《计算机视觉》实验报告

姓名：王子棠 学号：21122897

实验 2 图像操作

一. 任务 1

a) 核心代码：

1.1 平移：x 轴平移 100 像素，y 轴平移 150 像素

```
# 获取图像尺寸
height, width, _ = img.shape
# 定义平移矩阵 M:[[1,0,x],[0,1,y]]; x:水平方向偏移量
M = np.float32([[1, 0, 100], [0, 1, 150]])
# 平移操作
img_py = cv2.warpAffine(img,M,(width,height))
```

1.2 缩放

1.2.1 缩放到 1024*768

```
img_sf1 = cv2.resize(img, (1024, 768))
```

1.2.2 按比例缩小 60%

```
img_sf2 = cv2.resize(img, (0, 0), fx=0.6, fy=0.6,
interpolation=cv2.INTER_NEAREST)
```

1.3 翻转

```
flip_horiz_img = cv2.flip(img, 1) # 水平翻转
flip_verti_img = cv2.flip(img, 0) # 垂直翻转
flip_horandver_img = cv2.flip(img, -1) # 水平垂直翻转
```

1.4 旋转

```
rows, cols, _ = img.shape
# 90 度旋转 以图像的中心为旋转中心
```

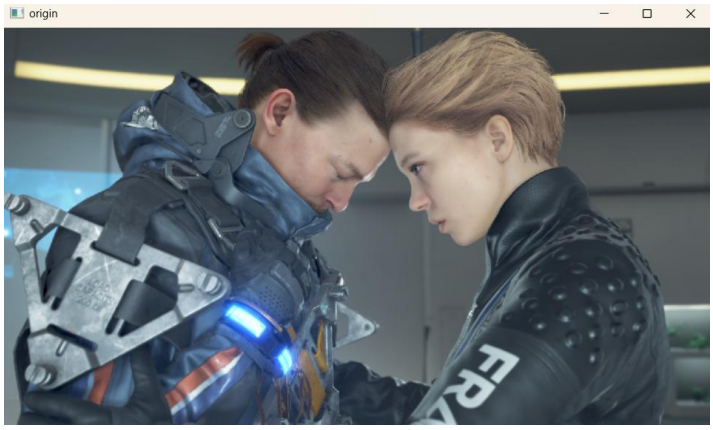
```
M = cv2.getRotationMatrix2D((cols/2, rows/2), 90, 1)
img_rotate = cv2.warpAffine(img, M, (cols,rows)
```

1.5 缩略：将图片缩小，放到原图的左上角

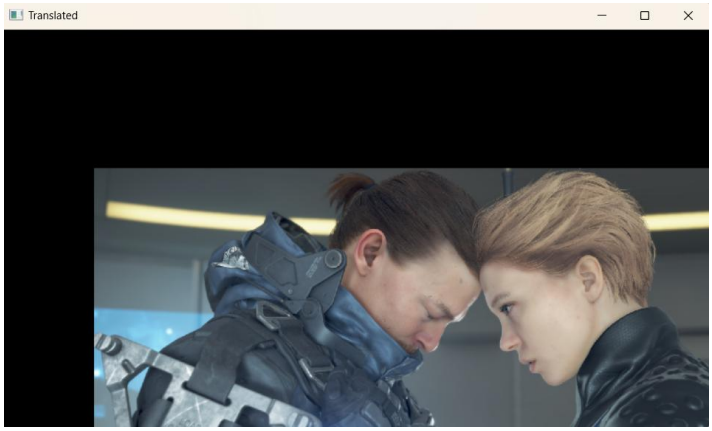
```
resized_img = cv2.resize(img, (0, 0), fx=0.4, fy=0.4,
interpolation=cv2.INTER_NEAREST ) # 缩小图片 40%
cv2.imwrite('small.png', resized_img) # 保存小图
height_, width_ = resized_img.shape[:2] # 获取小图的尺寸
img= Image.open("image2.png") # 原图
im_crop=Image.open("small.png") # 缩略图
img.paste(im_crop, (0,0)) # 括号中为左上角坐标，将小图叠加到原图上
```

b) 实验结果截图

1.1 平移

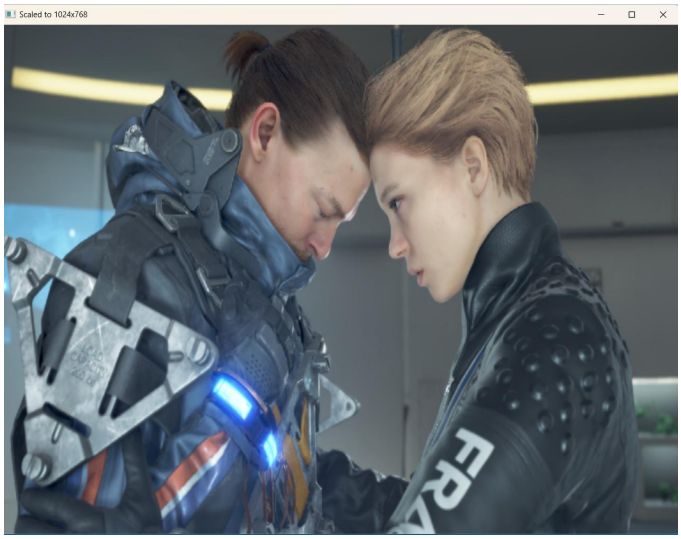


原图

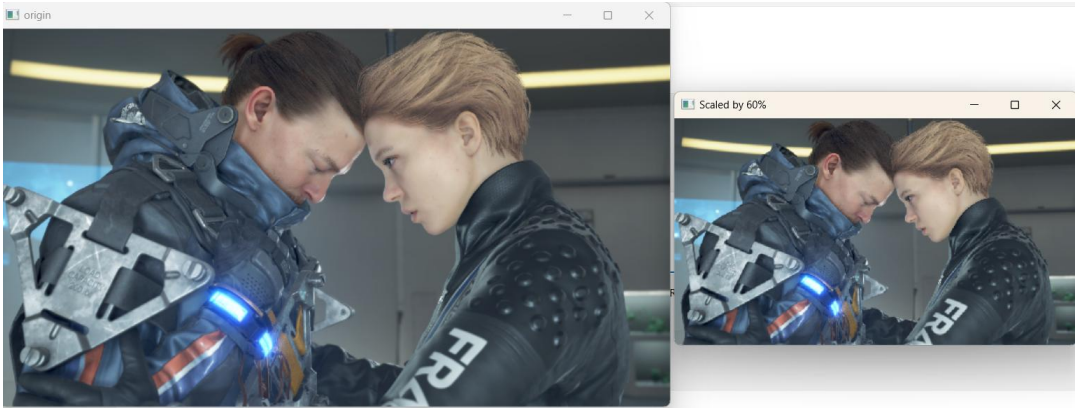


平移后的图像

1.2.1 缩放至 1023*768



1.2.2 按比例缩小 60%

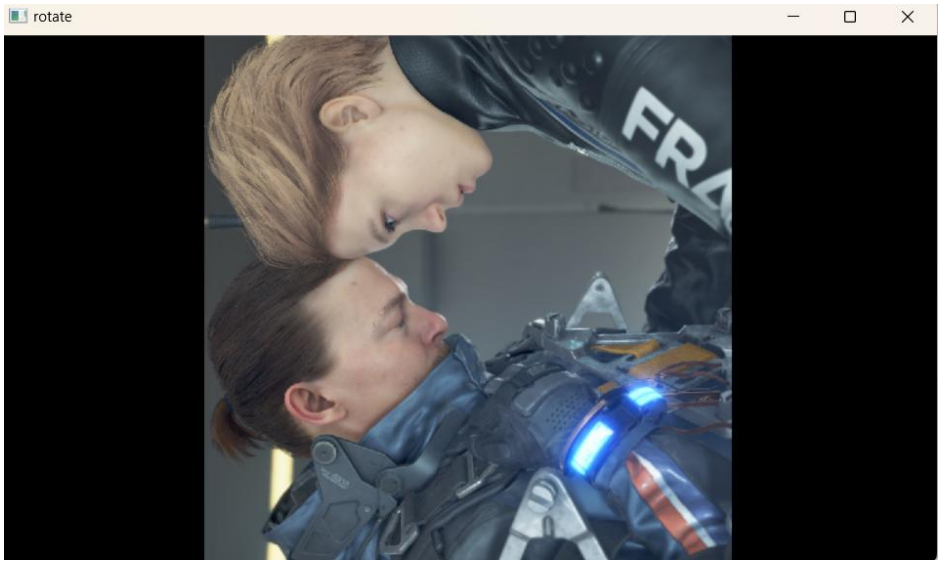


左边为原图，右边为缩小的图

1.3 翻转

左上为水平翻转，右上为垂直翻转，左下为水平垂直翻转，右下为原图

1.4 旋转



1.5 缩略

c) 实验小结

本次实验我学会了对图像的平移、缩放、旋转、翻转以及叠加的操作，这些操作主要都用到了 opencv 里的函数。对于缩略图的添加，我使用了 PIL 库里的 Image.paste(image, box, mask=None)，可以直接把缩小的图粘贴在原图上，比 opencv 更加方便简单。

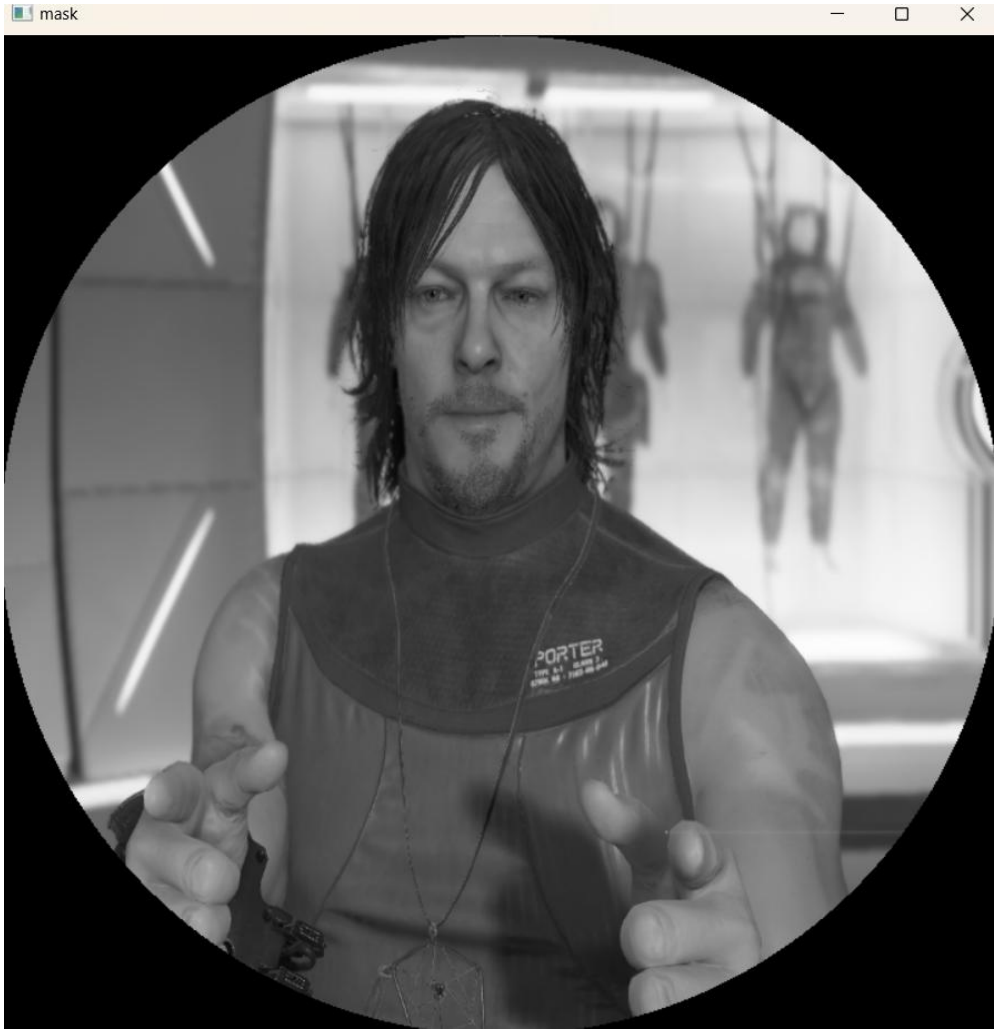
二. 任务 2

a) 核心代码：

```
image = cv2.imread('image.png')
# 将图像转换为灰度图像
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 将图像调整为正方形，720*720 大小
```

```
image_square = cv2.resize(gray_image, (720, 720)) int main() {
# 获取正方形图像的尺寸
height, width = image_square.shape[:2]
# 创建全黑掩膜，掩膜的遮蔽区域为黑色，非遮蔽区域为白色
mask = np.zeros((height, width), dtype=np.uint8)
# 在正方形掩膜上画圆形，center 为圆形的圆点，radius 为半径
center = (width//2, height//2)
radius = width//2
# 在全黑掩膜上画白色的圆形，形成圆形掩膜
cv2.circle(mask, center, radius, (255, 255, 255), -1)
# cv2.add 进行掩膜与灰度图的加法运算，对被掩模图像遮蔽的黑色区域不进行处理，保持黑色，非黑色区域进行加法运算
imgAddMask = cv2.add(image_square, np.zeros(np.shape(image_square), dtype=np.uint8), mask=mask)
```

b) 实验结果截图



c) 实验小结

这个实验首先需要把图片转为灰度图，接着调整成正方形的大小。在创建掩码后使用 cv2.add 进行掩码与图像的叠加。我学习到在掩膜图像与其他图像叠加中，黑色区域仍然为黑色，非黑色区域将与其他图像进行加法运算。