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

Python

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
font = {'family': 'MicroSoft YaHei',
        'weight': 'bold',
        'size': 'larger'}
(plt.rc("font", family='MicroSoft YaHei', weight="bold"))
def add_salt_and_pepper_noise(image, pa, pb):
    noisy_image = image.copy()
    total_pixels = image.size
    num_salt = int(pa * total_pixels)
    num_pepper = int(pb * total_pixels)

    # 添加盐噪声
    coords = [np.random.randint(0, i, num_salt) for i in image.shape]
    noisy_image[coords[0], coords[1]] = 255

    # 添加椒噪声
    coords = [np.random.randint(0, i, num_pepper) for i in image.shape]
    noisy_image[coords[0], coords[1]] = 0

    return noisy_image

# 加载图像 (a)
image = cv2.imread('a.jpg', cv2.IMREAD_GRAYSCALE)
image_b = cv2.imread('5_10(b).png', cv2.IMREAD_GRAYSCALE)
# 添加椒盐噪声
pa = pb = 0.2
noisy_image = add_salt_and_pepper_noise(image, pa, pb)

# 应用中值滤波
median_filtered_1 = cv2.medianBlur(noisy_image, 3)
median_filtered_2 = cv2.medianBlur(median_filtered_1, 3)
median_filtered_3 = cv2.medianBlur(median_filtered_2, 3)
median_filtered_4 = cv2.medianBlur(median_filtered_3, 3)
median_filtered_5 = cv2.medianBlur(median_filtered_4, 3)
# 显示结果
plt.figure(figsize=(10, 5))
plt.subplot(2, 4, 1)
plt.title("原始图像")
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 2)
plt.title("加入椒盐噪声")
```

```
plt.imshow(noisy_image, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 3)
plt.title("中值滤波 (1 次)")
plt.imshow(median_filtered_1, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 4)
plt.title("中值滤波 (2 次)")
plt.imshow(median_filtered_1, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 5)
plt.title("中值滤波 (3 次)")
plt.imshow(median_filtered_2, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 6)
plt.title("中值滤波 (4 次)")
plt.imshow(median_filtered_3, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 7)
plt.title("中值滤波 (5 次)")
plt.imshow(median_filtered_4, cmap='gray')
plt.axis('off')

plt.subplot(2, 4, 8)
plt.title("5.10(b)")
plt.imshow(image_b, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

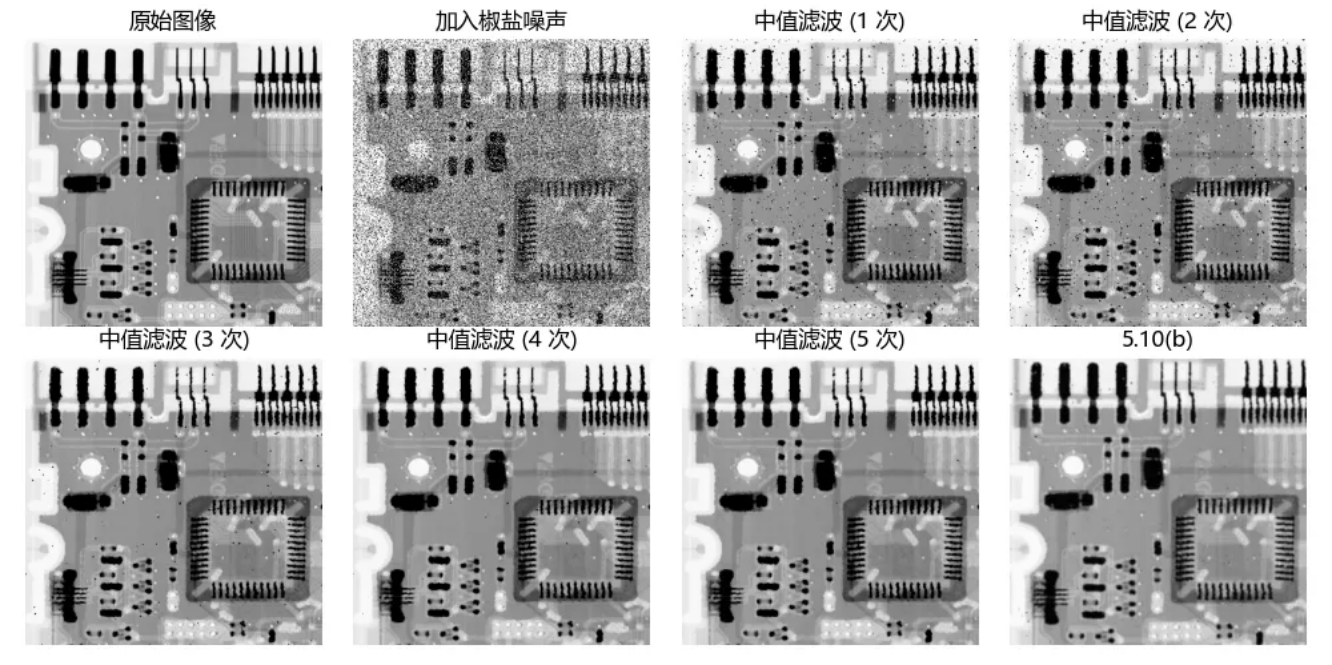
Python

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
font = {'family': 'MicroSoft YaHei',
        'weight': 'bold',
        'size': 'larger'}
(plt.rc("font", family='MicroSoft YaHei', weight="bold"))
# 生成滤波器 H(u,v) (式 5.77)
def generate_filter(shape, T):
    M, N = shape
    u = np.arange(-M//2, M//2)
    v = np.arange(-N//2, N//2)
    U, V = np.meshgrid(u, v)
    a = b = 0.1
    uv_product = U * a + V * b
    H = (T / (np.pi * uv_product)) * np.sin(np.pi * uv_product) * np.exp(-1j * np.pi * uv_product)
    H[uv_product == 0] = T  # 对于 uv_product = 0 的点，H 为 T
    return np.fft.fftshift(H)

# 添加高斯噪声
def add_gaussian_noise(image, mean=0, var=10):
    noise = np.random.normal(mean, np.sqrt(var), image.shape)
    noisy_image = np.clip(image + noise, 0, 255)  # 保证值在[0,255]
    return noisy_image.astype(np.uint8)

# 维纳滤波器恢复 (式 5.85)
def wiener_filter(blurred_img, H, K=0.01):
    H_conj = np.conj(H)
    H_abs2 = np.abs(H)**2
    G = np.fft.fft2(blurred_img)
    F_hat = (H_conj / (H_abs2 + K)) * G
    restored_img = np.fft.ifft2(F_hat)
    return np.abs(restored_img)

# 加载图像 (b)
image_b = cv2.imread('b.jpg', cv2.IMREAD_GRAYSCALE)

# 滤波器参数
T = 1
H = generate_filter(image_b.shape, T)

# 对图像 (b) 模糊处理
F = np.fft.fft2(image_b)
blurred_freq = F * H
```

```
blurred_image = np.abs(np.fft.ifft2(blurred_freq))

# 添加高斯噪声
noisy_image = add_gaussian_noise(blurred_image)

# 应用维纳滤波器
restored_image = wiener_filter(noisy_image, H, K=0.001)

# 显示结果
plt.figure(figsize=(15, 8))
plt.subplot(1, 4, 1)
plt.title("原始图像")
plt.imshow(image_b, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 2)
plt.title("模糊图像")
plt.imshow(blurred_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 3)
plt.title("加入高斯噪声")
plt.imshow(noisy_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 4, 4)
plt.title("维纳滤波器恢复")
plt.imshow(restored_image, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 8))
plt.subplot(1, 3, 1)
plt.title("K=0.001")
plt.imshow(restored_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("K=0.003")
restored_image_1 = wiener_filter(noisy_image, H, K=0.003)
plt.imshow(restored_image_1, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
restored_image_2 = wiener_filter(noisy_image, H, K=0.005)
plt.title("K=0.005")
```

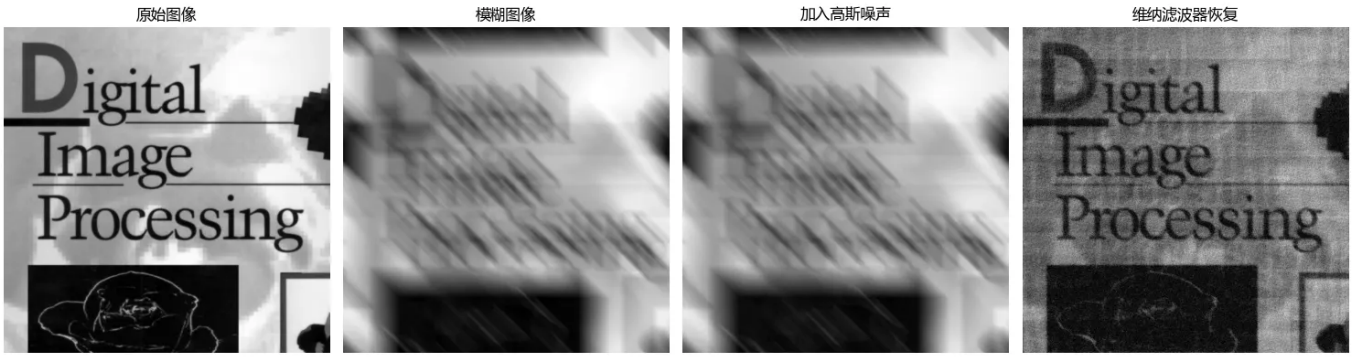
```
    plt.imshow(restored_image_2, cmap='gray')
    plt.axis('off')

    plt.tight_layout()
    plt.show()
```

结果分析

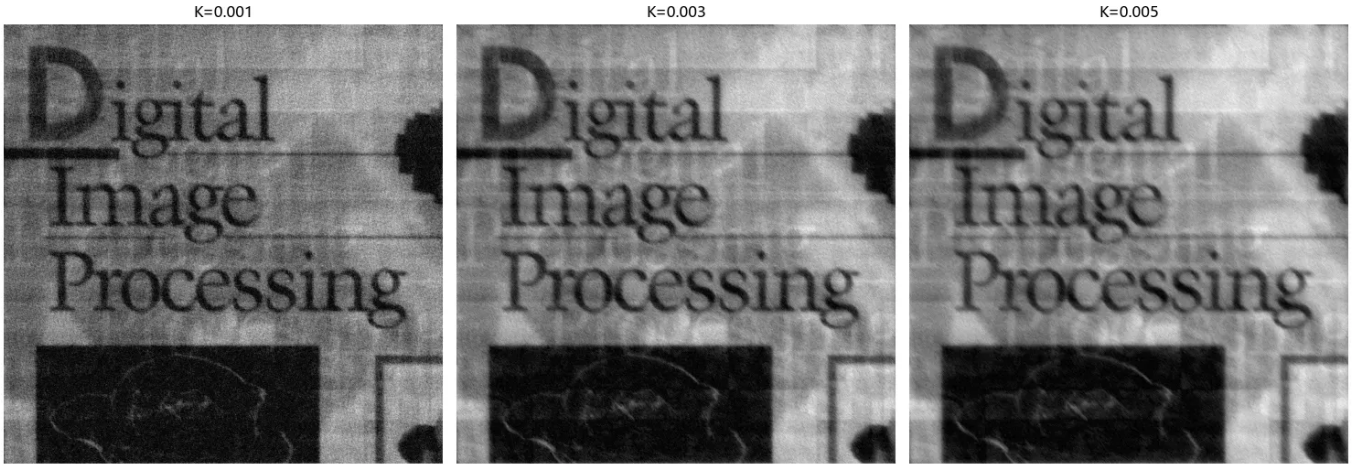


实验一结果

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。