OpenCV image processing

Before running the example program, you need to switch to the directory where the code is located. After SSH connects to the car, run in the terminal,

```
cd /home/sunrise/yahboomcar_ws/src/yahboomcar_astra/opencv_examples
```

3.1, Image grayscale

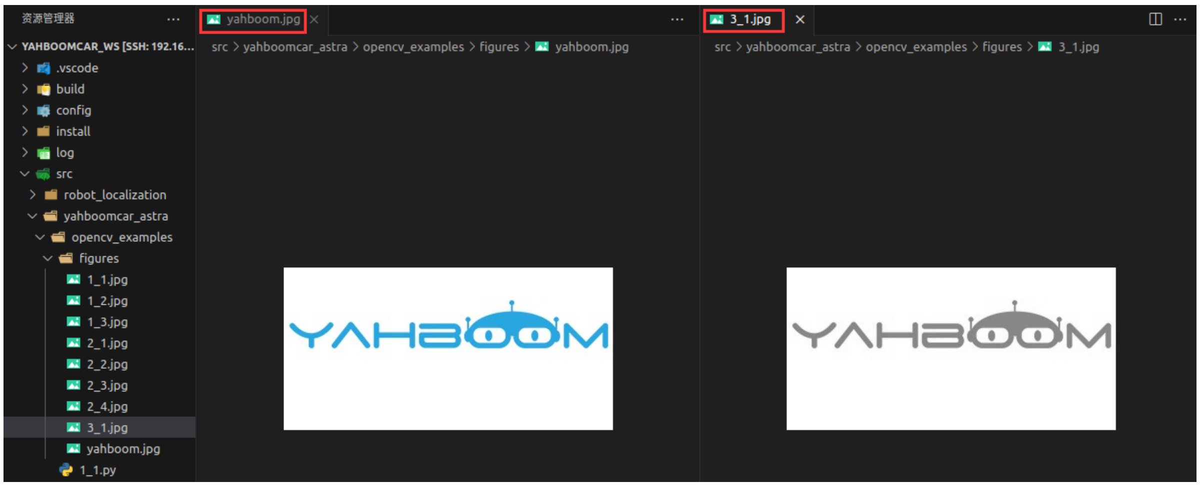
> Grayscale processing is the process of converting a color image into a grayscale image. Color images are divided into three components, R, G, and B, which respectively display various colors such as red, green, and blue. Grayscale is the process of making the R, G, and B components of the color equal. Pixels with large grayscale values are brighter (the maximum pixel value is 255, which is white), and vice versa, they are darker (the lowest pixel value is 0, which is black). The core idea of image grayscale is R = G = B, which is also called grayscale value.
>
> 1. Maximum value method: Make the converted R, G, B values 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 has high brightness.
> 2. Average value method: The converted R, G, B values are the average of the R, G, B values before conversion. That is: R=G=B=(R+G+B)/3. The grayscale image produced by this method is relatively soft.

After switching to the directory where the code is located in the car terminal, run the program,

```
python3 3_1.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # Image grayscale
cv2.imwrite('figures/3_1.jpg', gray)
```

The final image is as shown in the figure,



3.2, Image Binarization

> The core idea of binarization is to set a threshold value, and the value greater than the threshold value is 0 (black) or 255 (white), so that the image is called a black and white image. The threshold value can be fixed or adaptive. The adaptive threshold value is generally a comparison between a pixel at a point and the average value of the pixels in the region with this point as the middle order or the weighted sum of the Gaussian distribution, in which a difference value can be set or not.

After switching the car terminal to the directory where the code is located, run the program,

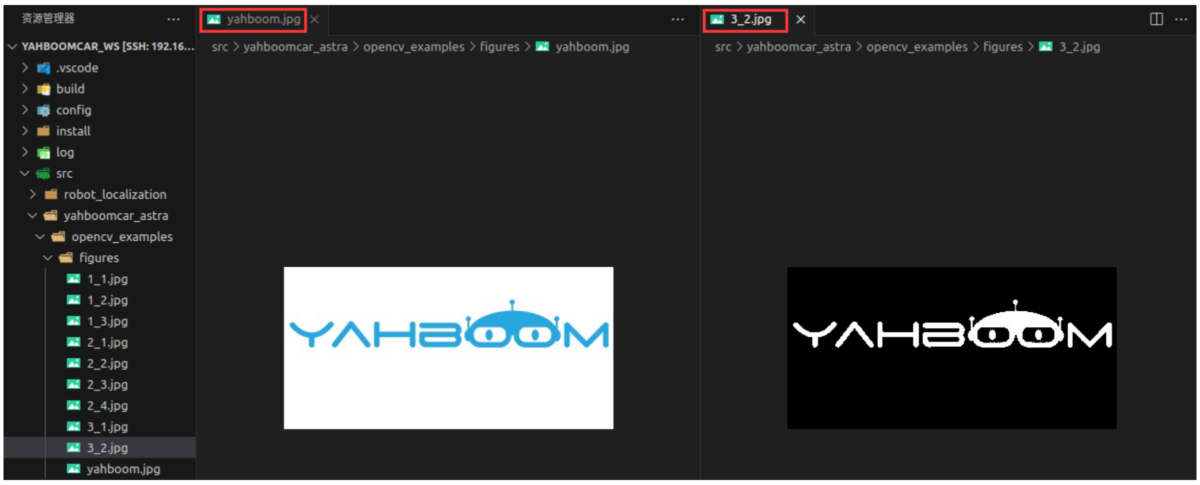
```
python3 3_2.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # Grayscale first
ret, binary = cv2.threshold(gray,180,255,cv2.THRESH_BINARY_INV)
cv2.imwrite('figures/3_2.jpg', binary)

'''
Image Binarization: ret,binary = cv2.threshold(src,thresh,maxVal,thresholdType)
Parameters:
src: original image
thresh: current threshold
maxVal: maximum threshold, usually 255
thresholdType: threshold type, usually has the following values,
THRESH_BINARY=0: grayscale values greater than the threshold are maxVal, and
those less than the threshold are 0.
THRESH_BINARY_INV=1: grayscale values greater than the threshold are 0, and those
less than the threshold are maxVal.
THRESH_TRUNC=2: grayscale values greater than the threshold are the threshold,
and those less than the threshold remain unchanged.
THRESH_TOZERO=3: grayscale values less than the threshold are 0, and those
greater than the threshold remain unchanged.
THRESH_TOZERO_INV=4: grayscale values greater than the threshold are 0, and those
less than the threshold remain unchanged.
Return value:
ret: consistent with parameter thresh
binary: result image
```

```
'''
```

The final image is shown in the figure,



3.3, Image edge detection

> Image edge detection can significantly reduce the data size of the image while retaining the original image properties. Among the commonly used edge detection methods, the Canny edge detection algorithm is one of the methods with strict definition and can provide good and reliable detection. Because it has the advantages of meeting the three criteria of edge detection and simple implementation process, 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:
>
> - Use Gaussian filter to smooth the image and filter out noise
> - Calculate the gradient strength and direction of each pixel in the image
> - Apply non-maximum suppression to eliminate the stray response caused by edge detection
> - Apply double-threshold detection to determine the real and potential edges
> - Complete edge detection by suppressing isolated weak edges

After switching to the directory where the code is located in the car terminal, run the program,

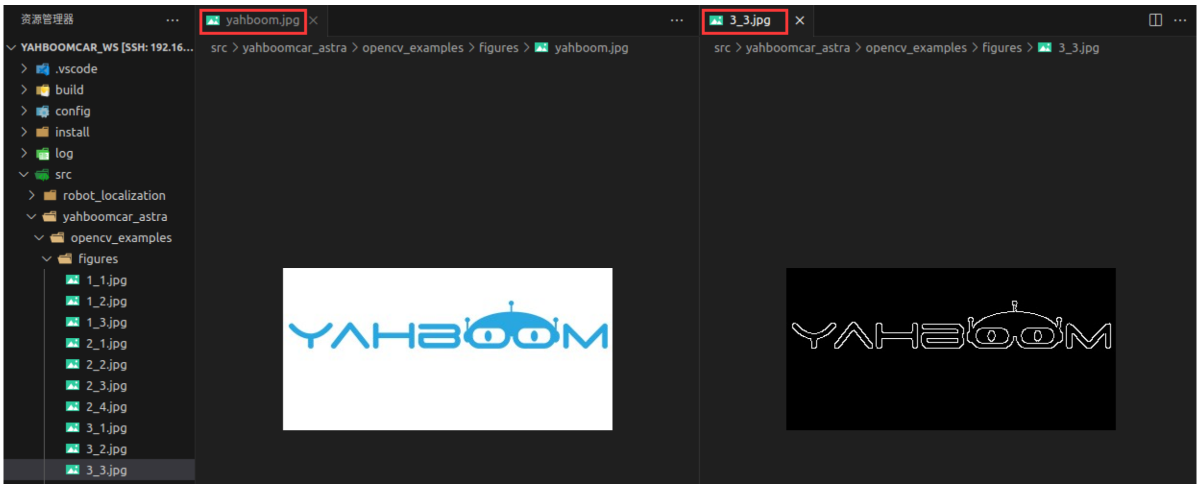
```
python3 3_3.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # Grayscale image
imgG = cv2.GaussianBlur(gray,(3,3),0)
dst = cv2.Canny(imgG,50,50)
cv2.imwrite('figures/3_3.jpg', dst)

'''
Gaussian filtering (noise reduction):
cv2.GaussianBlur(src,ksize,sigmaX[,dst[,sigmaY[,borderType]]]) -> dst
src: input image, usually grayscale image
ksize: Gaussian kernel size
sigmaX: Gaussian kernel standard deviation in the X direction
```

```
sigmaY: Gaussian kernel standard deviation in the Y direction
dst: processed image
Canny edge detection:
edges=cv2.Canny(image,threshold1,threshold2[,apertureSize[,L2gradient]])
edges: calculated edge image
image: calculated edge image, usually the image obtained after Gaussian
processing
threshold1: the first threshold in the processing process
threshold2: the second threshold in the processing process
apertureSize: the aperture size of the Sobel operator
L2gradient: the default value is False, using the L1 norm calculation; if True,
use the L2 norm
'''
```

The final image is as shown in the figure,



3.4, Line segment drawing

After switching to the directory where the code is located in the car terminal, run the program,

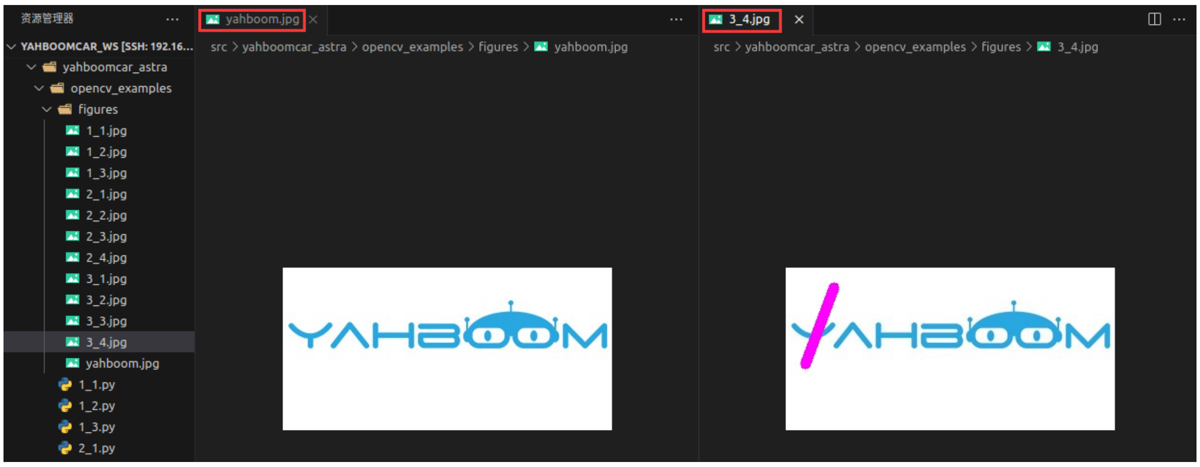
```
python3 3_4.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
line = cv2.line(img, (50,20), (20,100), (255,0,255), 10)
cv2.imwrite('figures/3_4.jpg', line)

'''
Line segment drawing: cv2.line
（img,pt1,pt2,color,thickness=None,lineType=None,shift=None）
img: canvas or carrier image
pt1,pt2: required parameters. Coordinate points of the line segment, representing
the starting point and the ending point respectively
color: required parameter. Used to set the color of the line segment
thickness: optional parameter. Used to set the width of the line segment
lineType: optional parameter. Used to set the type of line segment, the types are
as follows,
8:8 adjacent connecting line (default)
4:4 adjacent connecting line
cv2.LINE_AA: anti-aliasing
```

```
'''
```

The final image is as shown in the figure,



3.5, draw a rectangle

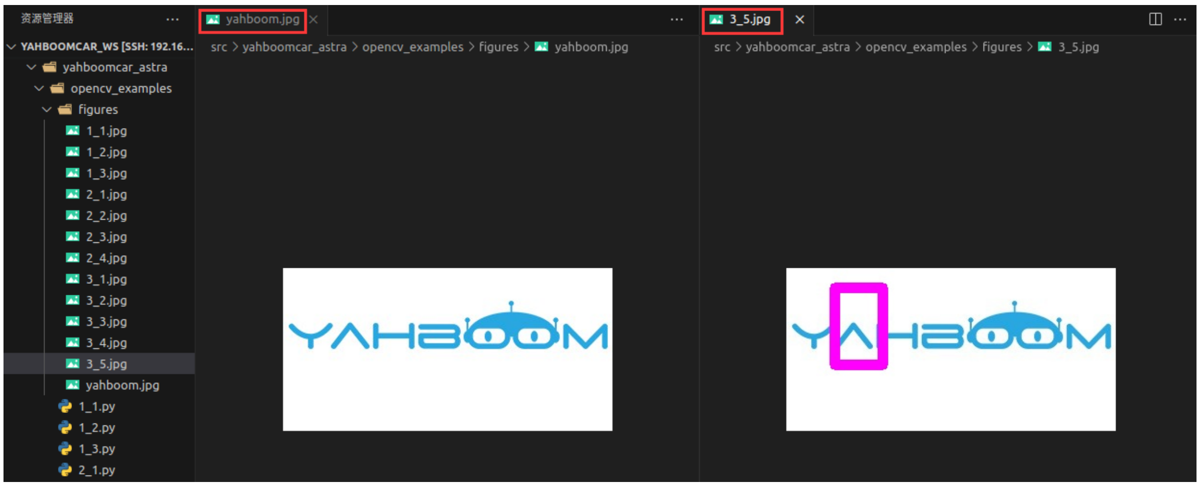
After switching to the directory where the code is located in the car terminal, run the program,

```
python3 3_5.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
rect = cv2.rectangle(img, (50,20), (100,100), (255,0,255), 10)
cv2.imwrite('figures/3_5.jpg', rect)

'''
Draw a rectangle: cv2.rectangle
（img,pt1,pt2,color,thickness=None,lineType=None,shift=None）
img: canvas or carrier image
pt1,pt2: required parameters. Respectively represent the upper left corner and
lower right corner of the rectangle, so as to determine the only rectangle
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 negative, it means filling the rectangle
lineType: optional parameter. Used to set the type of line segment, the types are
as follows,
8:8 adjacent connecting line (default)
4:4 adjacent connecting line
cv2.LINE_AA: anti-aliasing
cv2.FILLED: filling
'''
```

The final image is as shown in the figure,



3.6, draw a circle

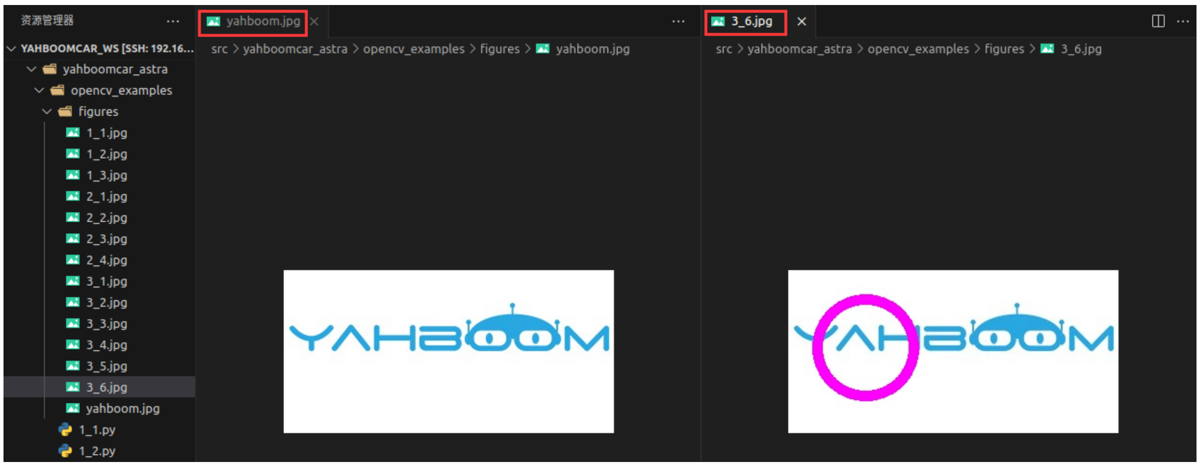
After switching to the directory where the code is located in the car terminal, run the program,

```
python3 3_6.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
circle = cv2.circle(img, (80,80), 50, (255,0,255), 10)
cv2.imwrite('figures/3_6.jpg', circle)

'''
Draw a circle: cv2.circle(img, center, radius, color[,thickness[,lineType]])
img: canvas or carrier image
center: required parameter. Center coordinates, format is (X coordinate value, Y
coordinate value)
radius: required parameter. Radius of the circle
thickness: optional parameter. Used to set the thickness of the circle outline.
When the value is negative, it means that the circle is filled
lineType: optional parameter. Used to set the type of line segment. The types are
as follows:
8:8 adjacent connecting lines (default)
4:4 adjacent connecting lines
cv2.LINE_AA: anti-aliasing
cv2.FILLED: filling
'''
```

The final image is as shown in the figure,



3.7, draw an ellipse

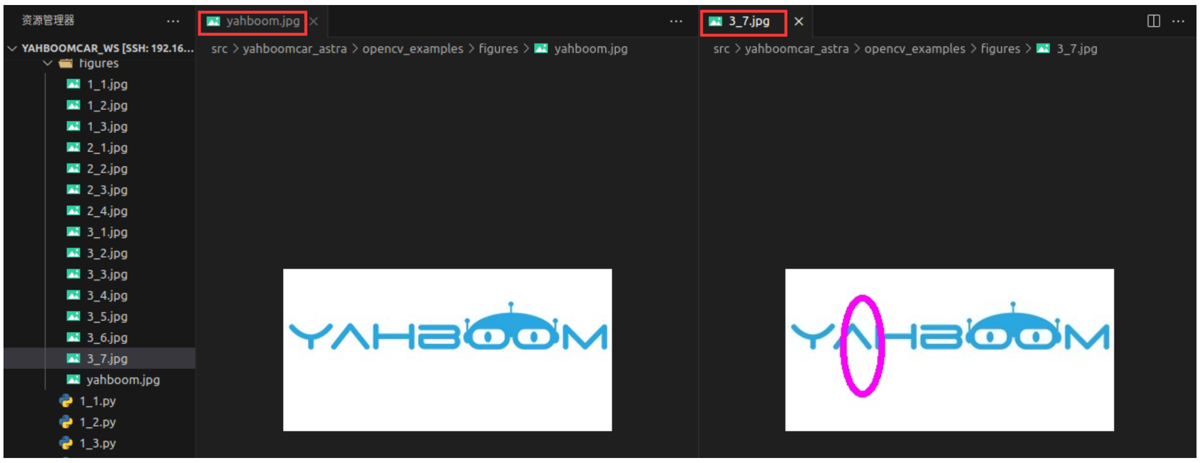
After switching to the directory where the code is located in the car terminal, run the program,

```
python3 3_7.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
ellipse = cv2.ellipse(img, (80,80), (20,50),0,0, 360,(255,0,255), 5)
cv2.imwrite('figures/3_7.jpg', ellipse)

'''
Draw an ellipse:
cv2.ellipse(img,center,axes,angle,StartAngle,endAngle,color[,thickness[,lineType]
)
center: the center point of the ellipse, (x, y)
axes: refers to the short radius and long radius, (x, y)
StartAngle: the angle of the arc's starting angle
endAngle: the angle of the arc's ending angle
img, color, thickness, lineType can refer to the description of the circle
'''
```

The final image is as shown in the figure,



3.8, draw polygons

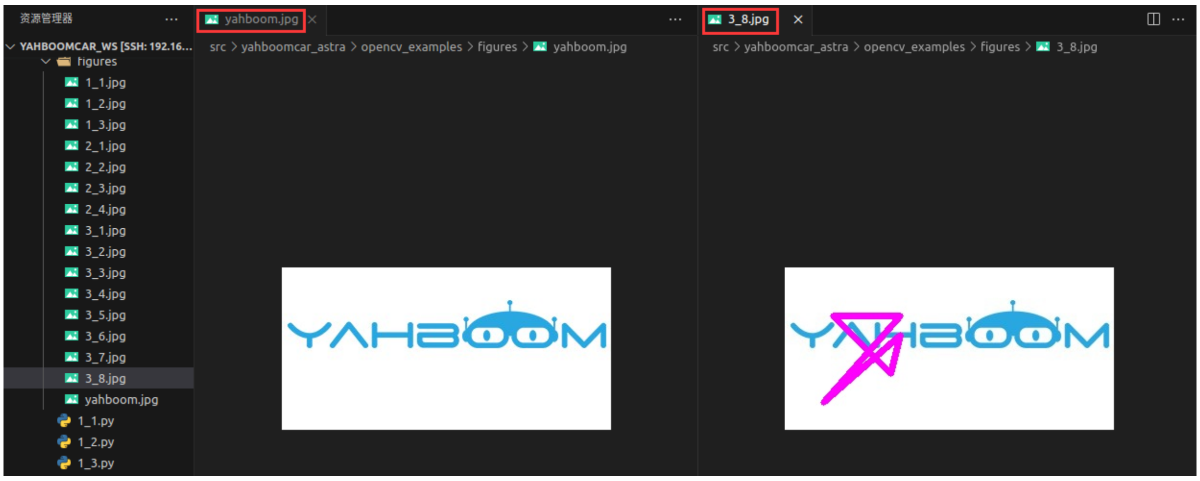
After switching to the directory where the code is located in the car terminal, run the program,

```
python3 3_8.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('figures/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)
cv2.imwrite('figures/3_8.jpg', polylines)

'''
Draw polygon: cv2.polylines(img,[pts],isClosed,color[,thickness[,lineType]])
pts: vertices of polygon
isClosed: whether closed (True/False)
Other parameters refer to the drawing parameters of the circle
'''
```

The final image is as shown in the figure,



3.9, Draw text

After switching the terminal to the directory where the code is located, run the program,

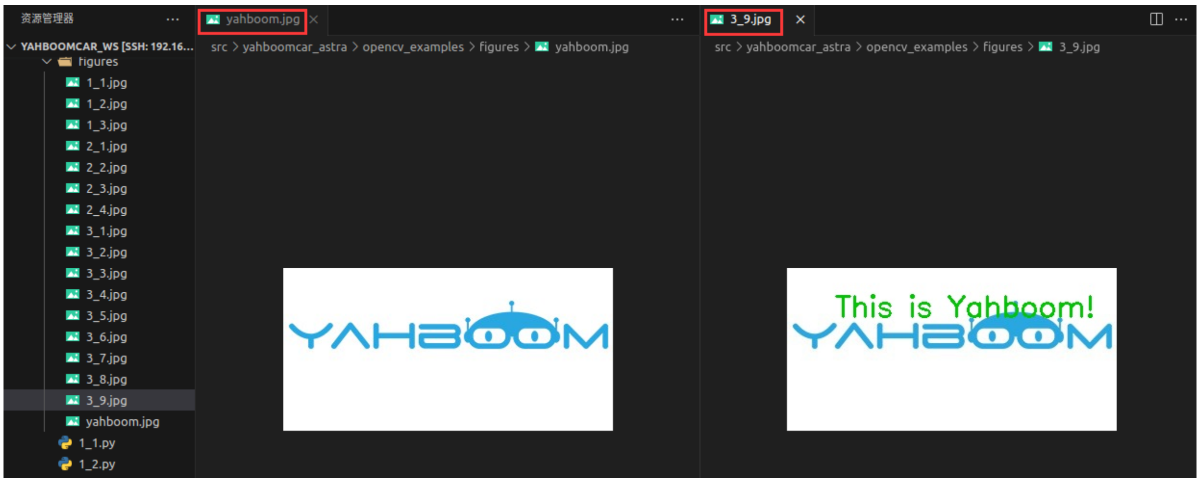
```
python3 3_9.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
cv2.putText(img,'This is Yahboom!',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,
(0,200,0),2)
cv2.imwrite('figures/3_9.jpg', img)

'''
Draw text: cv2.putText(img,str,origin,font,size,color,thickness)
img: input image
str: drawn text
origin: upper left corner coordinate (integer), which can be understood as where
the text starts
font: font
size: font size
```

```
color: font color
thickness: font thickness
'''
```

The final image is as shown below,


资源管理器
YAHBOOMCAR_WS [SSH: 192.16...
figures
1_1.jpg
1_2.jpg
1_3.jpg
2_1.jpg
2_2.jpg
2_3.jpg
2_4.jpg
3_1.jpg
3_2.jpg
3_3.jpg
3_4.jpg
3_5.jpg
3_6.jpg
3_7.jpg
3_8.jpg
3_9.jpg
yahboom.jpg
1_1.py
1_2.py
yahboom.jpg
src > yahboomcar_astra > opencv_examples > figures > yahboom.jpg
3_9.jpg
src > yahboomcar_astra > opencv_examples > figures > 3_9.jpg
YAHBOOM
This is Yahboom!
YAHBOOM