《数字图像处理》实验报告

姓名： 汪雨卿 学号：19120191

实验七

一. 任务 1

请编写代码，通过频域滤波方法，过滤掉 stripy_cameraman.png 中的条状噪声
提示：通过在频域比较 cameraman.tif 的频谱图像来确定滤掉哪个部分

a) 核心代码：

i. 进行傅里叶变换，实现空间到频率转换域

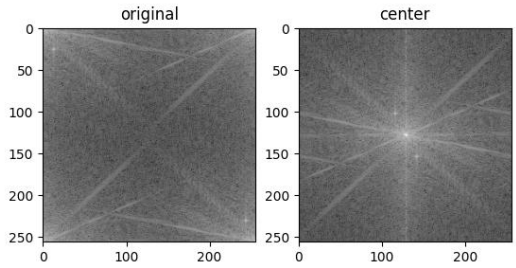
```
def FFT(img):
    img_f = np.fft.fft2(img)
    img_fs = np.fft.fftshift(img_f)  # 图像傅里叶变换并移到图像中央位置
    # img_fs = np.log(np.abs(img_fs))
    return img_fs
```

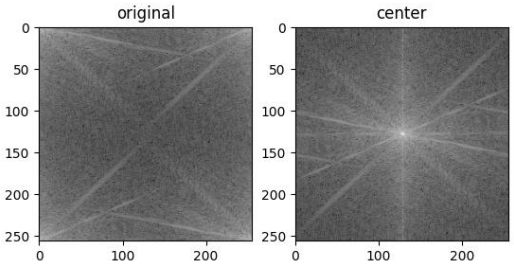
ii. 对比后消除条状噪声

```
if __name__ == "__main__":

    img = readImg('cameraman.tif')
    img_str = readImg('stripy_cameraman.png')

    weight, height = img.shape[:2]
    arr = FFT(img)
    arr_str = FFT(img_str)


    arr_str[80:110, 110:120] = 0
    arr_str[140:160, 135:145] = 0

    img_back = np.fft.ifftshift(arr_str)
    img_back = np.fft.ifft2(img_back)# 出来的是复数，无法显示
    img_back = np.abs(img_back)

    plt.subplot(121), plt.imshow(img_back, 'gray'), plt.title('img back')
    plt.show()

    print('done')
```

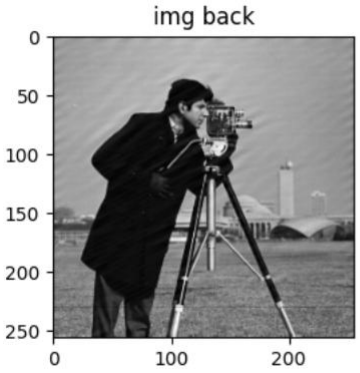
b) 实验结果截图

i. 原图与条状噪声图的傅里叶变换后图片





ii. 最终结果



二. 任务 2

将彩色图像'araras.jpg 的 RGB 3 个通道用灰度图显示出来

a) 核心代码：

```
if __name__ == "__main__":
    img = readImg('araras.jpg')
    # img_r,img_g,img_b = cv.split(img)
    img_r = img[:, :, 0]
    img_g = img[:, :, 1]
    img_b = img[:, :, 2]

    showImg('r', grayImg(img_r))
    showImg('g', grayImg(img_g))
    showImg('b', grayImg(img_b))

    saveImg('r.png', grayImg(img_r))
    saveImg('g.png', grayImg(img_g))
    saveImg('b.png', grayImg(img_b))

    cv.waitKey(0)
```

b) 实验结果截图

三. 任务 3

实现彩色图像的直方图均衡化，并用 stream.jpg 来验证

提示：转换到 HSI 空间，仅对亮度分量用直方图均衡化，再转换回 RGB

a) 核心代码：

i. RGB 图片转换为 HSI 图片

```
def RGB2HSI(img):
    weight, height = img.shape[:2]
    b, g, r = cv.split(img)
    # 归一化[0,1]
    b = b / 255.0
    g = g / 255.0
    r = r / 255.0
    img_HSI = img.copy()
    H, S, I = cv.split(img_HSI)
    for i in range(weight):
        for j in range(height):
            num = 0.5 * ((r[i, j] - g[i, j]) + (r[i, j] - b[i, j]))
            den = np.sqrt((r[i, j] - g[i, j]) ** 2 + (r[i, j] - b[i, j]) * (g[i, j] - b[i, j]))
            angle = float(np.arccos(num / den))

            if den == 0:
                H = 0
            elif b[i, j] <= g[i, j]:
                H = angle
            else:
                H = 2 * 3.1415926 - angle

            min_RGB = min(min(b[i, j], g[i, j]), r[i, j])
            sum = b[i, j] + g[i, j] + r[i, j]
            if sum == 0:
                S = 0
            else:
                S = 1 - 3 * min_RGB / sum

            H = H / (2 * 3.14159265)
            I = sum / 3.0
            # 扩充到255，一般H在[0,2pi],S和I在[0,1]之间
            img_HSI[i, j, 0] = H * 255
            img_HSI[i, j, 1] = S * 255
            img_HSI[i, j, 2] = I * 255
        return img_HSI
```

ii. HSI 图片转换为 RGB 图片

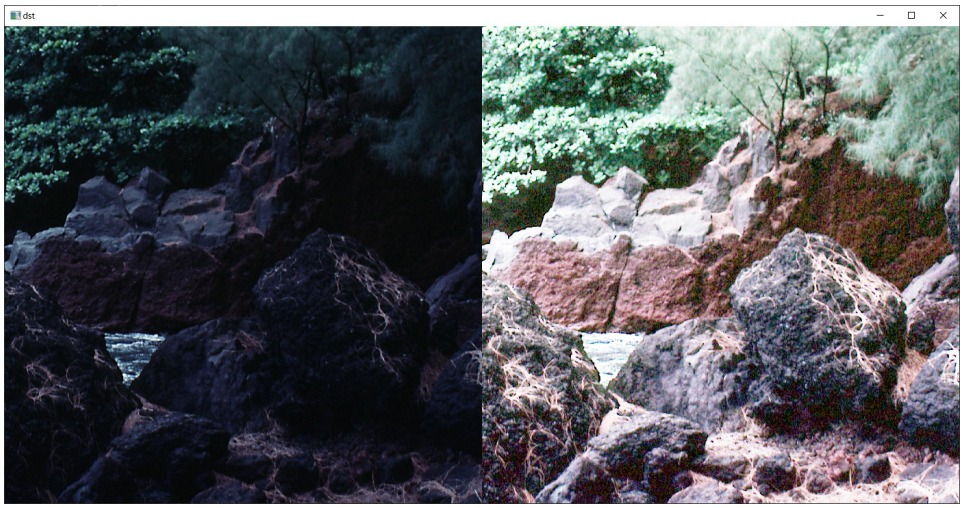
```python
def HSI2RGB(img):
    weight,height = img.shape[:2]
    img_rgb = img.copy()
    H,S,I = cv.split(img)
    [H, S, I] = [i / 255.0 for i in ([H, S, I])]
    R, G, B = H, S, I
    for i in range(weight):
        h = H[i] * 2 * np.pi

        # H大于等于0小于120度时
        a1 = h >= 0
        a2 = h < 2 * np.pi / 3
        a = a1 & a2  # 第一种情况
        tmp = np.cos(np.pi / 3 - h)
        b = I[i] * (1 - S[i])
        r = I[i] * (1 + S[i] * np.cos(h) / tmp)
        g = 3 * I[i] - r - b
        B[i][a] = b[a]
        R[i][a] = r[a]
        G[i][a] = g[a]

        # H大于等于120度小于240度
        a1 = h >= 2 * np.pi / 3
        a2 = h < 4 * np.pi / 3
        a = a1 & a2  # 第二种情况
        tmp = np.cos(np.pi - h)
        r = I[i] * (1 - S[i])
        g = I[i] * (1 + S[i] * np.cos(h - 2 * np.pi / 3) / tmp)
        b = 3 * I[i] - r - g
        R[i][a] = r[a]
        G[i][a] = g[a]
        B[i][a] = b[a]

        # H大于等于240度小于360度
        a1 = h >= 4 * np.pi / 3
        a2 = h < 2 * np.pi
        a = a1 & a2  # 第三种情况
        tmp = np.cos(5 * np.pi / 3 - h)
        g = I[i] * (1 - S[i])
        b = I[i] * (1 + S[i] * np.cos(h - 4 * np.pi / 3) / tmp)
        r = 3 * I[i] - g - b
        B[i][a] = b[a]
        G[i][a] = g[a]
        R[i][a] = r[a]
    img_rgb[:, :, 0] = B * 255
    img_rgb[:, :, 1] = G * 255
    img_rgb[:, :, 2] = R * 255
    return img_rgb
```

iii. **对亮度通道进行**直方图均衡化，并合并通道

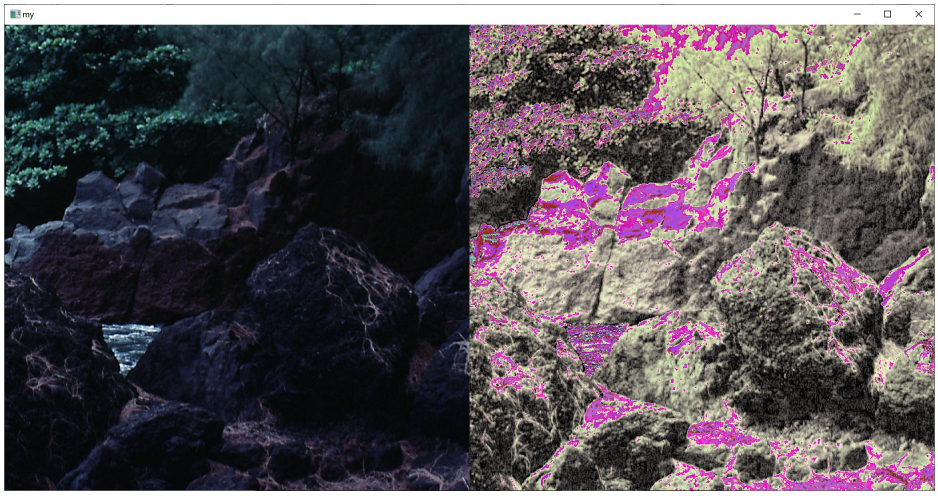
```python
#
img = readImg('stream.jpg')
img_HSI = RGB2HSI(img)
(H,S,I) = cv.split(img)
HH = cv.equalizeHist(H)
SH = cv.equalizeHist(S)
IH = cv.equalizeHist(I)
img_nHSI = cv.merge((H,S,IH), )
```

b)　实验结果截图

RGB 实现结果



HIS 实现结果



四、实验小结

本次实验更加深入的学习了如何对一些有特定噪声的图片进行优化和调整。而不是仅仅在空间滤波层面通过不同的算子进行图像优化。本次实验所涉及的大部分内容也是基于空间滤波的基础之上，通过傅里叶变换将图片转换成频域空间，进而通过在频域上的图像处理，消除一些有规律性有周期性的图像噪音。此外，本实验的后两个任务放眼图像处理的颜色通道，通过通道分离的处理方式，使得处理结构更为优化，有时也会没有期待的效果。