数字图像处理第四次作业

姓名: 刘贵涛

班级: 自动化钱 71

学号: 2176110040

提交日期: 2020/03/25

摘要：本次实验通过 python，首先对 test1 和 test2 进行低通滤波，再对 test3 和 test4 分别应用 Sebel、Laplace、Canny 算法进行边缘检测。

一．空域低通滤波器：分别用高斯滤波器和中值滤波器去平滑测试图像 test1 和 2，模板大小分别是 3x3，5x5，7x7；分析各自优缺点；

分析 test1 和 test2 原图，如图 1 所示：

图 1 test1 和 test2 原图

如图 1 所示，test1 中存在椒盐噪声，存在肉眼可见的白色像素点。而 test2 属于正常图像。

首先使用高斯滤波器平滑 test1，结果如图 2 所示：

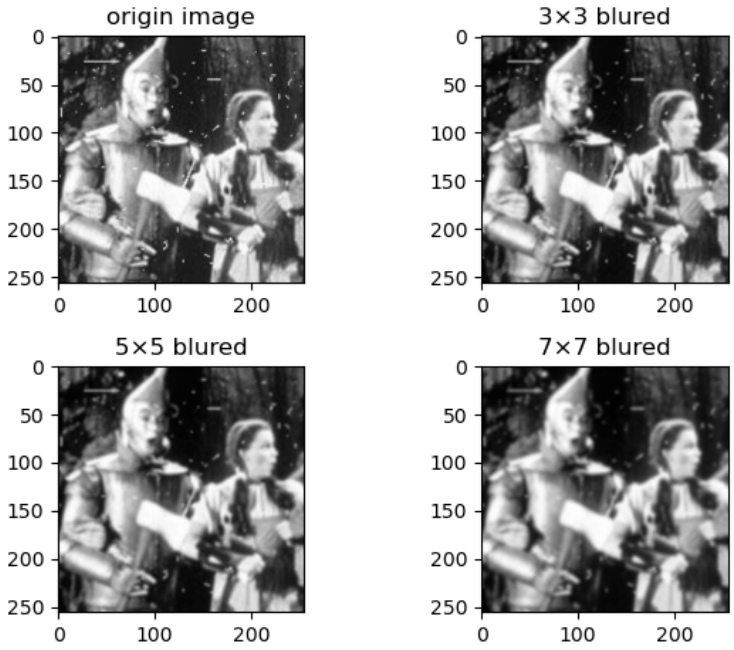

图 2 test1 图像分别使用不同大小的高斯滤波器平滑结果

如图 2 所示，高斯滤波器不能很好的去除椒盐噪声，对于处理后的图像来说，边沿的锯齿感减小，同时对于小面积的椒盐噪声能够有一定的去除效果，但是在模板增大之后，明显的可以看见图片边界模糊，细节减少。对 test2 同样使用相同的高斯平滑处理如图 3 所示：

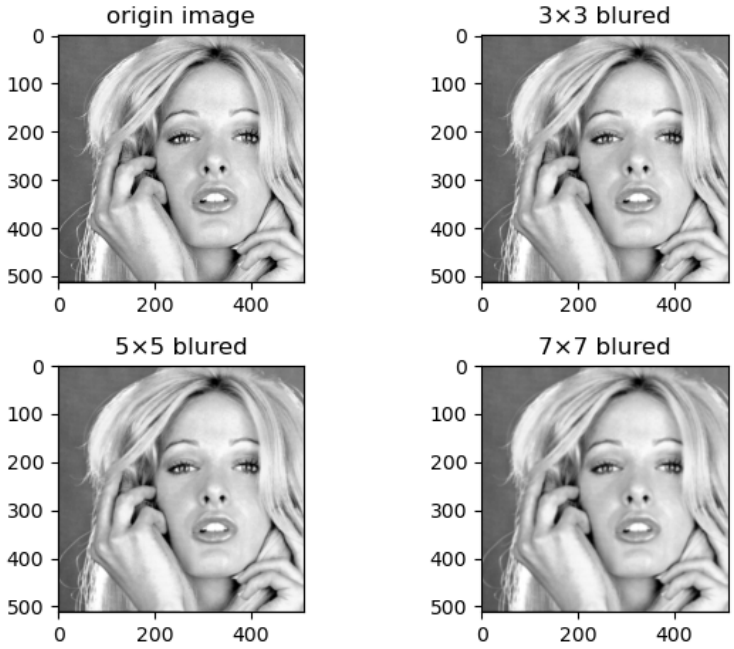

图 3 test2 经高斯平滑结果

如图 3 所示，test2 经高斯滤波器处理之后，对于整个图像来说，总体内容并未改变，可以看见细节处如发丝被平滑模糊，同时面部的纹理也被平滑，整体图片被模糊化。

接下来对 test1 使用不同大小的中值滤波器处理，结果如图 4 所示：

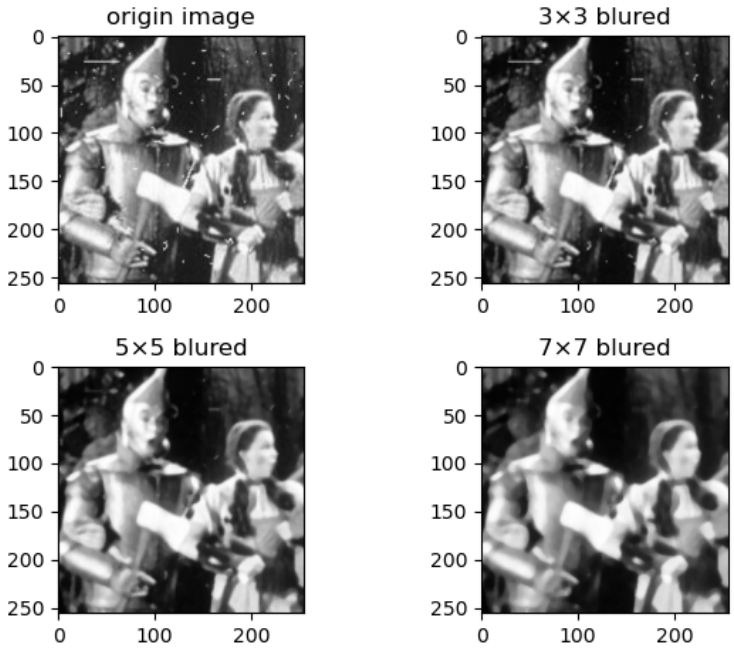

图 4 test1 中值滤波器平滑结果

如图 4 所示，对于这个图来说，中值滤波器能够很好的去除椒盐噪声，但是中值滤波器带来的是更加严重的边沿模糊效应，其中 5×5 大小的中值滤波器基本上去除了途中所有的盐噪声，但是此时图片已经非常模糊。到了 7×7 大小时，图片很多内容已经不可辨识，很多细节内容已经丢失。

同时对 test2 使用不同大小的中值滤波器进行处理，结果如图 5 所示：

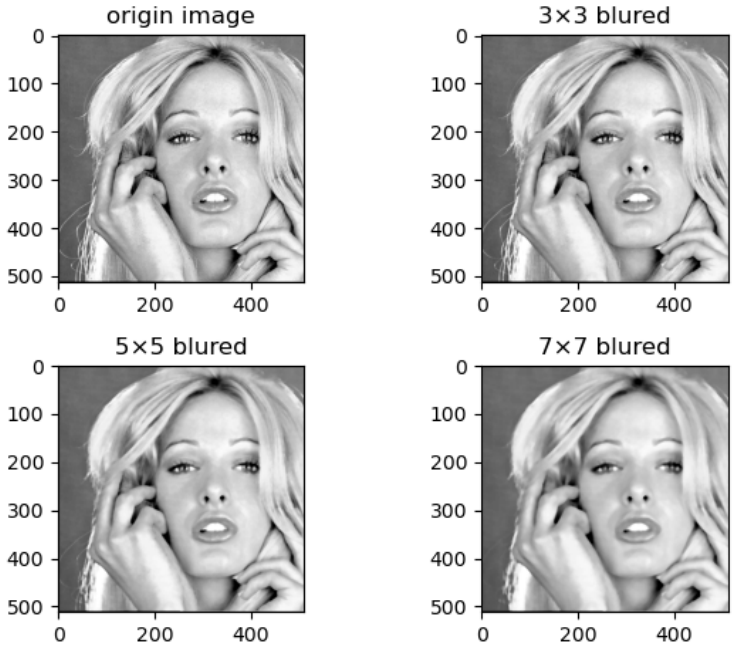

图 5 test2 经过中值滤波器处理结果

如图 5 所示，中值滤波器在去除背景中的噪声的同时，也把很多前景内容当成噪声处理掉了，比如左下角看起来像盐噪声的的细节内容被中值滤波器平滑掉了，根据对比可知中值滤波器能够更强的去除椒盐噪声，但是带来的是更多的图片细节丢失，应根据所追求的滤波处理目标合理选择滤波器。

二．利用固定方差 sigma=1.5 产生高斯滤波器.

高斯滤波器使用高斯函数产生，标准高斯函数为：

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

则对于一个 3×3 的模板，使用高斯中心化处理之后，设置$\sigma = 1.5$使用该式计算对应点的值再缩放到整数值，得到

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

同理计算得到 5×5，7×7 的高斯滤波器得：

$$\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 20 & 33 & 20 & 4 \\ 7 & 33 & 55 & 33 & 7 \\ 4 & 20 & 33 & 20 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 12 & 55 & 90 & 55 & 12 & 1 \\ 12 & 148 & 665 & 1097 & 665 & 148 & 12 \\ 55 & 665 & 2981 & 4915 & 2981 & 665 & 55 \\ 90 & 1097 & 4915 & 7103 & 4915 & 1097 & 90 \\ 55 & 665 & 2981 & 4915 & 2981 & 665 & 55 \\ 12 & 148 & 665 & 1097 & 665 & 148 & 12 \\ 1 & 12 & 55 & 90 & 55 & 12 & 1 \end{pmatrix}$$

对于上述三个滤波器，唯一区别在于模板大小，模板越大会导致处理后的图像更平滑即更加模糊，细节丢失更多，但是会滤去更多的噪音。

三．利用高通滤波器滤波测试图像 test3,4：包括 unsharp masking, Sobel edge detector, and Laplace edge detection；Canny algorithm.分析各自优缺点；

首先分析 test3，test4 两张图片

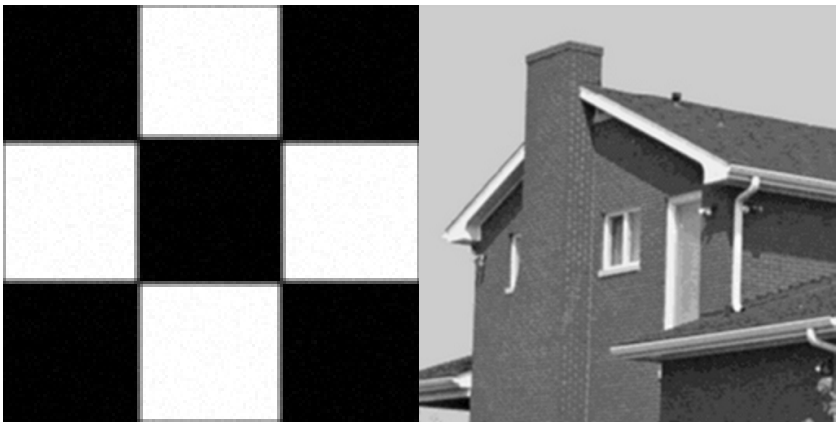

图 6 test3 和 test4 图片

如图 6 所示，test3 为黑白块，拥有明确的边界。而 test4 图为一个拥有明显边界的建筑物，均待使用边界检测算法进行检测。

首先使用 unsharp masking 算法，依靠原图和低通滤波器如高斯滤波器处理结果之差，对原图像进行增强。

$$G(x,y) = GaussianBlur\big(f(x,y)\big)$$

$$H(x,y) = f(x,y) + \beta\big(f(x,y) - G(x,y)\big)$$

分别对 test3 进行处理，处理结果如图 7 所示：

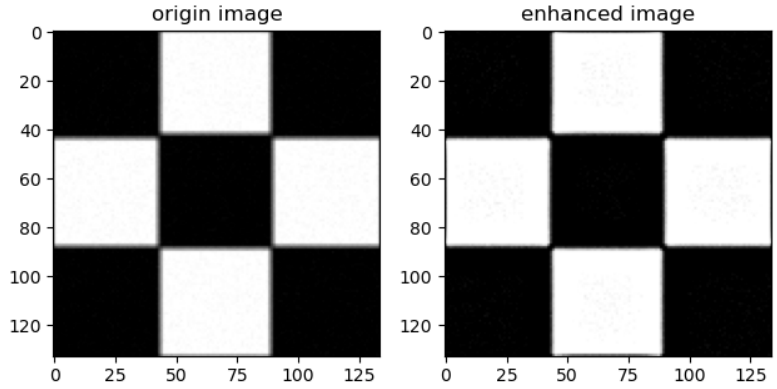

图 7 test3 经过 unsharp masking 处理结果

如图 7 所示，右图明显边界处被锐化，黑色和白色的交界处显得更加突出，边界得到了增强。

同时对 test4 使用 unsharp masking 处理，处理结果如图 8 所示：

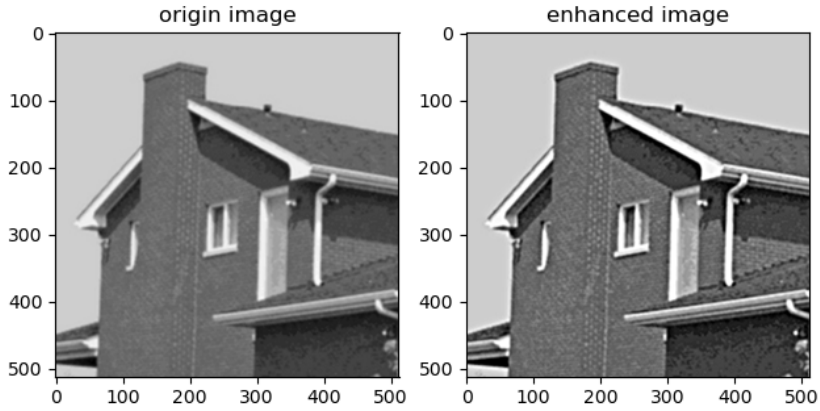

图 8 unsharp masking 处理 test4 结果

如图 8 所示，处理后的右图明显边界和细节得到了提升，边界得到了检测增强。

接下来分别使用 Sobel edge detecting algorithm、Laplace edge detecting algorithm、Canny edge detecting algorithm 对 test3 和 test4 进行边缘检测。首先是 test3 的检测结果如图 9 所示：

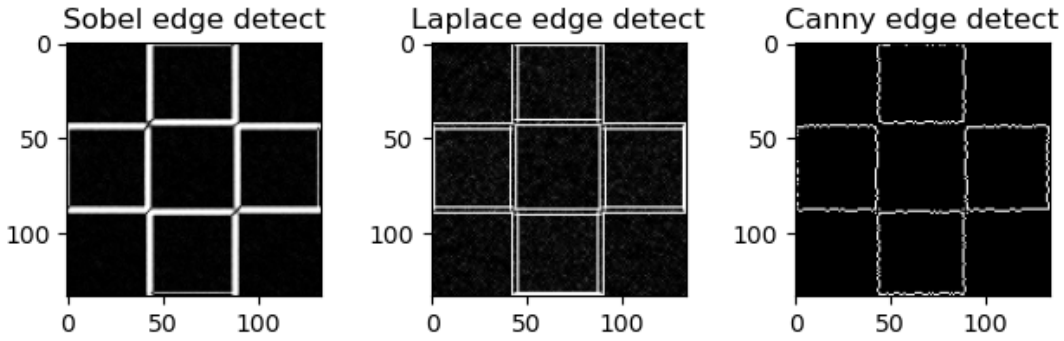

图 9 test3 边缘检测结果

对于 Sobel edge detect 是一种一维处理算法，分别对 x、y 方向上，进行差分处理，Sobel 卷积因子为：

$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

之后再将 x、y 方向上的梯度通过

$$f(x,y)=\sqrt{w_x^2+w_y^2}$$

得到最终边缘检测结果如图 9 中左图所示，可以看出，这种检测方式下，可以有效的检出边缘。

第二种方式是 Laplace edge detect algorithm，拉普拉斯算子使用二阶差分进行边缘检测，其卷积因子为：

$$\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}$$

对 test3 处理之后结果如图 9 中图所示，由于是二阶微分，则可以发现边缘处出现了两个峰值，可以看出能够有效的检测出边缘。

第三种方式是 Canny edge detecting algorithm，显示用高斯滤波器平滑图像，使用一阶差分计算梯度值和倒数方向，对结果使用非最大值抑制算法得到薄边缘的二值图片，再设定两个阈值 minVal 和 maxVal ，任何边缘的强度梯度大于 maxVal 的确定为边缘；而小于 minVal 的确定为非边缘，并丢弃。位于 maxVal 和 minVal 阈值间的边缘为待分类边缘，或非边缘，基于连续性进行判断. 如果边缘像素连接着 "确定边缘" 像素，则认为该边缘属于真正边缘的一部分；否则，丢弃该边缘。这里取 minVal = 50， maxVal = 150 结果如图 9 中右图所示，可以看出该算法能够有效检测出边缘并且边缘相对于前两种算法更细。

同样的对 test4 进行上述三种边缘提取策略，结果如图 10 所示：

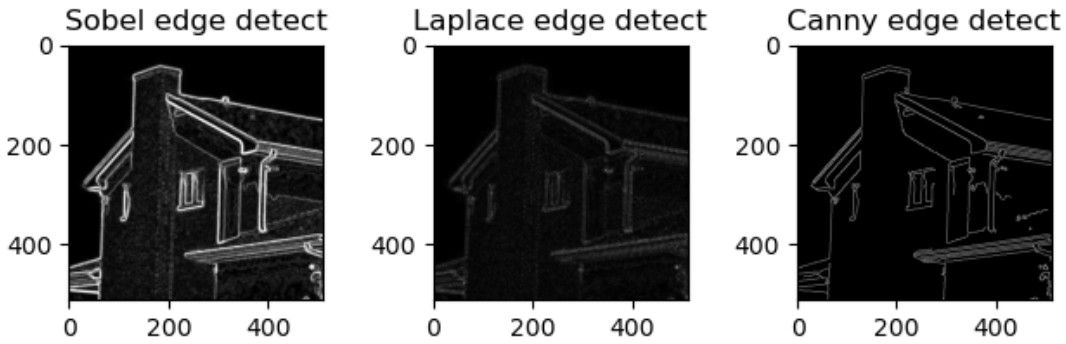

图 10 test4 边缘检测结果

这三种检测算法，Sobel 算法能够检测出边缘的粗细、强弱，抗噪型好，Laplace 算法对边缘敏感但是对噪音会更加敏感。Canny 产生的边缘很细，同时不能检测出边缘的强弱。

参考文献：

[1] Wikipedia contributors, "Median filter," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Median_filter&oldid=931347301 (accessed March 25, 2020).

[2] Wikipedia contributors, "Gaussian blur," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Gaussian_blur&oldid=927938738 (accessed March 25, 2020).

[3] Wikipedia contributors, "Sobel operator," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=946480613 (accessed March 25, 2020).

[4] Wikipedia contributors, "Edge detection," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Edge_detection&oldid=944663479 (accessed March 25, 2020).

代码：

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

def median_blur(img):
    imgs = {}
    for i, n in enumerate([3, 5, 7]):
        imgs[i+1] = cv2.medianBlur(img, n)
    return imgs


def gaussian_blur(img):
    imgs = {}
    for i, n in enumerate([(3, 3), (5, 5), (7, 7)]):
        imgs[i+1] = cv2.GaussianBlur(img, n, 1.5)
    return imgs


def show_comparison(img_ori, imgs_blured):
    imgs = {0: img_ori}
    imgs.update(imgs_blured)
    plt.figure()
    for i in range(len(imgs)):
        plt.subplot(2, 2, i+1)
        title = 'origin image' if i == 0 else str(2*i+1) + '×' + str(2*i+1) + ' blured'
        plt.title(title)
        plt.imshow(imgs[i], cmap='gray')
    plt.show()


if __name__ == '__main__':
    test1 = cv2.imread('images/test1.pgm', 0)
    # test1_median_blur = median_blur(test1)
    test1_gaussian_blur = gaussian_blur(test1)
    # show_comparison(test1, test1_median_blur)
    show_comparison(test1, test1_gaussian_blur)

    test2 = cv2.imread('images/test2.tif', 0)
    # test2_median_blur = median_blur(test2)
    test2_gaussian_blur = gaussian_blur(test2)
    # show_comparison(test2, test2_median_blur)
    show_comparison(test2, test2_gaussian_blur)
```

```
import numpy as np


def gaussian_func(x, y, sigma=1):
    return 1 / (2 * np.pi * sigma**2) * np.exp(- (x**2 + y**2) / (2 * sigma**2))


def mask_generate(k, mask='gaussian'):
    mask_out = np.zeros((k, k))
    if mask=='gaussian':
        for i in range(len(mask_out)):
            for j in range(len(mask_out[i])):
                x, y = j-len(mask_out[i])//2, i-len(mask_out)//2
                mask_out[i, j] = gaussian_func(x, y)

    return np.round(mask_out/mask_out[0, 0]).astype(np.int)


mask_3 = mask_generate(3)
mask_5 = mask_generate(5)
mask_7 = mask_generate(7)

if __name__ == "__main__":
    print(mask_3)
    print(mask_5)
    print(mask_7)


import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.filters import unsharp_mask

def unsharp_masking(img, do_plt=True):
    img_out = unsharp_mask(img, radius=5)

    # img_blur = cv2.GaussianBlur(img, (5, 5), 0)
    # img_out = cv2.addWeighted(img, 2, img_blur, -1, 0)
    if do_plt:
        plt.figure()
        plt.subplot(121)
        plt.imshow(img, cmap='gray')
        plt.title('origin image')
```

```
            plt.subplot(122)
            plt.title('enhanced image')
            plt.imshow(img_out, cmap='gray')
            plt.show()
        return img

def high_pass_filter(img):
        img_sobel_x_edge_detect = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
        img_sobel_y_edge_detect = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
        img_sobel_edge_detect = cv2.convertScaleAbs(
            np.sqrt(img_sobel_x_edge_detect**2 + img_sobel_y_edge_detect**2)
        )

        img_laplace_edge_detect = cv2.convertScaleAbs(cv2.Laplacian(img, cv2.CV_64F,
ksize=3))
        img_canny_edge_detect = cv2.convertScaleAbs(cv2.Canny(img, 50, 150))

        plt.figure()
        plt.subplot(131)
        plt.title('Sobel edge detect')
        plt.imshow(img_sobel_edge_detect, cmap='gray')
        plt.subplot(132)
        plt.title('Laplace edge detect')
        plt.imshow(img_laplace_edge_detect, cmap='gray')
        plt.subplot(133)
        plt.title('Canny edge detect')
        plt.imshow(img_canny_edge_detect, cmap='gray')
        plt.show()


if __name__ == '__main__':
        test3 = cv2.imread('images/test3_corrupt.pgm', cv2.IMREAD_GRAYSCALE)
        test4 = cv2.imread('images/test4.tif', cv2.IMREAD_GRAYSCALE)
        # unsharp_masking(test3)
        # unsharp_masking(test4)

        high_pass_filter(test3)
        high_pass_filter(test4)
```