《计算机视觉》（本科，2024）作业 2

1. 将附带的彩色图像（I0）转为灰度图像（记为 I1）。

【代码贴这里】

```python
import cv2

# 将附带的彩色图像转化为灰度图像
image = cv2.imread(r'.\assert\I0.jpg')
# cv 库自动转化为灰度图
image_gray_cv2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imwrite("assert/I1.jpg", image_gray_cv2)
```

【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】

```python
import random

import cv2
import numpy as np


def random_noise(image, noise_num):
```

```
    # 添加随机噪点（实际上就是随机在图像上将像素点的灰度值变为255 即白色）

    img = cv2.imread(image)
    img_noise = img
    rows, cols, chn = img_noise.shape
    for i in range(noise_num):
        x = np.random.randint(0, rows)
        y = np.random.randint(0, cols)
        img_noise[x, y, :] = 255
    return img_noise


def sp_noise(image, prob):
    # 添加椒盐噪声，椒盐噪声是指两种噪声，一种是盐噪声，另一种是椒噪声。
    # 盐=白色(0)，椒=黑色(255)。前者是高灰度噪声，后者属于低灰度噪声。
    # 一般两种噪声同时出现，呈现在图像上就是黑白杂点

    image = cv2.imread(image)
    output = np.zeros(image.shape, np.uint8)
    tmp = 1 - prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()
            if rdn < prob:
                output[i][j] = 0
            elif rdn > tmp:
                output[i][j] = 255
            else:
                output[i][j] = image[i][j]
    result = output
    return result


def Gauss_noise(image, mean=0, var=0.001):
    # 添加高斯噪声，方差越大，噪声越明显

    image = cv2.imread(image)

    # 将原始图像的像素值进行归一化，除以 255 使得像素值在 0-1 之间
    image = np.array(image / 255, dtype=float)
    # 创建一个均值为mean，方差为var 呈高斯分布的图像矩阵
```

```
    noise = np.random.normal(mean, var ** 0.5, image.shape)
    # 将噪声和原始图像进行相加得到加噪后的图像
    out = image + noise
    if out.min() < 0:
        low_clip = -1.
    else:
        low_clip = 0.

    # clip 函数将元素的大小限制在了 low_clip 和 1 之间了，小于 low_clip 的用 low_clip 代替，大于 1 的用 1 代替
    out = np.clip(out, low_clip, 1.0)
    # 解除归一化，乘以 255 将加噪后的图像的像素值恢复
    out = np.uint8(out * 255)
    return out


noise_Gauss = Gauss_noise("./assert/I1.jpg", mean=0, var=0.05)
noise_random = random_noise("./assert/I1.jpg", 100000)
noise_sp = sp_noise("./assert/I1.jpg", 0.1)
cv2.imwrite("assert/noise_Gauss.jpg", noise_Gauss)
cv2.imwrite("assert/noise_random.jpg", noise_random)
cv2.imwrite("assert/noise_sp.jpg", noise_sp)
```

【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】

```
import cv2
import numpy as np
from scipy.signal import convolve2d


def mean_filter(image, kernel_size):
    # 均值滤波器，在一个 kernel 内取这些像素点的平均值，然后滑动
    height, width = image.shape[:2]
    filtered_image = np.zeros_like(image)
    kernel_half = kernel_size // 2
```

```
    for i in range(height):
        for j in range(width):
            # 确定其实窗口位置，保证不超出图片范围
            x_start = max(0, i - kernel_half)
            x_end = min(height, i + kernel_half + 1)
            y_start = max(0, j - kernel_half)
            y_end = min(width, j + kernel_half + 1)

            window = image[x_start:x_end, y_start:y_end]
            # 求平均值
            filtered_image[i, j] = np.mean(window)

    return filtered_image.astype(np.uint8)


def median_filter(image, kernel_size):
    # 中值滤波器，在一个 kernel 内取这些像素点的中值，然后滑动
    height, width = image.shape[:2]
    filtered_image = np.zeros_like(image)
    kernel_half = kernel_size // 2
    for i in range(height):
        for j in range(width):
            x_start = max(0, i - kernel_half)
            x_end = min(height, i + kernel_half + 1)
            y_start = max(0, j - kernel_half)
            y_end = min(width, j + kernel_half + 1)

            window = image[x_start:x_end, y_start:y_end].ravel()
            # 求中值
            filtered_image[i, j] = np.median(window)

    return filtered_image.astype(np.uint8)


# 高斯函数 G(X, Y)
def gaussian(x, y, sigma):
    return 1 / (2 * np.pi * sigma**2) * np.exp(-(x**2 + y**2) / (2 * sigma**2))


def gaussian_filter(image, size=5, sigma=1.0):
    # 高斯滤波器，高斯滤波器的核心思想是根据高斯函数计算每个像素点的权重。
```

```
    # 距离中心像素越远的像素，其权重越小。这样可以使得滤波器更多地关注周围像素，减少噪声的影响
    tmp = np.zeros((size, size))
    center = size // 2
    for x in range(size):
        for y in range(size):
            tmp[x, y] = gaussian(x - center, y - center, sigma)
    # 计算好了kernel中的每一个像素的权重值
    kernel = tmp / np.sum(tmp)
    filtered_image = cv2.filter2D(image, -1, kernel)
    return filtered_image


image = cv2.imread(r'.\assert\I1.jpg')
mean_filter = mean_filter(image, 5)
median_filter = median_filter(image, 5)
gaussian_filter = gaussian_filter(image)
cv2.imwrite("assert/mean_filter.jpg", mean_filter)
cv2.imwrite("assert/median_filter.jpg", median_filter)
cv2.imwrite("assert/gaussian_filter.jpg", gaussian_filter)
```

【结果（去噪后的图像）贴这里】

【分析贴在这里】

- 由于手动实现的代码和内置函数的代码去噪效果相同，但是速度差异过大，所以这里使用的是内置函数的滤波器

通过实验发现：

高斯噪声的图片在 3 种滤波器的处理下效果都比较一般，相比之下高斯滤波器的去噪稍微好一些。 其中中值滤波器和均值滤波器随着核大小增加对于图片的模糊程度更大，高斯滤波器则还需要 sigma 的增加才能是图片模糊。

椒盐噪声和随机噪声图片在中值滤波器的处理下效果特别好，将核大小设置恰当能够做到很高的还原度，但是高斯滤波器和均值滤波器则没办法做到这一点。从原理上分析来看，椒盐噪声和随机噪声对图片的像素破坏主要为黑白两种颜色，像素值为 0 或 255，在中值滤波

器中能够被忽略，从而保留正确结果

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】

```
import cv2

from hw2.question2 import Gauss_noise, random_noise, sp_noise

noise_Gauss = Gauss_noise("./assert/I0.jpg", mean=0, var=0.05)
noise_random = random_noise("./assert/I0.jpg", 100000)
noise_sp = sp_noise("./assert/I0.jpg", 0.1)
cv2.imwrite("assert/noise_Gauss_I0.jpg", noise_Gauss)
cv2.imwrite("assert/noise_random_I0.jpg", noise_random)
cv2.imwrite("assert/noise_sp_I0.jpg", noise_sp)

img = noise_random
mean_filter = cv2.blur(img, (3, 3))
cv2.imwrite("assert/mean_filter_I0.jpg", mean_filter)
median_filter = cv2.medianBlur(img, 3)
cv2.imwrite("assert/median_filter_I0.jpg", median_filter)
gaussian_filter = cv2.GaussianBlur(img, (3, 3), 2)
cv2.imwrite("assert/gaussian_filter_I0.jpg", gaussian_filter)
```

【结果（噪声图像+去噪结果）贴这里】

【分析贴在这里】

- 这里以椒盐噪声和中值去噪为例子

通过实验发现，彩色照片的规律和灰度照片的规律大致相同。由此可以总结出结论：

1. 高斯滤波器适合处理高斯照片，能够较好的保留图像细节，但是仍然无法去掉大部分噪声，计算成本较大。
2. 均值滤波器也能够处理高斯照片，但是如果噪声过大则处理效果不明显，同时丢失图像细节，变得模糊，计算成本低
3. 中值滤波器感觉功能性较强，能够处理很多非高斯噪声类型的图片，对于添加黑白噪声的图片有很强的处理能力，但是相较于前两种滤波器，更容易使得图像模糊丢失细节，计算成本低