HW3

赵心怡 19307110452

1.编程实现图像域基于空间滤波器的（1）平滑操作、（2）锐化算法；并把算法应用与图片上，显示与原图的对比差别。

平滑操作的主要方法有均值，高斯核以及中位数。

对于直接求均值的滤波器，采用

$$kernel = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} * \frac{1}{9}$$

而高斯核需要计算每个点到中间的距离再进行归一化。我们代码中设sigma=2.5

$$\mathrm{weight(s,t) = K \exp^{-\frac{(s-\mu)^2+(t-\mu)^2}{2\sigma^2}}}$$

对结果进行卷积操作，即

$$(w * f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

代入公式进行计算。

对于中位数滤波器只需要找每个patch内灰度值的中位数。

具体代码如下：

```
def smoothing(img, len_patch, method):
    [height, width]=img.shape
    newImg = np.zeros([height,width],np.float64)
    half_length = (len_patch - 1) // 2 # half of patch length
    kernel = np.ones([len_patch,len_patch],np.float64)
    tempImg = np.zeros((height + len_patch - 1, width + len_patch - 1)) # temp matrix for convenient calculation
    tempImg[half_length: height + half_length, half_length: width + half_length] = img

    if method == 'mean':
        kernel = kernel*1.0/(len_patch*len_patch) # kernel for mean algorithm
    if method == 'gauss':
        x, y = np.meshgrid(np.arange(len_patch), np.arange(len_patch))
        sigma = 1
        mu = half_length
        kernel = np.exp(-((x - mu)**2 + (y - mu)**2) / (2*sigma**2))
        kernel /= kernel.sum()  # gaussian kernel

    for i in range(half_length, height+half_length):
```

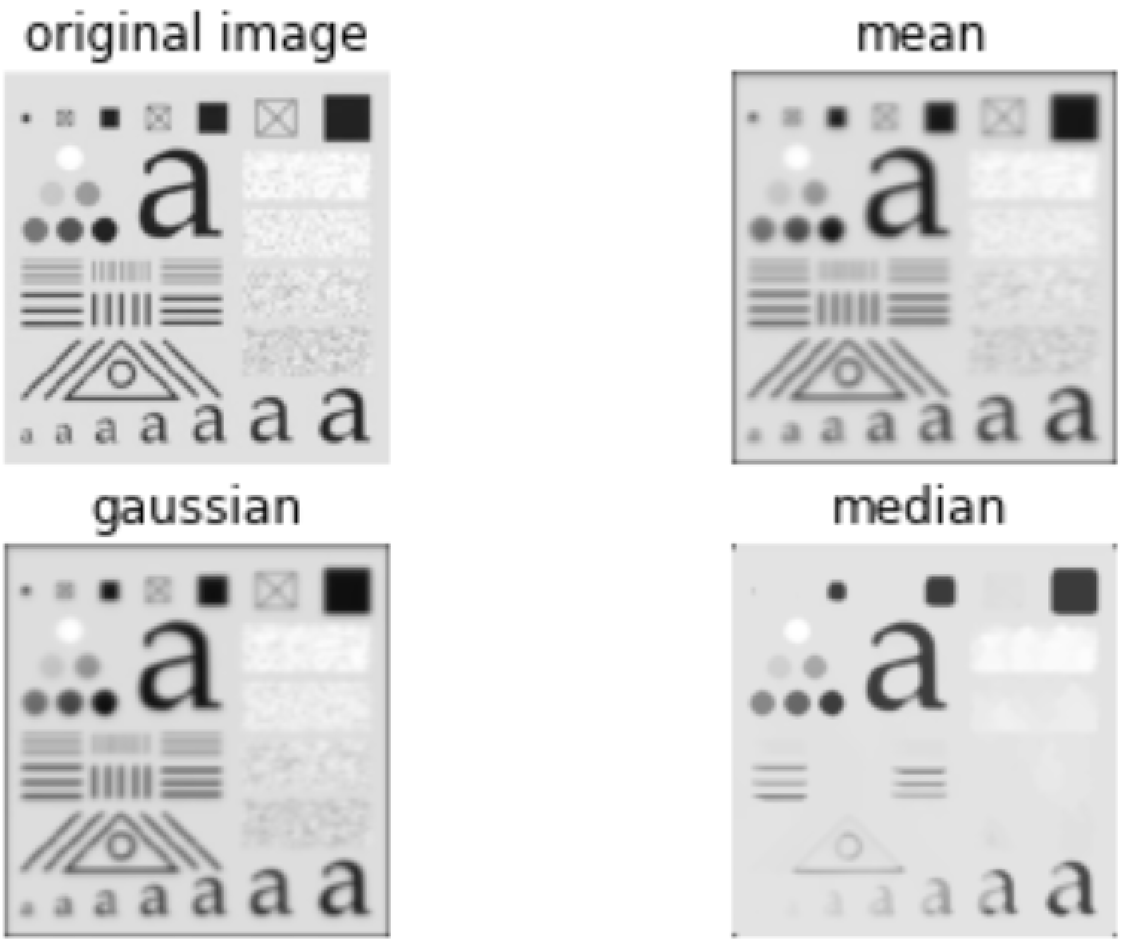
```
        for j in range(half_length, width+half_length): # for all x, y in temp img
            if method == 'median': # choose the median of patch, without kernel
                tempImg[i][j] = np.median(tempImg[i - half_length:i + half_length + 1,\
                                         j - half_length:j + half_length + 1])
            else: #else: convolution
                tempImg[i][j] = np.sum(kernel * tempImg[i - half_length:i + half_length + 1,\
                                               j - half_length:j + half_length + 1])

    newImg = tempImg[half_length:height + half_length, half_length:width + half_length]
    return newImg

```

我们测试patch=7, 15的情况，结果如下：

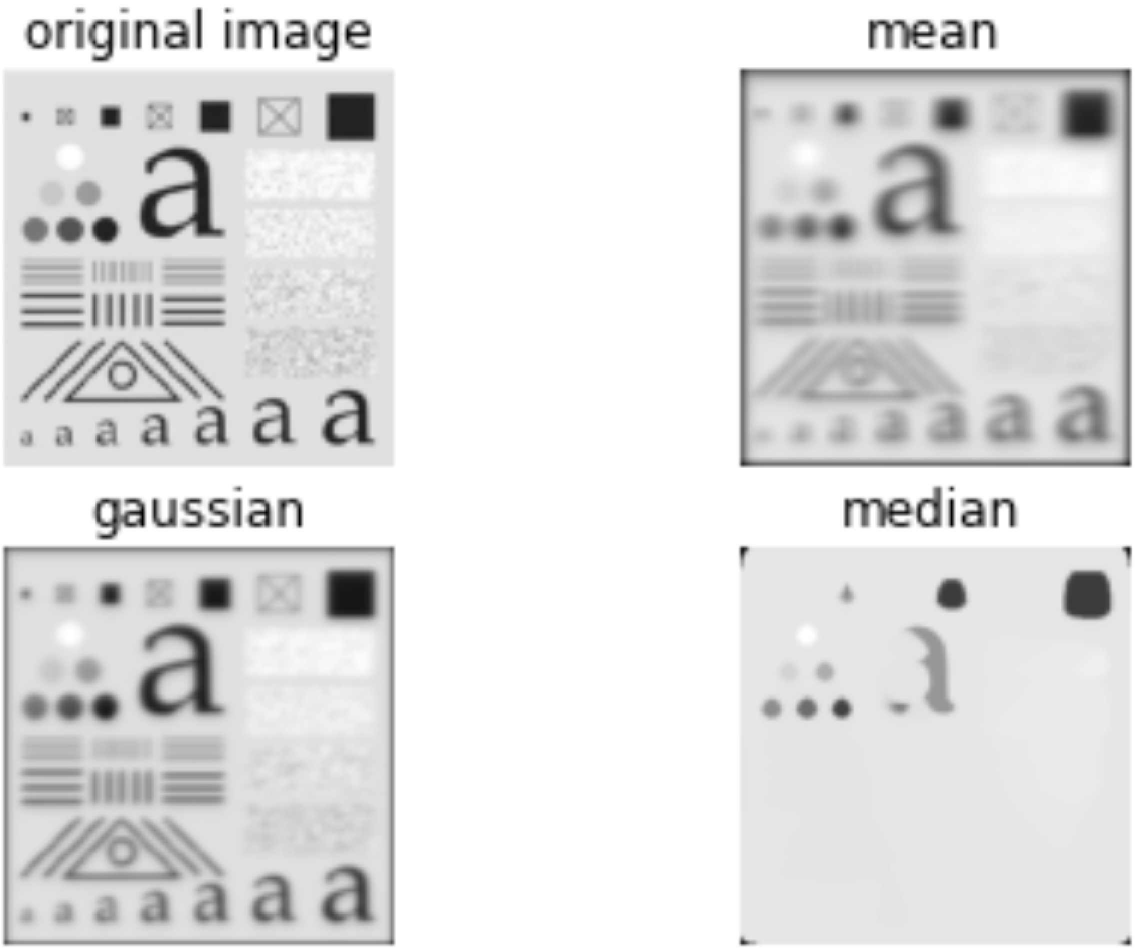
patch_length=7



在patch长度为7时，gauss核和直接均值的滤波器结果看起来相近，字的边缘变模糊，而中位数滤波器失去了很多信息，比如右边的噪点和三角形几乎看不见。一些很细的线条也无法识别。

周围黑色的边框出现的原因是我们加的tempimg边框默认是黑色。

patch_length=15



在patch长度为15时gauss核还基本保留原来的信息，均值滤波器看起来以及很模糊了，中位数滤波器丢失了大部分的信息几乎都看不见，只剩下了灰色的背景。

在patch减小为3的时候三种方法都较好的保留了原来的信息，模糊的不明显。

我们再次测试了不同sigma参数时候的高斯核滤波器的区别。增加sigma的值比较输出结果。

图中为sigma=20的结果

从图中可以看出当sigma变大的时候高斯分布更加平，中心点和距离为1的点的差距没有那么大，导致越来越接近均值滤波器的输出结果。

检验锐化算法：

拉普拉斯锐化与上一题的平滑滤波过程非常相似，不同的是我们的核不同，实现拉普拉斯算子的核：

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

然后用$f - w\nabla^2 f$得到最终图像。最后需要将图片范围归一到0-255范围。

具体代码如下:

```
def Laplace(img, len_patch, w):
    [height, width]=img.shape
    newImg = np.zeros([height,width],np.float64)
    half_length = (len_patch - 1) // 2 # half of patch length
    kernel = np.ones([len_patch,len_patch],np.float64)
    tempImg = np.zeros((height + len_patch - 1, width + len_patch - 1)) # temp matrix for convenient calculation
    tempImg[half_length: height + half_length, half_length: width + half_length] = img
    new_height, new_width = tempImg.shape

    kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])  # laplace operator
    hessian = np.zeros((new_height, new_width))      # Second derivative
    for i in range(half_length, new_height - half_length):  # for all x, y in temp img
        for j in range(half_length, new_width - half_length):
            hessian[i][j] = np.sum(kernel * tempImg[i - half_length:i + half_length + 1, j - half_length:j + half_length + 1]) #convolution
    tempImg = tempImg - w * hessian      # tempImg - hessian matrix
    newImg = tempImg[half_length:new_height - half_length, half_length:new_width - half_length]  # new img
    newImg = (newImg - newImg.min()) / (newImg.max() - newImg.min()) * 255  # restrict to [0, 255]
    return newImg
```

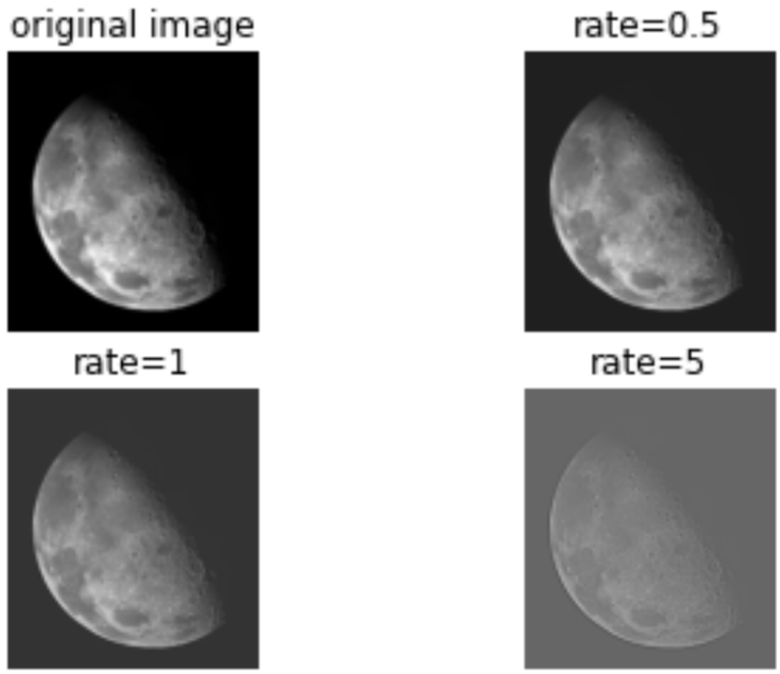
highboost锐化就是利用平滑的结果，将原图减去模糊的图像得到mask，再用$f + k \cdot mask$得到处理后的图像。

具体代码：

```
def highboost(img, len_patch, k):
    f_bar = smoothing(img, len_patch,'mean') # calculate f_bar
    mask = img-f_bar.astype(int)
    newImg = img + k * mask # new img
    newImg = (newImg - newImg.min()) / (newImg.max() - newImg.min()) * 255  # restrict to [0, 255]
    return newImg
```

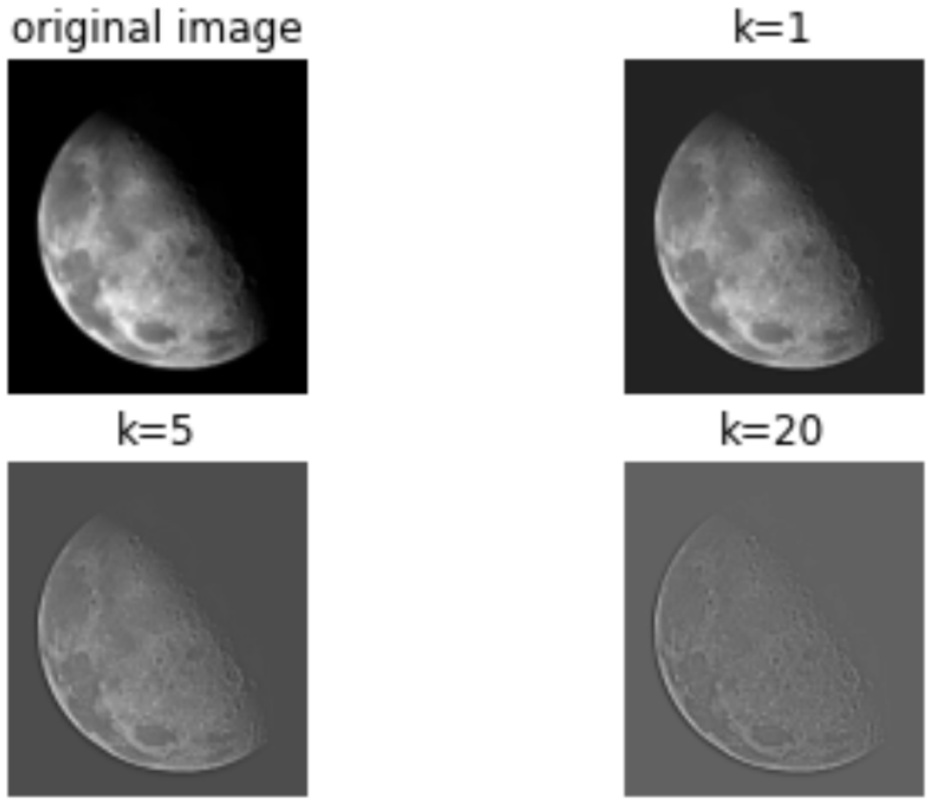
拉普拉斯结果

如图所示，我们测试了不同的w的输出结果

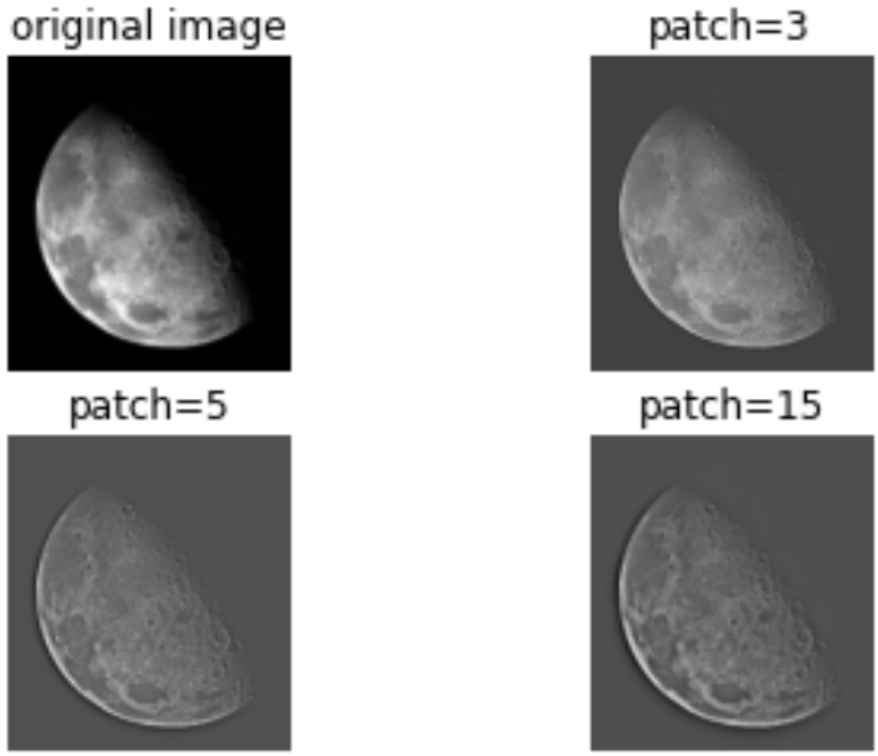


当w小的时候结果锐化结果不明显，当w逐渐变大，图片的边界变得清晰，但是图片的颜色也变灰了。

查看highboost的效果，可以看到随着k的改变，图片变化的趋势和上图w的趋势类似，也间接证明了二者的等价性。



同时对比了不同patch大小下的锐化结果



当patch变大的时候，图像物体边缘更清晰，但图像也有些失真。可能是因为patch过大以后损失信息过大。

证明：

（1）证明冲击窜（impulse train）的傅里叶变换后的频域表达式也是一个冲击窜。

(1) 证明冲击串的Frouier变换后频域表达式也是冲击串.

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

互换 μ t

$F(t)=\int_{-\infty}^{\infty} f(\mu)e^{-j2\pi\mu t}d\mu.$

令 μ 为 $-\mu$ $F(t)=\int_{-\infty}^{\infty} f(-\mu)e^{j2\pi(-\mu)t}d(-\mu)=\mathcal{F}^{-1}\{f(-\mu)\}$

$\therefore$ $F(t)\Longleftrightarrow F(-\mu)$ 是一个变换对.

$\mathcal{F}\{\delta(t-t_0)\}=e^{-j2\pi\mu t_0}$ 利用对称性.

$\mathcal{F}\{e^{-j2\pi t_0 t}\}=\delta(-\mu-t_0)=\delta(\mu+t_0)$ (1)

$\because$ $S_{\Delta T}(t)$ 周期为T. 写成Fourier级数则为:

$S_{\Delta T}(t)=\sum_{n=-\infty}^{\infty} C_n e^{j\frac{2\pi n}{\Delta T}t}$ 其中 $C_n=\frac{1}{\Delta T}\int_{-\frac{\Delta T}{2}}^{\frac{\Delta T}{2}} S_{\Delta T}(t)e^{-j\frac{2\pi n}{\Delta T}t}dt$

由采样性质 $C_n=\frac{1}{\Delta T}\int_{-\frac{\Delta T}{2}}^{\frac{\Delta T}{2}}\delta(t)e^{-j\frac{2\pi n}{\Delta T}t}dt$

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

$\therefore$ $S_{\Delta T}(t)=\sum_{n=-\infty}^{\infty}\frac{1}{\Delta T}e^{j\frac{2\pi n}{\Delta T}t}$

$\Rightarrow \mathcal{F}(S_{\Delta T}(t))=\mathcal{F}\{\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}e^{j\frac{2\pi n}{\Delta T}t}\}$

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

由(1), 上式 $=\frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta(u-\frac{n}{\Delta T})$

(2) 证明实信号f(x)的离散频域变换结果是共轭对称的。

2. 证明$f(x)$实信号的离散频域变换结果共轭对称

即. $F^*(u)=F(-u)$　　$F(u)$为$f(x)$的DFT

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

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

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

(积共轭=共轭的积)

$$= \sum_{x=0}^{N-1} f(x)^*\, e^{j2\pi ux/N}$$

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

$$= F(-u) \quad \text{QED.}$$

(3）证明二维变量的离散傅里叶变换的卷积定理。

3. 证明二维变量离散Fourier变换的卷积定理.

①. 先证 $f(x,y)*h(x,y) \Leftrightarrow F(u,v)\cdot H(u,v)$

$$\mathcal{F}(f(x,y)*h(x,y))$$

$$= \mathcal{F}\left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\, h(x-m,y-n)\right]$$

$$= \sum_x\sum_y\sum_m\sum_n f(m,n)\, h(x-m,y-n)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$= \sum_m\sum_n f(m,n)\left[\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x-m,y-n)\, e^{-j2\pi\left(\frac{u(x-m)}{M}+\frac{v(y-n)}{N}\right)}\right] \cdot e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

let $\alpha=x-m$ $\beta=y-n$

$$\text{上式} = \sum_m\sum_n f(m,n)\left[\sum_{\alpha=-m}^{M-m-1}\sum_{\beta=-n}^{N-n-1} h(\alpha,\beta)\, e^{-j2\pi\left(\frac{u\alpha}{M}+\frac{v\beta}{N}\right)}\right] \cdot e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

周期性.

$$= \sum_m\sum_n f(m,n)\left[\sum_{\alpha=0}^{M-1}\sum_{\beta=0}^{N-1} h(\alpha,\beta)\, e^{-j2\pi\left(\frac{u\alpha}{M}+\frac{v\beta}{N}\right)}\right] \cdot e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

提出来

$$= \sum_m\sum_n f(m,n)\, H(u,v)\, e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$= H(u,v)\left\{\sum_m\sum_n f(m,n)\, e^{-j2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}\right\}$$

$$= H(u,v)\, F(u,v)$$

$$\therefore f(x,y)*h(x,y) \Longleftrightarrow F(u,v)\cdot H(u,v)$$

(2). 证 $f(x,y)\,h(x,y) \Leftrightarrow \frac{1}{MN} F(u,v) * H(u,v)$

$\mathcal{F}^{-1}$ 表示IDFT

$$\mathcal{F}^{-1}\{\frac{1}{MN} F(u,v) * H(u,v)\}$$

$$= \mathcal{F}^{-1}\{\frac{1}{(MN)^2} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(m,n) H(u-m, v-n)\, e^{j2\pi(\frac{\mu x}{M} + \frac{\nu y}{N})}$$

$$= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F(m,n) \{\frac{1}{MN} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} H(\mu-m, \nu-n) \cdot$$

$$e^{j2\pi(\frac{x(\mu-m)}{M} + \frac{y(\nu-n)}{N})}\}\, e^{j2\pi(\frac{xm}{M} + \frac{yn}{N})}$$

let $\alpha = \mu - m$ $\beta = \nu - n$ 同(1)中周期性.

$$上式 = \frac{1}{MN} \sum_m \sum_n F(m,n) \{\frac{1}{MN} \sum_{\alpha=0}^{M-1} \sum_{\beta=0}^{N-1} H(\alpha, \beta)\, e^{j2\pi(\frac{x\alpha}{M} + \frac{y\beta}{N})}\}$$

$$\cdot\, e^{j2\pi(\frac{xm}{M} + \frac{yn}{N})}$$

$$= h(x,y) \cdot \{\frac{1}{MN} \sum_m \sum_n F(m,n)\, e^{j2\pi(\frac{xm}{M} + \frac{yn}{N})}\}$$

$$= h(x,y) \cdot f(x,y)$$

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

由 (1) (2) QED