Geometric transformation

2.1. OpenCV image scaling

2.1.1. In OpenCV, the function to achieve image scaling is: cv2.resize(InputArray src, OutputArray dst, Size, fx, fy, interpolation)

Parameter meaning:

InputArray src: input image

OutputArray ds: output picture

Size: output image size

fx, fy: scaling coefficients along the x-axis and y-axis

interpolation: Interpolation mode, you can choose INTER_NEAREST (nearest neighbor interpolation), INTER_LINEAR (bilinear interpolation (default setting)), INTER_AREA (resampling using pixel area relationships), INTER_CUBIC (bicubic interpolation of 4x4 pixel neighborhoods), INTER_LANCZOS4 (Lanczos interpolation of 8x8 pixel neighborhood)

requires attention:

1. The output size format is (width, height)
2. The default interpolation method is: bilinear interpolation

2.1.2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python 1_1.py
```

Raspberry Pi 5

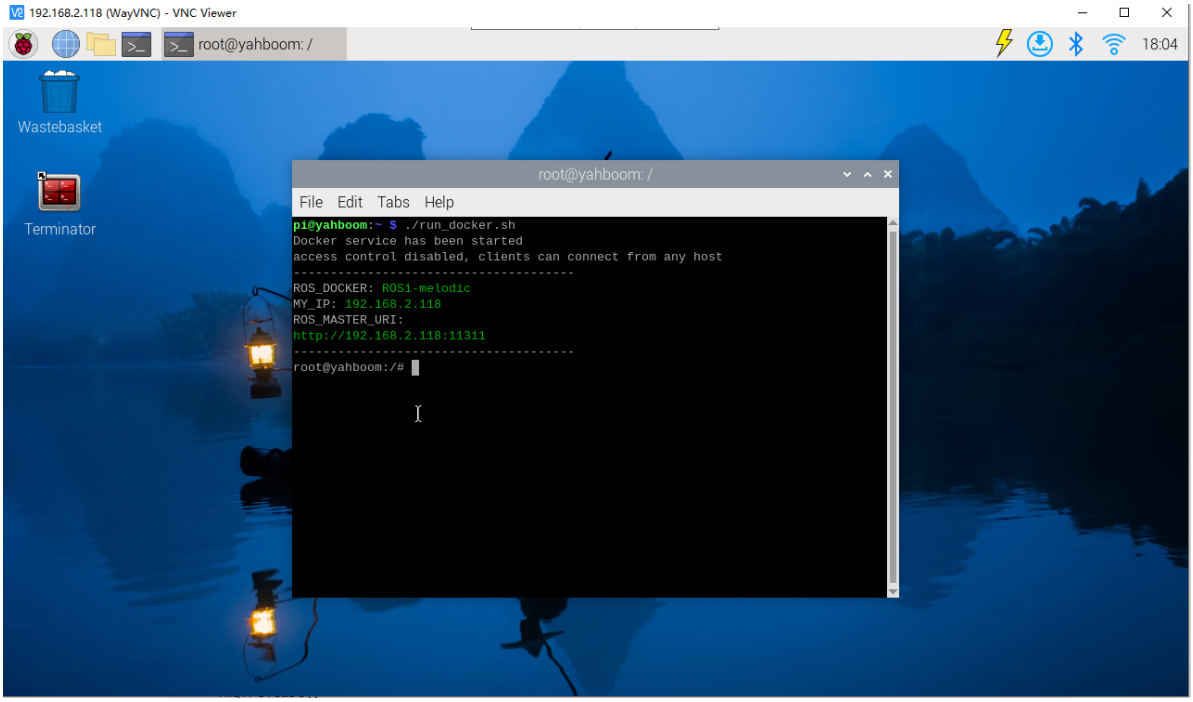
Before running, please confirm that the large program has been permanently closed

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

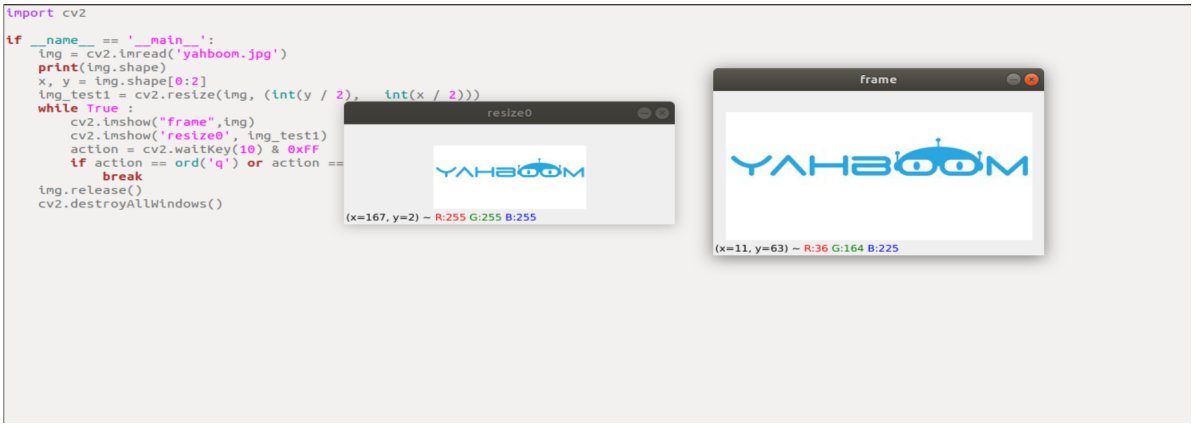
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_1.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
print(img.shape)
x, y = img.shape[0:2]
img_test1 = cv2.resize(img, (int(y / 2), int(x / 2)))
while True :
cv2.imshow("frame",img)
cv2.imshow('resize0', img_test1)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



2.2. OpenCV image cropping

2.2.1. Picture cutting

First read the image, and then get the pixel area in the array. In the following code, select the shape area X: 300-500 Y: 500-700. Note that the image size is 800*800, so the selected area should not exceed this resolution.

2.2.2. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_2.py
```

Raspberry Pi 5

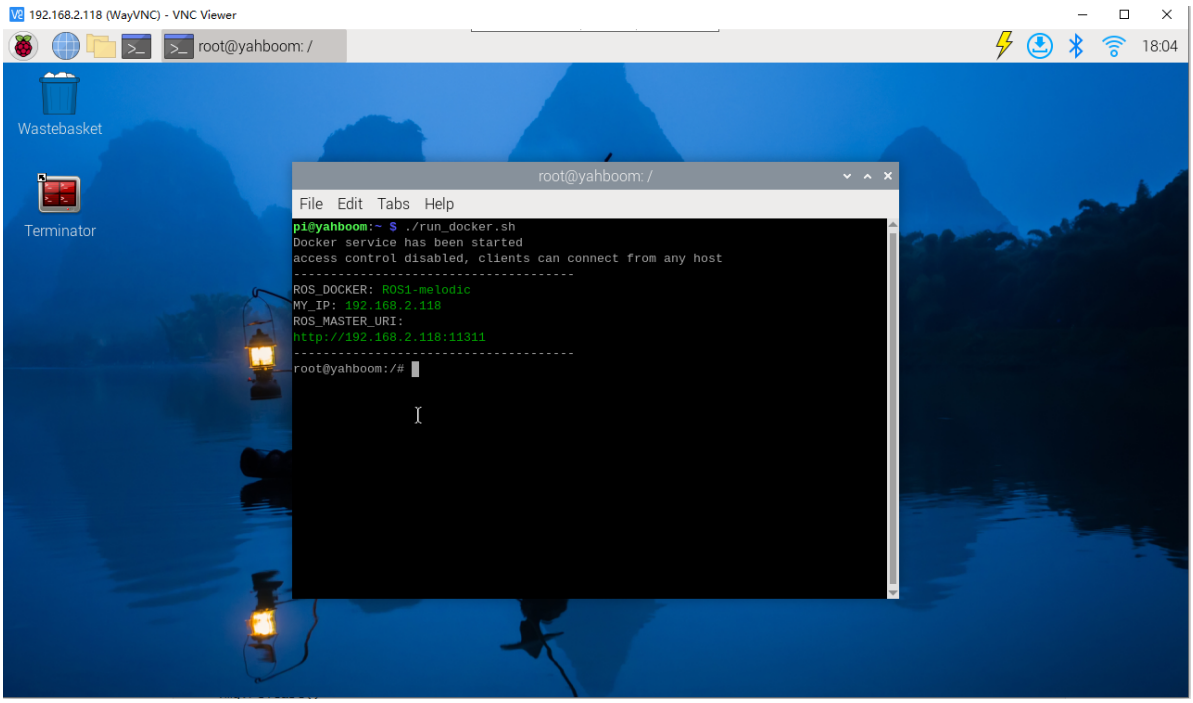
Before running, please confirm that the large program has been permanently closed

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

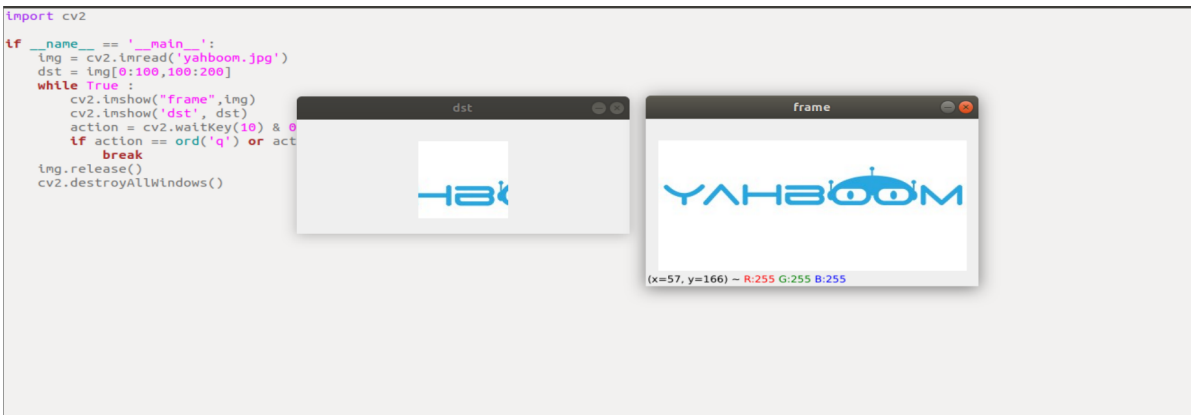
```
./run_docker.sh
```



Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_2.py
```

```
import cv2
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
dst = img[0:100,100:200]
while True :
cv2.imshow("frame",img)
cv2.imshow('dst', dst)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



2.3. OpenCV image translation

2.3.1. In OpenCV, image translation is achieved through affine transformation. The method used is cv2.warpAffine(src, M, dsize[,dst[, flags[, borderMode[, borderValue]]]])

Parameter meaning:

src - input image.
M - Transformation matrix.
dsize - the size of the output image.
flags - a combination of interpolation methods (of type int!)
borderMode - Border pixel mode (type int!)
borderValue - (emphasis!) border padding value; by default, it is 0.

Among the above parameters: M is an affine transformation matrix, which generally reflects the relationship of translation or rotation, and is a 2×3 transformation matrix of the InputArray type. In daily affine transformation, basic affine transformation effects can be achieved by setting only the first three parameters, such as cv2.warpAffine(img,M,(rows,cols)).

2.3.2. How to get the transformation matrix M? Here is an example to illustrate:

The original image src is converted into the target image dst through the transformation matrix M:

$$dst(x, y) = src(M11x + M12y+M13, M21x+M22y+M23)$$

Move the original image src to the right by 200 and down by 100 pixels, then the corresponding relationship is:

$$dst(x, y) = src(x+200, y+100)$$

Complete the above expression, that is:

$$dst(x, y) = src(1·x + 0·y + 200, 0·x + 1·y + 100)$$

According to the above expression, it can be determined that the value of each element in the corresponding transformation matrix M is:

M11=1

M12=0

M13=200

M21=0

M22=1

M23=100

Substituting the above values into the transformation matrix M, we get:

M = [ ]

2.3.3. Code and actual effect display

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_3.py
```

Raspberry Pi 5

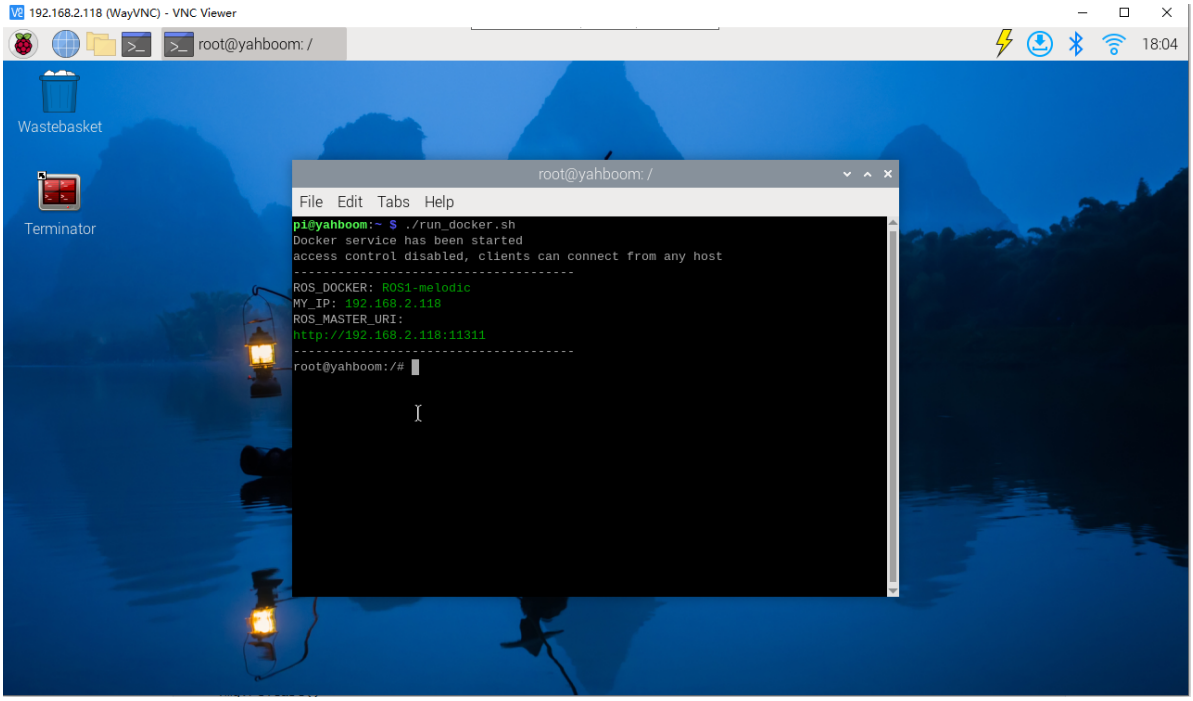
Before running, please confirm that the large program has been permanently closed

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
```

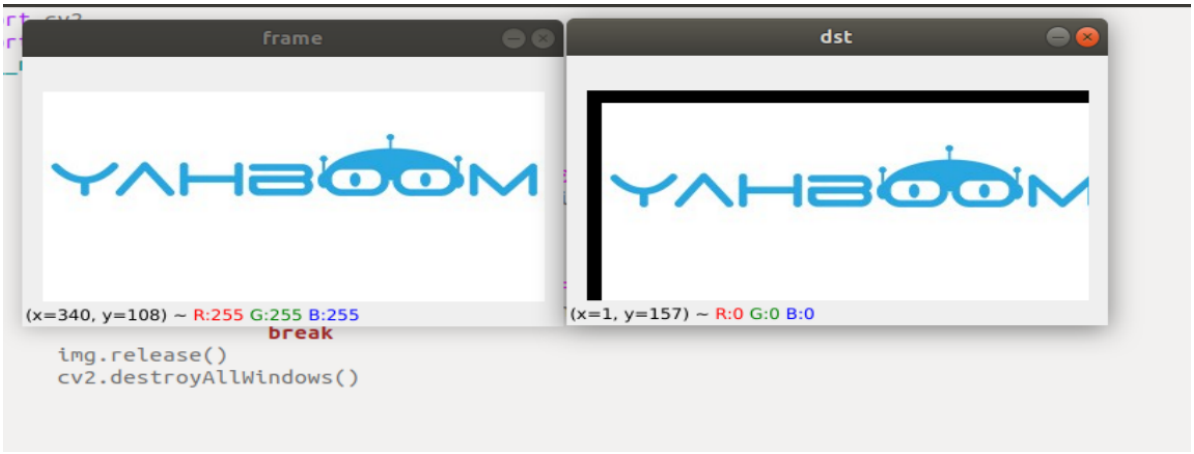


Run program

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_3.py
```

```
import cv2
import numpy as np
if __name__ == '__main__':
img = cv2.imread('yahboom.jpg')
imgInfo = img.shape
height = imgInfo[0]
width = imgInfo[1]
matShift = np.float32([[1,0,10],[0,1,10]])# 2*3
dst = cv2.warpAffine(img, matShift, (width,height))
```

```
while True :
cv2.imshow("frame",img)
cv2.imshow('dst', dst)
action = cv2.waitKey(10) & 0xFF
if action == ord('q') or action == 113:
break
img.release()
cv2.destroyAllWindows()
```



2.4. OpenCV image mirroring

2.4.1. Principle of image mirroring

There are two types of image mirroring transformation: horizontal mirroring and vertical mirroring. Horizontal mirroring uses the vertical centerline of the image as the axis to swap the pixels of the image, that is, swapping the left and right halves of the image. Vertical mirroring takes the horizontal centerline of the image as the axis and swaps the upper and lower parts 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

Up mapping: x=x0,y=height-y0-1

Down mapping: x0=x, y0=height-y-1

Summarize:

During horizontal mirror transformation, the entire image is traversed, and then each pixel is processed according to the mapping relationship. In fact, the horizontal mirror transformation is to change the image coordinate column to the right, and the right column to the left. The transformation can be done in column units. The same is true for vertical mirror transformation, which can be transformed in row units.

2.4.2. Take vertical transformation as an example to see how Python is written.

Run program

jetson motherboard/Raspberry Pi 4B

```
cd ~/transbot_ws/src/transbot_visual/opencv
python2_4.py
```

Raspberry Pi 5

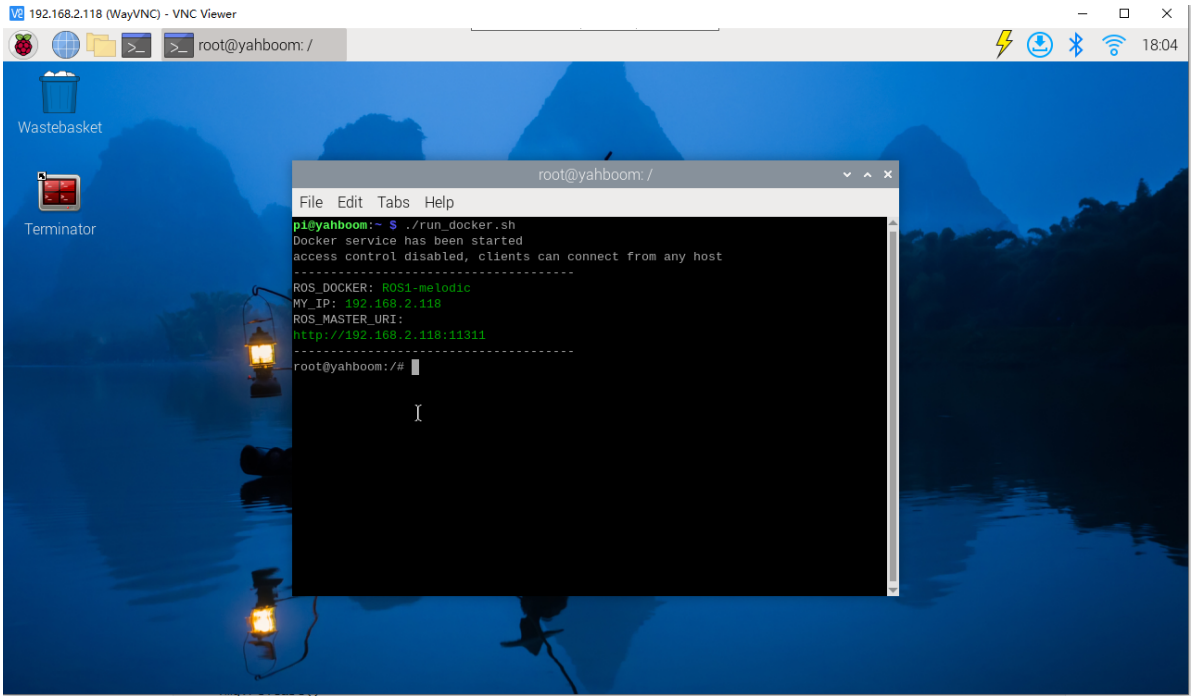
Before running, please confirm that the large program has been permanently closed

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
python2_4.py
```

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