计算机视觉与模式识别

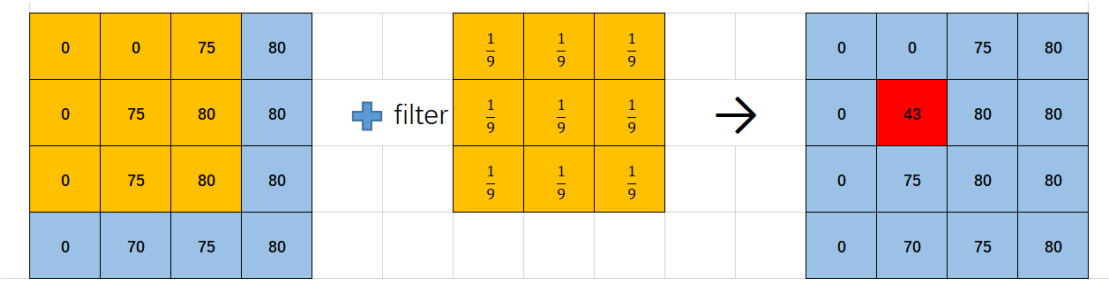
一、实验内容

（1）为图像添加椒盐噪声、高斯噪声；
（2）实现基于高斯滤波、均值滤波、中值滤波的降噪算法，并比较其效果；
（3）思考并回答下列问题：在图像去噪领域外，信号处理、机器学习、深度学习领域还有很多算法都是基于高斯噪声的假设，为什么高斯噪声的假设应用的如此广泛？

二、实验原理

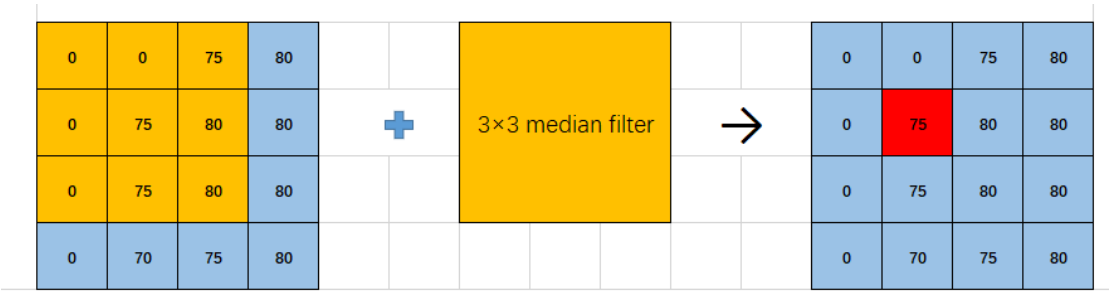
椒盐噪声和高斯噪声是图像中常见的噪声，高斯噪声指服从高斯分布的一类噪声，通常是因为光照不良或者高温引起的传感器噪声；椒盐噪声通常是因为图像传感器、传输信道、解压处理等产生的黑白相间的噪声，通常在灰度图中出现。

图像降噪常常采用的是用滤波器进行处理，滤波器往往根据某一个像素点周围的邻居像素进行计算，赋予该像素点新的值，常用的有高斯滤波器、均值滤波、中值滤波等。一个滤波器通常是 n×n 的矩阵，且 n 是奇数，例如 3×3、5×5 或者 7×7 的滤波器。对于一个有噪声的图片，对图片中的每个像素 P，根据其邻居像素（例如 3×3 的滤波器矩阵，像素 P 则处在该矩阵的中心）计算均值、高斯均值或者取中值，从而达到过滤噪声的效果（图一）。



图一：最左边是一张图片的像素值，中间是均值滤波器，对最左边图像黄色中间的75进行滤波操作，得到最右边的结果（即75取周围3×3像素值的均值变为42.7，取整后为43）

不同图像滤波算法之间的区别主要是滤波器的区别。图一中间是一个 3×3 的均值滤波器，如果是 3×3 的中值滤波器，则是在该 3×3 范围内取中位数；如果是高斯滤波器，则是对 3×3 范围内的像素进行高斯加权平均，高斯滤波器中的值呈现高斯分布，以滤波器的中心点为坐标零点，通过二维高斯分布函数计算滤波器其他点的值（图三）。



图二：最左边是一张图片的像素值，中间是中值滤波器，对最左边图像黄色中间的75进行滤波操作，得到最右边的结果（即75取周围3×3像素值[0,0,0,0,75,75,75,80,80]的中位数75）

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

图三：一个σ=1时的5×5高斯滤波器，
中心点坐标为（0，0），
代入二维高斯分布函数 $\frac{1}{2\pi\sigma^2}e^{\frac{-(x^2+y^2)}{2\sigma^2}}$
中计算得0.15915≈0.159

用滤波器之所以能达到降噪是效果，是因为噪声往往与周围像素有较大的不同，例如椒盐噪声，会有很多突然的黑或者白（0 或 255），与图片中的像素值往往差别很大，而滤波器则根据周围邻居像素进行计算，如果是中值滤波，则很容易就把与众不同的高频噪声去掉，即使是均值滤波，也能让原本不同于周围的像素趋于平滑。

三、实验结果与分析

3.1 算法实现

3.1.1 添加噪声

为图片添加 0 或 255 的椒盐噪声，随机生成坐标点，并随机选取 0 或 255，放入图片中：

```python
def add_salt_pepper_noise(image, proportion):
    '''
    Function: add salt and pepper noise
    image       : input image
    proportion  : noise ratio
    '''
    image_copy = image.copy() # a copy of the gray image
    height, width = image.shape # get image height and width

    # generate random index
    random_Xs = np.random.randint(width, size=(int(proportion*width*height)))
    random_Ys = np.random.randint(height, size=(int(proportion*width*height)))

    # randomly choose 0 or 255 assign to image
    image_copy[random_Ys, random_Xs] = np.random.choice([0, 255], size=(int(proportion*width*height)))
    return image_copy
```

图片四：添加椒盐噪声的 python 代码实现

根据 sigma 的值和高斯函数计算得到一个符合高斯分布的噪声矩阵，加到原图上称为噪声图像：

```python
def add_gaussian_noise(image, noise_mean, noise_sigma):
    '''
    Function: add gaussian noise
    image       : input image
    noise_mean        : the mean of the normal distribution
    noise_sigma       : standard deviation of the nomal distribution
    '''
    image = image / 255.0 # scale pixel value to 0~1
    height, width = image.shape # get image height and width

    # gaussion noise
    gaussian_noise = np.random.normal(noise_mean, noise_sigma, [height, width])
    # add gaussion noise
    gaussian_noised_img = image + gaussian_noise

    # pixel value must be between 0 and 1, clip them if necessary
    gaussian_noised_img = np.clip(gaussian_noised_img, 0, 1)
    # scale pixel value back to 0~255
    gaussian_noised_img = (gaussian_noised_img * 255)
    return gaussian_noised_img
```

图片五：添加高斯噪声的 python 代码实现

3.1.2 滤波器

高斯滤波器的生成比较简单，首先生成坐标矩阵，然后把坐标矩阵代入高斯分布函数计算并归一化得到高斯滤波器：

```python
def gaussian_filter(sigma, kernel_size):
    ''' generate a gaussian kernel given kernel_size and sigma '''
    # generate cordinate axis
    ax = np.linspace(-(kernel_size - 1) / 2, (kernel_size - 1) / 2, kernel_size)
    xx, yy = np.meshgrid(ax, ax)
    # generate gaussian kernel according to gaussian formula
    kernel = np.exp(-0.5 * (np.square(xx) + np.square(yy) ) / np.square(sigma))
    # normalize
    kernel = kernel / np.sum(kernel)
    return kernel
```

图片六：高斯滤波器的 python 实现

均值滤波器更简单，只需要求矩阵内的均值即可：

```python
def mean_filter(kernel_size):
    ''' generate a mean kernel given kernel_size and sigma '''
    # generate cordinate axis
    kernel = np.ones((kernel_size, kernel_size))
    # normalize
    kernel = kernel / np.sum(kernel)
    return kernel
```

图片七：均值滤波器的 python 实现

中值滤波器直接用 numpy.median()实现即可。

3.1.3 卷积操作

首先根据卷积核大小在图像四边进行 0 填充，然后逐个像素进行滤波操作:

```python
def convolution(img, kernel):
    ''' convolution in 2 dimension '''
    radius = kernel.shape[0] // 2 # compute kernel radius
    # padding matrix border
    img_copy = np.pad(img, ((radius, radius), (radius, radius)), 'constant')
    # convolution
    img_convolved = np.zeros((img.shape))
    for i in range(radius, img.shape[0] + radius):
        for j in range(radius, img.shape[1] + radius):
            img_convolved[i - radius, j - radius] = np.sum(img_copy[i-radius:i+radius+1, j-radius:j+radius+1] * kernel)
    return img_convolved
```

图片八：卷积操作的代码实现，根据输入的卷积核进行滤波卷积

计算 PSNR 的值:

```python
def PSNR(original_img, compare_img):
    ''' Peak Signal to Noise Ratio '''
    assert original_img.shape == compare_img.shape , "size of these two must be equal"
    # calculate MSE
    MSE = np.mean(np.square(original_img / 1.0 - compare_img / 1.0))
    PSNR = 10 * np.log10(255*255 / MSE)
    return PSNR
```

图片九：计算 PSNR 的 python 实现

3.2 实验结果与分析

3.2.1 椒盐噪声滤波结果和分析

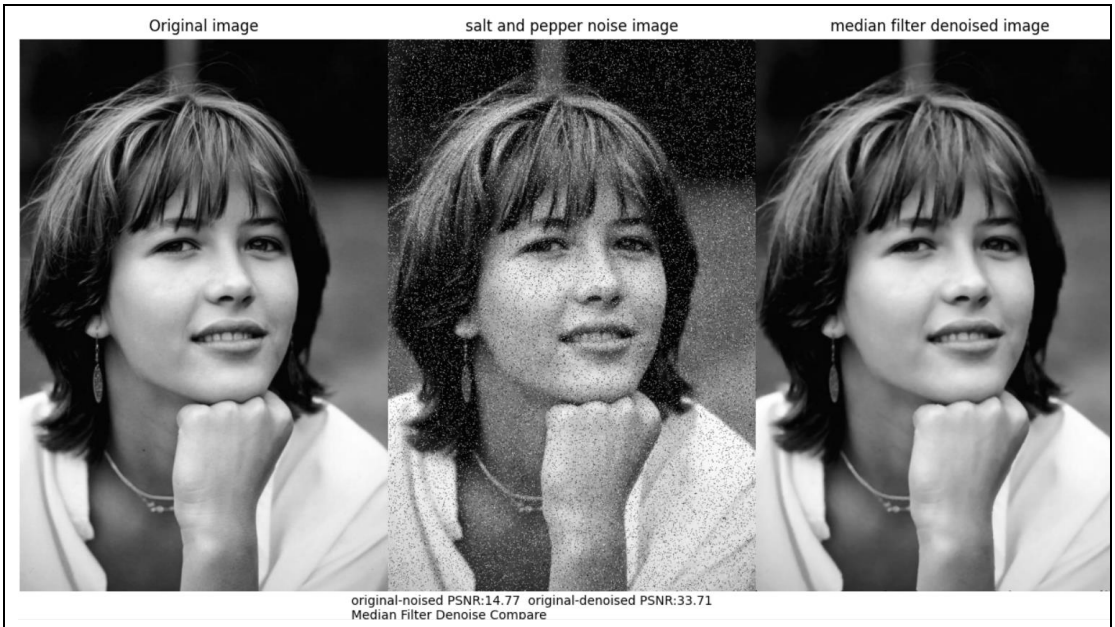
下表（表格 1）展示了 4 张图片分别用高斯滤波、均值滤波、中值滤波去除椒盐噪声的对比结果，其中滤波器大小均为 5×5，图像边缘采用 0 填充，高斯滤波器中 σ =2；并且每个图片下面写有滤波器的类型，以及降噪前和降噪后的 PSNR 数值。

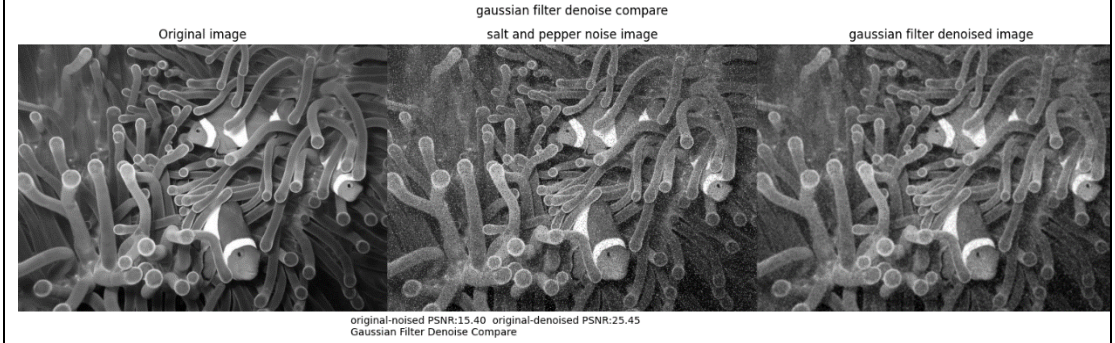
从视觉上来说，中值滤波对椒盐噪声的降噪效果最明显，噪声大部分都去除了，看不出来有噪声，但是也与原图有所不同，变得更平滑了，这是因为中值从 5×5 滤波器中的 25 个数中取中位数，不仅噪声被丢弃了，连原本图像的像素值的起伏也消除了，所以会显得更加平滑并且去掉了大部分噪声；高斯滤波和均值

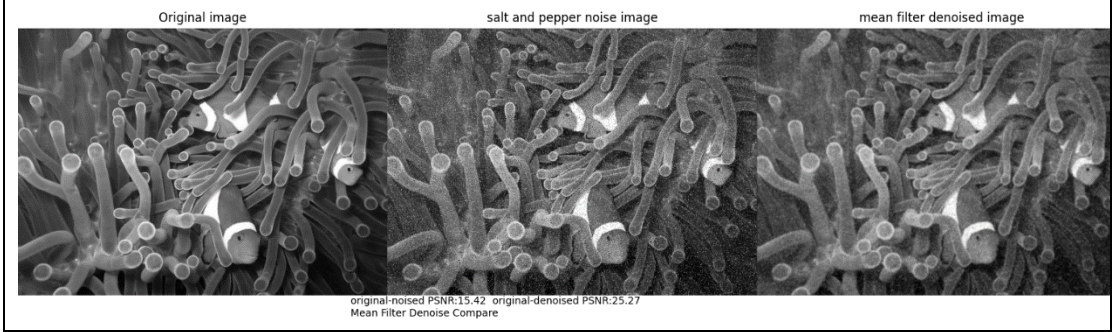
滤波效果差别不大，而且降噪效果也不是直接丢弃了噪声，而是区域像素取均值或者加权均值，这样的效果主要是让噪声没有那么尖锐突兀，但是仍然有一定的痕迹，并且图像本身也会有模糊的效果。

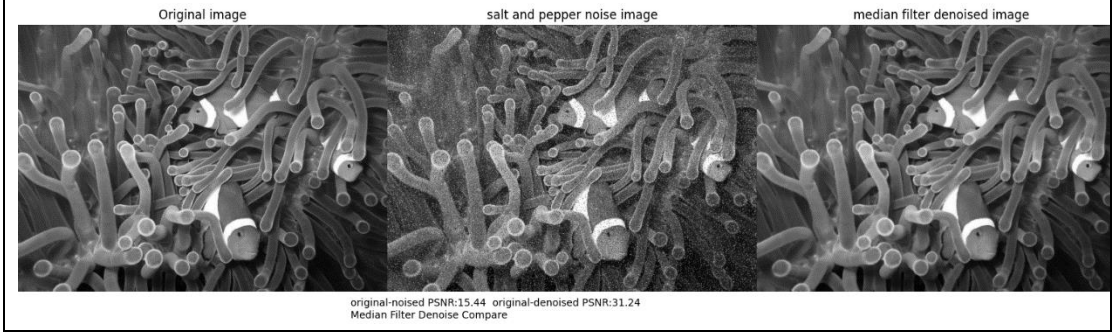
并且从图片下面的 PSNR 值的对比来看，三种滤波器均有一定的降噪效果，并且数值上也与肉眼上看到的直观感觉符合。高斯滤波和均值滤波的 PSNR 增加量基本一样，中值滤波增量则明显比其他方法多不少，效果最好；高斯滤波和均值滤波则接近，与肉眼分析一致。

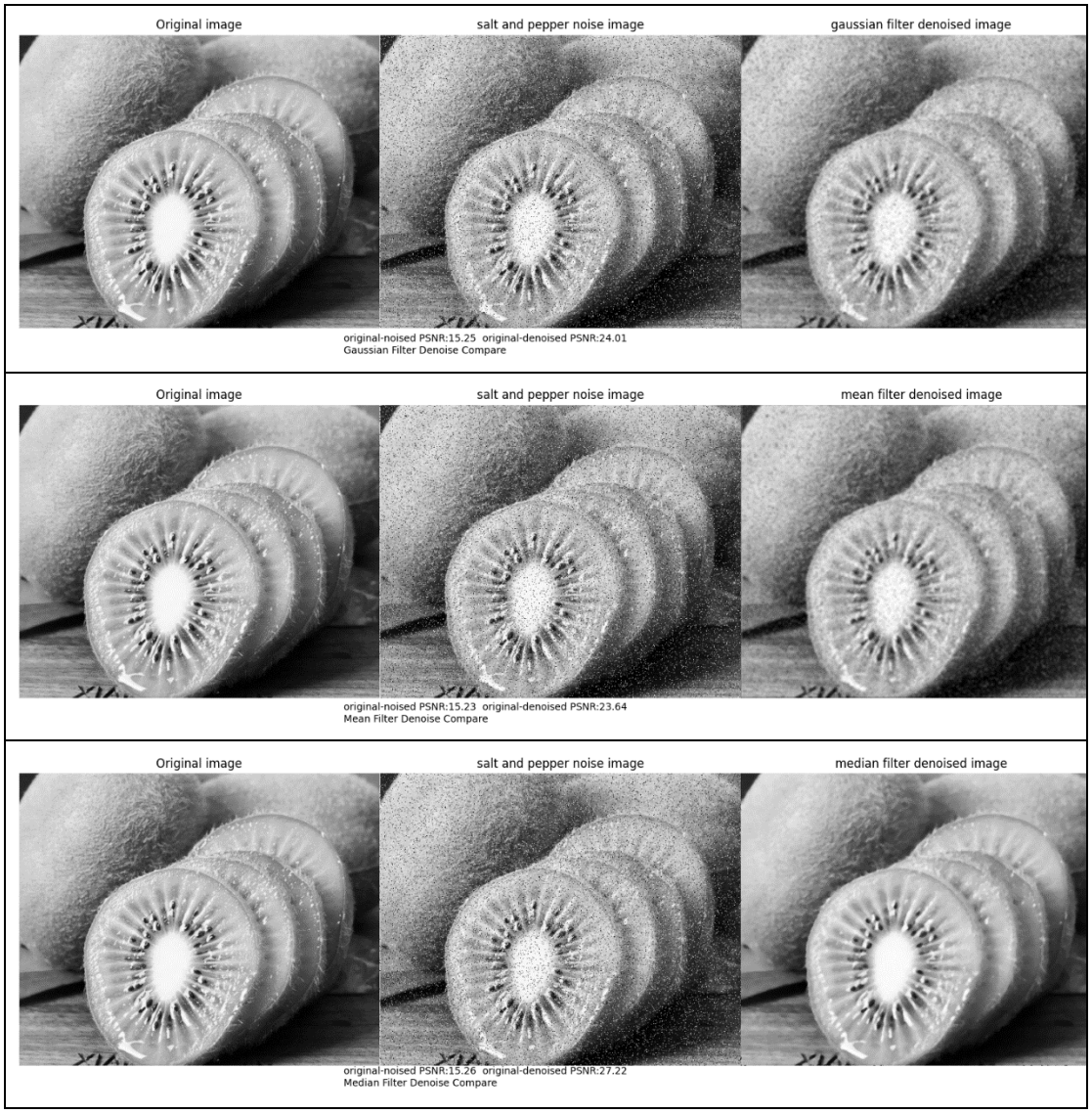

Original image
salt and pepper noise image
gaussian filter denoised image
original-noised PSNR:14.79 original-denoised PSNR:25.23
Gaussian Filter Denoise Compare
Original image
salt and pepper noise image
mean filter denoised image
original-noised PSNR:14.76 original-denoised PSNR:25.17
Mean Filter Denoise Compare


Original image
salt and pepper noise image
median filter denoised image
original-noised PSNR:14.77 original-denoised PSNR:33.71
Median Filter Denoise Compare


gaussian filter denoise compare
Original image
salt and pepper noise image
gaussian filter denoised image
original-noised PSNR:15.40 original-denoised PSNR:25.45
Gaussian Filter Denoise Compare


Original image
salt and pepper noise image
mean filter denoised image
original-noised PSNR:15.42 original-denoised PSNR:25.27
Mean Filter Denoise Compare


Original image
salt and pepper noise image
median filter denoised image
original-noised PSNR:15.44 original-denoised PSNR:31.24
Median Filter Denoise Compare



表格 1：高斯滤波、均值滤波、中值滤波对椒盐噪声进行去噪效果对比图

3.2.2 高斯噪声滤波结果和分析

下表（表格 2）展示了 4 张图片分别用高斯滤波、均值滤波、中值滤波去除椒盐噪声的对比结果，其中滤波器大小均为 5×5，图像边缘采用 0 填充，高斯滤波器中 σ =2；并且每个图片下面写有滤波器的类型，以及降噪前和降噪后的 PSNR 数值。

对于高斯噪声的降噪效果，则与椒盐噪声有所不同，椒盐噪声中高斯滤波和均值滤波效果基本一致，而高斯噪声则表现有所不同。均值滤波的降噪效果比高斯滤波更好一点，这是因为高斯滤波器跟 σ 有关，σ 越大，中心像素权值就越小，其周围的权值越大，这样参考周围像素的比例更多，加权均值下来考虑的像素周围原本的特征更多，降噪就更好；但无论 σ 如何变换，不可能像均值滤波那样所有点权重一样，故不如均值滤波那样考虑周围像素多，降噪效果就稍差于均值滤波；另外，高斯滤波和均值滤波均会让图片变模糊，而且高斯滤波 σ 越大，图像

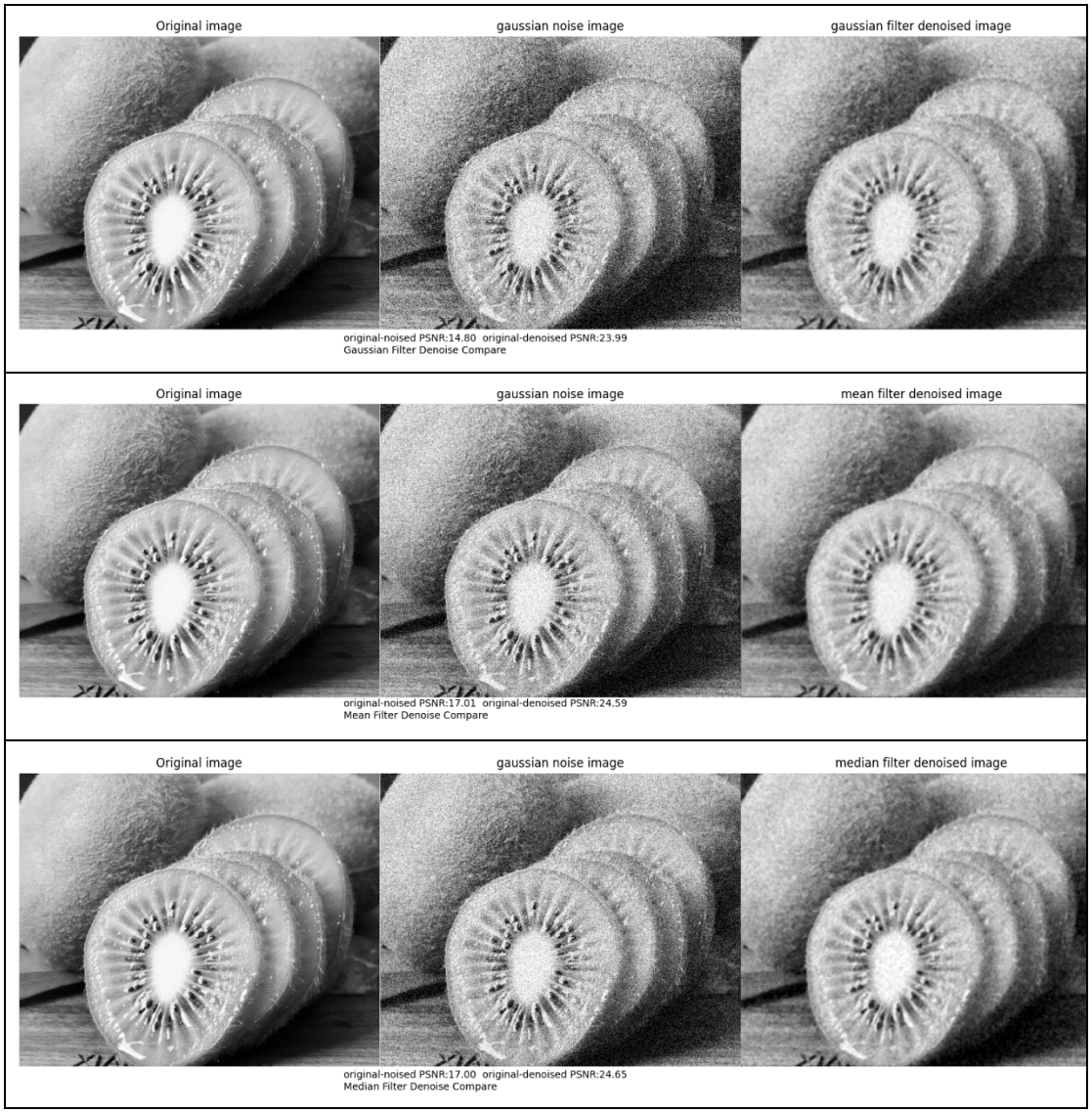
就会变得越模糊。

即使是在椒盐噪声中表现良好的中值滤波，处理高斯噪声时也没有那么显著了，这是因为高斯噪声分布的范围非常广，影响的像素比例非常大，甚至所有像素都有噪声干扰，故即使是利用中值丢掉了不少像素值，依然有一定比例的噪声存在，但依然比其他两种效果稍好。

从 PSNR 值的表现来看，符合上述分析，中值滤波的 PSNR 值提升最高，均值滤波次之，最后是高斯滤波。综上，三种滤波器在高斯噪声中的降噪效果差别不大，但依然有差别，降噪效果：中值滤波 > 均值滤波 > 高斯滤波。


Original image
gaussian noise image
gaussian filter denoised image
original-noised PSNR:15.19 original-denoised PSNR:24.81
Gaussian Filter Denoise Compare
Original image
gaussian noise image
mean filter denoised image
original-noised PSNR:17.45 original-denoised PSNR:26.79
Mean Filter Denoise Compare


Original image
gaussian noise image
median filter denoised image
original-noised PSNR:17.44 original-denoised PSNR:27.28
Median Filter Denoise Compare
Original image
gaussian noise image
gaussian filter denoised image
original-noised PSNR:14.65 original-denoised PSNR:25.33
Gaussian Filter Denoise Compare
Original image
gaussian noise image
mean filter denoised image
original-noised PSNR:16.88 original-denoised PSNR:26.43
Mean Filter Denoise Compare
Original image
gaussian noise image
median filter denoised image
original-noised PSNR:16.88 original-denoised PSNR:25.96
Median Filter Denoise Compare



表格 2：高斯滤波、均值滤波、中值滤波对高斯噪声进行去噪效果对比图

四、总结

4.1 思考题

在图像去噪领域外，信号处理、机器学习、深度学习领域还有很多算法都是基于高斯噪声的假设，为什么高斯噪声的假设应用的如此广泛？

答：现实生活中，噪声值为 0 或者 255 左右的椒盐噪声往往很少，不可能一个图片遍布大量椒盐噪声，因为椒盐噪声往往是影像讯号受到突如其来的强烈干扰导致模数转换器或者位元传输错误，如失效的感应器导致值为最小值，饱和的感应器导致像素值为最大值。而高斯噪声则不一样，受到的影响因素很多，比如亮度不够均匀、各个电路元件自身噪声和相互影响、图像传感器长期工作导致温度升高，这些比较小的干扰即可导致噪声，不像椒盐噪声那样主要是受强烈干扰才产生，而且这些干扰往往呈现出高斯分布，故容易导致高斯噪声的产生。

故现实生活中的噪声大部分都是高斯噪声而不是椒盐噪声，所以信号处理、机器学习、深度学习等领域很多基于高斯噪声的假设进行处理。

4.2 总结思考

滤波器是图像处理领域的经典算法和工具，即使现在的卷积神经网络依然适用，了解和熟悉它的适用非常有意义，滤波器除了降噪，还能提取竖直、水平等边缘特征或者其他特征，这里主要熟悉降噪滤波的原理以及操作和效果。

中值滤波对椒盐噪声的降噪效果最明显，噪声大部分都去除了，看不出来有噪声，但是也与原图有所不同，变得更平滑了，对高斯噪声的降噪效果没有椒盐噪声好，但是依然比其他两种滤波器效果好；高斯滤波和均值滤波处理椒盐噪声效果差别不大，而且降噪效果也不是直接丢弃了噪声，而是区域像素取均值或者加权均值，这样的效果主要是让噪声没有那么尖锐突兀，但是仍然有一定的痕迹，并且图像本身也会有模糊的效果，但是在高斯噪声中均值滤波略微优于高斯滤波器。