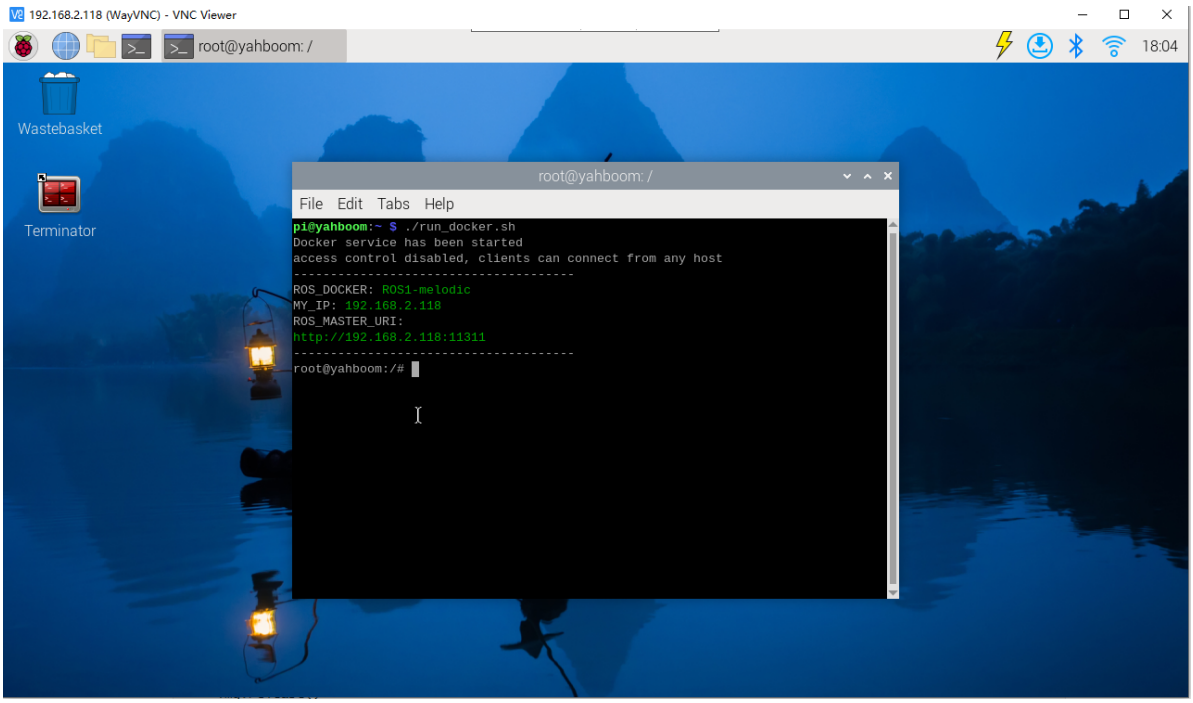
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```

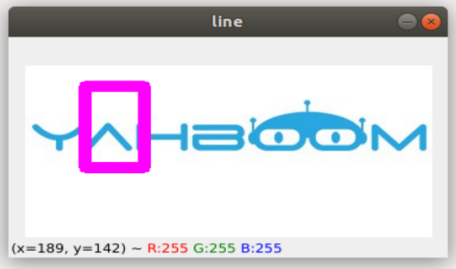


Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_5.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
rect = cv2.rectangle(img, (50,20), (100,100), (255,0,255), 10)
while True :
cv2.imshow("line",rect)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```

```
import cv2
import numpy as np
if __name__ == '__main__':
    img = cv2.imread('yahboom.jpg')
    rect = cv2.rectangle(img, (50,20), (100,100), (255,0,255), 10)
    while True :
        cv2.imshow("line",rect)
        action = cv2.waitKey(10) & 0xFF
        if action == ord('q') or action == 113:
            break
    img.release()
    cv2.destroyAllWindows()
```



3.6. OpenCV draws a circle

1. In OpenCV, drawing a circle is used

cv2.circle(img, center, radius, color[,thickness[,lineType]])

Parameter meaning:

img: painting or carrier image cloth

center: is the coordinate of the center of the circle, format: (50,50)

radius: radius

thickness: line thickness. The default is 1. If -1, it will be filled solid

lineType: line type. The default is 8, connection type. As shown in the following table

Parameters	Description
cv2.FILLED	fill
cv2.LINE_4	4 connection types
cv2.LINE_8	8 connection type
cv2.LINE_AA	Anti-aliasing, this parameter will make the lines smoother

2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_6.py
```

Raspberry Pi 5

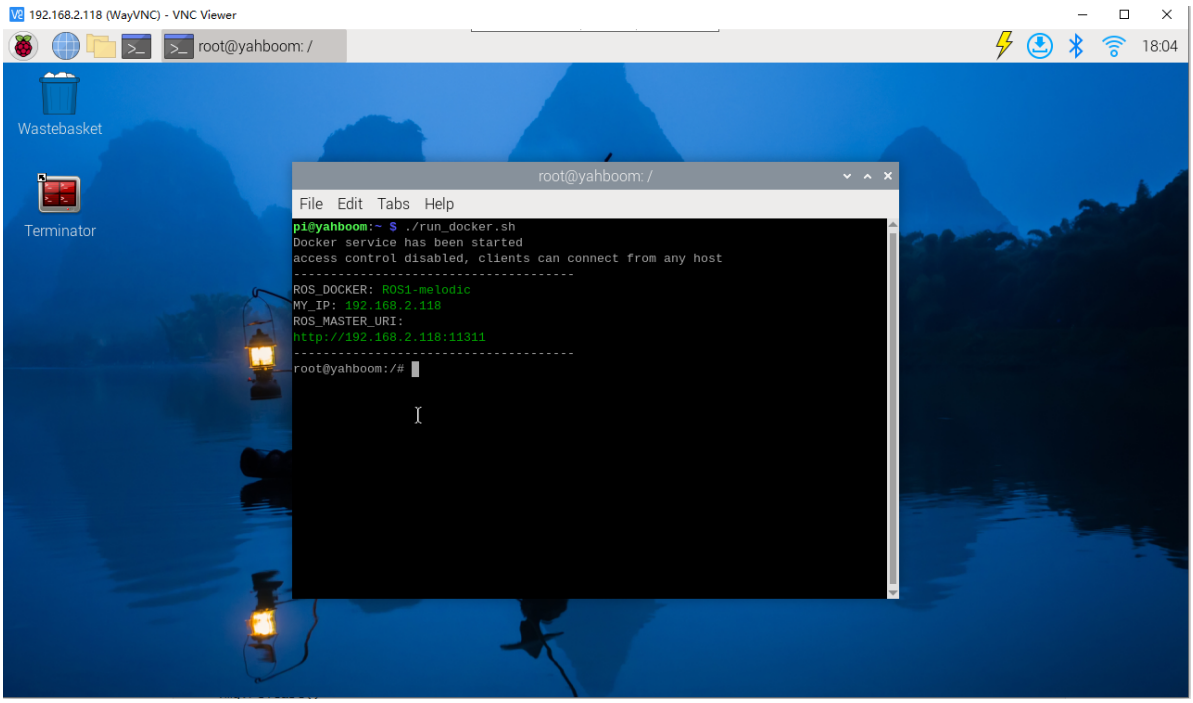
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_6.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
circle = cv2.circle(img, (80,80), 50, (255,0,255), 10)
while True :
cv2.imshow("circle",circle)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```

3.7. OpenCV draws ellipse

1. In OpenCV, the ellipse is drawn using

cv2.ellipse(img, center, axes, angle, StartAngle, endAngle, color[,thickness[,lineType]

Parameter meaning:

center: center point of the ellipse, (x, y)

axes: refers to the short radius and long radius, (x, y)

StartAngle: The angle of the starting angle of the arc

endAngle: the angle of the arc's end angle

For img, color, thickness, and lineType, please refer to the description of the circle.

2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_7.py
```

Raspberry Pi 5

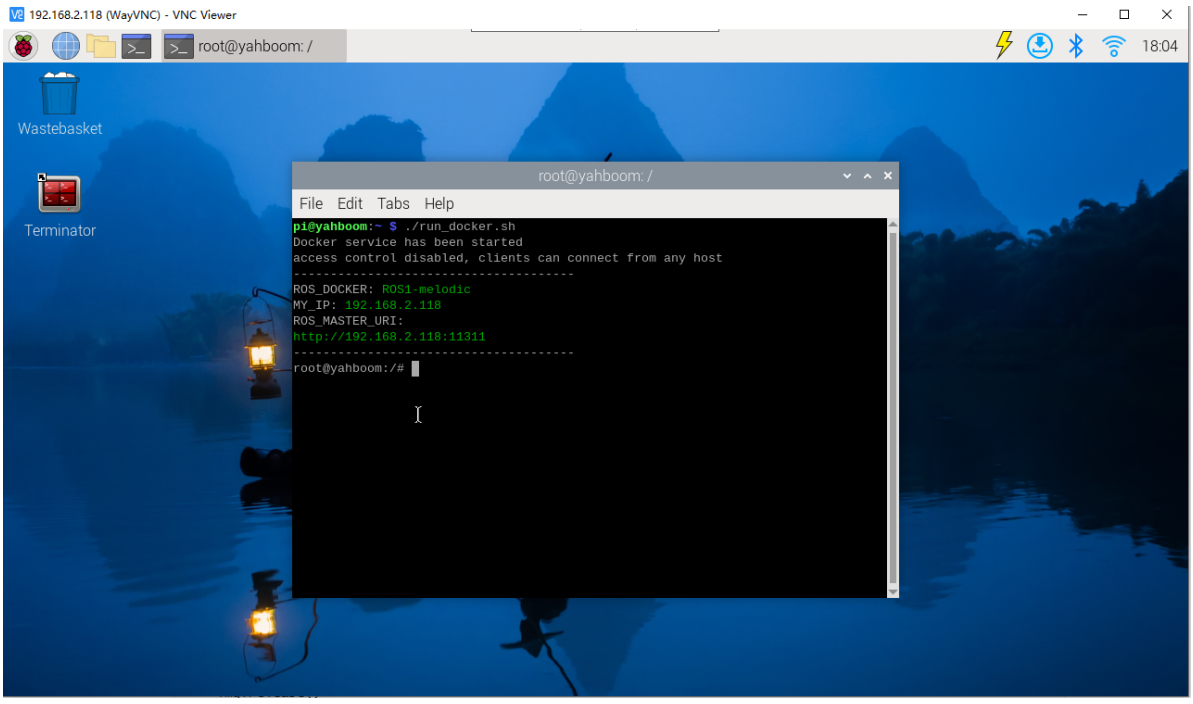
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

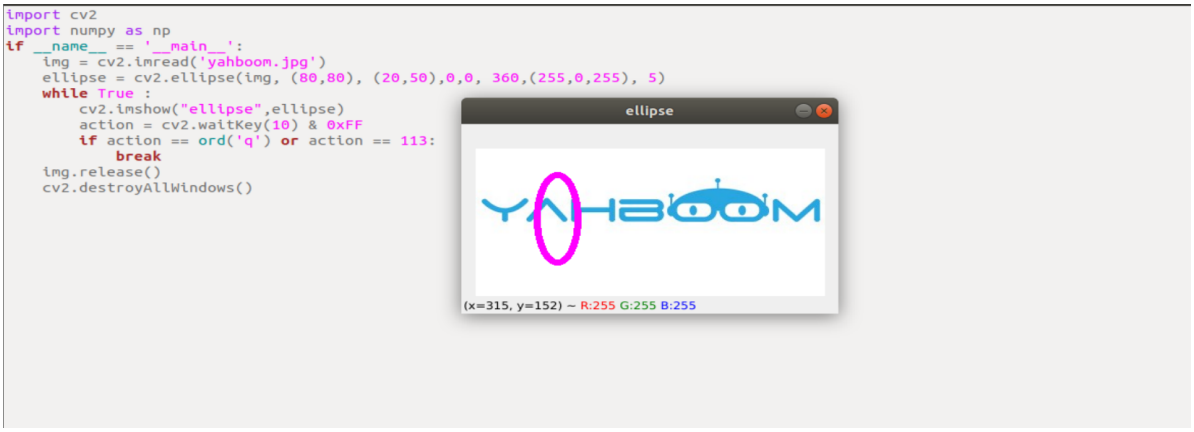
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_7.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
ellipse = cv2.ellipse(img, (80,80), (20,50),0,0, 360,(255,0,255), 5)
while True :
cv2.imshow("ellipse",ellipse)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



3.8. OpenCV draws polygons

1. In OpenCV, polygons are drawn using

cv2.polylines(img,[pts],isClosed, color[,thickness[,lineType]])

Parameter meaning:

pts: vertices of polygon

isClosed: Whether it is closed. (True/False)

Other parameters refer to the drawing parameters of the circle.

2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_8.py
```

Raspberry Pi 5

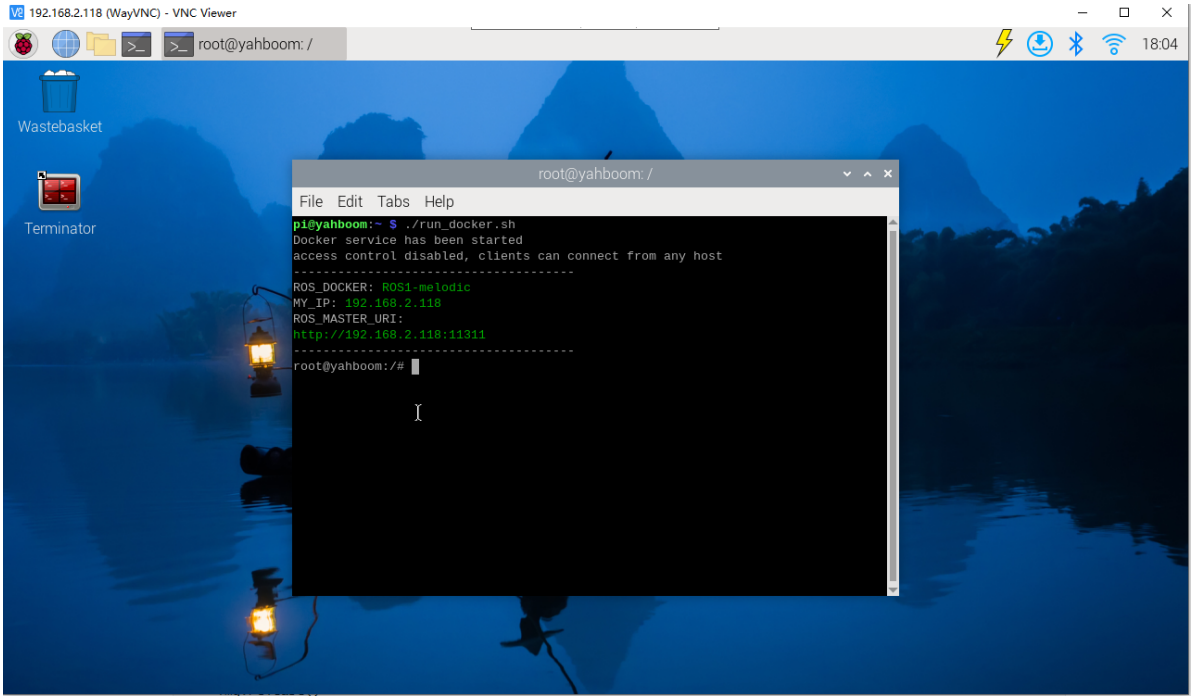
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```

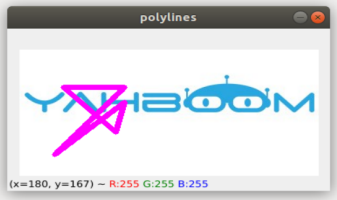


Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_8.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
points = np.array([[120,50], [40,140], [120,70], [110,110], [50,50]], np.int32)
polylines = cv2.polylines(img, [points],True,(255,0,255), 5)
while True :
cv2.imshow("polylines",polylines)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```

```python
import cv2
import numpy as np
if __name__ == '__main__':
    img = cv2.imread('yahboom.jpg')
    points = np.array([[120,50], [40,140], [120,70], [110,110], [50,50]], np.int32)
    polylines = cv2.polylines(img, [points],True,(255,0,255), 5)
    while True :
        cv2.imshow("polylines",polylines)
        action = cv2.waitKey(10) & 0xFF
        if action == ord('q') or action == 113:
            break
    img.release()
    cv2.destroyAllWindows()
```



3.9, OpenCV draws text

1. In OpenCV, drawing text is used

cv2.putText(img, str, origin, font, size, color, thickness)

Parameter meaning:

img: input image

str: drawn text

Origin: coordinates of the upper left corner (integer), which can be understood as where the text starts.

font: font

size: font size

color: font color

thickness: font thickness

Among them, the font is optional

FONT_HERSHEY_SIMPLEX Python: cv.FONT_HERSHEY_SIMPLEX	正常大小sans-serif字体
FONT_HERSHEY_PLAIN Python: cv.FONT_HERSHEY_PLAIN	小尺寸sans-serif字体
FONT_HERSHEY_DUPLEX Python: cv.FONT_HERSHEY_DUPLEX	正常大小的sans-serif字体（比FONT_HERSHEY_SIMPLEX更复杂）
FONT_HERSHEY_COMPLEX Python: cv.FONT_HERSHEY_COMPLEX	正常大小的衬线字体
FONT_HERSHEY_TRIPLEX Python: cv.FONT_HERSHEY_TRIPLEX	正常大小的serif字体（比FONT_HERSHEY_COMPLEX更复杂）
FONT_HERSHEY_COMPLEX_SMALL Python: cv.FONT_HERSHEY_COMPLEX_SMALL	较小版本的FONT_HERSHEY_COMPLEX
FONT_HERSHEY_SCRIPT_SIMPLEX Python: cv.FONT_HERSHEY_SCRIPT_SIMPLEX	手写风格的字体
FONT_HERSHEY_SCRIPT_COMPLEX Python: cv.FONT_HERSHEY_SCRIPT_COMPLEX	更复杂的FONT_HERSHEY_SCRIPT_SIMPLEX变体
FONT_ITALIC Python: cv.FONT_ITALIC	标志为斜体字体

2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python3_9.py
```

Raspberry Pi 5

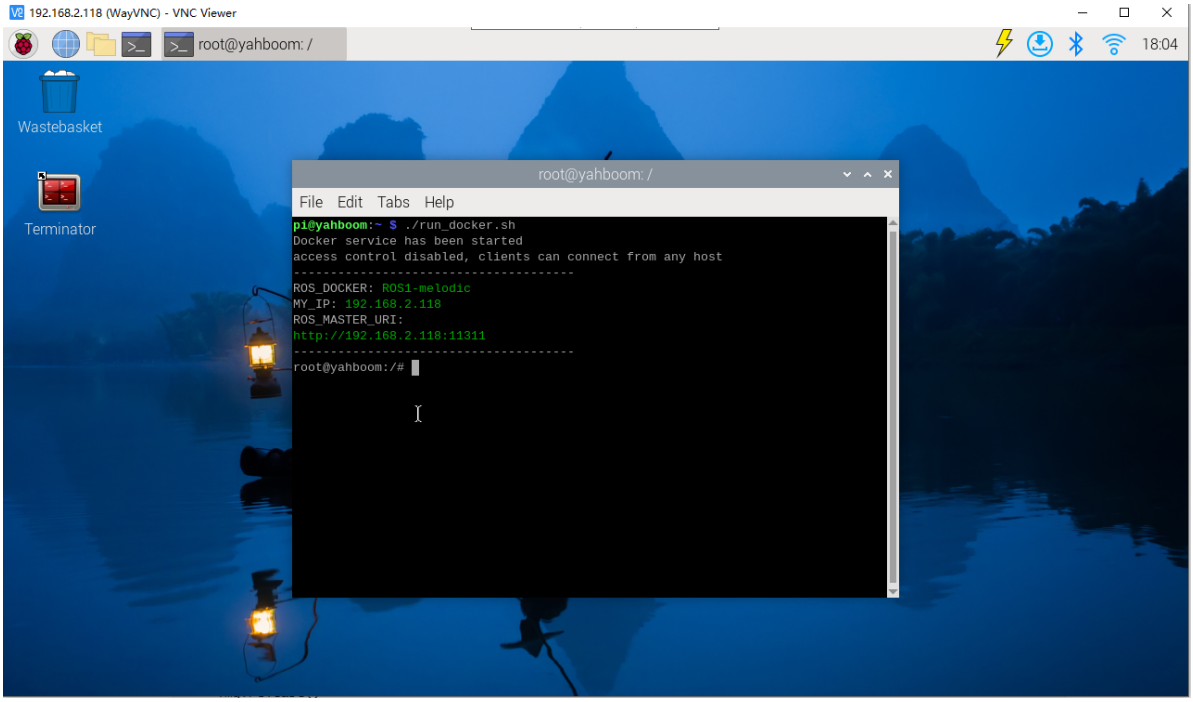
Before running, please confirm that the large program has been permanently closed

Enter docker

Note: If you have a terminal that automatically starts docker, you can directly enter the docker terminal to run the command, and there is no need to manually start docker

Start docker manually

```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python 3_9.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
    img = cv2.imread('yahboom.jpg')
    cv2.putText(img,'This is Yahboom!',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,
(0,200,0),2)
    while True :
        cv2.imshow("img",img)
        action = cv2.waitKey(10) & 0xFF
        if action == ord('q') or action == 113:
            break
    img.release()
    cv2.destroyAllWindows()
```

```python
import cv2
import numpy as np
if __name__ == '__main__':
    img = cv2.imread('yahboom.jpg')
    cv2.putText(img,'This is Yahboom!',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,200,0),2)
    while True :
        cv2.imshow("img",img)
        action = cv2.waitKey(10) & 0xFF
        if action == ord('q') or action == 113:
            break
    img.release()
    cv2.destroyAllWindows()
```