实验三、几何变换

一、 实验目的

- 学习对图像进行各种几个变换，例如移动，旋转，仿射变换等。
- 将要学到的函数有：cv2.getPerspectiveTransform。

二、 实验原理

OpenCV 提供了两个变换函数，cv2.warpAffine 和 cv2.warpPerspective，使用这两个函数 M =cv2.warpPerspective 接收的参数是 3*3 的变换矩阵。

三、 实验步骤

1 扩展缩放

扩展缩放只是改变图像的尺寸大小。OpenCV 提供的函数 cv2.resize()可以实现这个功能。图像的尺寸可以自己手动设置，也可以指定缩放因子和不同的插值方法。在缩放时，推荐使用 cv2.INTER_AREA，在扩展时推荐使用 v2.INTER_CUBIC（慢）和 v2.INTER_LINEAR。默认情况下所有改变图像尺寸大小的操作使用的插值方法都是 cv2.INTER_LINEAR。

可以使用下面任意一种方法改变图像的尺寸：

```
import cv2
import numpy as np
img=cv2.imread('flower.jpg')
# 下面的 None 本应该是输出图像的尺寸，但是因为后边我们设置了缩放因子
# 因此这里为 None
res=cv2.resize(img,None,fx=2,fy=2,interpolation=cv2.INTER_CUBIC)
#OR
# 直接设置输出图像的尺寸，所以不用设置缩放因子
height,width=img.shape[:2]
res=cv2.resize(img,(2*width,2*height),interpolation=cv2.INTER_CUB
IC)
while(1):#注意缩进
```

```
        cv2.imshow('res',res)
        cv2.imshow('img',img)
        if cv2.waitKey(1) & 0xFF == 27:
            break
    cv2.destroyAllWindows()
```

结果为：

2 旋转

对一个图像旋转角度，需要使用到下面形式的旋转矩阵。

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

但是 OpenCV 允许在任意地方进行旋转，但是旋转矩阵的形式应该修改为

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

其中：

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

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

为了构建这个旋转矩阵，OpenCV 提供了一个函数：**cv2.getRotationMatrix2D**。

下面的例子是在缩放的情况下将图像旋转 45 度。

```
import cv2
import numpy as np
img=cv2.imread('flower.tif',0)
rows,cols=img.shape
# 这里的第一个参数为旋转中心，第二个为旋转角度，第三个为旋转后的缩放因子
# 可以通过设置旋转中心，缩放因子，以及窗口大小来防止旋转后超出边界的问题
M=cv2.getRotationMatrix2D((cols/2,rows/2),45,0.6)
# 第三个参数是输出图像的尺寸中心
dst=cv2.warpAffine(img,M,(cols,rows))
cv2.imwrite('before',dst)
cv2.imshow('after',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

警告：函数 cv2.warpAffine() 的第三个参数的是输出图像的大小，它的格式应该是图像的（宽，高）。应该记住的是图像的宽对应的是列数，高对应的是行数。

下面是结果。

原图

旋转后

3 仿射变换

在仿射变换中，图中所有的平行线在结果图像中同样平行。为了创建这个矩阵，需要从原图像中找到三个点以及它们在输出图像中的位置。然后 **cv2.getAffineTransform** 会创建一个 2x3 的矩阵，最后这个矩阵会被传给函数 **cv2.warpAffine**。

来看看下面的例子，以及选择的点（被标记为绿色的点）

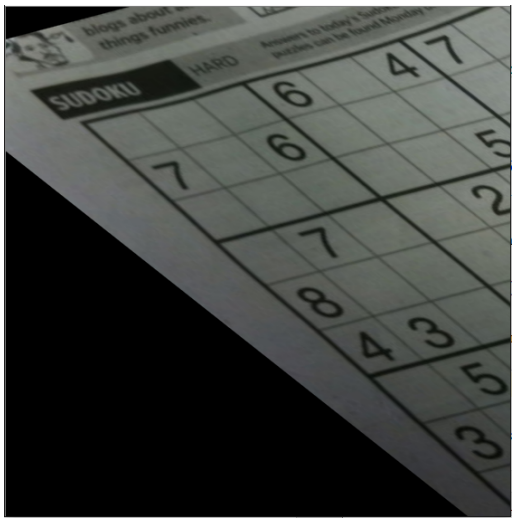
```
import cv2
import numpy as np

img=cv2.imread('drawing.png')
rows,cols,ch=img.shape
pts1=np.float32([[50,50],[200,50],[50,200]])
pts2=np.float32([[10,100],[200,50],[100,250]])
M=cv2.getAffineTransform(pts1,pts2)
dst=cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('Input',img)
```

```
cv2.imshow('Output',dst)
cv2.imwrite('getAffineTransformImg.jpg',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

下面是结果：

原图　　仿射变换后



4 透视变换

对于视角变换，需要一个 3x3 变换矩阵。在变换前后直线还是直线。要构建这个变换矩阵，需要在输入图像上找 4 个点，以及他们在输出图像上对应的位置。这四个点中的任意三个都不能共线。这个变换矩阵可以由函数

cv2.getPerspectiveTransform() 构建。然后把这个矩阵传给函数 cv2.warpPerspective。

代码如下：

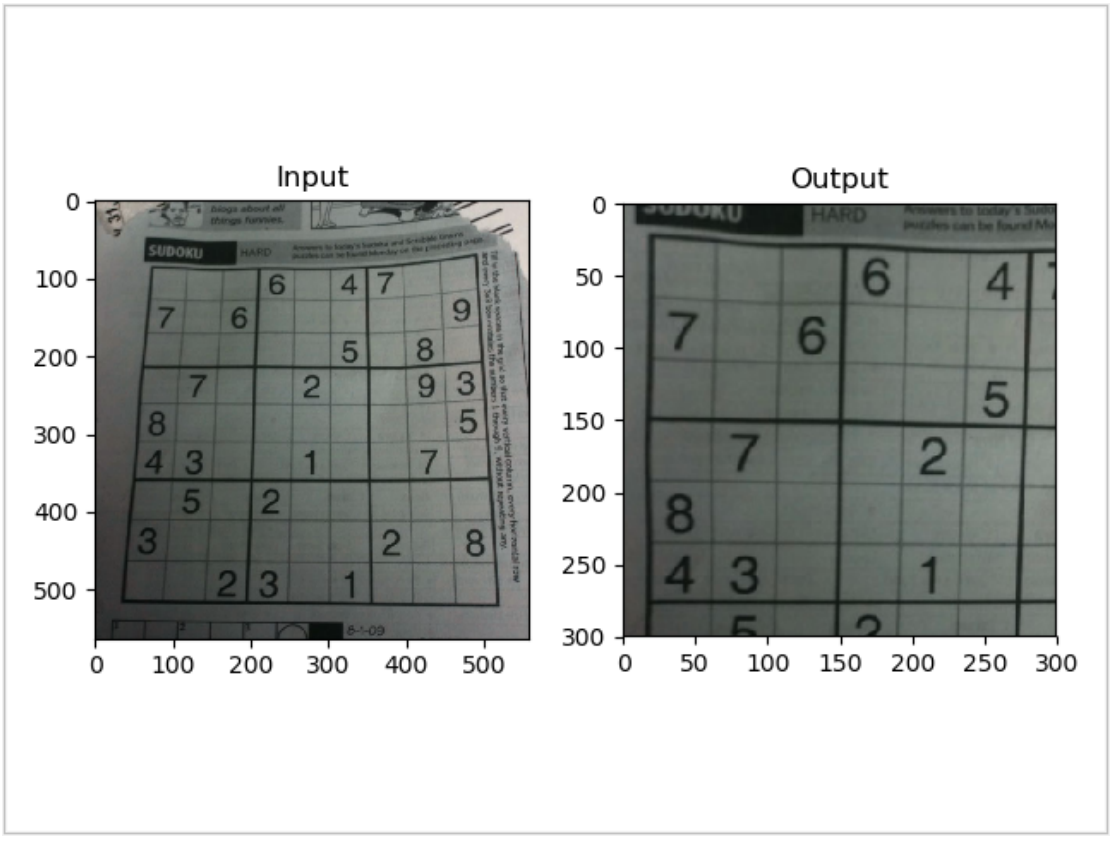
```
import cv2
import numpy as np
import cv2
import numpy as np

img=cv2.imread('news.jpg')
rows,cols,ch=img.shape
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])
M=cv2.getPerspectiveTransform(pts1,pts2)
dst=cv2.warpPerspective(img,M,(200,200))
cv2.imshow('Input',img)
cv2.imshow('Output',dst)
cv2.imwrite('getPerspectiveTransformImg.jpg',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

结果如下：

原图

透视变换后



四、实验图像

四、 实验报告要求

1 描述实验的基本步骤，用数据和图片给出各个步骤中取得的实验结果并进行必要的讨论。

2 必须包括原始图像及其计算处理后的图像以及相应的解释。

五、 思考题

通过鼠标操作在图像上选取任意 4 边形区域，通过仿射变换到指定的矩形区域。例如梯形到矩形的变换。

提示：应用场景，例如将手机拍摄的梯形图片，通过几何操作，变的方方正正。