HW4

夏浩 19307130268

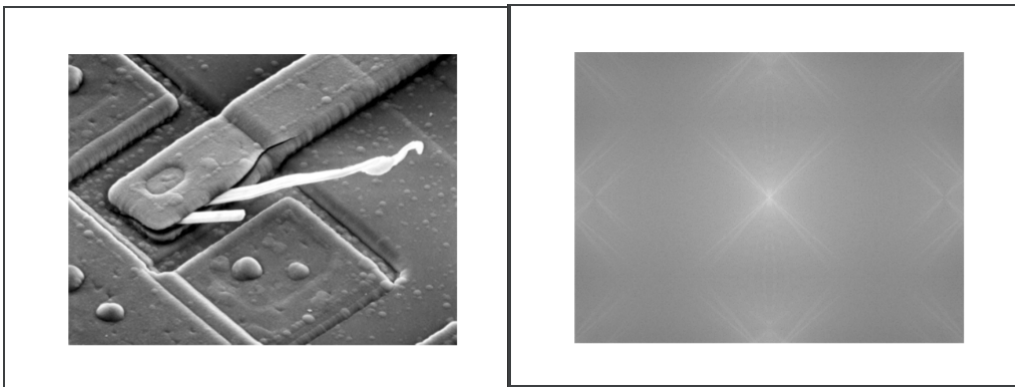
P1.编程实现基于课件中频率域滤波**5**步骤的：

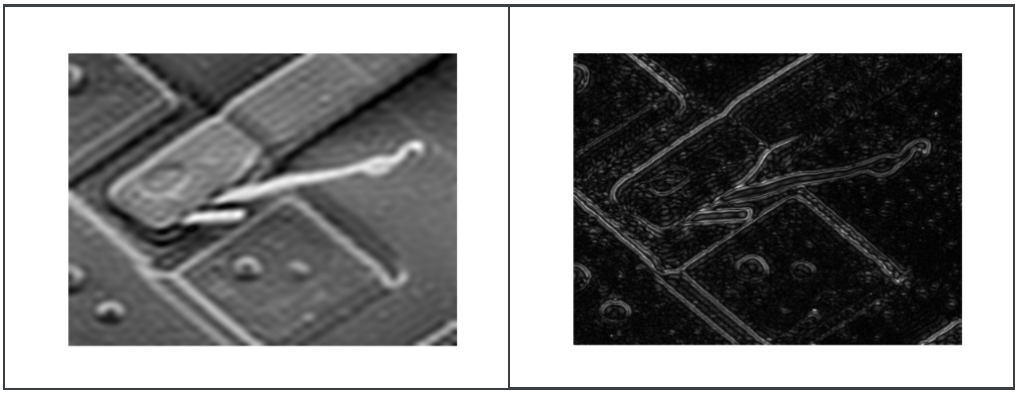
（1）低通平滑操作，并把算法应用与图片上，显示原图的频谱图、频域操作结果的频谱图，以及操作结果；
（2）实现至少一种图像的锐化操作，该操作是基于频域操作的。
备注：图像的时空-频域变换（即离散频域/傅里叶变换和逆变换）可以调用库函数。

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt


def PassFilter(image, d,p='low'):
    m, n = image.shape
    image = cv.copyMakeBorder(image, 0, m, 0, n,
cv.BORDER_REFLECT_101)

    f = np.fft.fft2(image)
    fshift = np.fft.fftshift(f)
    magnitude_spectrum = 20 * np.log(np.abs(fshift))

    def cal_distance(pa, pb):
        from math import sqrt
        dis = sqrt((pa[0] - pb[0]) ** 2 + (pa[1] - pb[1]) ** 2)
        return dis

    def make_transform_matrix(d,p):
        transfor_matrix = np.zeros(image.shape)
        center_point = tuple(map(lambda x: (x - 1) / 2,
image.shape))
        for i in range(transfor_matrix.shape[0]):
            for j in range(transfor_matrix.shape[1]):

                dis = cal_distance(center_point, (i, j))
                if p == 'low':
```

```
                    k = 1
                else:
                    k = 0
                if dis <= d:
                    transfor_matrix[i, j] = k
                else:
                    transfor_matrix[i, j] = 1-k
        return transfor_matrix

    d_matrix = make_transform_matrix(d,p)
    new_img = np.abs(np.fft.ifft2(np.fft.ifftshift(fshift *
d_matrix)))
    return magnitude_spectrum,new_img[:m, :n]


# read image
img = cv.imread('test2.tif',0)

# show the origin image
print(type(img),img.dtype,img.shape)
plt.imshow(img,cmap='gray')
plt.axis('off')
plt.show()

# process
magnitude_spectrum,low_new_img = PassFilter(img,60,p='low')
_,high_new_img = PassFilter(img,60,p='high')
# show the result
plt.imshow(magnitude_spectrum,cmap="gray")
plt.axis('off')
plt.show()
plt.imshow(low_new_img,cmap="gray")
plt.axis('off')
plt.show()
plt.imshow(high_new_img,cmap="gray")
plt.axis('off')
plt.show()
```

原图，频谱，低通滤波，高通滤波





采用镜像填充。

注意到对于低通滤波，有一定的振铃现象。

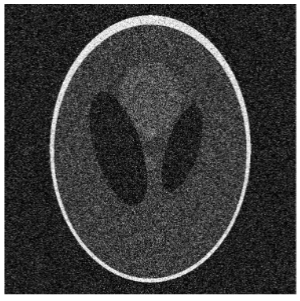
P2.实现噪声的生成（不可以调用别的库实现的函数）

针对对大脑、心脏图像（或其他多类图像），生成以下两种不同类型、不同强度的噪声，并使用生成的噪声污染图像，对比展示噪声污染前后的图像：
（1）生成白噪声；（2）生成其他一种噪声（如高斯、瑞利、椒盐噪声）。

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
import random

def white_noise(img):
    f = np.fft.fft2(img)
    m, n = img.shape
    x,y = int(m/2) -20 , int(n/2) -20
    fshift = np.fft.fftshift(f)
    print(fshift)
    fshift[x, y] = 11115984.30350322+5.04331046e+03j
    fshift[m - x, n - y] = 11115984.30350322+5.04331046e+03j
    new_img = np.abs(np.fft.ifft2(np.fft.ifftshift(fshift)))
```

```
    return new_img


def gauss_noise(img, mean=0, var=0.1):
    noise = np.random.normal(mean, var ** 0.5, img.shape)
    noisy = img / 256 + noise
    noisy = np.clip(noisy, 0.0, 1.0)
    return noisy

# read image
img = cv.imread('test2.tif', 0)

# show the origin image
print(type(img),img.dtype, img.shape)
plt.imshow(img,cmap='gray')
plt.axis('off')
plt.show()

# process
noisy1 = gauss_noise(img, 0, 0.1)
noisy2 = white_noise(img)
# show the result
plt.imshow(noisy1,cmap="gray")
plt.axis('off')
plt.show()
plt.imshow(noisy2,cmap="gray")
plt.axis('off')
plt.show()
```

原图，高斯噪声，白噪声



在生成高斯噪声时，要注意可能产生的随机数超出范围，需要进行截断操作。

P3.编程实现基于频域的选择滤波器方法，去除大脑CT体膜图像（Shepp-Logan）中的条纹；或自己设计一个有周期噪声的图片，并用频域选择滤波器去除噪声。
备注：图像的时空-频域变换（即离散频域/傅里叶变换和逆变换）可以调用库函数。

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

def notch_filter(img,d=15):
    m, n = img.shape
    img = cv.copyMakeBorder(img, 0, m, 0, n, cv.BORDER_REFLECT_101)
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)

    abs_max = np.max(np.abs(fshift))
    thre_h = abs_max / 1.3
    thre_l = abs_max/1200
    magnitude_spectrum1 = 20 * np.log(np.abs(fshift)+1)
    def make_transform_matrix(d):
        transfor_matrix = np.ones(img.shape)
        m,n = transfor_matrix.shape
        for i in range(d,m -d):
            for j in range(d,n -d):
```

```
                if ((i < 21/50 * m or i > 29 /50 * m) and (j <
21/50 * n or j > 29 /50 * n )) \
                        and thre_l < np.abs(fshift[i][j]) < thre_h:
                    transfor_matrix[i-d:i + d,j-d:j+d] = 0
            print(i)
        return transfor_matrix

    d_matrix = make_transform_matrix(d)
    magnitude_spectrum2 = 20 * np.log(np.abs(fshift * d_matrix)+1)
    new_img = np.abs(np.fft.ifft2(np.fft.ifftshift(fshift *
d_matrix)))
    return new_img[:m, :n],magnitude_spectrum1,magnitude_spectrum2


# read image
img = cv.imread('test3.PNG', 0)

# show the origin image
print(type(img),img.dtype, img.shape)
plt.imshow(img,cmap='gray')
plt.axis('off')
plt.show()

# process
new_img,p1,p2 = notch_filter(img)

# show the result
plt.imshow(new_img,cmap="gray")
plt.axis('off')
plt.show()
plt.imshow(p1,cmap="gray")
plt.axis('off')
plt.show()
plt.imshow(p2,cmap="gray")
plt.axis('off')
plt.show()
```

原图，处理结果，原图频谱，处理后频谱