《计算机视觉》实验报告

姓名：汪江豪 学号：22121630

实验二

一. 任务 1

读取一张图片，完成以下任务：

(1) 平移：x 轴平移 100 像素，y 轴平移 150 像素

(2) 缩放：缩放到 1024*768；按比例缩小（60%）

(3) 翻转：水平翻转，垂直翻转，水平+垂直翻转

(4) 旋转：给出旋转中心，旋转角度，对图片旋转

(5) 缩略：将图片缩小，放到原图的左上角

1. 核心代码：

(1) 平移：

```
def shift(img):
    rows, cols = img.shape[:2]
    M = np.float32([[1, 0, 100], [0, 1, 150]])
    img_shift = cv.warpAffine(img, M, (cols, rows))
    cv.imwrite('img_shift.jpg', img_shift)
```

(2) 缩放：

```
def scale(img):
    rows, cols = img.shape[:2]
    img_scale1 = cv.resize(img, (1024, 768), interpolation=cv.INTER_LINEAR)
    cv.imwrite('img_scale1.jpg', img_scale1)

    img_scale2 = cv.resize(img, None, fx=0.6, fy=0.6,
interpolation=cv.INTER_LINEAR)
    cv.imwrite('img_scale2.jpg', img_scale2)
```

(3) 翻转：

```
def flip(img):
    img_flip1 = cv.flip(img, 1) # >0 水平翻转
    cv.imwrite('img_flip_h.jpg', img_flip1)

    img_flip2 = cv.flip(img, 0) # =0 垂直翻转
    cv.imwrite('img_flip_v.jpg', img_flip2)

    img_flip3 = cv.flip(img, -1) # <0 水平+垂直翻转
    cv.imwrite('img_flip_hv.jpg', img_flip3)
```

(4) 旋转：

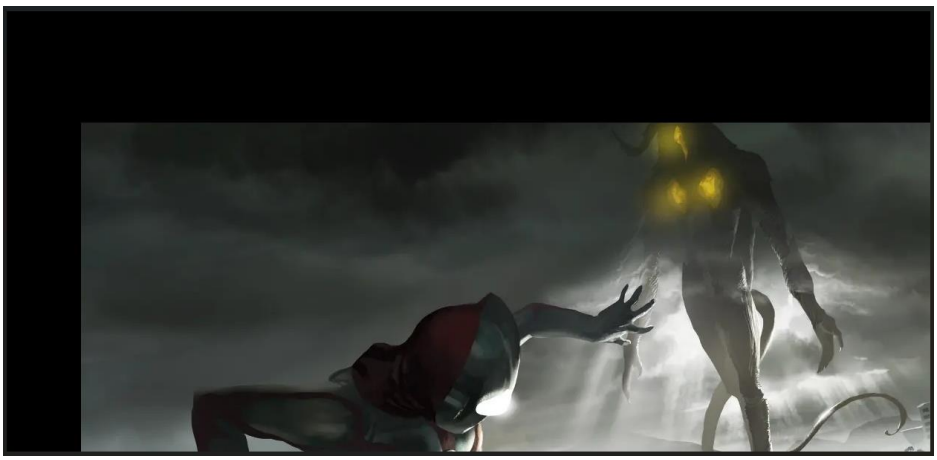
```
def rotate(img):
    rows, cols = img.shape[:2]
    M = cv.getRotationMatrix2D((cols/2, rows/2), 45, 1) # 旋转中心，旋转角度，缩放比例
    img_rotate = cv.warpAffine(img, M, (cols, rows))
    cv.imwrite('img_rotate.jpg', img_rotate)
```

(5) 缩略：

```
def thumbnail(img):
    rows, cols = img.shape[:2]
    img_thumbnail = cv.resize(img, (cols//2, rows//2), interpolation=cv.INTER_LINEAR)
    # 放到原图左上角
    img[0:rows//2, 0:cols//2] = img_thumbnail
    cv.imwrite('img_thumbnail.jpg', img)
```

2. 实验结果截图

(1) 平移：



(2) 缩放：

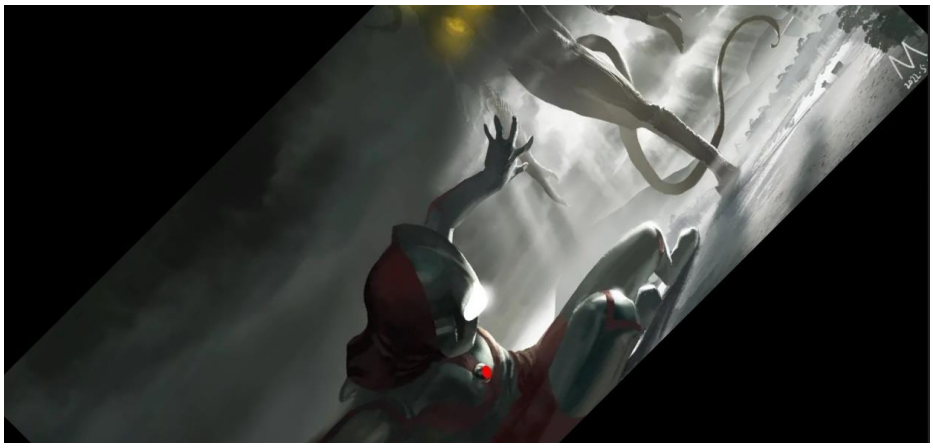
(3) 翻转：

水平：

垂直：

水平+垂直：

(4) 旋转：



(5) 缩略：

a) 实验小结

Opencv 库包含丰富的对图像进行旋转，缩放，平移等处理。对图像像素的处理和 python 中对二维数组的处理基本一致，而且频繁使用到切片操作和 shape 对象，shape[0]和 shape[1]分别表示图像的行和列。(1) 记住平移函数 warpAffine,缩放函数 resize,翻转函数 flip

3. 任务 2

1. 读取一张新的图片，将其转换为灰度图片；
2. 调整灰度图片为正方形（边长不小于 500 像素）；
3. 用圆形掩膜对图片进行切片，并保存切片后的图像（如图所示）

a) 核心代码：

```
def main():
    img = cv.imread('img.jpg')
    if img is None:
        print('图片读取失败')
```

```
        return
    # 转换为灰度图片
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    # 调整为正方形
    suqare_size = 500
    img_square = cv.resize(img_gray, (suqare_size, suqare_size),
interpolation=cv.INTER_LINEAR)
    # 画圆形掩膜,创建一个正方形黑幕，再画一个圆形掩膜
    mask = np.zeros((suqare_size, suqare_size), np.uint8)
    # 圆心，半径，颜色，-1 表示填充，返回值是一个圆形掩膜
    cv.circle(mask, (suqare_size//2, suqare_size//2), suqare_size//2, 255,
-1)
    # 与操作
    img_new = cv.bitwise_and(img_square, img_square, mask=mask)
    cv.imwrite('img_new.jpg', img_new)
```

b) 实验结果截图

c) 实验小结

Opencv 中的掩膜是一个很重要的方法。先创建一个黑幕矩阵，然后创建指定形状的

掩膜，指定像素值均为 255，再与原图像像素矩阵相与，bitwise_and 即可实现掩膜操作。