OpenCV Geometric Transformation

Before running the sample 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
```

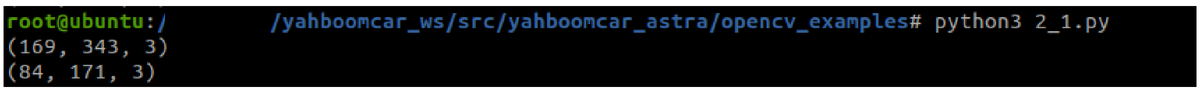
2.1, Image scaling

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

```
python3 2_1.py
```

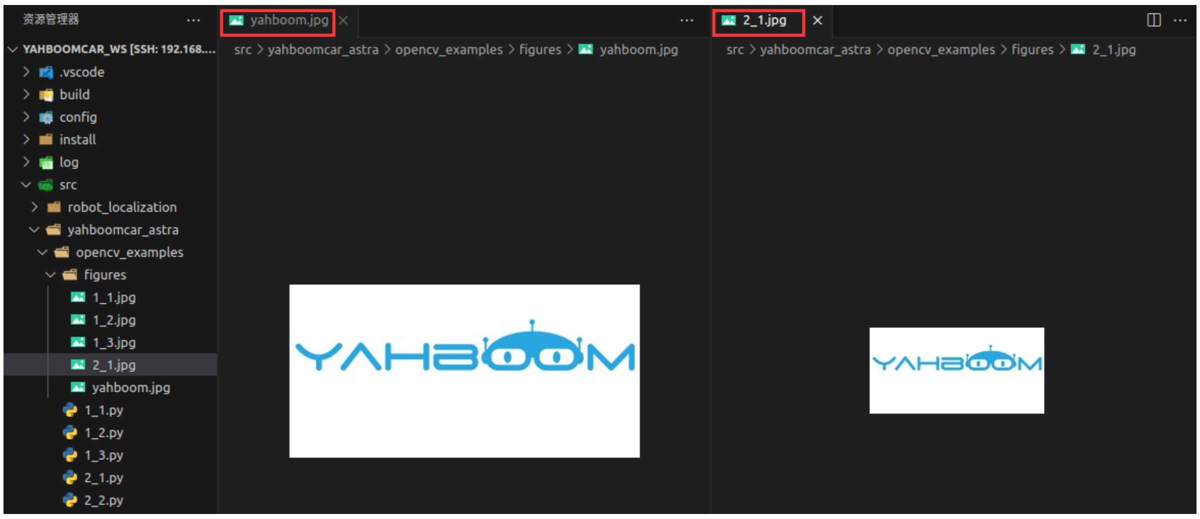
```python
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
print(img.shape)
x, y = img.shape[0:2]
img_resize = cv2.resize(img, (int(y / 2), int(x / 2)))
cv2.imwrite('figures/2_1.jpg', img_resize)
print(img_resize.shape)

'''
Image scaling: cv2.resize(InputArray src,OutputArray
dst,Size,fx,fy,interpolation)
InputArray src: input image
OutputArray ds: output image
Size: output image size
fx,fy: scaling factors along the x-axis and y-axis
interpolation: interpolation method, with the following options,
INTER_NEAREST (nearest neighbor interpolation)
INTER_LINEAR (bilinear interpolation, default setting)
INTER_AREA (resample using pixel area relationship)
INTER_CUBIC (bicubic interpolation of 4x4 pixel neighborhood)
INTER_LANCZOS4 (Lanczos interpolation of 8x8 pixel neighborhood)
'''
```

The terminal outputs the size of the original image and the reduced image (representing the width, height and number of channels of the image respectively),



```
root@ubuntu:/        /yahboomcar_ws/src/yahboomcar_astra/opencv_examples# python3 2_1.py
(169, 343, 3)
(84, 171, 3)
```

You can also see the difference in size when you open the image,



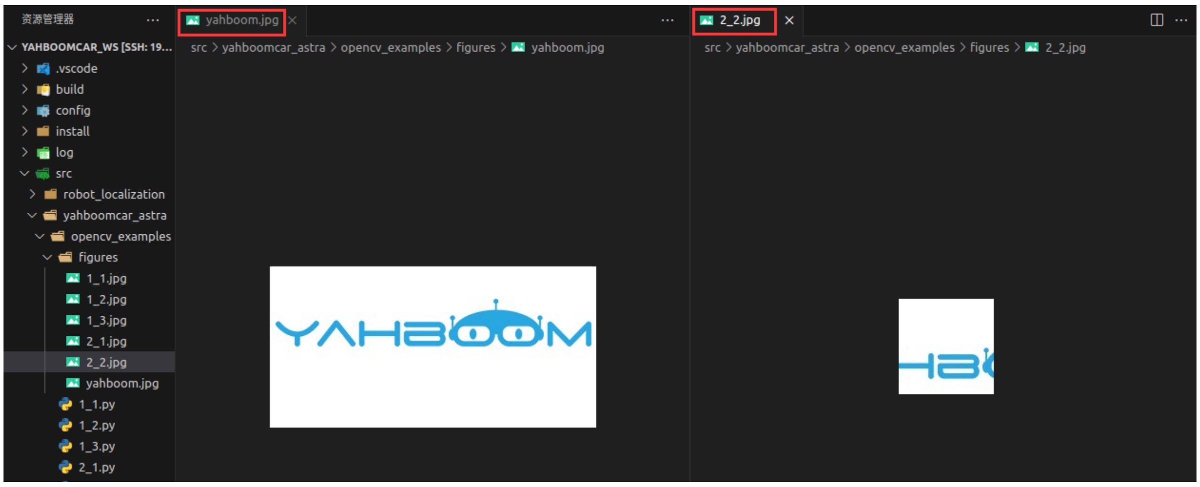
2.2, Image cropping

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

```
python3 2_2.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
dst = img[0:100,100:200] # Select area X: 0-100 Y: 100-200, cannot exceed the
original image resolution
cv2.imwrite('figures/2_2.jpg', dst)
```

The final image is as shown in the figure,



2.3, Image translation

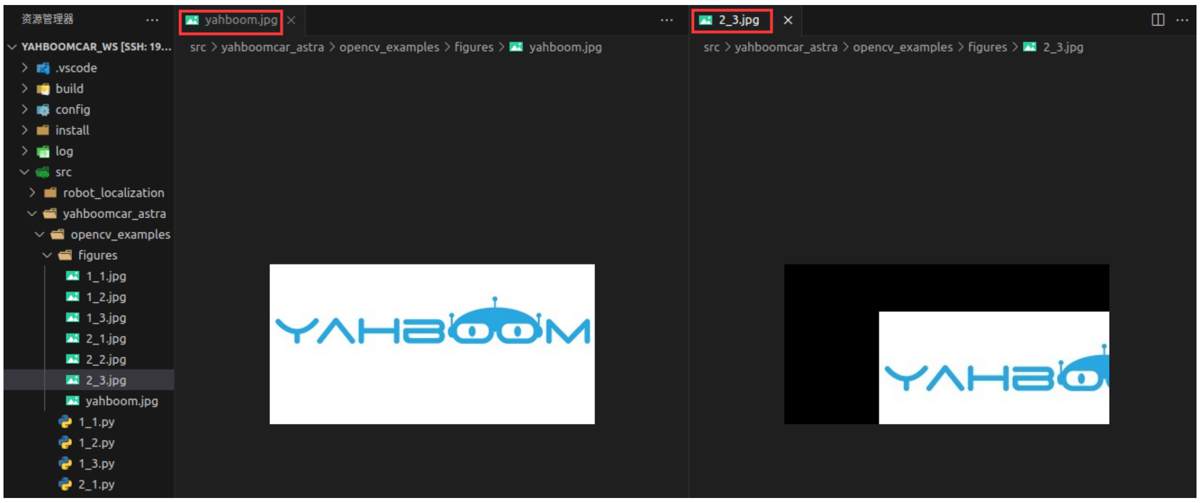
After switching to the directory where the code is located in the car terminal, run the program,

```
python3 2_3.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
```

```
imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
matShift = np.float32([[1,0,100],[0,1,50]]) # 2*3
dst = cv2.warpAffine(img, matShift, (width,height))
cv2.imwrite('figures/2_3.jpg', dst)

'''
Image translation:
cv2.warpAffine(src,M,dsize[,dst[,flags[,borderMode[,borderValue]]]])
src: input image
M: affine transformation matrix, generally reflects the relationship of
translation or rotation (InputArray type)
dsize: output image size
flags: combination of interpolation methods (int type)
borderMode: border pixel mode (int type)
borderValue: (important!) border filling value; by default, it is 0
Generally, only the first three parameters can be set to achieve basic affine
transformation effects.
'''
```

The final image is as shown in the figure,



> **How to get the transformation matrix M? The following is an example to illustrate,**
>
> The original image src is converted to the target image dst through the conversion matrix M:
>
> dst(x, y) = src(M11x + M12y+M13, M21x+M22y+M23)
>
> Move the original image src to the right by 100 pixels and move it down by 50 pixels, then the corresponding relationship is:
>
> dst(x, y) = src(x+100, y+50)
>
> Complete the above expression, that is:
>
> dst(x, y) = src(1·x + 0·y + 100, 0·x + 1·y + 50)
>
> According to the above expression, the values of each element in the corresponding conversion matrix M can be determined as:
>
> M11=1, M12=0, M13=100, M21=0, M22=1, M23=50.
>
> Substitute the above values into the transformation matrix M, and we get:

> M = [[1,0,100],[0,1,50]]

2.4, Image Mirroring

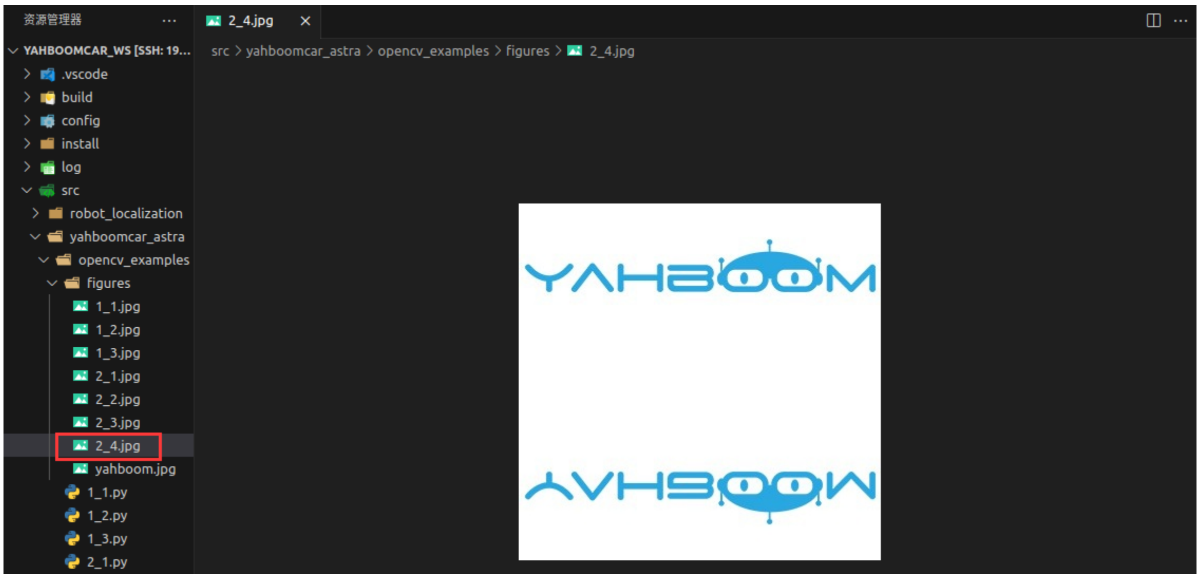
> There are two types of image mirroring: horizontal mirroring and vertical mirroring. Horizontal mirroring uses the vertical center line of the image as the axis to swap the pixels of the image, that is, swap the left and right halves of the image. Vertical mirroring uses the horizontal center line of the image as the axis to swap the upper and lower halves of the image.
>
> Transformation principle:
>
> Let the width of the image be width and the length be height. (x, y) are the transformed coordinates, (x0, y0) are the coordinates of the original image,
>
> - Horizontal mirror transformation
>
> Forward mapping: x=width-x0-1, y=y0
>
> Backward mapping: x0=width-x-1, y0=y
>
> - Vertical mirror transformation
>
> Upward mapping: x=x0, y=height-y0-1
>
> Downward mapping: x0=x, y0=height-y-1
>
> Summary: In horizontal mirror transformation, the entire image is traversed, and then each pixel is processed according to the mapping relationship. In fact, horizontal mirror transformation is to swap the image coordinate columns to the right and the right columns to the left, and the transformation can be done in columns. The same is true for vertical mirror transformation, which can be transformed in rows.

Take vertical transformation as an example. After switching the terminal to the directory where the code is located, run the program.

```
python3 2_4.py
```

```python
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('figures/yahboom.jpg')
imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
deep = imgInfo[2]
newImgInfo = (height*2,width,deep)
dst = np.zeros(newImgInfo,np.uint8)#uint8
for i in range(0,height):
for j in range(0,width):
dst[i,j] = img[i,j]
dst[height*2-i-1,j] = img[i,j]
cv2.imwrite('figures/2_4.jpg', dst)
```

The final image is as shown in the figure,



资源管理器
2_4.jpg
YAHBOOMCAR_WS [SSH: 19...
.vscode
build
config
install
log
src
robot_localization
yahboomcar_astra
opencv_examples
figures
1_1.jpg
1_2.jpg
1_3.jpg
2_1.jpg
2_2.jpg
2_3.jpg
2_4.jpg
yahboom.jpg
1_1.py
1_2.py
1_3.py
2_1.py
src > yahboomcar_astra > opencv_examples > figures > 2_4.jpg