实验目的

掌握基于拉普拉斯变换的图像锐化算法

实验内容

基本拉普拉斯变换图像增强
拉普拉斯变体变换图像增强
高斯-拉普拉斯变换图像增强

算法设计

基本拉普拉斯变换图像增强

基于以下卷积核模板的图像变换

0	-1	0
-1	5	-1
0	-1	0

拉普拉斯变体变换图像增强

基于以下卷积核模板的图像变换

-1	-1	-1
-1	9	-1
-1	-1	-1

高斯-拉普拉斯变换图像增强

基于以下卷积核模板的图像变换

0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

算法解释说明

基本原理

以模板

0	1	0
1	-4	1
0	1	0

为例，用$g(x,y)$代表变换后的像素值，$f(x,y)$代表变换前的像素值，

则有如下公示：

$g(x,y) = -4 \times f(x,y) + [f(上) + f(下) + f(左) + f(右)]$

如果该区域为平坦区域，则 $g(x,y)$ 结果为0，即对平坦区域实现0响应。

如果该区域为边缘，则在$[f(上)、f(下)、f(左)、f(右)]$中有一个与其它像素值不同，那么$g(x,y)$的值主要取决于该边缘值，所以对边缘有强烈的响应。

以模板

0	-1	0
-1	5	-1
0	-1	0

为例：

$g(x,y) = 5 \times f(x,y) - [f(上) + f(下) + f(左) + f(右)]$

化简后得

$g(x,y) = f(x,y) + [4 \times f(x,y) - [f(上) + f(下) + f(左) + f(右)]]$

如果该区域为平坦区域，则 $g(x,y)$ 结果为$f(x,y) + 0$，即对平坦区域实现0响应,该店像素值不变。

如果该区域为边缘，则在$[f(上)、f(下)、f(左)、f(右)]$中有一个与其它像素值不同，那么$g(x,y) = f(x,y) + [f(x,y) - f(边缘像素值差值)]$，所以对边缘有强烈的响应，并且锐化增强图像。

对于上述的**基本**拉普拉斯变换图像增强和拉普拉斯**变体**变换图像增强，其卷积核中间系数与其它系数相加不等于0是因为需要加上原图，而系数等于0则是该卷积核对边缘区域的响应非常剧烈而对平坦区域（即非边缘区域）响应为0。

代码

```
from PIL import Image
import math

def read_image(file_path):
    image = Image.open(file_path).convert("L")
    return image

def save_image(image,file_path):
    image.save(file_path)

def progress_img(image,kernel):
    width,height = image.size
```

```
    reslut_pixels = []
    kernel_size = len(kernel)
    offset = kernel_size // 2


    for y in range(height):
        for x in range(width):
            # 计算卷积后的像素值
            pixel_sum = 0

            for m in range(len(kernel)):
                for n in range(len(kernel[0])):
                    if(((x + n - 1)>=width or (x + n - 1)<0) or ((y + m -
1)>=height or (y + m - 1)<0 )):
                        pixel_sum+=0
                    else:
                        pixel_sum += image.getpixel((x + n - 1, y + m - 1)) *
kernel[m][n]

            reslut_pixels.append(pixel_sum)

    # 形成新图像
    result_img = Image.new("L",(width,height))
    result_img.putdata(reslut_pixels)
    return result_img


def main():
    input_filepath = './img1/img1.tif'
    output_fielpath = './img1/bGaussian_lpls_17.bmp'

    # 拉普拉斯变换卷积核
    basic_lpls_kernel = [[0,-1,0],
                         [-1,4,-1],
                         [0,-1,0]]

    variant_lpls_kernel = [[0,1,0],
                           [1,-4,1],
                           [0,1,0]]

    Gaussian_lpls_kernel = [[0,0,-1,0,0],
                            [0,-1,-2,-1,0],
                            [-1,-2,17,-2,-1],
                            [0,-1,-2,-1,0],
                            [0,0,-1,0,0]]

    # 读取图片
    origin_img = read_image(input_filepath)

    # 处理并获得基本拉普拉斯变换图片
    # basic_lpls_img = progress_img(origin_img,basic_lpls_kernel)

    # # 变体拉普拉斯变换图像
    # variant_lpls_img = progress_img(origin_img,variant_lpls_kernel)
```

```
    # 高斯拉普拉斯变换图像
    Gaussian_lpls_img = progress_img(origin_img,Gaussian_lpls_kernel)

    # 展示图片
    origin_img.show(title='Origin image')
    Gaussian_lpls_img.show(title='variant_lpls5_image')

    # 保存图片
    save_image(Gaussian_lpls_img,output_fielpath)

if __name__ == "__main__":
    main()
```

实验结果

图1

Image 1: Original Image

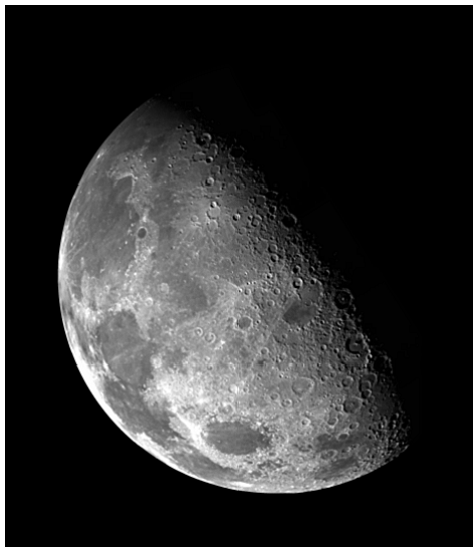

Image 2: Basic Laplacian

Image 3: Variant Laplacian

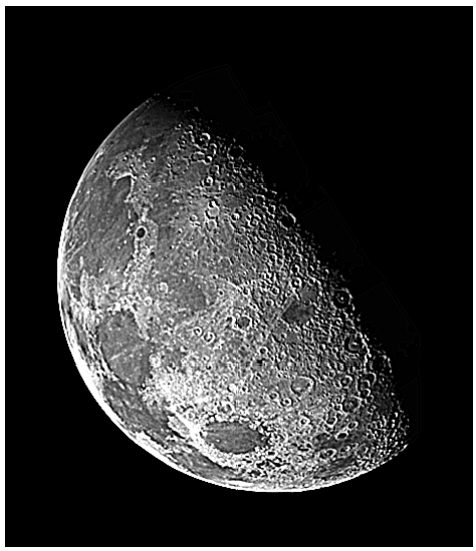

Image 2: Gaussian Laplacian

图2

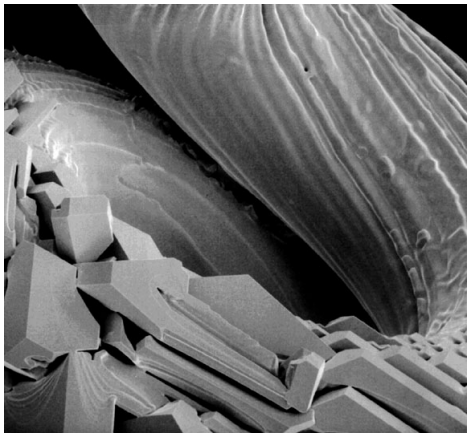

Image 5: Original Image

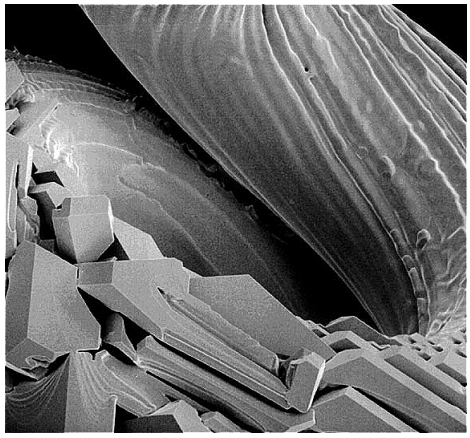

Image 6: Basic Laplacian

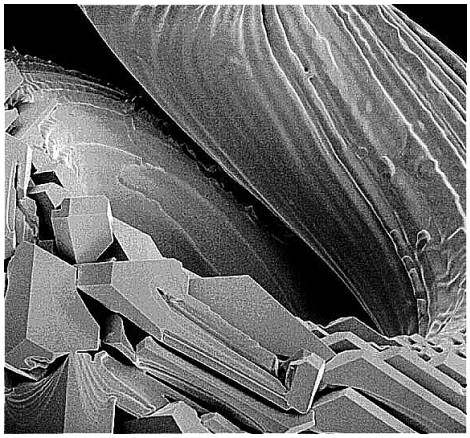

Image 7: Variant Laplacian

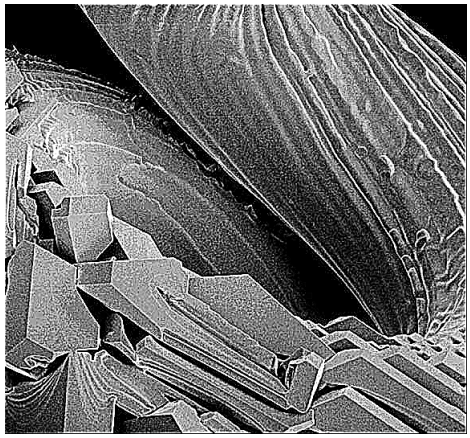

Image 8: Gaussian Laplacian

结论

对于图一：

基本拉普拉斯变换,图像在原图的基础上，额外增加了如*Image 9*的细节,实现了边缘的锐化

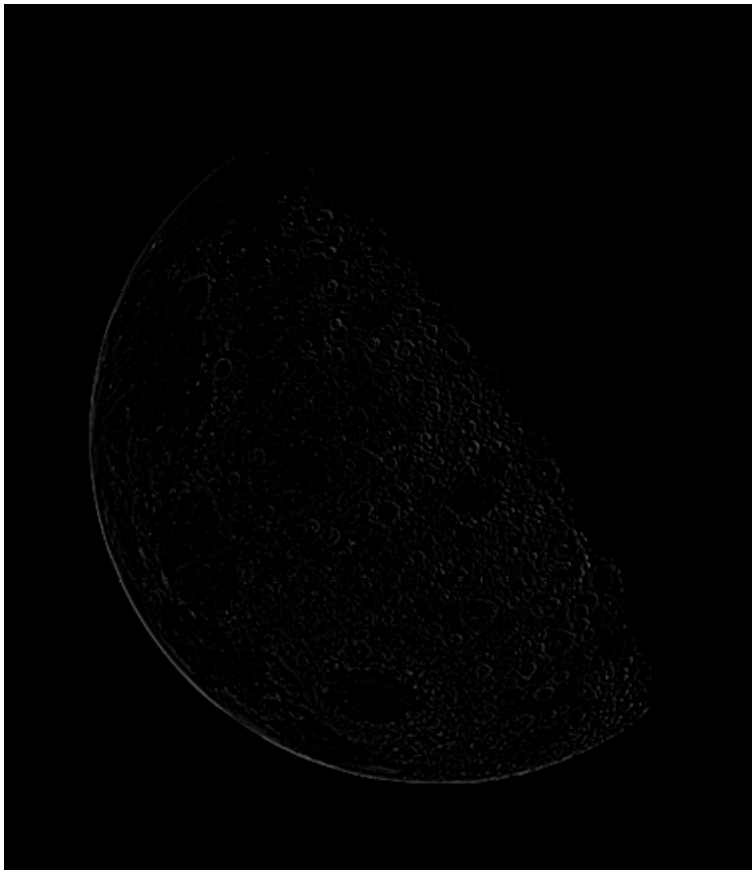

Image 9: 基本拉普拉斯变换增强的细节

变体拉普拉斯变换，与基本拉普拉斯变换相比，更加强调了中心像素值的贡献程度，从而使得图像在变换后更加突出。

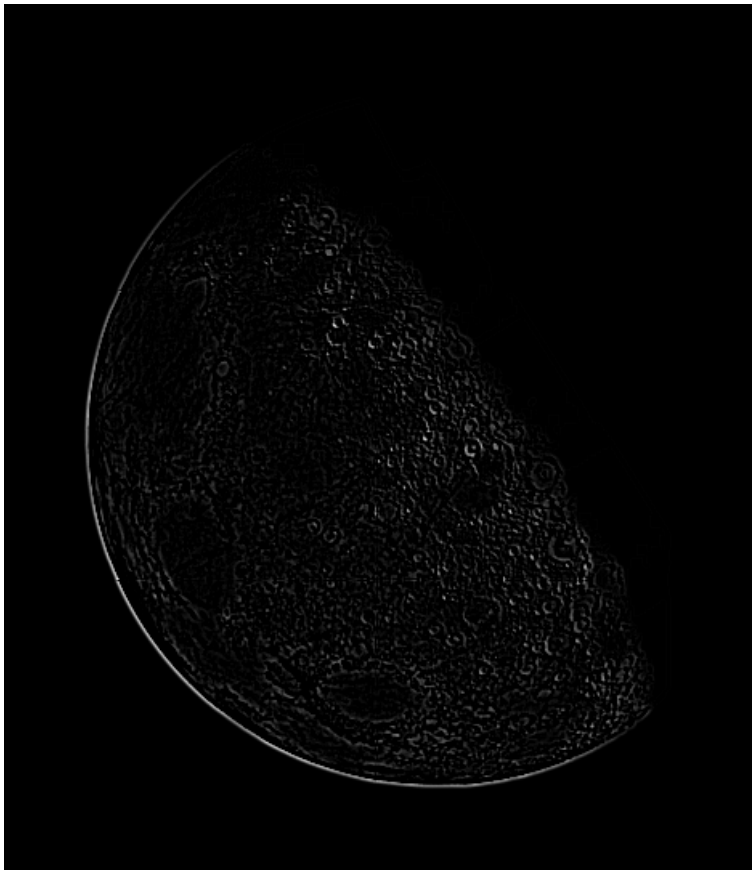

Image 10: 变体拉普拉斯变换增强的细节

上述两种变换都是3×3的卷积核，对细节的增强较少

高斯拉普拉斯变换与上述两种变换相比，采用了5×5的卷积核，对细节增强更多，从*Image 11*即可体现，且因为拉普拉斯变换对噪声非常敏感，采用高斯拉普拉斯变换后会先除去部分噪声，再增强细节，效果会好上很多。

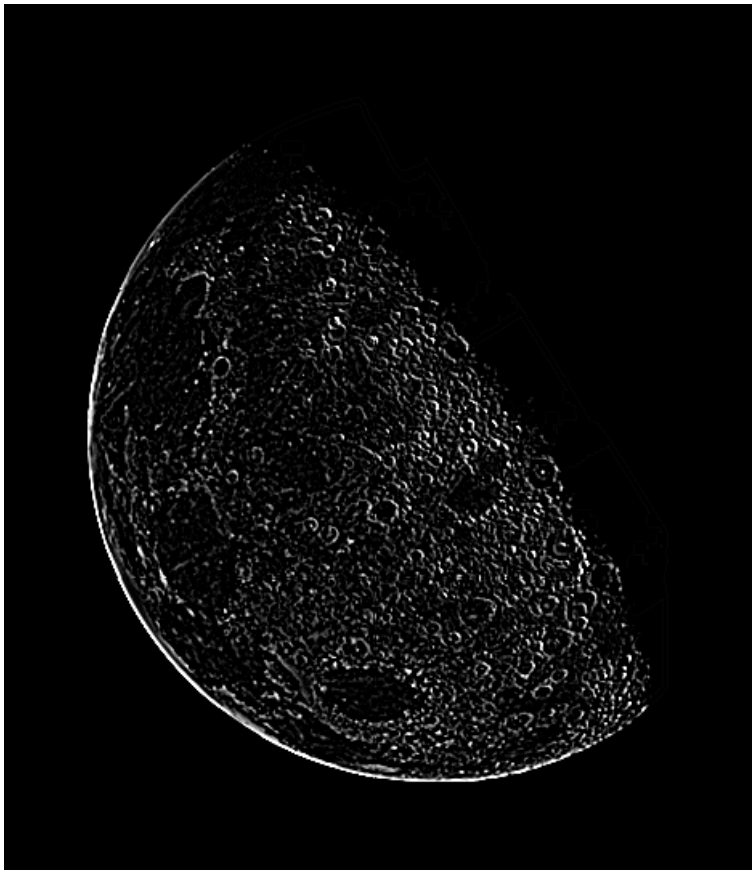

Image 11: 高斯拉普拉斯变换增强的细节

对于图二：

基本拉普拉斯便函和变体拉普拉斯变换对图二的细节增强如图*Image 12* 和 *Image 13* 和 *Image 14*

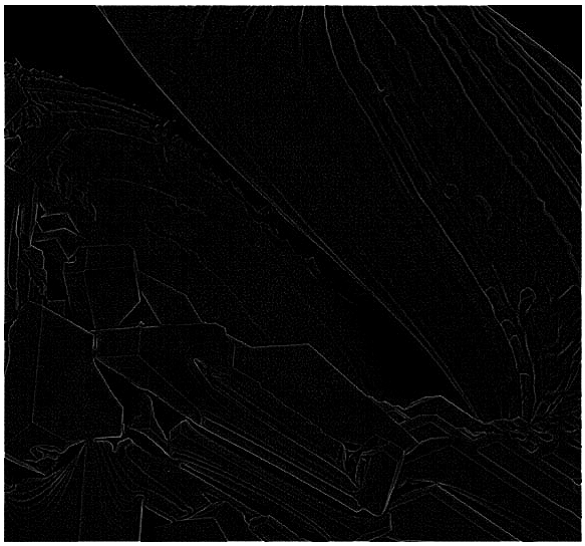

Image 12: 基本拉普拉斯变换增强的细节

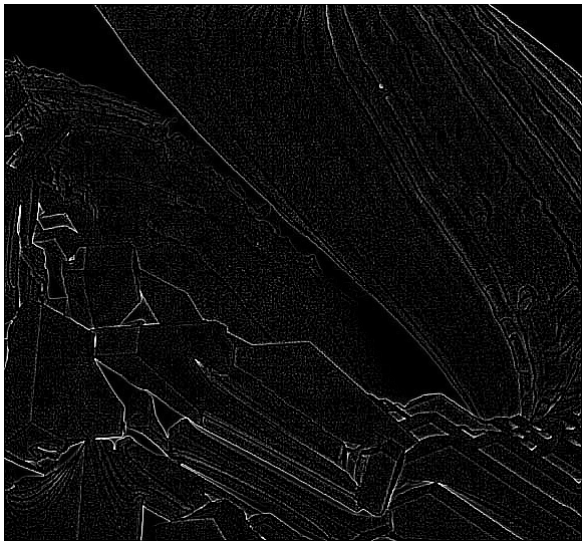

Image 13: 变体拉普拉斯变换增强的细节

Image 14: 高斯拉普拉斯变换增强的细节

可以看出效果与图一基本一致