

《计算机视觉》（本科，2025）作业 2

- 将附带的彩色图像 (I0) 转为灰度图像 (记为 I1)。

【代码贴这里】

```
to_gray.py <input>
1 import cv2
2
3 # 读取彩色图像 (替换为你的图片路径)
4 I0 = cv2.imread('I0.jpeg')
5
6 # 转为灰度图像
7 I1 = cv2.cvtColor(I0, cv2.COLOR_BGR2GRAY)
8
9 # 保存结果
10 cv2.imwrite( filename: 'I1.jpg' , I1)
```

【结果（灰度图像 I1）贴这里】



2. 在灰度图像 I1 上增加不同类型（类型>=3）的噪声，分别生成噪声图像。

【代码贴这里】

```
import cv2
import numpy as np

# 均值高斯噪声
def add_gaussian_noise(image, mean=0, sigma=30):
    """
    :param image: 灰度图
    :param mean: 均值
    :param sigma: 标准差
    :return: 带有高斯噪声的图像
    """
    rows, cols = image.shape
    gauss = np.random.normal(mean, sigma, [rows, cols])
    noisy = image.astype(np.float32) + gauss
    noisy = np.clip(noisy, 0.0, 255).astype(np.uint8)
    return noisy

# 盐椒噪声
def add_salt_pepper_noise(image, prob=0.001):
    """
    :param image: 灰度图
    :param prob: 带盐椒噪声的概率 (建议在0.001-0.1)
    :return: 带有盐椒噪声的图像
    """
    output = np.copy(image)
    num_salt = int(prob * image.size)
    thresh = 1 - prob
    random_matrix = np.random.uniform(image.shape)
    output[random_matrix > thresh] = 0
    output[random_matrix < prob/2] = 255
    return output

# 均匀分布噪声
def add_uniform_noise(image, intensity=50):
    """
    :param image: 灰度图 (噪声强度 [-intensity/2, intensity/2])
    :param intensity: 噪声强度 (噪声范围 [-intensity/2, intensity/2], image.shape)
    :return: 带均匀分布噪声的图像
    """
    uniform_noise = np.random.uniform(-intensity/2, intensity/2, image.shape)
    noisy = image.astype(np.float32) + uniform_noise
    noisy = np.clip(noisy, 0.0, 255).astype(np.uint8)
    return noisy

# 读取灰度图像 (建议使用灰度)
I1 = cv2.imread('I1.jpg', cv2.IMREAD_GRAYSCALE)

# 生成带噪声的图像
I_gaussian = add_gaussian_noise(I1, sigma=30) # 高斯噪声
I_salt_pepper = add_salt_pepper_noise(I1, prob=0.1) # 盐椒噪声
I_uniform = add_uniform_noise(I1, intensity=80) # 均匀噪声

# 保存结果
cv2.imwrite('filename=' + 'noisy_gaussian.jpg', I_gaussian)
cv2.imwrite('filename=' + 'noisy_salt_pepper.jpg', I_salt_pepper)
cv2.imwrite('filename=' + 'noisy_uniform.jpg', I_uniform)
```

【结果（噪声图像）贴这里】



3. 设计不同类型（类型>=3）的滤波器，对上述噪声图像分别进行去噪。对结果进行分析。

【滤波器贴这里】

```
# 均值滤波器
3 usages
def mean_filter(image, kernel_size=3):
    return cv2.blur(image, [kernel_size, kernel_size])

# 中值滤波器
3 usages
def median_filter(image, kernel_size=3):
    return cv2.medianBlur(image, kernel_size)

# 高斯滤波器
3 usages
def gaussian_filter(image, kernel_size=3, sigma=0):
    return cv2.GaussianBlur(image, [kernel_size, kernel_size], sigma)

# 去噪处理
gaussian_mean = mean_filter(I_gaussian)
gaussian_median = median_filter(I_gaussian)
gaussian_gaussian = gaussian_filter(I_gaussian)

salt_pepper_mean = mean_filter(I_salt_pepper)
salt_pepper_median = median_filter(I_salt_pepper)
salt_pepper_gaussian = gaussian_filter(I_salt_pepper)

uniform_mean = mean_filter(I_uniform)
uniform_median = median_filter(I_uniform)
uniform_gaussian = gaussian_filter(I_uniform)
```

【结果（去噪后的图像）贴这里】



【分析贴在这里】

均值滤波器：

原理：用邻域内像素的平均值替换中心像素值。

效果：对高斯噪声和均匀噪声有一定的平滑效果，但会使图像边缘和细节模糊。对于椒盐噪声，由于噪声点的值与周围像素差异较大，均值滤波不能很好地去除，反而会在噪声点周围形成模糊区域。

中值滤波器：

原理：用邻域内像素值的中值替换中心像素值。

效果：对椒盐噪声有很好的去除效果，因为中值可以有效抑制孤立的噪声点。对于高斯噪声和均匀噪声，中值滤波的去噪效果不如高斯滤波器，可能会保留部分噪声，同时对图像细节的保留较好。

高斯滤波器：

原理：根据高斯函数对邻域内像素进行加权平均，离中心越近的像素权重越大。

效果：对高斯噪声有较好的去除效果，因为高斯噪声本身符合正态分布，高斯滤波可以有效平滑噪声。对于均匀噪声也有一定的去噪效果，但对椒盐噪声的去除效果不佳。

4. 尝试对彩色图像 I0 添加噪声，并设计滤波器进行去噪。对结果进行分析。

【代码贴这里】

```
!usage
def add_gaussian_noise(image, mean=0, sigma=25):
    """
    添加高斯噪声
    - mean: 噪声均值
    - sigma: 噪声标准差 (控制噪声强度)
    """
    noise = np.random.normal(mean, sigma, image.shape).astype(np.float32)
    noisy = image.astype(np.float32) + noise
    noisy = np.clip(noisy, a_min=0, a_max=255).astype(np.uint8)
    return noisy

! usage
def add_salt_pepper_noise(image, prob=0.05):
    """
    添加椒盐噪声
    - prob: 噪声点出现的概率 (盐噪声和椒盐噪声各占一半)
    """
    output = np.copy(image)
    half_prob = prob / 2
    random_matrix = np.random.random(image.shape[2])
    for c in range(image.shape[2]):
        output[random_matrix < half_prob, c] = 0
        output[random_matrix > 1 - half_prob, c] = 255
    return output

! usage
def add_uniform_noise(image, intensity=50):
    """
    添加均匀分布噪声
    - intensity: 噪声强度 (噪声范围是 [-intensity/2, intensity/2])
    """
    noise = np.random.uniform(-intensity / 2, intensity / 2, image.shape).astype(np.float32)
    noisy = image.astype(np.float32) + noise
    noisy = np.clip(noisy, a_min=0, a_max=255).astype(np.uint8)
    return noisy
```

```
# 均值滤波器
@usage
def mean_filter(image, kernel_size=3):
    return cv2.blur(image, ksize=(kernel_size, kernel_size))

# 中值滤波器
@usage
def median_filter(image, kernel_size=3):
    return cv2.medianBlur(image, kernel_size)

# 高斯滤波器
@usage
def gaussian_filter(image, kernel_size=3, sigma=0):
    return cv2.GaussianBlur(image, ksize=(kernel_size, kernel_size), sigma)

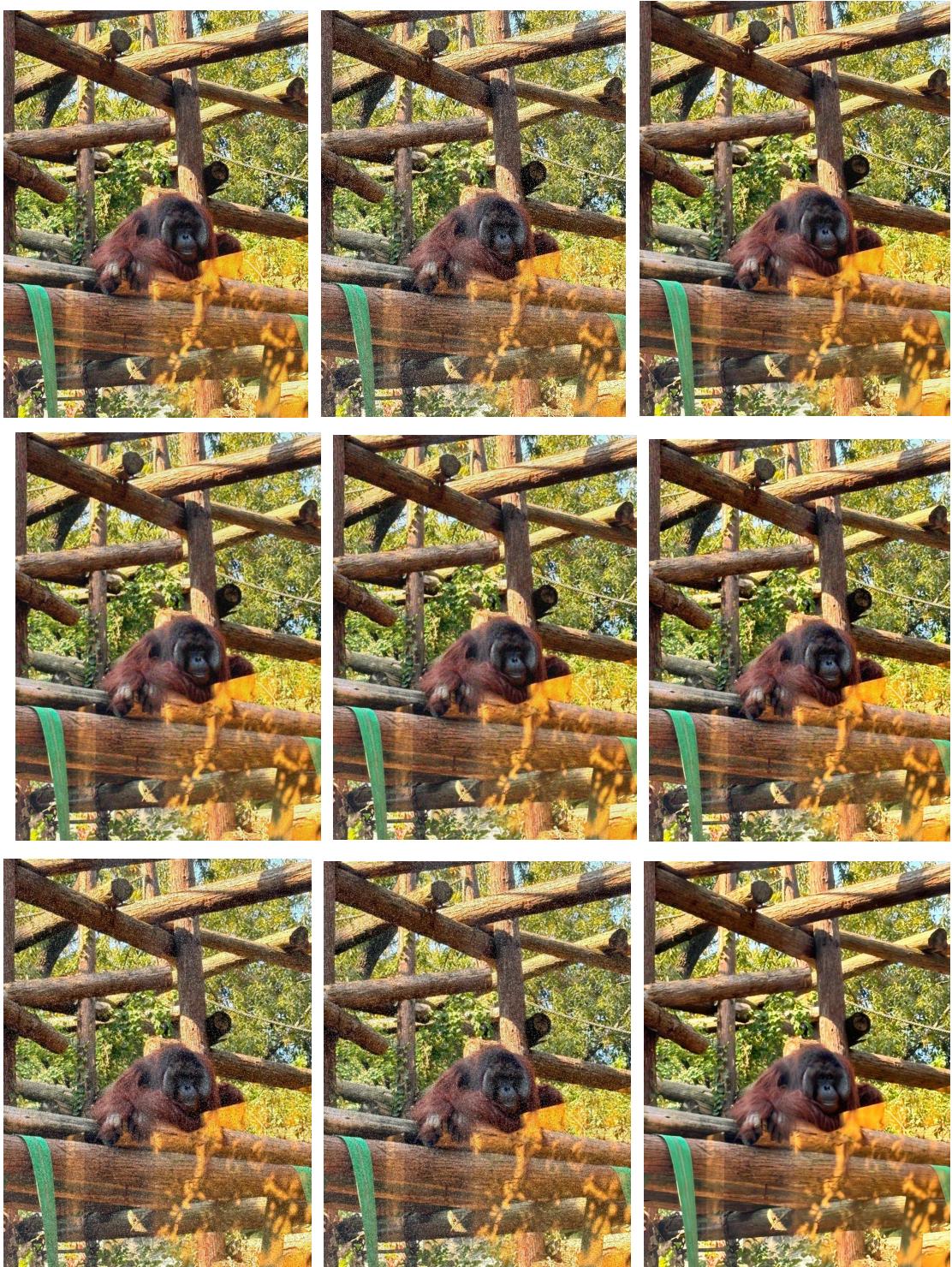
# 读取彩色图像
I0 = cv2.imread('I0.jpeg')
if I0 is None:
    print("无法读取图像，请检查文件路径。")
else:
    # 生成带噪声的图像
    I0_gaussian = add_gaussian_noise(I0, sigma=30) # 高斯噪声
    I0_salt_pepper = add_salt_pepper_noise(I0, prob=0.1) # 椒盐噪声
    I0_uniform = add_uniform_noise(I0, intensity=80) # 均匀噪声

    # 去噪处理
    gaussian_mean = mean_filter(I0_gaussian)
    gaussian_median = median_filter(I0_gaussian)
    gaussian_gaussian = gaussian_filter(I0_gaussian)

    salt_pepper_mean = mean_filter(I0_salt_pepper)
    salt_pepper_median = median_filter(I0_salt_pepper)
    salt_pepper_gaussian = gaussian_filter(I0_salt_pepper)

    uniform_mean = mean_filter(I0_uniform)
    uniform_median = median_filter(I0_uniform)
    uniform_gaussian = gaussian_filter(I0_uniform)
```

【结果（噪声图像+去噪结果）贴这里】





【分析贴在这里】

不同噪声添加效果

高斯噪声：图像整体变得模糊，出现随机的亮度和颜色波动，类似图像被蒙上一层“雾”。

椒盐噪声：图像中随机分布着白色和黑色的像素点，像撒上了盐和胡椒粒，严重影响图像的视觉效果。

均匀噪声：图像颜色和亮度出现均匀的随机变化，图像细节变得模糊，整体视觉效果变差。

均值滤波器

高斯噪声：能在一定程度上平滑噪声，使图像整体变得更柔和，但会导致图像边缘和细节模糊，颜色过渡更平缓。

椒盐噪声：效果较差，无法有效去除孤立的黑白噪声点，反而会使噪声点周围的颜色变得模糊，让图像整体更加模糊不清。

均匀噪声：可以减轻噪声的影响，但也会使图像细节丢失，图像变得模糊。

中值滤波器

高斯噪声：去噪效果不如高斯滤波器，会保留部分噪声，但对图像细节的保留相对较好，不会像均值滤波器那样严重模糊图像。

椒盐噪声：去噪效果显著，能有效去除孤立的黑白噪声点，较好地保留图像的边缘和细节，图像整体清晰度较高。

均匀噪声：有一定的去噪效果，但不如高斯滤波器，可能会使图像产生一些块状效应。

高斯滤波器

高斯噪声：去噪效果较好，能有效平滑噪声，因为高斯滤波器的特性与高斯噪声的分布相匹配，能在去除噪声的同时较好地保留图像的部分细节。

椒盐噪声：对椒盐噪声的去除效果不佳，不能有效去除孤立的黑白噪声点，图像中仍会残留较多噪声。

均匀噪声：可以在一定程度上减轻噪声，使图像变得更平滑，但也会导致图像细节有所损失。