图像去噪实验报告

3190103683 张嘉浩

1. 实验要求

1. 任意选取1张图像，添加高斯噪声和椒盐噪声。
2. 实现均值滤波、中值滤波和双边滤波等去噪方法，比较各方法去噪结果，并分析各去噪方法的特点。

2. 实验步骤

2.1 图像加噪声

1. 高斯噪声的概率密度函数服从高斯分布，在加噪时，只需按照高斯分布生成噪声即可，代码如下：

```
def gasuss_noise(image, mean=0, var=0.001):
    image = cv2.imread(image)
    print(image.shape)
    image = np.array(image/255, dtype=float)  # 将原始图像的像素值进行归一化
    # 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)
    out = image + noise  # 将噪声和原始图像进行相加得到加噪后的图像
    if out.min() < 0:
        low_clip = -1.
    else:
        low_clip = 0.
    out = np.clip(out, low_clip, 1.0)
    out = np.uint8(out*255)  # 解除归一化，乘以255将加噪后的图像的像素值恢复
    return out
```

2. 椒盐噪声指在一幅图像里随机将一个像素点变为椒噪声或盐噪声，其中椒噪声像素值为“0”，盐噪声像素值为“255”，代码如下：

```
def sp_noise(image, prob):
    image = cv2.imread(image)
    output = np.zeros(image.shape, np.uint8)
    noise_out = np.zeros(image.shape, np.uint8)
    thres = 1 - prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()  # 随机生成0-1之间的数字
            if rdn < prob:  # 如果生成的随机数小于噪声比例则将该像素点添加黑点，即椒噪声
                output[i][j] = 0
                noise_out[i][j] = 0
            elif rdn > thres:  # 如果生成的随机数大于（1-噪声比例）则将该像素点添加白点，即盐噪
声
                output[i][j] = 255
```

```
                noise_out[i][j] = 255
            else:
                output[i][j] = image[i][j]  # 其他情况像素点不变
                noise_out[i][j] = 100
    print(output.shape)
    return output
```

2.2 均值滤波

均值滤波的思想是通过一点和邻域内像素点求平均来去除突变的像素点，从而滤掉一定的噪声，其主要优点是算法简单，计算速度快， 但其代价是会造成图像一定程度上的模糊，用下式表示：

$$g(x,y)=\frac{1}{M}\sum_{(i,j)\in S}f(i,j)$$

式中：$x, y = 0, 1, ..., N-1$；S是以(x, y)为中心的邻域的集合，M是S内的点数。

实现代码如下：

```
def mean_filter(img, size):
    # set image kernal
    kernal = 1/(size[0]*size[1])*np.ones(size, np.float32)
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.sum(window*kernal)
    return img_new
```

2.3 中值滤波

中值滤波就是用一个奇数点的移动窗口， 将窗口中心点的值用窗口内各点的中值代替，由下式表示：

$$y_{ij}=\underset{A}{Med}\{f_{ij}\}$$

实现代码如下：

```
def median_filter(img, size):
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.median(window)
    return img_new
```

2.4 双边滤波

双边滤波是在高斯滤波的基础上加入了像素值权重项，即既要考虑距离因素，也要考虑像素值差异的影响，像素值越相近，权重越大。将像素值权重表示为G_r，空间距离权重表示为G_s，有：

$$G_s = exp(-\frac{\|p-q\|^2}{2\sigma_s^2}), G_r = exp(-\frac{\|I_p-I_q\|^2}{2\sigma_r^2})$$

$$BF = \frac{1}{W_q}\sum_{p\in S} G_s(p)G_r(p) * I_p$$

$$= \frac{1}{W_q}\sum_{p\in S} exp(-\frac{\|p-q\|^2}{2\sigma_s^2})exp(-\frac{\|I_p-I_q\|^2}{2\sigma_r^2}) * I_p$$

实现代码如下：

```
def bilateral_filter(img, size=(5, 5), sigma_s=10, sigma_r=10):
    kernal = np.ones(size, np.float32)
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            for i in range(window.shape[0]):
                for j in range(window.shape[1]):
                    dist = ((i-(window.shape[0]-1)/2)**2 +
                            (j-(window.shape[0]-1)/2)**2)/2/sigma_s
                    dValue = ((float(window[i][j]) - float(
                        window[int((window.shape[0]-1)/2)]
[int((window.shape[1]-1)/2)]))**2)/2/sigma_r
```

```
                    kernal[i][j] = math.exp(-dist)*math.exp(-dValue)
            kernal /= np.sum(kernal)
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.sum(window*kernal)
    return img_new
```

3. 实验结果

3.1 加入了高斯噪声和椒盐噪声的图像如下图所示：

图1. 加入高斯噪声后图片

图2. 加入椒盐噪声后图片

3.2 均值滤波后的图片如下图所示：

图3. 均值滤波高斯噪声图

图4. 均值滤波椒盐噪声图

3.3 中值滤波后的图片如下图所示：

图5. 中值滤波高斯噪声图

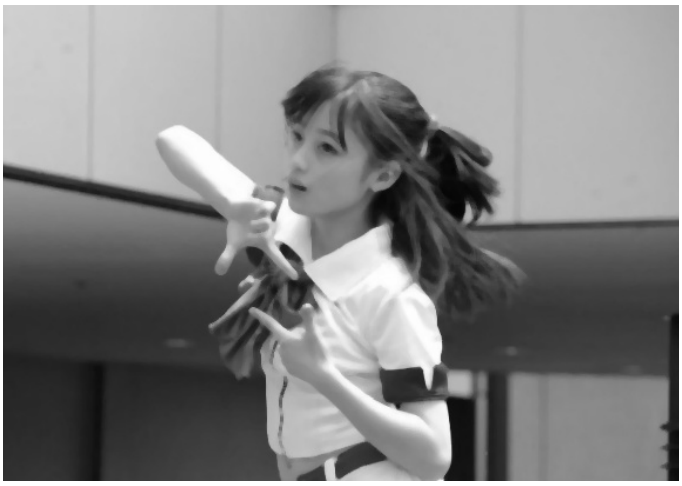

图6. 中值滤波椒盐噪声图

3.4 双边滤波后的图片如下图所示：

图5. 双边滤波高斯噪声图

图6. 双边滤波椒盐噪声图

4. 比较分析

4.1 均值滤波

从图中可以看到，均值滤波对高斯噪声和椒盐噪声均有一定程度的抑制，但不能完全去除。与此同时，均值滤波不能很好地保护图像细节，丢失图像特征信息，使图像模糊。

4.2. 中值滤波

从图中可以看到，中值滤波对高斯噪声和椒盐噪声均有较大程度的抑制，但也不能完全去除，处理后图像的模糊程度相比均值滤波有所改善，但涂抹感较为严重。

4.3 双边滤波

双边滤波的效果与上面两种滤波方式相比有了较大的提升，同时滤波的效果与两个σ的值选择有很大关系。如果想更多的去除平滑区域的噪声，应该提高σ_s，如果像保持边缘，则应该减小σ_r。

附录：源代码

`add_noise.py`

```
# -*- coding: utf-8 -*-
import cv2
import random
import numpy as np
import convert


def sp_noise(image, prob):
    '''
    添加椒盐噪声
    image:原始图片
    prob:噪声比例
    '''
    image = cv2.imread(image)
    # print(image.shape)
    output = np.zeros(image.shape, np.uint8)
    noise_out = np.zeros(image.shape, np.uint8)
    thres = 1 - prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()  # 随机生成0-1之间的数字
            if rdn < prob:  # 如果生成的随机数小于噪声比例则将该像素点添加黑点，即椒噪声
                output[i][j] = 0
                noise_out[i][j] = 0
            elif rdn > thres:  # 如果生成的随机数大于（1-噪声比例）则将该像素点添加白点，即盐噪声
                output[i][j] = 255
                noise_out[i][j] = 255
            else:
                output[i][j] = image[i][j]  # 其他情况像素点不变
                noise_out[i][j] = 100
    # result = [noise_out, output]  # 返回椒盐噪声和加噪图像
    print(output.shape)
    return output


def gasuss_noise(image, mean=0, var=0.001):
    '''
        添加高斯噪声
        image:原始图像
```

```
        mean : 均值
        var : 方差,越大，噪声越大
    '''
    image = cv2.imread(image)
    print(image.shape)
    image = np.array(image/255, dtype=float)  # 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
    # 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)
    # print(image.shape)
    # print(noise.shape)
    out = image + noise  # 将噪声和原始图像进行相加得到加噪后的图像
    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，小于的用low_clip代替，大于1的用1代替
    out = np.clip(out, low_clip, 1.0)
    out = np.uint8(out*255)  # 解除归一化，乘以255将加噪后的图像的像素值恢复
    #cv.imshow("gasuss", out)
    noise = noise*255
    return out


img_sp_noise = sp_noise('./RawImg.jpeg', 0.01)
img_gasuss_noise = gasuss_noise('./RawImg.jpeg')

cv2.imwrite('img_sp_noise.jpg', convert.RGB2GRAY(img_sp_noise))
cv2.imwrite('img_gasuss_noise.jpg', convert.RGB2GRAY(img_gasuss_noise))
```

filters.py

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


def mean_filter(img, size):
    # set image kernal
    kernal = 1/(size[0]*size[1])*np.ones(size, np.float32)
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
```

```
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.sum(window*kernal)
    return img_new


def median_filter(img, size):
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.median(window)
    return img_new


def bilateral_filter(img, size=(5, 5), sigma_s=10, sigma_r=10):
    kernal = np.ones(size, np.float32)
    # store the new image
    img_new = np.zeros(
        [int(img.shape[0]-2*(size[0]-1)/2), int(img.shape[1]-2*(size[1]-1)/2)])
    # convolution
    # in convolution area
    for x in range(int((size[0]-1)/2), int(img.shape[0]-(size[0]-1)/2)):
        for y in range(int((size[1]-1)/2), int(img.shape[1]-(size[1]-1)/2)):
            window = img[int(x-(size[0]-1)/2):int(x+(size[0]-1)/2+1),
                         int(y-(size[1]-1)/2):int(y+(size[1]-1)/2+1)]
            for i in range(window.shape[0]):
                for j in range(window.shape[1]):
                    dist = ((i-(window.shape[0]-1)/2)**2 +
                            (j-(window.shape[0]-1)/2)**2)/2/sigma_s
                    dValue = ((float(window[i][j]) - float(
                        window[int((window.shape[0]-1)/2)]
[int((window.shape[1]-1)/2)]))**2)/2/sigma_r
                    kernal[i][j] = math.exp(-dist)*math.exp(-dValue)
            kernal /= np.sum(kernal)
            img_new[x-int((size[0]-1)/2)][y-int((size[0]-1)/2)
                                          ] = np.sum(window*kernal)
    return img_new


if __name__ == '__main__':
```

```
    # read image
    img_gasuss_noise = cv2.imread('img_gasuss_noise.jpg', 0)
    img_sp_noise = cv2.imread('img_sp_noise.jpg', 0)

    # mean filter
    img_mean_gasuss_noise = mean_filter(img_gasuss_noise, (5, 5))
    cv2.imwrite("img_mean_gasuss_noise.jpg", img_mean_gasuss_noise)
    img_mean_sp_noise = mean_filter(img_sp_noise, (5, 5))
    cv2.imwrite("img_mean_sp_noise.jpg", img_mean_sp_noise)

    # median filter
    img_median_gasuss_noise = median_filter(img_gasuss_noise, (5, 5))
    cv2.imwrite("img_median_gasuss_noise.jpg", img_median_gasuss_noise)
    img_median_sp_noise = median_filter(img_sp_noise, (5, 5))
    cv2.imwrite("img_median_sp_noise.jpg", img_median_sp_noise)

    # bilateral filter
    img_bilateral_gasuss_noise = bilateral_filter(
        img_gasuss_noise, (5, 5), 500**2, 150**2)
    cv2.imwrite("img_bilateral_gasuss_noise.jpg", img_bilateral_gasuss_noise)
    img_bilateral_sp_noise = bilateral_filter(
        img_sp_noise, (5, 5), 500**2, 150**2)
    cv2.imwrite("img_bilateral_sp_noise.jpg", img_bilateral_sp_noise)
```