

22336216_陶宇卓

实验三

要求

1. 通过添加椒盐噪声并使用中值滤波器对图像进行去噪处理，观察不同次数的中值滤波效果。
2. 通过添加高斯噪声并使用维纳滤波器对图像进行去噪处理，观察不同参数下的维纳滤波效果。

原理

1. 中值滤波是一种非线性滤波技术，主要用于去除图像中的椒盐噪声。其基本原理是用窗口内像素的中值代替窗口中心像素的值，从而达到去噪的效果。具体步骤为：
 - a. 加载图像。
 - b. 添加椒盐噪声。
 - c. 应用多次中值滤波。
 - d. 显示原始图像、噪声图像和滤波后的图像。
2. 维纳滤波是一种线性滤波技术，主要用于去除图像中的噪声。其基本原理是通过最小化均方误差来恢复被噪声污染的图像。维纳滤波器在频域中进行操作，利用图像的频谱信息来实现去噪。具体步骤为：
 - a. 加载图像。
 - b. 应用H并添加高斯噪声。
 - c. 生成维纳滤波器。
 - d. 应用维纳滤波器进行去噪。
 - e. 显示原始图像、噪声图像和滤波后的图像。

具体实现

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 font = {'family': 'Microsoft YaHei',
5         'weight': 'bold',
6         'size': 'larger'}
7 (plt.rc("font", family='Microsoft YaHei', weight="bold"))
8 def add_salt_and_pepper_noise(image, pa, pb):
9     noisy_image = image.copy()
10    total_pixels = image.size
11    num_salt = int(pa * total_pixels)
12    num_pepper = int(pb * total_pixels)
13
14    # 添加盐噪声
15    coords = [np.random.randint(0, i, num_salt) for i in image.shape]
16    noisy_image[coords[0], coords[1]] = 255
17
18    # 添加椒噪声
19    coords = [np.random.randint(0, i, num_pepper) for i in image.shape]
20    noisy_image[coords[0], coords[1]] = 0
21
22    return noisy_image
23
24 # 加载图像 (a)
25 image = cv2.imread('a.jpg', cv2.IMREAD_GRAYSCALE)
26 image_b = cv2.imread('5_10(b).png', cv2.IMREAD_GRAYSCALE)
27 # 添加椒盐噪声
28 pa = pb = 0.2
29 noisy_image = add_salt_and_pepper_noise(image, pa, pb)
30
31 # 应用中值滤波
32 median_filtered_1 = cv2.medianBlur(noisy_image, 3)
33 median_filtered_2 = cv2.medianBlur(median_filtered_1, 3)
34 median_filtered_3 = cv2.medianBlur(median_filtered_2, 3)
35 median_filtered_4 = cv2.medianBlur(median_filtered_3, 3)
36 median_filtered_5 = cv2.medianBlur(median_filtered_4, 3)
37 # 显示结果
38 plt.figure(figsize=(10, 5))
39 plt.subplot(2, 4, 1)
40 plt.title("原始图像")
41 plt.imshow(image, cmap='gray')
42 plt.axis('off')
43
44 plt.subplot(2, 4, 2)
45 plt.title("加入椒盐噪声")
```

```

46 plt.imshow(noisy_image, cmap='gray')
47 plt.axis('off')
48
49 plt.subplot(2, 4, 3)
50 plt.title("中值滤波 (1 次)")
51 plt.imshow(median_filtered_1, cmap='gray')
52 plt.axis('off')
53
54 plt.subplot(2, 4, 4)
55 plt.title("中值滤波 (2 次)")
56 plt.imshow(median_filtered_1, cmap='gray')
57 plt.axis('off')
58
59 plt.subplot(2, 4, 5)
60 plt.title("中值滤波 (3 次)")
61 plt.imshow(median_filtered_2, cmap='gray')
62 plt.axis('off')
63
64 plt.subplot(2, 4, 6)
65 plt.title("中值滤波 (4 次)")
66 plt.imshow(median_filtered_3, cmap='gray')
67 plt.axis('off')
68
69 plt.subplot(2, 4, 7)
70 plt.title("中值滤波 (5 次)")
71 plt.imshow(median_filtered_4, cmap='gray')
72 plt.axis('off')
73
74 plt.subplot(2, 4, 8)
75 plt.title("5.10(b)")
76 plt.imshow(image_b, cmap='gray')
77 plt.axis('off')
78
79 plt.tight_layout()
80 plt.show()
81

```

```

1  import numpy as np
2  import cv2
3  import matplotlib.pyplot as plt
4  font = {'family': 'Microsoft YaHei',
5          'weight': 'bold',
6          'size': 'larger'}
7  (plt.rc("font", family='Microsoft YaHei', weight="bold"))
8  # 生成滤波器 H(u,v) (式 5.77)
9  def generate_filter(shape, T):
10     M, N = shape
11     u = np.arange(-M//2, M//2)
12     v = np.arange(-N//2, N//2)
13     U, V = np.meshgrid(u, v)
14     a = b = 0.1
15     uv_product = U * a + V * b
16     H = (T / (np.pi * uv_product)) * np.sin(np.pi * uv_product) * np.exp(-
17         1j * np.pi * uv_product)
18     H[uv_product == 0] = T # 对于 uv_product = 0 的点, H 为 T
19     return np.fft.fftshift(H)
20 # 添加高斯噪声
21 def add_gaussian_noise(image, mean=0, var=10):
22     noise = np.random.normal(mean, np.sqrt(var), image.shape)
23     noisy_image = np.clip(image + noise, 0, 255) # 保证值在[0,255]
24     return noisy_image.astype(np.uint8)
25
26 # 维纳滤波器恢复 (式 5.85)
27 def wiener_filter(blurred_img, H, K=0.01):
28     H_conj = np.conj(H)
29     H_abs2 = np.abs(H)**2
30     G = np.fft.fft2(blurred_img)
31     F_hat = (H_conj / (H_abs2 + K)) * G
32     restored_img = np.fft.ifft2(F_hat)
33     return np.abs(restored_img)
34
35 # 加载图像 (b)
36 image_b = cv2.imread('b.jpg', cv2.IMREAD_GRAYSCALE)
37
38 # 滤波器参数
39 T = 1
40 H = generate_filter(image_b.shape, T)
41
42 # 对图像 (b) 模糊处理
43 F = np.fft.fft2(image_b)
44 blurred_freq = F * H

```

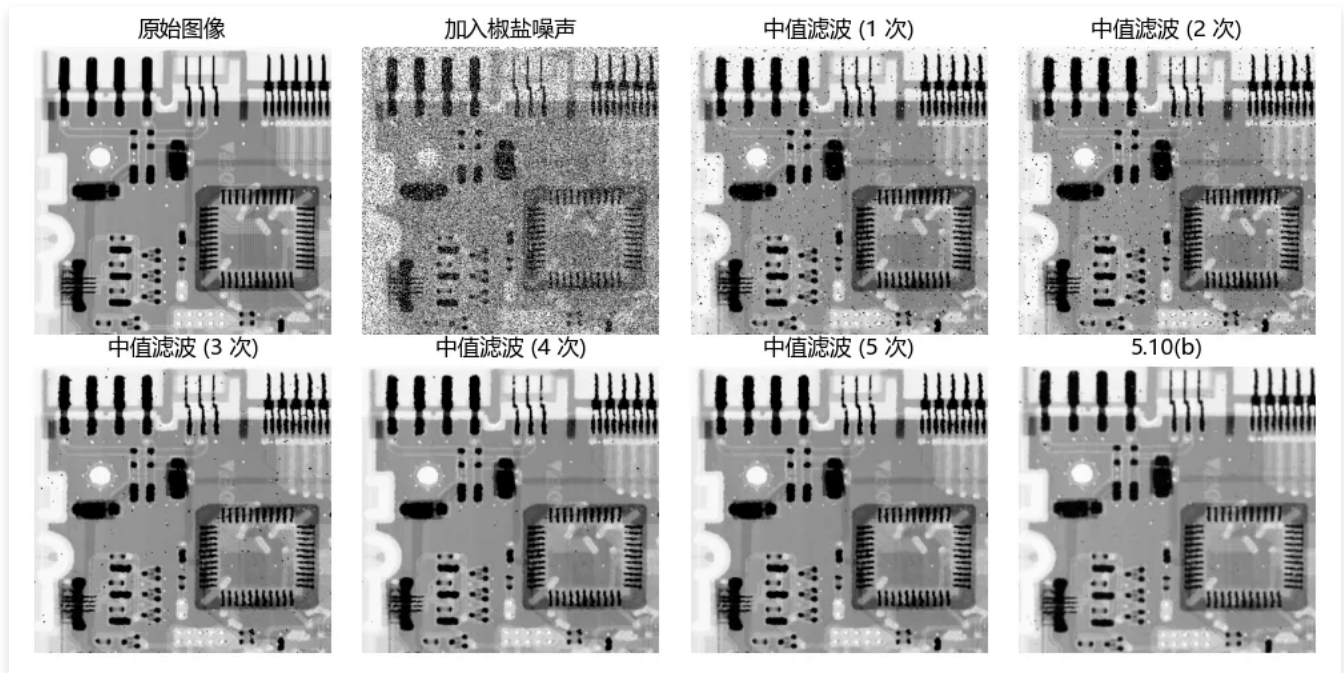
```

45 blurred_image = np.abs(np.fft.ifft2(blurred_freq))
46
47 # 添加高斯噪声
48 noisy_image = add_gaussian_noise(blurred_image)
49
50 # 应用维纳滤波器
51 restored_image = wiener_filter(noisy_image, H, K=0.001)
52
53 # 显示结果
54 plt.figure(figsize=(15, 8))
55 plt.subplot(1, 4, 1)
56 plt.title("原始图像")
57 plt.imshow(image_b, cmap='gray')
58 plt.axis('off')
59
60 plt.subplot(1, 4, 2)
61 plt.title("模糊图像")
62 plt.imshow(blurred_image, cmap='gray')
63 plt.axis('off')
64
65 plt.subplot(1, 4, 3)
66 plt.title("加入高斯噪声")
67 plt.imshow(noisy_image, cmap='gray')
68 plt.axis('off')
69
70 plt.subplot(1, 4, 4)
71 plt.title("维纳滤波器恢复")
72 plt.imshow(restored_image, cmap='gray')
73 plt.axis('off')
74
75 plt.tight_layout()
76 plt.show()
77
78 plt.figure(figsize=(15, 8))
79 plt.subplot(1, 3, 1)
80 plt.title("K=0.001")
81 plt.imshow(restored_image, cmap='gray')
82 plt.axis('off')
83
84 plt.subplot(1, 3, 2)
85 plt.title("K=0.003")
86 restored_image_1 = wiener_filter(noisy_image, H, K=0.003)
87 plt.imshow(restored_image_1, cmap='gray')
88 plt.axis('off')
89
90 plt.subplot(1, 3, 3)
91 restored_image_2 = wiener_filter(noisy_image, H, K=0.005)
92 plt.title("K=0.005")

```

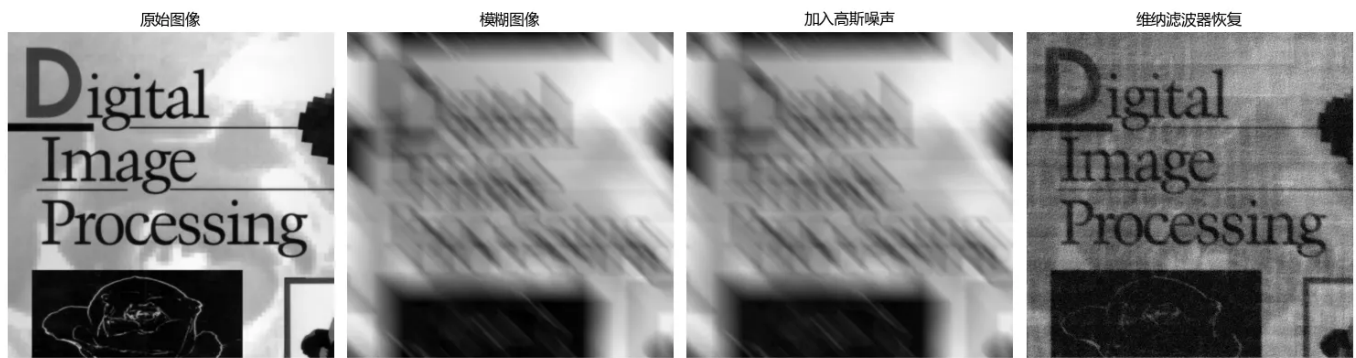
```
93 plt.imshow(restored_image_2, cmap='gray')
94 plt.axis('off')
95
96 plt.tight_layout()
97 plt.show()
98
```

结果分析

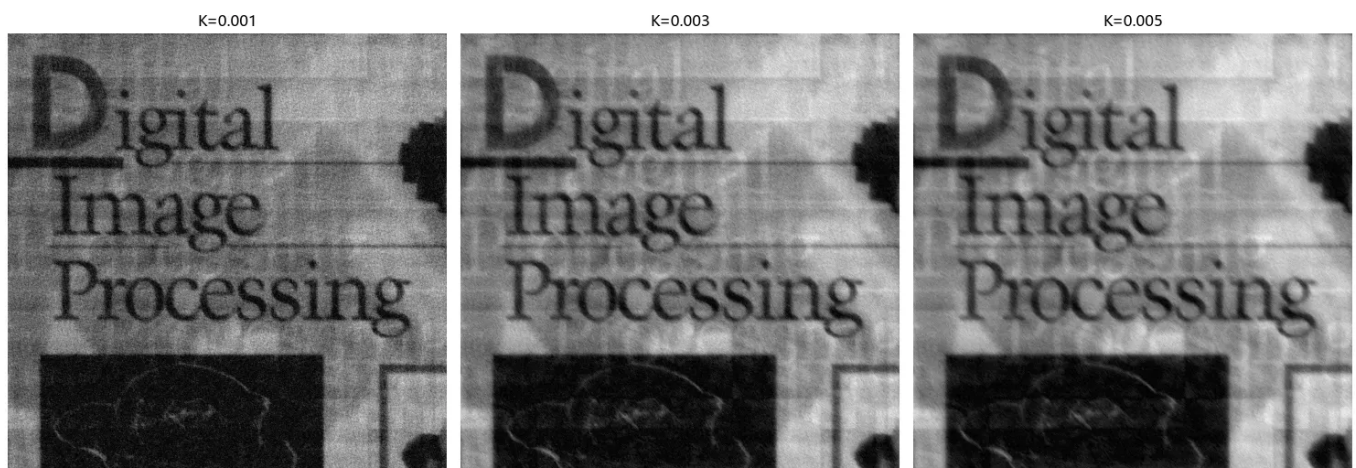


实验一结果

通过实验结果可以看出，随着中值滤波次数的增加，图像中的椒盐噪声逐渐减少，图像变得更加平滑。然而，过多的中值滤波会导致图像细节的丢失。



实验二结果



通过实验结果可以看出，维纳滤波器能够有效去除图像中的高斯噪声。不同的参数 K 对去噪效果有显著影响，较小的 K 值可以更好地恢复图像细节，但可能会保留一些噪声；较大的 K 值可以更好地去除噪声，但可能会导致图像细节的丢失。

总结

1. 中值滤波在去除椒盐噪声方面效果显著，但需要控制滤波次数以避免图像细节的丢失。实验表明，适当的中值滤波次数可以在去噪和保持图像细节之间取得平衡。
2. 维纳滤波在去除高斯噪声方面效果显著，但需要选择合适的参数 K 以在去噪和保持图像细节之间取得平衡。实验表明，适当的 K 值可以在去噪和保持图像细节之间取得良好的平衡。