课堂实验二： 图像增强—空域滤波

一、 实验目的

进一步了解python 软件/语言，学会使用 python对图像作滤波处理，使学生有机会掌握滤波算法，体会滤波效果。

了解几种不同滤波方式的使用和使用的场合，培养处理实际图像的能力，并为课堂教学提供配套的实践机会

二、 实验内容及步骤

a) 调入并显示原始图像 Sample2-1.jpg 。

b) 利用 imnoise 命令在图像 Sample2-1.jpg 上加入高斯(gaussian) 噪声

c)利用预定义函数 fspecial 命令产生平均(average)滤波器

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

d）分别采用 3x3 和 5x5 的模板，分别用平均滤波器以及中值滤波器，对加入噪声的图像进行处理并观察不同噪声水平下，上述滤波器处理的结果；

e）选择不同大小的模板，对加入某一固定噪声水平噪声的图像进行处理，观察上述滤波器处理的结果。

f）利用 imnoise 命令在图像 Sample2-1.jpg 上加入椒盐噪声(salt & pepper)

g）重复 c)~ e）的步骤

h）输出全部结果并进行讨论。

三、 实验代码及结果

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# a) 调入并显示原始图像 obtu.jpg
original_image1 = cv2.imread('obtu.jpg')
plt.subplot(2, 4, 1)
plt.imshow(cv2.cvtColor(original_image1, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

original_image = cv2.cvtColor(original_image1,cv2.COLOR_BGR2GRAY)

# b) 利用imnoise命令在图像obtu.jpg上加入高斯（gaussian）噪声
# 这里设置均值为0，方差为0.05，你可以调整方差来改变噪声水平
def add_gaussian_noise(image, mean=0, var=0.1):
```

```
    std = var ** 0.5
    noise = np.random.normal(mean, std, image.shape).astype(np.float32)
    noisy_image = image + noise*128
    noisy_image = np.clip(noisy_image, 0, 255).astype(np.uint8)
    return noisy_image

gaussian_noisy_image = add_gaussian_noise(original_image)
plt.subplot(2, 4, 2)
plt.imshow(cv2.cvtColor(gaussian_noisy_image, cv2.COLOR_GRAY2RGB))
plt.title('Gaussian Noisy Image')
plt.axis('off')

# c) 利用预定义函数fspecial命令产生平均(average)滤波器
def create_average_filter(size):
    return np.ones((size, size), dtype=np.float32) / (size * size)

average_filter_3x3 = create_average_filter(3)
average_filter_5x5 = create_average_filter(5)

# d) 分别采用3x3和5x5的模板，分别用平均滤波器以及中值滤波器，
# 对加入噪声的图像进行处理并观察不同噪声水平下，上述滤波器处理的结果
def apply_filter(image, filter_kernel):
    return cv2.filter2D(image, -1, filter_kernel)

def median_filter(image, kernel_size):
    return cv2.medianBlur(image, kernel_size)

# 3x3平均滤波
gaussian_3x3_avg_filtered = apply_filter(gaussian_noisy_image,
average_filter_3x3)
plt.subplot(2, 4, 3)
plt.imshow(cv2.cvtColor(gaussian_3x3_avg_filtered, cv2.COLOR_BGR2RGB))
plt.title('3x3 Average Filter on Gaussian Noisy Image')
plt.axis('off')

# 5x5平均滤波
gaussian_5x5_avg_filtered = apply_filter(gaussian_noisy_image,
average_filter_5x5)
plt.subplot(2, 4, 4)
plt.imshow(cv2.cvtColor(gaussian_5x5_avg_filtered, cv2.COLOR_BGR2RGB))
plt.title('5x5 Average Filter on Gaussian Noisy Image')
plt.axis('off')

# 3x3中值滤波
gaussian_3x3_median_filtered = median_filter(gaussian_noisy_image, 3)
plt.subplot(2, 4, 5)
plt.imshow(cv2.cvtColor(gaussian_3x3_median_filtered, cv2.COLOR_BGR2RGB))
plt.title('3x3 Median Filter on Gaussian Noisy Image')
plt.axis('off')

# 5x5中值滤波
gaussian_5x5_median_filtered = median_filter(gaussian_noisy_image, 5)
plt.subplot(2, 4, 6)
plt.imshow(cv2.cvtColor(gaussian_5x5_median_filtered, cv2.COLOR_BGR2RGB))
plt.title('5x5 Median Filter on Gaussian Noisy Image')
plt.axis('off')
```

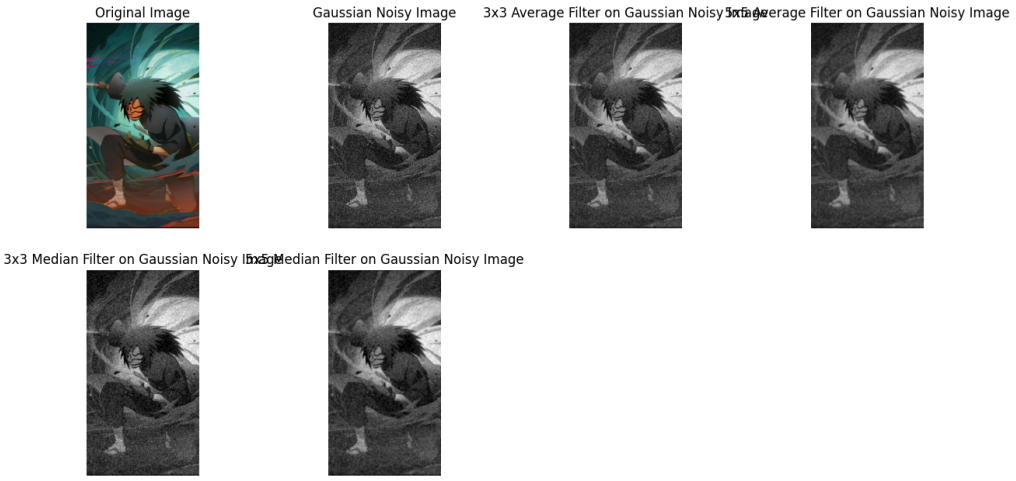
```python
# e) 选择不同大小的模板，对加入某一固定噪声水平噪声的图像进行处理，
# 观察上述滤波器处理的结果
def add_gaussian_noise_fixed(img, mean=0, sigma=25):
    np.random.seed(42)  # 设置随机种子以确保噪声水平固定
    gauss_noise = np.random.normal(mean, sigma, img.shape).astype(np.int32)
    noisy_image = cv2.add(img, gauss_noise.astype(np.uint8))
    return noisy_image

fixed_noise_image = add_gaussian_noise_fixed(original_image)
plt.figure(figsize=(8, 6))
plt.title('Fixed Noise Image')
plt.axis('off')
plt.imshow(fixed_noise_image, cmap='gray')
plt.show()

# 对 fixed_noise_image 进行 3x3 和 5x5 平均滤波处理
fixed_3x3_avg_filtered = apply_filter(fixed_noise_image, average_filter_3x3)
fixed_5x5_avg_filtered = apply_filter(fixed_noise_image, average_filter_5x5)

# 对 fixed_noise_image 进行 3x3 和 5x5 中值滤波处理
fixed_3x3_median_filtered = median_filter(fixed_noise_image, 3)
fixed_5x5_median_filtered = median_filter(fixed_noise_image, 5)

# 显示处理后的结果
plt.figure(figsize=(12, 8))

# 原始固定噪声图像
plt.subplot(2, 3, 1)
plt.imshow(fixed_noise_image, cmap='gray')
plt.title('Fixed Noise Image')
plt.axis('off')

# 3x3 平均滤波
plt.subplot(2, 3, 2)
plt.imshow(fixed_3x3_avg_filtered, cmap='gray')
plt.title('3x3 Average Filter')
plt.axis('off')

# 5x5 平均滤波
plt.subplot(2, 3, 3)
plt.imshow(fixed_5x5_avg_filtered, cmap='gray')
plt.title('5x5 Average Filter')
plt.axis('off')

# 3x3 中值滤波
plt.subplot(2, 3, 4)
plt.imshow(fixed_3x3_median_filtered, cmap='gray')
plt.title('3x3 Median Filter')
plt.axis('off')

# 5x5 中值滤波
plt.subplot(2, 3, 5)
plt.imshow(fixed_5x5_median_filtered, cmap='gray')
plt.title('5x5 Median Filter')
plt.axis('off')
```

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

# f) 利用imnoise命令在图像obtu.jpg上加入椒盐噪声(salt&pepper)
# 这里设置概率为0.01，你可以调整概率来改变噪声水平
def add_salt_pepper_noise(image, prob=0.01):
    output = np.zeros(image.shape, np.uint8)
    thres = 1 - prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = np.random.rand()
            if rdn < prob:
                output[i][j] = 0
            elif rdn > thres:
                output[i][j] = 255
            else:
                output[i][j] = image[i][j]
    return output

salt_pePPER_noisy_image = add_salt_pepper_noise(original_image)
plt.figure(figsize=(12, 8))
plt.subplot(2, 4, 1)
plt.imshow(cv2.cvtColor(salt_pePPER_noisy_image, cv2.COLOR_BGR2RGB))
plt.title('Salt & Pepper Noisy Image')
plt.axis('off')

# g) 重复c)~e)的步骤
# 3x3平均滤波
salt_pepper_3x3_avg_filtered = apply_filter(salt_pePPER_noisy_image,
average_filter_3x3)
plt.subplot(2, 4, 2)
plt.imshow(cv2.cvtColor(salt_pepper_3x3_avg_filtered, cv2.COLOR_BGR2RGB))
plt.title('3x3 Average Filter on Salt & Pepper Noisy Image')
plt.axis('off')

# 5x5平均滤波
salt_pepper_3x3_avg_filtered_5x5 = apply_filter(salt_pePPER_noisy_image,
average_filter_5x5)
plt.subplot(2, 4, 3)
plt.imshow(cv2.cvtColor(salt_pepper_3x3_avg_filtered_5x5, cv2.COLOR_BGR2RGB))
plt.title('5x5 Average Filter on Salt & Pepper Noisy Image')
plt.axis('off')

# 3x3中值滤波
salt_pepper_3x3_median_filtered = median_filter(salt_pePPER_noisy_image, 3)
plt.subplot(2, 4, 4)
plt.imshow(cv2.cvtColor(salt_pePPER_noisy_image, cv2.COLOR_BGR2RGB))
plt.title('3x3 Median Filter on Salt & Pepper Noisy Image')
plt.axis('off')

# 5x5中值滤波
salt_pepper_5x5_median_filtered = median_filter(salt_pePPER_noisy_image, 5)
plt.subplot(2, 4, 5)
plt.imshow(cv2.cvtColor(salt_pePPER_noisy_image, cv2.COLOR_BGR2RGB))
plt.title('5x5 Median Filter on Salt & Pepper Noisy Image')
```

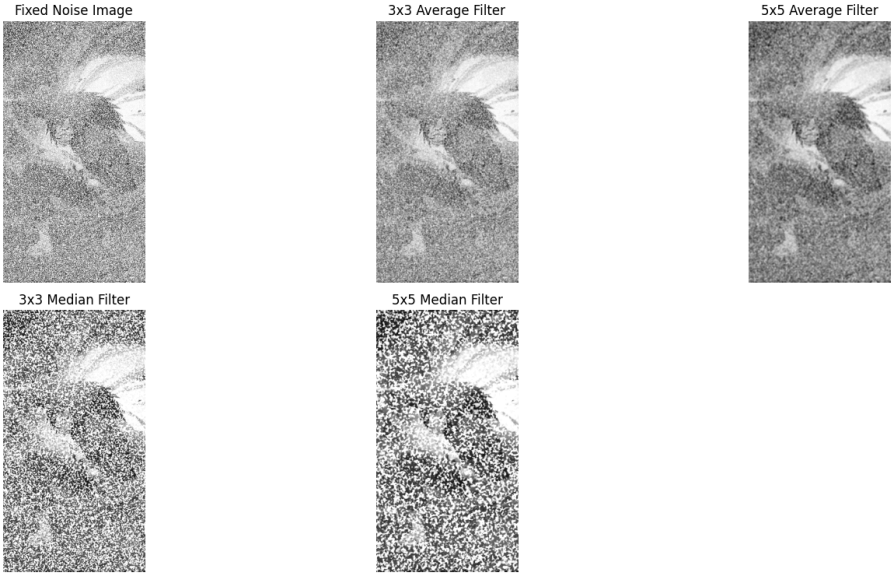
```
plt.axis('off')


# h) 输出全部结果并进行讨论
plt.show()

# 讨论部分：
# 1. 对于高斯噪声，平均滤波器在平滑噪声方面表现较好，随着模板尺寸的增大，噪声的平滑效果增强，
#    但图像也会变得更模糊。中值滤波器对于高斯噪声也有一定的抑制作用，并且在保持图像边缘方面比平均滤波器更好。
# 2. 对于椒盐噪声，中值滤波器的效果明显优于平均滤波器。平均滤波器会使椒盐噪声扩散，而中值滤波器能够有效地去除椒盐噪声，
#    同时较好地保留图像的边缘和细节。随着模板尺寸的增大，中值滤波器对椒盐噪声的去除效果会更好，但也会使图像变得更模糊。
```

原图像

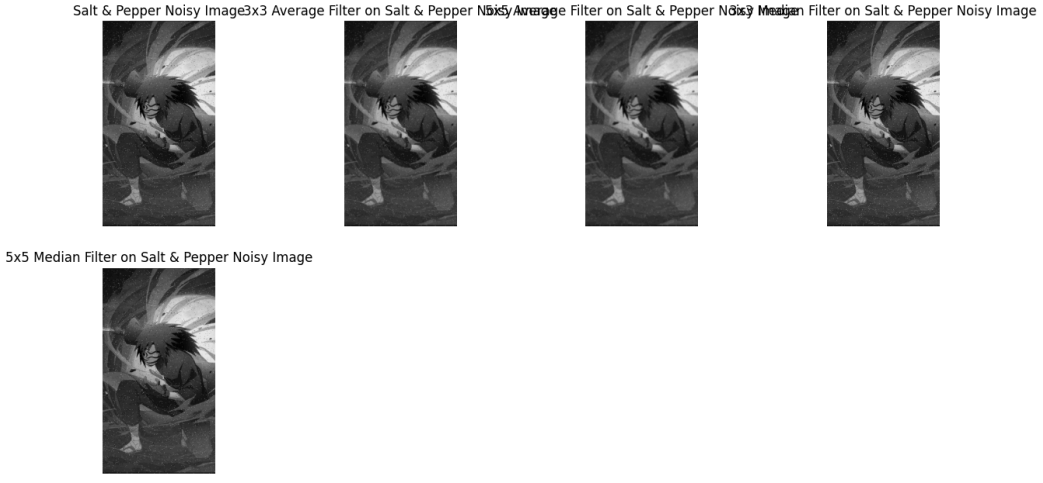
加入高斯噪声以及平均滤波/中值滤波



加入固定噪声



加入椒盐噪声



四、 思考题

(1) 高斯噪声和椒盐噪声的特点

高斯噪声（Gaussian Noise）

• 特点：

噪声值服从高斯分布（正态分布），即大部分噪声值集中在均值附近，远离均值的噪声值较少。

噪声强度由均值（μ）和方差（σ^2）决定，通常均值设为0，方差控制噪声强度。

噪声值连续变化，表现为图像中的细微颗粒状干扰。

• 视觉效果：

图像整体变模糊，细节被平滑，但边缘可能仍然可见（取决于噪声强度）。

类似于胶片照片的颗粒感或传感器噪声。

椒盐噪声（Salt & Pepper Noise）

• 特点：

噪声值随机分布在图像中，表现为纯黑（0）或纯白（255）的像素点。

噪声密度由概率参数控制（如0.01表示1%的像素被噪声污染）。

噪声值离散，表现为图像中的黑白斑点。

• 视觉效果：

图像中出现明显的黑白点状干扰，严重时可能影响图像识别。

类似于老式电视信号中断时的雪花点或传感器故障。

---

(2) 平均滤波器和中值滤波器的去噪效果对比

高斯噪声

• 平均滤波器：

• 优点：通过平滑邻域像素值，能有效降低高斯噪声的强度。

• 缺点：会模糊图像边缘，导致细节丢失（尤其是大尺寸滤波窗口时）。

• 实验观察：随着滤波窗口增大（如3×3→5×5），噪声抑制效果增强，但图像模糊更明显。

• 中值滤波器：

• 优点：对高斯噪声有一定抑制作用，且能较好保留边缘（优于平均滤波器）。

• 缺点：对高斯噪声的抑制效果不如平均滤波器（需结合其他方法如非局部均值）。

• 实验观察：中值滤波后的图像边缘更清晰，但噪声仍残留。

椒盐噪声

• 平均滤波器：

• 缺点：会扩散椒盐噪声（如将黑点周围的像素值拉低，白点周围的像素值拉高），导致噪声更明显。

• 实验观察：椒盐噪声经平均滤波后可能形成灰色斑块，效果较差。

• 中值滤波器：

• 优点：通过排序取中值，能直接去除椒盐噪声（黑/白点被邻域中值替代）。

• 缺点：大尺寸窗口可能导致边缘模糊（但比平均滤波器好）。

• 实验观察：中值滤波后的图像几乎无椒盐噪声残留，边缘保留较好。

---

(3) 滤波窗口对去噪效果的影响

高斯噪声
• 小窗口（如3×3）：

噪声抑制较弱，图像细节保留较好。

适合噪声强度较低的情况。

• 大窗口（如5×5）：

噪声抑制更强，图像更平滑。

但边缘模糊更明显，细节丢失严重。

• 实验结论：

需权衡噪声抑制和边缘保留，通常选择中等窗口（如5×5）。

椒盐噪声
• 小窗口（如3×3）：

对椒盐噪声去除效果有限（需更高密度噪声才能被中值替代）。

• 大窗口（如5×5）：

去噪效果更好，但可能模糊边缘。

• 实验结论：

椒盐噪声对窗口大小不敏感（中值滤波总能有效去除），但需避免过大窗口导致过度模糊。

总结

噪声类型	平均滤波器	中值滤波器
高斯噪声	抑制强但模糊边缘	抑制较弱但保留边缘
椒盐噪声	效果差（扩散噪声）	效果好（直接去除）

滤波窗口建议：
• 高斯噪声：优先用中值滤波器，若必须用平均滤波器，选择中等窗口（5×5）。

• 椒盐噪声：直接用中值滤波器，窗口大小对效果影响较小（通常3×3或5×5即可）。