编程实现小波域维纳滤波

图像小波变换:

python 里面的 pywavelets 包进行的变换是将图像分解成了四个部分，具体是 Approximation, horizontal detail, vertical detail and diagonal detail coefficients respectively. 其中第一部分对应的是低频部分，其余的是高频信号，第二个是水平部分，第三个是垂直方向的，最后一部分是对角线形式的分解。

维纳滤波:

维纳滤波类似于求解一个最大后验概率，这里对于后验概率中的概率估计都是以高斯模型的形式进行估计的，最后使用的是求导数的方法求得最大的原来的图像 X_i

其中，对于有噪声的部分的条件概率的方差的估计，是使用的对于 HH 部分，即 diagonal detail coefficients 部分的数值进行估计:

$$\sigma_n = \frac{meadian(|HH|)}{0.6745}$$

然后对于实际的想要滤波得出来的结果的 X_i 的方差的估计使用输入的有噪声的图像 $Y(i)$ 进行估计:

$$\sigma^2 = \frac{1}{M}\sum_i Y(i)^2 - \sigma_n^2$$

小波域的维纳滤波:

现将图像进行小波分解，对于高频部分进行维纳滤波处理，最后再进行反变换变回原来的小波图像，经过了一次分解之后的情况如下:

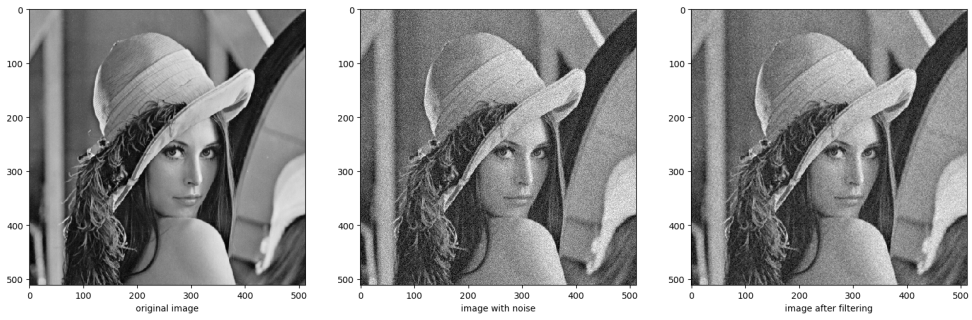

图 1: 只经过一次分解，然后进行小波域维纳滤波的图像

这样效果实际上并不好，从图中可以比较直观的看出来，所以我参考网上其他人写的维纳滤波算法，尝试进行多尺度的金字塔形式的小波分解，然后对于每一尺度进行维纳滤波，最后组合起来，参考网上资料，这里使用的是类似于 PPT 中提到的 Mallat 算法类似金字塔结构的小波分解，然后使用递归的思路进行维纳滤波，效果如下 (5 层的小波分解):

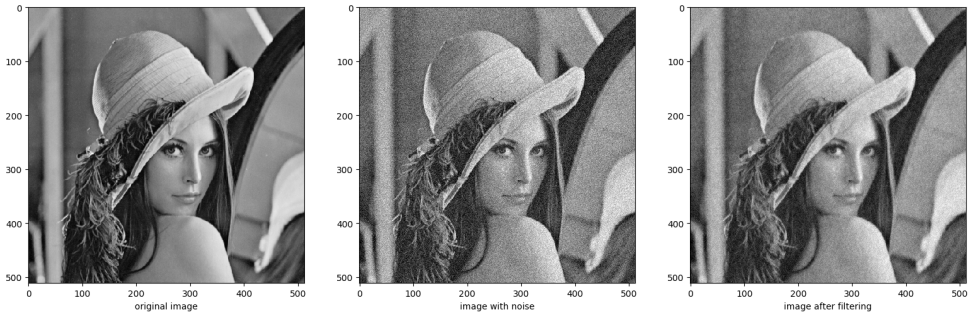

图 2: 只经过一次分解，然后进行小波域维纳滤波的图像

可以看到,效果实际上好了一点,细看的话,图像实际上是变模糊的,但是确实相对而言起到了滤波的效果。

附录:

代码 waveletwiener.py 直接运行就可以

需要环境变量:

numpy 1.16.5

opencv-python 4.1.1.26

matplotlib 3.1.1

pywavelets 1.1.1

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


def wiener_filter(img,HH):
    """ wiener_filter
    wiener_filter of the wavelet-wiener part

    args:
        img(np.array): the input image Y(i) with noise
        HH(np.array): the image HH with sigma of the normal noise in the wavelet compos

    return:
        filter_image(np.array): X(i) the output image (in the process of wavelet transfo
    """
    noise_sigma = (np.median(np.abs(HH))/0.6745)**2  # according to the PPT, estimate th
    img_sigma = 1/(img.shape[0]*img.shape[1])*(np.sum(img*img)-noise_sigma)
# calculate the sigma**2
    filter_image = img_sigma/(img_sigma+noise_sigma)*img  # calculate the output Xi afte

    return filter_image



def wiener_wavelet(img,times=5):
    """wiener in the wavelet space
    the wiener filter in the wavelet space based on a pyramid structure

    args:
        img(np.array): the input image
        times(int): times of wiener filtering

    return:
```

```
        filter_image(np.array): the output image without noise
    """
    img = pywt.dwt2(img, 'bior4.4')
    LL, (LH, HL, HH) = img
    LH = wiener_filter(LH, HH)
    HL = wiener_filter(HL, HH)
    HH = wiener_filter(HH, HH)
    if times>0:
        LL = wiener_wavelet(LL,times-1)
        row, col = LL.shape
        # take care, the wavelet pywt may affect the image's shape, try to fix it
        # referring to https://blog.csdn.net/qq_36293056/article/details/113575634
        d_row = row - HH.shape[0]
        d_col = col - HH.shape[1]
        if d_row > 0 or d_col > 0:
            d_row = row - np.arange(d_row) - 1
            d_col = col - np.arange(d_col) - 1
            LL = np.delete(LL, d_row, axis=0)
            LL = np.delete(LL, d_col, axis=1)
    filter_image = pywt.idwt2((LL, (LH, HL, HH)), 'bior4.4')

    return filter_image



def add_noise(img, sigma):
    """ add noise for the image
    add noise for testing the wavelet-wiener

    args:
        img(np.array): the input image
        sigma(float): the sigma of the normal noise

    return:
        img(np.array): the output image with noise
    """
    row, col = img.shape
    img_noise = img + np.random.randn(row,col)*sigma  # add the gaussin noise
    img_noise = np.where(img_noise <= 0, 0, img_noise)
    img_noise = np.where(img_noise > 255, 255, img_noise)
```

```
    return np.uint8(img_noise)


if __name__ == '__main__':

    img = cv2.imread('lena_gray_512.tif',cv2.IMREAD_GRAYSCALE)

    plt.subplot(1,3,1)
    plt.imshow(img,cmap='gray')
    plt.xlabel("original image")  # the orignal image

    img_noise = add_noise(img, 20)
    plt.subplot(1,3,2)
    plt.imshow(img_noise,cmap='gray')
    plt.xlabel("image with noise")  # the image with noise

    img_filter = wiener_wavelet(img_noise,4)
    img_filter = np.where(img_filter < 0, 0, img_filter)
    img_filter = np.where(img_filter > 255, 255, img_filter)
    img_filter = np.uint8(img_filter)
    plt.subplot(1,3,3)
    plt.imshow(img_filter,cmap='gray')
    plt.xlabel("image after filtering")  # the image after filtering

    plt.show()
```