《数字图像处理》实验报告

姓名： 汪雨卿 学号：19120191

实验六

一. 任务 1

请编写代码，实现一个拉普拉斯算子，对图像 blurry_moon.tif 进行锐化，并和函数库自带的拉普拉斯算子滤波函数进行结果比较。

a) 核心代码：

方法一：实现自己的拉普拉斯算子。*实现对 blurry_moon.tif 进行转换输出 my_sharp.png*

```
def myLaplace():
    # 1.读入图片，处理灰度
    img = readImg()
    img_grey = cv.cvtColor(img, cv.COLOR_RGB2GRAY)

    # 2. 存入数组，并设置宽和高
    img_gaus = cv.GaussianBlur(img_grey, (3, 3), 0, 0)
    img_grey = np.array(img_gaus)
    weight, height = img.shape[:2]

    img_pad = np.pad(img_grey, ((1, 1), (1, 1)), 'edge')

    arr = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
    img_out = np.zeros((weight, height))
    for i in range(weight - 2):
        for j in range(height - 2):
            img_out[i, j] = np.sum(img_pad[i:i + 3, j:j + 3] * arr)
            if img_out[i, j] < 0:
                img_out[i, j] = 0

    img_sharp = img_grey - img_out
    cv.imshow('sharp', img_sharp)
    cv.imwrite('my_sharp.jpg', img_sharp)
    cv.waitKey(0)
```

方法二：调库实现拉普拉斯算子。*实现对 blurry_moon.tif 进行转换输出 lab_sharp.jpg*

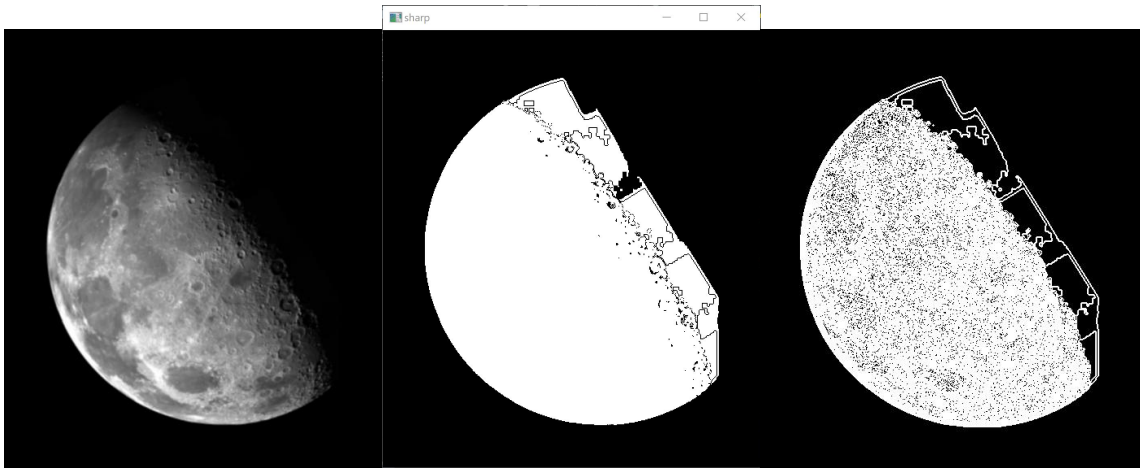
```python
def lab_Laplace():
    img = readImg()
    img_gaus = cv.GaussianBlur(img, (3, 3), 0, 0)

    img_grey = cv.cvtColor(img_gaus, cv.COLOR_RGB2GRAY)

    img_lap = cv.Laplacian(img_grey, cv.CV_16S, 3)
    img_abs = cv.convertScaleAbs(img_lap)
    _, img_out = cv.threshold(img_abs, 0, 255, cv.THRESH_BINARY)
    # cv.namedWindow('pic',cv.WINDOW_AUTOSIZE)
    # cv.imshow('pic',img)
    # cv.namedWindow('out_pic', cv.WINDOW_AUTOSIZE)
    # cv.imshow('out_pic', img_out)
    cv.imwrite('lab_sharp.jpg', img_out)
    cv.waitKey(0)
```

b) 实验结果截图

原图：　　自己实现的锐化：　　调库做的 gamma 变换



二. 任务 2

请编写代码，实现一种梯度算子，并和函数库自带的梯度算子函数进行结果比较。至少在以下图像上测试：magic.png，plate.png，lane.png

a) 核心代码：

方法一：实现自己的 sobel 算子。*实现对 magic.png，plate.png，lane.png 进行转换输出。*

```python
def my_sobel(img, title):
    img = readImg(img)
    img = np.array(img)
    weight, height = img.shape[:2]
    img_new = np.zeros((weight, height))
    img_newx = np.zeros(img.shape[:2])
    img_newy = np.zeros(img.shape[:2])
    sob_x = np.array(([-1, 0, 1], [-2, 0, 2], [-1, 0, 1]))
    sob_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
    for i in range(weight - 2):
        for j in range(height - 2):
            img_newx[i + 1, j + 1] = abs(np.sum(img[i:i + 3, j:j + 3] * sob_x))
            img_newy[i + 1, j + 1] = abs(np.sum(img[i:i + 3, j:j + 3] * sob_y))
            img_new[i + 1, j + 1] = (img_newx[i + 1, j + 1] * img_newx[i + 1, j + 1] + img_newy[i + 1, j + 1] * img_newy[i + 1, j + 1]) ** 0.5
    img_newxy = np.uint8(img_new)
    showImg(title, img_newxy)
    saveImg(title, img_newxy)
```

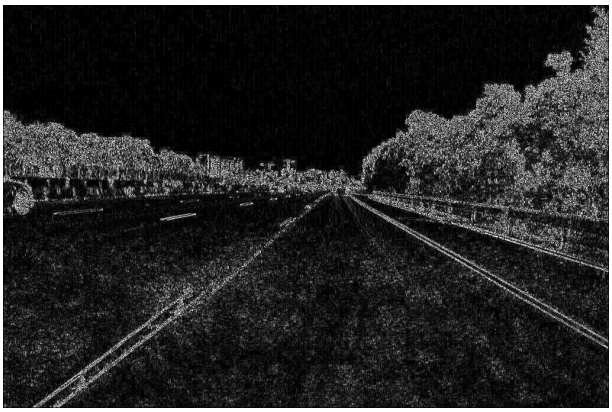
方法二：调库实现 sobel 算子。*实现对 magic.png，plate.png，lane.png 进行转换输出。*

```
18    def lab_sobel(img, title):
19        img = readImg(img, 0)
20        img_sobx = cv.Sobel(img, cv.CV_64F, 1, 0, ksize=3)
21        img_soby = cv.Sobel(img, cv.CV_64F, 0, 1, ksize=3)
22        img_sobxy = np.sqrt(img_sobx ** 2 + img_soby ** 2)
23        showImg(title, img_sobxy)
24        saveImg(title, img_sobxy)
25
```

b)　实验结果截图

lane 原图：

自己实现：



调库实现：

magic 原图：　　自己实现：　　调库实现：

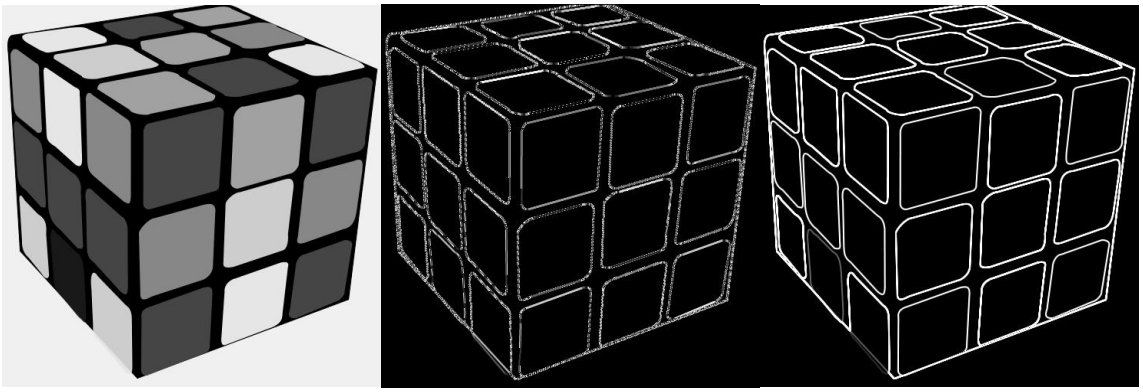


plate 原图：

自己实现：

调库实现：

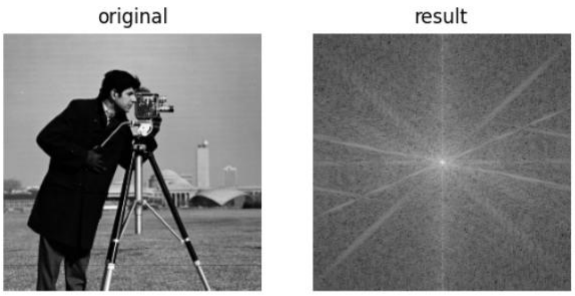
三． 任务 3

请编写代码，对 cameraman.tif 进行傅里叶变换，并对结果可视化

a)　核心代码：

```
21
22  # img = cv2.imread("cameraman.tif")
23  img = readImg("cameraman.tif", 0)
24  print(img)
25  dft = cv2.dft(np.float32(img), flags = cv2.DFT_COMPLEX_OUTPUT)
26  dftShift = np.fft.fftshift(dft)
27  result = 20*np.log(cv2.magnitude(dftShift[:,:,0], dftShift[:,:,1]))
28
29  plt.subplot(121)
30  plt.imshow(img, cmap = 'gray')
31  plt.title('original')
32  plt.axis('off')
33
34  plt.subplot(122)
35  plt.imshow(result, cmap = 'gray')
36  plt.title('result')
37  plt.axis('off')
38
39  plt.show()
```

b)　实验结果截图



四、实验小结

本次实验通过学习不同算子对于原始图像的优化调整原理，利用图像处理中的卷积处理，最终实现了针对不同图片实现了对应的优化调整。本次实验也和课程内容相结合，帮我更好的理解了图像处理的原理。