

# 作业4

## 1、为什么一般情况下对离散图像的直方图均衡并不能产生完全平坦的直方图？

因为灰度级是离散的，同一灰度级的像素不能映射到不同的灰度级上，所以无法产生每个灰度级的值都一样的直方图。

## 2、设已用直方图均衡化技术对一幅图像进行了增强，试证明再用这个方法对所得结果增强并不会改变其结果

直方图均衡化所用的变换函数为原始图像的累积直方图，均衡化后得到的增强图像的累积直方图除有些项合并外，其余项与原始图像的累积直方图相同。再次进行均衡化，所用的是均衡化后的增强图像的累积直方图(并且不会有新的合并项)，所以不会改变其结果。

Prof:

第一次：第一次概率直方图为 $s(j)$ ，计算累计直方图： $t(j) = \sum_{j=0}^k s(j)$ ，映射后灰度：

$p(j) = \text{int}[(N-1)t(j) + 0.5]$ ，灰度映射关系： $k- > p(k)$

第二次：概率直方图为 $n(k)$ ，

$$n_k = \sum_{j=a_k}^{b_k} s_j, (a_k \leq j \leq b_k, \forall j, p_j = k)$$

计算累计直方图： $m(k) = \sum_{j=0}^k n(j) = \sum_{j=0}^k \sum_{i=a_j}^{b_j} s(i) = \sum_{j=0}^{b_j} s(j) = t(bk)$ ，映射后灰度：

$q(k) = \text{int}[(N-1)t(bk) + 0.5] = p(bk) = k$ ，此时  $k- > k$ ，所以得证

## 3、讨论用于空间滤波的平滑滤波器和锐化滤波器的相同点、不同点以及联系。

- 相同点：都是通过减弱或消除信号傅里叶空间的某些分量来达到增强图像的效果；实现方法类似。
- 不同点：
  - 平滑滤波器：减弱或消除高频分量，增强低频，平滑图像。
  - 锐化滤波器：减弱或消除低频分量，增强高频，锐化图像。
- 联系：两者效果相反，互为补充，从原始图像中减去平滑滤波器的结果得到锐化滤波器的效果，反之亦然。

## 4、有一种常用的图象增强技术是将高频增强和直方图均衡化结合起来以达到使边缘锐化的反差增强效果，以上两个操作的先后次序对增强结果有影响吗？为什么？

不能。因为高频增强是一种线性操作，直方图均衡是一种非线性操作，所以不可以交换先后次序

## 5、编程实现对lena.bmp分别加入高斯噪声和椒盐噪声，再进行局域平均和中值滤波。

伪代码：

```
1 def add_gaussian_noise(image, mean=0, sigma=25):
2     """
3     Add Gaussian noise to the image.
4     """
5     gauss = np.random.normal(mean, sigma, image.shape)
6     noisy_image = np.clip(image + gauss, 0, 255).astype(np.uint8)
7     return noisy_image
8
9 def add_salt_and_pepper_noise(image, salt_vs_pepper=0.5, amount=0.04):
10    """
11    Add salt and pepper noise to the image.
12    """
13    noisy_image = np.copy(image)
14    num_salt = np.ceil(amount * image.size * salt_vs_pepper)
15    coords = [np.random.randint(0, i - 1, int(num_salt)) for i in image.shape]
16    noisy_image[coords] = 255
17
18    num_pepper = np.ceil(amount * image.size * (1.0 - salt_vs_pepper))
19    coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in
20              image.shape]
21    noisy_image[coords] = 0
22    return noisy_image
23
24 def local_average_filter(image, kernel_size=3):
25    """
26    Apply local average filter to the image.
27    """
28    return cv2.blur(image, (kernel_size, kernel_size))
29
30 def median_filter(image, kernel_size=3):
31    """
32    Apply median filter to the image.
33    """
34    return cv2.medianBlur(image, kernel_size)
35
36 # Load Lena image
37 lena_image = cv2.imread('D:\lab\VSCodeProject\dip\imagedata\lena.bmp',
38                          cv2.IMREAD_GRAYSCALE)
39
40 # Add Gaussian noise
41 gaussian_noisy_image = add_gaussian_noise(lena_image)
```

```

42 salt_and_pepper_noisy_image = add_salt_and_pepper_noise(lena_image)
43
44 # Apply local average filter
45 gaussian_filtered_image = local_average_filter(gaussian_noisy_image)
46 salt_and_pepper_filtered_image =
    local_average_filter(salt_and_pepper_noisy_image)
47
48 # Apply median filter
49 gaussian_median_filtered_image = median_filter(gaussian_noisy_image)
50 salt_and_pepper_median_filtered_image =
    median_filter(salt_and_pepper_noisy_image)
51
52 # Display results
53 cv2.imshow('Original Lena', lena_image)
54 cv2.imshow('Gaussian Noisy Lena', gaussian_noisy_image)
55 cv2.imshow('Salt and Pepper Noisy Lena', salt_and_pepper_noisy_image)
56 cv2.imshow('Gaussian Filtered Lena', gaussian_filtered_image)
57 cv2.imshow('Salt and Pepper Filtered Lena', salt_and_pepper_filtered_image)
58 cv2.imshow('Gaussian Median Filtered Lena', gaussian_median_filtered_image)
59 cv2.imshow('Salt and Pepper Median Filtered Lena',
    salt_and_pepper_median_filtered_image)
60
61 cv2.waitKey(0)
62 cv2.destroyAllWindows()
63

```

结果：

加高斯噪声	加椒盐噪声

 Gaussian Noisy Lena



 Salt and Pepper Noisy Lena



局域平均

中值滤波

局域平均

中值滤波

 Gaussian Filtered Lena




 Gaussian Median Filtered Lena



 Salt and Pepper Filtered Lena



 Salt and Pepper Median Filtered Lena

