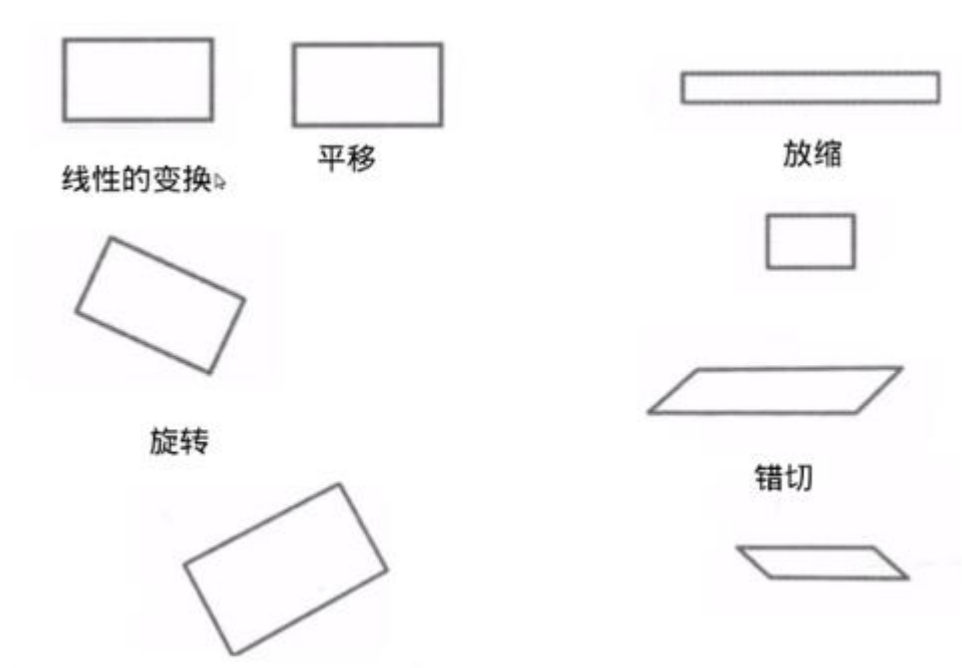


# 8 图像处理-几何变换

## 1.简述

图像几何变换又称为图像空间变换，它将一幅图像中的坐标位置映射到另一幅图像中的新坐标位置。几何变换并不改变图像的像素值，只是在图像平面上进行像素的重新安排。

根据OpenCV函数的不同，本节课将映射关系划分为缩放、翻转、仿射变换、透视等。



## 2.缩放

图像缩放，是指对图像的大小进行调整的过程。比如放大图片，缩小图片。在OpenCV中，使用函数cv2.resize()实现对图像的缩放，此函数的具体形式为：

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

其中dst代表输出的目标图像，该图像的类型与src相同，其大小为dsize（当该值非零时），或者可以通过src.size()、fx、fy计算得到。

1) src代表需要缩放的原始图像。

- 2) `dsize`代表输出图像大小。
- 3) `fx`代表水平方向的缩放比例，非0。
- 4) `fy`代表垂直方向的缩放比例，非0。
- 5) `interpolation` 代表插值方式，具体如下表所示：

类型	说明
<code>cv2.INTER_NEAREST</code>	最临近插值
<code>cv2.INTER_LINEAR</code>	双线性插值（默认方式）
<code>cv2.INTER_CUBIC</code>	三次样条插值。首先对源图像附近的 $4 \times 4$ 近邻区域进行三次样条拟合，然后将目标像素对应的三次样条值作为目标图像对应像素点的值。
<code>cv2.INTER_AREA</code>	区域插值，根据当前像素点周边区域的像素实现当前像素点的采样。该方法类似最临近插值方式。
<code>cv2.INTER_LANCZOS4</code>	一种使用 $8 \times 8$ 近邻的 Lanczos 插值方法
<code>cv2.INTER_LINEAR_EXACT</code>	位精确双线性插值
<code>cv2.INTER_MAX</code>	差值编码掩码
<code>cv2.WARP_FILL_OUTLIERS</code>	标志，填补目标图像中的所有像素。如果它们中的一些对应源图像中的奇异点（离群值），则将它们设置为零
<code>cv2.WARP_INVERSE_MAP</code>	标志，逆变换。例如，极坐标变换：如果 <code>flag</code>

	<p>未被设置，则进行转换：<math>\text{dst}(\rho, \phi) = \text{src}(x, y)</math></p> <p>如果 flag 被设置，则进行转换：<math>\text{dst}(x, y) = \text{src}(\rho, \phi)</math></p>
--	--

## 2.1 实验步骤

本节例程会对图像分别进行缩放。



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Scale”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

2) 输入指令“`cd /mnt/hgfs/Share/Scale`”，并按下回车，进入共享文件夹。

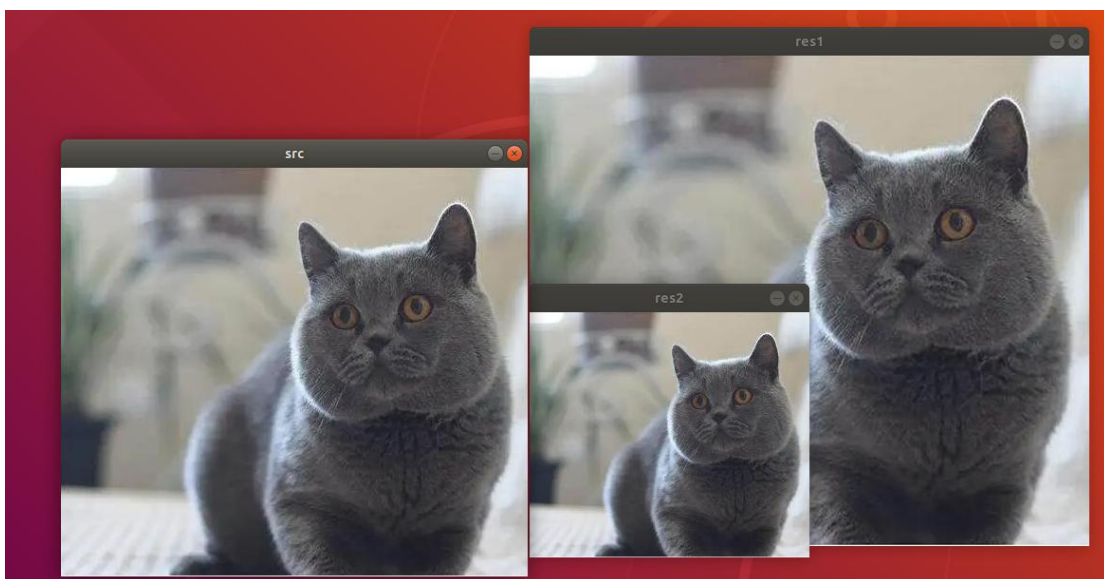
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Scale/
hiwonder@ubuntu:/mnt/hgfs/Share/Scale$
```

3) 输入指令“`python3 Scale.py`”，并按下回车，运行例程。

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

## 2.2 实验效果

执行程序后，图像处理的结果如下图所示：



- 1) **src**: 原始图片。尺寸为宽度492像素，高度为430像素。
- 2) **res1**: 原始图片放大的大小。尺寸为宽度590像素，高度512像素。
- 3) **res2**: 原始图片缩小的大小。尺寸为295像素，高度为258像素。

## 2.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码->Scale”下查看例程“Scale.py”。

```
import numpy as np
import cv2 as cv

src = cv.imread('1.jpg')
# method 直接设置输出尺寸
height, width = src.shape[:2] # 获得原尺寸
res1 = cv.resize(src, (int(1.2*width), int(1.2*height)), interpolation=cv.INTER_CUBIC)
res2 = cv.resize(src, (int(0.6*width), int(0.6*height)), interpolation=cv.INTER_CUBIC)
cv.imshow("src", src)
cv.imshow("res1", res1)
cv.imshow("res2", res2)
print("src.shape=", src.shape)
print("res1.shape=", res1.shape)
print("res2.shape=", res2.shape)
cv.waitKey()
cv.destroyAllWindows()
```

首先，通过import语句导入所需模块。

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

通过调用cv2模块中的imread()函数，读取用于缩放处理的图像。

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

函数括号内的参数是图像名称。

原始图片宽度492像素，高度为430像素。我们这里通过参数dsize（输出图像大小）的方式来指定目标图像res1（自定义命名）和res2（自定义命名）的大小。

dsize内第1个参数对应缩放后图像的宽度（width，即列数cols，与参数fx相关），第2个参数对应缩放后图像的高度（height，即行数 rows，与参数fy相关）。

这种方式，如果指定参数dsize的值，则无论是否指定了参数fx和fy的值，都由参数dsize来决定目标图像的大小。

```
6 height, width = src.shape[:2] # 获得原尺寸
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)
```

所以在这里，我们需要让程序先获取原图尺寸信息，然后直接在宽度和高度上进行处理。示例将res1宽放大为原来的1.2倍，高放大为原来的1.2倍，计算后得到为宽度492x1.2=590像素，高度430x 1.2=516像素，最终呈现原图放大效果。

缩小这里示例将res2宽度变为原来0.6倍，高度变为原来0.6倍，计算后得到492x0.6=295像素，高度430x0.6=258像素，最终呈现原图缩小的效果。如下图所示，我们可将处理前后的图片尺寸信息打印出来：

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

### 3.仿射变换

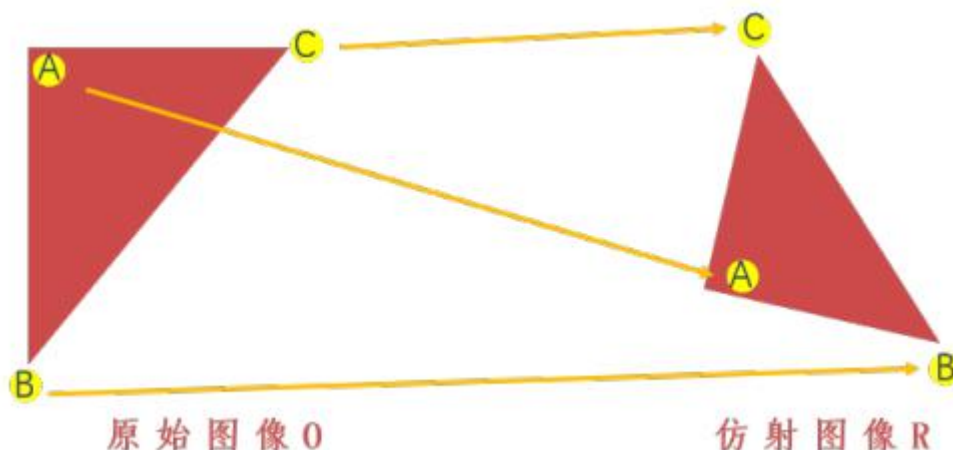
仿射变换是指图像可以通过一系列的几何变换来实现平移、旋转等多种操作。该变换能够保持图像的平直性和平行性。

平直性是指图像经过仿射变换后，直线仍然是直线，平行性是指图像在完成仿射变换后，平行线仍是平行线。

平移和旋转都是仿射变换的特例，OpenCV所用函数都为cv2.warpAffine()，其通过一个变换矩阵（映射矩阵）M 实现变换，平移和旋转转换矩阵不同）具体为：

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

如下图所示，可以通过一个变换矩阵 M，将原始图像O变换为仿射图像R。



cv2.warpAffine()函数的语法格式如下：

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

式中：

- 1) **dst**: 代表仿射后的输出图像，该图像的类型和原始图像的类型相同。dsize决定输出图像的实际大小。
- 2) **src**: 代表要仿射的原始图像。
- 3) **M**: 代表一个2×3的变换矩阵。使用不同的变换矩阵，就可以实现不同的仿射变换。  
dsize 代表输出图像的尺寸大小。
- 4) **flags**: 代表插值方法，默认为INTER\_LINEAR。当该值为WARP\_INVERSE\_MAP时，

意味着M是逆变换类型，实现从目标图像dst到原始图像src的逆变换。具体可选值borderMode代表边类型，默认为BORDER\_CONSTANT。

当该值为BORDER\_TRANSPARENT时，意味着目标图像内的值不做改变，这些值对应原始图像内的异常值。

5) **borderValue**: 代表边界值，默认是0。

通过以上分析可知，在OpenCV中使用函数cv2.warpAffine()实现仿射变换，忽略其可选参数后的语法格式为：

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

其通过转换矩阵 M 将原始图像 src 转换为目标图像 dst：

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

因此，进行何种形式的仿射变换完全取决于转换矩阵 M。下面分别介绍通过不同的转换矩阵M实现的不同的仿射变换。

### 3.1 平移

平移是物体的移动，如果知道物体平移的坐标，可以创建如下变换矩阵：

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

将其放入类型为np.float32的数组中，将M矩阵赋值给cv2.warpAffine() 函数，即可实现平移。

#### 3.1.1 实验步骤

本节例程会对图像向右平移。

开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Translation”文件夹复制到共享文件夹。



关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。



---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

2) 输入指令“**cd /mnt/hgfs/Share/Translation/**”，并按下回车，进入共享文件夹。

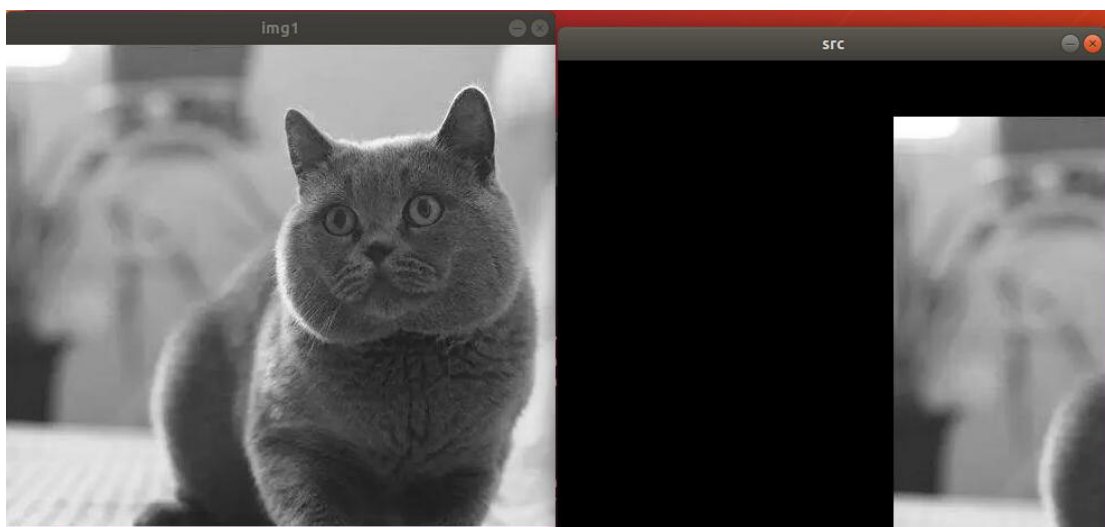
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Translation/  
hiwonder@ubuntu:/mnt/hgfs/Share/Translation$
```

3) 输入指令“**python3 Translation.py**”，并按下回车，运行例程。

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

### 3.1.2 实验效果

执行程序后，图像处理的结果如下图所示：



### 3.1.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码-TranslaTion”下查看例程“Translation.py”。



```
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) 首先导入模块，通过import语句导入所需模块。

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

- 2) 通过调用cv2模块中的imread()函数，读取用于平移处理的图像。

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

- 3) 将图片的像素的行数、列数、频道返回给rows、cols、ch。

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

- 4) 根据前面学习的公式，如果知道物体平移的坐标，可以创建变换矩阵。这里以偏移（300，50）为例：

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

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

设置完成后，我们通过imshow函数分别显示原图及平移后的图片。

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

- 5) 最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

`cv2.waitKey()`是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

`cv2.destroyAllWindows()` 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 3.2 旋转

平移和旋转都是仿射变换的特例，所用函数都是`cv2.warpAffine`，只是转换矩阵M有所不同。在使用函数`cv2.warpAffine()`对图像进行旋转时，可通过函数`cv2.getRotationMatrix2D()`来获取转换矩阵。

该函数的语法格式为：

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

式中：

- 1) `center`为旋转的中心点。
- 2) `angle`为旋转角度，设为正数表示逆时针旋转，负数表示顺时针旋转。
- 3) `scale`为变换尺度（缩放大小）。

图像旋转 $\theta$ 度由变换矩阵M可得到：

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

OpenCV改进了这个矩阵，如下图，使得提供了缩放旋转与可调的旋转中心。

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

$$\alpha = scale \cdot \cos \theta,$$

$$\beta = scale \cdot \sin \theta$$

上述矩阵表示绕center.x，center.y旋转 $\theta$ 度。

例如，想要以图像中心为圆点，逆时针旋转45°，并将目标图像缩小为原始图像的0.6倍，则在调用函数cv2.getRotationMatrix2D()生成转换矩阵M时，如下设置：

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

### 3.2.1 实验步骤

本节例程会对图像逆时针旋转90°。



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Revolve”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

**注意：**输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

2) 输入指令“cd /mnt/hgfs/Share/Revolve/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 Revolve.py”，并按下回车，运行例程。

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

### 3.2.2 实验效果

执行程序后，图像处理的结果如下图所示：



### 3.2.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码->Revolve”下查看例程“Revolve.py”。

```
import cv2
import numpy as np

img = cv2.imread('1.jpg')
rows, cols, ch = img.shape
# 旋转中心 旋转角度 缩放系数
M = cv2.getRotationMatrix2D(((cols-1) / 2.0, (rows-1)/2.0), 90, 1)
# 原图像 变换矩阵 输出图像尺寸中心
dst = cv2.warpAffine(img, M, (cols, rows))

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

- 1) 首先导入模块，通过import语句导入所需模块。

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

- 2) 通过调用cv2模块中的imread()函数，读取用于旋转处理的图像。

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

- 3) 将图片的像素的行数、列数、频道返回给rows、cols、ch。

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

4) 然后以图像中心为圆点，逆时针旋转90°，图像大小和原来保持不变。

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

5) 输出原图的尺寸中心。

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

6) 设置完成后，我们通过imshow函数分别显示原图及旋转后的图片。

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

7) 最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

cv2.waitKey()是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

cv2.destroyAllWindows()用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 4.透视变换

如果说仿射变换是在二维空间中的旋转、平移及缩放，而透视变换则是在三维空间中视角的变化。

透视变换通过函数 cv2.warpPerspective()实现，该函数的语法是：

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

式中：

1) dst代表透视处理后的输出图像，该图像和原始图像具有相同的类型。dsize 决定输出图

像的实际大小。

- 2) `src`代表要透视的图像。
- 3) `M`代表一个 $3 \times 3$ 的变换矩阵。
- 4) `dsize`代表输出图像的尺寸大小。
- 5) `flags`代表插值方法，默认为 `INTER_LINEAR`。当该值为`WARP_INVERSE_MAP` 时，意味着 `M` 是逆变换类型，能实现从目标图像`dst`到原始图像`src`的逆变换。具体可选值 `borderMode`代表边类型， 默认为`BORDER_CONSTANT`。

当该值为`BORDER_TRANSPARENT`时，意味着目标图像内的值不做改变，这些值对应原始图像内的异常值。

## 4.1 实验步骤

本节例程会对图像进行透视变换。



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“**Perspective**”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

**注意：**输入指令时需严格区分大小写，且可以使用“**Tab**”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

2) 输入指令“`cd /mnt/hgfs/Share/Perspective/`”，并按下回车，进入共享文件夹。

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

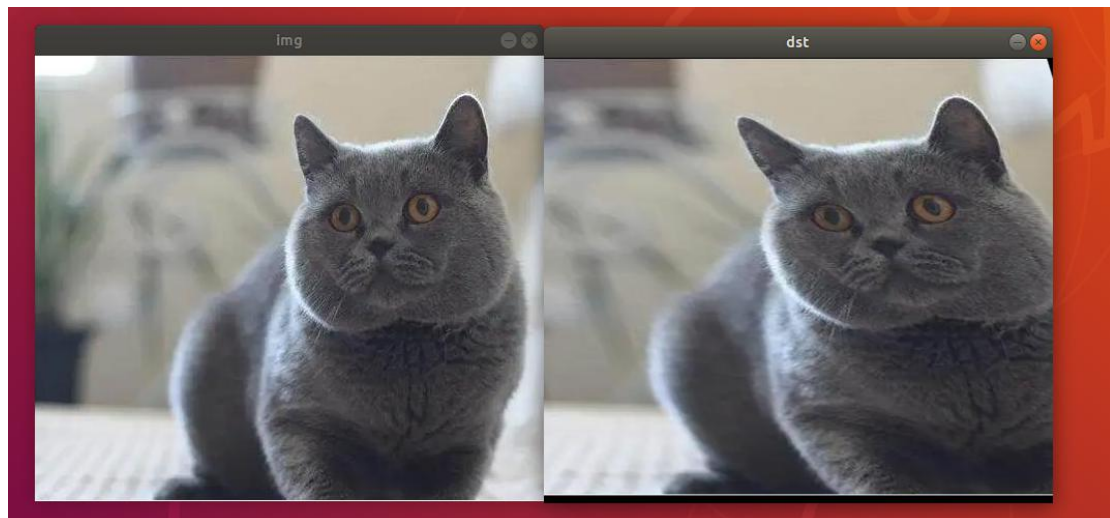
3) 输入指令“`python3 Perspective.py`”，并按下回车，运行例程。

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



## 4.2 实验效果

执行程序后，图像处理的结果如下图所示：



## 4.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码->Perspective”下查看例程“Perspective.py”。

```
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) 首先导入模块，通过import语句导入所需模块。

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

- 2) 通过调用cv2模块中的imread()函数，读取用于透视变换的图像。



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

- 3) 将图片的像素的行数、列数、频道返回给rows、cols、ch。

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

- 4) 在本例中，指定原始图像中平行四边形的四个顶点pts1，指定目标图像中矩形的四个顶点pts2，使用 `M=cv2.getPerspectiveTransform(pts1,pts2)`生成转换矩阵 M。接下来，使用语句`dst=cv2.warpPerspective(img,M,(cols,rows))`完成从平行四边形到矩形的转换。

```
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) 设置完成后，我们通过imshow函数分别显示原图及旋转后的图片。

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

- 6) 最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

`cv2.waitKey()`是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

`cv2.destroyAllWindows()`用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.重映射

什么是重映射？简单来说把一副图像内的像素点按照规则映射到另外一幅图像内的对应位置上去，形成一张新的图像。

因为原图像与目标图像的像素坐标不是一一对应的。一般情况下，我们通过重映射来表

达每个像素的位置 (x,y) , 像这样:

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

在这里g()是目标图像, f()是源图像, 而h(x,y)的映射方法函数, 下面来看个例子。若有一幅图像I, 对其按照下面条件作重映射:

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

图像会按照x轴方向发生翻转, OpenCV提供了多种重映射方式, OpenCV内的重映射函数cv2.remap()提供了更方便、更自由的映射方式, 其语法格式如下:

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

式中:

- 1) dst代表目标图像, 它和src具有相同的大小和类型。
- 2) src代表原始图像。
- 3) map1参数有两种可能的值:

表示(x,y)点的一个映射, 也可以表示 CV\_16SC2 , CV\_32FC1, CV\_32FC2 类型(x,y)点的 x 值。

- 4) map2参数同样有两种可能的值:

当map1表示(x,y)时, 该值为空。

当map1表示(x,y)点的 x 值时, 该值是 CV\_16UC1, CV\_32FC1 类型(x,y)点的y值。

需要注意, map1指代的是像素点所在位置的列号, map2指代的是像素点所在位置的行号。所以通常情况下, 我们将map1写为mapx, 并且将map2写成mapy, 以方便理解。

- 5) Interpolation代表插值方式。
- 6) borderMode代表边界模式。当该值为 BORDER\_TRANSPARENT 时, 表示目标图像内的对应源图像内奇异点 (outliers) 的像素不会被修改。
- 7) borderValue 代表边界值, 该值默认为0。

## 5.1 复制像素点

### 5.1.1 实验步骤

现在我们假设有一个需求，将目标图像内的所有像素点都映射为原始图像内的第100行，200列上的像素点。



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Remap”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

2) 输入指令“`cd /mnt/hgfs/Share/Remap/`”，并按下回车，进入共享文件夹。

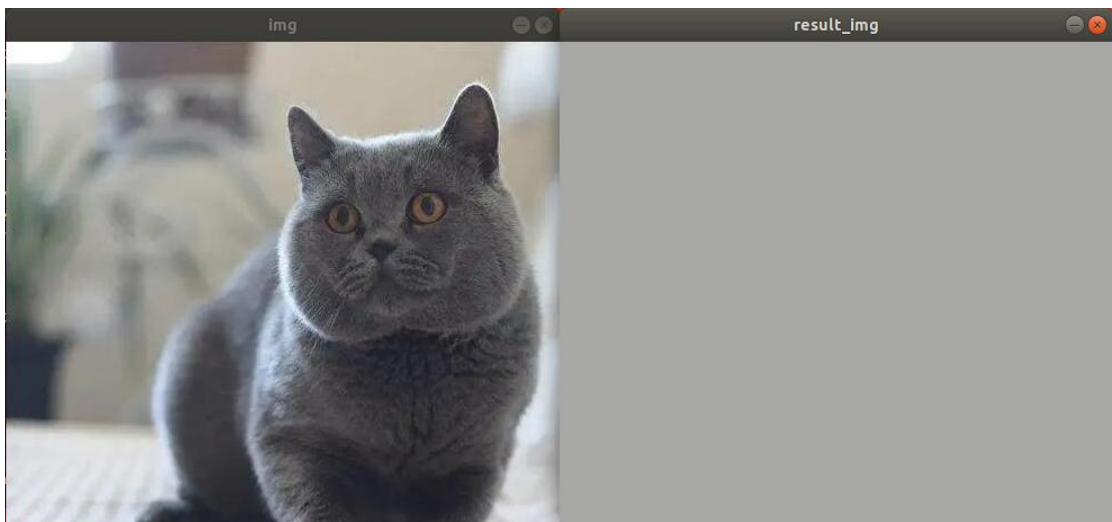
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) 输入指令“`python3 copy.py`”，并按下回车，运行例程。

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

### 5.1.2 实验效果

执行程序后，会得到一个非常纯色的图像。效果如下图所示：



### 5.1.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码->Remap”下查看例程“copy.py”。

```
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) 首先导入模块，通过import语句导入所需模块。

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

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

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

- 3) 将图片的像素的行数、列数、频道返回给rows、cols、ch。

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

4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标，将目标图像的所有像素点都映射为原始图像内的第100行，200列上的像素点。

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

5) 设置完成后，我们通过imshow函数分别显示原图及复制像素点后的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

cv2.waitKey()是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

cv2.destroyAllWindows() 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.2 复制整个图像

### 5.2.1 实验步骤

那么既然能copy某个像素点，那么也肯定能copy整个图像。下面，我们将原图全部copy到右边。



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Remap”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

- 2) 输入指令“`cd /mnt/hgfs/Share/Remap/`”，并按下回车，进入共享文件夹。

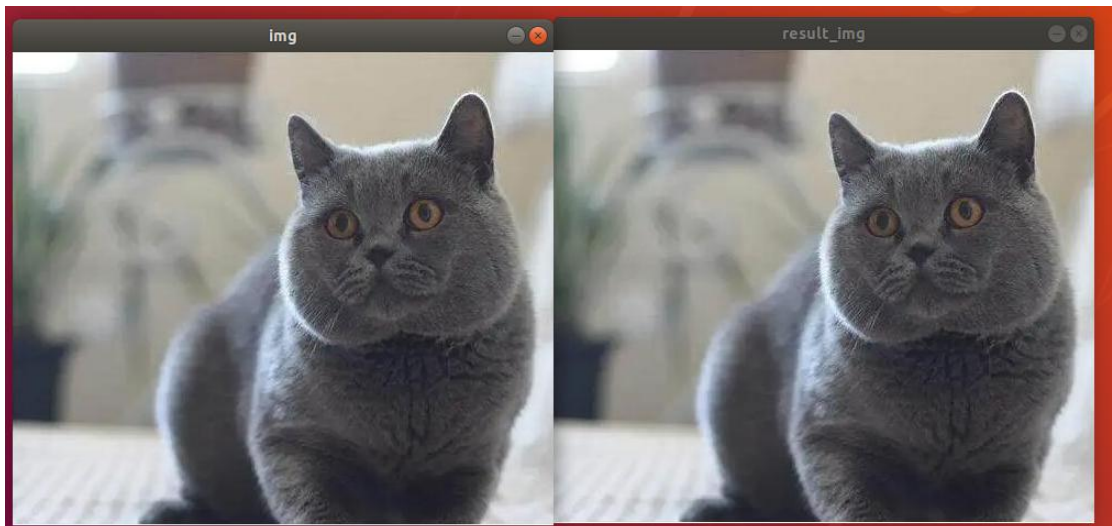
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

- 3) 输入指令“`python3 copy_all.py`”，并按下回车，运行例程。

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

### 5.2.2 实验效果

我们将所有像素点一一对应，就完成了copy原图像。执行程序后，图像处理的结果如下图所示：



### 5.2.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码”下查看例程“`copy_all.py`”。



```

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)#设置每个点映射原图的y坐标
11         mapy.itemset((i,j),i)#设置每个点映射原图的x坐标
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()

```

- 1) 首先导入模块，通过import语句导入所需模块。

```

1  import cv2
2  import numpy as np

```

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

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

- 3) 将图片的像素的行数、列数、频道返回给rows、cols、channels。

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

- 4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标。

```

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)#设置每个点映射原图的y坐标
11         mapy.itemset((i,j),i)#设置每个点映射原图的x坐标

```

- 5) 设置完成后，我们通过imshow函数分别显示原图及复制后的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

```

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

```



`cv2.waitKey()`是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

`cv2.destroyAllWindows()` 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.3 绕X轴旋转

如果想让图像绕着x轴翻转，意味着在映射过程中：

- 1) x坐标轴的值保持不变。
- 2) y坐标轴的值以x轴为对称轴进行交换。

反映在map1和map2上：

- 1) map1的值保持不变。
- 2) map2的值调整为“总行数-1-当前行号”。

需要注意，OpenCV 中行号的下标是从 0 开始的，所以在对称关系中存在“当前行号+对称行号=总行数-1”的关系。因此，在绕着x轴翻转时，map2中当前行的行号调整为“总行数-1-当前行号”。

### 5.3.1 实验步骤

通过`cv2.remap()`函数，我们不仅可以重映射像素点，还可以翻转过来映射，也就是通过它实现X轴的翻转效果，只要保证X轴不变，并且Y坐标值以X轴为对称进行交换即可。



开始操作前，需要先将目录“**第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码**”下的例程“**Remap**”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“**第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法**”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

2) 输入指令“**cd /mnt/hgfs/Share/Remap/**”，并按下回车，进入共享文件夹。

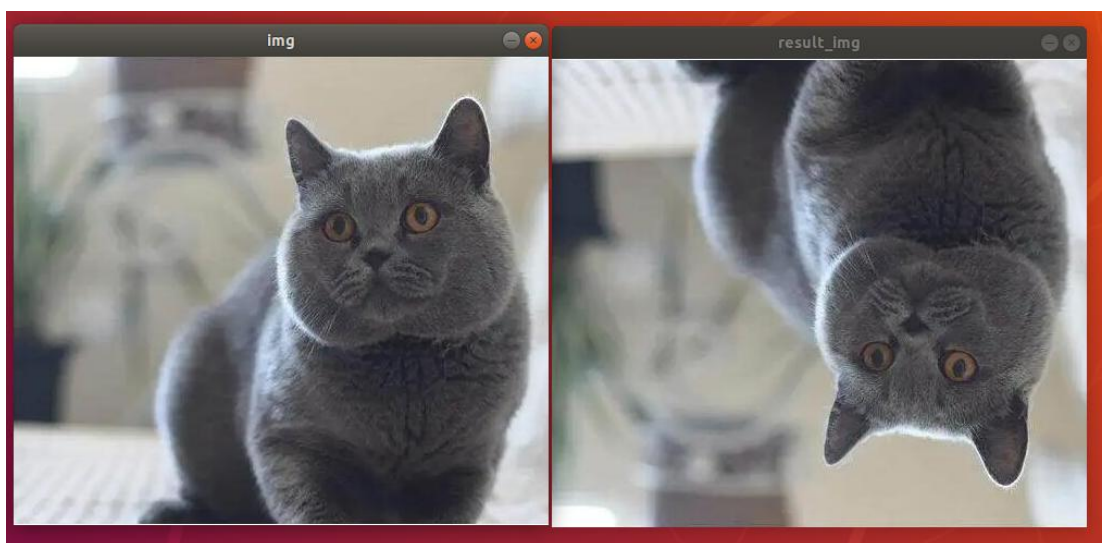
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) 输入指令“**python3 x\_rotation.py**”，并按下回车，运行例程。

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

### 5.3.2 实验效果

执行程序后，实现了X轴的翻转效果,图像处理的结果如下图所示：



### 5.3.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码”下查看例程“**x\_rotation.py**”。

```

import cv2
import numpy as np

img = cv2.imread("1.jpg")
rows, cols, ch = img.shape
mapx = np.ones(img.shape[:2], np.float32)
mapy = np.ones(img.shape[:2], np.float32)
for i in range(rows):
    for j in range(cols):
        mapx.itemset((i,j),j)
        mapy.itemset((i,j),rows-1-i)#修改这一行即可，对称
result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
cv2.imshow("img", img)
cv2.imshow("result_img", result_img)
cv2.waitKey()
cv2.destroyAllWindows()

```

- 1) 首先导入模块，通过import语句导入所需模块。

```

1 import cv2
2 import numpy as np

```

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

```

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

```

- 3) 将图片的像素的行数、列数返回给rows、cols、channels。

```

5 rows, cols, ch = img.shape

```

- 4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标。所以mapx的值保持不变，mapy的值调整为“总行数-1-当前行号”。

```

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)#修改这一行即可，对称

```

- 5) 设置完成后，我们通过imshow函数分别显示原图及绕X轴旋转的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

cv2.waitKey()是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无限等待键盘的输入。

cv2.destroyAllWindows() 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.4 绕Y轴旋转

如果想让图像绕着y轴翻转，意味着在映射过程中：

- 1) y坐标轴的值保持不变。
- 2) x坐标轴的值以y轴为对称轴进行交换。

反映在map1和map2上：

- 1) map2的值保持不变。
- 2) map1的值调整为“总列数-1-当前列号”。

需要注意，OpenCV 中列号的下标是从0开始的，所以在对称关系中存在“**当前列号+对称列号=总列数-1**”的关系。因此，在绕着y轴翻转时，map1中当前列的列号调整为“**总列数-1-当前列号**”。

### 5.4.1 实验步骤

只要将X坐标值以Y轴为对称进行交换即可。开始操作前，需要先将目录“**第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码**”下的例程“**Remap**”文件夹复制到共享文件夹。



关于共享文件夹的配置方法，可参考目录“**第2章 Linux系统简介及使用入门->Linux**

基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

2) 输入指令“`cd /mnt/hgfs/Share/Remap/`”，并按下回车，进入共享文件夹。

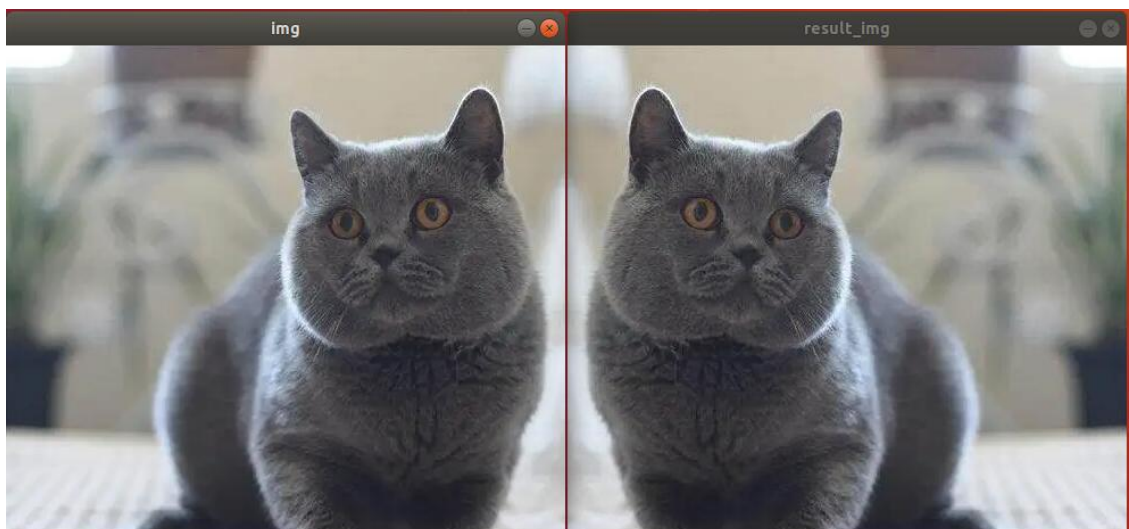
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) 输入指令“`python3 copy_all.py`”，并按下回车，运行例程。

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

### 5.4.2 实验效果

执行程序后，实现了Y轴的翻转效果，图像处理的结果如下图所示：



### 5.4.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码”下查看例程“`y_rotation.py`”。



```

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)#修改这一行即可
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) 首先导入模块，通过import语句导入所需模块。

```

1  import cv2
2  import numpy as np

```

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

```

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

```

- 3) 将图片的像素的行数、列数返回给rows、cols、channels。

```

5  rows, cols, ch = img.shape

```

- 4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标。mapy的值保持不变，mapx的值调整为“总列数-1-当前列号”。

```

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)#修改这一行即可
11         mapy.itemset((i,j),i)#

```

- 5) 设置完成后，我们通过imshow函数分别显示原图及绕Y轴旋转的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

`cv2.waitKey()`是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无限等待键盘的输入。

`cv2.destroyAllWindows()` 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.5 绕XY轴翻转

如果想让图像绕着 x 轴、y 轴翻转，意味着在映射过程中：

- 1) x坐标轴的值以y轴为对称轴进行交换。
- 2) y坐标轴的值以x轴为对称轴进行交换。

反映在map1和map2上：

- 1) map1的值调整为“总列数-1-当前列号”。
- 2) map2的值调整为“总行数-1-当前行号”。

### 5.5.1 实验步骤



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Remap”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

**注意：**输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

- 1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。



2) 输入指令“`cd /mnt/hgfs/Share/Remap/`”，并按下回车，进入共享文件夹。

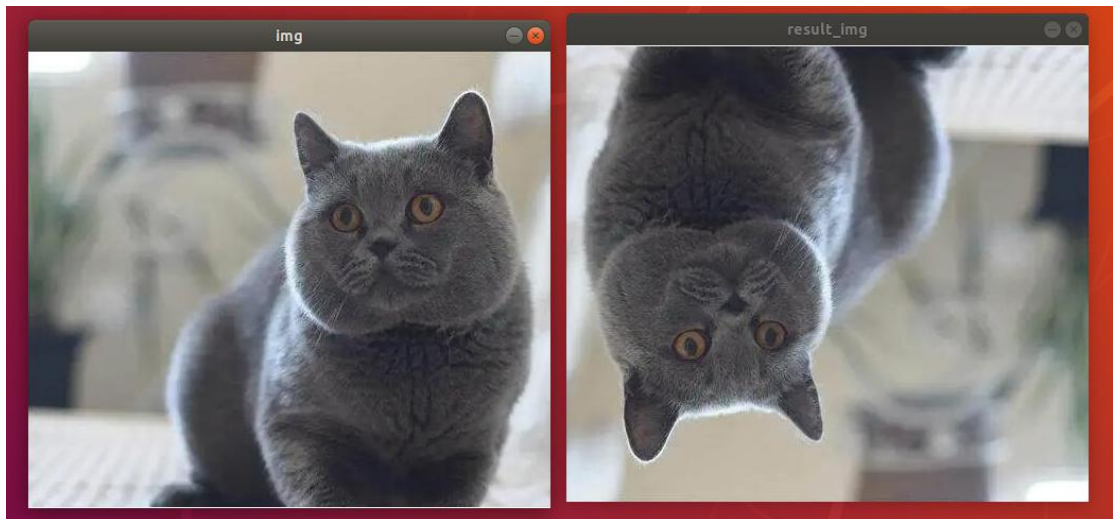
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/Remap/  
hiwonder@ubuntu:/mnt/hgfs/Share/Remap$
```

3) 输入指令“`python3 xy_rotation.py`”，并按下回车，运行例程。

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

### 5.5.2 实验效果

执行程序后，实现了绕XY轴的翻转效果，图像处理的结果如下图所示：



### 5.5.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码”下查看例程“`xy_rotation.py`”。

```

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) 首先导入模块，通过import语句导入所需模块。

```

1  import cv2
2  import numpy as np

```

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

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

- 3) 将图片的像素的行数、列数、频道返回给rows, cols, channels。

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

- 4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标。mapy的值调整为“总行数-1-当前行号”，mapx的值调整为“总列数-1-当前列号”。

```

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) 设置完成后，我们通过imshow函数分别显示原图及复制后的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

cv2.waitKey()是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待（n）里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无限等待键盘的输入。

cv2.destroyAllWindows() 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。

## 5.6 压缩

压缩即将原图缩小一半，在处理更复杂的问题时，就需要对行、列值进行比较复杂的运算来实现。

### 5.6.1 实验步骤



开始操作前，需要先将目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何转换->例程源码”下的例程“Remap”文件夹复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux系统简介及使用入门->Linux基础课程->第3课 Linux系统安装及换源方法”下的文档。

---

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

---

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“Ctrl+Alt+T”，打开命令行终端。

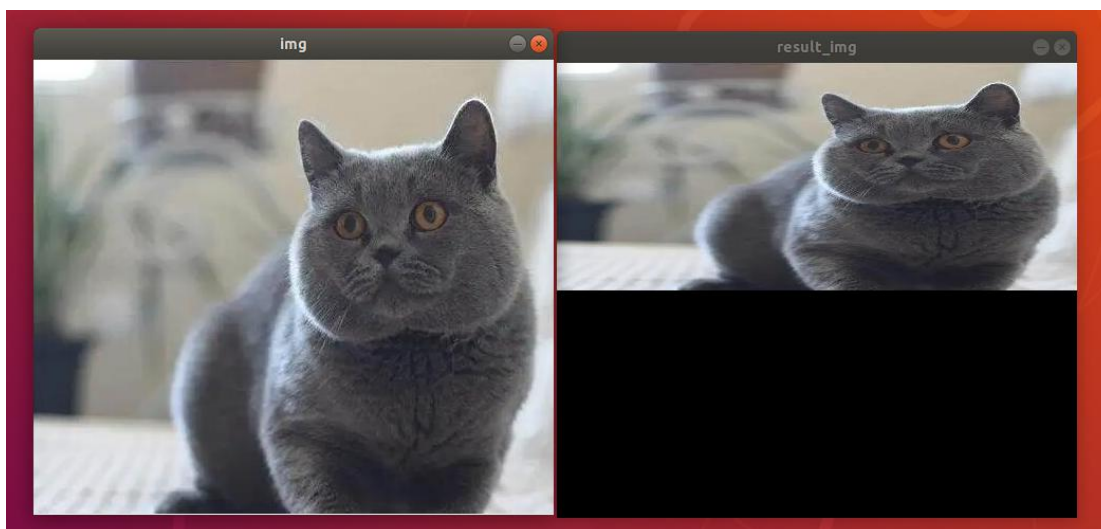
2) 输入指令“cd /mnt/hgfs/Share/Remap/”，并按下回车，进入共享文件夹。

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

3) 输入指令“python3 half\_size.py”，并按下回车，运行例程。

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

## 5.6.2 实验效果



## 5.6.3 实验分析

可以在目录“第4章 OpenCV计算机视觉学习->图像处理进阶篇->第8课 图像处理—几何变换->例程源码”下查看例程“half\_size.py”。

```
import cv2
import numpy as np

img = cv2.imread("1.jpg")
rows, cols, ch = img.shape
mapx = np.ones(img.shape[:2], np.float32)
mapy = np.ones(img.shape[:2], np.float32)
for i in range(rows):
    for j in range(cols):
        mapx.itemset((i,j),j)
        mapy.itemset((i,j),2*i)#修改这行代码即可
result_img = cv2.remap(img, mapx, mapy, cv2.INTER_LINEAR)
cv2.imshow("img", img)
cv2.imshow("result_img", result_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

- 1) 首先导入模块，通过import语句导入所需模块。

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

- 2) 通过调用cv2模块中的imread()函数，读取用于复制像素点的图像。

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

3) 将图片的像素的行数、列数返回给rows、cols、channels。

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

4) 参数mapx和参数mapy分别设置了x轴方向的坐标和y轴方向的坐标。这里将Y轴缩小一半，只需要将X轴设置为2倍即可。

```
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) #修改这行代码即可
```

5) 设置完成后，我们通过imshow函数分别显示原图及复制后的图片，最后通过函数来关闭窗口，这里使用键盘上的任意按键即可退出程序。

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

cv2.waitKey()是一个键盘绑定函数。它的时间量度是毫秒ms。函数会等待(n)里面的n毫秒，看是否有键盘输入。若有键盘输入，则返回按键的ASCII值。没有键盘输入，则返回-1。一般设置为0，将无线等待键盘的输入。

cv2.destroyAllWindows() 用来删除窗口的，()里不指定任何参数，则删除所有窗口，删除特定的窗口，往()输入特定的窗口值。