

# Lesson 8 Image Processing --- Geometric Transformation

## 1.Introduction

A spatial transformation of an image is a geometric transformation of the image coordinate system. It map the coordinate of a picture to a new coordinate of other picture. And geometric transformation will not change the pixel of the image, but rearrange the pixels on the image plane.

According to OpenCV functions, we divide mapping into scaling, flipping,affine transformation, perspective, etc.

## 2.Scaling

Scaling is to adjust the size of the picture, for example zoom in or zoom out the picture. In OpenCV, cv2.resize() function is used to scale the image.

**`dst = cv2.resize(src,  dsize[,  fx[,  fy[,  interpolation] ] ] )`**

dst represents the output image whose type is the same as src. And its size is dsize (when it is not 0) or can be calculated through src.size(), fx and fy.

- 1) src represents the original picture
- 2) dsize stands for the size of the output image
- 3) fx indicates the horizontal scaling ratio
- 4) fy denotes the vertical scaling ratio
- 5) interpolation is for interpolation method.

Type	Description
cv2.INTER_NEAREST	nearest neighbor interpolation
cv2.INTER_LINEAR	linear interpolation
cv2.INTER_CUBIC	Cubic spline interpolation. First, the cubic spline fitting is performed on the 4 x 4 nearest neighbors near the original image, and then the cubic spline value corresponding to the target pixel is taken as the value of the corresponding pixel in the target image.
cv2.INTER_AREA	Area interpolation similar to nearest interpolation. Sample the current pixel according to the pixels in the surrounding area of the current pixel.
cv2.INTER_LANCZOS4	Lanczos interpolation over $8 \times 8$ neighborhood
cv2.INTER_LINEAR_EXACT	Bit accurate bilinear interpolation
cv2.INTER_MAX	Difference encoding mask
cv2.WARP_FILL_OUTLIERS	Flag, fills all of the destination image pixels. If some of them correspond to outliers in the source image, they are

	set to zero
cv2.WARP_INVERSE_MAP	<p>flag, inverse transformation. For example, polar transformation. If flag is not set, perform transformation:</p> $\text{dst}(\rho, \phi) = \text{src}(x, y)$ <p>For example, If flag is set, perform transformation: <math>\text{dst}(x, y) = \text{src}(\rho, \phi)</math></p>

## 2.1 Operation Steps

**The program will scale the image.**



Before operation, please copy the routine **"Scale"** in "4.OpenCV->Image Processing --- Geometric Transformation->Routine Code" to the shared folder.

For how to configure the shared folder, please refer to the file in **"2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement"**.

---

**Note: the input command should be case sensitive and the keywords can be complemented by "Tab" key.**

---

1) Open virtual machine and start the system. Click , and then  or press **"Ctrl+Alt+T"** to open command line terminal.

2) Input command **"cd /mnt/hgfs/Share/Scale"** and press Enter to enter the shared folder.

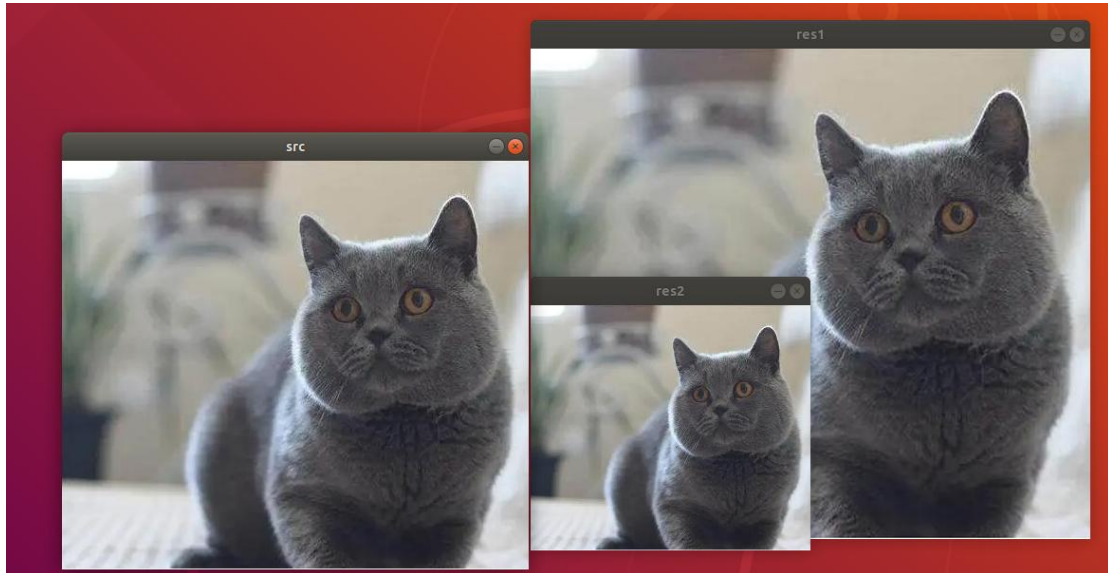
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Scale/
hiwonder@ubuntu:/mnt/hgfs/Share/Scale$
```

3) Input command **"python3 Scale.py"** and press Enter to run the code.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Scale$ python3 Scale.py
```

## 2.2 Program Outcome

The final output picture is as follow.



- 1) **src**: Original picture. Its size is 492\*430 pixels (width\*height)
- 2) **res1**: The size of the picture after zoomed in. Its size is 590\*512 pixels (width\*height)
- 3) **res2**: The size of the picture after zoomed out. And its size is 295\*258 pixels(width\*height)

## 2.3 Program Analysis

The routine “**Scale.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code->Scale.py**”.

```

1  import numpy as np
2  import cv2 as cv
3
4  src = cv.imread('1.jpg')
5  # method output the dimension directly
6  height, width = src.shape[:2] # acquire the original dimension
7  res1 = cv.resize(src, (int(1.2*width), int(1.2*height)), interpolation=cv.INTER_CUBIC)
8  res2 = cv.resize(src, (int(0.6*width), int(0.6*height)), interpolation=cv.INTER_CUBIC)
9  cv.imshow("src", src)
10 cv.imshow("res1", res1)
11 cv.imshow("res2", res2)
12 print("src.shape=", src.shape)
13 print("res1.shape=", res1.shape)
14 print("res2.shape=", res2.shape)
15 cv.waitKey()
16 cv.destroyAllWindows()
17

```

Firstly, import the required module through import statement.

```

1  import numpy as np
2  import cv2 as cv

```

Then call **imread()** function in cv2 module to read the image that needs to be scaled.

```

4  src = cv.imread('1.jpg')

```

In the bracket is the name of image.

The original width of the picture is 492 pixel, and height is 430 pixel. Parameter dsize can be used to designate the size of target image res1 and res2 (The name of the image can be customized)

The first parameter in dsize corresponds to the width after scaling. (width i.e. the number of columns which is related to parameter fx.) And the second parameter corresponds to the height after scaling. (height i.e. the number of row which is related to parameter fy)

If the value of dsize is specified, the size of the target image is determined by dsize regardless of whether the parameters fx and fy are specified.

```

6  height, width = src.shape[:2] # acquire the original dimension
7  res1 = cv.resize(src, (int(1.2*width), int(1.2*height)), interpolation=cv.INTER_CUBIC)
8  res2 = cv.resize(src, (int(0.6*width), int(0.6*height)), interpolation=cv.INTER_CUBIC)

```

Therefore, program to acquire the original dimension first, and then directly scale the width and height. To zoom in this picture, this routine enlarges the width of res1 to 1.2 times the original and height to 1.2 times the original. Through calculation, the width is 590 pixels (492x1.2) and the height is 516 pixels (430x1.2).

To zoom out the picture, this routine will shrink the res2 width to 0.6 times the original, and the height to 0.6 times the original. The final width is 295 pixels (492x0.6) and height is 258 pixels (430x0.6). And the image size, before and after processing, can printed.

```
src.shape= (430, 492, 3)
res1.shape= (516, 590, 3)
res2.shape= (258, 295, 3)
```

### 3.Affine Transformation

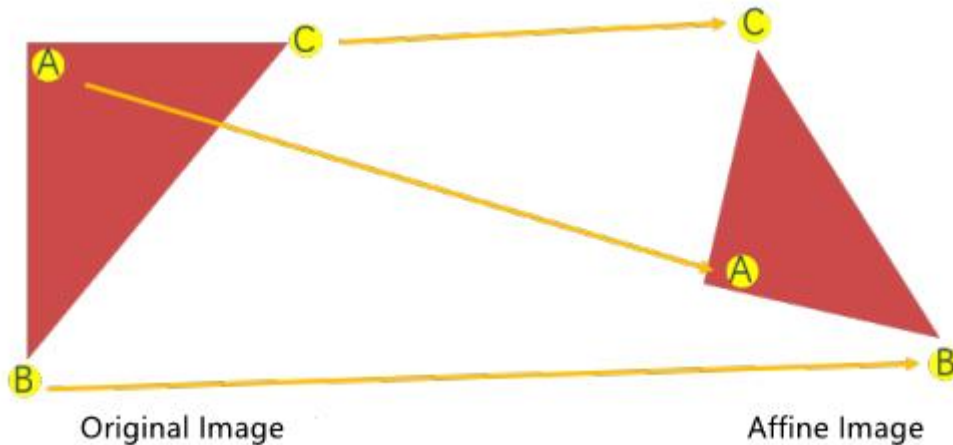
Affine transformation is that images can be translated, rotated, etc. through a series of geometric transformations, while lines and parallelism can be preserved.

Linearity means that the straight lines of the image can still be preserved after affine transformation. And parallelism indicates that parallel lines can be preserved after affine transformation.

Translation and rotation are special cases of affine transformation which is realized by the function **cv2.warpAffine()** in OpenCV. This function execute transformation by a transformation matrix M (transformation matrix of translation and rotation is different)

$$\text{dst}(x, y) = \text{src}(M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23})$$

As the picture below shown, the original image O can be transformed into affine image R by a transformation matrix M.



The format of **cv2.warpAffine()** function is as follow.

```
dst = cv2.warpAffine( src, M, dsize[, flags[, borderMode[, borderValue]]] )
```

- 1) **dst**: Represent the output image after affine transformation. The type of this image is similar to that of the original image. And the actual size of the output image is finally determined by dsize.
- 2) **src**: Represent the original image
- 3) **M**: Stand for a 2x3 transformation matrix. Various affine transformation can be realized by using different transformation matrix. And the size of the output image is finally determined by dsize.
- 4) **flags**: Represents the interpolation method which defaults to INTER\_LINEAR. When it is **WARP\_INVERSE\_MAP**, M is an inverse transformation from the target image dst to the original image src.  
**borderMode**, optional parameter, represents the edge type, BORDER\_CONSTANT by default. When it is **BORDER\_TRANSPARENT**, the values in the target image do not change, and these values correspond to the outliers in the original



image.

- 5) **borderValue**: Refer to border value, 0 by default.

The optional parameters of cv2.warpAffine() function can be omitted, and its final format is as follow.

```
dst = cv2.warpAffine( src , M , dsize )
```

By transformation matrix M, transform the original image src into the target image dst.

$$\text{dst}(x, y) = \text{src}(M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23})$$

Therefore, the type of affine transformation relies on the transformation matrix M.

### 3.1 Translation

Translation is the movement of the object. If the coordinates of the object translation is obtained, the following transformation matrix can be created.

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

Put the transformation matrix into the array whose type is np.float32, and assign M matrix to cv2.warpAffine() function so as to realize translation.

#### 3.1.1 Operation Steps

**This routine will translate the image to right.**



Before operation, please copy the routine code in “4.OpenCV->Image Processing --- Geometric Transformation->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “2. Linux



## Basic Lesson->Lesson 3 Linux Installation and Source Replacement

**Note:** the input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Translation/**” and press Enter to enter the shared folder.

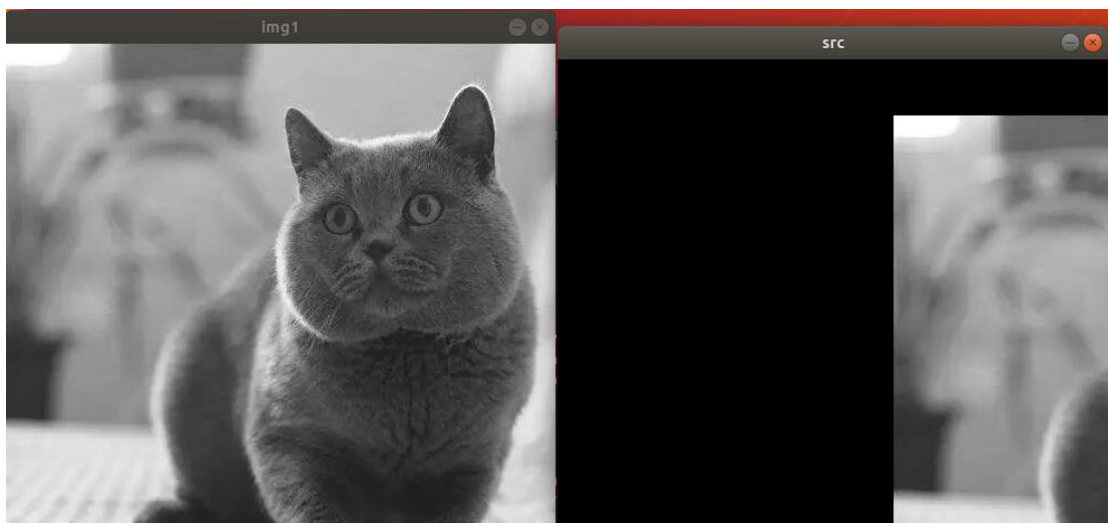
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Translation/  
hiwonder@ubuntu:/mnt/hgfs/Share/Translation$
```

3) Input command “**python3 Translation.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Translation$ python3 Translation.py
```

### 3.1.2 Program Outcome

The final output picture is as follow.



### 3.1.3 Program Analysis

The routine “**Translation.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code->TranslaTion**”.

```
import numpy as np
import cv2

img = cv2.imread('1.jpg')
rows, cols, ch = img.shape
M = np.float32([[1, 0, 300], [0, 1, 50]])

dst = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('img1', img)
cv2.imshow('src', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1) Firstly, import the required module through import statement.

```
1 import cv2
2 import numpy as np
```

2) Then call **imread()** function in cv2 module to read the image that needs to be translated.

```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
5 rows, cols, ch = img.shape
```

4) As mentioned before, if the coordinate of the object translation can be obtained, the transformation matrix can be created.

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

```
6 M = np.float32([[1, 0, 300], [0, 1, 50]])
```

After setting, the picture before and after translation can be displayed through imshow function.

```
10 cv2.imshow('img1', img)
11 cv2.imshow('src', dst)
```

5) Lastly, close the window through the function, and you can press any key to exit the program.

```
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

## 3.2 Rotation

Both translation and rotation are the examples of the affine transformation, and employ cv2.warpAffine function to realize affine transformation. But their transformation matrix is different. When rotating the image with function cv2.warpAffine(), obtain the transformation matrix with function cv2.getRotationMatrix2D().

The function format is

```
retval=cv2.getRotationMatrix2D(center, angle, scale)
```

- 1) center refers to the center of rotation.
- 2) angle stands for rotation angle. When it is positive, the image will be rotated counterclockwise. When it is negative, the image will be rotated clockwise.
- 3) scale means scaled size

The rotation angle  $\theta$  can be obtained from matrix M.

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

OpenCV has improved this matrix to provide scaling rotation and adjustable rotation center, as the picture shown below.

$$\begin{bmatrix} \alpha & \beta & (1 - \alpha) \cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1 - \alpha) \cdot center.y \end{bmatrix}$$

$$\begin{aligned} \alpha &= scale \cdot \cos \theta, \\ \beta &= scale \cdot \sin \theta \end{aligned}$$

The above matrix represents a rotation around center.x and center.y by  $\theta$  degrees.

For example, set the function as below to rotate the image around the image center counterclockwise by 45 degree, and zoom out the image 0.6 times the original. Then call this function to generate the matrix.

```
M=cv2.getRotationMatrix2D((height/2,width/2),45,0.6)
```

### 3.2.1 Operation Steps

**This routine will rotate the image 90 degree counterclockwise.**



Before operation, please copy the routine “**Revolve**” in “**4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code**” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

---

**Note:** the input command should be case sensitive and the keywords can be complemented by “**Tab**” key.

---

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Revolve/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Revolve/  
hiwonder@ubuntu:/mnt/hgfs/Share/Revolve$
```

3) Input the command “**python3 Revolve.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Revolve$ python3 Revolve.py
```

### 3.2.2 Program Outcome

The output picture is as follow.



### 3.2.3 Program Analysis

The routine “**Revolve.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code->Revolve**”.

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread('1.jpg')
5  rows, cols, ch = img.shape
6  # rotate the center rotation angle scale factor
7  M = cv2.getRotationMatrix2D(((cols-1) / 2.0, (rows-1)/2.0), 90, 1)
8  # original picture convert matrix output the image center
9  dst = cv2.warpAffine(img, M, (cols, rows))
10
11 cv2.imshow('img', img)
12 cv2.imshow('dst', dst)
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
15
```

- 1) Firstly, import the required module through import statement.

```
1  import cv2
2  import numpy as np
```

- 2) Then call **imread()** function in cv2 module to read the image that needs to be rotated.



```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
5 rows, cols, ch = img.shape
```

4) The image will rotate around the image center 90 degree counterclockwise. And its size remains the same.

```
7 M = cv2.getRotationMatrix2D(((cols-1) / 2.0,(rows-1)/2.0), 90,1)
```

5) Output the original image center

```
9 dst = cv2.warpAffine(img, M, (cols, rows))
```

6) After setting, we can call imshow function to display the pictures before and after rotation.

```
12 cv2.imshow('img', img)
13 cv2.imshow('dst', dst)
```

7) Lastly, call function below to close the window, and you can press any key to exit the program.

```
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait **n** ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the



specific value of the window, the designated window will be removed.

## 4.Perspective Transformation

Affine transformation are rotation, translation and scaling in 2D space, while perspective transformation is in 3D space.

Perspective transformation is realized by function `cv2.warpPerspective()`, and the function format is as follow.

```
dst = cv2.warpPerspective( src, M, dsize[, flags[, borderMode[,  
borderValue]]] )
```

- 1) dst represents the output image after perspective transformation, whose type is the same as the original picture. And its size is determined by dsize.
- 2) src represents the image to be processed.
- 3) M stands for a 3x3 transformation matrix
- 4) dsize indicates the dimension of the output image.
- 5) flags represents the interpolation method which defaults to `INTER_LINEAR`. When it is **WARP\_INVERSE\_MAP**, M is an inverse transformation from the target image dst to the original image src.  
**borderMode**, optional parameter, represents the edge type, `BORDER_CONSTANT` by default. When it is **BORDER\_TRANSPARENT**, the values in the target image do not change, and these values correspond to the outliers in the original image.

### 4.1 Operation Steps

This routine will perform perspective transformation.



Before operation, please copy the routine “**Perspective**” in  
“4.OpenCV->Lesson 8 Image Processing --- Geometric  
Transformation->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux  
Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

---

**Note: the input command should be case sensitive and the keywords  
can be complemented by “Tab” key.**

---

1) Open virtual machine and start the system. Click “”, and then “”  
or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Perspective/**” and press Enter  
to enter the shared folder.

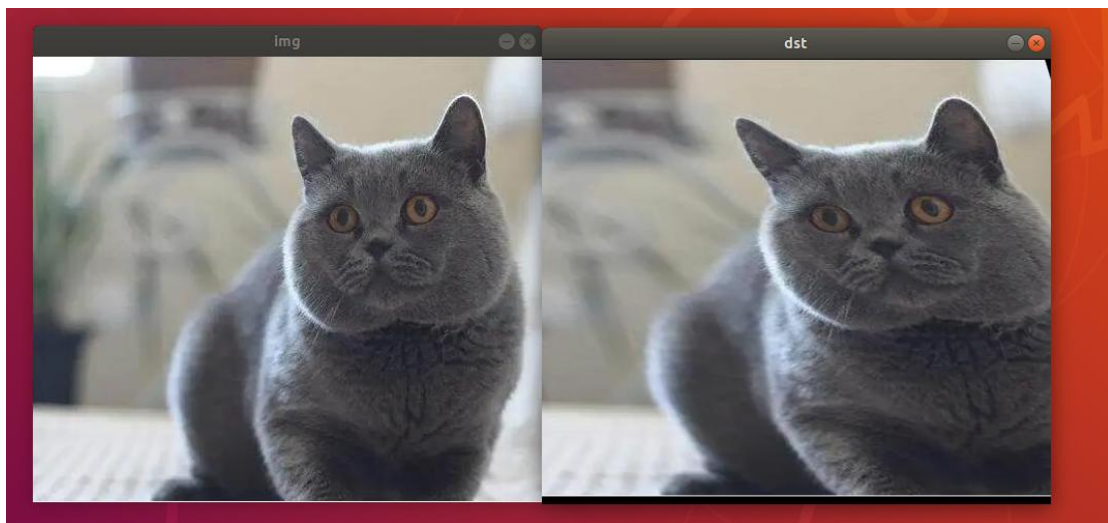
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Perspective/  
hiwonder@ubuntu:/mnt/hgfs/Share/Perspective$
```

3) Input command “**python3 Perspective.py**” and press Enter to run the  
routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Perspective$ python3 Perspective.py
```

## 4.2 Program Outcome

The final output picture is as follow.



### 4.3 Program Analysis

The routine “**Perspective.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code->Perspective**”.

```
import cv2
import numpy as np

img=cv2.imread('1.jpg')
rows, cols = img.shape[:2]
print(rows,cols)

pts1 = np.float32([[150,50],[400,50],[60,450],[310,450]])
pts2 = np.float32([[50,50],[rows-50,50],[50,cols-50],[rows-50,cols-50]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img,M,(cols,rows))

cv2.imshow("img",img)
cv2.imshow("dst",dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

1) Firstly, import the required module through import statement.

```
1 import cv2
2 import numpy as np
```

2) Then call **imread()** function in cv2 module to read the image for perspective transformation.

```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
rows, cols = img.shape[:2]
```

4) In this example, specify four vertices pts1 of the parallelogram in the original image, and specify four vertices pts2 of the rectangle in the target image. Next, generate the transformation matrix M with **dst=cv2.warpPerspective(img,M,(cols,rows))**. Next, employ **dst=cv2.warpPerspective(img,M,(cols,rows))** statement to convert parallelogram to rectangle.

```
8 pts1 = np.float32([[150,50],[400,50],[60,450],[310,450]])
9 pts2 = np.float32([[50,50],[rows-50,50],[50,cols-50],[rows-50,cols-50]])
10 M = cv2.getPerspectiveTransform(pts1,pts2)
11 dst = cv2.warpPerspective(img,M,(cols,rows))
```

5) After setting, the picture before and after translation can be displayed through imshow function.

```
13 cv2.imshow("img",img)
14 cv2.imshow("dst",dst)
```

6) Lastly, close the window through the function, and you can press any key to exit the program.

```
15 cv2.waitKey()
16 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the

specific value of the window, the designated window will be removed.

## 5.Remap

Remap is that the pixels are mapped from one picture to the corresponding positions in another image according to the rules to form a new image.

As the pixel coordinates of the original image do not correspond to that of target image, in general, we describe the position (x, y) of each pixel by remapping.

$$g(x,y)=f(h(x,y))$$

g() refers to target image, f() is the original image and h(x,y) is the image after remapping. Take the below function for example. And **image I** will be remapped under the following conditions.

$$h(x,y)=(I.cols-x,y)$$

The image will flip in the x direction. cv2.remap() function in OpenCV makes it more convenient and free to remap. And its format is as follow.

```
dst = cv2.remap( src, map1, map2, interpolation[, borderMode[,  
borderValue]] )
```

式中:

- 1) dst represents the output image whose type and size are the same as the original picture.
- 2) src represents the original image
- 3) There are two possible values of map1 parameter. It represents a map of (x,y), or x value of (x,y) of CV\_16SC2 , CV\_32FC1, CV\_32FC2 type.

- 4) There are also two possible values of map2 parameter.

When map1 represents (x,y), its value is none.

When map1 represents x value of (x,y), its value is the y value of (x,y) in CV\_16UC1, CV\_32FC1 type.

Note: map1 refers to the column where the pixel is located, and map2 refers to the row where the pixel is located. So usually, map1 is written as mapx and map2 as mapy for better understanding.

- 5) Interpolation is for interpolation method.
- 6) borderMode refers to border value. When it is BORDER\_TRANSPARENT, the pixel of target image corresponding to outliers in the original image will not be modified.
- 7) borderValue refers to border value, 0 by default.

## 5.1 Copy Pixel

### 5.1.1 Operation Steps

All pixels in the target image are mapped to the pixels on the 100th row and 200th column in the original image.



Before operation, please copy the routine “**Remap**” in “4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

---

**Note: the input command should be case sensitive and the keywords can be complemented by “Tab” key.**

---

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Remap/**” and press Enter to enter the shared folder.

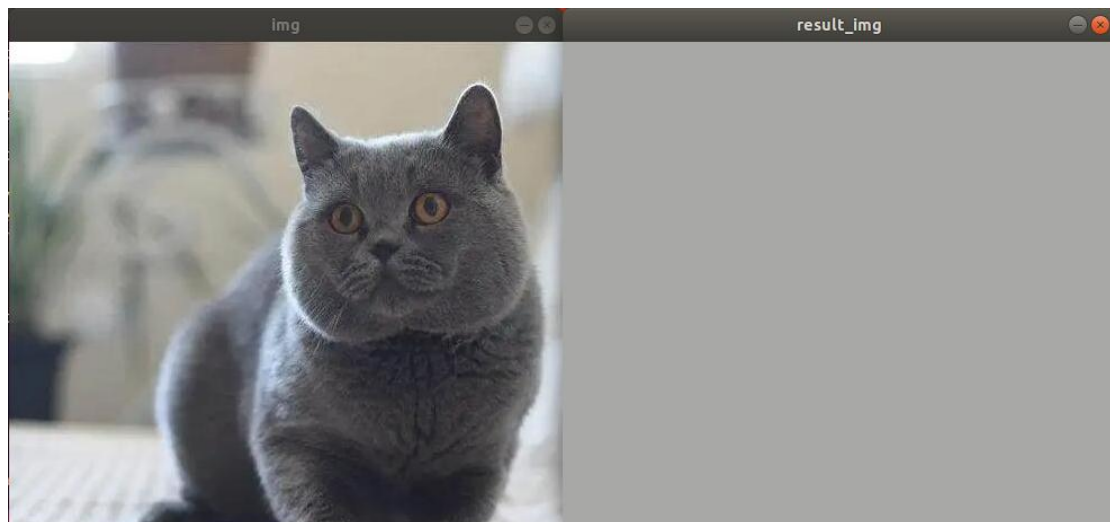
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) Input command “**python3 copy.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 copy.py
```

### 5.1.2 Program Outcome

A pure-colored picture will be output.



### 5.1.3 Program Analysis

The routine “**copy.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code->Remap**”.



```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("1.jpg")
5 rows, cols, ch = img.shape
6 mapx = np.ones(img.shape[:2], np.float32) * 200
7 mapy = np.ones(img.shape[:2], np.float32) * 100
8 result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
9
10 cv2.imshow("img", img)
11 cv2.imshow("result_img", result_img)
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

- 1) Firstly, import the required module through import statement.

```
1 import cv2
2 import numpy as np
```

- 2) Then call **imread()** function in cv2 module to read the image that needs to be scaled.

```
img = cv2.imread('1.jpg')
```

- 3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
5 rows, cols, ch = img.shape
```

- 4) mapx and mapy separately set the x axis and y axis coordinate. Map all the pixels on the target image to the pixels on 100<sup>th</sup> row, 200<sup>th</sup> column of the original image.

```
6 mapx = np.ones(img.shape[:2], np.float32) * 200
7 mapy = np.ones(img.shape[:2], np.float32) * 100
```

- 5) After setting, the picture before and after the pixels are copied can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program.

```
10 cv2.imshow("img", img)
11 cv2.imshow("result_img", result_img)
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

`cv2.waitKey()` is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

`cv2.destroyAllWindows()` is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

## 5.2 Copy the Whole Image

### 5.2.1 Operation Steps

Besides the pixels can be copied, the whole image can also be copied. For example, copy the whole original picture to the right.



Before operation, please copy the routine “**Remap**” in “**4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code**” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

---

**Note: the input command should be case sensitive and the keywords can be complemented by “Tab” key.**

---

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Remap/**” and press Enter to

enter the shared folder.

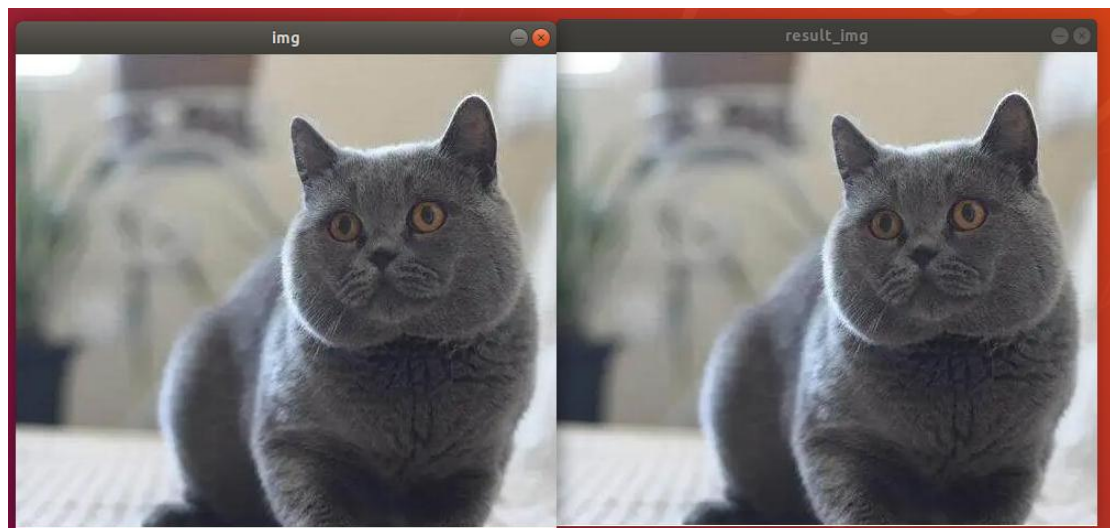
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) Input command “**python3 copy\_all.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 copy_all.py
```

### 5.2.2 Program Outcome

Correspond all the pixels of the original image to those of original image.  
The final output image is as follow.



### 5.2.3 Program Analysis

The routine “**copy\_all.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code**”.

```

1  import cv2
2  import numpy as np
3
4  img = cv2.imread("1.jpg")
5  rows, cols, ch = img.shape
6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),j)#set Y-axis coordinate of each point mapped on the original picture
11         mapy.itemset((i,j),i)#set X-axis coordinate of each point mapped on the original picture
12 result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
13 cv2.imshow("img", img)
14 cv2.imshow("result_img", result_img)
15 cv2.waitKey()
16 cv2.destroyAllWindows()
17

```

1) Firstly, import the required module through import statement.

```

1  import cv2
2  import numpy as np

```

2) Then call **imread()** function in cv2 module to read the image.

```

img = cv2.imread('1.jpg')

```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```

5  rows, cols, ch = img.shape

```

4) mapx and mapy separately set the x axis and y axis coordinate.

```

6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),j)#set Y-axis coordinate of each point mapped on the original picture
11         mapy.itemset((i,j),i)#set X-axis coordinate of each point mapped on the original picture

```

5) After setting, the picture before and after the pixels are copied can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program.

```

10  cv2.imshow("img", img)
11  cv2.imshow("result_img", result_img)
12  cv2.waitKey()
13  cv2.destroyAllWindows()

```

`cv2.waitKey()` is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait `n ms` set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

`cv2.destroyAllWindows()` is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

### 5.3 Rotate Around X Axis

If make the image flip around x axis,

- 1) x axis coordinate remains unchanged.
- 2) The y-axis coordinate after rotation is symmetric with respect to x axis.

Or:

- 1) `map1` remains unchanged
- 2) `map2 = total number of row - 1 - current row number`

#### 5.3.1 Operation Steps



With `cv2.remap()` function, the pixels can be remapped, and also be flipped and then remapped. Ensure the x axis coordinate remains unchanged and y-axis coordinate after rotation is symmetric with respect to x axis.

Before operation, please copy the routine “**Remap**” in “**4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code**” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.



**Note: the input command should be case sensitive and the keywords can be complemented by “Tab” key.**

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input the command “**cd /mnt/hgfs/Share/Remap/**” and press Enter to enter the shared folder.

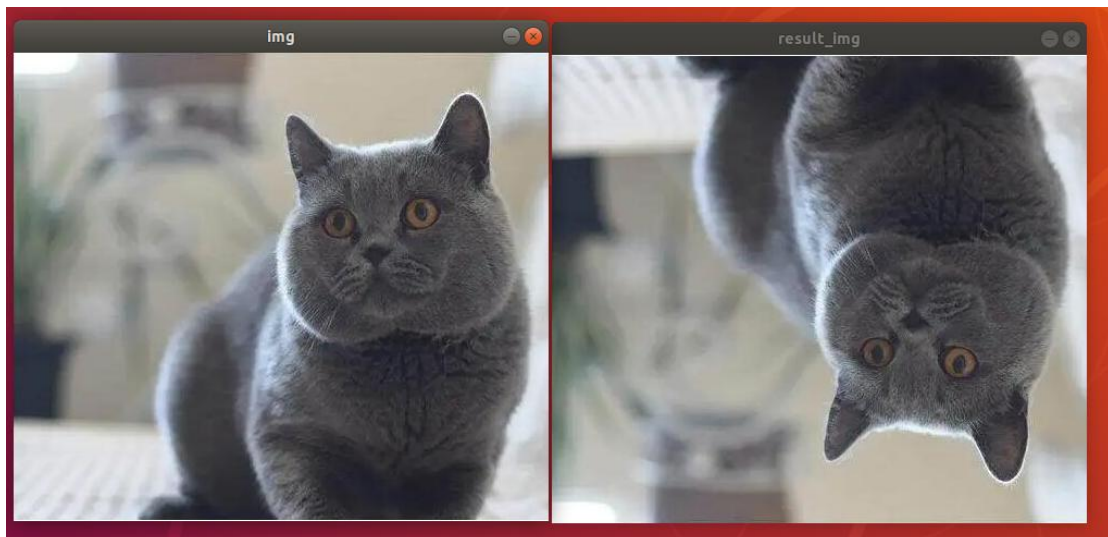
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) Input command “**python3 x\_rotation.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 x_rotation.py
```

### 5.3.2 Program Outcome

The final output image is as follow.



### 5.3.3 Program Analysis

The routine “**x\_rotation.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric**

## Transformation->Routine Code”.

```

1  import cv2
2  import numpy as np
3
4  img = cv2.imread("1.jpg")
5  rows, cols, ch = img.shape
6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),j)
11         mapy.itemset((i,j),rows-1-i)#just modify this line of code. Symmetrical
12
13 result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
14 cv2.imshow("img", img)
15 cv2.imshow("result_img", result_img)
16 cv2.waitKey()
17 cv2.destroyAllWindows()
18

```

- 1) Firstly, import the required module through import statement.

```

1  import cv2
2  import numpy as np

```

- 2) Then call **imread()** function in cv2 module to read the image that needs to be scaled.

```
img = cv2.imread('1.jpg')
```

- 3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
5  rows, cols, ch = img.shape
```

- 4) mapx and mapy separately set the x axis and y axis coordinate. map1 remains unchanged, and map2 = “**total number of row - 1 - current row number**”

```

6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),j)
11         mapy.itemset((i,j),rows-1-i)#just modify this line of code. Symmetrical

```



5) After setting, the picture before and after can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program.

```
10 cv2.imshow("img", img)
11 cv2.imshow("result_img", result_img)
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

## 5.4 Rotate Around Y Axis

If make the image flip around y axis,

- 1) y axis coordinate remains unchanged.
- 2) The x-axis coordinate after rotation is symmetric with respect to y axis.

Or:



- 1) Map2 remains unchanged
- 2) map2 = "total number of column - 1 - current column number"

### 5.4.1 Operation Steps

Before operation, please copy the routine "**Remap**" in "**4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code**" to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

**Note:** the input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “”, and then “” or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/Remap/**” and press Enter to enter the shared folder.

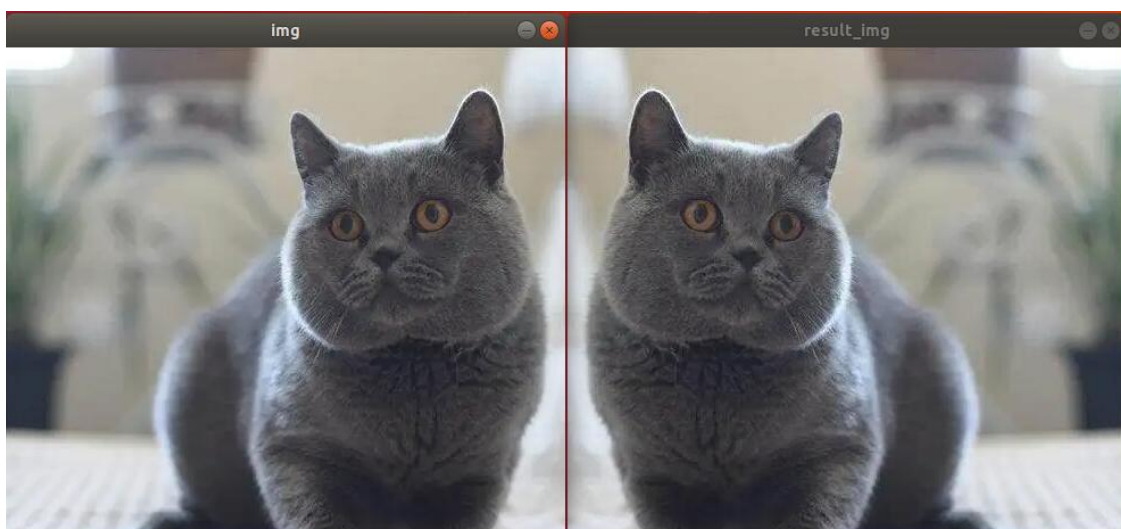
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) Input command “**python3 copy\_all.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 y_rotation.py
```

## 5.4.2 Program Outcome

The final output image is as follow.



## 5.4.3 Program Analysis

The routine “**y\_rotation.py**” can be found in “**4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code**”.

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("1.jpg")
5 rows, cols, ch = img.shape
6 mapx = np.ones(img.shape[:2], np.float32)
7 mapy = np.ones(img.shape[:2], np.float32)
8 for i in range(rows):
9     for j in range(cols):
10         mapx.itemset((i,j),cols-1-j)#just modify this line of code.
11         mapy.itemset((i,j),i)#
12 result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
13 cv2.imshow("img", img)
14 cv2.imshow("result_img", result_img)
15 cv2.waitKey()
16 cv2.destroyAllWindows()
17
```

1) Firstly, import the required module through import statement.

```
1 import cv2
2 import numpy as np
```

2) Then call **imread()** function in cv2 module to read the image that needs to be scaled.

```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
5 rows, cols, ch = img.shape
```

4) mapx and mapy separately set the x axis and y axis coordinate. mapy remains unchanged, and mapx = “**total number of column - 1 - current column number**”

```
6 mapx = np.ones(img.shape[:2], np.float32)
7 mapy = np.ones(img.shape[:2], np.float32)
8 for i in range(rows):
9     for j in range(cols):
10         mapx.itemset((i,j),cols-1-j)#just modify this line of code.
11         mapy.itemset((i,j),i)#
```

5) After setting, the picture before and after can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program.

```
13 cv2.imshow("img", img)
14 cv2.imshow("result_img", result_img)
15 cv2.waitKey()
16 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

## 5.5 Rotate Around XY Axis

If make the image rotate around x axis and y axis,

- 1) The x-axis coordinate after rotation is symmetric with respect to y axis.
- 2) The y-axis coordinate after rotation is symmetric with respect to x axis.

Or:

- 1) map1 = "total number of row - 1 - current row number"
- 2) map2= "total number of row - 1 - current row number"

### 5.5.1 Operation Steps


Before operation, please copy the routine “**Remap**” in “4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

---

**Note:** the input command should be case sensitive and the keywords can be complemented by “Tab” key.

---

1) Open virtual machine and start the system. Click “

2) Input command “**cd /mnt/hgfs/Share/Remap/**” and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

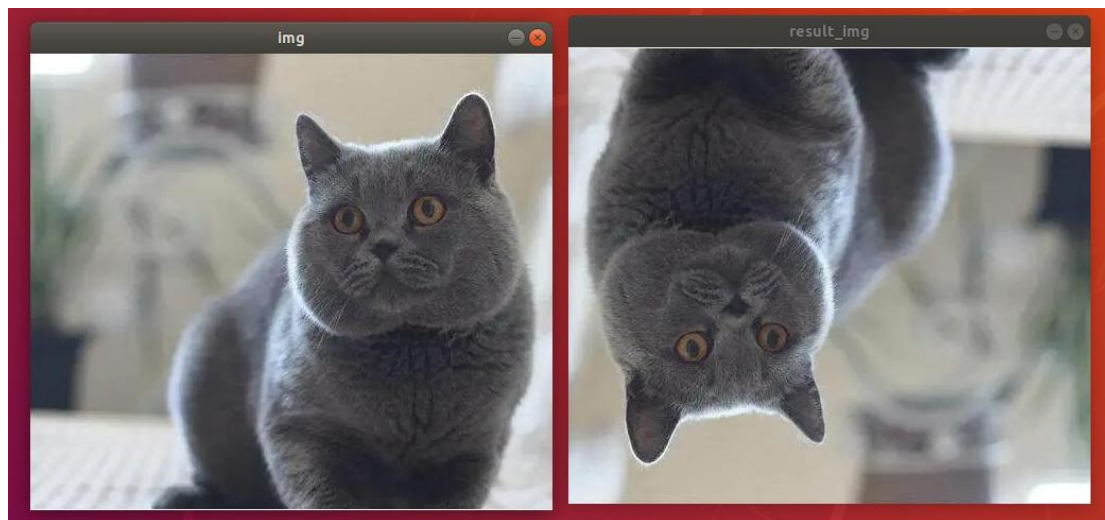
3) Input command “**python3 xy\_rotation.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 xy_rotation.py
```

### 5.5.2 Program Outcome

The final output picture is as follow.





### 5.5.3 Program Analysis

The routine “**xy\_rotation.py**” can be found in “4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code”.

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("1.jpg")
5  rows, cols, ch = img.shape
6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),cols-1-j)
11         mapy.itemset((i,j),rows-1-i)
12  result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
13  cv2.imshow("img", img)
14  cv2.imshow("result_img", result_img)
15  cv2.waitKey()
16  cv2.destroyAllWindows()
17
```

1) Firstly, import the required module through import statement.

```
1  import cv2
2  import numpy as np
```

2) Then call **imread()** function in cv2 module to read the image that needs to be scaled.

```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
rows, cols, ch = img.shape
```

4) mapx and mapy separately set the x axis and y axis coordinate. The value of mapy is changed as “total number of row - 1 - current row number”, and mapx = “total number of column - 1 - current column number”

```
6 mapx = np.ones(img.shape[:2], np.float32)
7 mapy = np.ones(img.shape[:2], np.float32)
8 for i in range(rows):
9     for j in range(cols):
10         mapx.itemset((i,j),cols-1-j)
11         mapy.itemset((i,j),rows-1-i)
```

5) After setting, the picture before and after can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program.

```
10 cv2.imshow("img", img)
11 cv2.imshow("result_img", result_img)
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.



cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.

## 5.6 Compress Image

Compressing image is to compress the original image by half.

### 5.6.1 Operation Steps



Before operation, please copy the routine **"Scale"** in **"4.OpenCV->Lesson 8 Image Processing --- Geometric Transformation->Routine Code"** to the shared folder.

For how to configure the shared folder, please refer to the file in **"2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement"**.

---

**Note: the input command should be case sensitive and the keywords can be complemented by "Tab" key.**

---

1) Open virtual machine and start the system. Click "", and then "" or press **"Ctrl+Alt+T"** to open command line terminal.

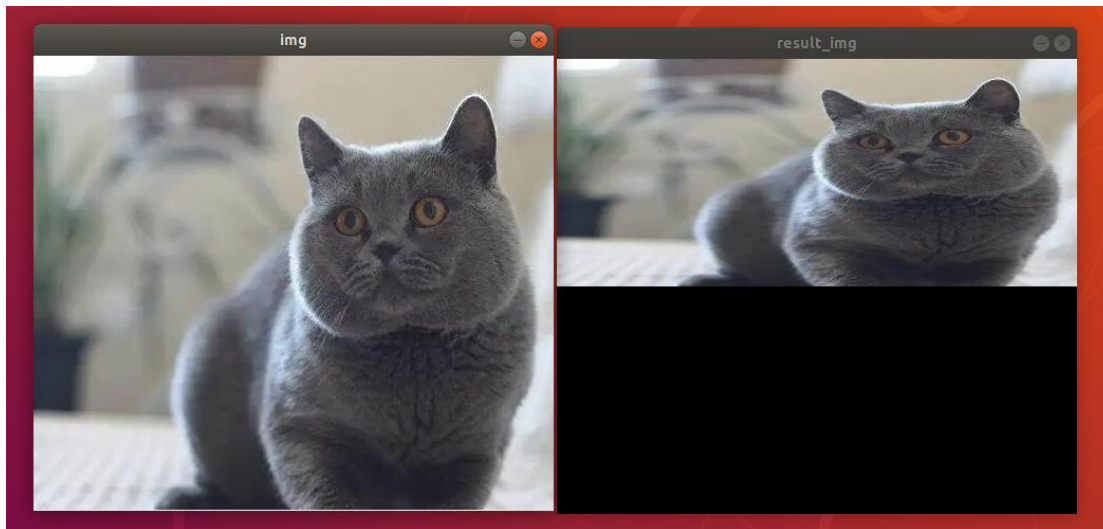
2) Input command **"cd /mnt/hgfs/Share/Remap/"** and press Enter to enter the shared folder.

```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) Input command **"python3 half\_size.py"** and press Enter to run the code.

```
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$ python3 half_size.py
```

### 5.6.2 Program Outcome



### 5.6.3 Program Analysis

The routine “half\_size.py” can be found in “4. OpenCV Computer Vision Lesson->Lesson 8 Image Processing---Geometric Transformation->Routine Code”.

```

1  import cv2
2  import numpy as np
3
4  img = cv2.imread("1.jpg")
5  rows, cols, ch = img.shape
6  mapx = np.ones(img.shape[:2], np.float32)
7  mapy = np.ones(img.shape[:2], np.float32)
8  for i in range(rows):
9      for j in range(cols):
10         mapx.itemset((i,j),j)
11         mapy.itemset((i,j),2*i) #just modify this line of code
12 result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
13 cv2.imshow("img", img)
14 cv2.imshow("result_img", result_img)
15 cv2.waitKey()
16 cv2.destroyAllWindows()
17

```

- 1) Firstly, import the required module through import statement.

```

1  import cv2
2  import numpy as np

```

- 2) Then call **imread()** function in cv2 module to read the image.

```
img = cv2.imread('1.jpg')
```

3) Return the number of row, column and channel of the image pixel to rows, cols and ch.

```
rows, cols, ch = img.shape
```

4) mapx and mapy separately set the x axis and y axis coordinate and double the X axis

```
6 mapx = np.ones(img.shape[:2], np.float32)
7 mapy = np.ones(img.shape[:2], np.float32)
8 for i in range(rows):
9     for j in range(cols):
10         mapx.itemset((i,j),j)
11         mapy.itemset((i,j),2*i) #just modify this line of code
```

5) After setting, the picture before and after can be displayed through imshow function. Lastly, close the window through the function, and you can press any key to exit the program

```
10 cv2.imshow("img", img)
11 cv2.imshow("result_img", result_img)
12 cv2.waitKey()
13 cv2.destroyAllWindows()
```

cv2.waitKey() is a keyboard binding function. Its time unit is milliseconds (ms). The function will wait n ms set in bracket to check if there is any keyboard input. If there is, the ASCII value of the key is returned. -1 will be returned if there is no keyboard input. Generally we set it to 0, the function will wait for keyboard input endlessly.

cv2.destroyAllWindows() is used to delete the window. If there is no parameter in the bracket, all the windows will be deleted. If you input the specific value of the window, the designated window will be removed.