图像处理与可视化 HW1

赵心怡 19307110452

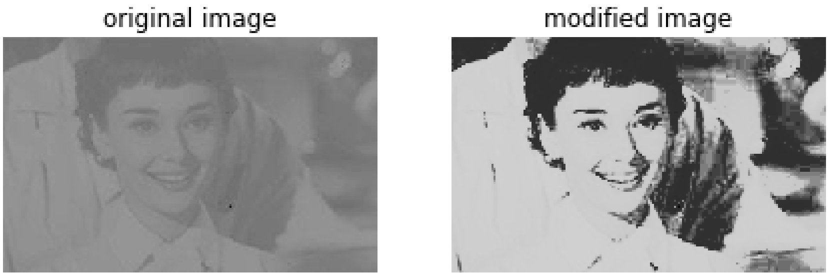
2022.3.21

1.

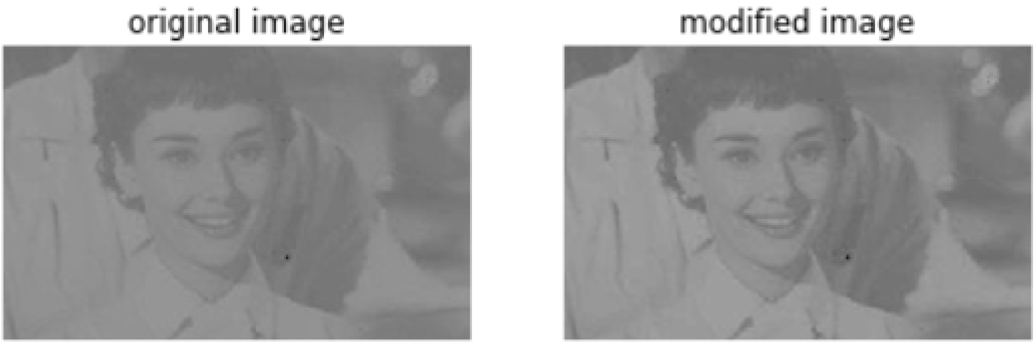
1. Implement a piecewise linear transformation function for image contrast stretching. The code should read in an image; for intensity of all pixels, use the function to compute new intensity values; and finally output/ save the image with new intensity values.

```python
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from skimage import io
#读入图像
PATH = 'herpen2.jpg'
img = Image.open(PATH).convert('L')
img = np.array(img)

rows,cols=img.shape[:2]

plt.subplot(121)
plt.title('original image')

io.imshow(img)
plt.axis('off')

#可以修改线性函数的参数达到更好的效果。
r2=130
r1=120
s1=50
s2=200
img2=img

#灰度变换
for i in range(rows):
    for j in range(cols):
        if (img[i,j]<=r1):
            img2[i,j]= img[i,j]*s1/r1
        elif  (img[i,j]>=r2):
            img2[i,j]= (img[i,j]-r2)*(255-s2)/(255-r2)+s2
        else :
            img2[i,j]= (img[i,j]-r1)*(s2-s1)/(r2-r1)+s1

plt.subplot(122)
plt.title('modified image')
# myIm = Image.fromarray(img)
io.imshow(img2)#可视化图像
plt.axis('off')
```

r=(120,130):



r=(70,180):



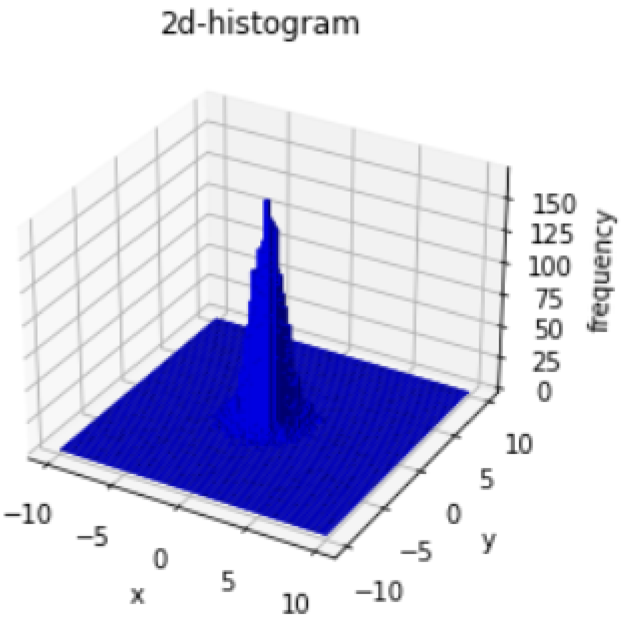
讨论： 当选取的图片初始的对比度高的时候，灰度值的拉伸效果并不明显。在选取的图像对比度很低时才能有较大影响。

调整不同的r值会让图片的灰度值拉伸效果有所不同，r值越接近对比度越高，相差越大对比度越低

2

2. (1) implement n-dimensional joint histogram and test the code on two-dimensional data; plot the results. (2) implement computation of local histograms of an image using the efficient update of local histogram method introduced in local histogram processing.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def n_d_hist(data, dim, numBin, max_range, min_range):
    # 生成直方图的函数
    gridsize = tuple([numBin[i] for i in range(dim)]) # 统计各维度bin的数量
    hist = np.zeros(gridsize)
    binSize = (max_range - min_range) / numBin # bin的宽度
    for each_data in range(np.size(data, 0)):
        index_bin = np.floor((data[each_data] - min_range) / binSize) # 判断落在哪个区间内
        index_bin = np.maximum(index_bin, 0)
        index_bin = np.minimum(index_bin, numBin - 1)
        hist[tuple(index_bin.astype(np.int))] += 1
    return hist

np.random.seed(1234)
data = np.random.randn(10000, 2) # 生成2元正态分布数据
numBin = np.array([64, 64])
max_range, min_range = np.array([10, 10]),   np.array([-10, -10])
histogram = n_d_hist(data, 2, numBin, max_range, min_range)
# 绘图
xsize = (max_range[0] - min_range[0])/numBin[0]
ysize = (max_range[1] - min_range[1])/numBin[1]
x_coord, y_coord = np.meshgrid(np.arange(min_range[0], max_range[0], xsize),np.arange(min_range[1], max_range[1], ysize))
z_coord = histogram
x_coord, y_coord ,z_coord = x_coord.ravel(), y_coord.ravel(),z_coord.ravel() # 生成坐标等需要的数据

fig = plt.figure()
ax = fig.gca(projection='3d')

ax.bar3d(x_coord, y_coord, np.zeros_like(x_coord), xsize, ysize, z_coord,color='b')
ax.set_title('2d-histogram')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('frequency')
plt.show()
```



总体逻辑是将点的分布分到一个个bin中，然后统计bin的个数画二维直方图。

3.2

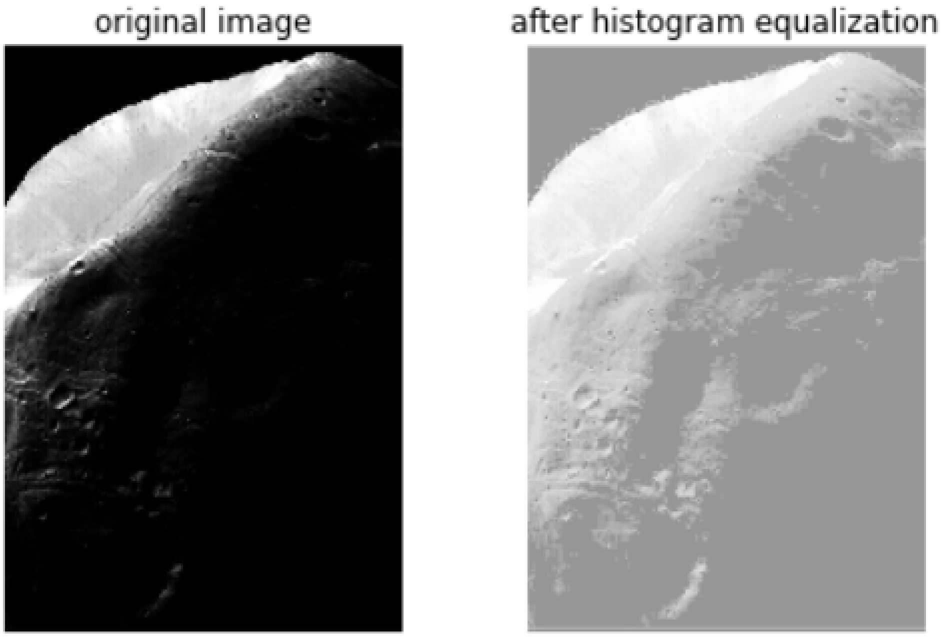
```
def update_local_hist(img, k, old_hist, center_idx):
    '''
    input: img: m*n array
    k: batch length
    old_hist: previous hist generated: Counter
    center_idx: previous batch center
    '''
    (x, y) = center_idx
    center_length = (k-1)//2
    if center_idx==(center_length,center_length): # 第一个
        return Counter(img[:k, :k].reshape(-1))
    if (x - center_length) % 2 == 0:
        if y == center_length: # 下移
            return (old_hist + Counter(img[x + center_length, :k]) - \
                Counter(img[x - 1 - center_length, :k]))
        else: #继续移动
            return old_hist + Counter(img[(x - center_length):(x + center_length + 1), y + center_length]) -\
                 Counter(img[(x - center_length):(x + center_length + 1), y - 1 - center_length])
    pic_width = img.shape[1]
    # (x - center_length) % 2 ==1 的情况
    if y == pic_width - center_length - 1: #下移
        return old_hist + Counter(img[x + center_length, pic_width - k:]) -\
             Counter(img[x - 1 - center_length, pic_width - k:])
    else: # 否则继续移动
        return old_hist + Counter(img[(x - center_length):(x + center_length + 1), y - center_length]) -\
             Counter(img[(x - center_length):(x + center_length + 1), y + center_length + 1])
```

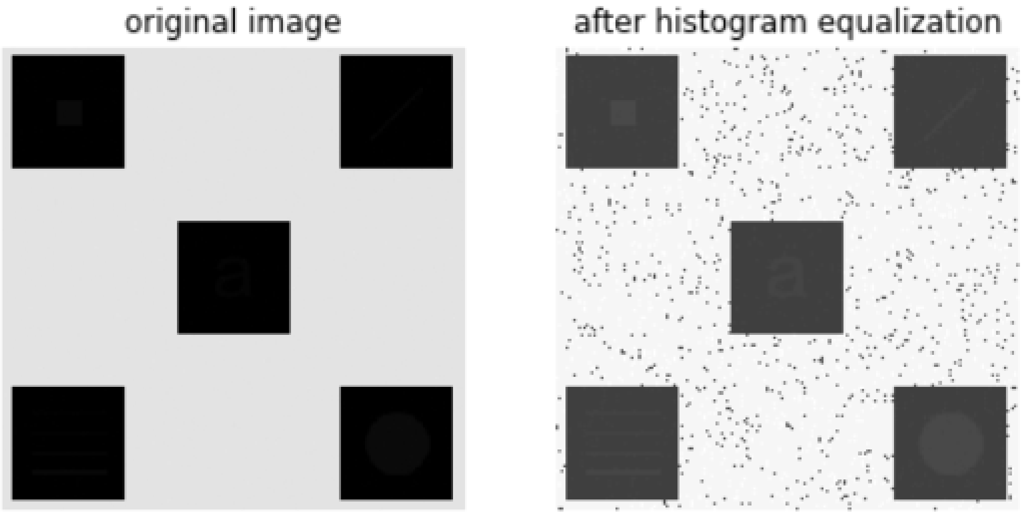
更新局部直方图均衡化的函数，在3.2中会用到这个函数，因此不作过多赘述。

3

3. Implement the algorithm of local histogram equalization:
(1) first implement histogram equalization algorithm

```
PATH = 'medical.jpg'
img = Image.open(PATH).convert('L')
img = np.array(img)
plt.subplot(121)
rows,cols=img.shape[:2]

plt.subplot(121)
plt.title('original image')

io.imshow(img)
plt.axis('off')

(m,n)=img.shape;
count=np.zeros(256);
for i in range(m):
    for j in range(n):
        count[img[i,j]]+=1
count=count/(m*n)
for i in range(1,256):
    count[i]=count[i-1]+count[i]
count=np.uint8(count*255);
for i in range(m):
    for j in range(n):
        img[i,j]=count[img[i,j]]
plt.subplot(122)
io.imshow(img)
plt.title('after histogram equalization')
plt.axis('off')
```



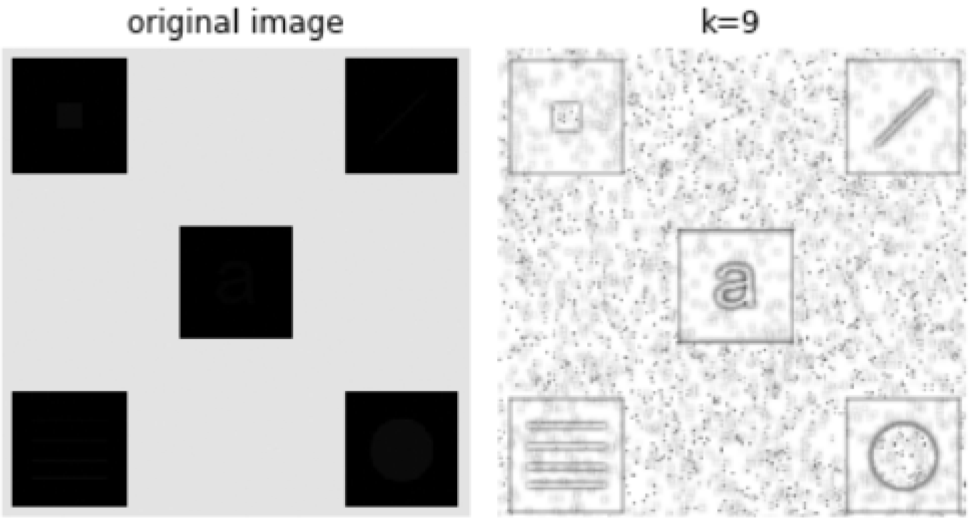


分析：直方图均衡化使得图片中对比度较低的地方得到了改善，但是由于灰度值较深的地方较多，整个图像有过曝问题，出现了一定比例的失真。

同时看到在灰度值分布在黑白两端的图像，直方图均衡化也无法将整个图像拉均匀。人眼依然无法看清黑色方块中的信息。直方图均衡化有一定的局限性。

(2) implement the local histogram equalization using efficient computation of local histogram. Please test your code on images and show the results in your report.

```
def localHist(img,k):
    """
    input: img: k*k*3 array
            k: param
    output: histogram
    """
    (m,n)=img.shape
    center=(k-1)//2
    temp_img=np.zeros((m+k-1,n+k-1),int)
    new_img=np.zeros((m+k-1,n+k-1),int)
    temp_img[center : m + center, center : n + center] = img
    local_hist = Counter()
    #小窗进行z字移动
    for x in range(center, m + center):
        visit_seq = range(center, n + center)
        if (x - center) % 2:
            visit_seq = reversed(visit_seq)#如果是奇数则反过来
        for y in visit_seq:
            local_hist = update_local_hist(temp_img,k, local_hist,(x, y)) # 2.2的函数更新直方图信息
            center_value = temp_img[x,y]
            cdf = np.zeros(center_value + 1)
            for i in range(center_value + 1):
                cdf[i] = cdf[i - 1] + local_hist[i]
            cdf = np.round(255 * cdf / (k**2))
            new_img[x, y] = cdf[temp_img[x , y]] # 进行转换
    new_img = new_img[center:m - center, center: n - center]
    return new_img
```



分析：通过局部直方图均衡化可以让原本黑色的部分的信息更清晰的显示出来，比上题中的全局均衡化效果更优秀。