《计算机视觉》实验报告

姓名：严昕宇 学号：20121802

实验 2

一. 任务 1

a) 核心代码：

第一题

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

[1]:
```python
# cv2.warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])
# 首先定义平移矩阵Matrix，再调用warpAffine()函数实现平移
import cv2
import numpy as np

Img_Path = "./Mouse.png"
Mouse_Img = cv2.imread(Img_Path, cv2.IMREAD_COLOR)
# 原图的高、宽
rows, cols, channel = Mouse_Img.shape

dx, dy = 100, 150  # dx=100 向右偏移量, dy=150 向下偏移量
Matrix = np.float32([[1, 0, dx], [0, 1, dy]])  # 构造平移变换矩阵
New_MouseImg = cv2.warpAffine(Mouse_Img, Matrix, (cols, rows))  # 默认为黑色填充

cv2.imshow("Warp", New_MouseImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

(2) 缩放：缩放到1024*768；按比例缩小 (60%)

[2]:
```python
# cv2.resize(src, dsize[, result[. fx[, fy[, interpolation]]]])
# src表示原始图像;dsize表示缩放大小，fx和fy表示缩放倍数，dsize或fx/fy设置一个即可实现图像缩放
Resize_MouseImg = cv2.resize(Mouse_Img, (1024, 768))
Resize_MouseImg1 = cv2.resize(Resize_MouseImg, None, fx=0.6, fy=0.6)

cv2.imshow("Resize", Resize_MouseImg1)

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

(3) 翻转：水平翻转，垂直翻转，水平+垂直翻转

[3]:
```
# cv2.flip(src, flipCode)
# src表示原始图像；flipCode表示翻转方向，如果flipCode为0，则以X轴为对称轴翻转
# >0则以Y轴为对称轴翻转，如果flipCode<0则在X轴、Y轴方向同时翻转

# 图像翻转
# 0以X轴为对称轴翻转 >0以Y轴为对称轴翻转 <0以X轴Y轴翻转
MouseImg_Flip1 = cv2.flip(Mouse_Img, 1)
MouseImg_Flip2 = cv2.flip(Mouse_Img, 0)
MouseImg_Flip3 = cv2.flip(Mouse_Img, -1)

cv2.imshow("Flip1", MouseImg_Flip1)
cv2.imshow("Flip2", MouseImg_Flip2)
cv2.imshow("Flip3", MouseImg_Flip3)

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

(4) 旋转：给出旋转中心，旋转角度，对图片旋转

[4]:
```
# 图像旋转主要调用getRotationMatrix2D()函数和warpAffine()函数实现
# cv2.getRotationMatrix2D(旋转中心, 旋转度数, scale)
# cv2.warpAffine(原始图像, 旋转参数, 原始图像宽高)

# 原图的高、宽 以及通道数
rows, cols, channel = Mouse_Img.shape

# 绕图像的中心旋转
# 参数：旋转中心 旋转度数，scale
Matrix1 = cv2.getRotationMatrix2D((cols / 2, rows / 2), 30, 1)

# 参数：原始图像 旋转参数 元素图像宽高
MouseImg_Rotated = cv2.warpAffine(Mouse_Img, Matrix1, (cols, rows))

cv2.imshow("Rotated", MouseImg_Rotated)

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

(5) 缩略：将图片缩小，放到原图的左上角

[5]:
```
Small_MouseImg = cv2.resize(Mouse_Img, None, fx=0.3, fy=0.3)
# 缩小后的图片大小为(90,160)
Mouse_Img[0:90, 0:160] = Small_MouseImg

cv2.imshow('SmallAddImg', Mouse_Img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

b) 实验结果截图

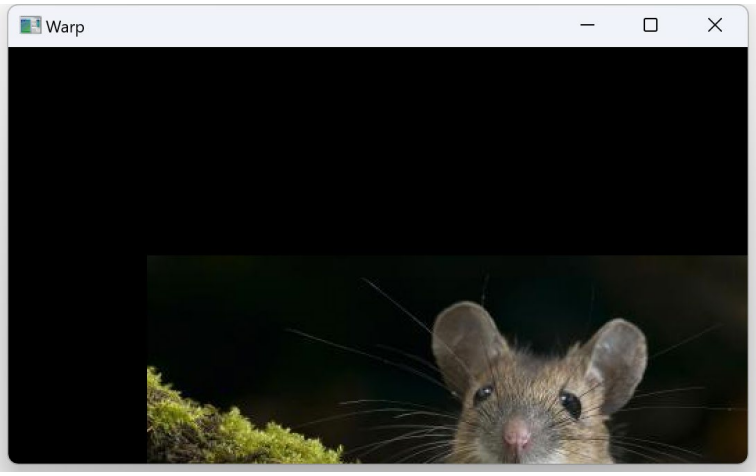

图 1 平移结果

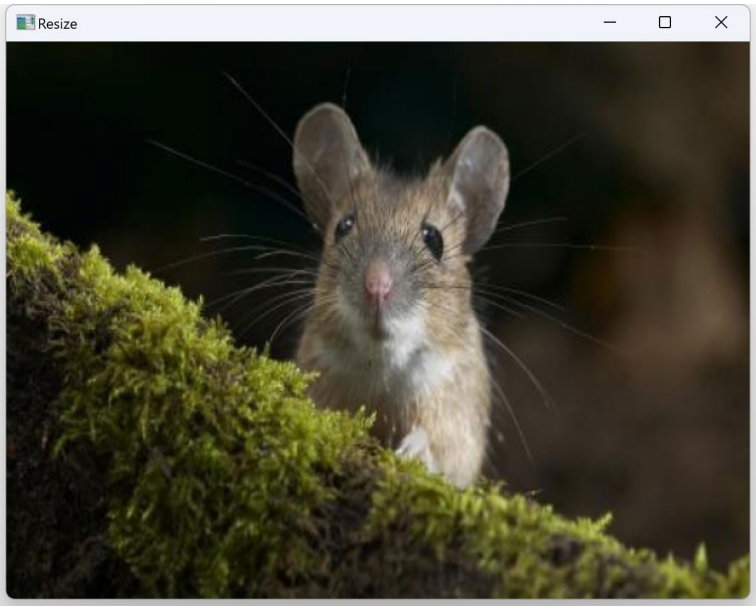

图 2 缩放结果

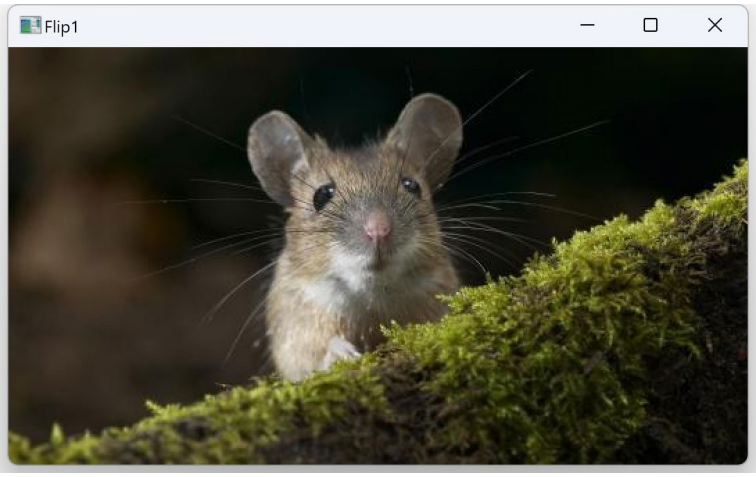

图 3 翻转结果-水平翻转

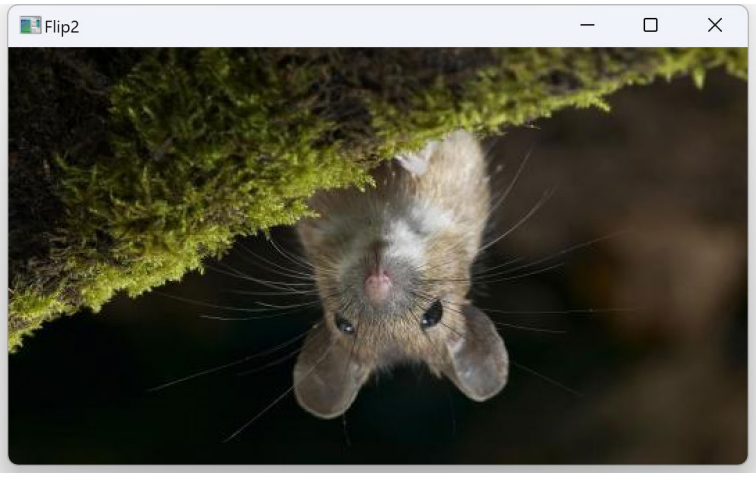

图 4 翻转结果-垂直翻转

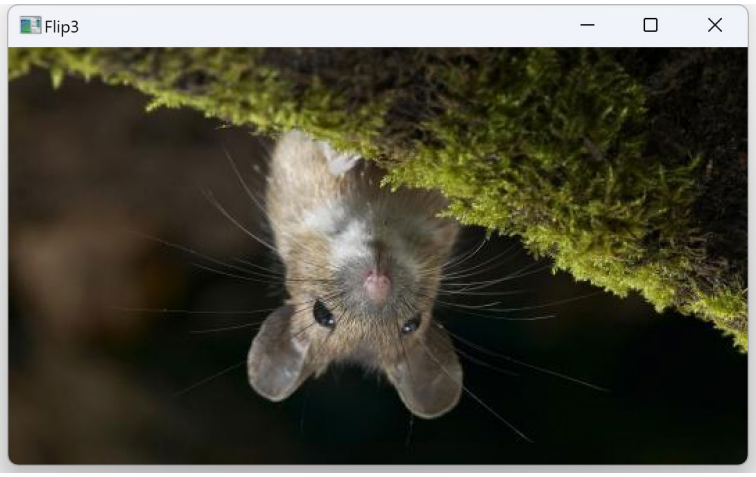

图 5 翻转结果-水平+垂直翻转

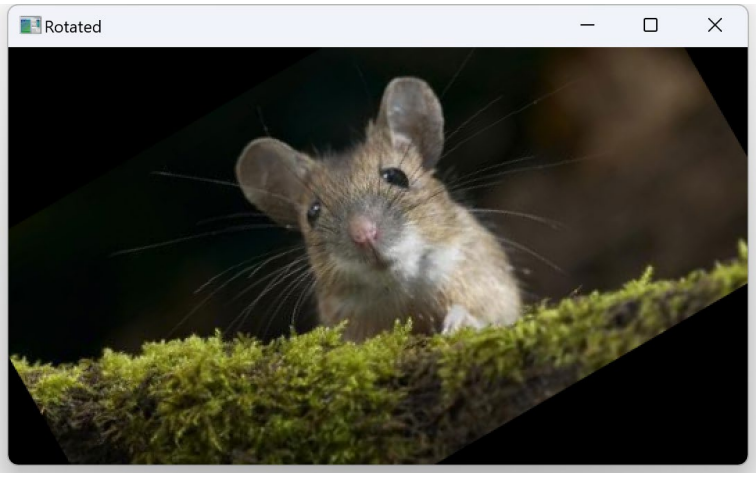

图 6 旋转结果

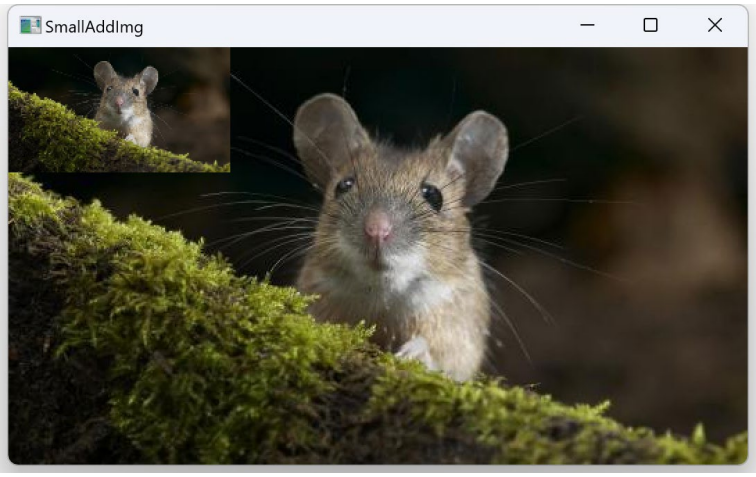

图 7 缩略结果

c) 实验小结

图像的几何变换主要包括：平移、缩放、旋转、仿射、透视等。图像变换是建立在矩阵运算基础上的，通过矩阵运算可以很快的找到不同图像的对应关系。理解变换的原理需要理解变换的构造方法以及矩阵的运算方法。

二. 任务 2

a) 核心代码：

第二题

1. 读取一张新的图片，将其转换为灰度图片

方法一：读取图片候直接读取为灰度图

[6]:
```python
import cv2

Img_Path = "./Leopard.png"
Leopard_GrayImg = cv2.imread(Img_Path, cv2.IMREAD_GRAYSCALE)

cv2.imshow("Gray Leopard", Leopard_GrayImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

方法二：使用cv2.cvtColor()函数将彩色图转为灰度图

[7]:
```python
import cv2

Img_Path = "./Leopard.png"
Leopard_Img = cv2.imread(Img_Path)

# cv2.cvtColor(p1,p2)是颜色空间转换函数，p1是需要转换的图片，p2是转换成何种格式。
# cv2.COLOR_BGR2RGB将BGR格式转换成RGB格式
# cv2.COLOR_BGR2GRAY将BGR格式转换成灰度图片
Leopard_GrayImg = cv2.cvtColor(Leopard_Img, cv2.COLOR_RGB2GRAY)

cv2.imshow("Gray Leopard", Leopard_GrayImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. 调整灰度图片为正方形 (边长不小于500像素)

[8]:
```
# 函数原型：cv.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])
# fx和fy为x和y方向上的缩放比例，interpolation为插值方式
Leopard_NewImg = cv2.resize(Leopard_GrayImg, dsize=(500, 500))
cv2.imshow("New Gray Leopard", Leopard_NewImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

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)

[9]:
```
import numpy as np

# 返回与图像尺寸相同的全零数组
Mask = np.zeros((Leopard_NewImg.shape[0], Leopard_NewImg.shape[1]), dtype=np.uint8)
# circle函数原型：cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]])
#center圆心位置；radius圆的半径；thickness圆形轮廓的粗细（如果为正）负厚度表示要绘制实心圆
cv2.circle(Mask, (250, 250), 250, (255, 255, 255), -1)  # -1 表示实心
# 提取圆形ROI
Img_AddMask = cv2.add(Leopard_NewImg, np.zeros(np.shape(Leopard_NewImg), dtype=np.uint8), mask=Mask)

# 显示掩模加法结果 imgAddMask1
cv2.imshow("Img_AddMask", Img_AddMask)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

b) 实验结果截图

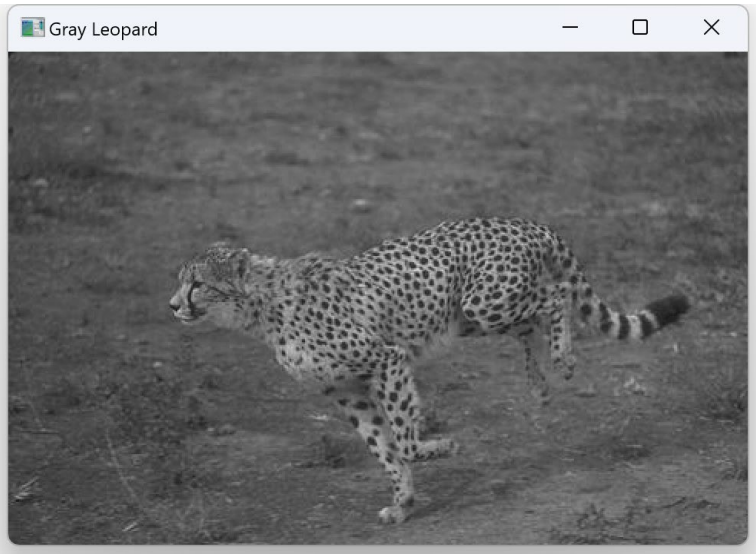

图 8 灰度图片

图 9 调整后的正方形图

图 10 切片后的图片

c) 实验小结

在半导体制造中，许多芯片工艺步骤采用光刻技术，用于这些步骤的图形“底片”称为掩膜，其作用是：在硅片上选定的区域中对一个不透明的图形模板遮盖，继而下面的腐蚀或扩散将只影响选定的区域以外的区域。

图像掩膜与其类似，用选定的图像、图形或物体，对处理的图像（全部或局部）进

行遮挡，来控制图像处理的区域或处理过程。用于覆盖的特定图像或物体称为掩模或模板。数字图像处理中，掩模为二维矩阵数组,有时也用多值图像。