HW3

April 17, 2022

HW3-1

[2]:
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
def filter2d(image, kernel):
    """

    :param image:    size
    :param kernel:
    :return:
    """
    m, n = kernel.shape
    if m!=n:
        print("Wrong kernel shape!")
        return 0
    x, y = image.shape
    # padding
    pad_x = x + m - 1
    pad_y = y + m - 1
    pad_image = np.zeros((pad_x,pad_y))
    pad_image[m//2:pad_x-m//2,m//2:pad_y-m//2] = image
    # store result
    new_image = np.zeros((x,y))
    for i in range(x):
        for j in range(y):
            # new_image[i][j] = np.sum(pad_image[i:i+m, j:j+m]*kernel)   #
            new_image[i,j] = np.sum(pad_image[i:i+m, j:j+m] * kernel[m-1::
↪-1,m-1::-1])  #
    return new_image
```

[4]:
```
def Gaussian2d(k_size,sigma):
    """

    :param k_size:    ,
    :param sigma:
    :return:
```

```
    """
    f = lambda x, y: np.exp(-(x**2+y**2)/(2*sigma**2))
    kernel = np.zeros((k_size,k_size))
    trans_size = int((k_size-1)/2)
    for i in range(k_size):
        for j in range(k_size):
            kernel[i,j] = f(i-trans_size,j-trans_size)
    return kernel/np.sum(kernel)
```

[5]:
```
def Smoothing(imary, k_size, Average=False, Gaussian=False, sigma=1):
    """

    :param image:
    :param k_size:
    :param Average:
    :param Gaussian:
    :param sigma:
    :return:
    """
    if Average:
        kernel = np.ones((k_size,k_size))/k_size**2
        newary = filter2d(imary,kernel)
        return newary
    elif Gaussian:
        kernel = Gaussian2d(k_size,sigma)
        newary = filter2d(imary,kernel)
        return newary
```

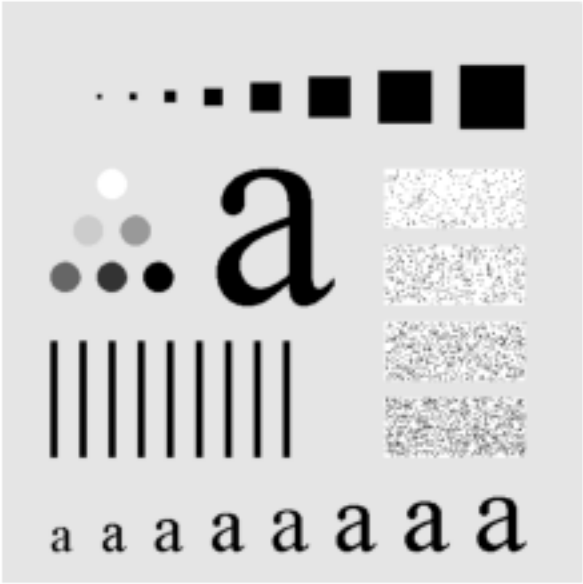
[6]:
```
def Sharpening(imary, Laplace=False, Unsharpmask=False, Highboost=False, k=1):
    """

    :param imary:
    :param Laplace:   Laplace
    :param Unsharpmask:    Unsharp masking
    :param Highboost:    Highboost filtering
    :param k: Highboost filtering
    :return:
    """
    #  laplace
    if Laplace:
        kernel = np.zeros((3,3))
        for i in range(3):
            for j in range(3):
                if abs(i-1) + abs(j-1) == 1:
                    kernel[i,j] = 1
                elif abs(i-1) + abs(j-1) == 0:
                    kernel[i,j] = -4
```

```
        g_ary = filter2d(imary,kernel)
        newary = imary - g_ary
    elif Unsharpmask == True:
        smoothary = Smoothing(imary, 3, Average=True)
        g_mask = imary - smoothary
        newary = imary + g_mask
    elif Highboost == True:
        smoothary = Smoothing(imary, 3, Average=True)
        g_mask = imary - smoothary
        newary = imary + k * g_mask
    return newary
```

（1）平滑操作：

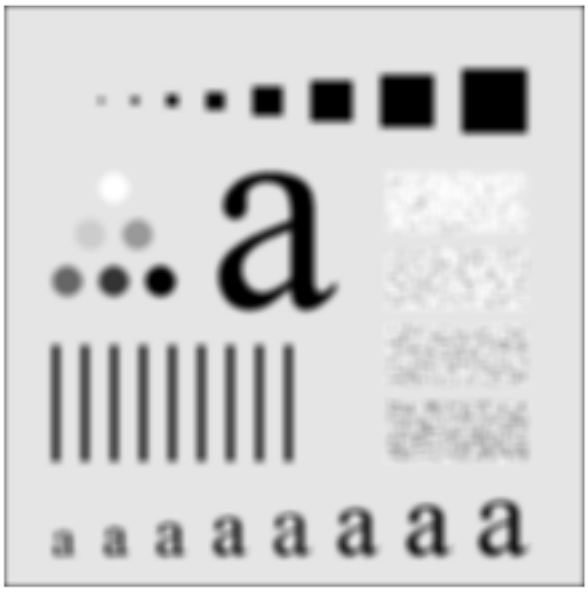
原图如下所示：

[7]:
```
im = Image.open('test_pattern.tif').convert('L')
im_ary = np.array(im, dtype='int32')
plt.imshow(im_ary, cmap='gray')
plt.axis("off")
plt.show()
```



我们首先使用卷积核大小为9的均值滤波器进行平滑处理：

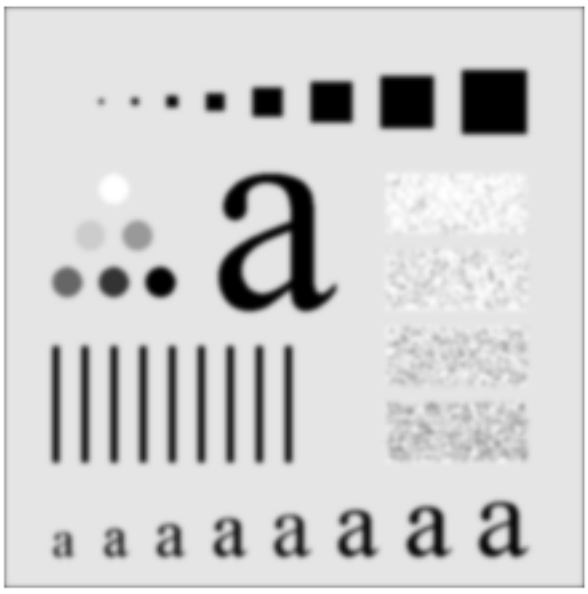
[8]:
```
new_imary1 = Smoothing(im_ary, 9, Average=True)
plt.imshow(new_imary1, cmap='gray')
plt.axis("off")
plt.show()
```



可以发现图片变得模糊。

我们再使用高斯滤波器进行平滑处理，此处我们取高斯滤波器的标准差为2.

[9]:
```
new_imary2= Smoothing(im_ary, 13, Gaussian=True, sigma=2)
plt.imshow(new_imary2, cmap='gray')
plt.axis("off")
plt.show()
```



同样实现了平滑操作，图片变得模糊。

（2）锐化：

原图如下所示：

[11]:
```
im = Image.open('blurry_moon.tif').convert('L')
im_ary = np.array(im, dtype='int32')
plt.imshow(im_ary, cmap='gray')
plt.axis("off")
plt.show()
```

我们首先直接使用拉普拉斯滤波进行锐化操作：

[13]:
```
new_imary1 = Sharpening(im_ary, Laplace=True)
plt.imshow(new_imary1, cmap='gray')
plt.axis("off")
plt.show()
```

可以发现边界变得清晰。

我们再使用unsharp masking方法进行锐化操作：

[14]:
```
new_imary2= Sharpening(im_ary,  Unsharpmask=True)
plt.imshow(new_imary2, cmap='gray')
plt.axis("off")
plt.show()
```

同样实现锐化效果，图片边界变得清晰。

最后我们使用highboost方法进行锐化处理：

[15]:
```
new_imary2= Sharpening(im_ary,  Highboost=True, k=2)
plt.imshow(new_imary2, cmap='gray')
plt.axis("off")
plt.show()
```

可以看出，我们的锐化结果图相比于原图，边界更加清晰。

HW3-2

(1):

(1) 冲击串的 Fourier 变换也是一个冲击串

$F(\mu)=$ ① 证冲激信号 $\delta(t-t_0)$ 的 Fourier 变换：

$$F(\mu)=\int_{-\infty}^{\infty}\delta(t-t_0)e^{-j2\pi\mu t}dt=e^{-j2\pi\mu t_0}$$

② 若 $f(t)\Longleftrightarrow F(\mu)$

则 $\int_{-\infty}^{\infty}F(t)e^{-j2\pi\mu t}dt=\int_{-\infty}^{\infty}F(t)\cdot e^{j2\pi t(-\mu)}dt=f(-\mu)$

~~$\therefore e^{-j2\pi\mu t_0}$ Fourier变换 δ~~

$F(t)$ 的傅利叶变换为 $f(-\mu)$

③ $F(\mu)=e^{-j2\pi\mu t_0}$ $f(t)=\delta(t-t_0)$

$\therefore F(t)\xrightarrow{\text{Fourier变换}}\delta(-\mu-t_0)$

令 $-t_0=a$ 则 $\delta(t+a)\xrightarrow{\text{Fourier}}\delta e^{j2\pi\mu a}$

$e^{j2\pi\mu a}\xrightarrow{\text{Fourier}}\delta(-\mu+a)=\delta(\mu-a)$

↑ 定义上等价

$\therefore e^{j2\pi ta}\xrightarrow{\text{Fourier}}\delta(\mu-a)$

④ 冲击串：~~$S_{\Delta T}(t)=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}e^{j2\pi\frac{n}{\Delta T}t}$~~

$$S_{\Delta T}(t)=\sum_{n=-\infty}^{\infty}\delta(t-n\Delta T)$$

$$=\sum_{n=-\infty}^{\infty}C_n e^{j2\pi\frac{n}{\Delta T}t}$$ Fourier 展开

$$C_n=\frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2}S_{\Delta T}(t)e^{-j2\pi\frac{n}{\Delta T}t}dt$$

$$=\frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2}\delta(t)\cdot e^{-j2\pi\frac{n}{\Delta T}t}dt$$

$$=\frac{1}{\Delta T}e^{0}=\frac{1}{\Delta T}$$

$\therefore S_{\Delta T}(t)=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}e^{j2\pi\frac{n}{\Delta T}t}$ $\quad\mathcal{F}\{e^{j2\pi\frac{n}{\Delta T}t}\}=\delta(\mu-\frac{n}{\Delta T})$

$\therefore S_{\Delta T}(t)\xrightarrow{\text{Fourier}}S(\mu)$

$S(\mu)=\mathcal{F}\{S_{\Delta T}(t)\}=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta(\mu-\frac{n}{\Delta T})$ 仍为冲击串

(2):

(2) 实信号 $f(x)$ 的离散频域变换结果共轭对称

$f(x) \xrightarrow{\text{Fourier}} F(u)$

pf: $F(u)=\sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M}$ $\quad u=0,1,2,\cdots M-1$

$f(x)=\sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M}$ $\quad x=0,1,2,\cdots M-1$

$F^*(u)=\left[\sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M}\right]^*$

$=\sum_{x=0}^{M-1} f^*(x) e^{-j2\pi(-u)x/M}$

$=\sum_{x=0}^{M-1} f(x) e^{-j2\pi(-u)x/M}$ $\quad$ 实信号 $f^*(x)=f(x)$

$=F(-u)$

$\therefore F^*(u)=F(-u)$

(3)

(3) 二维离散傅利叶变换的卷积定理

pf: 2维Fourier变换对

$$\begin{cases} F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\,e^{-j2\pi(ux/M+vy/N)} \\ f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} e^{j2\pi(ux/M+vy/N)}\,F(u,v) \end{cases}$$

$$\begin{aligned} \Im[f(x,y)*h(x,y)] &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)*h(x,y)\,e^{-j2\pi(ux/M+vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)\right] e^{-j2\pi(ux/M+vy/N)} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x-m,y-n)\,e^{-j2\pi(ux/M+vy/N)} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\,e^{-j2\pi(um/M+vn/N)}\cdot H(u,v) \\ &= F(u,v)\cdot H(u,v) \end{aligned}$$

其中, $f(x,y)\xrightarrow{\text{Fourier}}F(u,v)$

$h(x,y)\xrightarrow{\text{Fourier}}H(u,v)$

$\therefore f(x,y)*h(x,y)\xrightarrow{\text{Fourier}}F(u,v)H(u,v)$

由离散傅利叶变换对的一一对应特性

$f(x,y)*h(x,y)\Leftrightarrow F(u,v)H(u,v)$

同理可证 $f(x,y)h(x,y)\Leftrightarrow F(u,v)*H(u,v)$