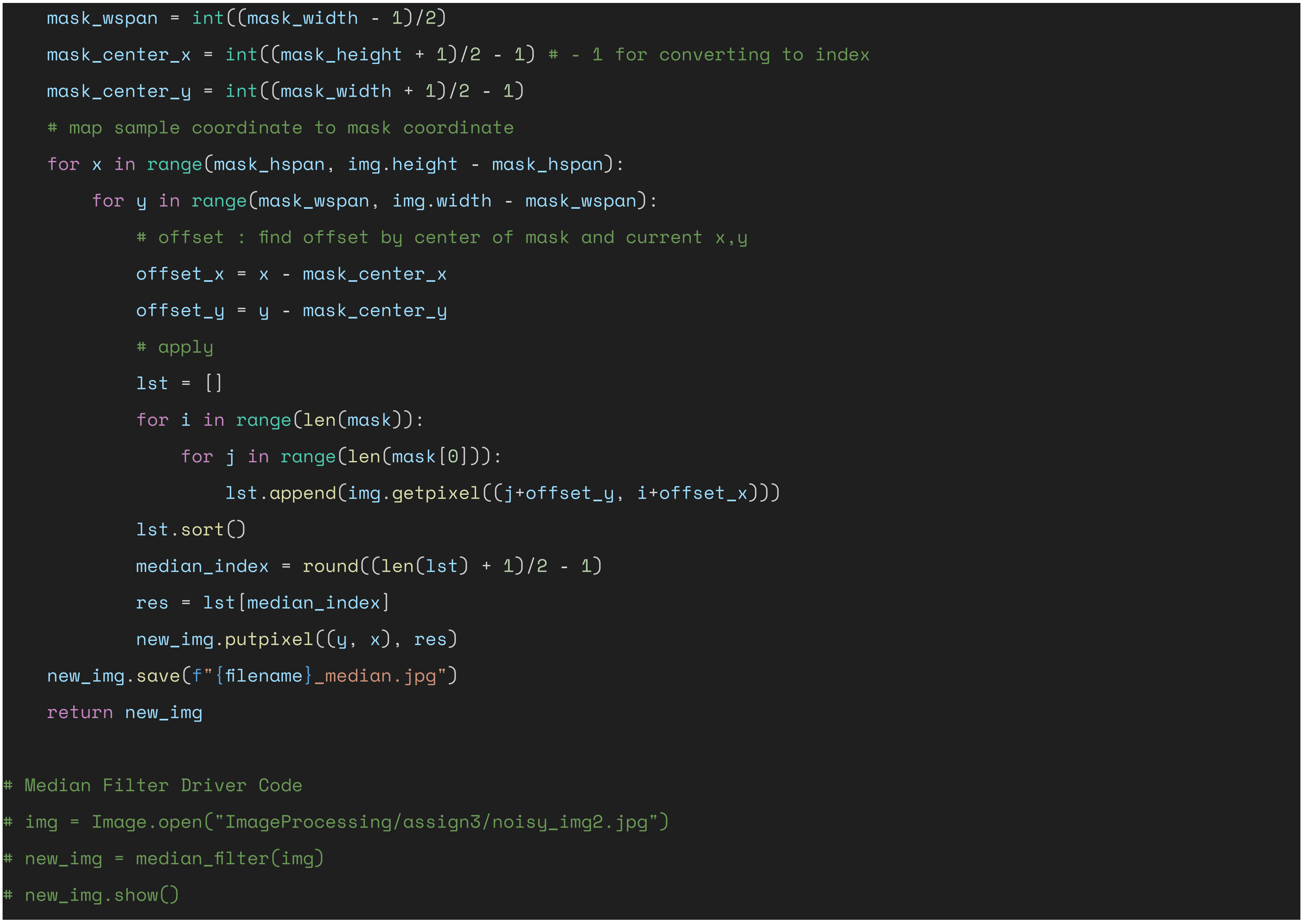
พสิษฐ์ โวศรี 6434453423

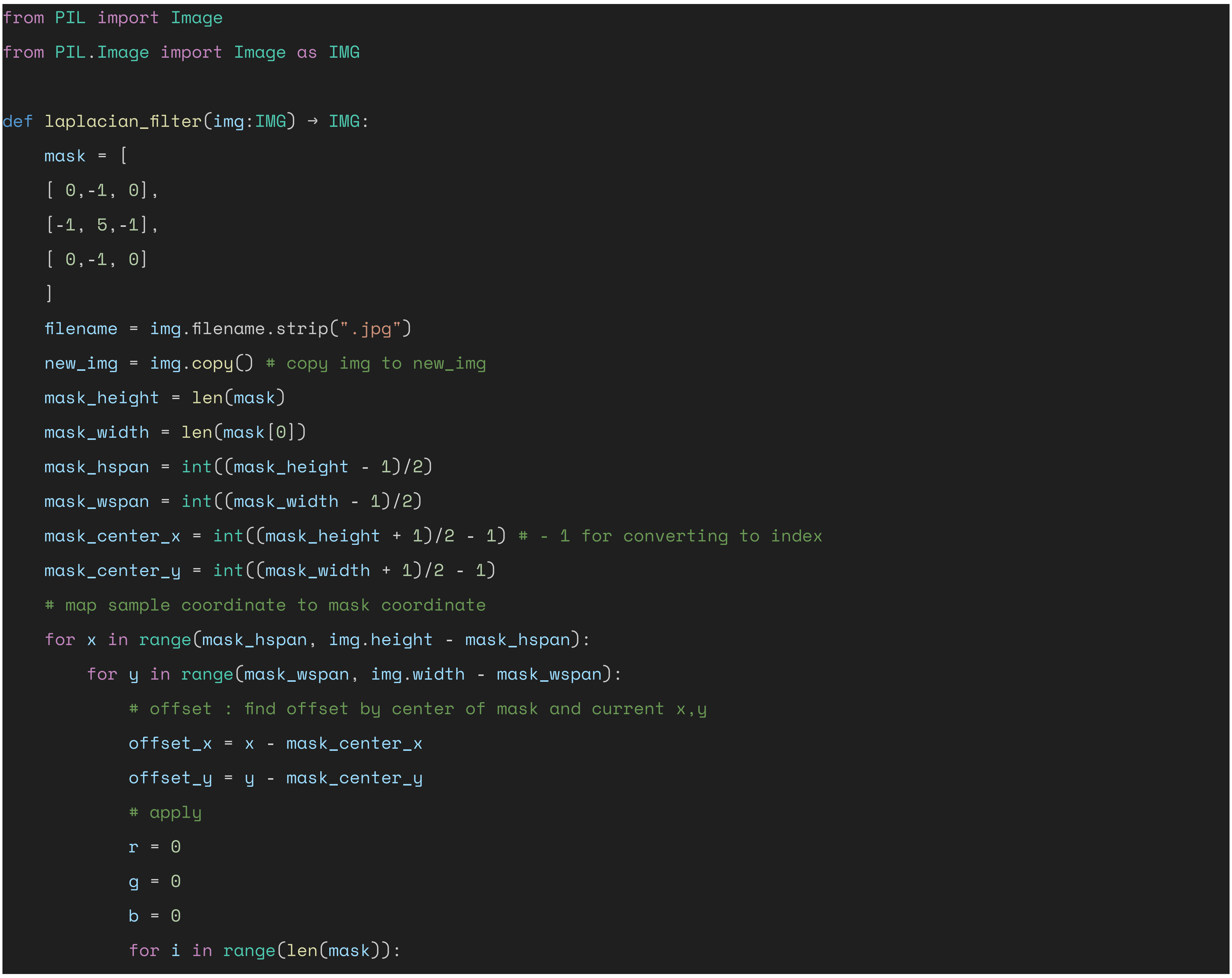
1.Smoothing Filter

```
from PIL import Image
from PIL.Image import Image as IMG

def averaging_filter(img:IMG) -> IMG:
    mask = [
    [1,1,1],
    [1,1,1],
    [1,1,1]
    ]
    constant = len(mask) * len(mask[0])
    filename = img.filename.strip(".jpg")
    img = img.convert("L") # convert to grayscale
    new_img = img.copy() # copy img to new_img
    mask_height = len(mask)
    mask_width = len(mask[0])
    mask_hspan = int((mask_height - 1)/2)
    mask_wspan = int((mask_width - 1)/2)
    mask_center_x = int((mask_height + 1)/2 - 1) # - 1 for converting to index
    mask_center_y = int((mask_width + 1)/2 - 1)
    # map sample coordinate to mask coordinate
    for x in range(mask_hspan, img.height - mask_hspan):
        for y in range(mask_wspan, img.width - mask_wspan):
            # offset : find offset by center of mask and current x,y
            offset_x = x - mask_center_x
            offset_y = y - mask_center_y
            # apply
            res = 0
            for i in range(len(mask)):
                for j in range(len(mask[0])):
                    res += mask[i][j] * img.getpixel((j+offset_y, i+offset_x))
            res = round(res/constant)
            new_img.putpixel((y, x), res)
    new_img.save(f"{filename}_avg.jpg")
    return new_img

# Averaging Filter Driver Code
img = Image.open("ImageProcessing/assign3/noisy_img2.jpg")
new_img = averaging_filter(img)
new_img.show()


def median_filter(img:IMG) -> IMG:
    mask = [
    [1,1,1],
    [1,1,1],
    [1,1,1]
    ]
    filename = img.filename.strip(".jpg")
    img = img.convert("L") # convert to gray scale
    new_img = img.copy() # copy img to new_img
    mask_height = len(mask)
    mask_width = len(mask[0])
    mask_hspan = int((mask_height - 1)/2)
```

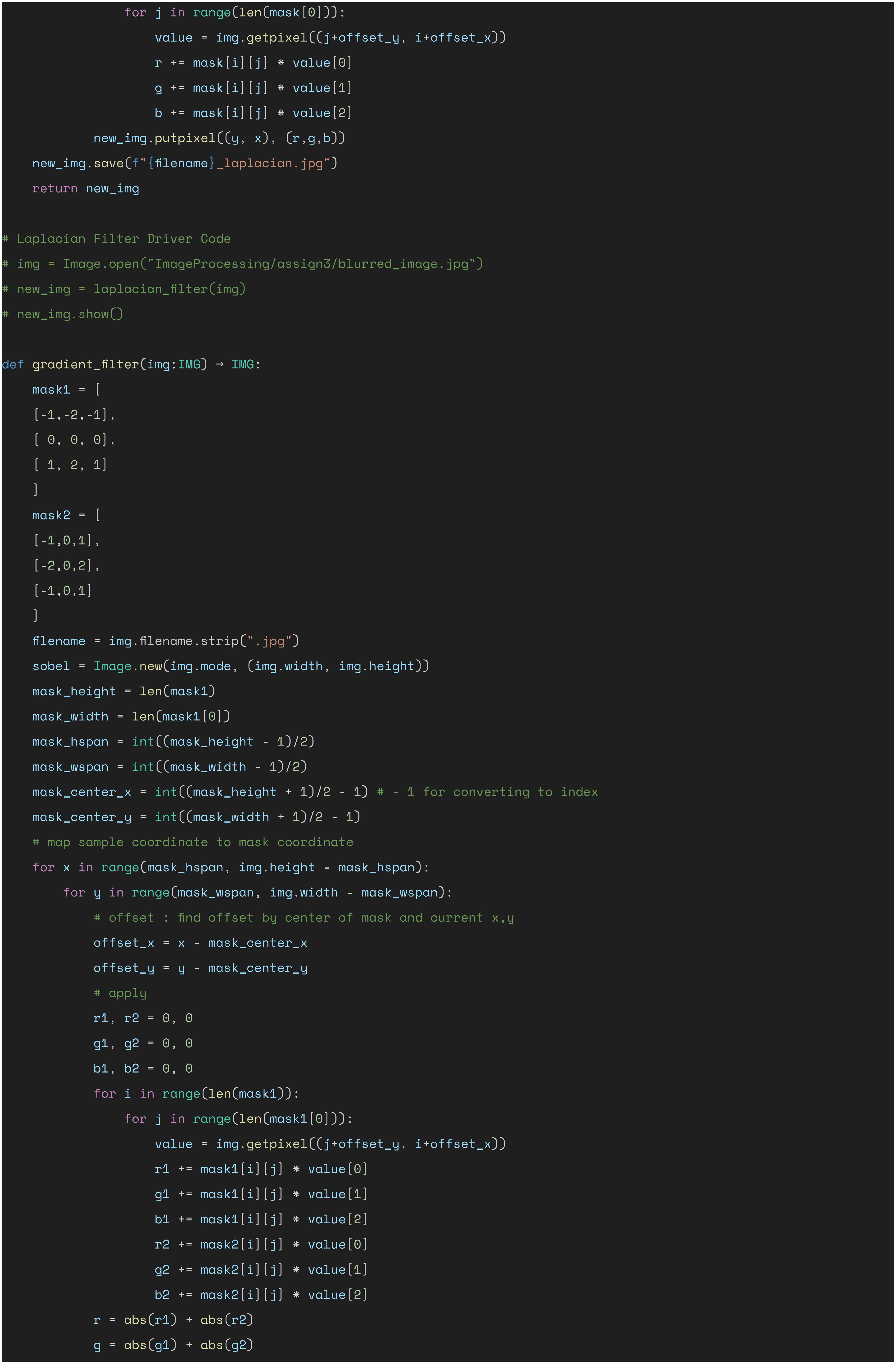


```
    mask_wspan = int((mask_width - 1)/2)
    mask_center_x = int((mask_height + 1)/2 - 1) # - 1 for converting to index
    mask_center_y = int((mask_width + 1)/2 - 1)
    # map sample coordinate to mask coordinate
    for x in range(mask_hspan, img.height - mask_hspan):
        for y in range(mask_wspan, img.width - mask_wspan):
            # offset : find offset by center of mask and current x,y
            offset_x = x - mask_center_x
            offset_y = y - mask_center_y
            # apply
            lst = []
            for i in range(len(mask)):
                for j in range(len(mask[0])):
                    lst.append(img.getpixel((j+offset_y, i+offset_x)))
            lst.sort()
            median_index = round((len(lst) + 1)/2 - 1)
            res = lst[median_index]
            new_img.putpixel((y, x), res)
    new_img.save(f"{filename}_median.jpg")
    return new_img

# Median Filter Driver Code
# img = Image.open("ImageProcessing/assign3/noisy_img2.jpg")
# new_img = median_filter(img)
# new_img.show()
```

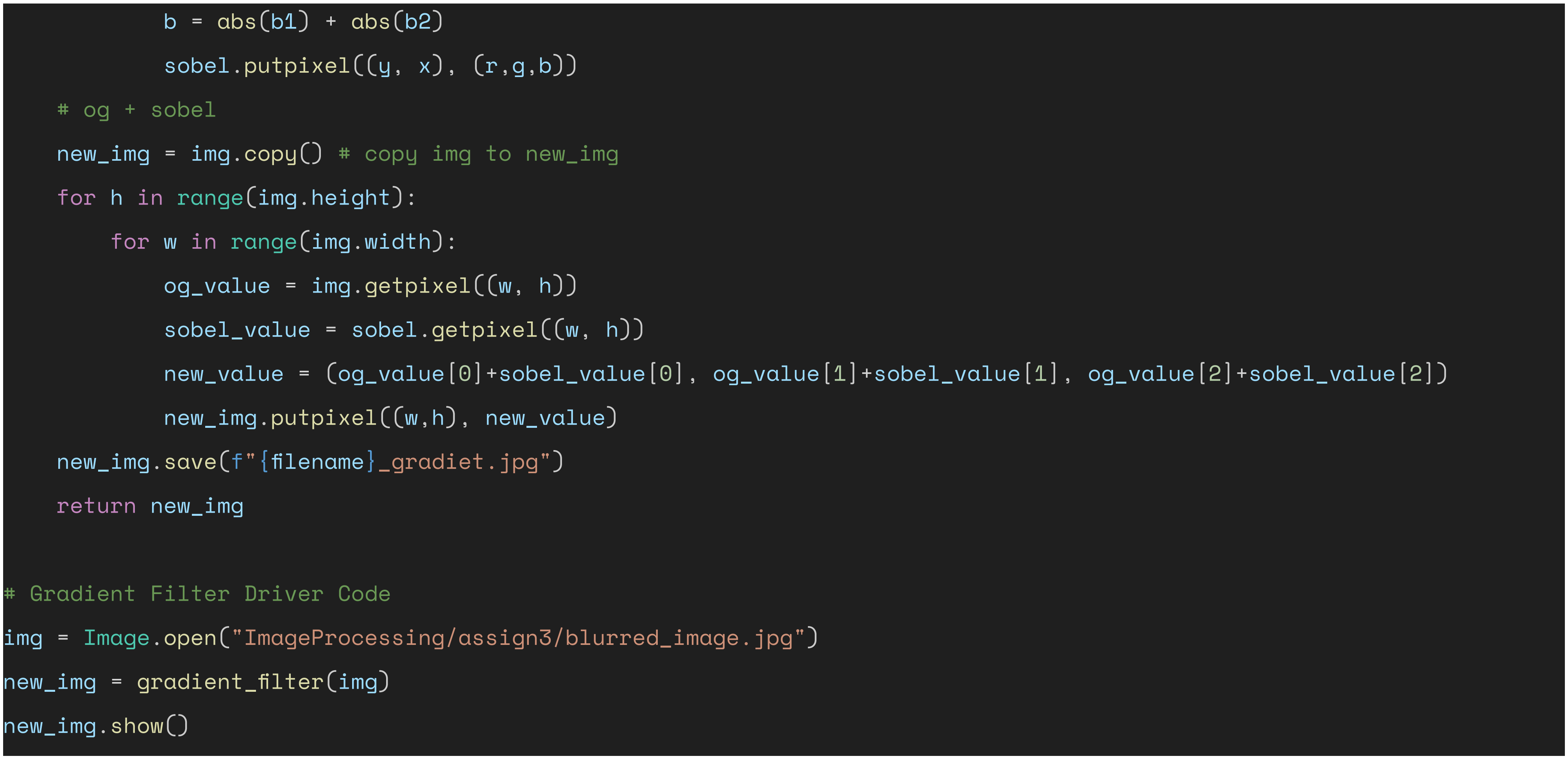
2.Sharpening Filter



```
from PIL import Image
from PIL.Image import Image as IMG

def laplacian_filter(img:IMG) → IMG:
    mask = [
    [ 0,-1, 0],
    [-1, 5,-1],
    [ 0,-1, 0]
    ]
    filename = img.filename.strip(".jpg")
    new_img = img.copy() # copy img to new_img
    mask_height = len(mask)
    mask_width = len(mask[0])
    mask_hspan = int((mask_height - 1)/2)
    mask_wspan = int((mask_width - 1)/2)
    mask_center_x = int((mask_height + 1)/2 - 1) # - 1 for converting to index
    mask_center_y = int((mask_width + 1)/2 - 1)
    # map sample coordinate to mask coordinate
    for x in range(mask_hspan, img.height - mask_hspan):
        for y in range(mask_wspan, img.width - mask_wspan):
            # offset : find offset by center of mask and current x,y
            offset_x = x - mask_center_x
            offset_y = y - mask_center_y
            # apply
            r = 0
            g = 0
            b = 0
            for i in range(len(mask)):
```



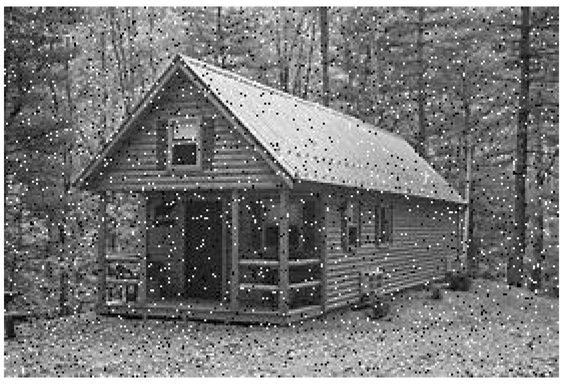
```
                for j in range(len(mask[0])):
                    value = img.getpixel((j+offset_y, i+offset_x))
                    r += mask[i][j] * value[0]
                    g += mask[i][j] * value[1]
                    b += mask[i][j] * value[2]
            new_img.putpixel((y, x), (r,g,b))
    new_img.save(f"{filename}_laplacian.jpg")
    return new_img

# Laplacian Filter Driver Code
# img = Image.open("ImageProcessing/assign3/blurred_image.jpg")
# new_img = laplacian_filter(img)
# new_img.show()

def gradient_filter(img:IMG) -> IMG:
    mask1 = [
    [-1,-2,-1],
    [ 0, 0, 0],
    [ 1, 2, 1]
    ]
    mask2 = [
    [-1,0,1],
    [-2,0,2],
    [-1,0,1]
    ]
    filename = img.filename.strip(".jpg")
    sobel = Image.new(img.mode, (img.width, img.height))
    mask_height = len(mask1)
    mask_width = len(mask1[0])
    mask_hspan = int((mask_height - 1)/2)
    mask_wspan = int((mask_width - 1)/2)
    mask_center_x = int((mask_height + 1)/2 - 1) # - 1 for converting to index
    mask_center_y = int((mask_width + 1)/2 - 1)
    # map sample coordinate to mask coordinate
    for x in range(mask_hspan, img.height - mask_hspan):
        for y in range(mask_wspan, img.width - mask_wspan):
            # offset : find offset by center of mask and current x,y
            offset_x = x - mask_center_x
            offset_y = y - mask_center_y
            # apply
            r1, r2 = 0, 0
            g1, g2 = 0, 0
            b1, b2 = 0, 0
            for i in range(len(mask1)):
                for j in range(len(mask1[0])):
                    value = img.getpixel((j+offset_y, i+offset_x))
                    r1 += mask1[i][j] * value[0]
                    g1 += mask1[i][j] * value[1]
                    b1 += mask1[i][j] * value[2]
                    r2 += mask2[i][j] * value[0]
                    g2 += mask2[i][j] * value[1]
                    b2 += mask2[i][j] * value[2]
            r = abs(r1) + abs(r2)
            g = abs(g1) + abs(g2)
```



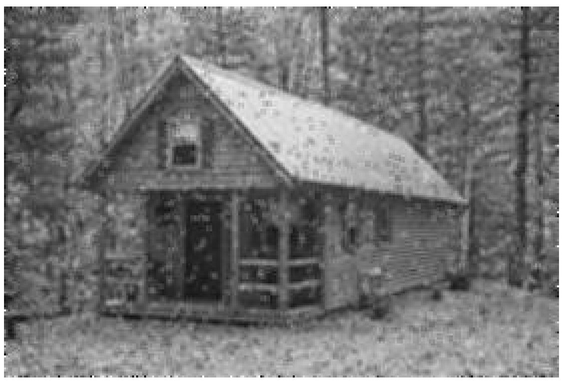
```
            b = abs(b1) + abs(b2)
            sobel.putpixel((y, x), (r,g,b))
    # og + sobel
    new_img = img.copy() # copy img to new_img
    for h in range(img.height):
        for w in range(img.width):
            og_value = img.getpixel((w, h))
            sobel_value = sobel.getpixel((w, h))
            new_value = (og_value[0]+sobel_value[0], og_value[1]+sobel_value[1], og_value[2]+sobel_value[2])
            new_img.putpixel((w,h), new_value)
    new_img.save(f"{filename}_gradiet.jpg")
    return new_img

# Gradient Filter Driver Code
img = Image.open("ImageProcessing/assign3/blurred_image.jpg")
new_img = gradient_filter(img)
new_img.show()
```

Output

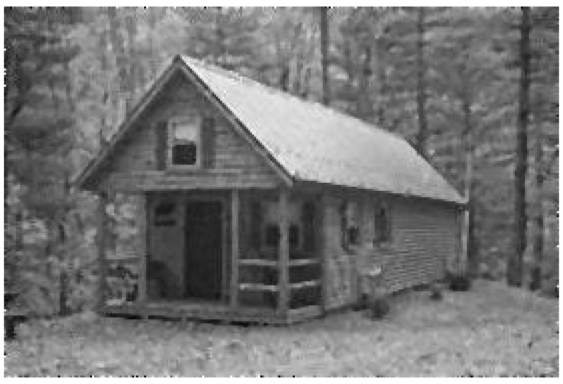
OG



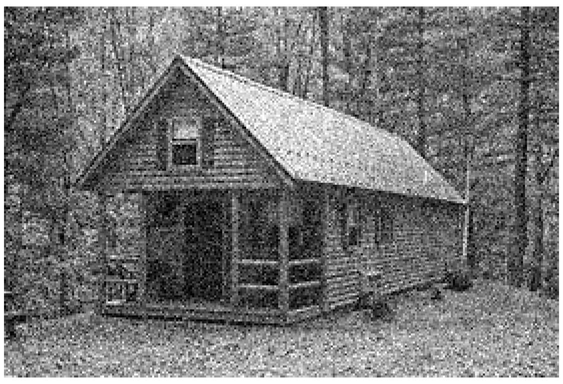
Averaging



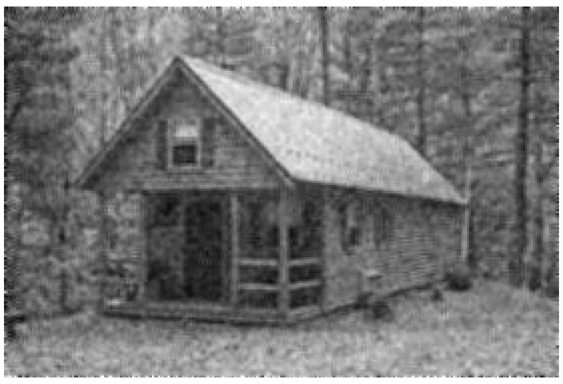
Median



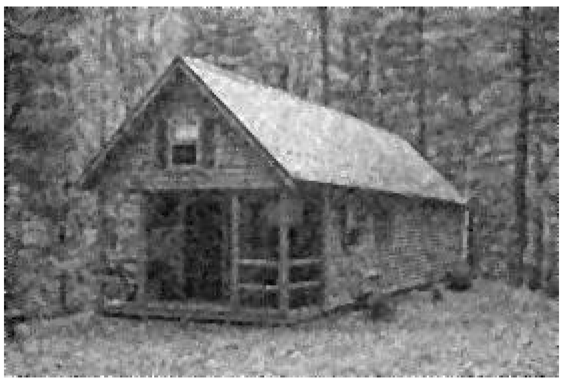
OG



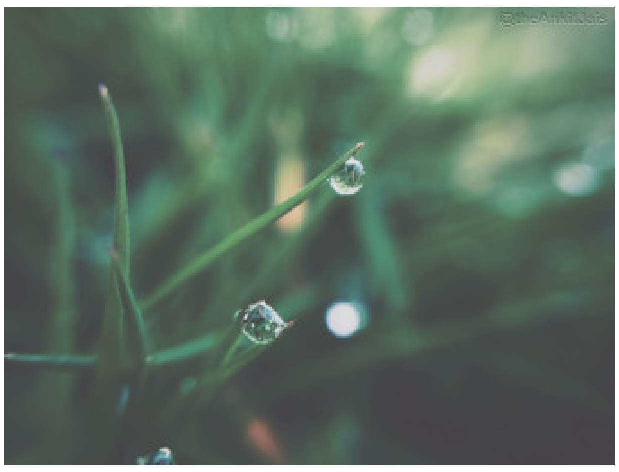
Averaging



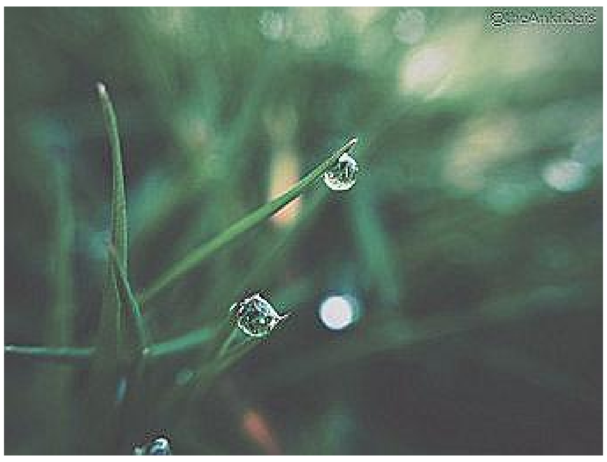
Median



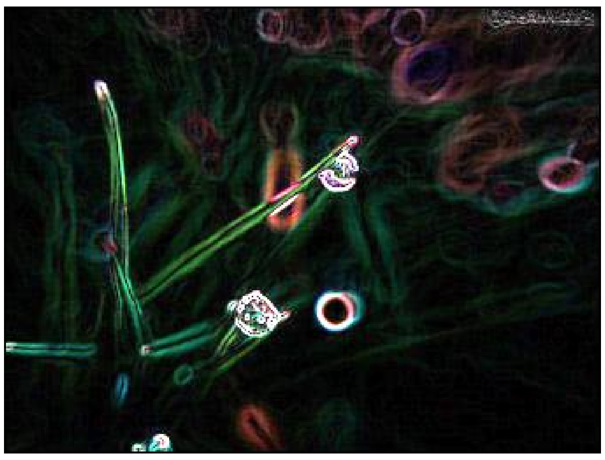
OG



Laplacian



Sobel



OG + Sobel