实验 1 图像操作和处理基础实验

编写代码完成以下实验内容，可将代码截图插入到问题下方。

第一部分 PIL

1、 利用 PIL 中的函数读取图像"empire.jpg", 并将其显示出来。再将其颜色转换为灰度，并显示出来。

```
'''T1.1'''
pil_im1 = Image.open('empire.jpg')
pil_im1.show()
pil_im2 = Image.open('empire.jpg').convert('L')
pil_im2.show()
```

2、 编写函数 get_imlist(path)，用来创建一个包含文件夹中所有".bmp"图像文件的文件名列表。然后，再将该文件名列表中的文件转换为 JPEG 文件。

```
'''T1.2'''


def get_imlist(path):
    return [os.path.join(path, tmp) for tmp in os.listdir(path) if tmp.endswith('.bmp')]

filelist = get_imlist('.')

for infile in filelist:
    outfile = os.path.splitext(infile)[0] + ".jpg"
    if infile != outfile:
        try:
            Image.open(infile).save(outfile)
        except IOError:
            print("cannot convert", infile)
```

3、 创建图像"empire.jpg"的缩略图，并将其显示出来。

```
35  '''T1.3'''
36  pil_im3 = Image.open('empire.jpg')
37  pil_im3.thumbnail((128, 128))
38  pil_im3.show()
```

4、 裁剪图像"empire.jpg"的指定区域的图像，四元组坐标依次为(100,100,400,400)。裁

剪后将其旋转后再贴回原图像，并显示该图像。

```
'''T1.4'''
pil_im4 = Image.open('empire.jpg')
box = (100, 100, 400, 400)
region = pil_im4.crop(box)
region = region.transpose(Image.ROTATE_180)
pil_im4.paste(region, box)
pil_im4.show()
```

5、 将图像"empire.jpg"尺寸调整为 128*128，显示该图像。继续将该图像旋转 45 度，显示该图像。

```
'''T1.5'''
pil_im5 = Image.open('empire.jpg')
out = pil_im5.resize((128, 128))
out.show()
out = out.rotate(45)
out.show()
```

第二部分 Matplotlib

1、 读取图像“empire.jpg”。已知四个点的坐标为(100,200)，(100,500)，(400,200)，(400,500)。使用红色星状标记在图中绘制这四个点，并绘制连接前两个点的直线。然后向图像中增加一个标题。最后将图像显示出来。

```
'''T2.1'''
im = array(Image.open('empire.jpg'))
imshow(im)
x = [100,100,400,400]
y = [200,500,200,500]
plot(x, y, 'r*')
plot(x[:2],y[:2])
title('Plotting: "empire.jpg"')
show()
```

2、 绘制图像“empire.jpg”的轮廓和直方图。

```
'''T2.2'''
im = array(Image.open('empire.jpg').convert('L'))
figure()
gray()
contour(im, origin='image')
axis('equal')
axis('off')
figure()
hist(im.flatten(),128)
show()
```

第三部分 NumPy

1、 将图像“empire.jpg”转换为数组对象，并打印其大小和元素数据类型。将图像“empire.jpg”的灰度图转换为数组对象，并指定其元素数据类型为浮点型，打印其大小和元素数据类型。

```
'''T3.1'''
im = array(Image.open('empire.jpg'))
print(im.shape, im.dtype)
im2 = array(Image.open('empire.jpg').convert('L'), 'f')
print(im2.shape, im2.dtype)
```

2、 对图像“empire.jpg”的灰度图像进行如下变换：（1）对图像进行反相处理。（2）将图像的像素值变换到 100…200 区间。（3）对图像的每个像素值进行平方运算。输出三种变换的图像结果。

```
'''T3.1'''
im = array(Image.open('empire.jpg'))
print(im.shape, im.dtype)
im2 = array(Image.open('empire.jpg').convert('L'), 'f')
print(im2.shape, im2.dtype)

'''T3.2'''
im3 = 255 - im2
figure()
imshow(im3)
im4 = (100.0 / 255) * im + 100
figure()
imshow(im4)
im5 = 255.0 * (im / 255.0) ** 2
figure()
imshow(im5)
show()
```

3、 编写一个用于图像缩放的函数 imresize(im,sz), 其中 im 为图像, sz 为缩放后的大小, 返回值为缩放后的图像数组表示。用该函数对图像“empire.jpg”缩放处理为大小 100*100。查看缩放后的图像大小。

```
'''T3.3'''


def imresize(im, sz):
    pil_im = Image.fromarray(uint8(im))
    return array(pil_im.resize(sz))


im = Image.open('empire.jpg')
a = imresize(im, (100, 100))
print(a.shape, a.dtype)
```

4、 编写函数 histeq(im,nbr_bins=256)，用来实现图像灰度变换中的直方图均衡化，即：将一幅图像的灰度直方图变平，使变换后的图像中每个灰度值的分布概率都相同。通过该函数将图像“AquaTermi_lowcontrast.jpg”进行直方图均衡化，并将原图和变换后图分别输出。

```
'''T3.4'''


def histeq(im, nbr_bins=256):
    imhist, bins = histogram(im.flatten(), nbr_bins, normed=True)
    cdf = imhist.cumsum()
    cdf = 255 * cdf / cdf[-1]
    im2 = interp(im.flatten(), bins[:-1], cdf)
    return im2.reshape(im.shape), cdf

im = Image.open('AquaTermi_lowcontrast.jpg').convert('L')
im.show()
im2, CDF = histeq(array(im))
figure()
imshow(im2)
show()
```

5、 图像平均操作是一种减少图像噪声的方式。编写函数 compute_average(imlist)，实现从图像列表 imlist 中计算出一幅平均图像，假设所有图像的大小一样。可以将这些图像相加，然后除以图像的数目，从而得到平均图像。

```
'''T3.5'''


def compute_average(imlist):
    averageim = array(Image.open(imlist[0]), 'f')
    for imname in imlist[1:]:
        try:
            averageim += array(Image.open(imname))
        except:
            print(imname + '...skipped')
    averageim /= len(imlist)
    return array(averageim, 'uint8')
```

第四部分 SciPy

1、 图像模糊就是将灰度图像 I 和一个高斯核进行卷积操作。高斯模糊通常是其它图像处理操作的一部分，比如图像插值操作、兴趣点计算等。SciPy 有用来做滤波操作的 scipy.ndimage.filters 模块，其中的函数 gaussian_filter()使用快速一维分离的方式来计算卷积。该函数需要指定标准差，随着标准差的增加，一幅图像被模糊的程度变大，丢失的细节越多。编写代码对图像“empire.jpg”进行高斯模糊处理，分别将

标准差设置为 2、5、10，并将原图与处理后的模糊图分别输出。

```
'''T4.1'''
im = Image.open('empire.jpg')
im.show()
im = array(Image.open('empire.jpg').convert('L'))
im2 = filters.gaussian_filter(im, 2)
figure()
imshow(im2)
im3 = filters.gaussian_filter(im, 5)
figure()
imshow(im3)
im4 = filters.gaussian_filter(im, 10)
figure()
imshow(im4)
show()
```

2、 图像强度的变化是非常重要的信息。强度的变化可以用灰度图像的 x 和 y 方向导数来进行描述。图像导数大多数可以通过卷积来实现，可以使用 scipy.ndimage.filters 中的标准卷积操作来实现导数滤波器。编写代码使用 Sobel 滤波器来计算图像“empire.jpg”的 x 和 y 方向的导数和梯度大小，并将相应的 x 导数图像、y 导数图像和梯度图像分别输出。

```
'''T4.2'''
im = array(Image.open('empire.jpg').convert('L'))
imx = zeros(im.shape)
filters.sobel(im,1,imx)
figure()
imshow(imx)
imy = zeros(im.shape)
filters.sobel(im,0,imy)
figure()
imshow(imy)
magnitude = sqrt(imx**2+imy**2)
figure()
imshow(magnitude)
show()
```

3、 形态学是度量和分析基本形状的图像处理方法的基本框架与集合，通常用于处理二

值图像。二值图像通常是在计算物体的数目或者度量其大小时，对一幅图像进行阈值化后的结果。scipy.ndimage 中的 morphology 模块可以实现形态学操作，其中的 measurements 模块可以实现二值图像的计数和度量功能。编写代码计算图像“houses.png”中对象的个数，并将标签图像输出。由于图像中不少对象存在连接的情况，请通过二进制开操作将其移除，并重新统计图像中对象的个数，并将标签图像输出。

```
'''T4.3'''
im = array(Image.open('houses.png').convert('L'))
im = 1 * (im < 128)
labels, nbr_objects = measurements.label(im)
print("Number of objects1:", nbr_objects)
figure()
imshow(labels)
im_open = morphology.binary_opening(im,ones((9,5)),iterations=2)
labels_open, nbr_objects_open = measurements.label(im_open)
figure()
imshow(labels_open)
print("Number of objects2:", nbr_objects_open)
show()
```