Image Processing and Pattern Recognition (IPPR)

Chapter3:Image Enhancement in the Frequency Domain

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

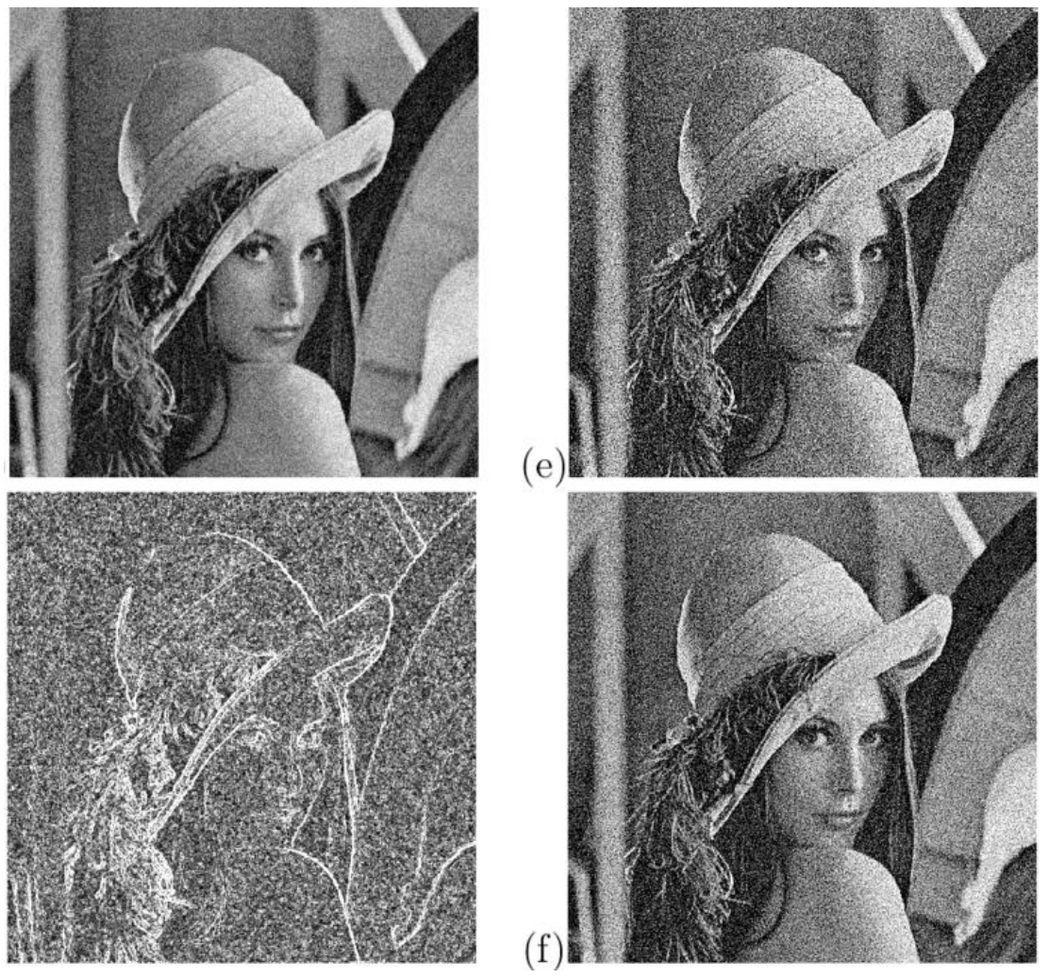
Member, Laboratory for ICT Research and Development (LICT)

Institute of Engineering

basanta@ioe.edu.np

http://www.basantajoshi.com.np

https://scholar.google.com/citations?user=iocLiGcAAAAJ
https://www.researchgate.net/profile/Basanta_Joshi2



Contents

In this lecture we will look at image enhancement in the frequency domain

- Jean Baptiste Joseph Fourier
- The Fourier series & the Fourier transform
- Image Processing in the frequency domain
  - Image smoothing
  - Image sharpening
- Fast Fourier Transform

Jean Baptiste Joseph Fourier

orn in Auxerre,

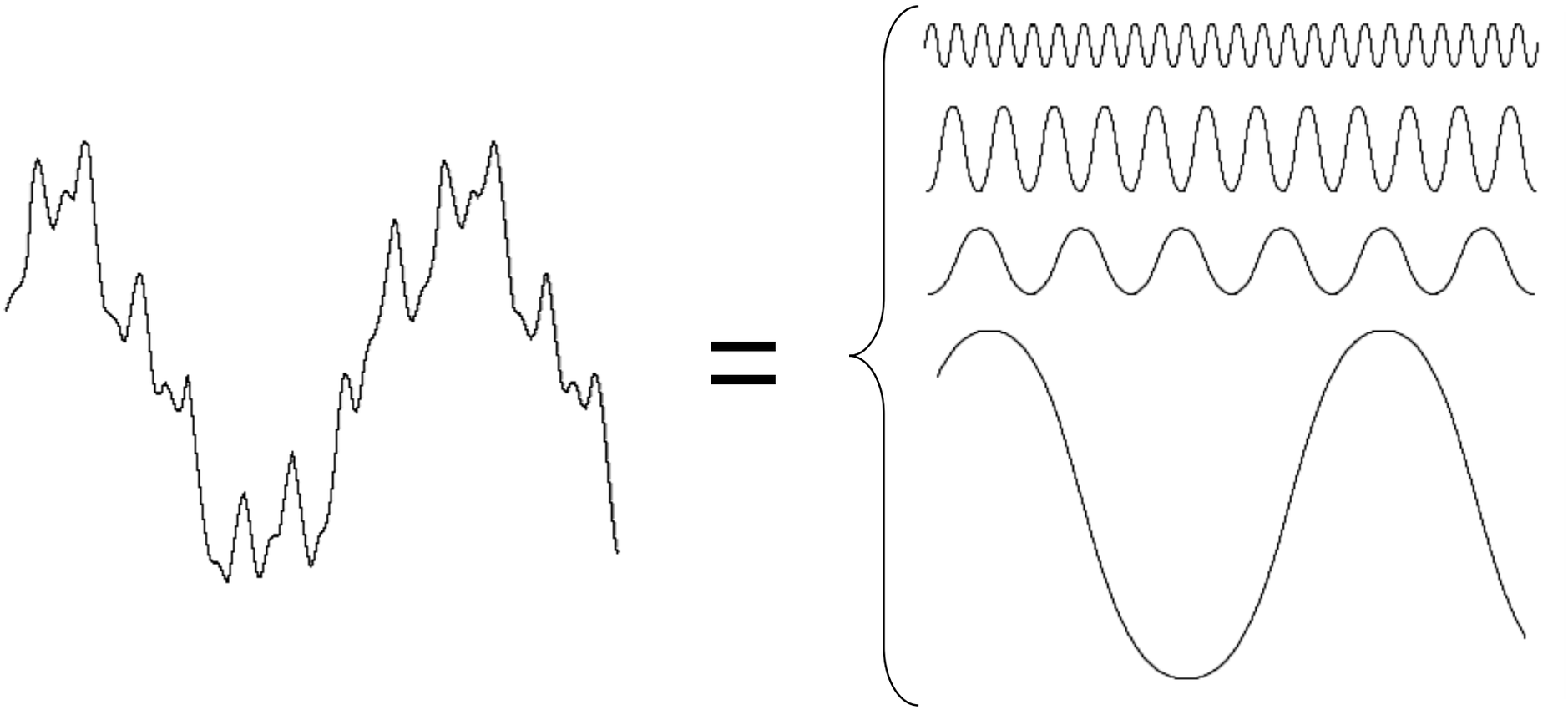
8

- Most famous for his work "*La Théorie Analitique de la Chaleur*" published in 1822
- Translated into English in 1878: "*The Analytic Theory of Heat*"

Nobody paid much attention when the work was first published

One of the most important mathematical theories in modern engineering

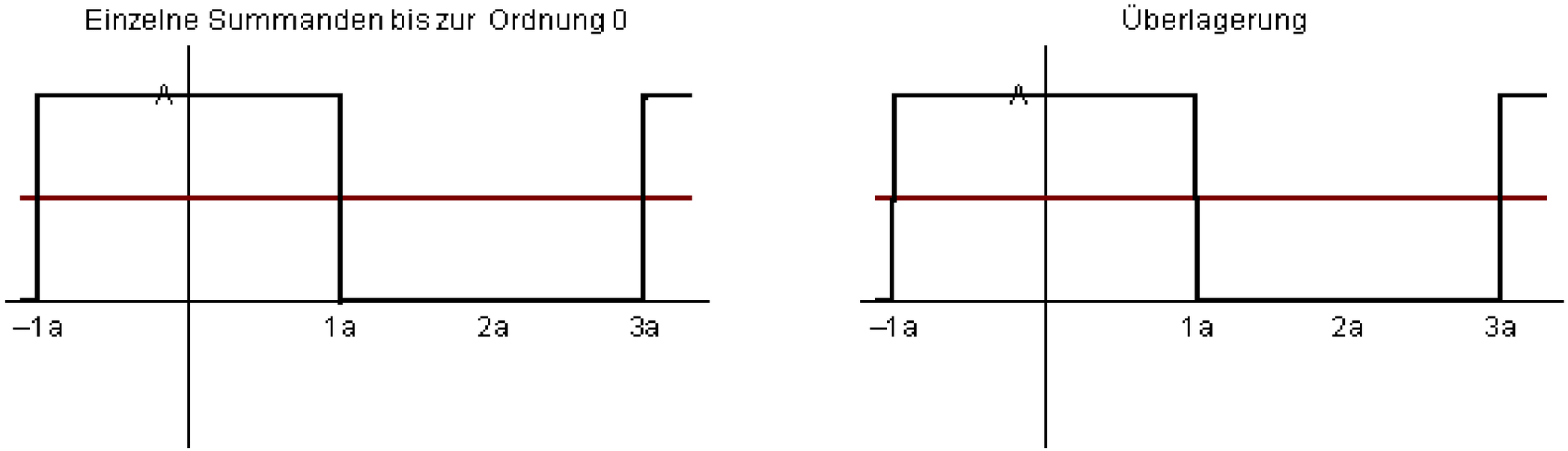
The Big Idea



Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

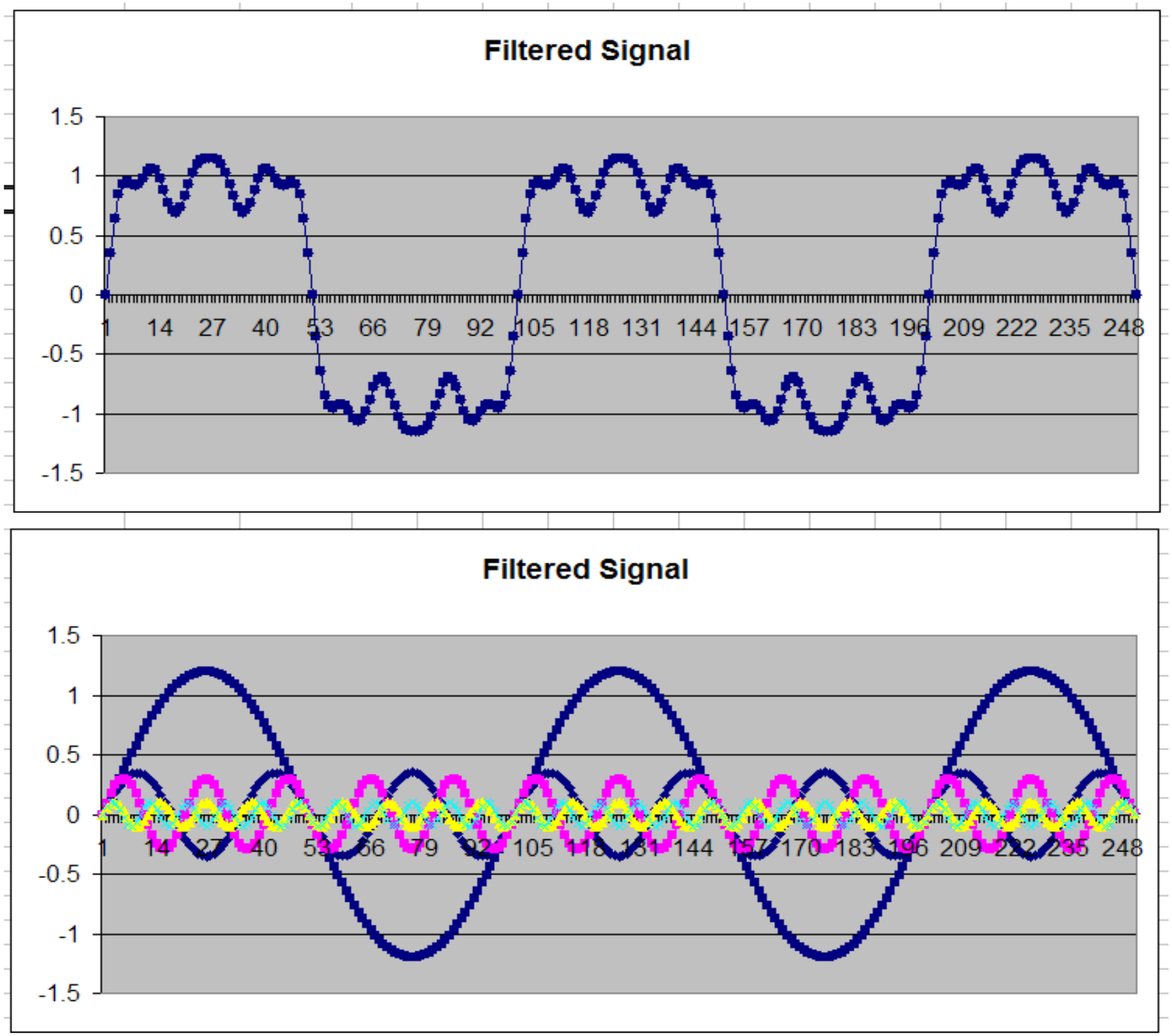
The Big Idea (cont…)



Notice how we get closer and closer to the original function as we add more and more frequencies

Taken from www.tfh-berlin.de/~schwenk/hobby/fourier/Welcome.html

The Big Idea (cont…)



Frequency domain signal processing example in Excel

Introduction to the Fourier Transform

- Let *f(x)* be a continuous function of a real variable *x*
- The Fourier transform of *f(x)*, denoted by $\Im\{f(x)\}$ is given by:

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{+\infty} f(x)\exp[-j2\pi ux]dx$$

- where $j = \sqrt{-1}$
- Given *F(u)*, *f(x)* can be obtained by using the *inverse Fourier transform:*

$$\Im^{-1}\{F(u)\} = f(x)$$
$$= \int_{-\infty}^{+\infty} F(u)\exp[j2\pi ux]du.$$

The Fourier Transform (continued)

- These two equations, called the Fourier transform pair, exist if f(x) is continuous and integrable and F(u) is integrable.
- These conditions are almost always satisfied in practice.
- We are concerned with functions f(x) which are real, however the Fourier transform of a real function is, generally, complex. So,

$$F(u) = R(u) + jI(u)$$

- where R(u) and I(u) denote the real and imaginary components of F(u) respectively.

The Fourier Transform (continued)

- Expressed in exponential form, F(u) is:

$$F(u) = |F(u)|e^{j\varphi(u)}$$

- where $|F(u)| = \sqrt{R^2(u) + I^2(u)}$

- and $\varphi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]$

- The magnitude function |F(u)| is called the *Fourier spectrum* of f(x)
- and φ(u) is the phase angle.

The Fourier Transform (continued)

- The square of the spectrum,

$$P(u) = |F(u)|^2$$
$$= R^2(u) + I^2(u)$$

- is commonly called the *power spectrum* (or the *spectral density*) of f(x).
- The variable *u* is often called the *frequency variable*. This name arises from the expression of the exponential term exp[-j2πux] in terms of sines and cosines (from Euler's formula):

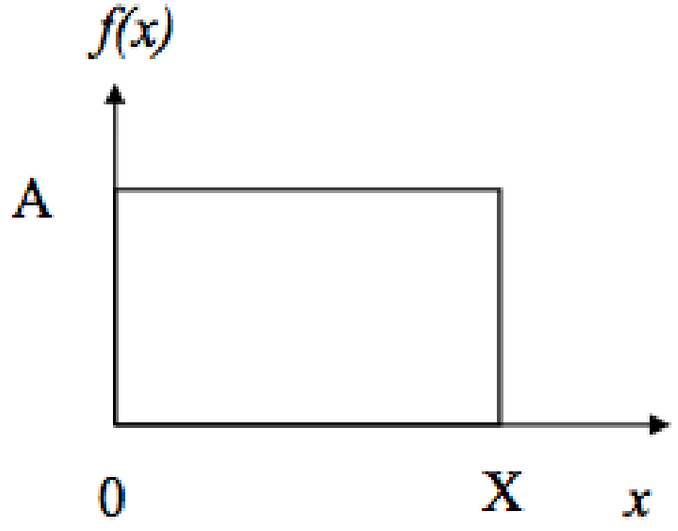
$$\exp[-j2\pi ux] = \cos(2\pi ux) - j\sin(2\pi ux)$$

The Fourier Transform (continued)

- Interpreting the integral in the Fourier transform equation as a limit summation of discrete terms make it obvious that:
  - *F(u)* is composed of an infinite sum of sine and cosine terms.
  - Each value of *u* determines the *frequency* of its corresponding sine-cosine pair.

The Fourier Transform (Example)

- Consider the following simple function. The Fourier transform is:



$$F(u) = \int_{-\infty}^{+\infty} f(x)\exp[-j2\pi ux]dx$$

$$= \int_{0}^{X} A\exp[-j2\pi ux]dx$$

$$= \frac{-A}{j2\pi u}[e^{-j2\pi ux}]_0^X = \frac{-A}{j2\pi u}[e^{-j2\pi uX} - 1]$$

$$= \frac{A}{j2\pi u}[e^{j\pi uX} - e^{-j\pi uX}]e^{-j\pi uX}$$

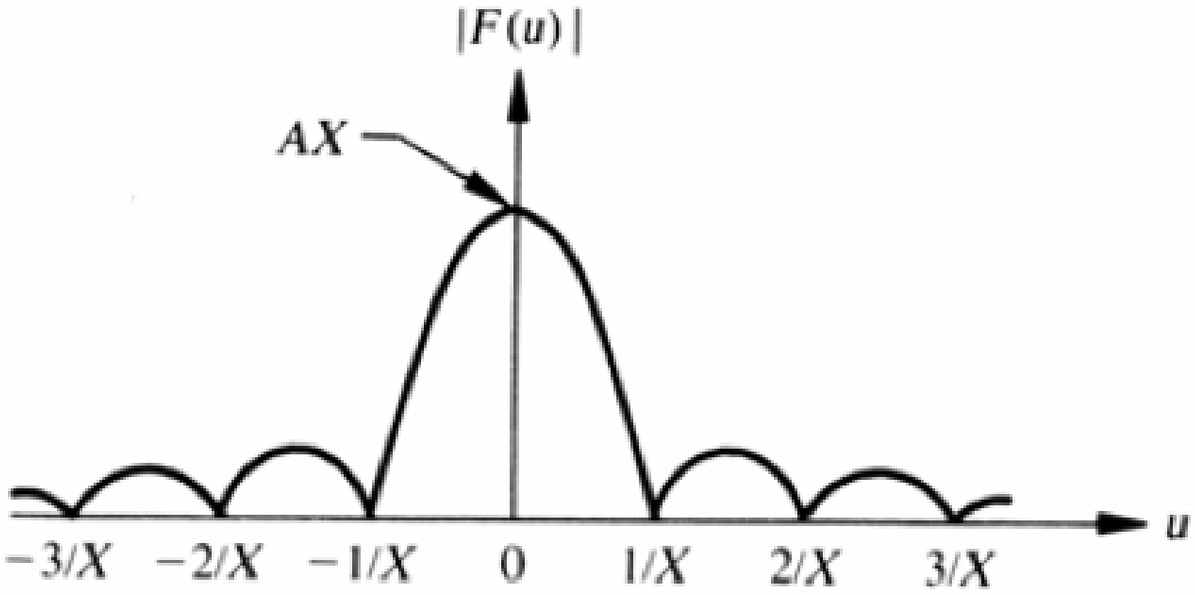
$$= \frac{A}{\pi u}\sin(\pi uX)e^{-j\pi uX}$$

The Fourier Transform (Example)

- This is a complex function. The Fourier spectrum is:

$$|F(u)| = \left|\frac{A}{\pi u}\right| \left|\sin(\pi u X)\right| \left|e^{-j\pi u X}\right|$$

$$= AX \left|\frac{\sin(\pi u X)}{(\pi u X)}\right|$$



- A plot of |F(u)| looks like the following:

The 2-D Fourier Transform

- The Fourier transform can be extended to 2 dimensions:

$$\Im\{f(x,y)\} = F(u,v) = \int_{-\infty}^{+\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy.$$

- and the inverse transform

$$\Im^{-1}\{F(u,v)\} = f(x,y) = \int_{-\infty}^{+\infty}\int F(u,v)\exp[j2\pi(ux+vy)]dudv.$$

The 2-D Fourier Transform (Continued)

- The 2-D *Fourier spectrum* is:

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$$

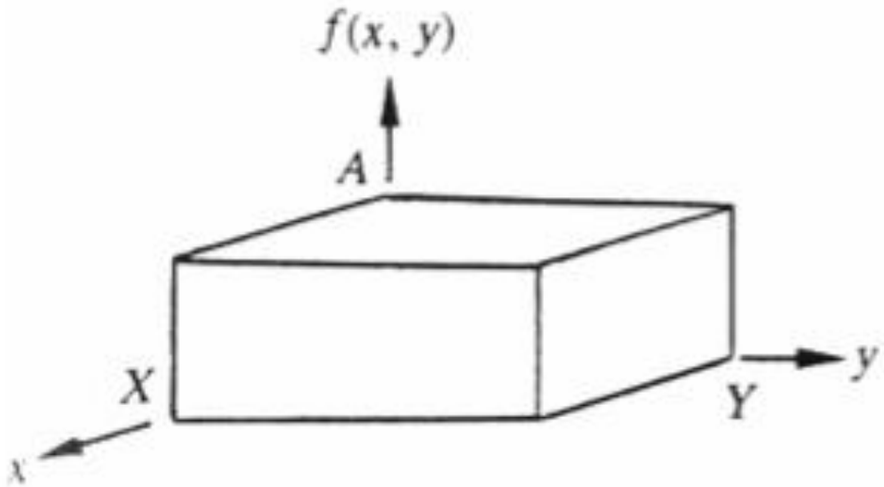
- The *phase angle* is:

$$\varphi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$$

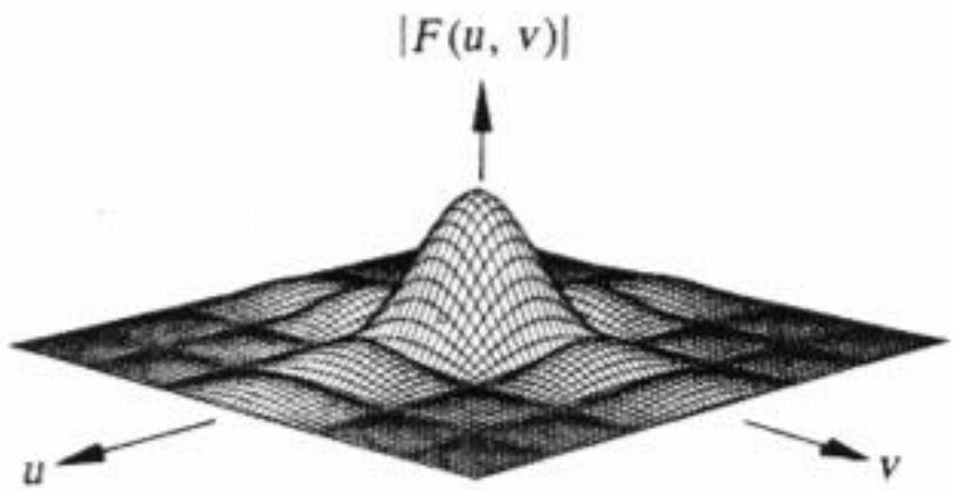
- The *power spectrum* is:

$$P(u,v) = |F(u,v)|^2$$
$$= R^2(u,v) + I^2(u,v)$$

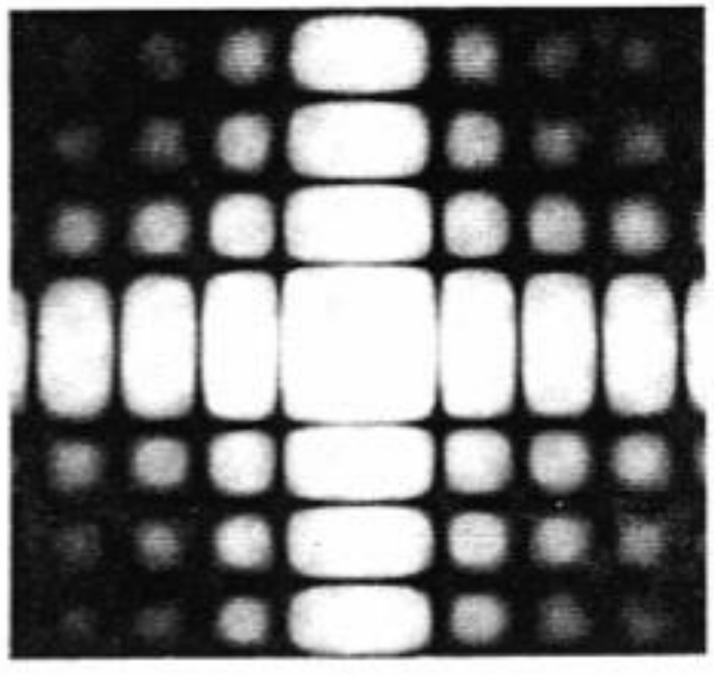
Sample 2-D function and its Fourier spectrum



(a)



(b)



(c)

Example 2-D Fourier transform

$$F(u,v) = \int_{-\infty}^{+\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy$$

$$= A\int_0^X \exp[-j2\pi ux]dx \int_0^Y \exp[-j2\pi vy]dy$$

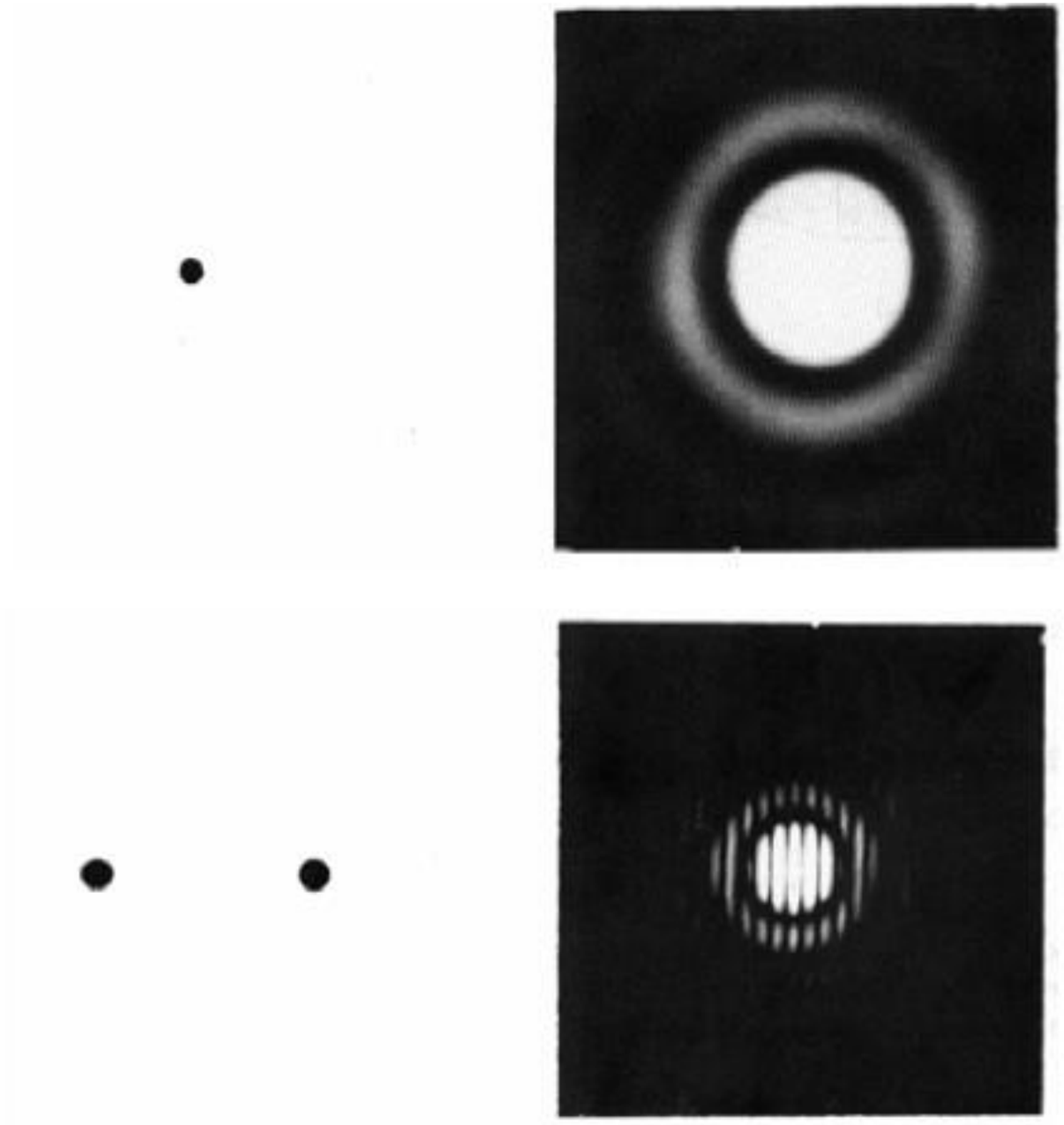
$$= A[\frac{e^{-j2\pi ux}}{-j2\pi u}]_0^X [\frac{e^{-j2\pi vy}}{-j2\pi v}]_0^Y$$

$$= \frac{A}{-j2\pi u}[e^{-j2\pi uX} - 1]\frac{1}{-j2\pi y}[e^{-j2\pi vY} - 1]$$

$$= AXY[\frac{\sin(\pi uX)e^{-j\pi uX}}{(\pi uX)}][\frac{\sin(\pi vY)e^{-j\pi vY}}{(\pi vY)}]$$

The spectrum is $|F(u,v)| = AXY[\frac{\sin(\pi uX)}{(\pi uX)}][\frac{\sin(\pi vY)}{(\pi vY)}]$

Example 2-D functions and their spectra



The discrete Fourier transform

- Suppose a continuous function, f(x), is discretized into a sequence

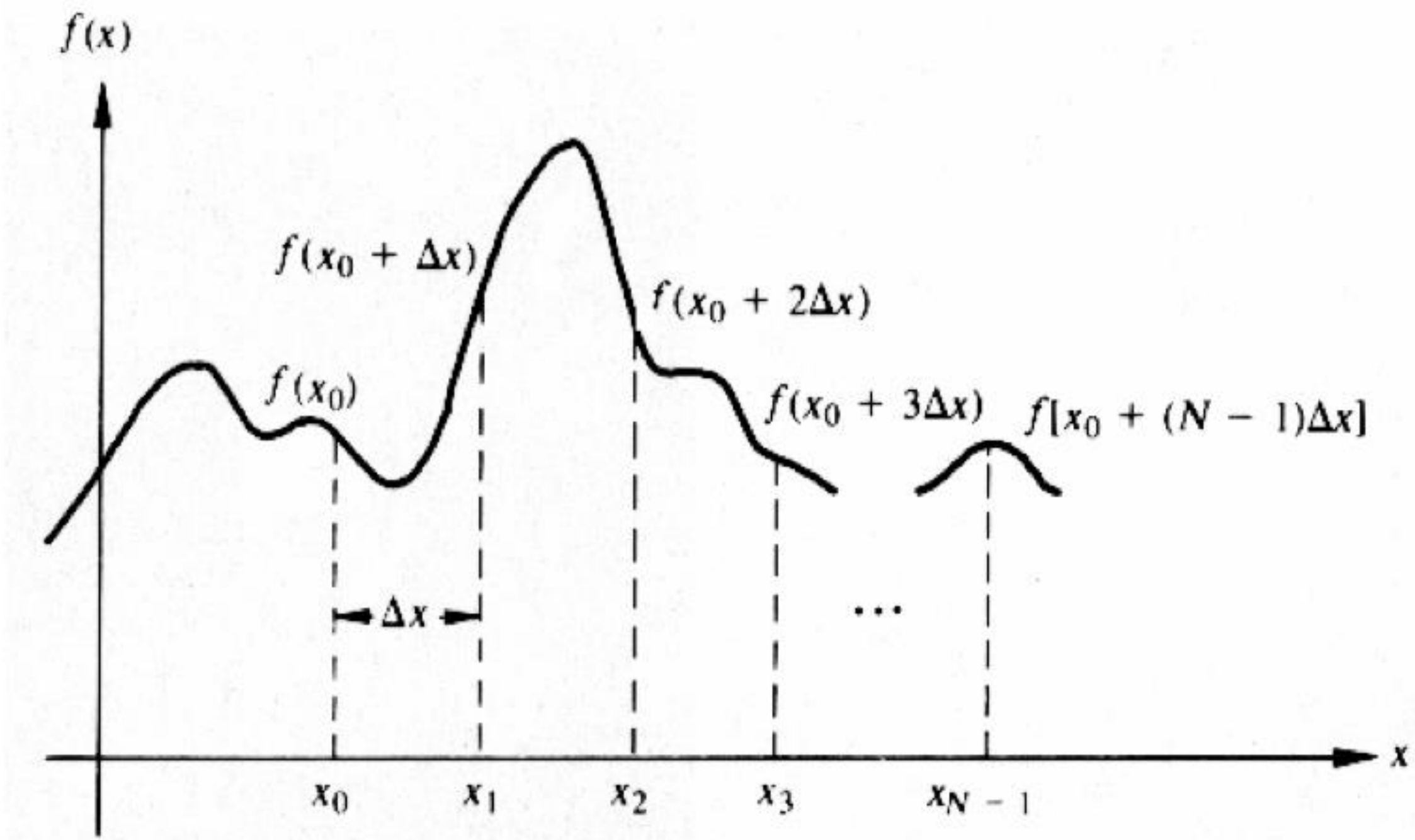
  $\{f(x_0), f(x_0+\Delta x), f(x_0+2\Delta x), \ldots, f(x_0+[N-1]\Delta x)\}$

- by taking N samples Δx units apart
- Let *x* refer to either a continuous or discrete value by saying

$$f(x) = f(x_0 + x\Delta x)$$

- where *x* assumes the discrete values 0, 1, …, N-1 and
- {f(0),f(1),…,f(N-1)} denotes any N uniformly spaced samples from a corresponding continuous function

Sampling a continuous function



The discrete Fourier transform pair

- The discrete Fourier transform is given by:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux / N]$$

- for u=0, 1, … ,N-1
- The discrete inverse Fourier transform is given by:

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux / N]$$

- for x=0, 1, … ,N-1
- The values of u=0, 1, … ,N-1in the discrete case correspond to samples of the continuous transform at 0, Δu, 2Δu, …, (N-1)Δu
  Δu and Δx are related by Δu=1/(N Δx)

The 2-D discrete Fourier transform

- In the 2-D case:

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

- for *u=0* $\rightarrow$ *M-1* and *v=0* $\rightarrow$ *N-1*

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux/M + vy/N)]$$

- for *x=0* $\rightarrow$ *M-1* and *y=0* $\rightarrow$ *N-1*
- The discrete function f(x,y) represents samples of the continuous function at $f(x_0+x\Delta x, y_0+y\Delta y)$

  $\Delta u = 1/(M\Delta x)$ *and* $\Delta v = 1/(N\Delta y)$

The 2-D discrete Fourier transform (continued)

- For the case when N=M (such as in a square image)

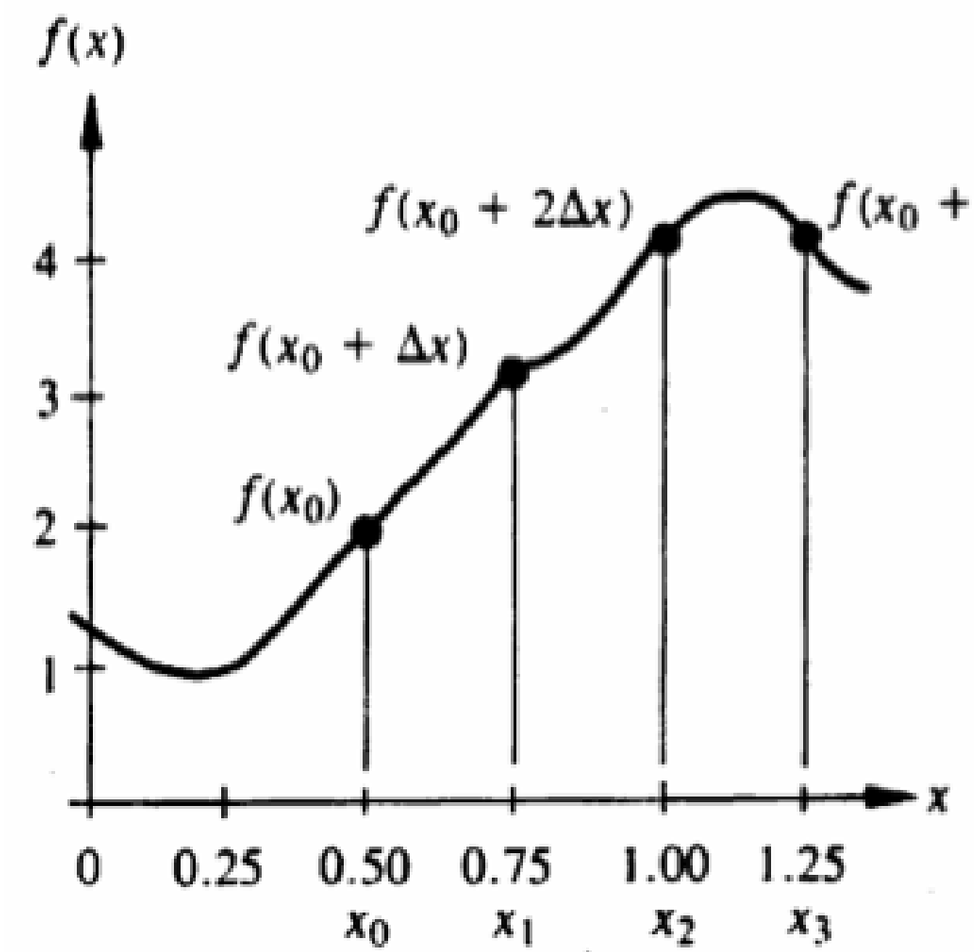
$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux+vy)/N]$$

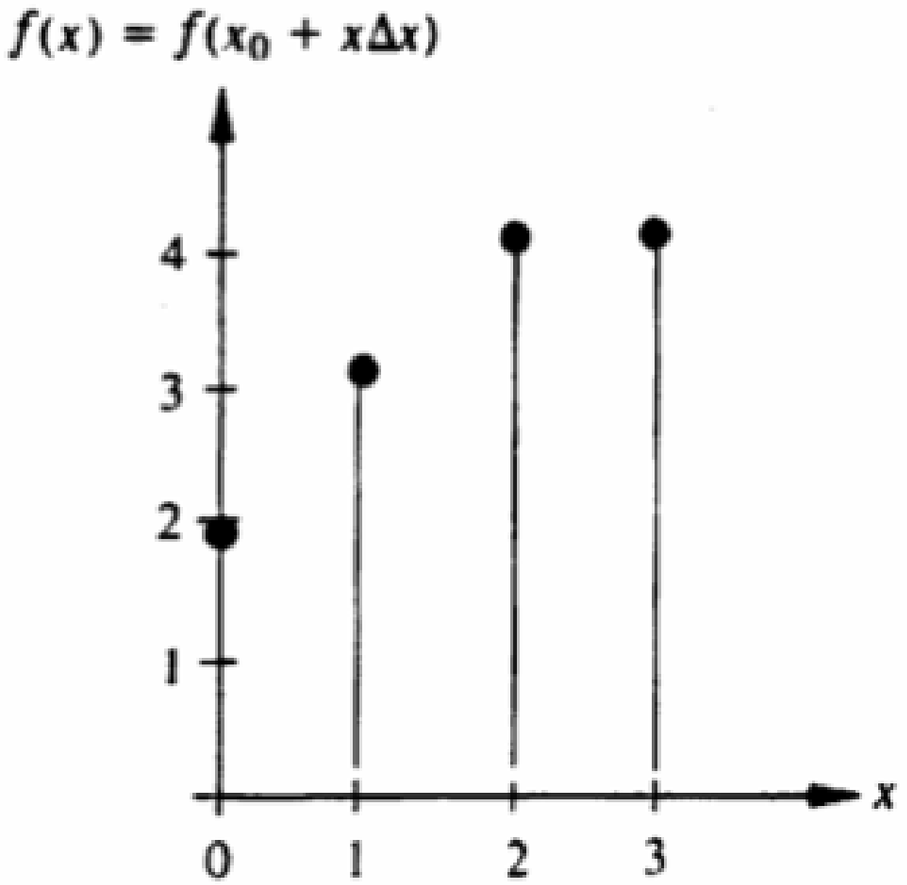
- and

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux+vy)/N]$$

- Note each expression in this case has a 1/N term. The grouping of these constant multiplier terms in the Fourier transform pair is arbitrary.

Discrete Fourier transform example





- Consider sampling at x_0=.5, x_1=.75, x_2=1.0, and x_3=1.25
- Here Δx=.25 and x ranges from 0 $\rightarrow$ 3

Discrete Fourier transform example (continued)

- The four corresponding Fourier transform terms are

$$F(0) = \frac{1}{4}\sum_{x=0}^{3} f(x)\exp[0]$$
$$= \frac{1}{4}[f(0)+f(1)+f(2)+f(3)]$$
$$= \frac{1}{4}[2+3+4+4]$$
$$= 3.25$$

$$F(1) = \frac{1}{4}\sum_{x=0}^{3} f(x)\exp[-j2\pi/4]$$
$$= \frac{1}{4}[2e^{0}+3e^{-j\pi/2}+4e^{-j\pi}+4e^{-j3\pi/2}]$$
$$= \frac{1}{4}[-2+j]$$

$$F(2) = -\frac{1}{4}[1+0j]$$

$$F(3) = -\frac{1}{4}[2+j]$$

Discrete Fourier transform example (continued)

- The Fourier spectrum is then

$$|F(0)| = 3.25$$

$$|F(1)| = [(2/4)^2 + (1/4)^2]^{1/2} = \sqrt{5}/4$$

$$|F(2)| = [(1/4)^2 + (0/4)^2]^{1/2} = 1/4$$

$$|F(3)| = [(2/4)^2 + (1/4)^2]^{1/2} = \sqrt{5}/4$$

Properties of the 2-D Fourier transform

- The dynamic range of the Fourier spectra is generally higher than can be displayed
- A common technique is to display the function

$$D(u,v) = c\log\left[1 + \left|F(u,v)\right|\right]$$

- where *c* is a scaling factor and the logarithm function performs a “compression” of the data
- *c* is usually chosen to scale the data into the range of the display device, [0-255] typically ([1-256] for 256 gray-level MATLAB image)

Separability

- The discrete transform pair can be written in separable forms

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\exp[-j2\pi ux/N]\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi vy/N]$$

- for $u,v = 0,1,\ldots,N-1$

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\exp[j2\pi ux/N]\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi vy/N]$$

- for $x,y = 0,1,\ldots,N-1$
- So, F(u,v) or f(x,y) can be obtained in 2 steps by successive applications of the 1-D Fourier transform or its inverse.

Separability (continued)

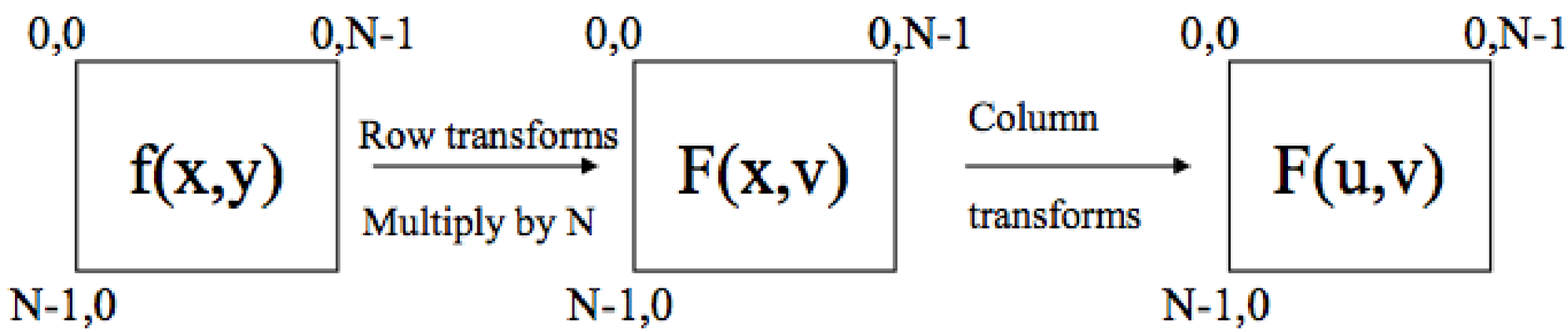
- The 2-D transform can be expressed as

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} F(x,v)\exp[-j2\pi ux/N]$$

- where

$$F(x,v) = N\left[\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi vy/N]\right]$$

- Graphically, the process is as follows



Translation

- The translation properties of the Fourier transform pair are

$$f(x,y)\exp[j2\pi(u_0x+v_0y)/N] \Leftrightarrow F(u-u_0,v-v_0)$$

- and

$$f(x-x_0,y-y_0) \Leftrightarrow F(u,v)\exp[-j2\pi(ux_0+vy_0)/N]$$

- where the double arrow indicates a correspondence between a function and its Fourier transform (or vice versa)
- Multiplying *f(x,y)* by the exponential and taking the transform results in a shift of the origin of the frequency plane to the point (u_0,v_0).

Translation (continued)

- For our purposes, $u_0=v_0=N/2$. Therefore,

$$\exp[j2\pi(u_0x+v_0y)/N] = e^{j\pi(x+y)}$$
$$= (-1)^{x+y}$$

- and

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-N/2, v-N/2)$$

- So, the origin of the Fourier transform of f(x,y) can be moved to the center of the corresponding NxN simply by multiplying f(x,y) by $(-1)^{x+y}$ before taking the transform
- Note: This does not affect the magnitude of the Fourier transform

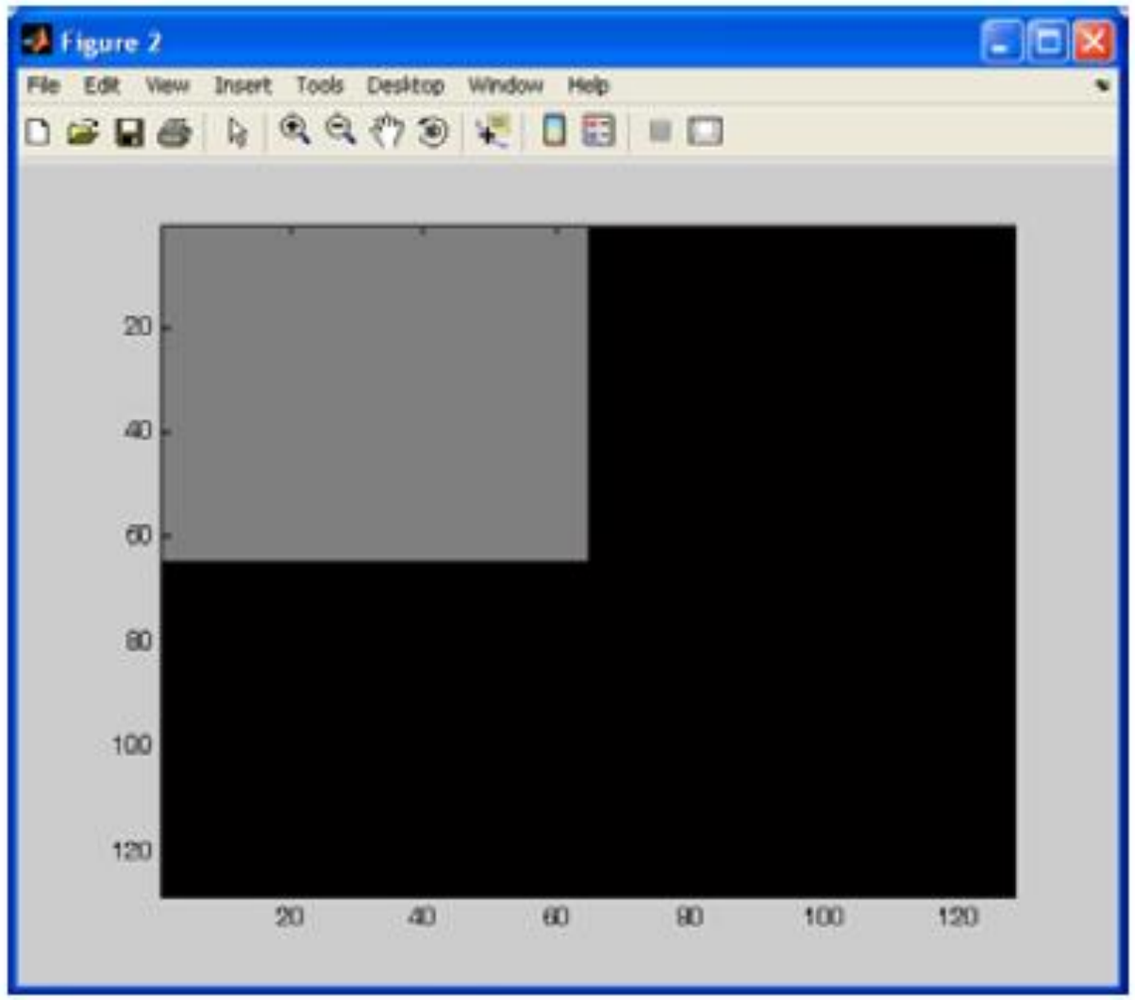
Matlab example

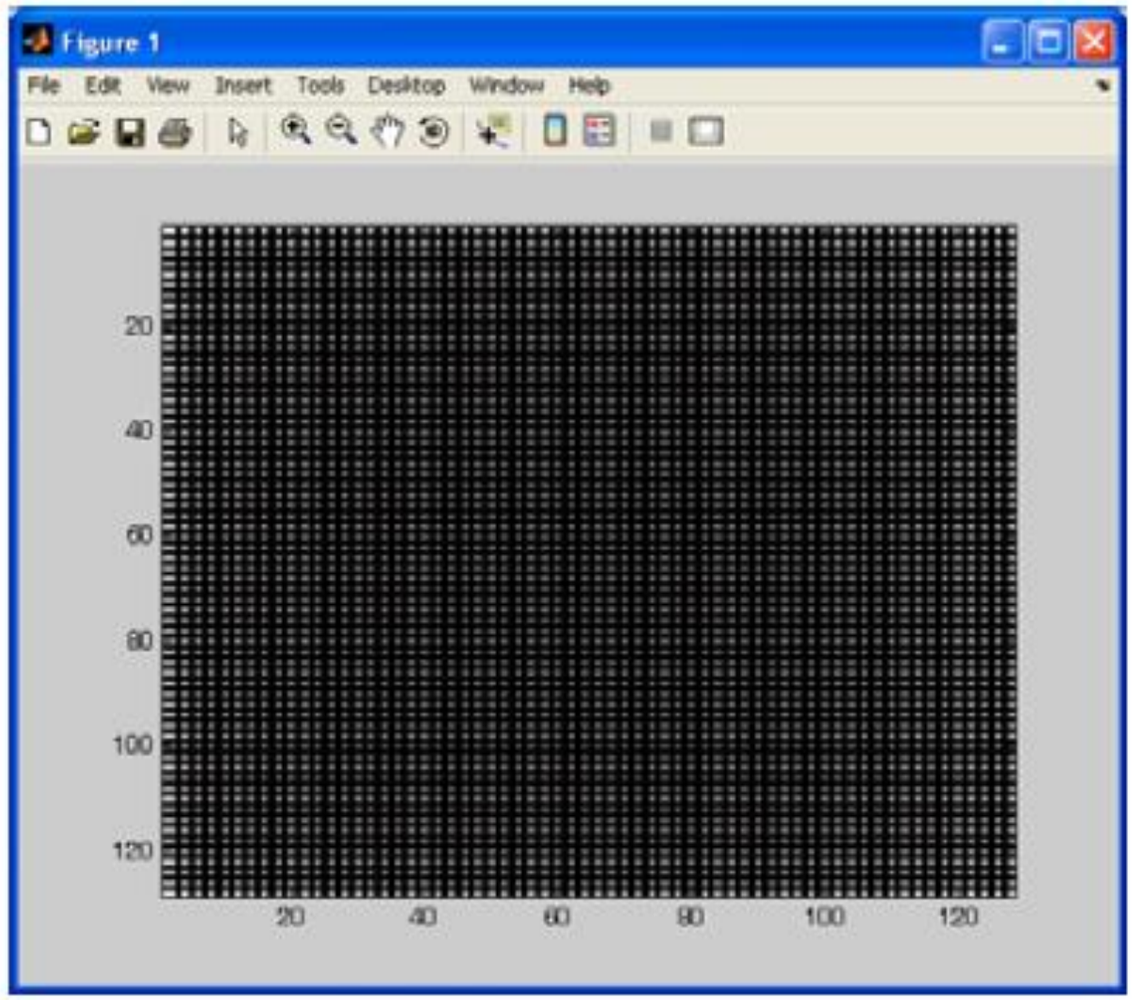
```
%Create data for the test
f=zeros(128);
for x=1:64
    for y=1:64
        f(x,y)=128;
    end
end
% Perform a translation shift on f(x,y)
for x=1:128
    for y=1:128
        f(x,y)=f(x,y)*((-1)^(x+y));
    end
end
```

Matlab example (continued)

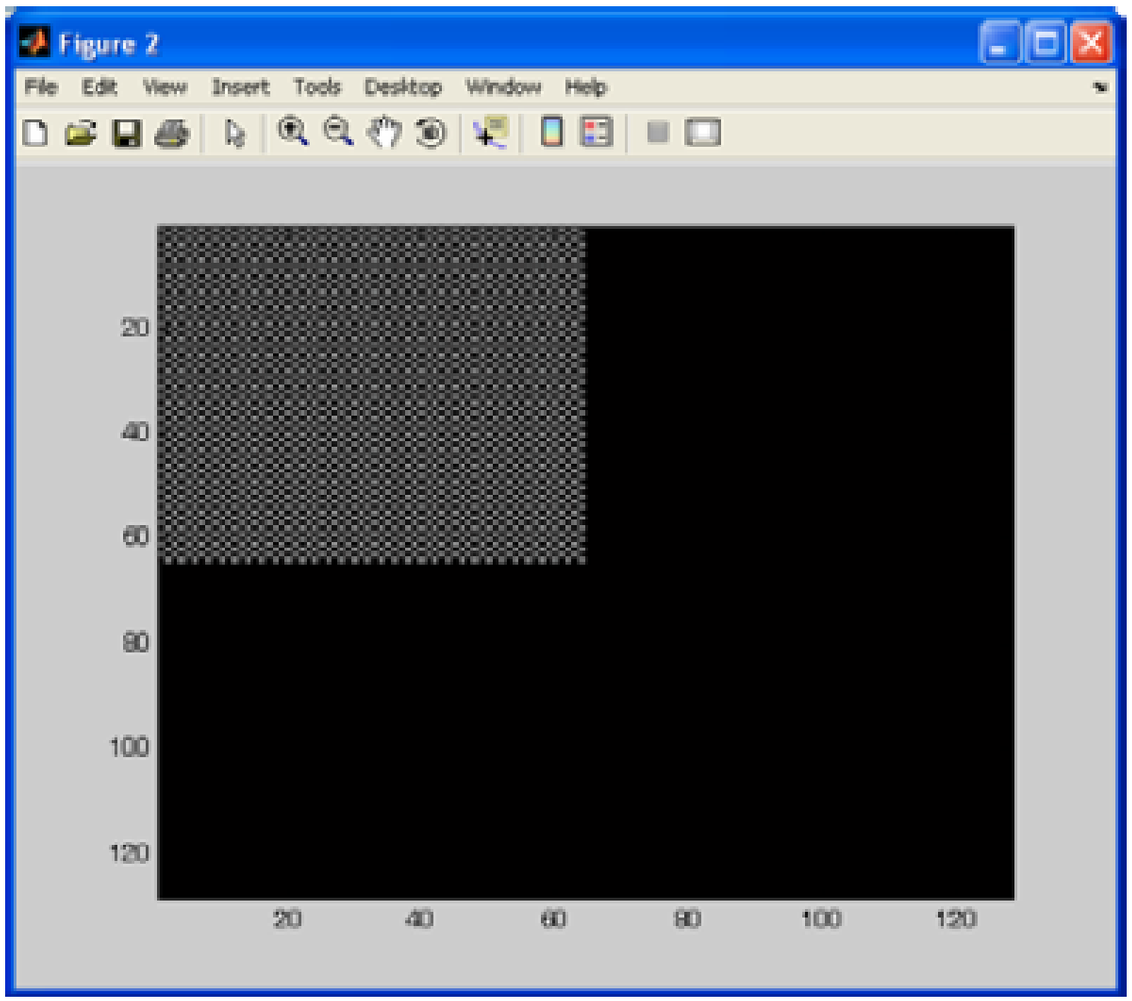
```
% Compute the 2-D discrete Fourier transform
F=fft2(f);
% Compute the Fourier spectrum
Fspect=sqrt(real(F).^2+imag(F).^2);
% Construct a scaling factor based on
% the dynamic range of the spectrum
FspectMAX=max(max(Fspect));
% Compute D, the scaled data
D=(256/(log(1+FspectMAX)))*log(1+Fspect);
figure(1);
% Plot, as an image, a subset of D
image(D(56:74,56:74));colormap(gray(256));
```

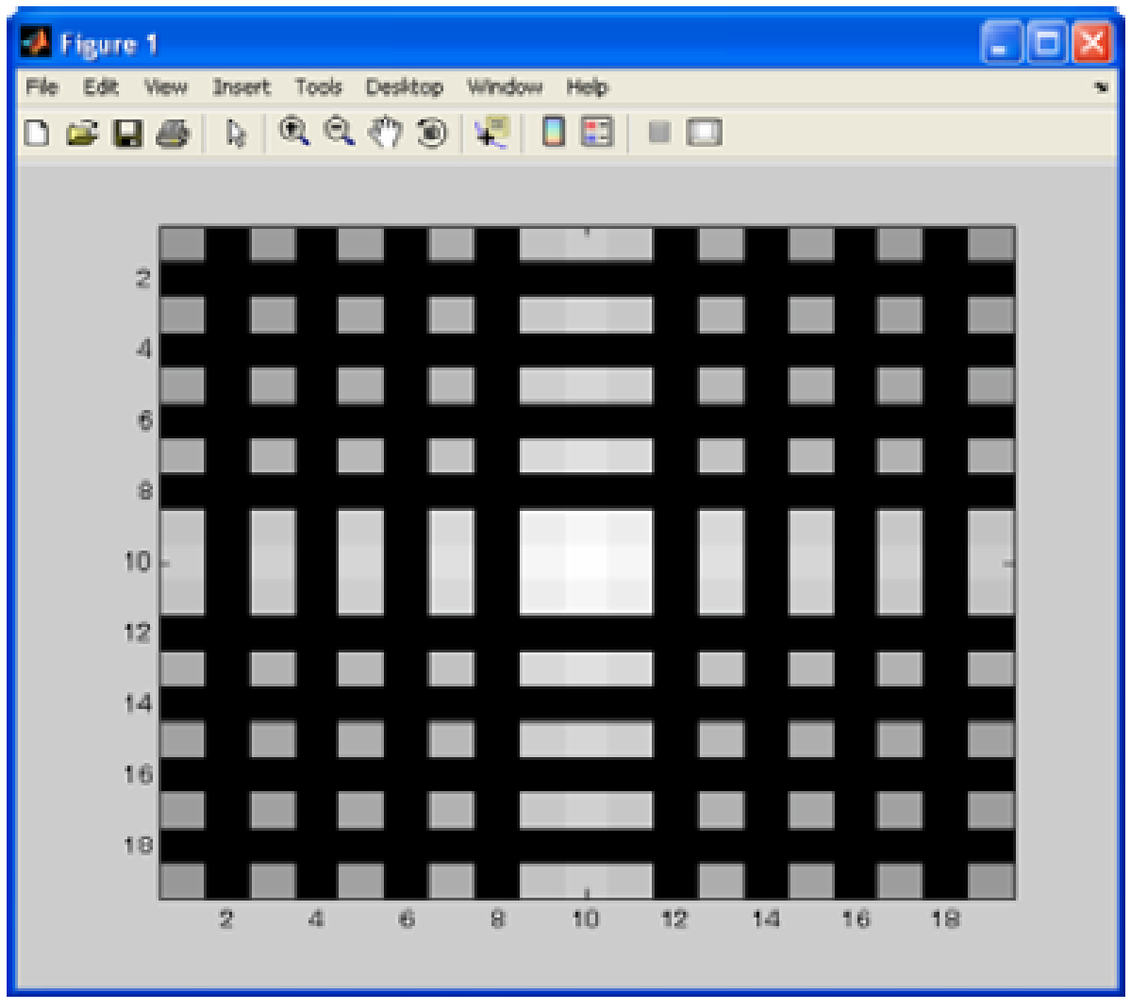
Example image and complete, scaled Fourier spectrum plot





Example image and partial, scaled Fourier spectrum plot (with shifted f(x,y))





Periodicity of the Fourier transform

- The discrete Fourier transform (and its inverse) are *periodic* with period N.

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N)$$

- Although F(u,v) repeats itself infinitely for many values of *u* and *v*, only *N* values of each variable are required to obtain f(x,y) from F(u,v)
  - i.e. Only one period of the transform is necessary to specify F(u,v) in the frequency domain.
  - Similar comments may be made for f(x,y) in the spatial domain

Conjugate symmetry of the Fourier transform

- If *f(x,y)* is real (true for all of our cases), the Fourier transform exhibits *conjugate symmetry*

$$F(u,v)=F^*(-u,-v)$$

or, the more interesting

$$|F(u,v)| = |F(-u,-v)|$$

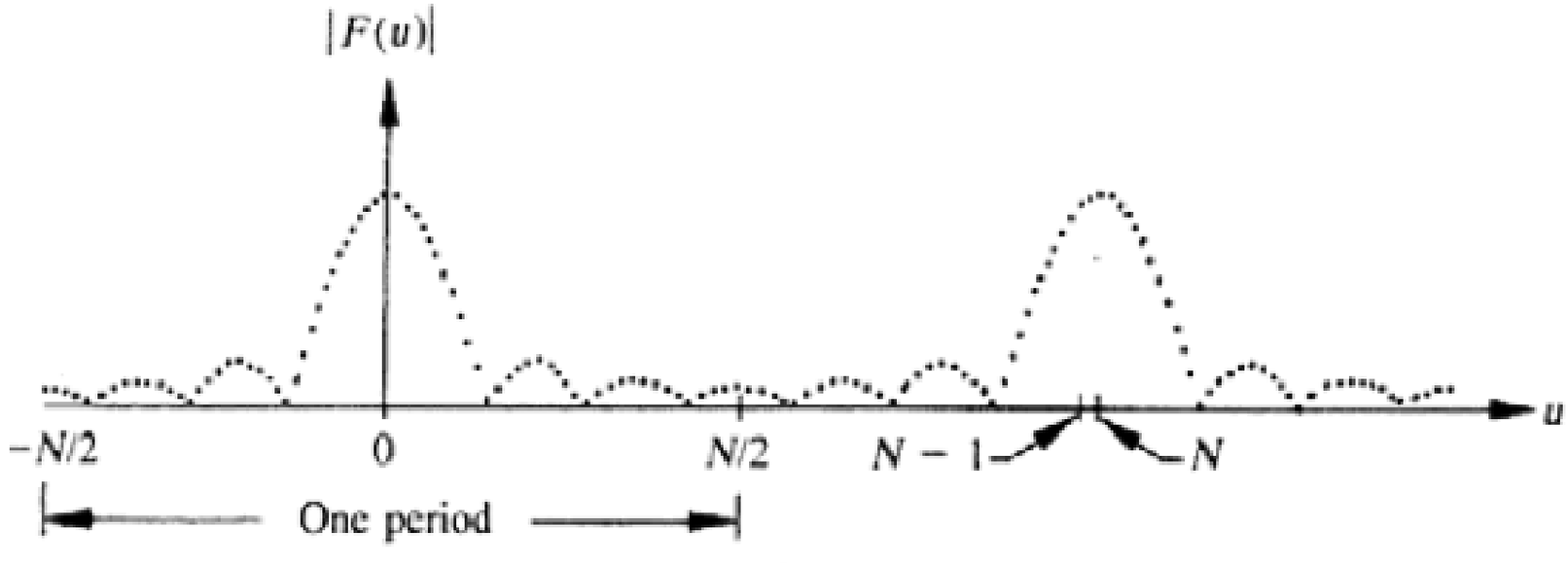
where $F^*(u,v)$ is the complex conjugate of $F(u,v)$

Implications of periodicity & symmetry

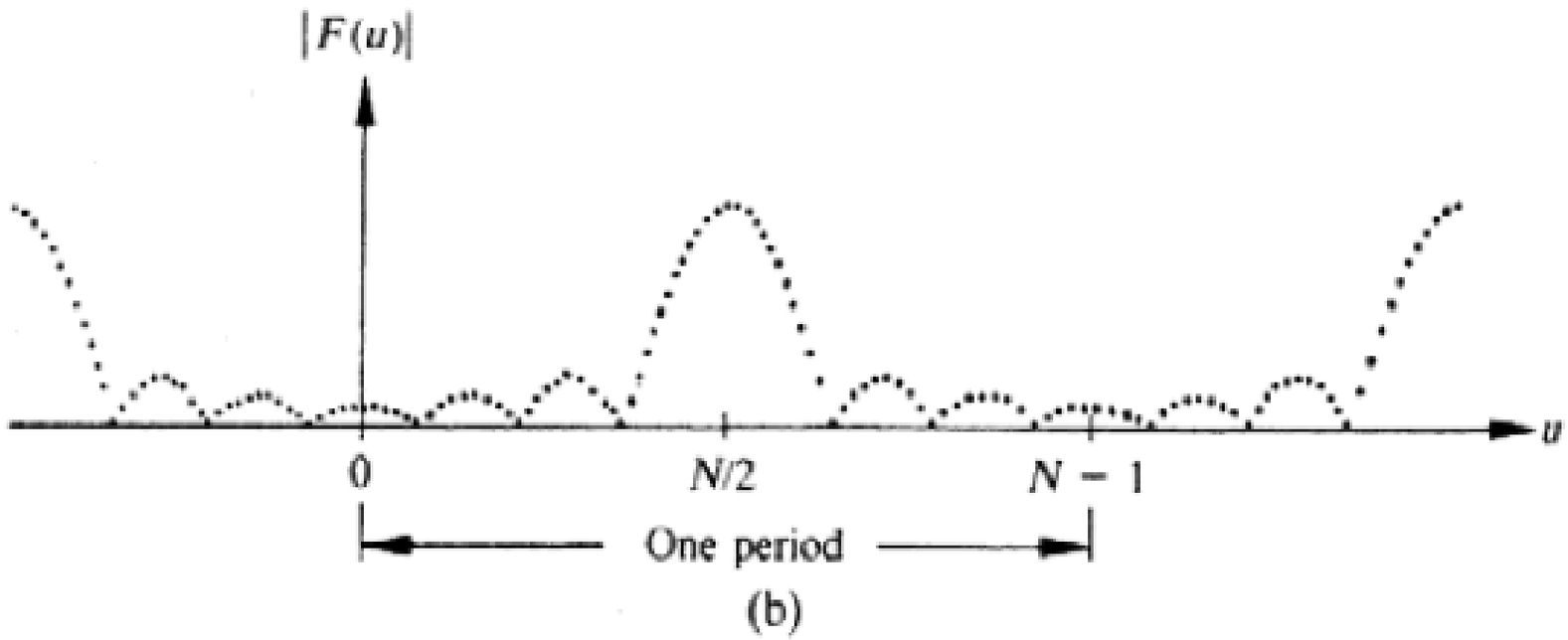
- Consider a 1-D case:
  - F(u) = F(u+N) indicates F(u) has a period of length N
  - |F(u)| = |F(-u)| shows the magnitude is centered about the origin
- Because the Fourier transform is formulated for values in the range from [0,N-1], the result is two back-to-back half periods in this range
- To display one full period in the range, move (shift) the origin of the transform to the point u=N/2

Periodicity properties

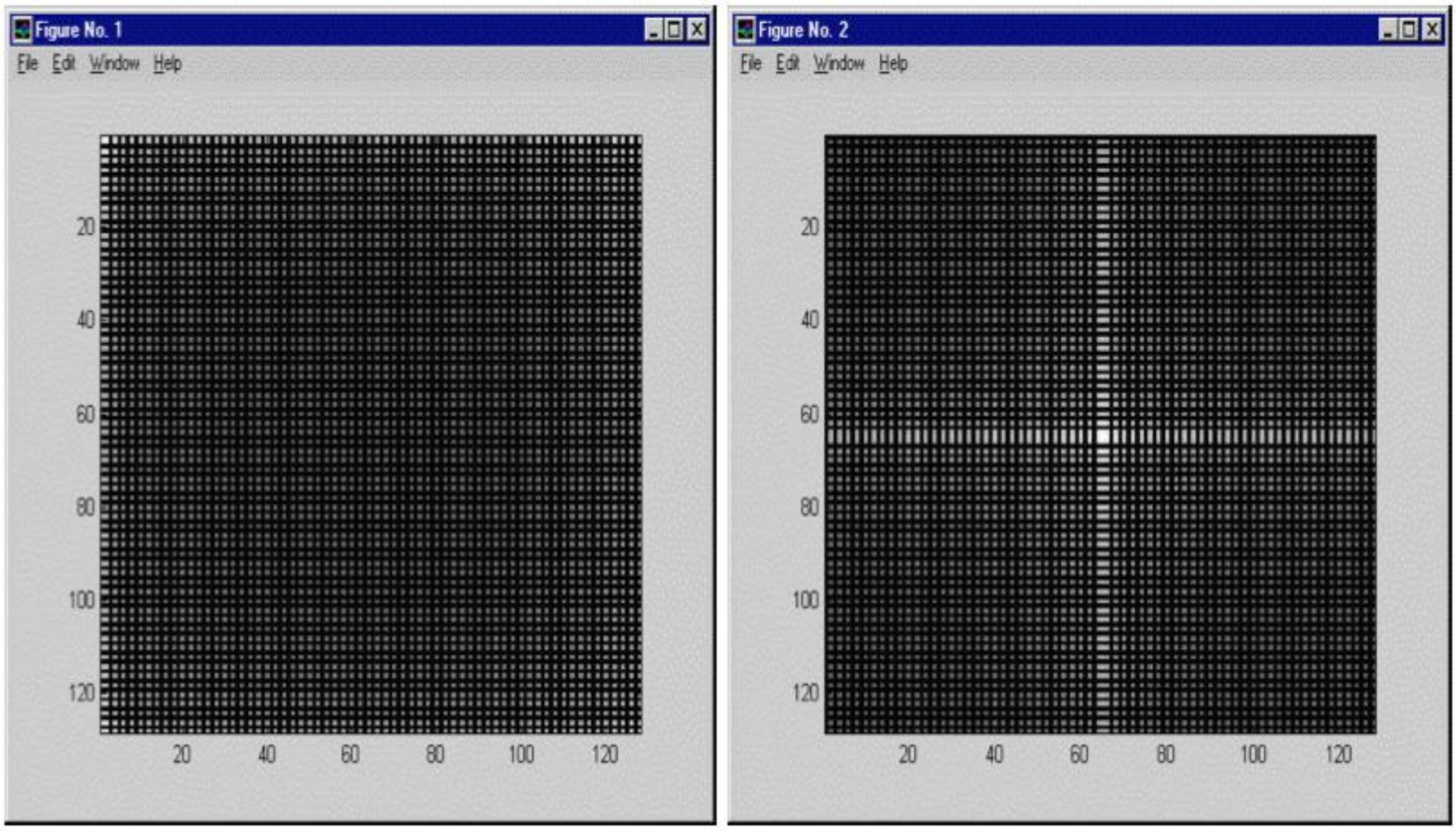
Fourier spectrum with back-to-back half periods in the range [0,n-1]



Shifted spectrum with a full period in the same range



Periodicity properties: 2-D Example



Distributivity & Scaling

- The Fourier transform (and its inverse) are distributive over addition but not over multiplication
- So,

$$\Im\{f_1(x,y)+f_2(x,y)\}=\Im\{f_1(x,y)\}+\Im\{f_2(x,y)\}$$

$$\Im\{f_1(x,y)\times f_2(x,y)\}\neq\Im\{f_1(x,y)\}\times\Im\{f_2(x,y)\}$$

- For two scalars *a* and *b*,

$$af(x,y)\Leftrightarrow aF(u,v)$$

$$f(ax,by)\Leftrightarrow\frac{1}{|ab|}F(u/a,v/b)$$

Average Value

- A widely used expression for the average value of a 2-D discrete function is:

$$\bar{f}(x,y) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$$

- From the definition of F(u,v), for u=v=0,

$$F(0,0) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$$

- Therefore,

$$\bar{f}(x,y) = \frac{1}{N}F(0,0)$$

The Laplacian

- The Laplacian of a two variable function f(x,y) is given as:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

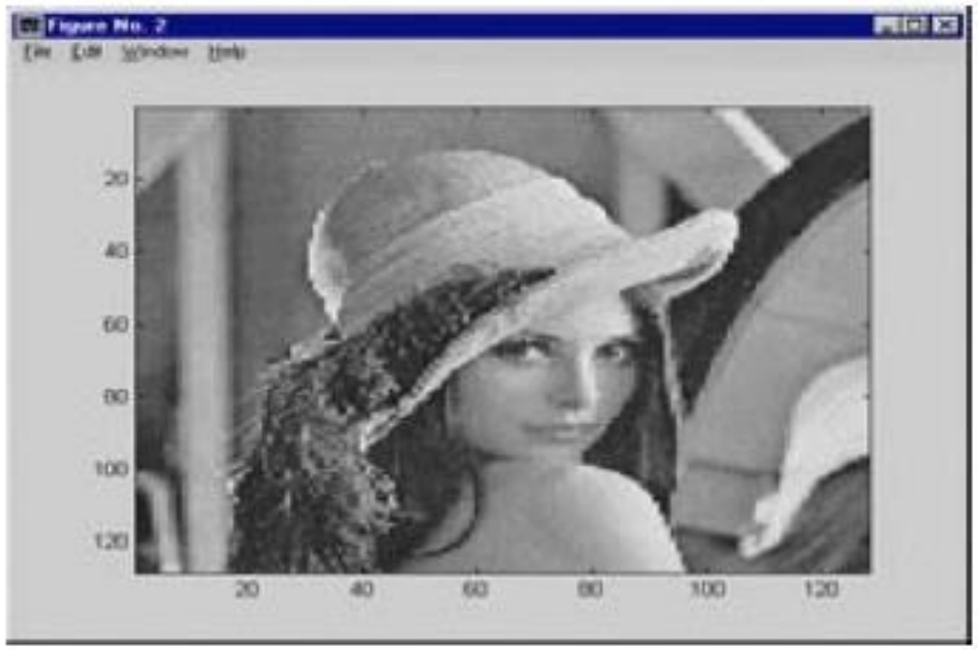
- From the definition of the 2-D Fourier transform,

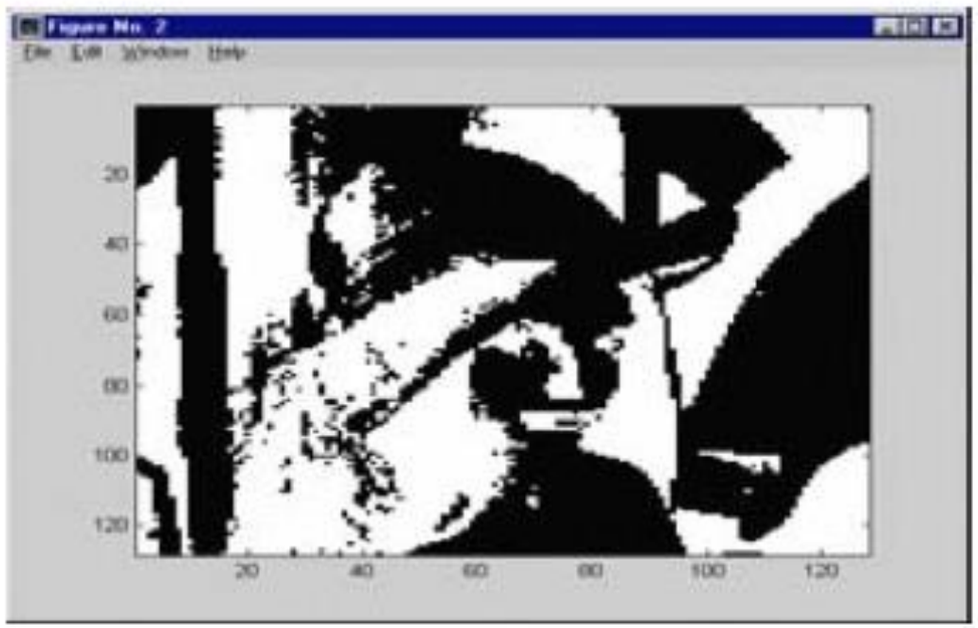
$$\Im\{\nabla^2 f(x,y)\} \Leftrightarrow -(2\pi)^2 (u^2 + v^2) F(u,v)$$

- The Laplacian operator is useful for outlining edges in an image

The Laplacian: Matlab example

```
% Given F(u,v), use the Laplacian
% to construct an edge outlined
% representation of the f(x,y)
[f,fmap]=bmpread('lena128.bmp');
F=fft2(f);
Fedge=zeros(128);
for u=1:128
 for v=1:128
  Fedge(u,v)=-
   (2*pi).^2*(u.^2+v.^2)*F(u,v);
 end
end
fedge=ifft2(Fedge);
image(real(fedge));colormap(gray(256));
```



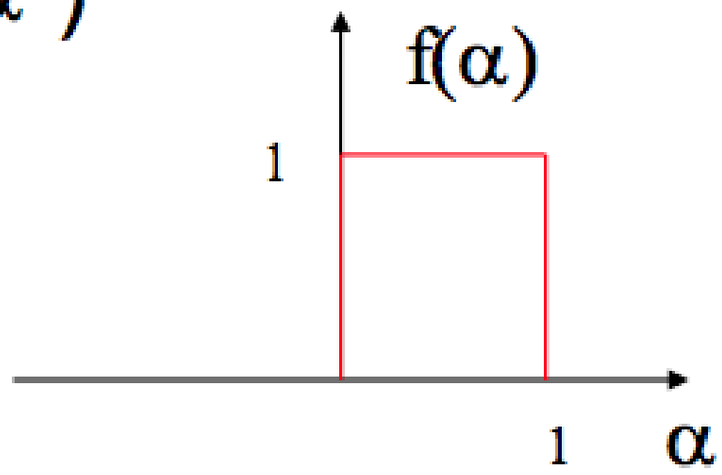


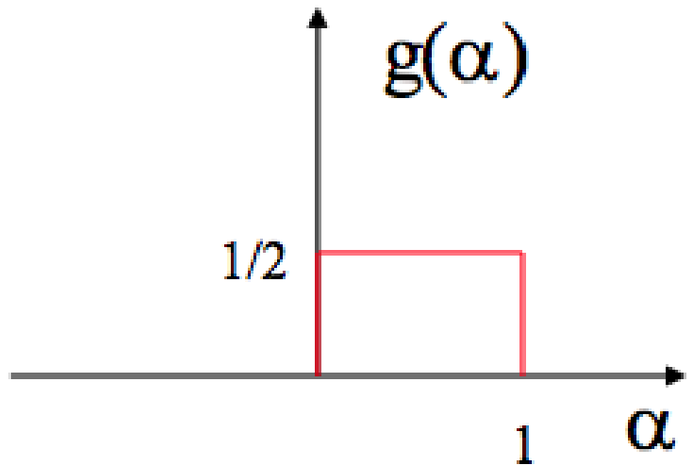
Convolution & Correlation

- The convolution of two functions f(x) and g(x) is denoted f(x)*g(x) and is given by:

$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(\alpha) g(x-\alpha) d\alpha$$

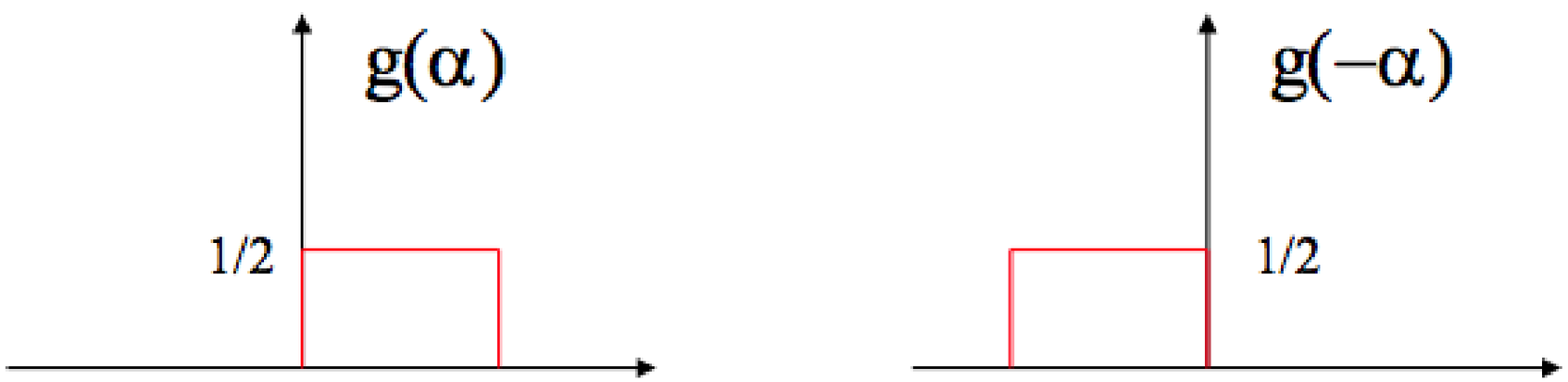
- Where α is a dummy variable of integration.
- Example: Consider the following functions f(α) and g(α)



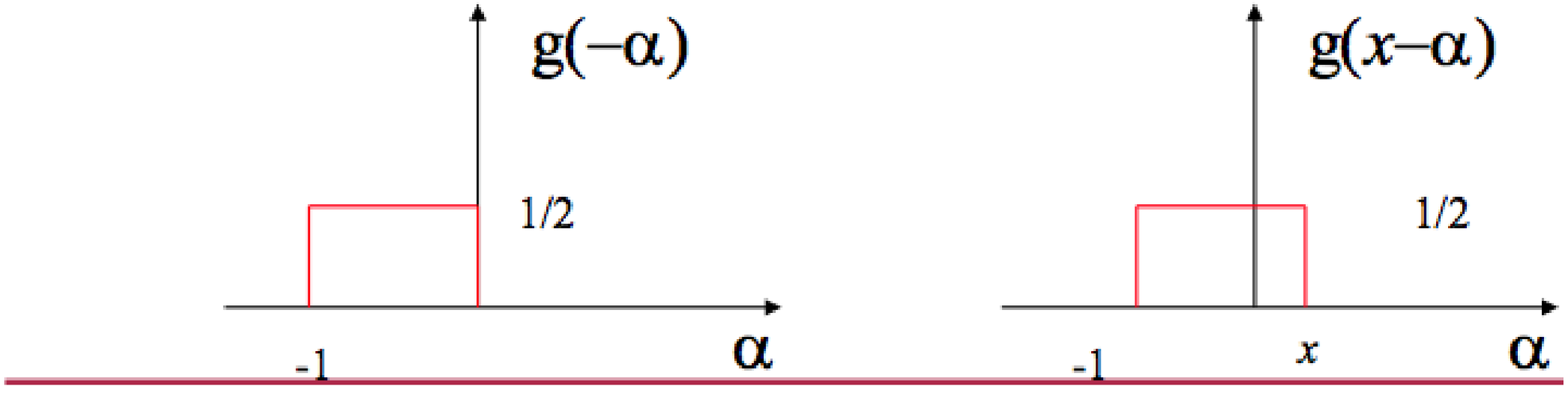


1-D convolution example

- Compute g(-α) by folding g(α) about the origin

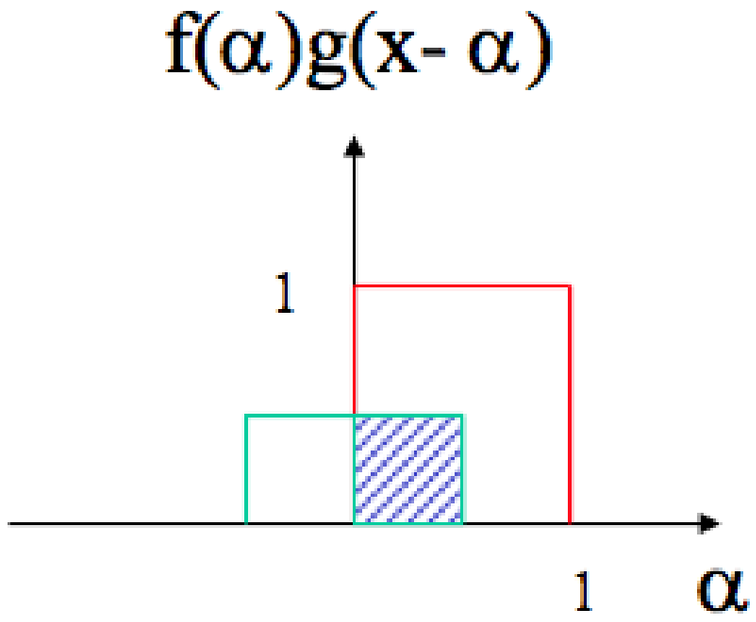


- Compute g(x-α) by displacing g(-α) by the value x

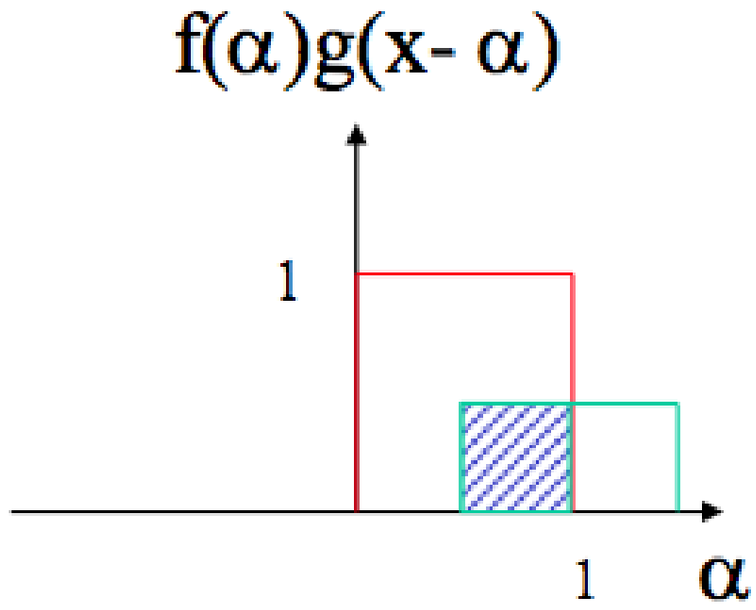


1-D convolution example (continued)

- Then, for any value x, we multiply $g(x-\alpha)$ and $f(\alpha)$ and integrate from $-\infty$ to $+\infty$
- For $0 \leq x \leq 1$ we have



For $1 \leq x \leq 2$ we have

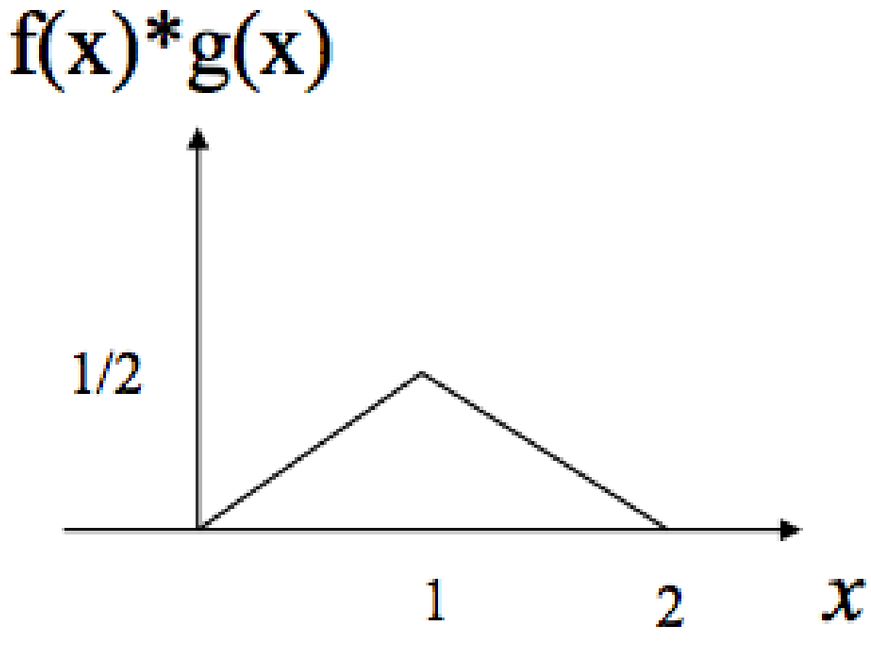


1-D convolution example (continued)

- Thus we have

$$f(x)*g(x)=\begin{cases} x/2 & 0\le x\le 1 \\ 1-x/2 & 1\le x\le 2 \\ 0 & \textit{elsewhere.} \end{cases}$$

- Graphically,



Convolution and impulse functions

- Of particular interest will be the convolution of a function *f(x)* with an impulse function $\delta(x-x_0)$

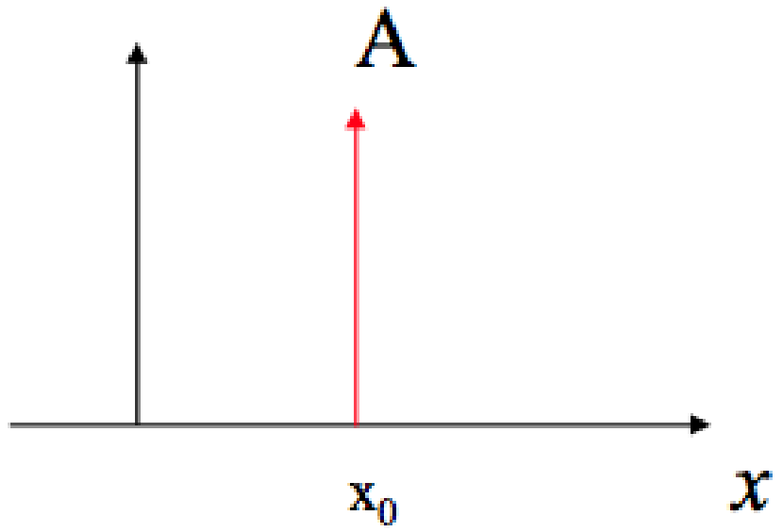
$$\int_{-\infty}^{+\infty} f(x)\delta(x-x_0)dx = f(x_0)$$

- The function $\delta(x-x_0)$ may be viewed as having an area of unity in an infinitesimal neighborhood around x_0 and 0 elsewhere. That is

$$\int_{-\infty}^{+\infty} \delta(x-x_0)dx = \int_{x_0^-}^{x_0^+} \delta(x-x_0)dx = 1$$

Convolution and impulse functions (continued)

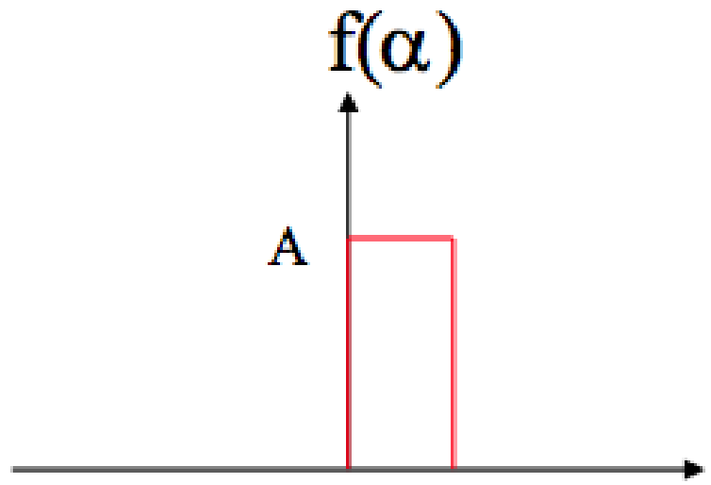
- We usually say that $\delta(x-x_0)$ is located at $x=x_0$ and the strength of the impulse is given by the value of f(x) at $x=x_0$
- If f(x)=A then, $A\delta(x-x_0)$ is impulse of strength A at $x=x_0$.
- Graphically this is:



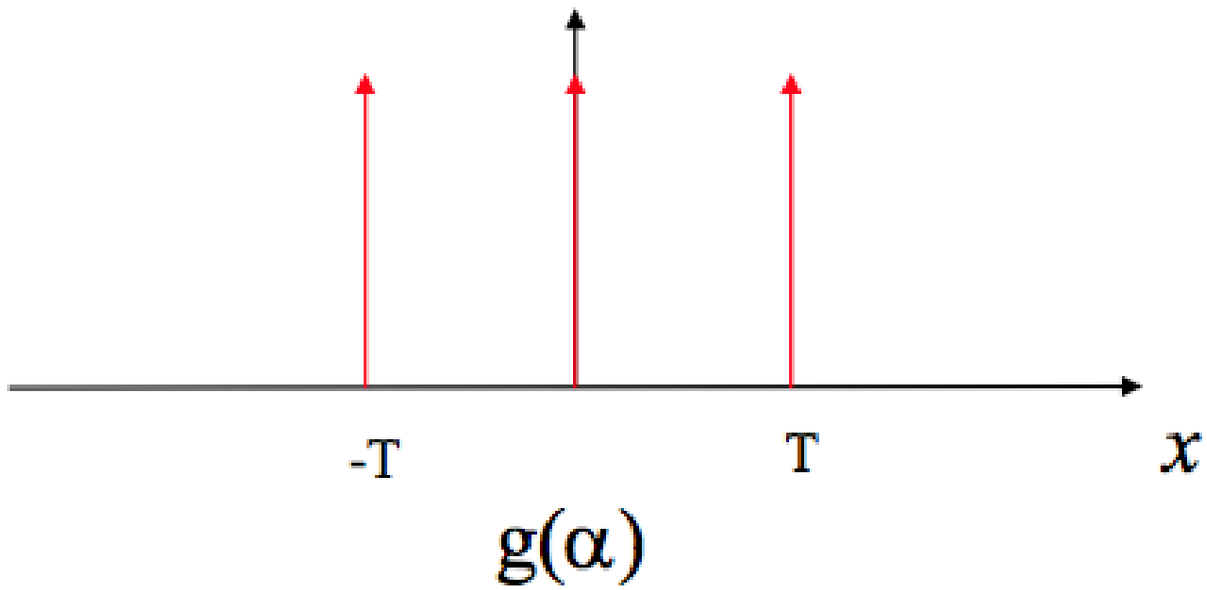
$A\delta(x-x_0)$

Convolution with an impulse function

- Given f(x) is

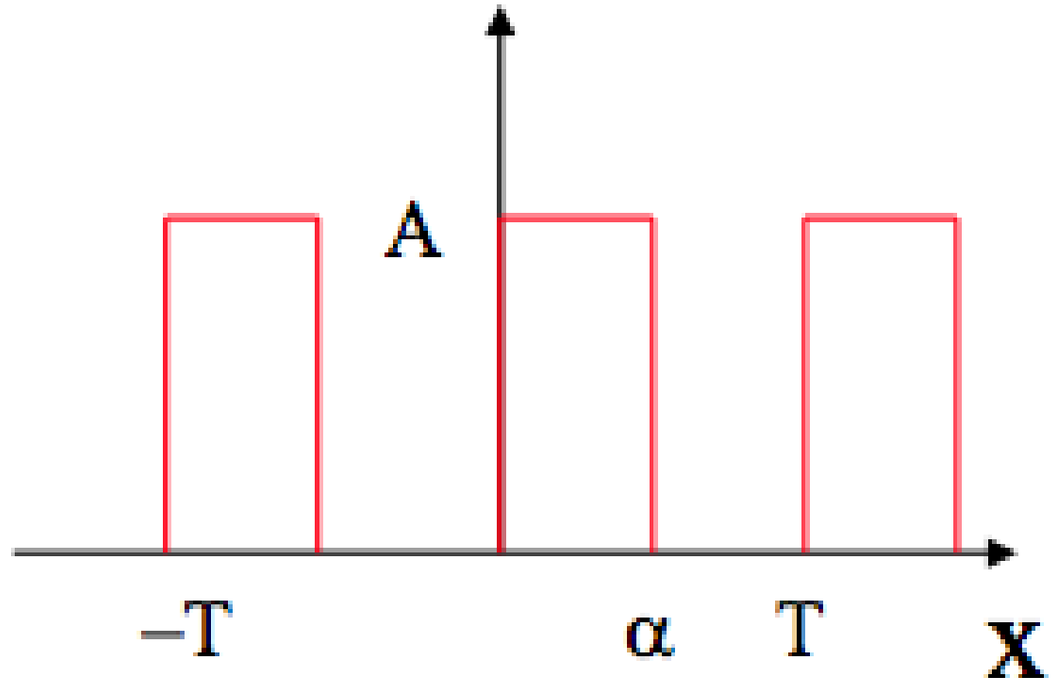


- and g(x)=δ(x+T)+ δ(x)+ δ(x-T)



Convolution with an impulse function (continued)

- f(x)*g(x) is



Convolution and the Fourier transform

- f(x)*g(x) and F(u)G(u) form a Fourier transform pair
- If f(x) has transform F(u) and g(x) has transform G(u) then f(x)*g(x) has transform F(u)G(u)

$$f(x) * g(x) \Leftrightarrow F(u)G(u)$$

$$f(x)g(x) \Leftrightarrow F(u) * G(u)$$

- These two results are commonly referred to as the *convolution theorem*

Frequency domain filtering

- Enhancement in the frequency domain is straightforward
  - Compute the Fourier transform
  - Multiply the result by a filter transform function
  - Take the inverse transform to produce the enhanced image
- In practice, small spatial masks are used considerably more than the Fourier transform because of their simplicity of implementation and speed of operation
- However, some problems are not easily addressable by spatial techniques
  - Such as homomorphic filtering and some image restoration techniques

Lowpass frequency domain filtering

- Given the following relationship

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

- where F(u,v) is the Fourier transform of an image to be smoothed
- The problem is to select an H(u,v) that yields an appropriate G(u,v)
- We will consider *zero-phase-shift* filters that do not alter the phase of the transform (i.e. they affect the real and imaginary parts of F(u,v) in exactly the same manner)

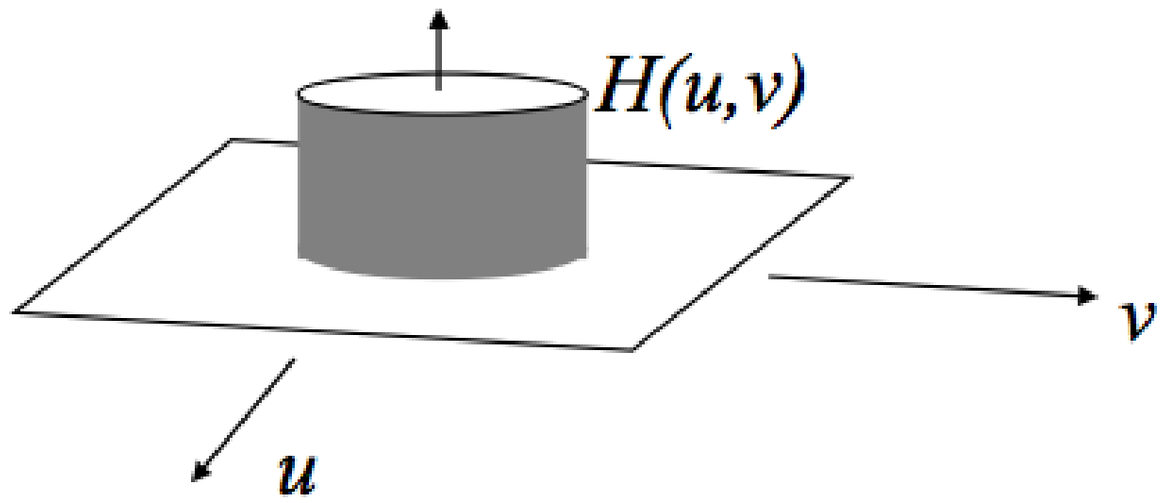
Ideal lowpass filter (ILPF)

- A transfer function for a 2-D ideal lowpass filter (ILPF) is given as

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

- where D_0 is a stated nonnegative quantity (the cutoff frequency) and D(u,v) is the distance from the point (u,v) to the center of the frequency plane

$$D(u,v) = \sqrt{u^2 + v^2}$$



Ideal lowpass filter (ILPF) (continued)

- The point D_0 traces a circle from the frequency origin giving a locus of cutoff frequencies (all are at distance D_0 from the origin)
- One way to establish a set of "standard" loci is to compute circles that encompass various amounts of the total signal power P_T
- P_T is given by

$$P_T = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} P(u,v)$$

- where P(u,v) is given as

$$P(u,v) = \left|F(u,v)\right|^2 = R^2(u,v) + I^2(u,v)$$

- For the centered transform, a circle of radius *r* encompasses β percent of the power, where

$$\beta = 100\left[\sum_u \sum_v P(u,v) / P_T\right] \text{ (the summation is over all points (u, v) encompassed by the circle)}$$

The Discrete Fourier Transform (DFT)

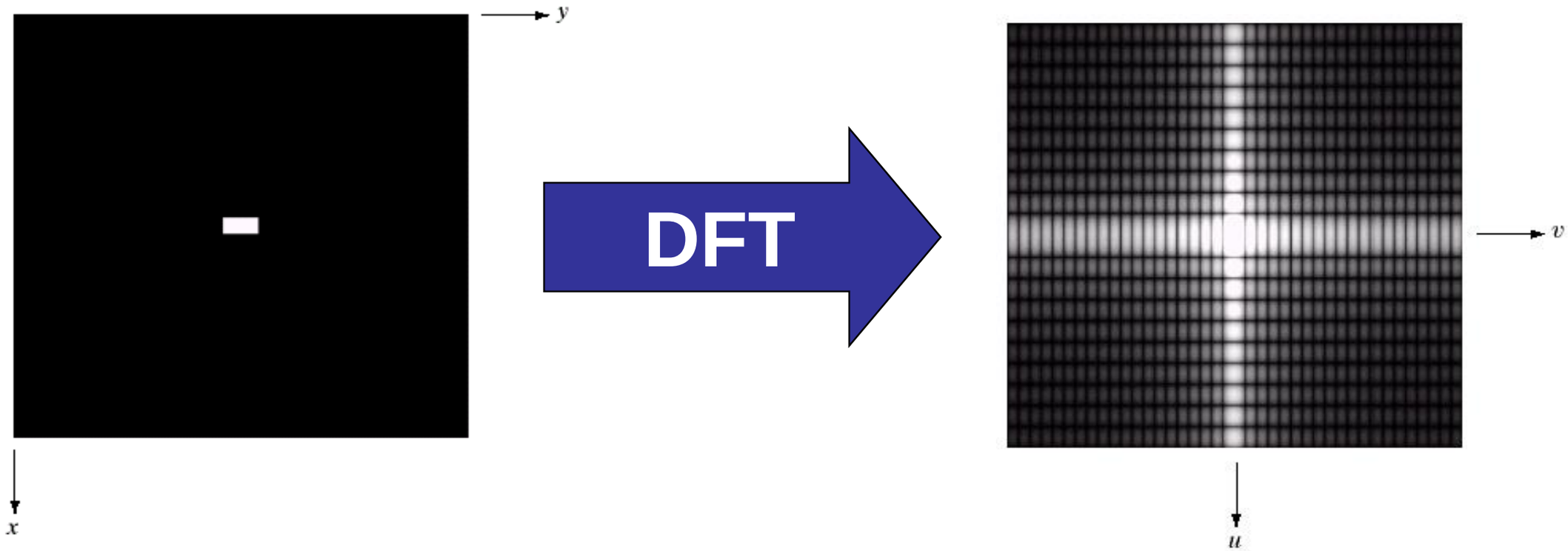
The *Discrete Fourier Transform* of $f(x, y)$, for x = 0, 1, 2…M-1 and y = 0,1,2…N-1, denoted by $F(u, v)$, is given by the equation:

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

for u = 0, 1, 2…M-1 and v = 0, 1, 2…N-1.

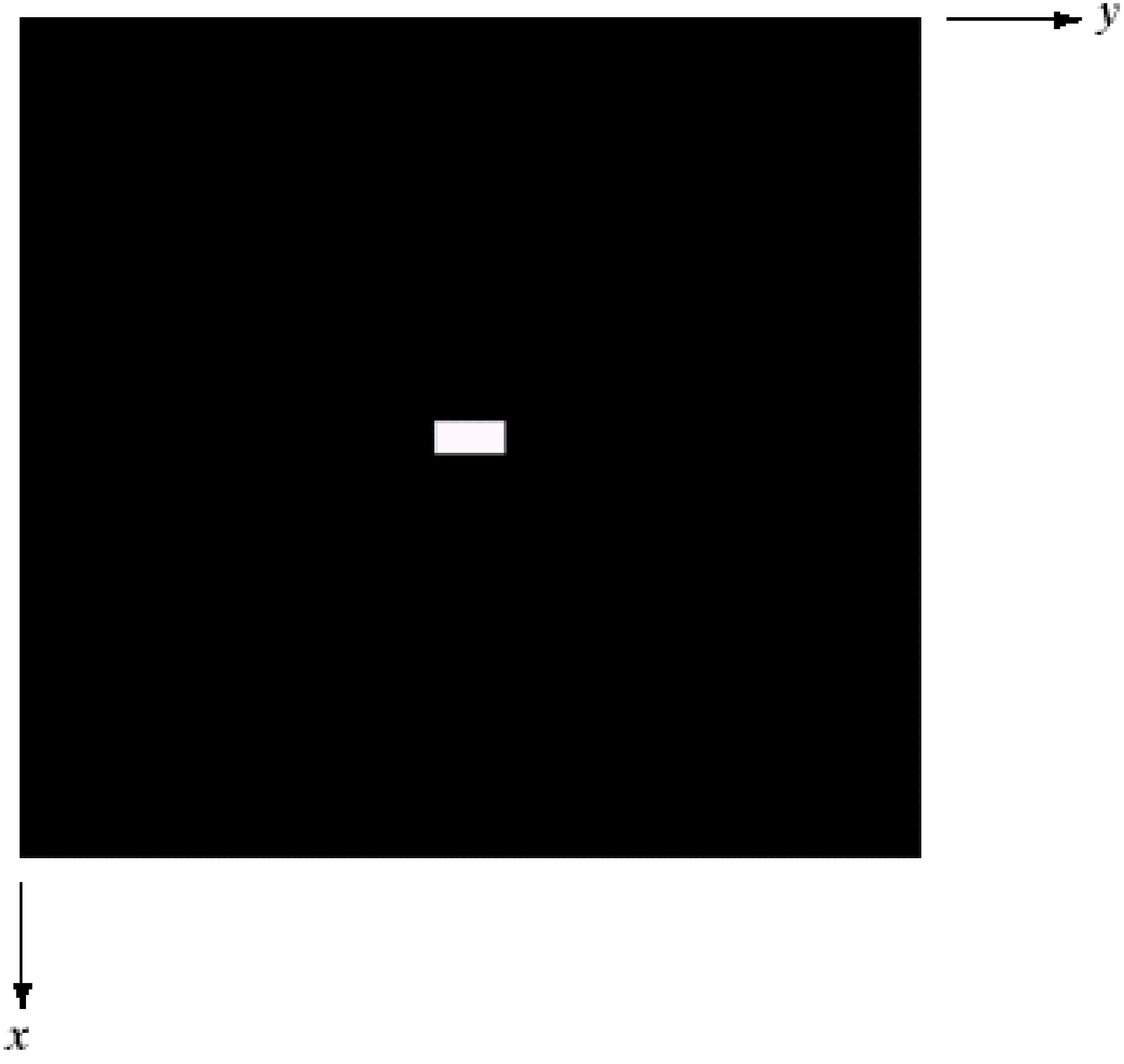
DFT & Images

The DFT of a two dimensional image can be visualised by showing the spectrum of the images component frequencies

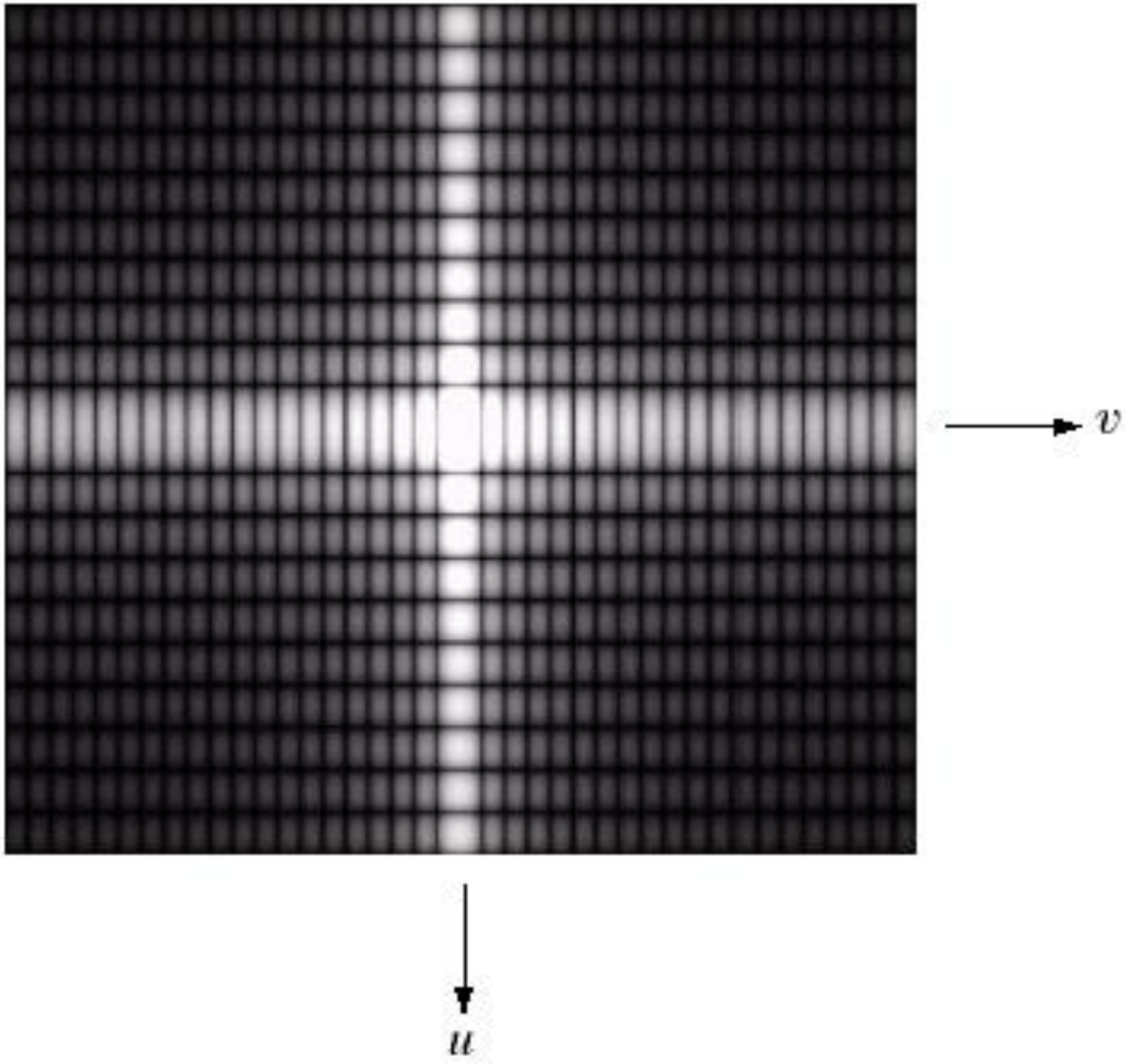


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

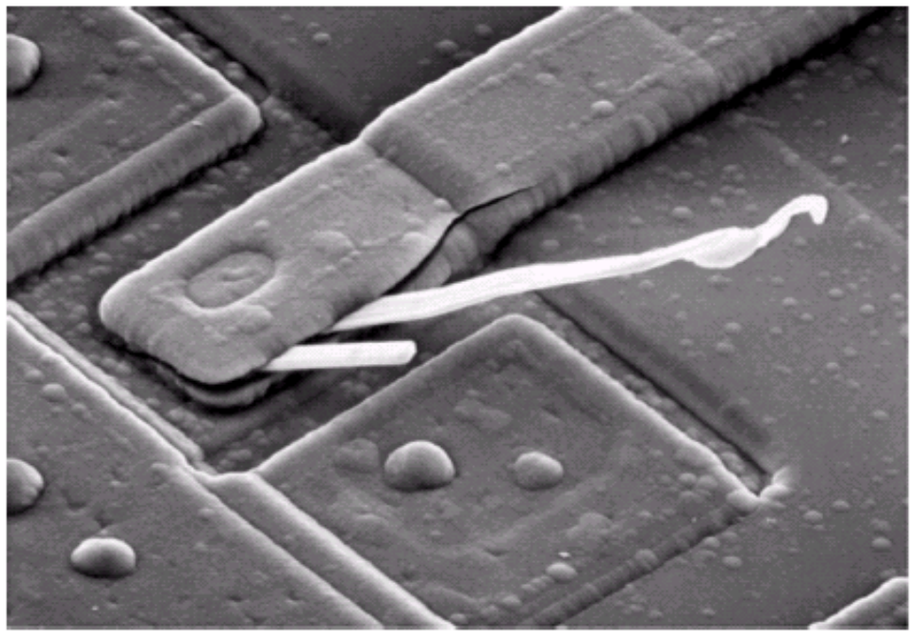
DFT & Images



DFT & Images

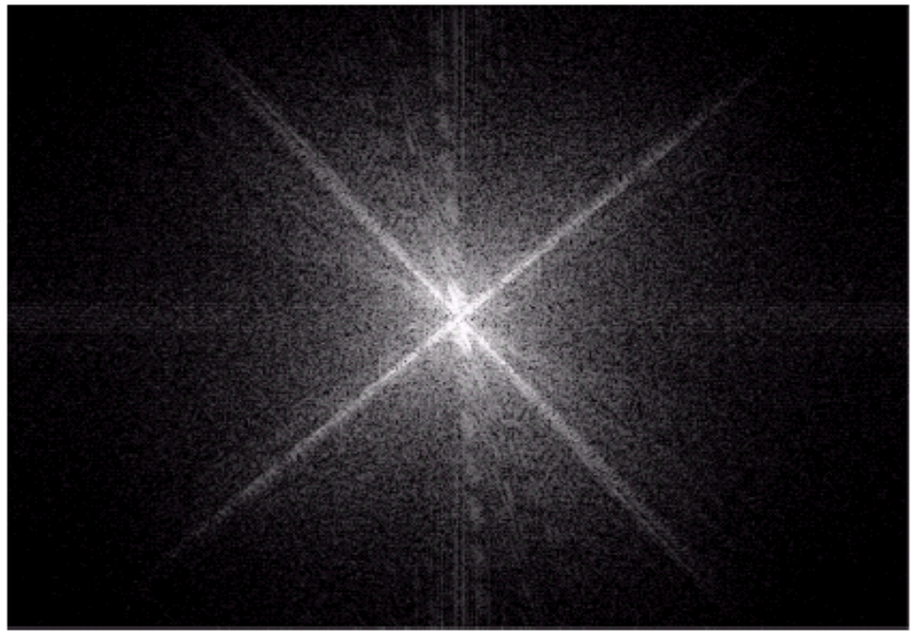


DFT & Images (cont…)



Scanning electron microscope image of an integrated circuit magnified ~2500 times

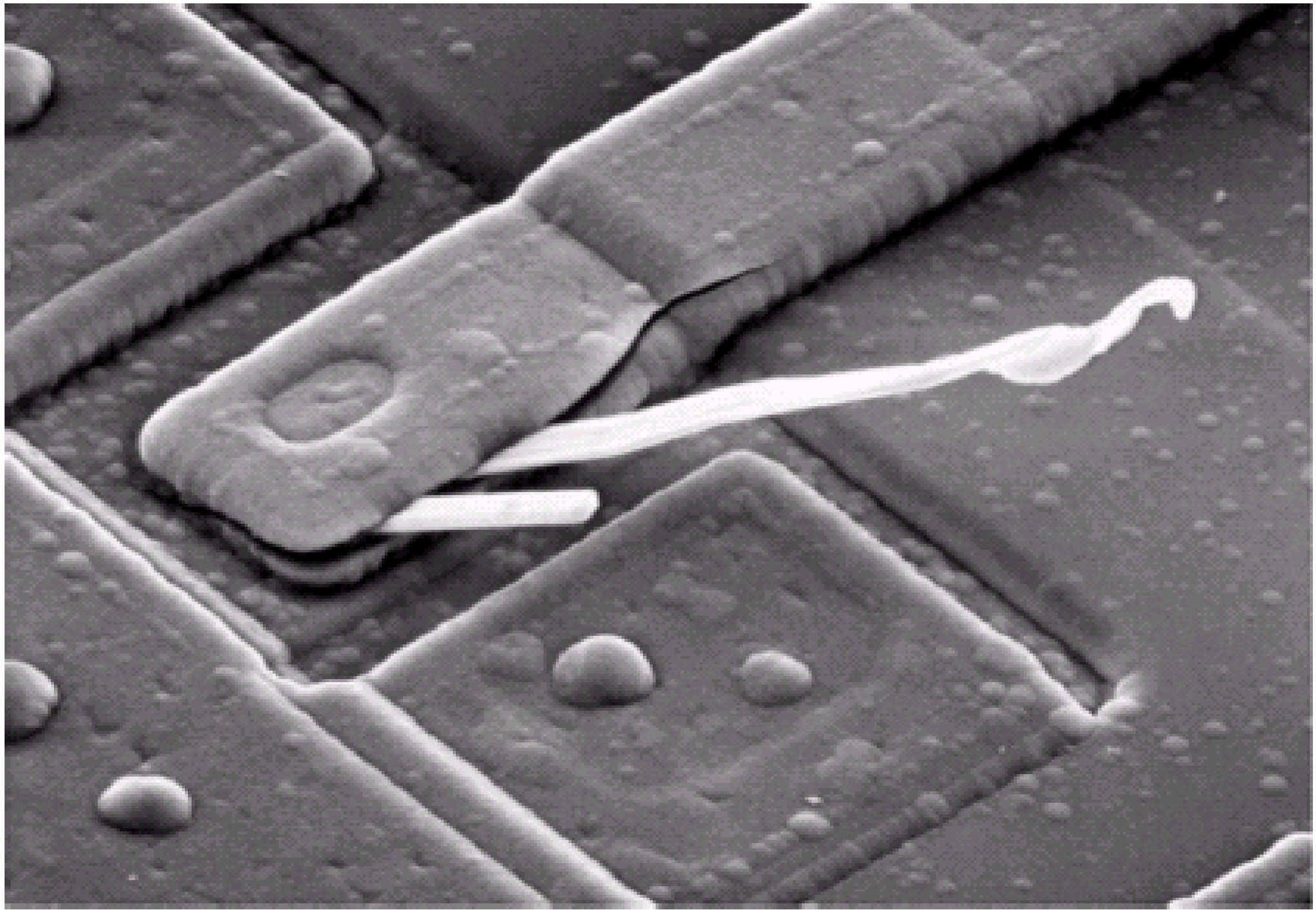
DFT



Fourier spectrum of the image

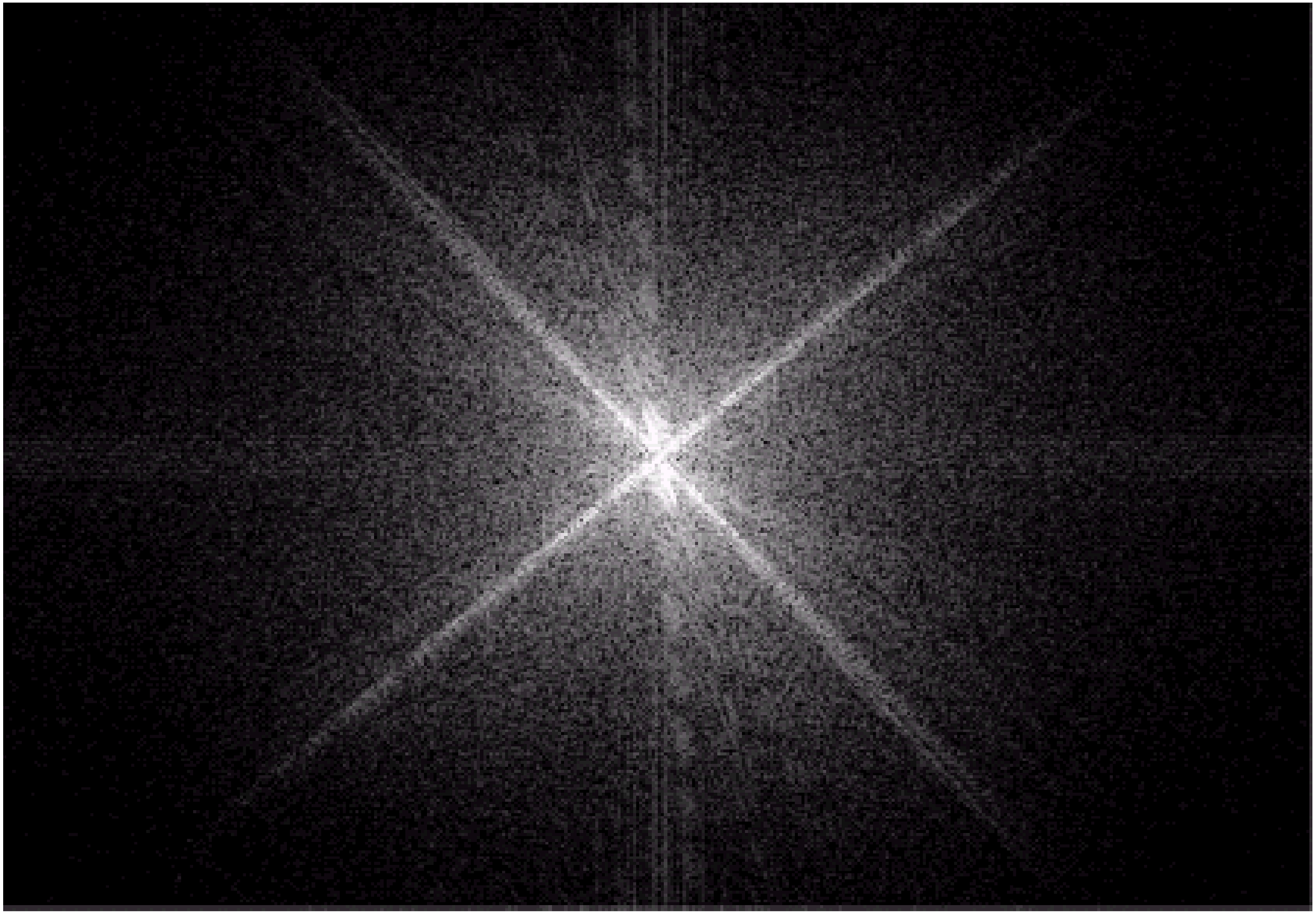
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

DFT & Images (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

DFT & Images (cont…)



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

The Inverse DFT

It is really important to note that the Fourier transform is completely **reversible**

The inverse DFT is given by:

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

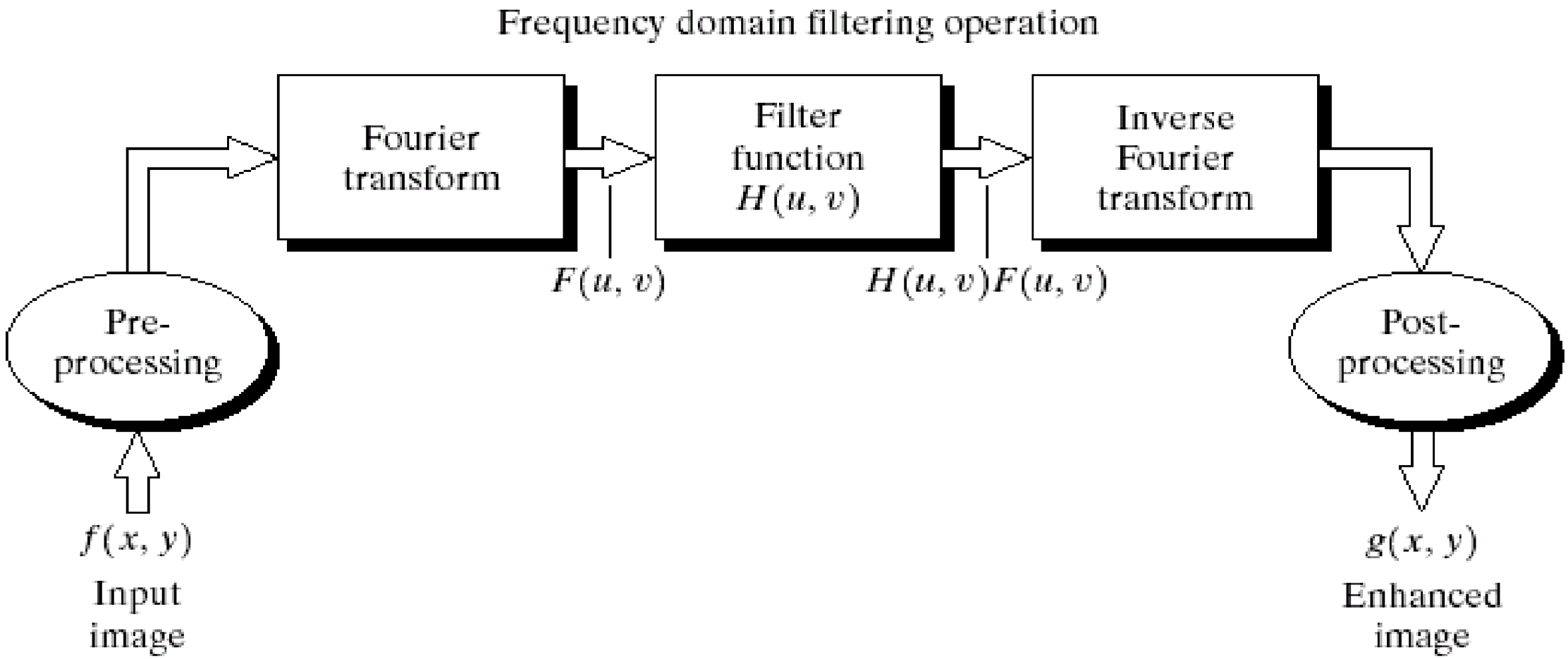
for x = 0, 1, 2…M-1 and y = 0, 1, 2…N-1

The DFT and Image Processing

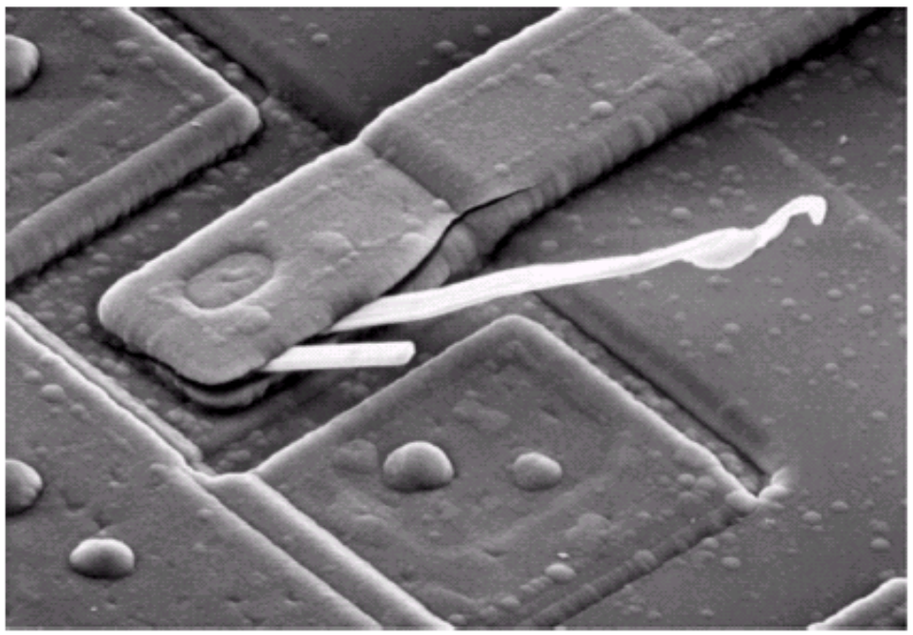
To filter an image in the frequency domain:

1. Multiply the input image by $(-1)^{x+y}$ to center the transform.
2. Compute $F(u,v)$ the DFT of the image
3. Multiply $F(u,v)$ by a filter function $H(u,v)$
4. Compute the inverse DFT of the result in (3)
5. Obtain the real part of the result in (4)
6. Multiply the result in (5) by $(-1)^{x+y}$

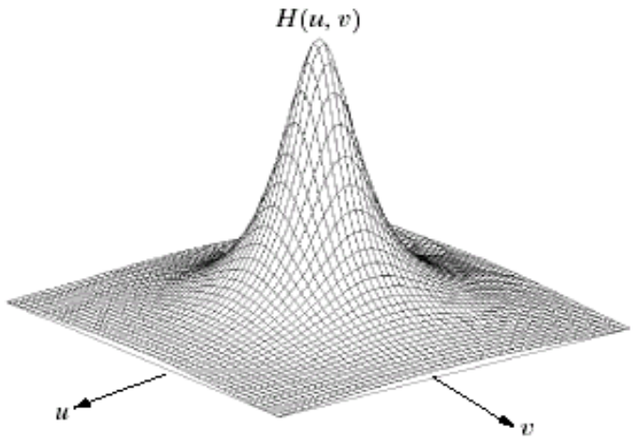
The DFT and Image Processing

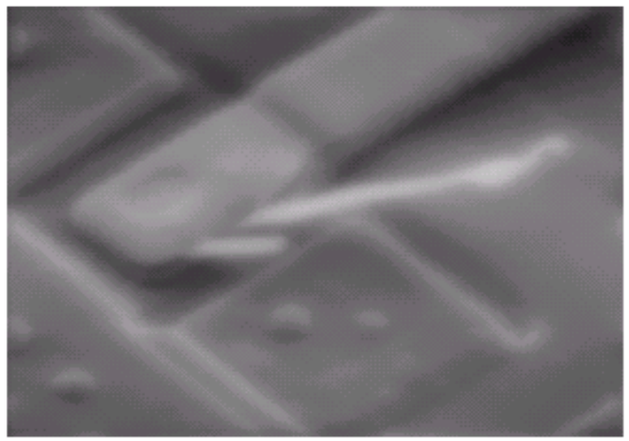


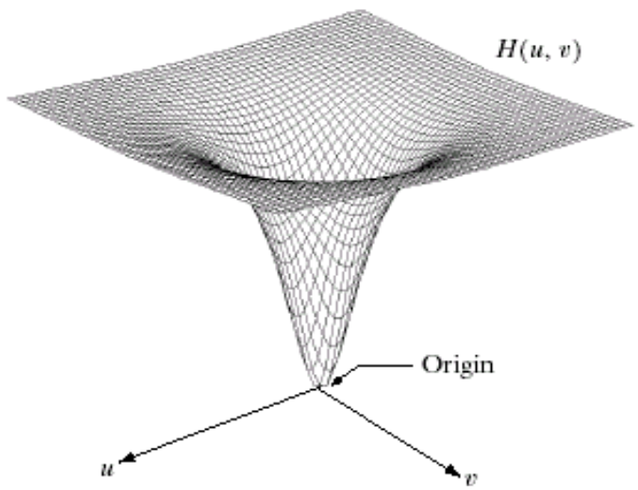
Some Basic Frequency Domain Filters

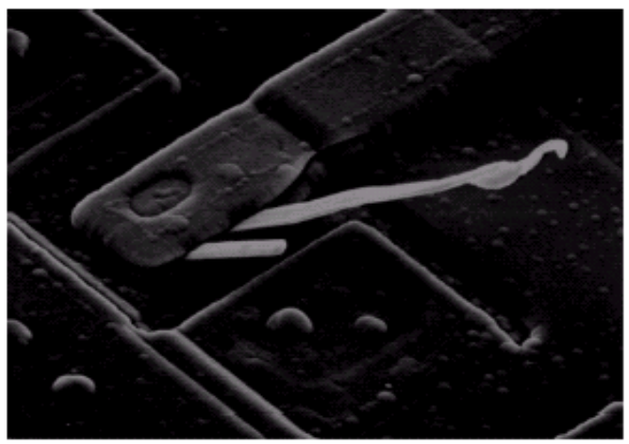


Low Pass Filter



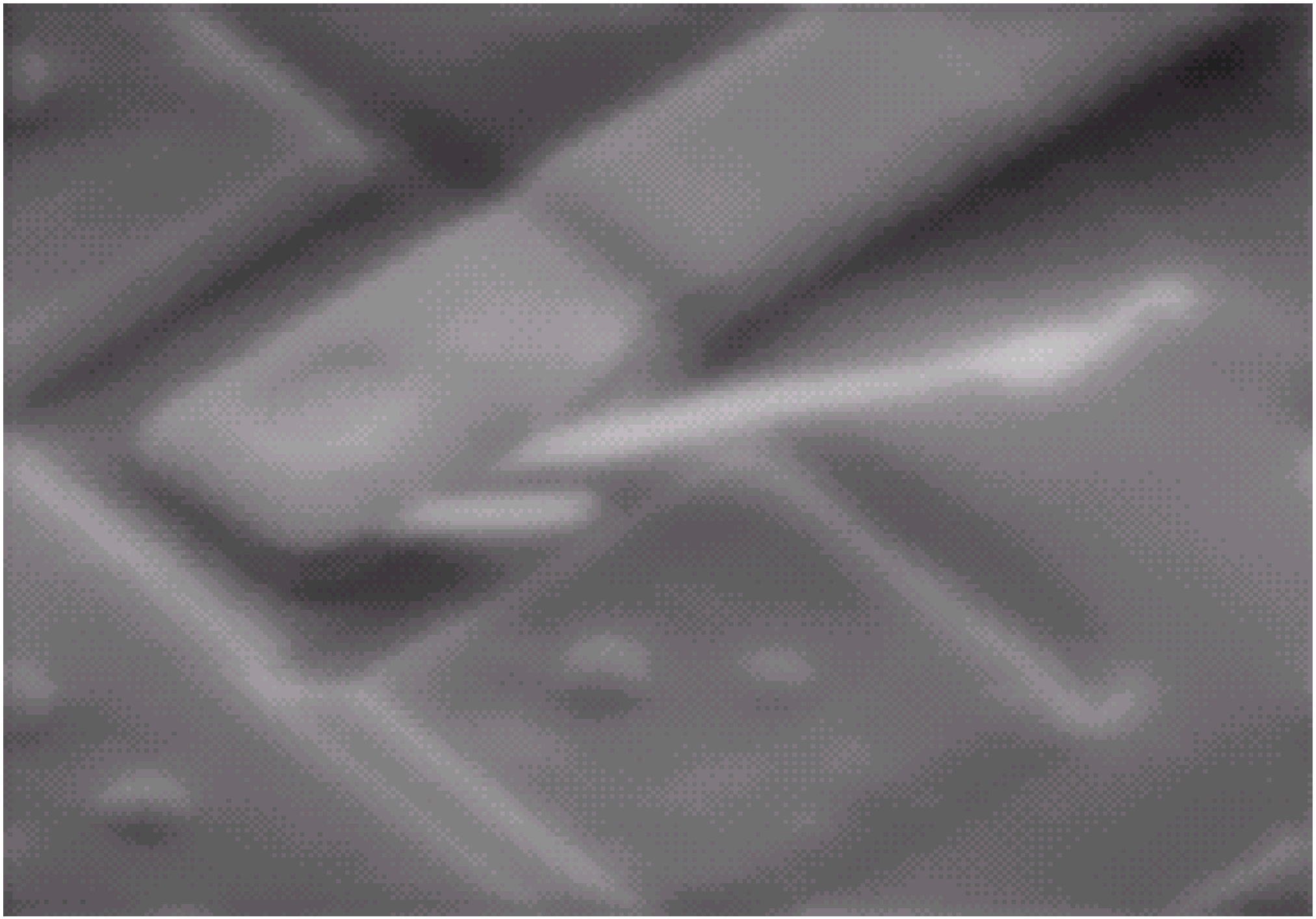




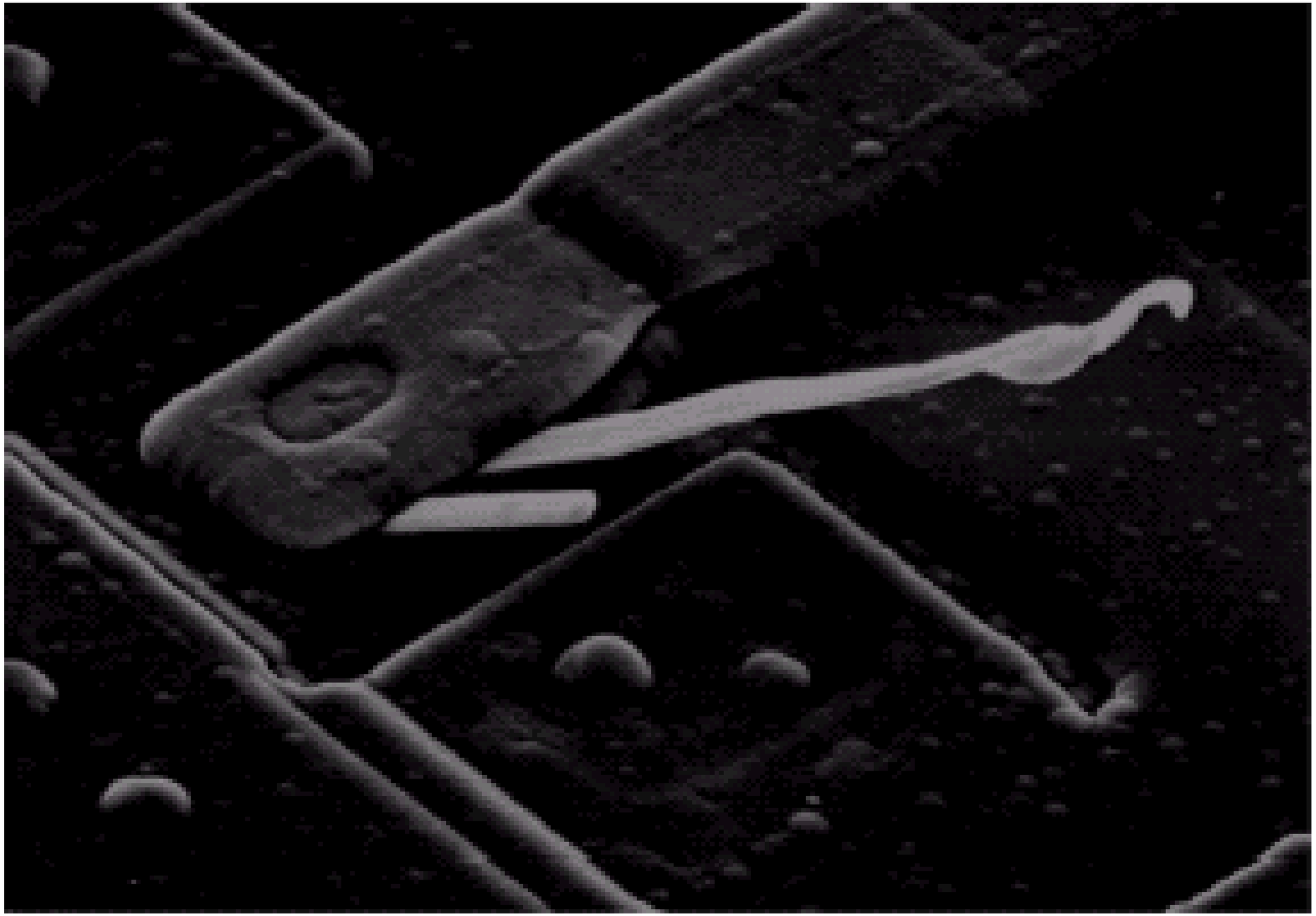


High Pass Filter

Some Basic Frequency Domain Filters



Some Basic Frequency Domain Filters



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Smoothing Frequency Domain Filters

Smoothing is achieved in the frequency domain by dropping out the high frequency components

The basic model for filtering is:

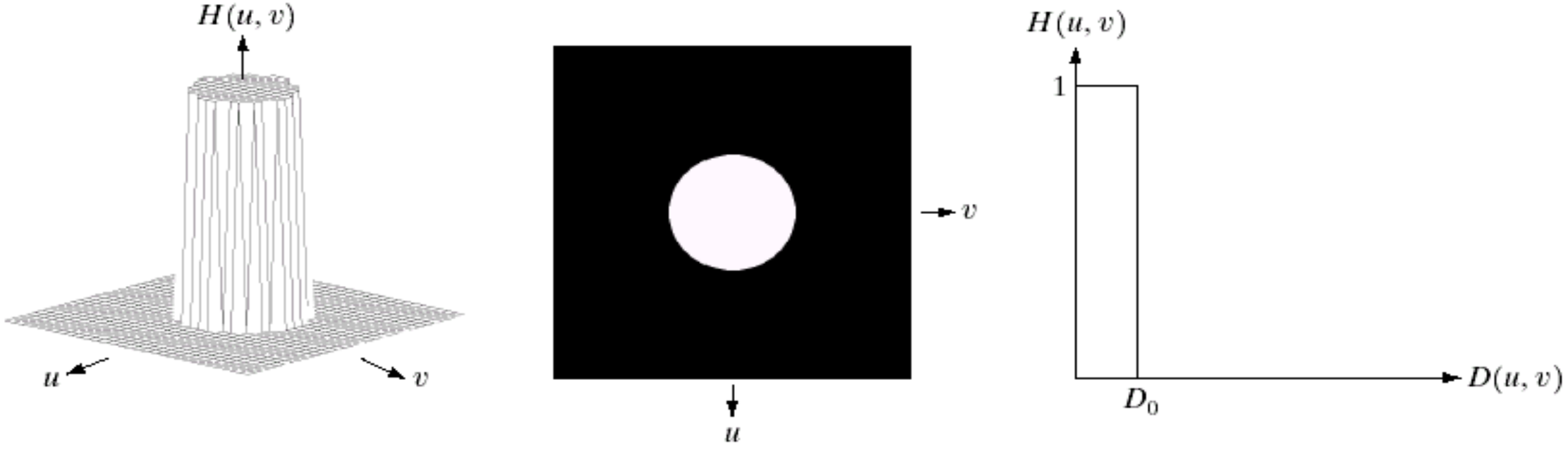
$$G(u,v) = H(u,v)F(u,v)$$

where $F(u,v)$ is the Fourier transform of the image being filtered and $H(u,v)$ is the filter transform function

Low pass filters – only pass the low frequencies, drop the high ones

Ideal Low Pass Filter

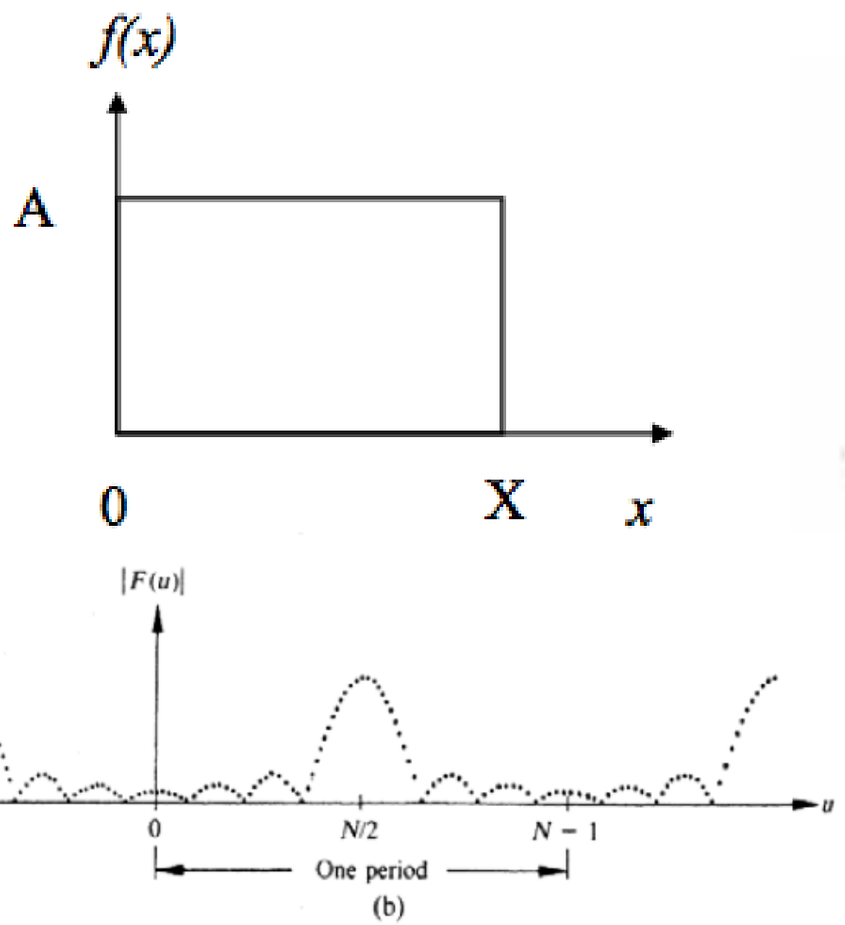
Simply cut off all high frequency components that are a specified distance D_0 from the origin of the transform

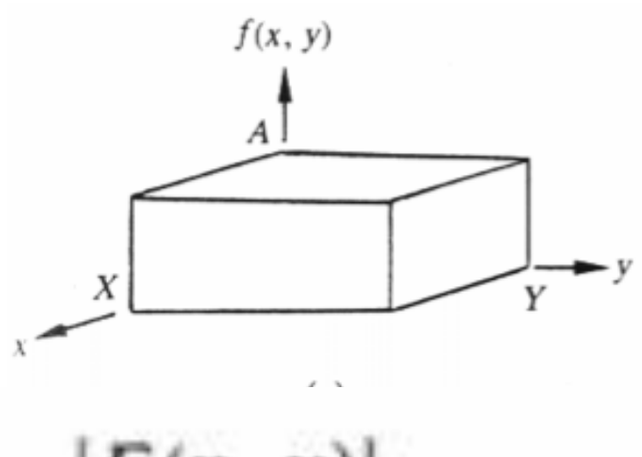


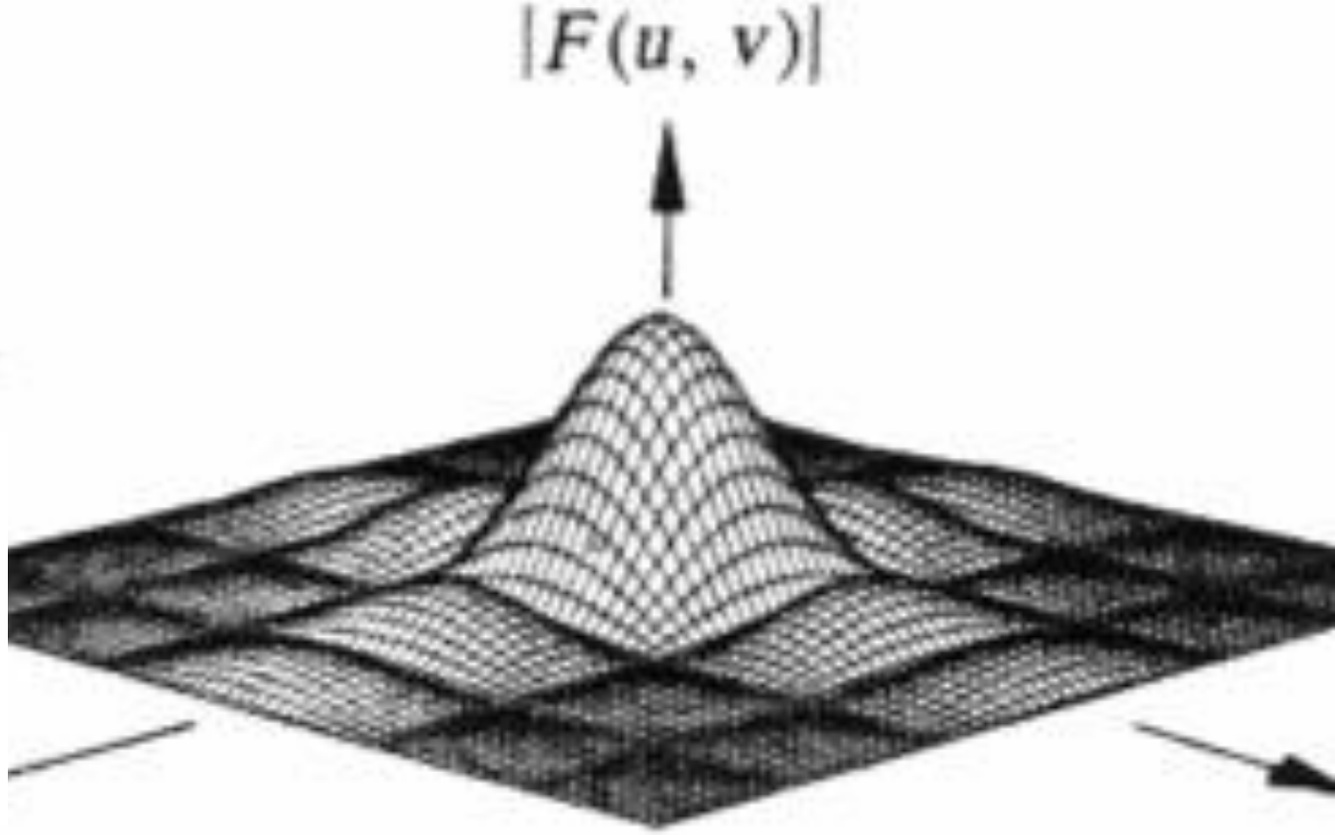
changing the distance changes the behaviour of the filter

Recall Fourier Transform results

- Low frequency components lies near center
- High frequency components lies far from center







Ideal Low Pass Filter (cont…)

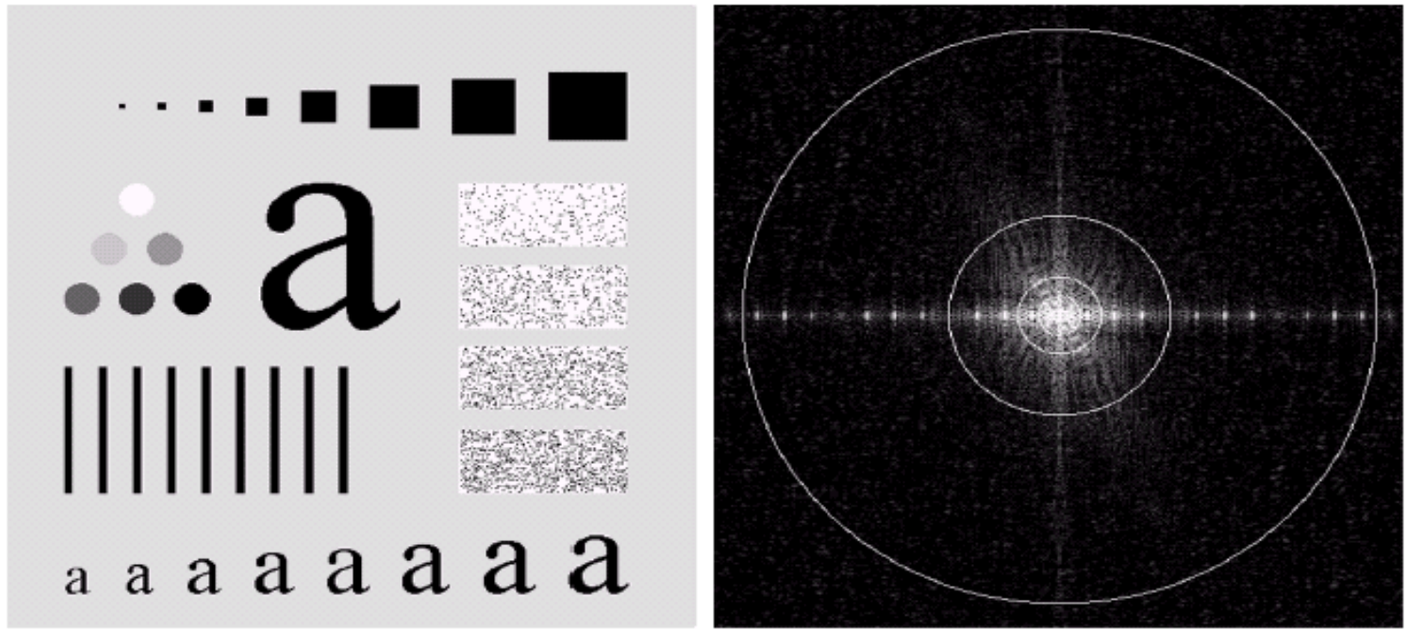
The transfer function for the ideal low pass filter can be given as:

$$H(u,v) = \begin{cases} 1 \text{ if } D(u,v) \le D_0 \\ 0 \text{ if } D(u,v) > D_0 \end{cases}$$

where $D(u,v)$ is given as:

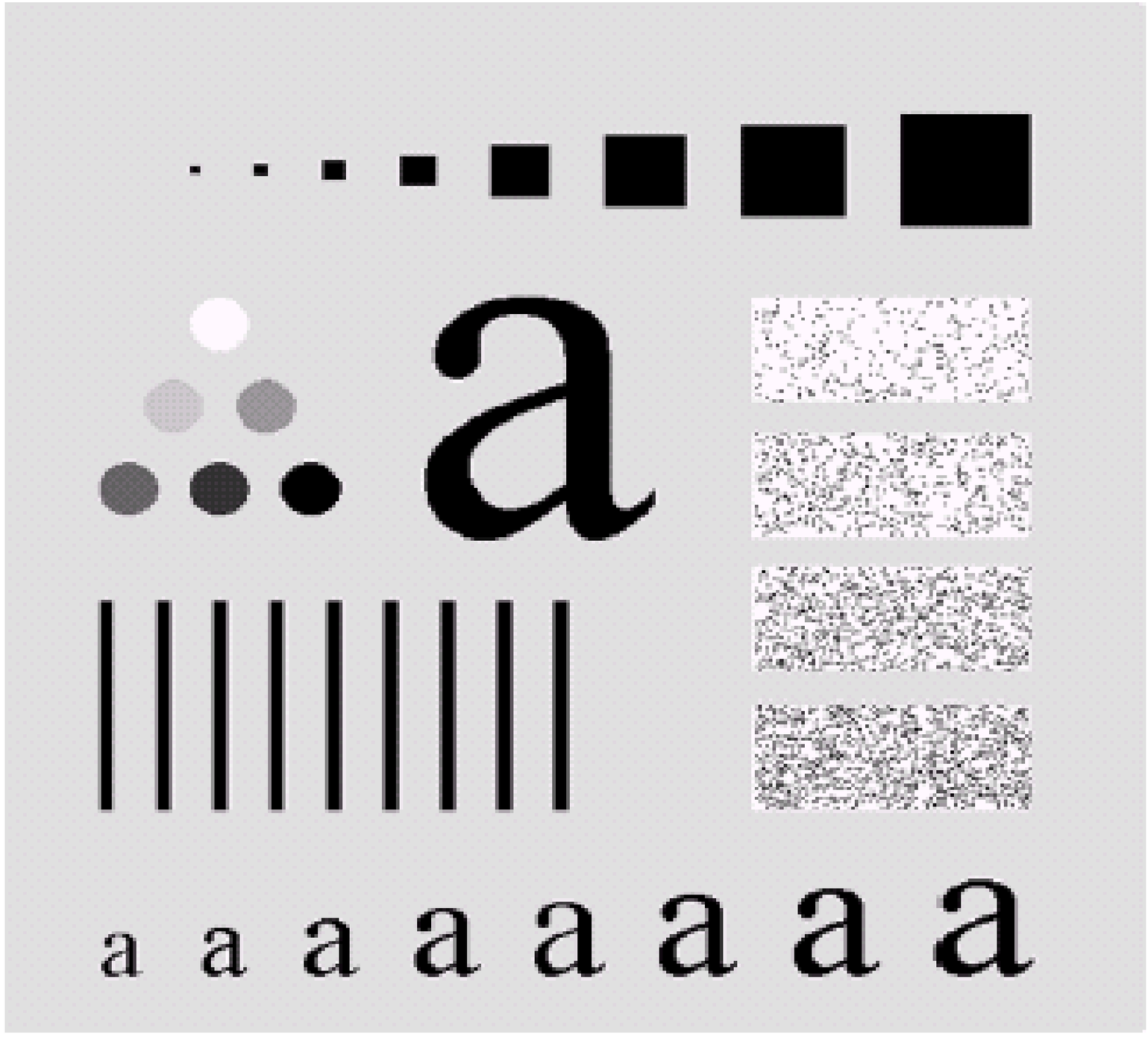
$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

Ideal Low Pass Filter (cont…)

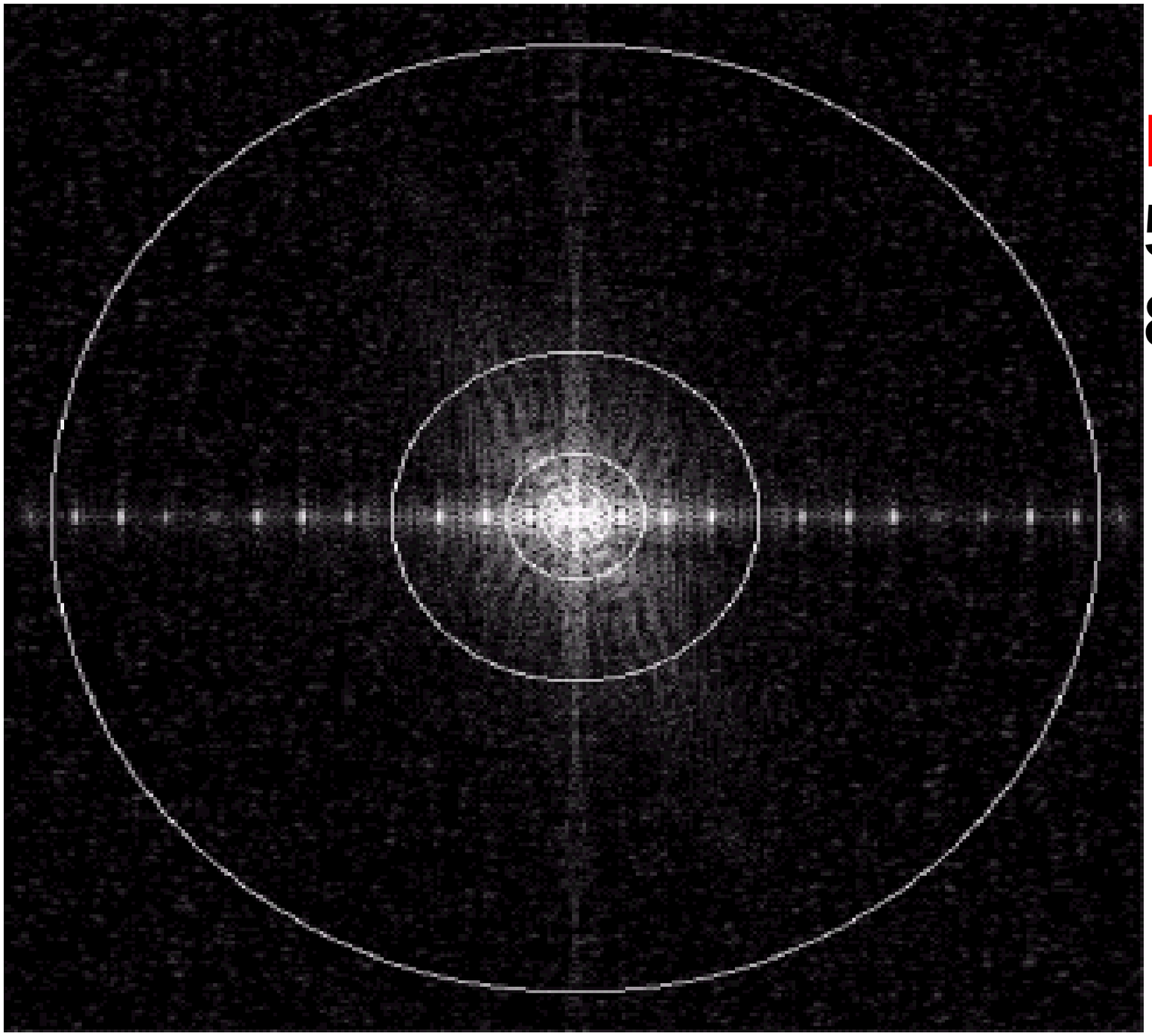


Above we show an image, it’s Fourier spectrum and a series of ideal low pass filters of radius 5, 15, 30, 80 and 230 superimposed on top of it

Ideal Low Pass Filter (cont…)



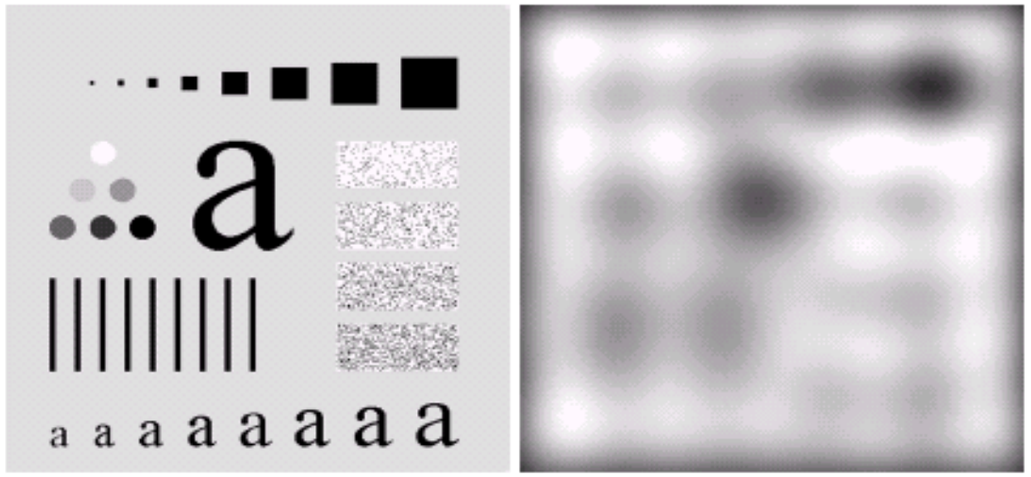
Ideal Low Pass Filter (cont…)



Radius (D_0)
5, 15, 30,
80 and 230

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Ideal Low Pass Filter (cont…)



Original image

Result of filtering with ideal low pass filter of radius 5

Result of filtering with ideal low pass filter of radius 15

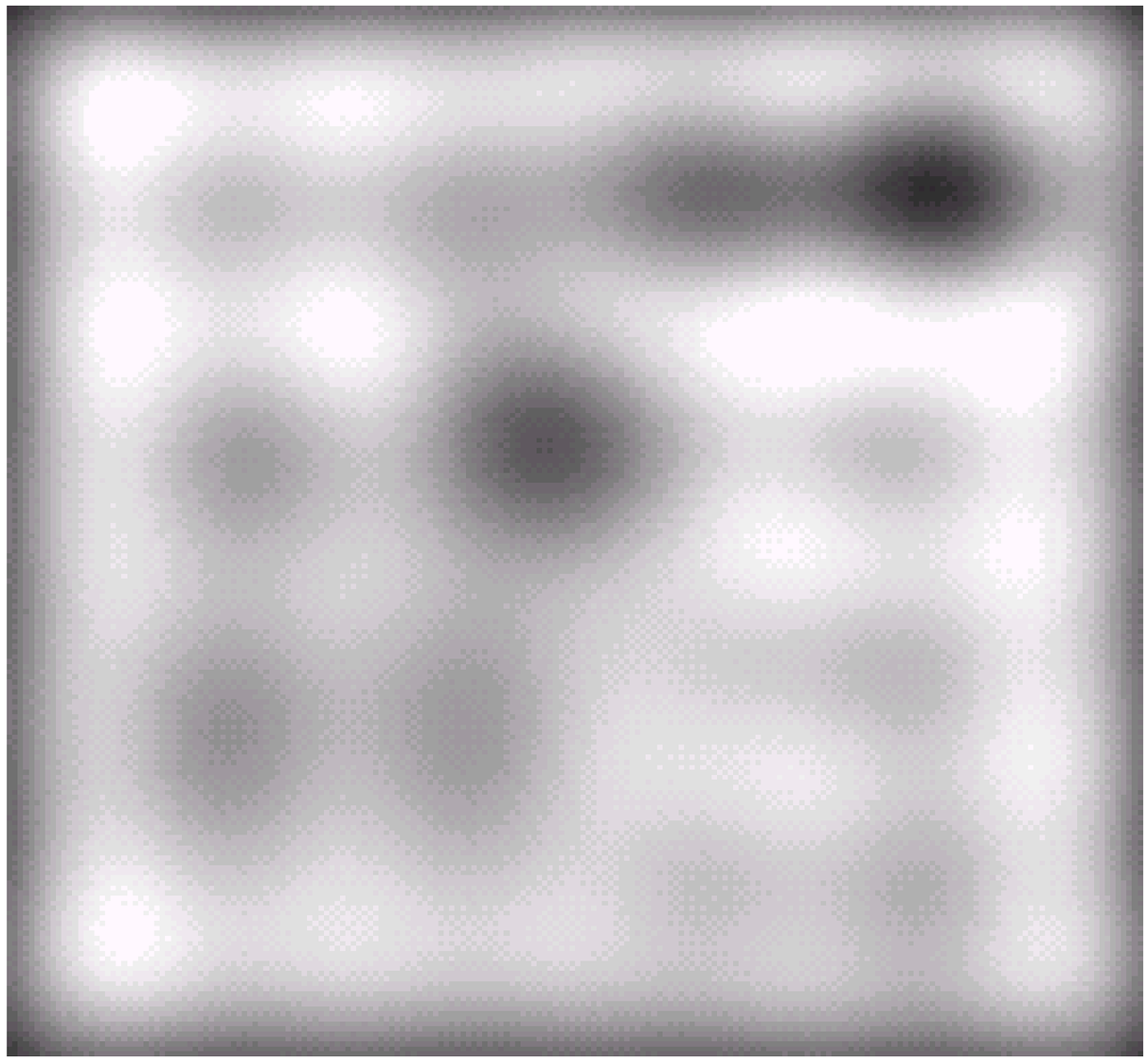
Result of filtering with ideal low pass filter of radius 30

Result of filtering with ideal low pass filter of radius 80

Result of filtering with ideal low pass filter of radius 230

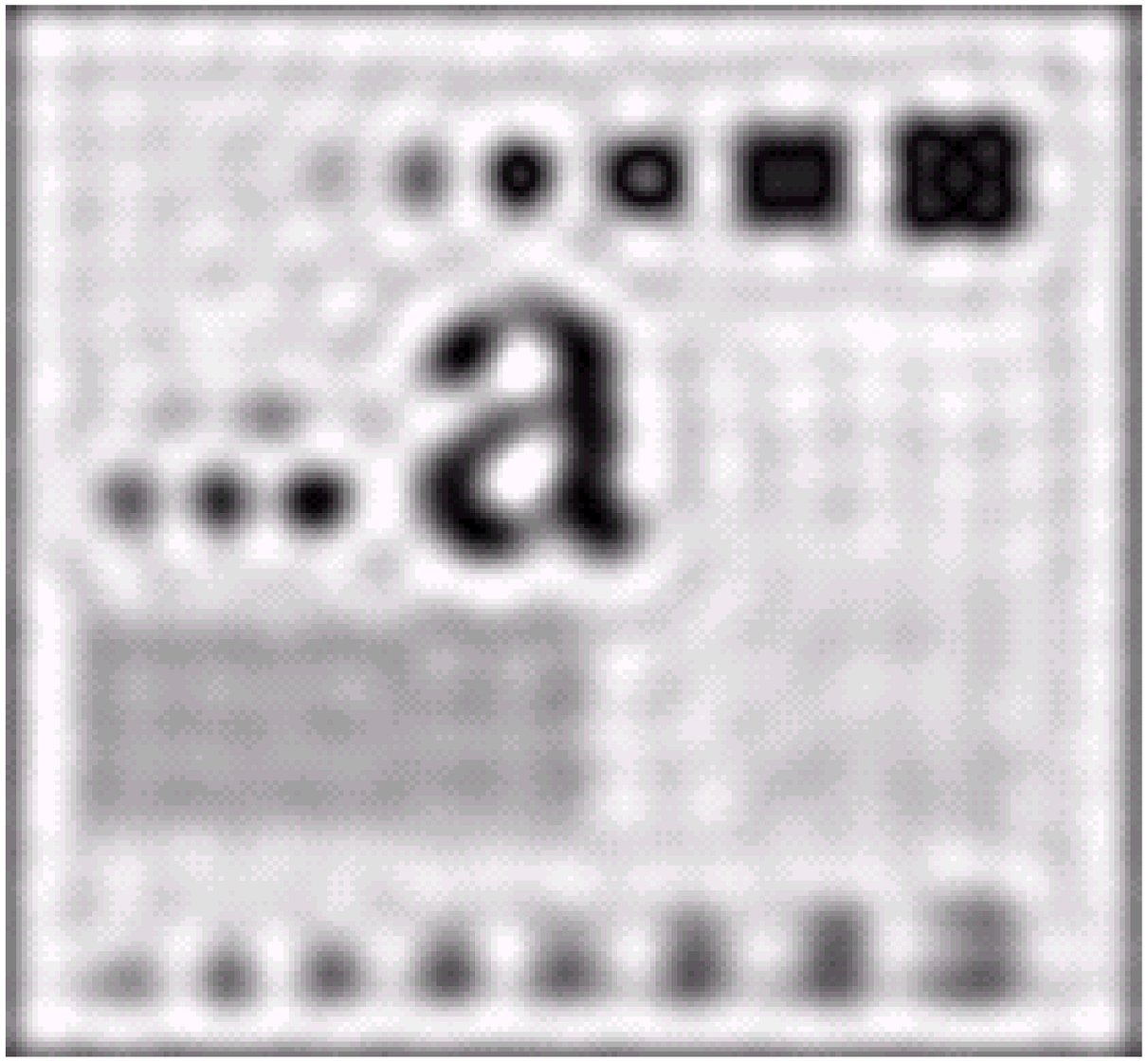
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Ideal Low Pass Filter (cont…)



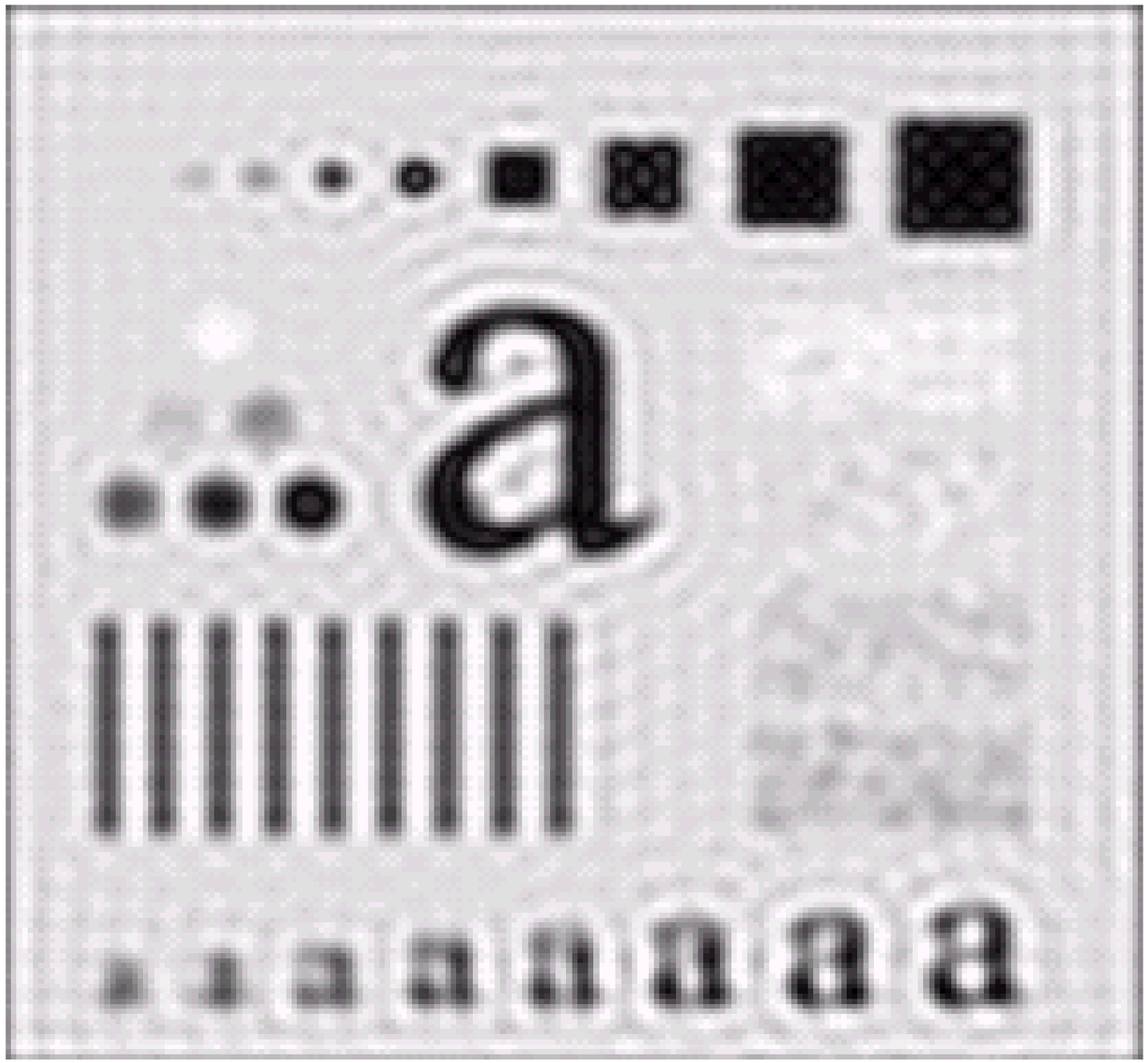
Result of filtering with ideal low pass filter of radius 5

Ideal Low Pass Filter (cont…)



Result of filtering with ideal low pass filter of radius 15

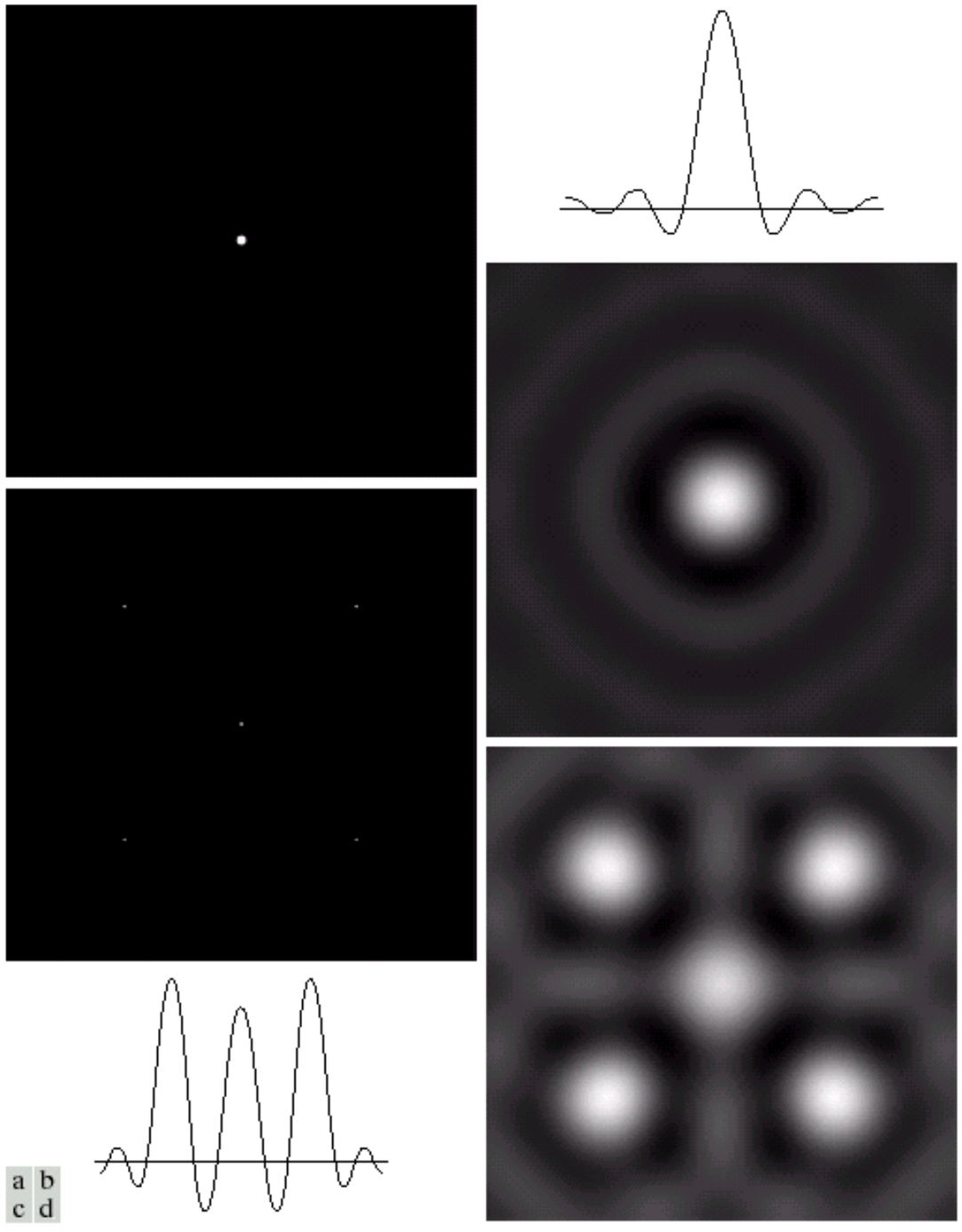
Ideal Low Pass Filter (cont…)



Result of filtering with ideal low pass filter of radius 30

Ripple or ringing effect

Ideal Low Pass Filter (cont…)

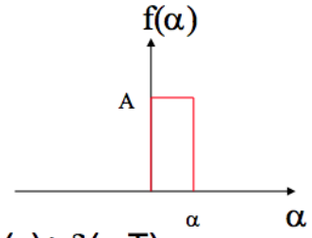


a b
c d

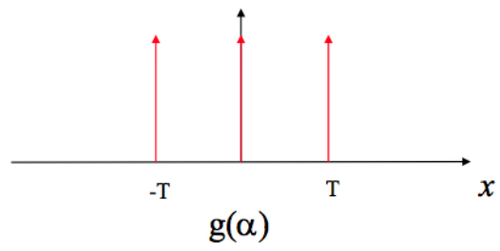
FIGURE 4.13 (a) A frequency-domain ILPF of radius 5. (b) Corresponding spatial filter (note the ringing). (c) Five impulses in the spatial domain, simulating the values of five pixels. (d) Convolution of (b) and (c) in the spatial domain.

Convolution with impulse functions

- Given f(x) is

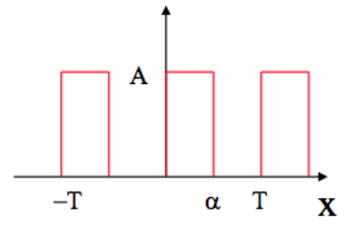


- and g(x)=δ(x+T)+ δ(x)+ δ(x-T)



- f(x)*g(x) is

  Ripple or ringing effect

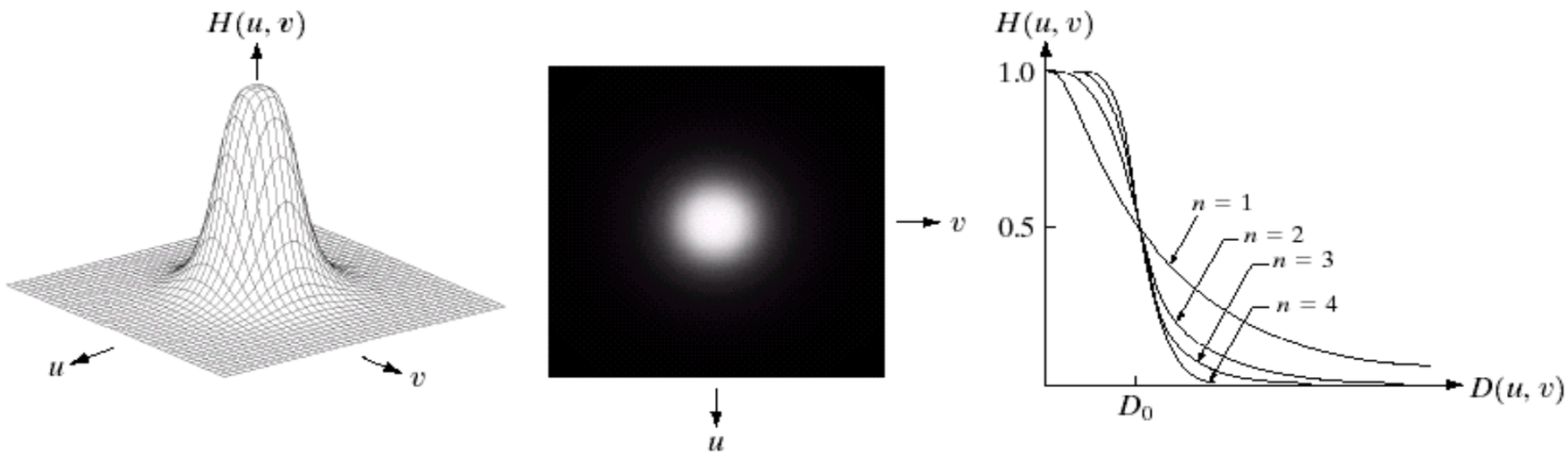


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

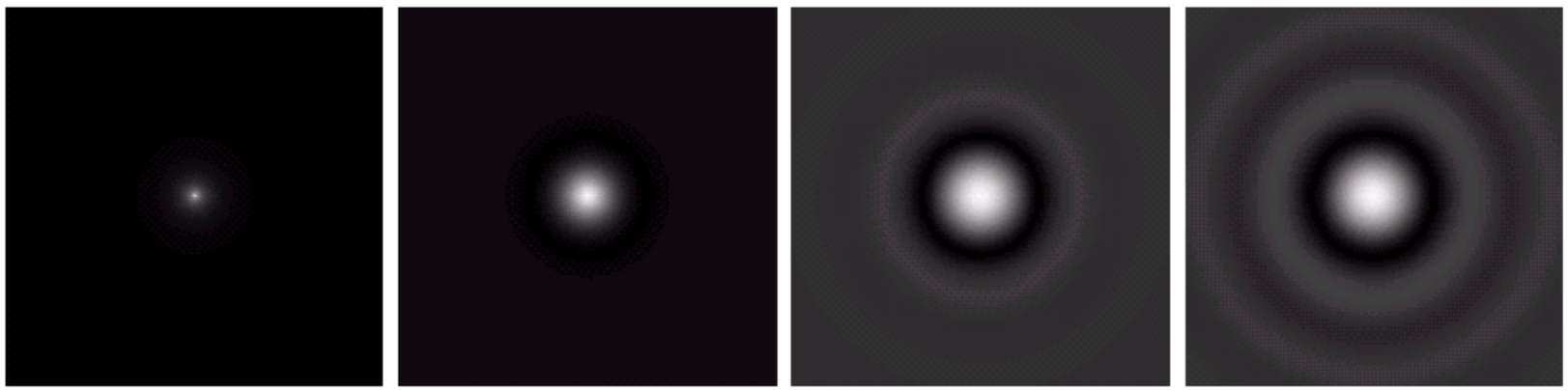
Butterworth Lowpass Filters

The transfer function of a Butterworth lowpass filter of order n with cutoff frequency at distance D_0 from the origin is defined as:

$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$



Butterworth Lowpass Filter (cont…)



a b c d

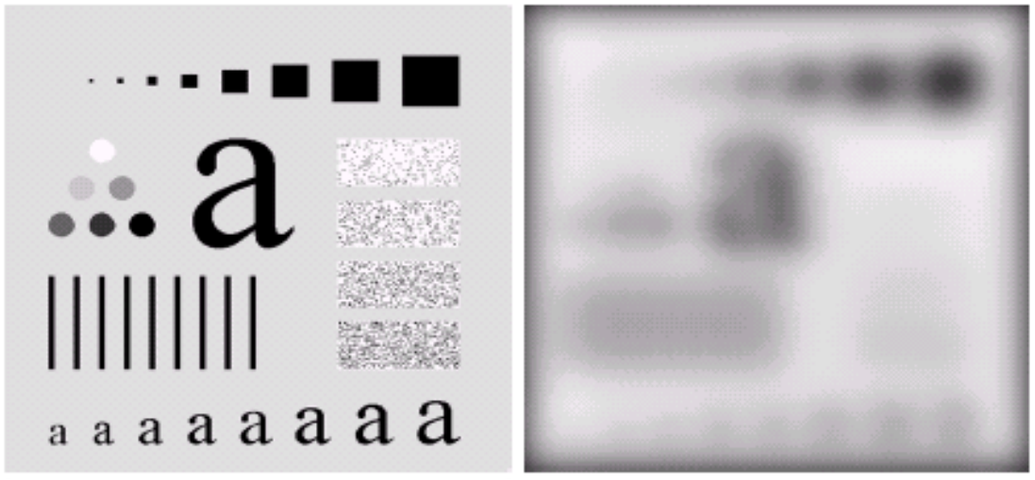
FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

Ripple or ringing effect increases with the increasing of order

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

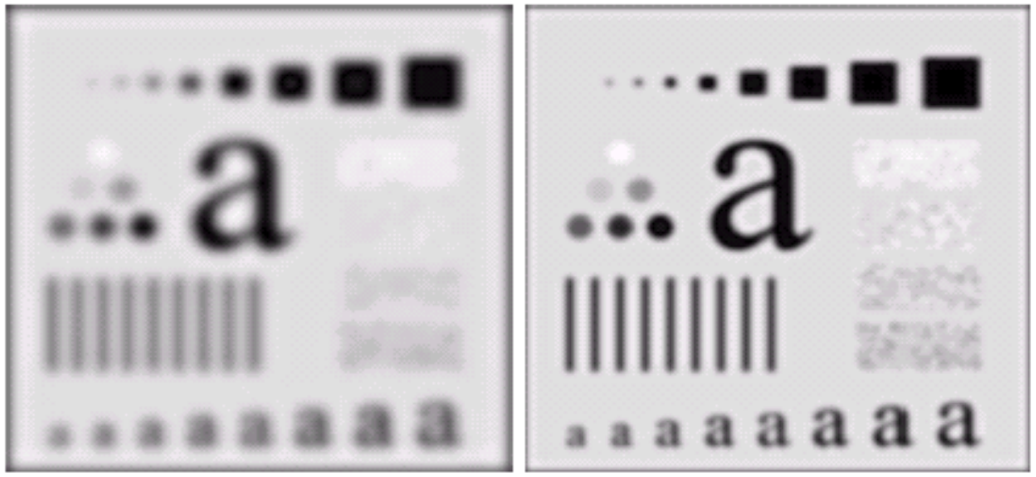
Butterworth Lowpass Filter (cont…)

Original image



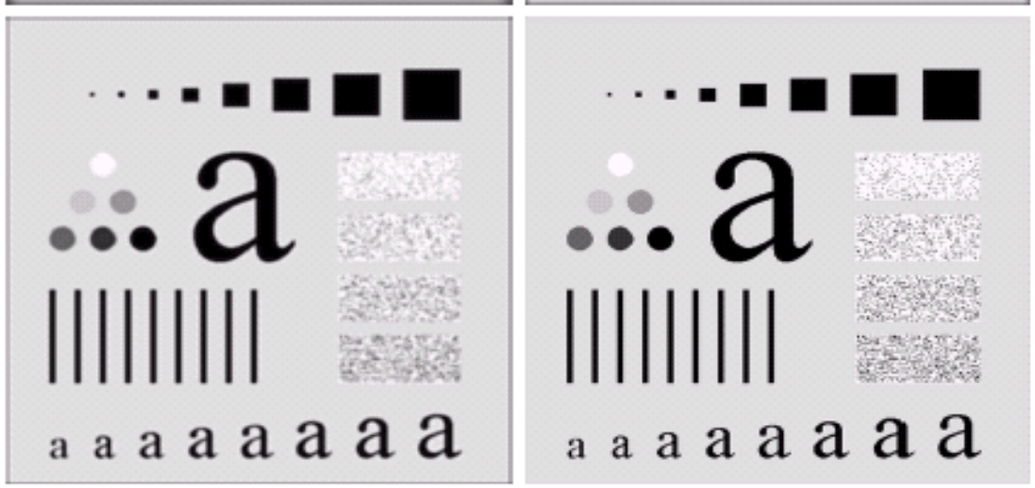
Result of filtering with Butterworth filter of order 2 and cutoff radius 5

Result of filtering with Butterworth filter of order 2 and cutoff radius 15



Result of filtering with Butterworth filter of order 2 and cutoff radius 30

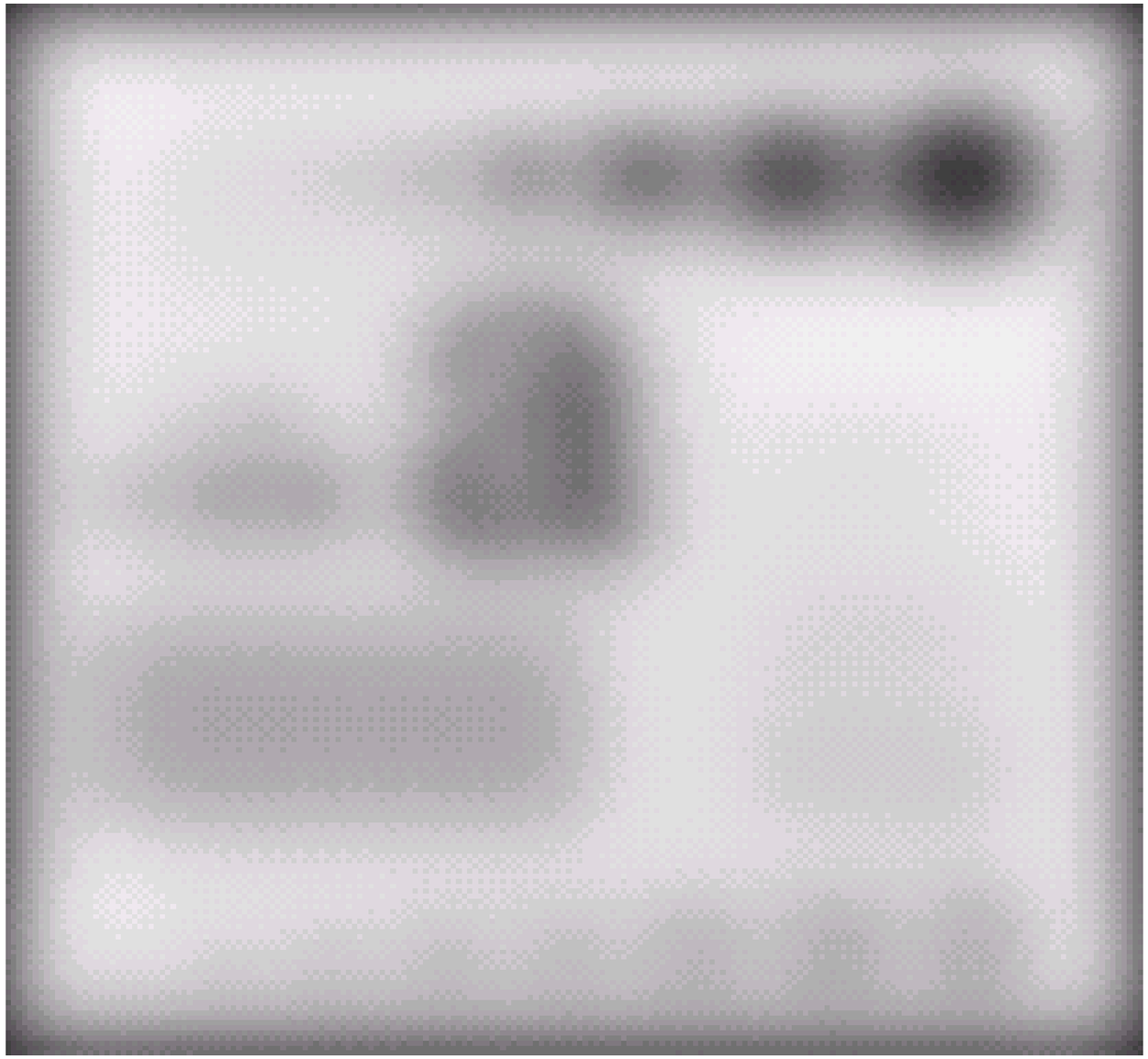
Result of filtering with Butterworth filter of order 2 and cutoff radius 80



Result of filtering with Butterworth filter of order 2 and cutoff radius 230

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

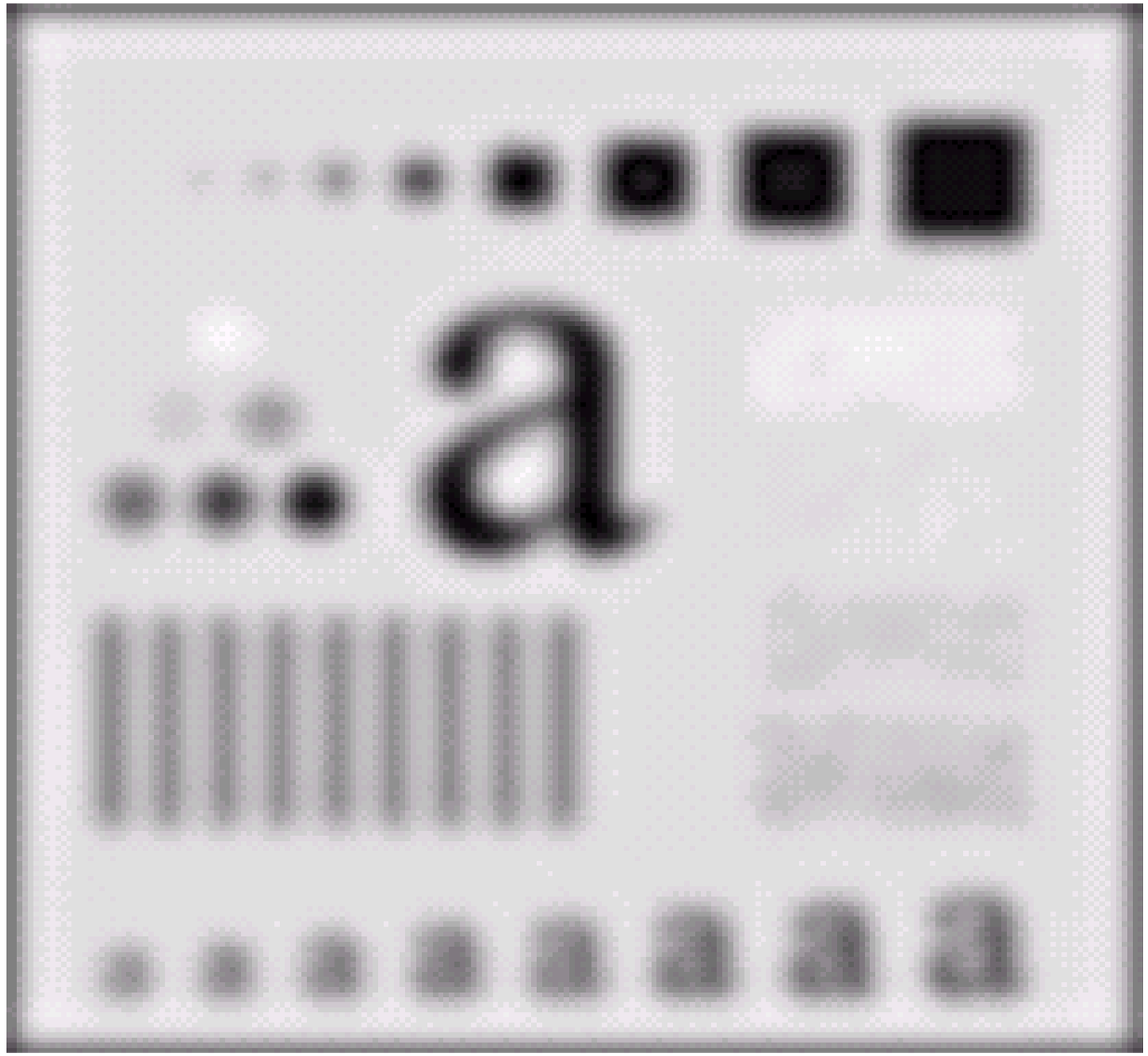
Butterworth Lowpass Filter (cont…)



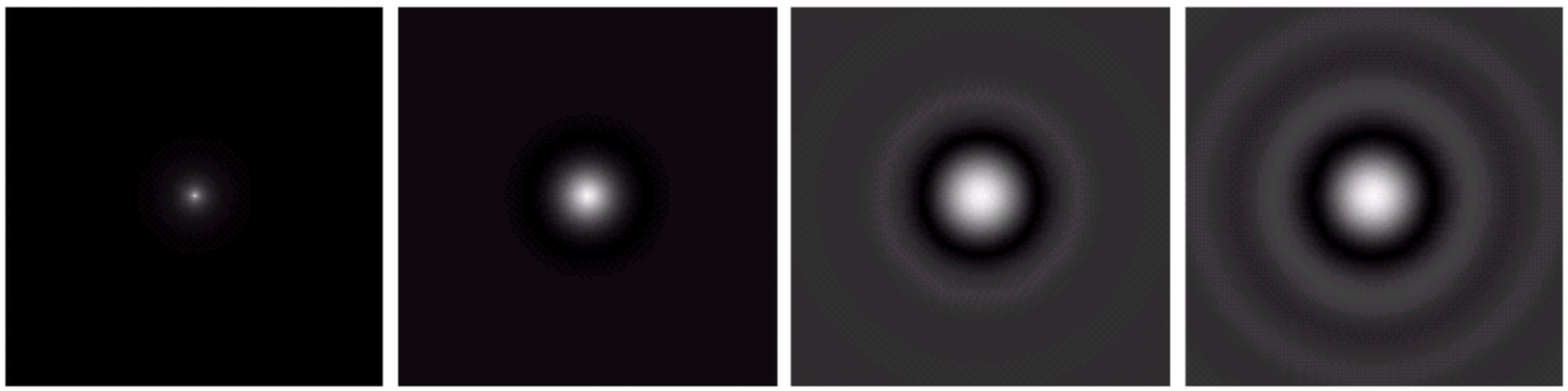
Result of filtering with Butterworth filter of order 2 and cutoff radius 5

Butterworth Lowpass Filter (cont…)

Result of filtering with Butterworth filter of order 2 and cutoff radius 15



Butterworth Lowpass Filter (cont…)



a b c d

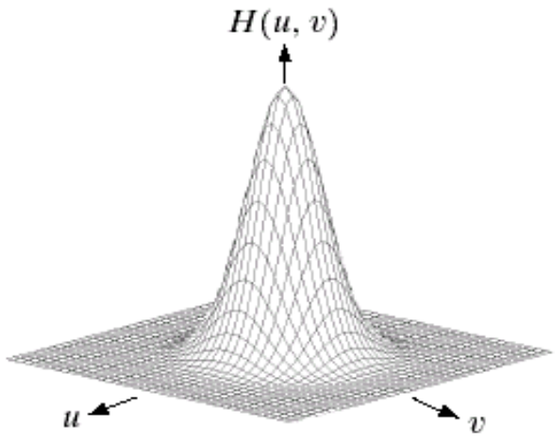
FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

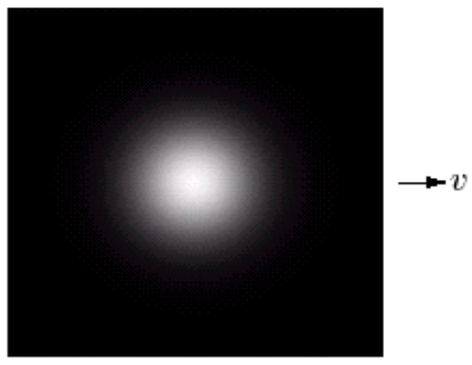
Gaussian Lowpass Filters

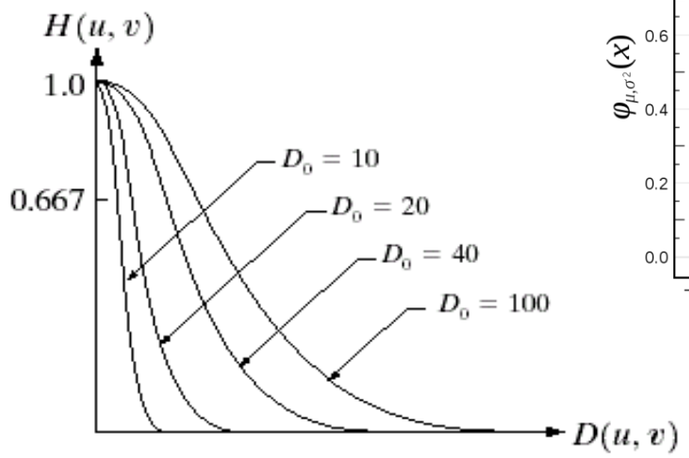
The transfer function of a Gaussian lowpass filter is defined as:

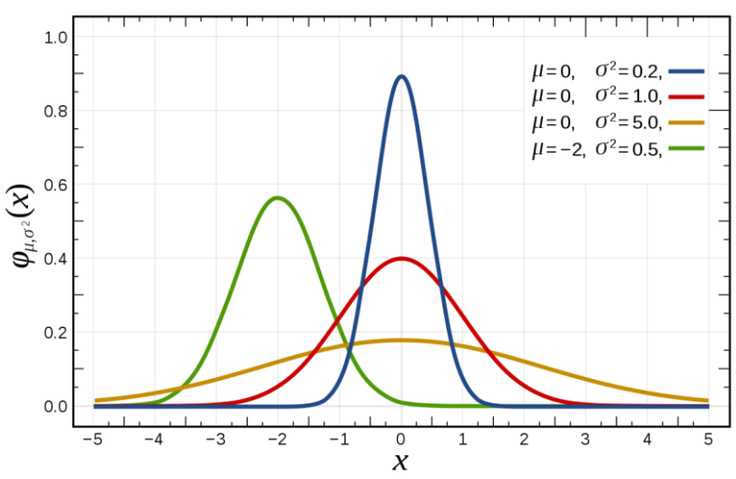
$$H(u,v) = e^{-D^2(u,v)/2D_0{}^2}$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$





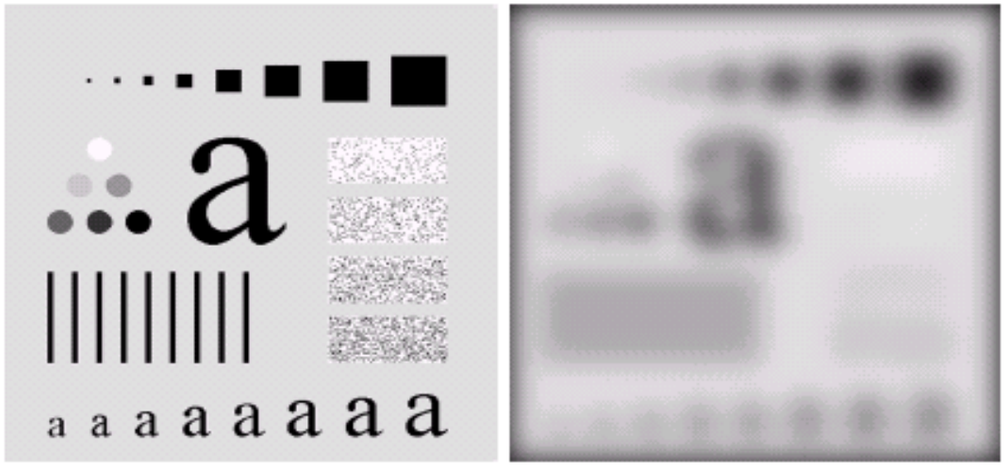




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

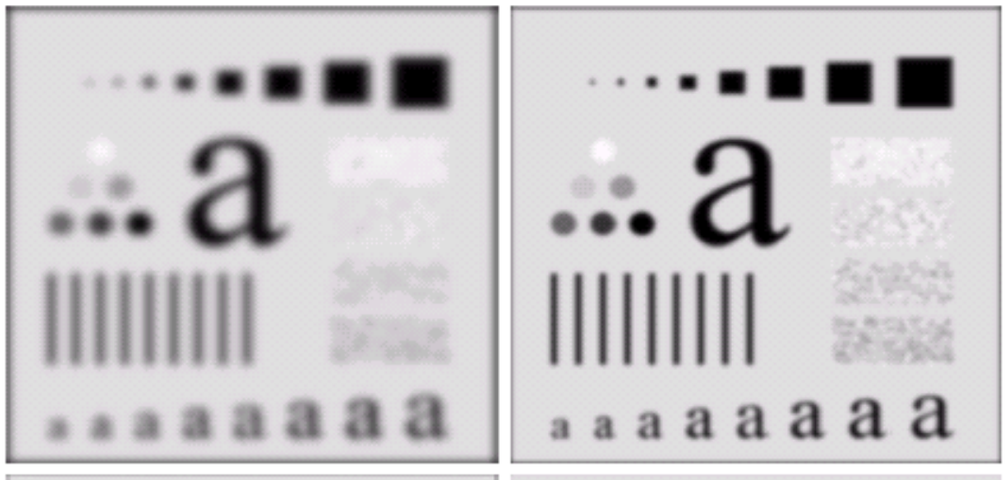
Gaussian Lowpass Filters (cont…)

Original image



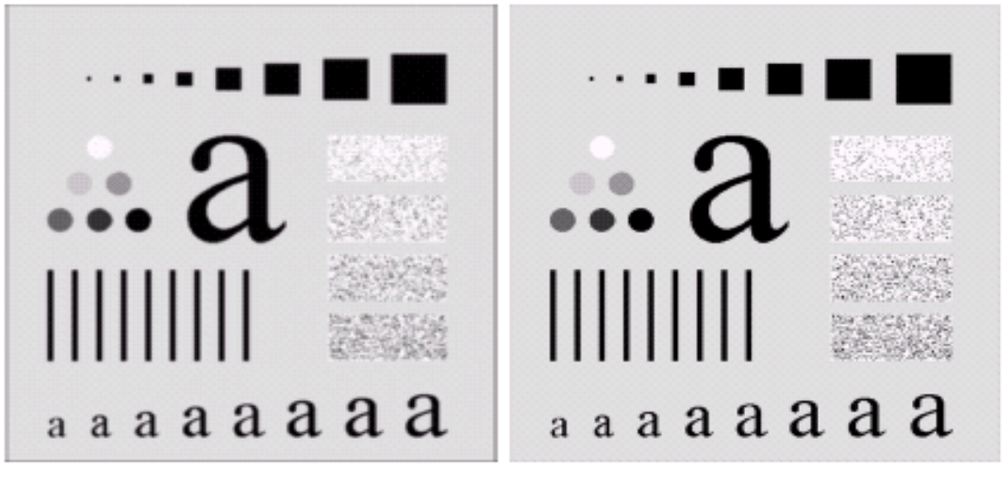
Result of filtering with Gaussian filter with cutoff radius 5

Result of filtering with Gaussian filter with cutoff radius 15



Result of filtering with Gaussian filter with cutoff radius 30

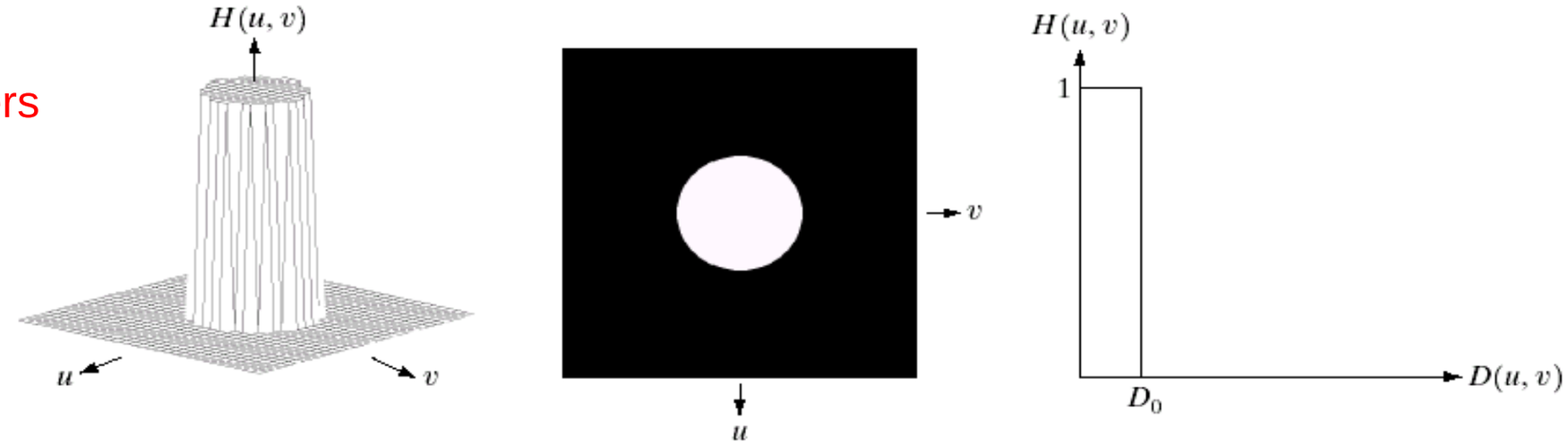
Result of filtering with Gaussian filter with cutoff radius 85



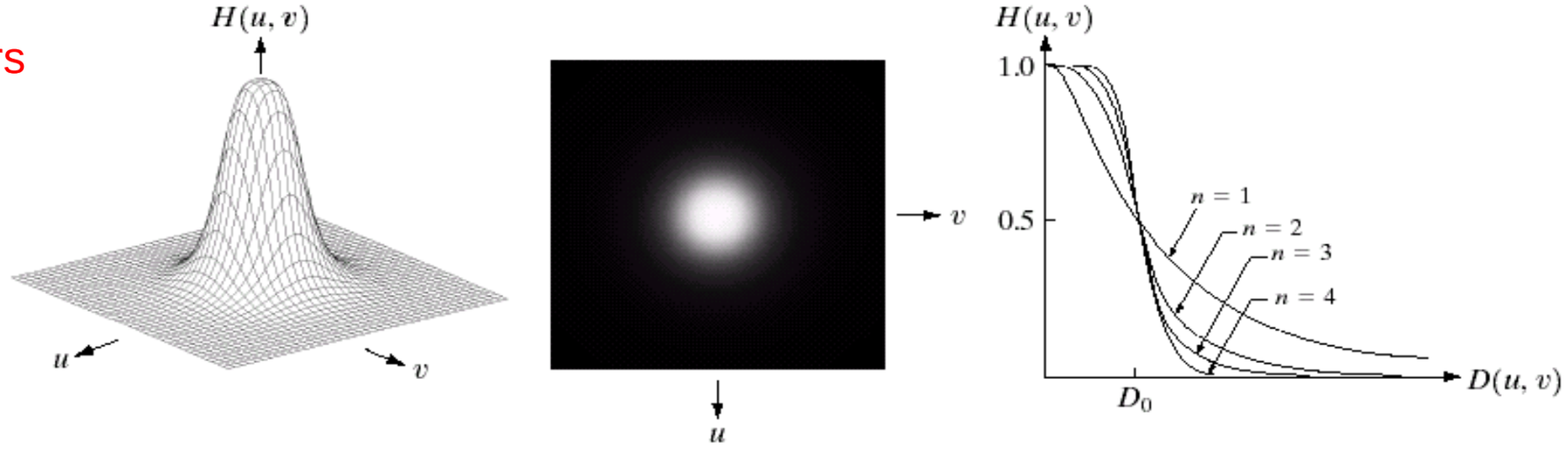
Result of filtering with Gaussian filter with cutoff radius 230

Lowpass Filters Comparison

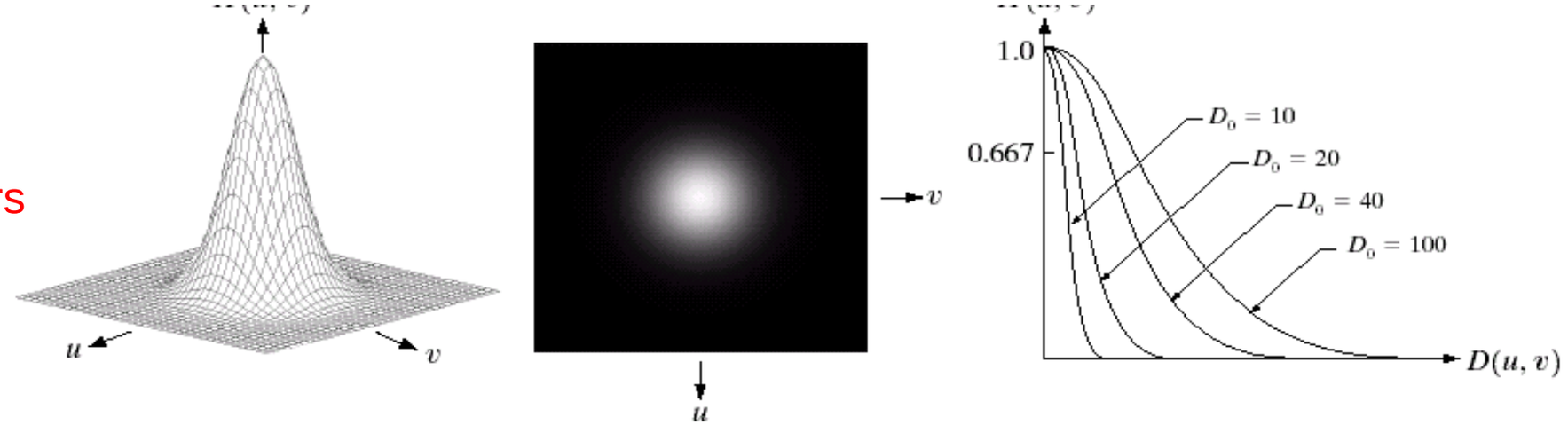
Ideal Lowpass Filters



Butterworth Lowpass Filters

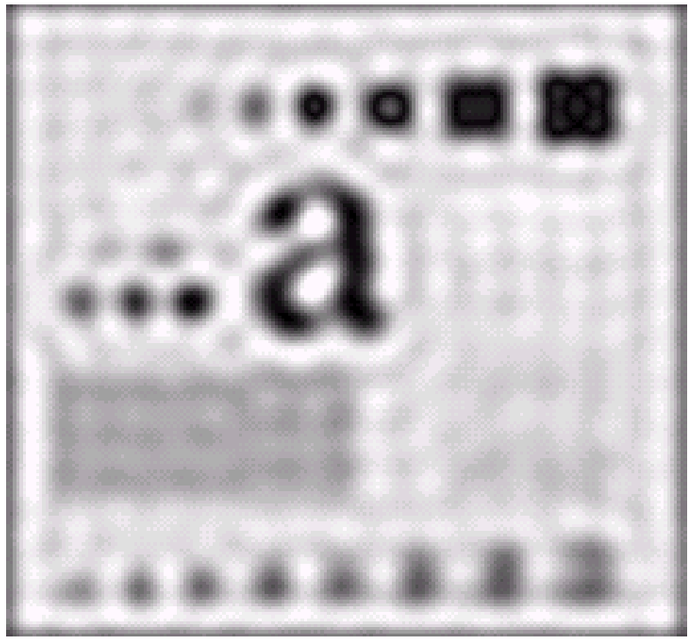


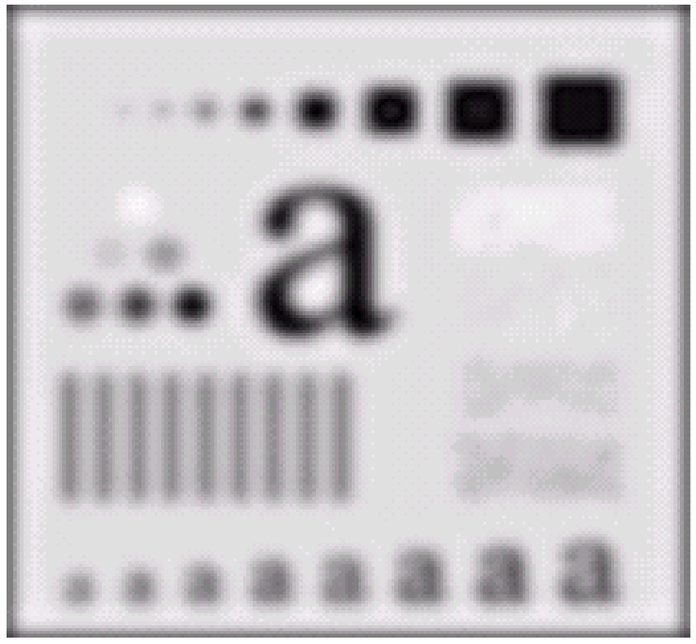
Gaussian Lowpass Filters



Lowpass Filters Compared

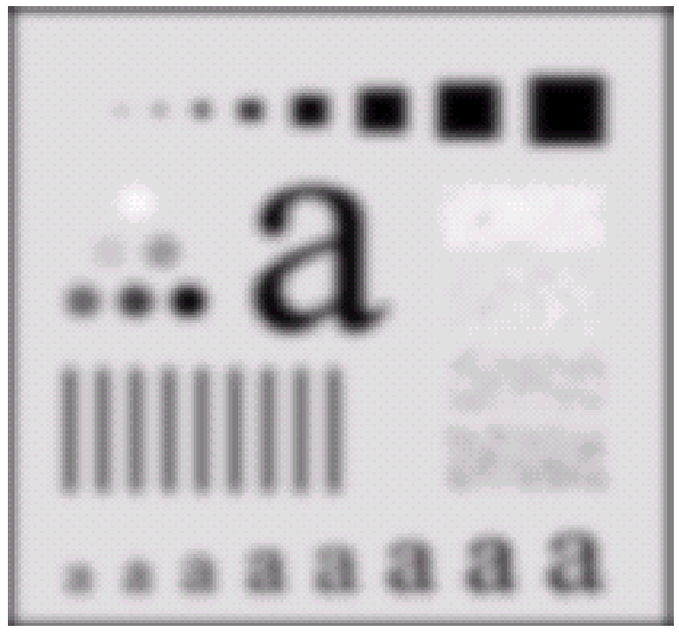
Result of filtering with ideal low pass filter of radius 15





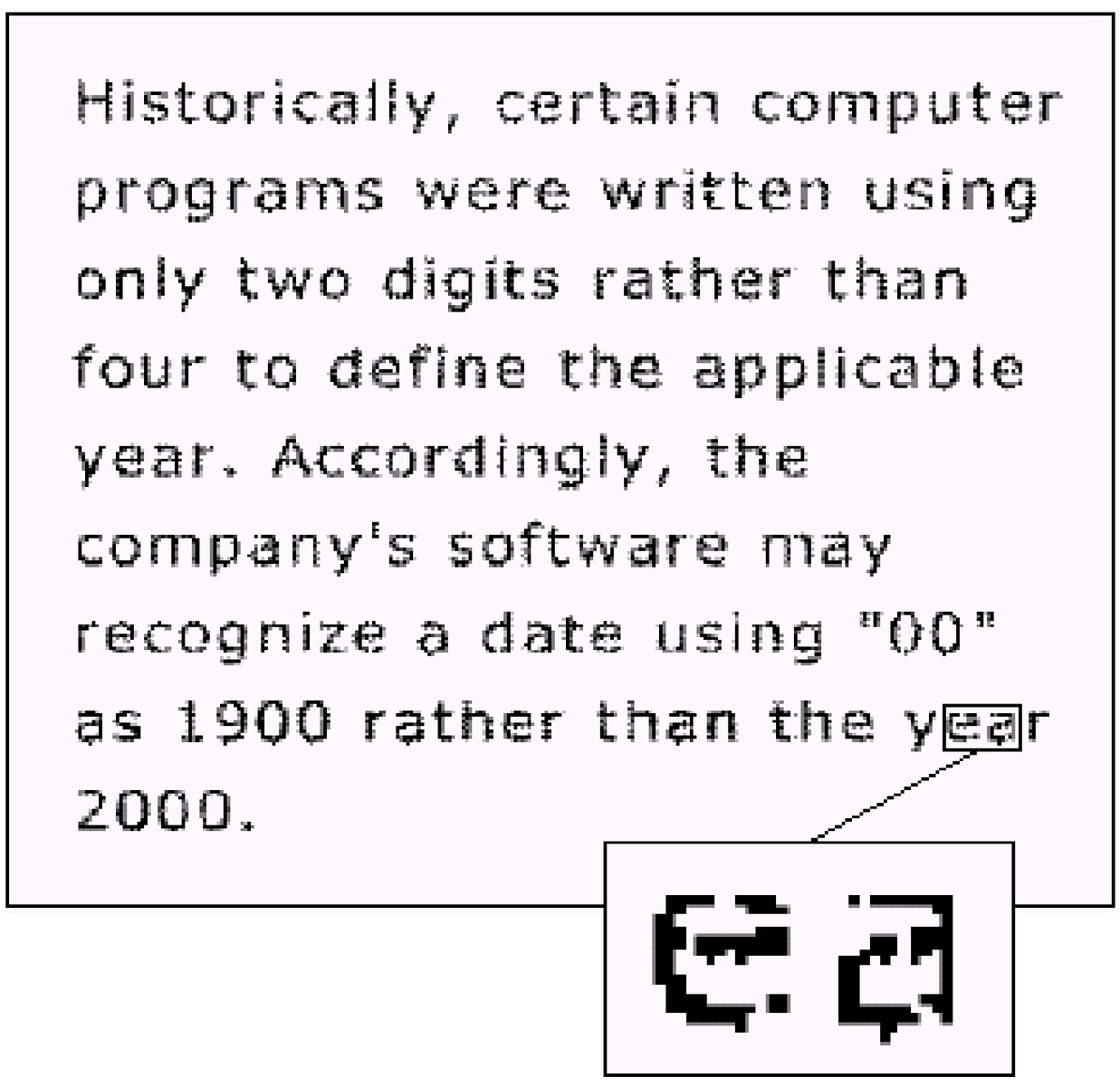
Result of filtering with Butterworth filter of order 2 and cutoff radius 15

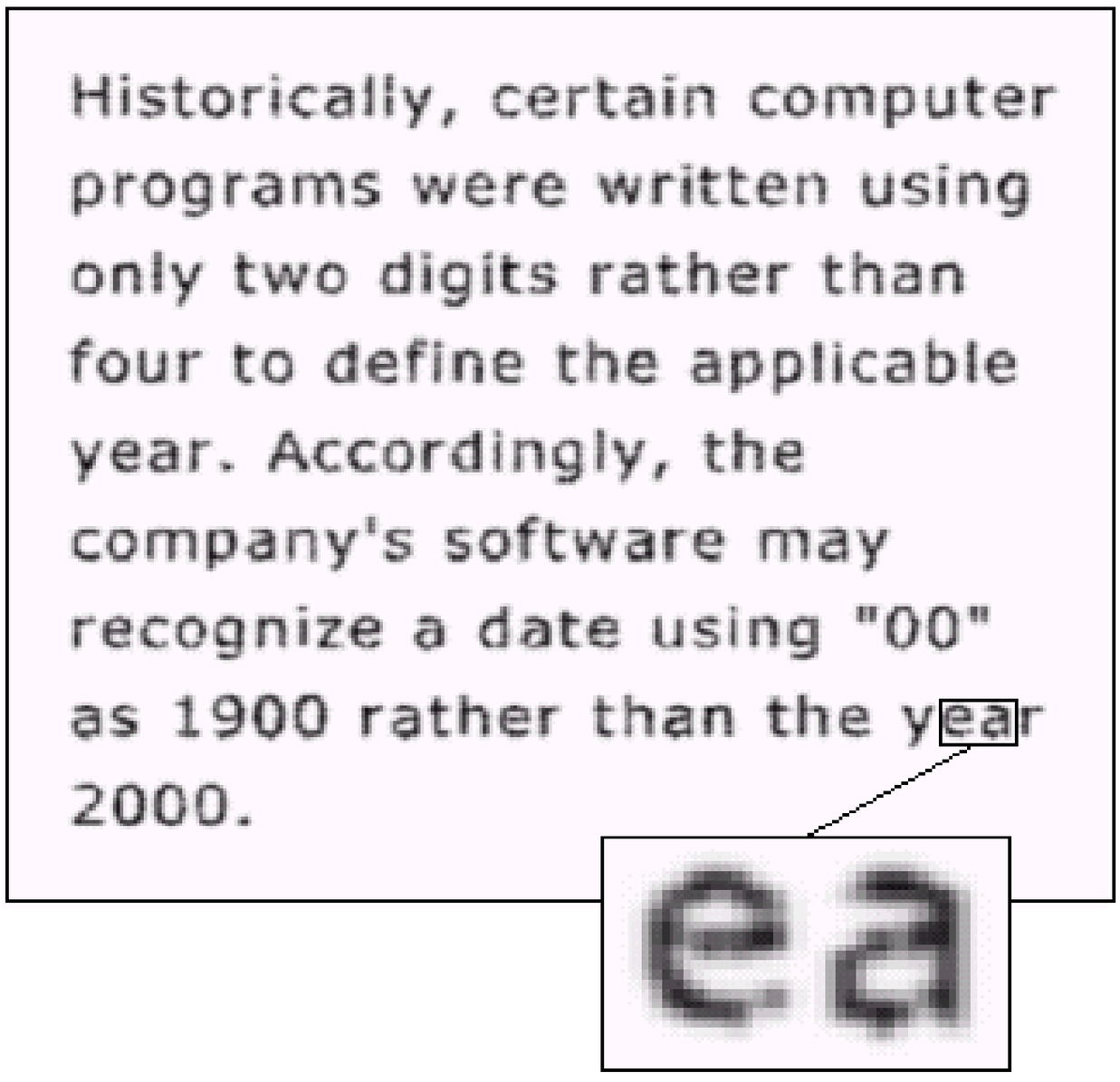
Result of filtering with Gaussian filter with cutoff radius 15



Lowpass Filtering Examples

A low pass Gaussian filter is used to connect broken text





Images taken from Gonzalez & Woods, Digital Image Processing (2002)

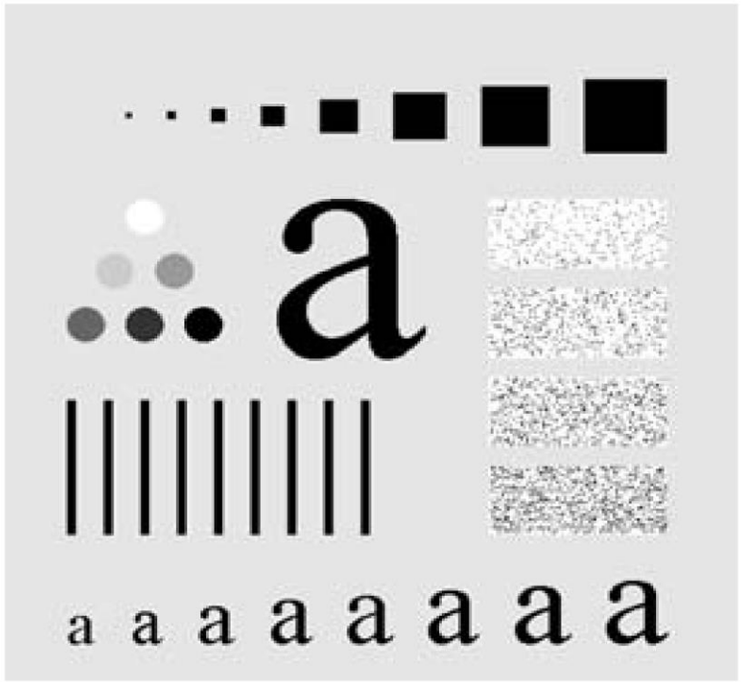
Lowpass Filtering Examples (cont…)

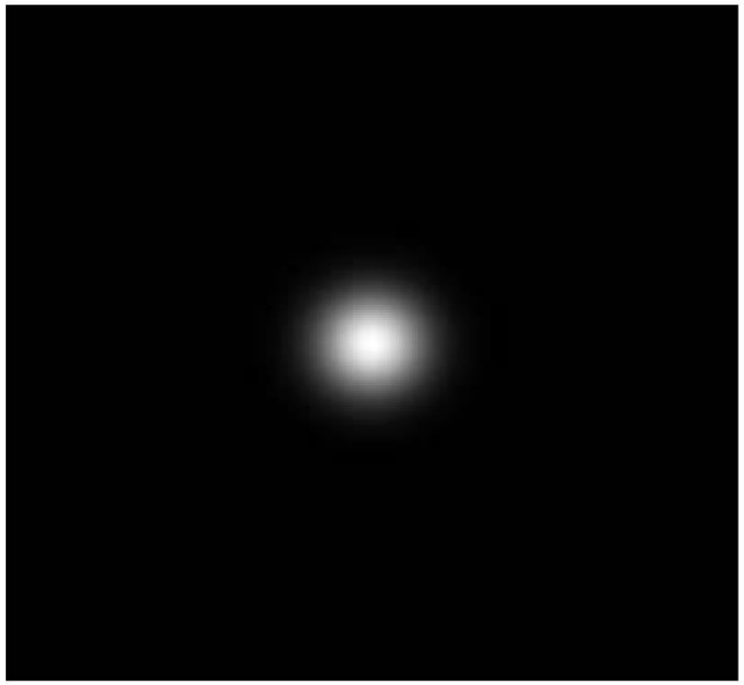
Different lowpass Gaussian filters used to remove blemishes in a photograph

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Lowpass Filtering Examples (cont…)

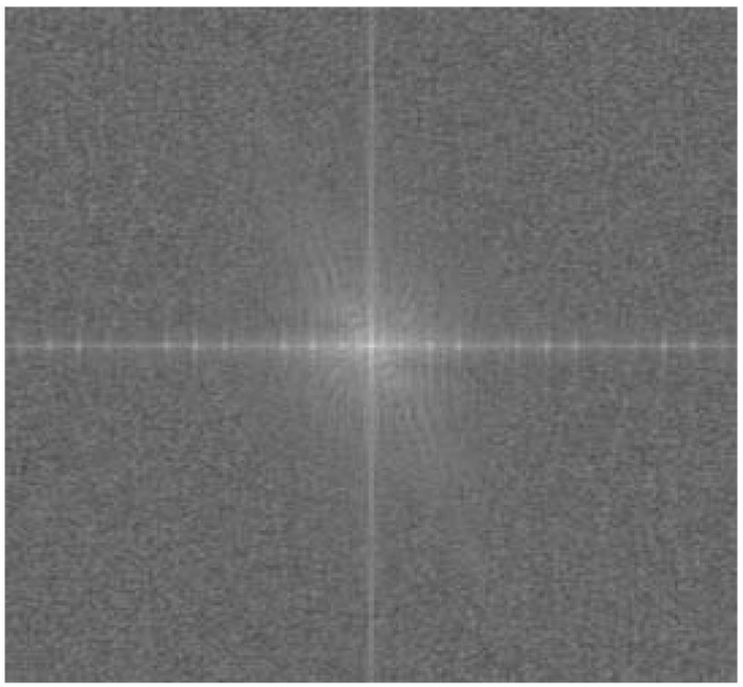
Original image





Gaussian lowpass filter

Spectrum of original image



Processed image

Sharpening in the Frequency Domain

Edges and fine detail in images are associated with high frequency components

High pass filters – only pass the high frequencies, drop the low ones

High pass frequencies are precisely the reverse of low pass filters, so:

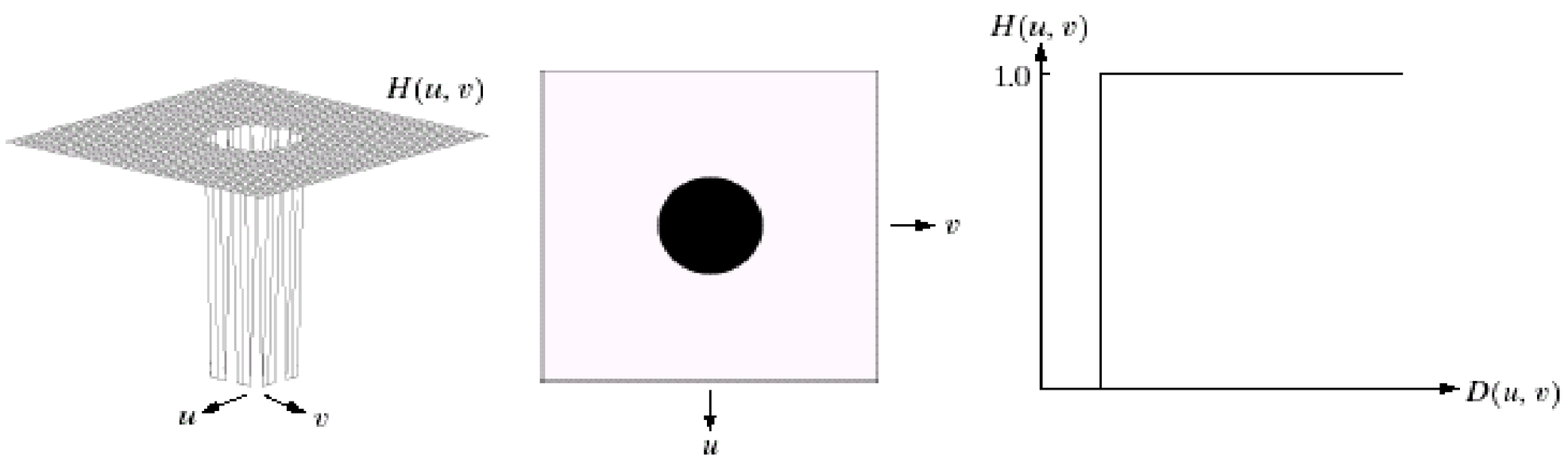
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

Ideal High Pass Filters

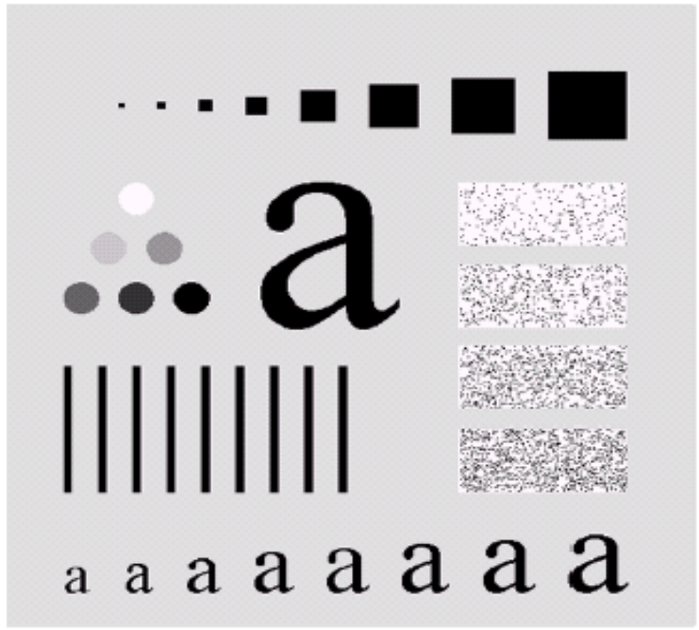
The ideal high pass filter is given as:

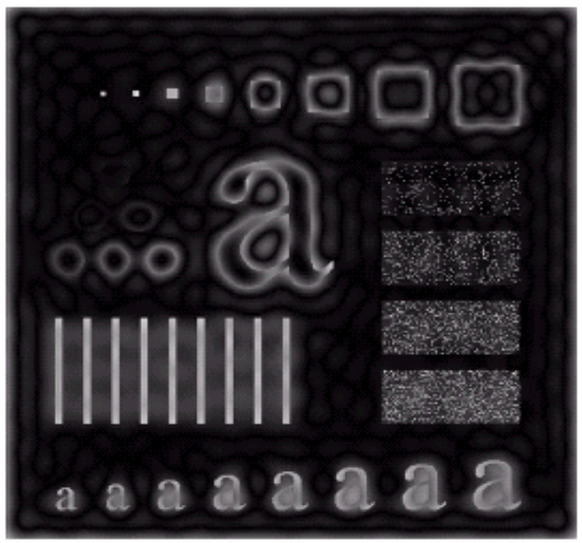
$$H(u,v) = \begin{cases} 0 \text{ if } D(u,v) \le D_0 \\ 1 \text{ if } D(u,v) > D_0 \end{cases}$$

where D_0 is the cut off distance as before

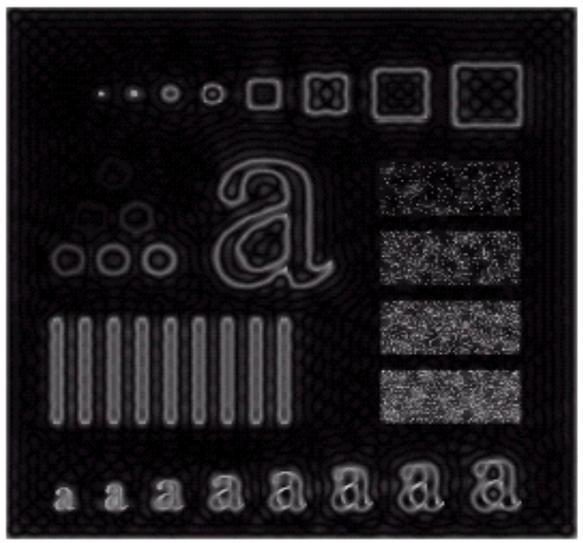


Ideal High Pass Filters (cont…)

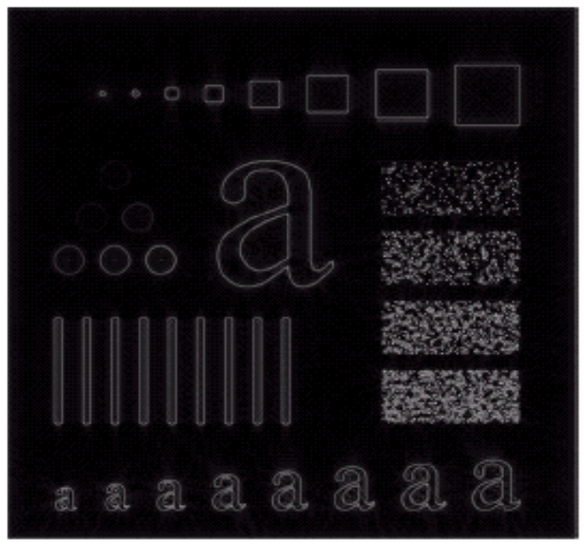




Results of ideal high pass filtering with D_0 = 15



Results of ideal high pass filtering with D_0 = 30



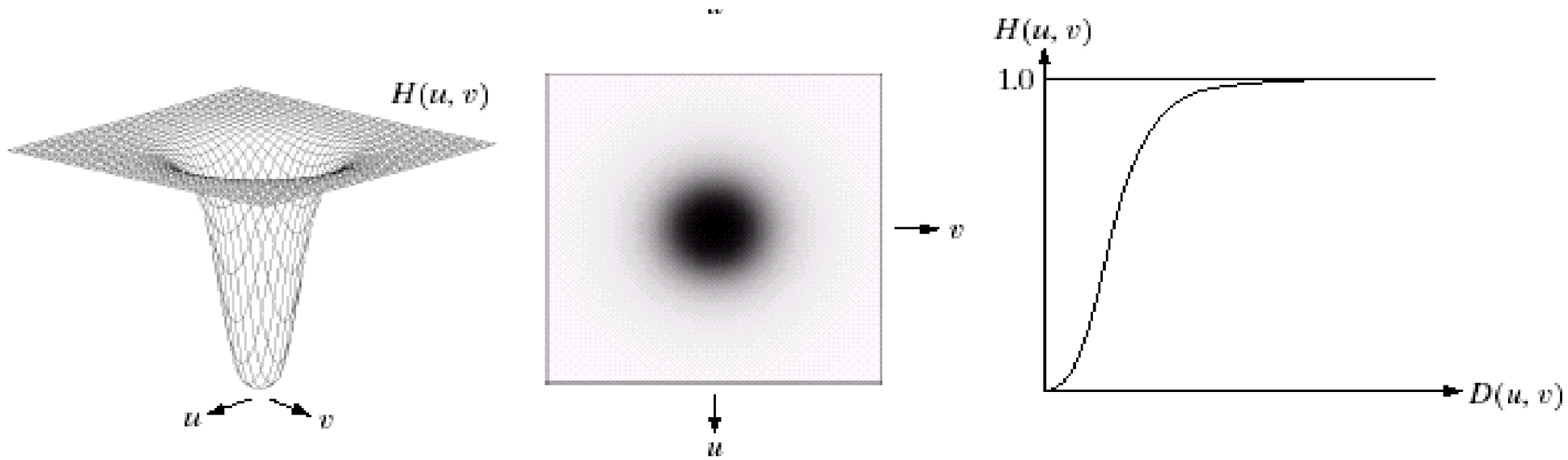
Results of ideal high pass filtering with D_0 = 80

Butterworth High Pass Filters

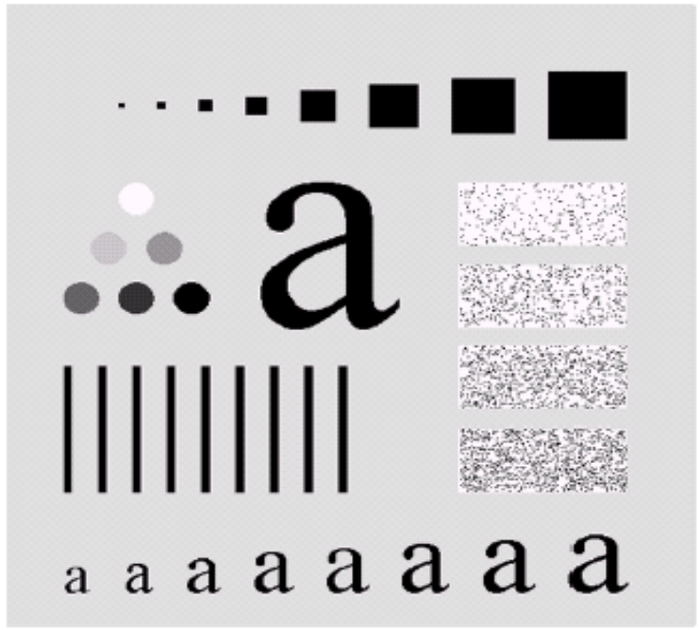
The Butterworth high pass filter is given as:

$$H(u,v) = \frac{1}{1+[D_0 / D(u,v)]^{2n}}$$

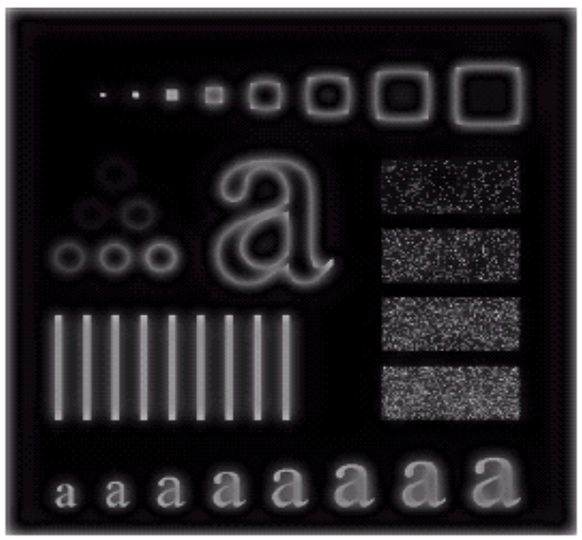
where n is the order and D_0 is the cut off distance as before

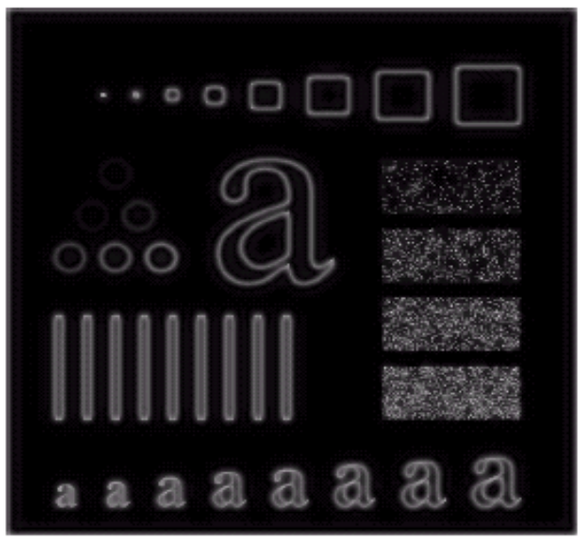


Butterworth High Pass Filters (cont…)



Results of Butterworth high pass filtering of order 2 with D_0 = 15





Results of Butterworth high pass filtering of order 2 with D_0 = 80

Results of Butterworth high pass filtering of order 2 with D_0 = 30

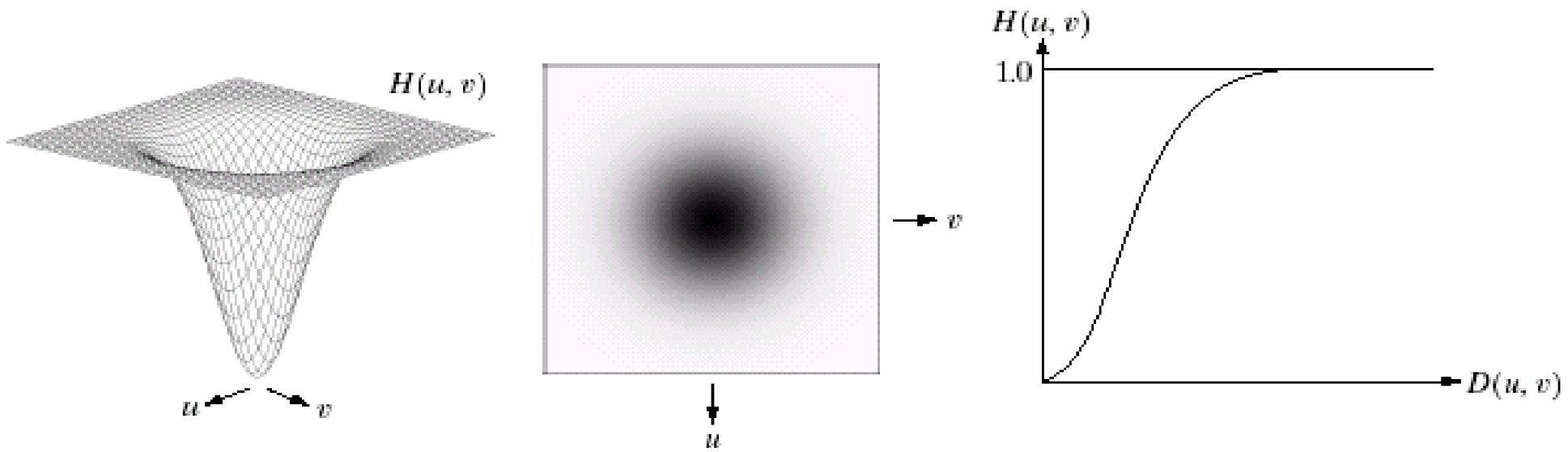
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Gaussian High Pass Filters

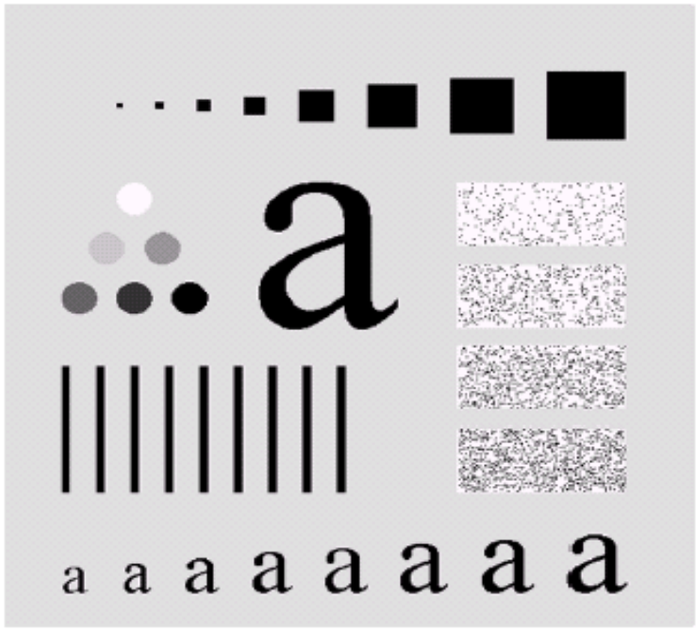
The Gaussian high pass filter is given as:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0{}^2}$$

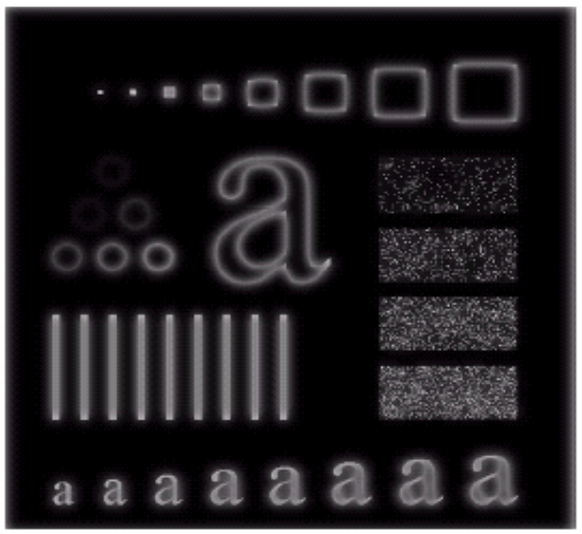
where D_0 is the cut off distance as before

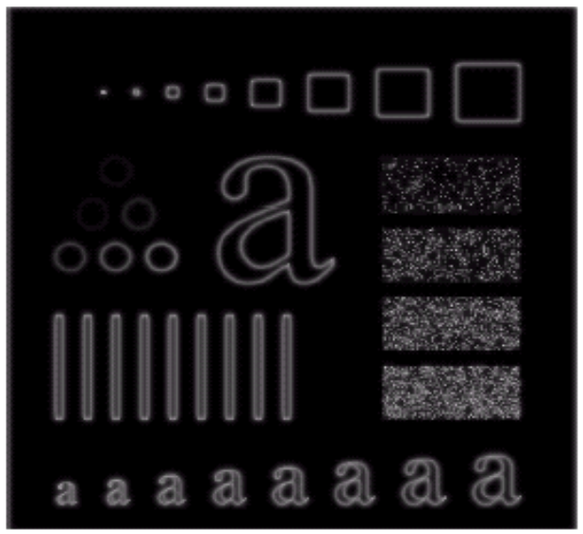


Gaussian High Pass Filters (cont…)



Results of Gaussian high pass filtering with D_0 = 15

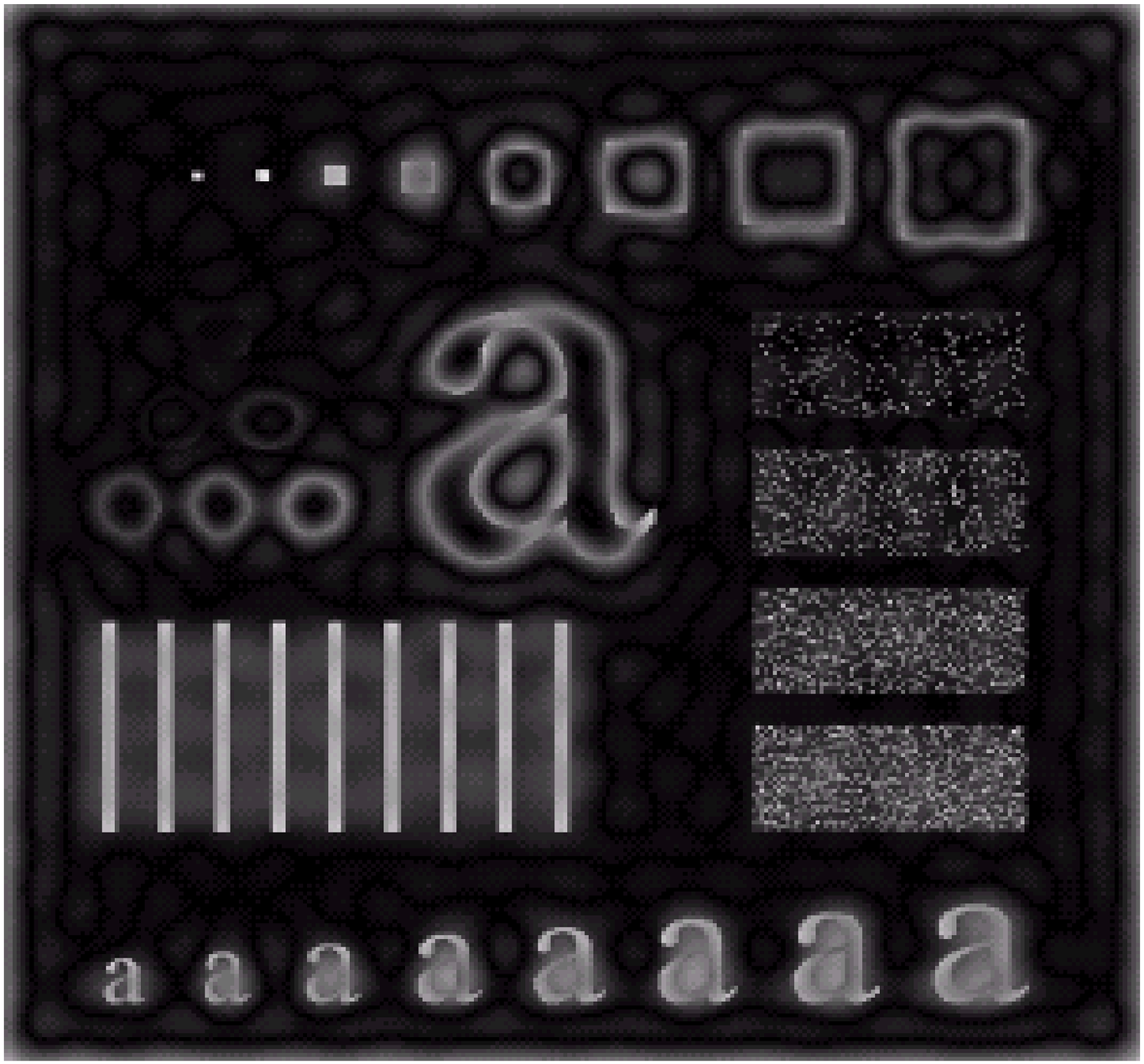




Results of Gaussian high pass filtering with D_0 = 80

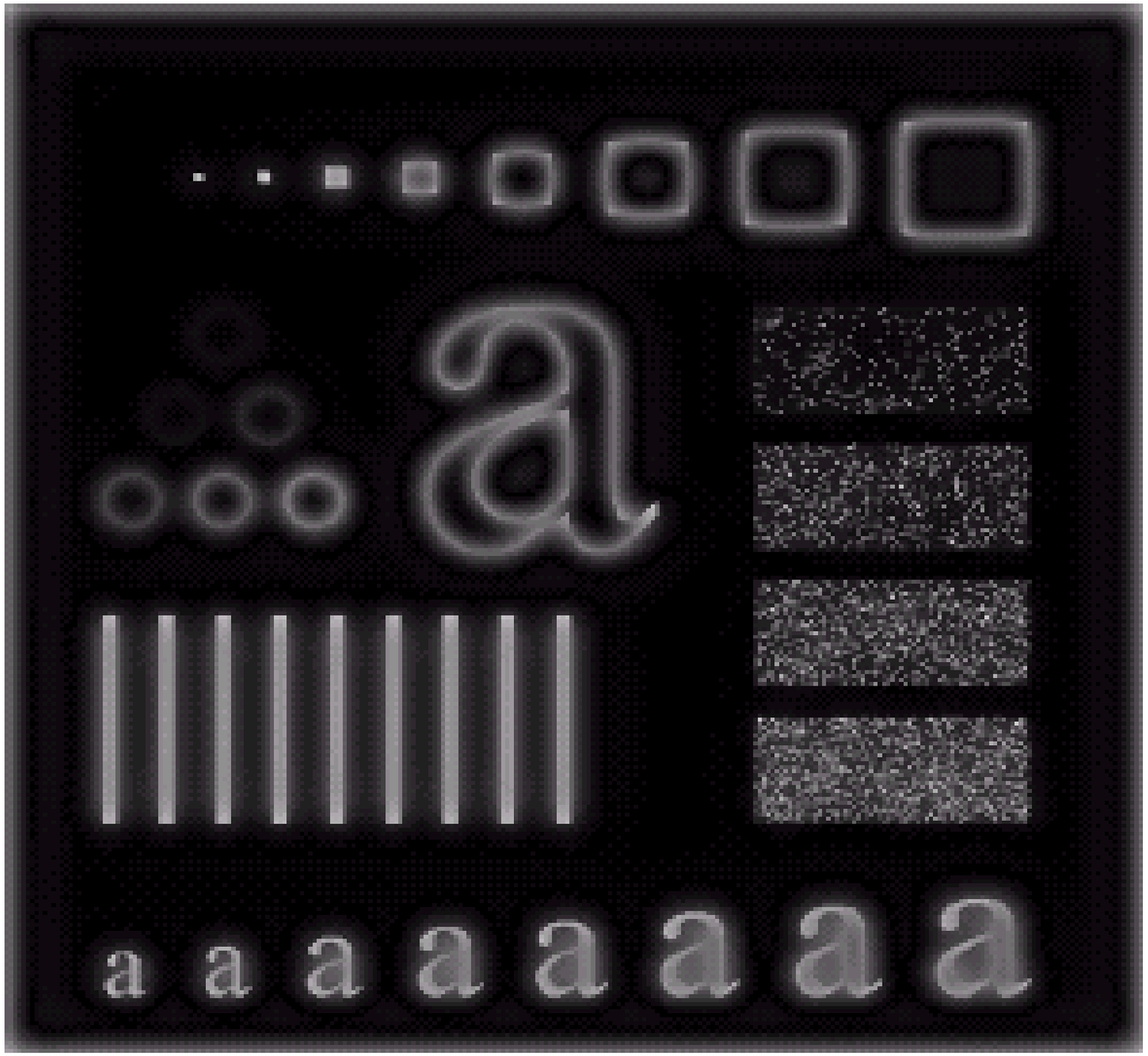
Results of Gaussian high pass filtering with D_0 = 30

Highpass Filter Comparison



Results of ideal high pass filtering with $D_0 = 15$

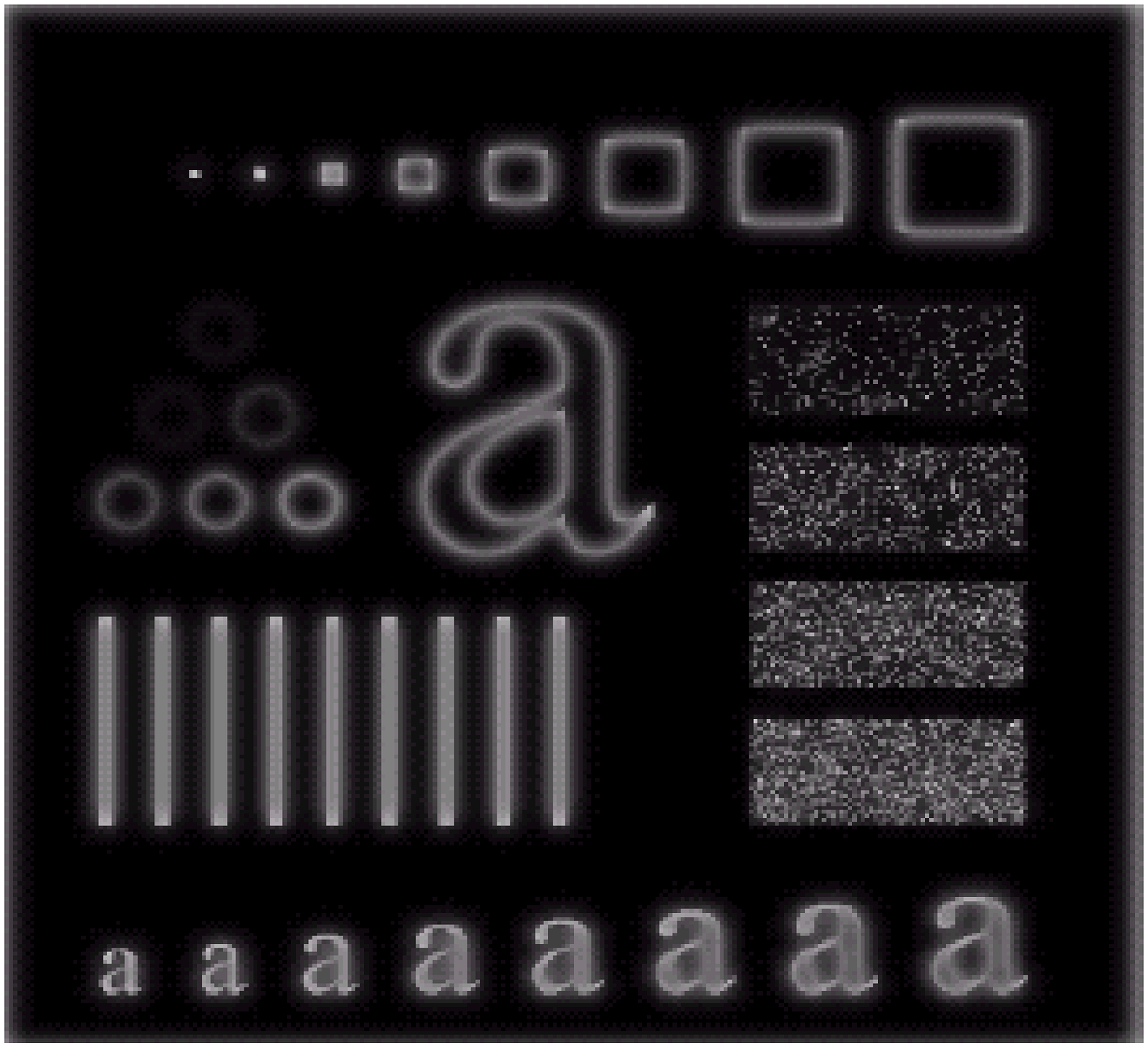
Highpass Filter Comparison



Results of Butterworth high pass filtering of order 2 with D_0 = 15

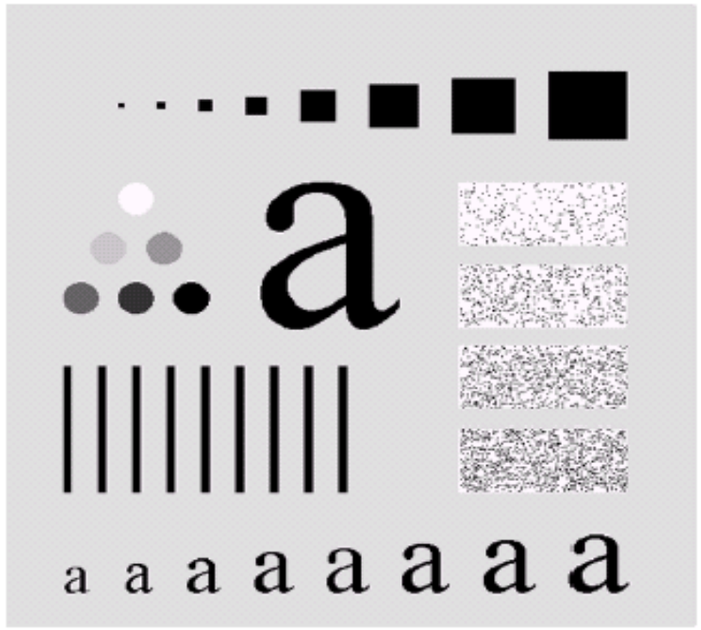
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

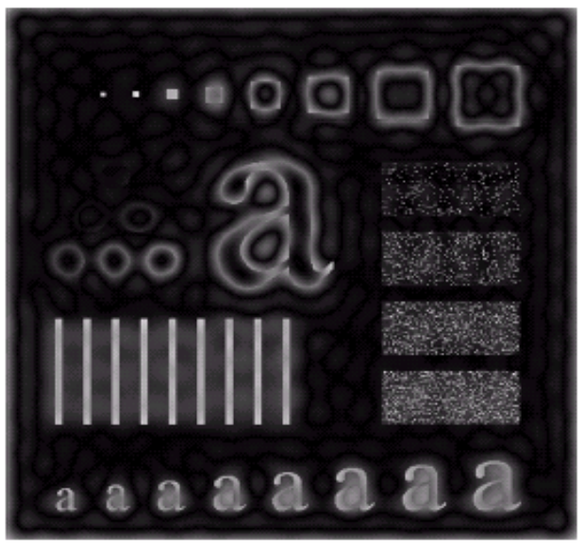
Highpass Filter Comparison



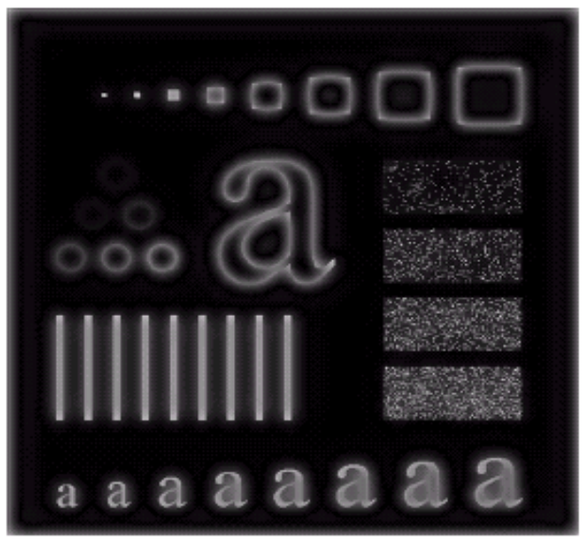
Results of Gaussian high pass filtering with $D_0 = 15$

Highpass Filter Comparison

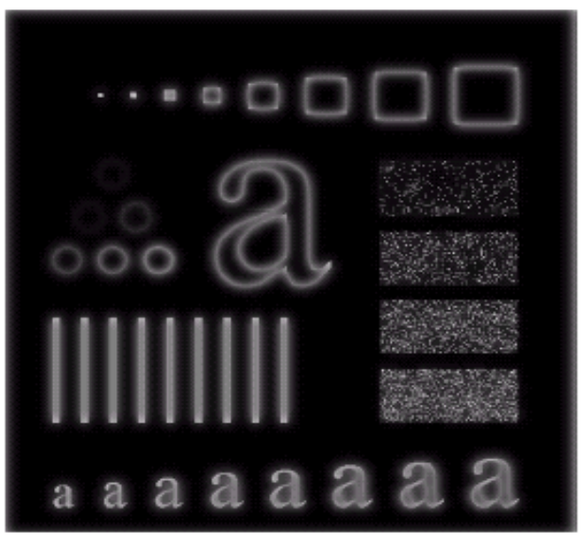




Results of ideal high pass filtering with D_0 = 15

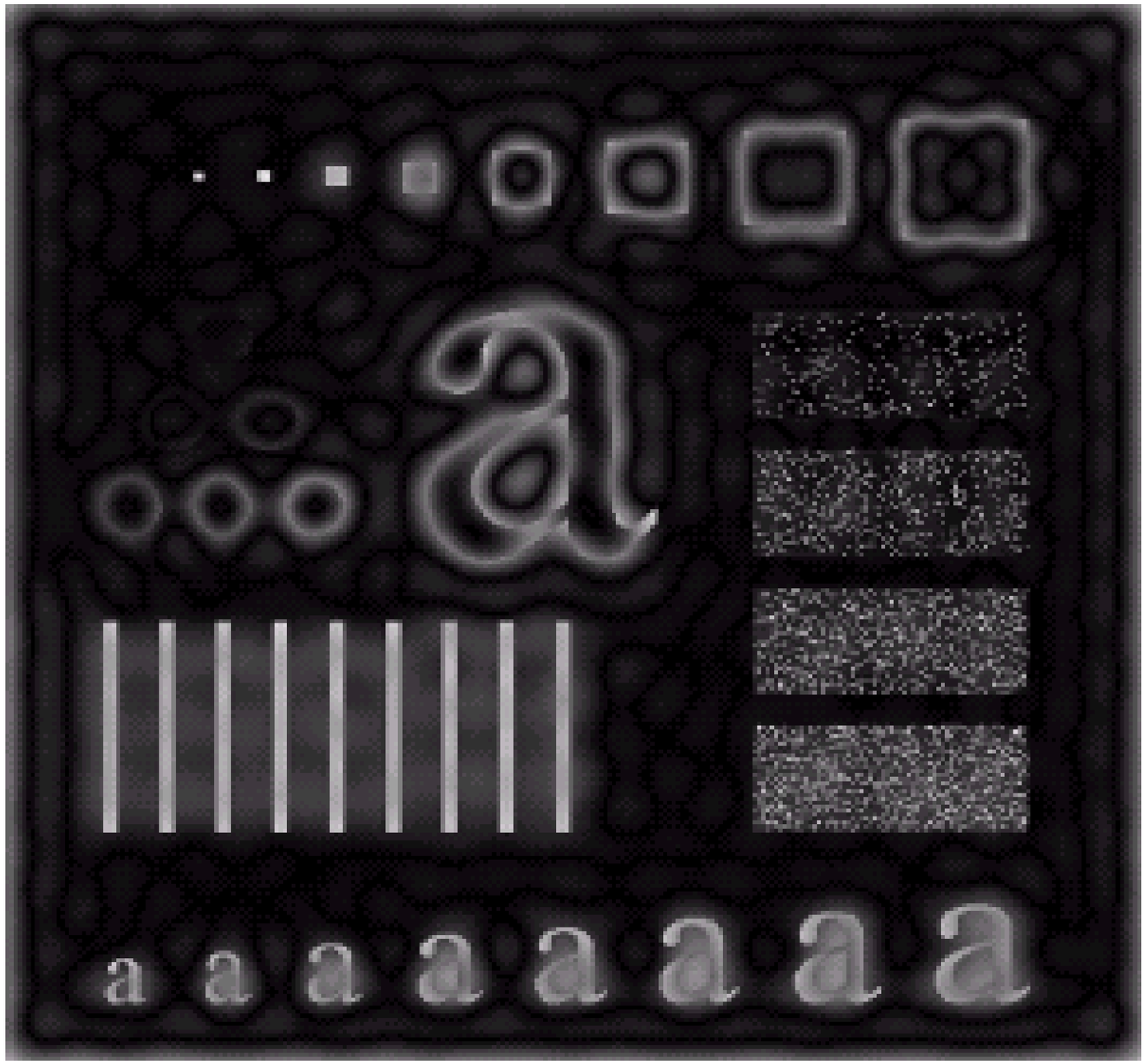


Results of Butterworth high pass filtering of order 2 with D_0 = 15



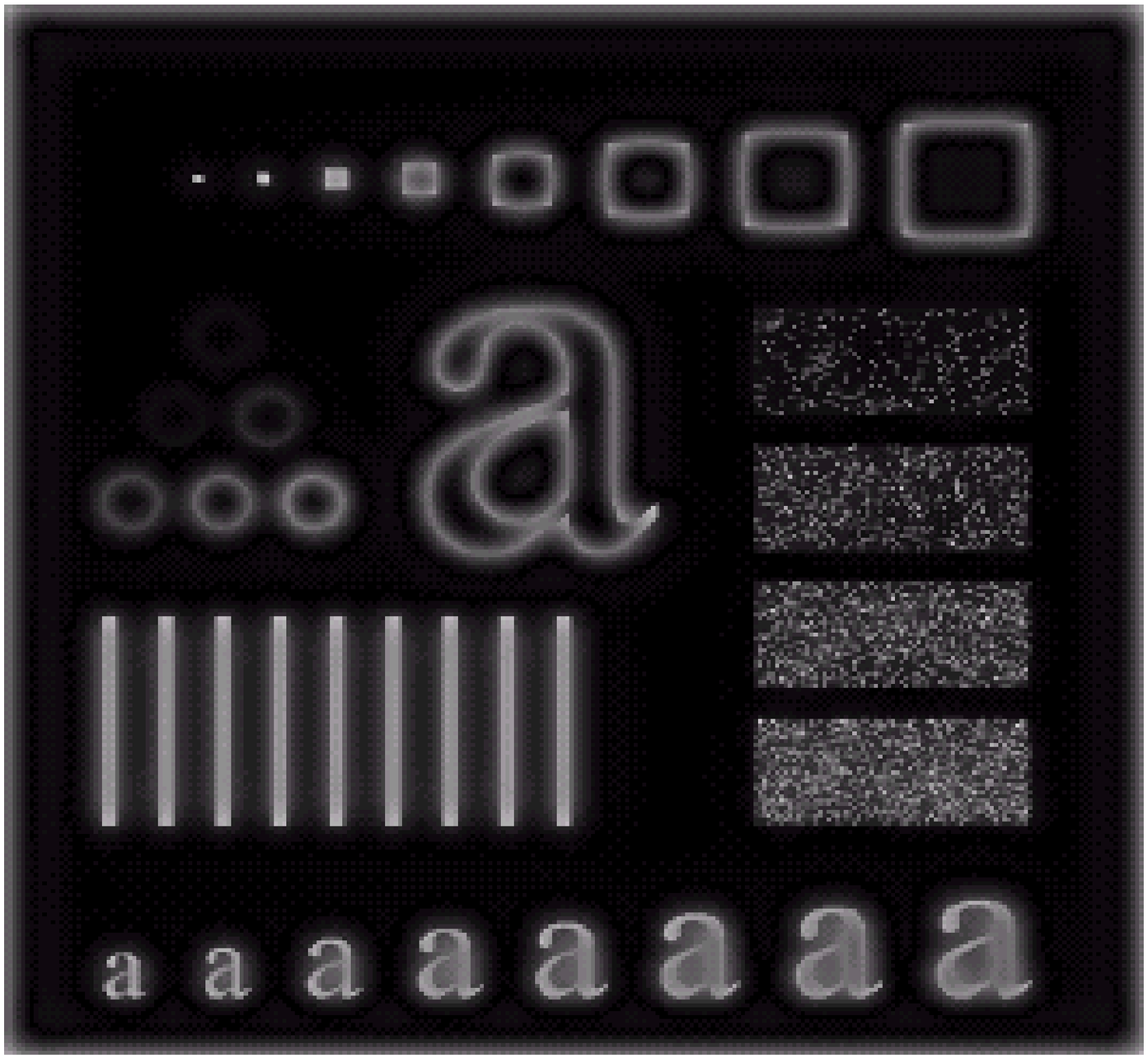
Results of Gaussian high pass filtering with D_0 = 15

Highpass Filter Comparison



Results of ideal high pass filtering with $D_0 = 15$

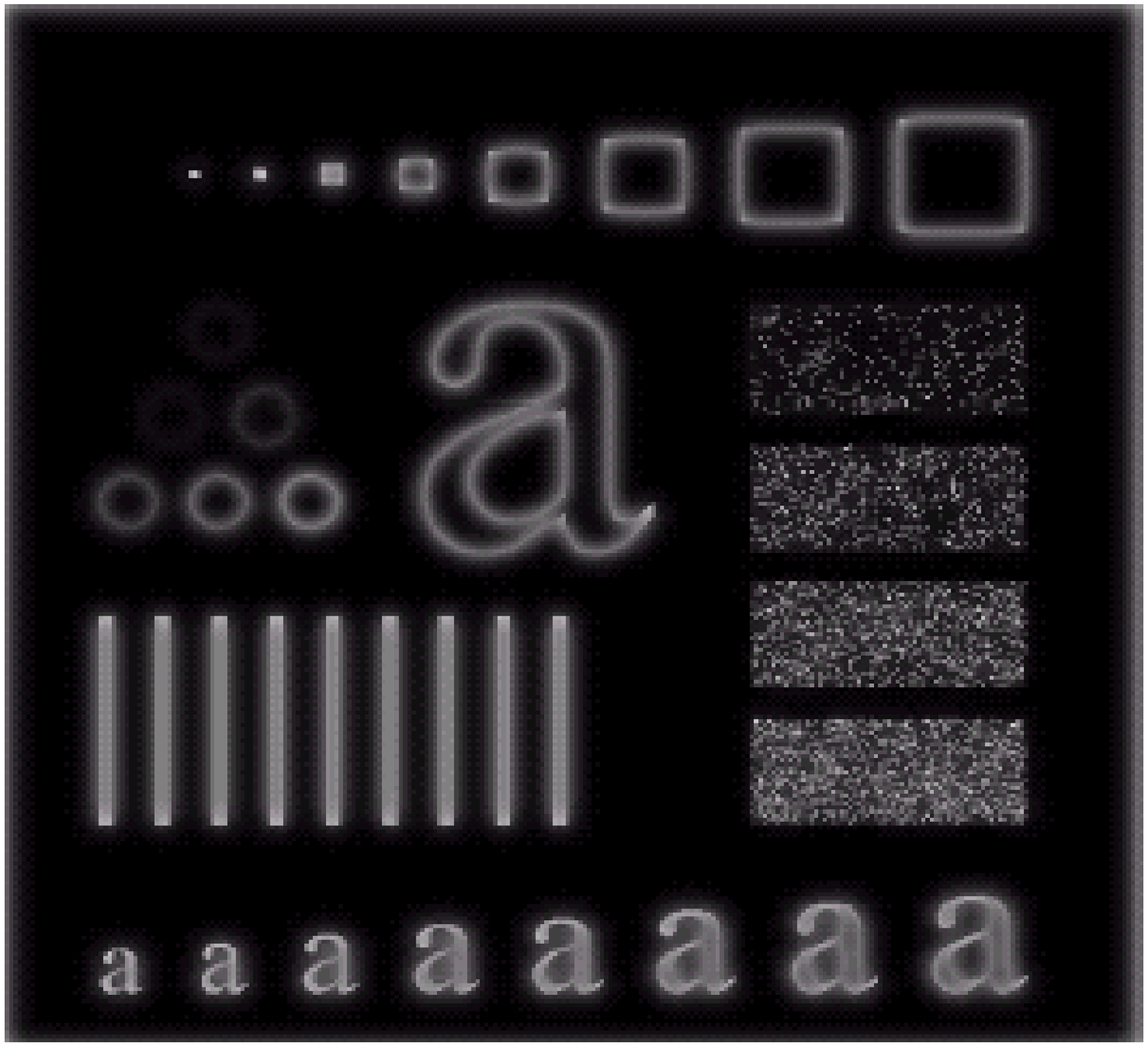
Highpass Filter Comparison



Results of Butterworth high pass filtering of order 2 with D_0 = 15

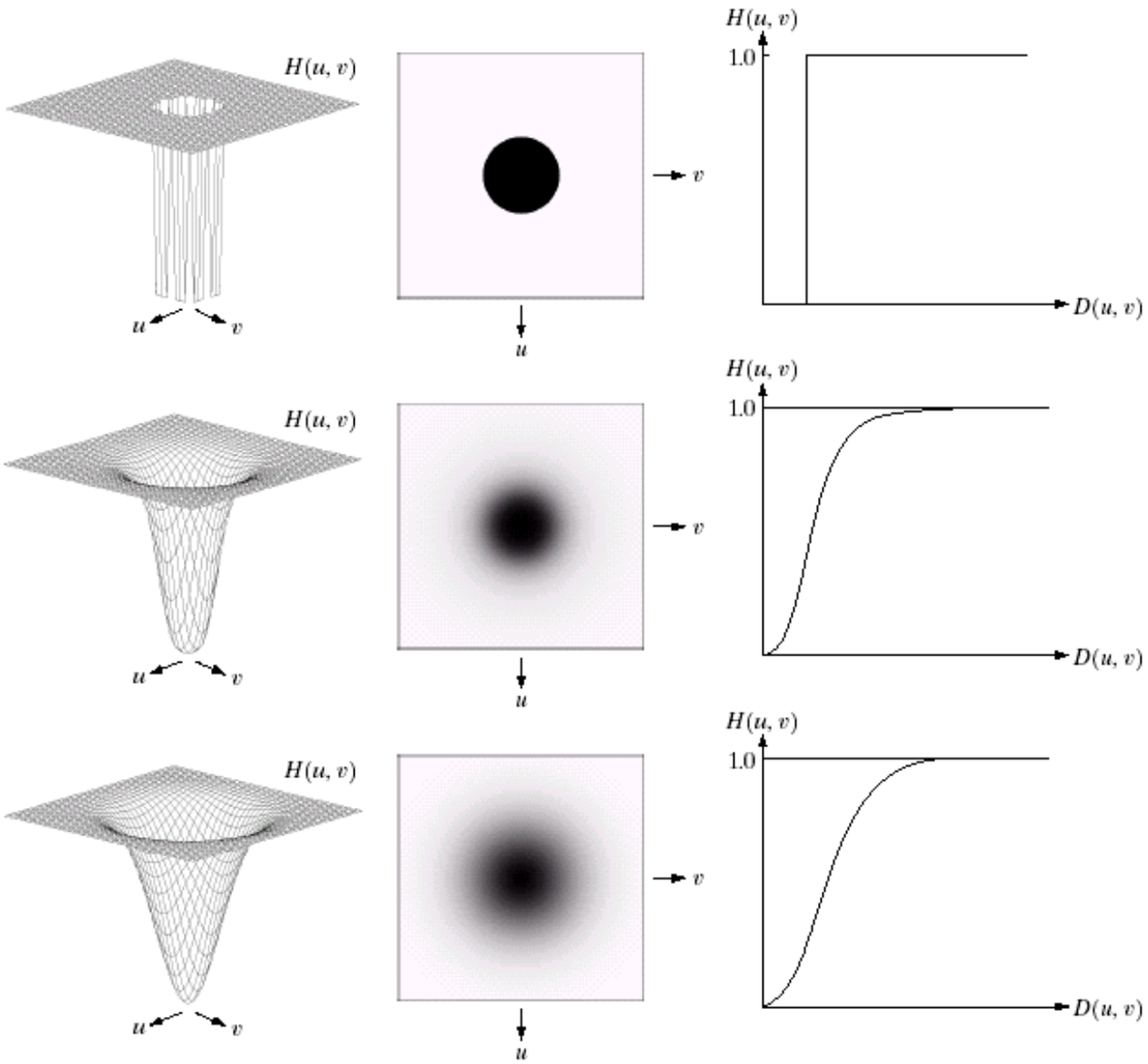
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Highpass Filter Comparison



Results of Gaussian high pass filtering with D_0 = 15

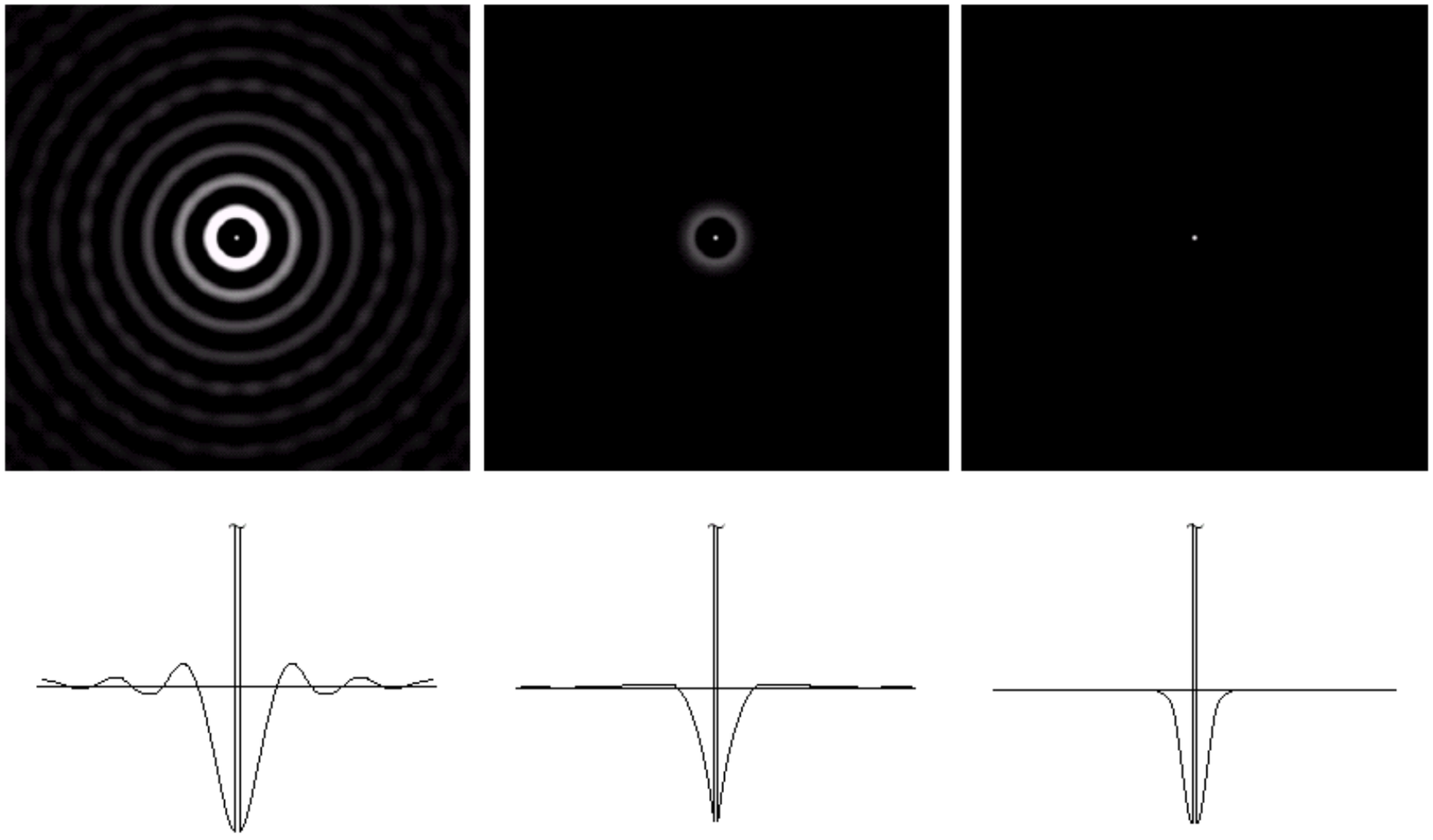
Highpass Filter Comparison



a b c
d e f
g h i

FIGURE 4.22 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Highpass Filter Comparison



a b c

FIGURE 4.23 Spatial representations of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding gray-level profiles.

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Highpass Filter Comparison

$$Unsharp\ Masking\ f_{hp}(x,y) = f(x,y) - f_{lp}(x,y) \tag{1}$$

$$High-boost\ filtering\ f_{hb}(x,y) = Af(x,y) - f_{lp}(x,y)\ where\ (A \geq 1) \tag{2}$$

$$f_{hb}(x,y) = (A-1)\,f(x,y) + f(x,y) - f_{lp}(x,y) \tag{3}$$

$$f_{hb}(x,y) = (A-1)\,f(x,y) + f_{hp}(x,y) \tag{4}$$

Highpass Filter Comparison

$$From\ (1)\ for\ Frequency\ Domain\ F_{hp}(u,v) = F(u,v) - F_{lp}(u,v) \quad (5)$$

$$F_{lp}(u,v) = H_{lp}(u,v)\,F(u,v) \quad (6)$$

$$Unsharp\ Masking\ H_{hp}(u,v) = 1 - H_{lp}(u,v) \quad (7)$$

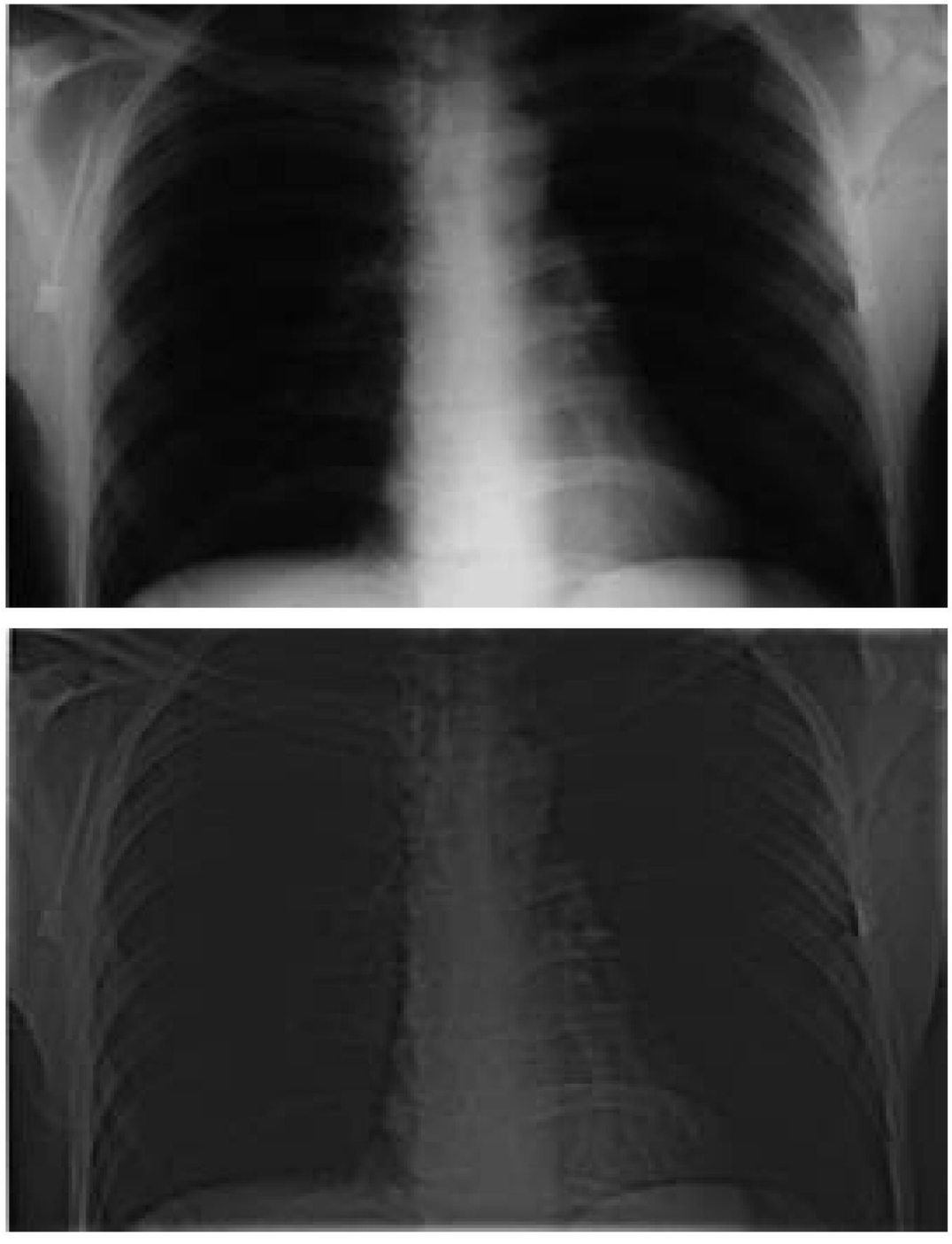
$$High-boost\ filtering\ H_{hb}(u,v) = (A-1) + H_{hp}(u,v) \quad (8)$$

$$High-frequency\ emphasis\ H_{hfe}(u,v) = a + bH_{hp}(u,v)\ \ where\ \ (a \geq 0,\ b > a) \quad (9)$$

Highpass Filter Comparison

The process consists of multiplying the composite filter by the (centered) transform of the input image and then taking the inverse transform of the product. Multiplication of the real part of this result by $(-1)^{x+y}$ gives us the high-boost filtered image $f_{hb}(x,y)$ in the spatial domain.

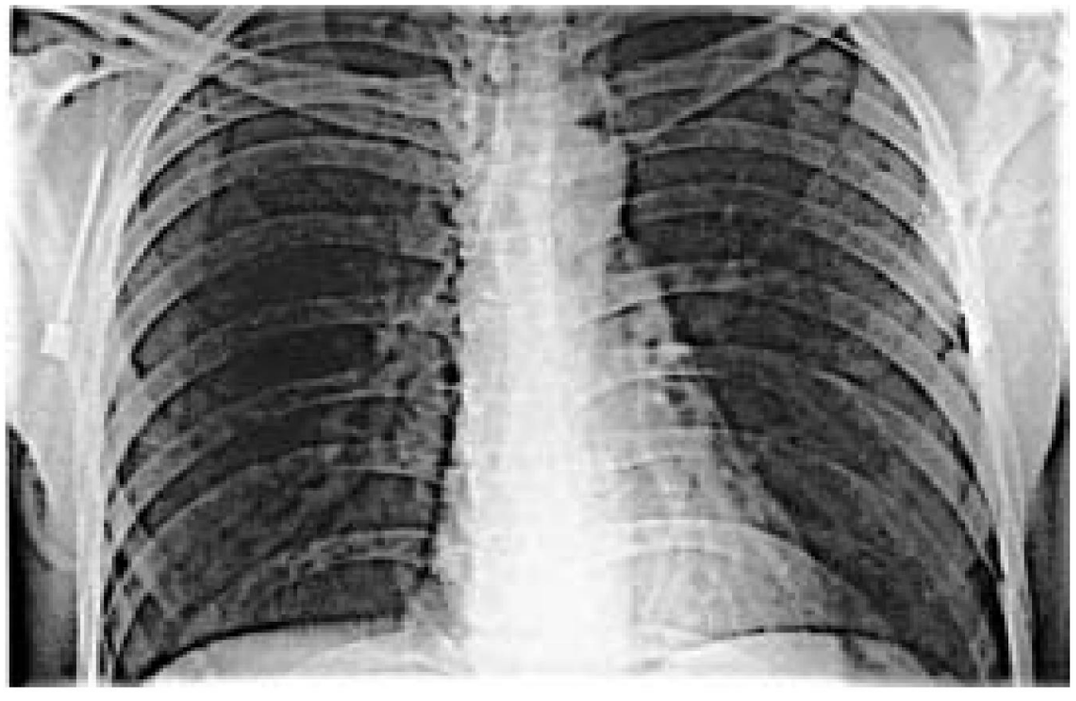
Highpass Filtering Example



Original image

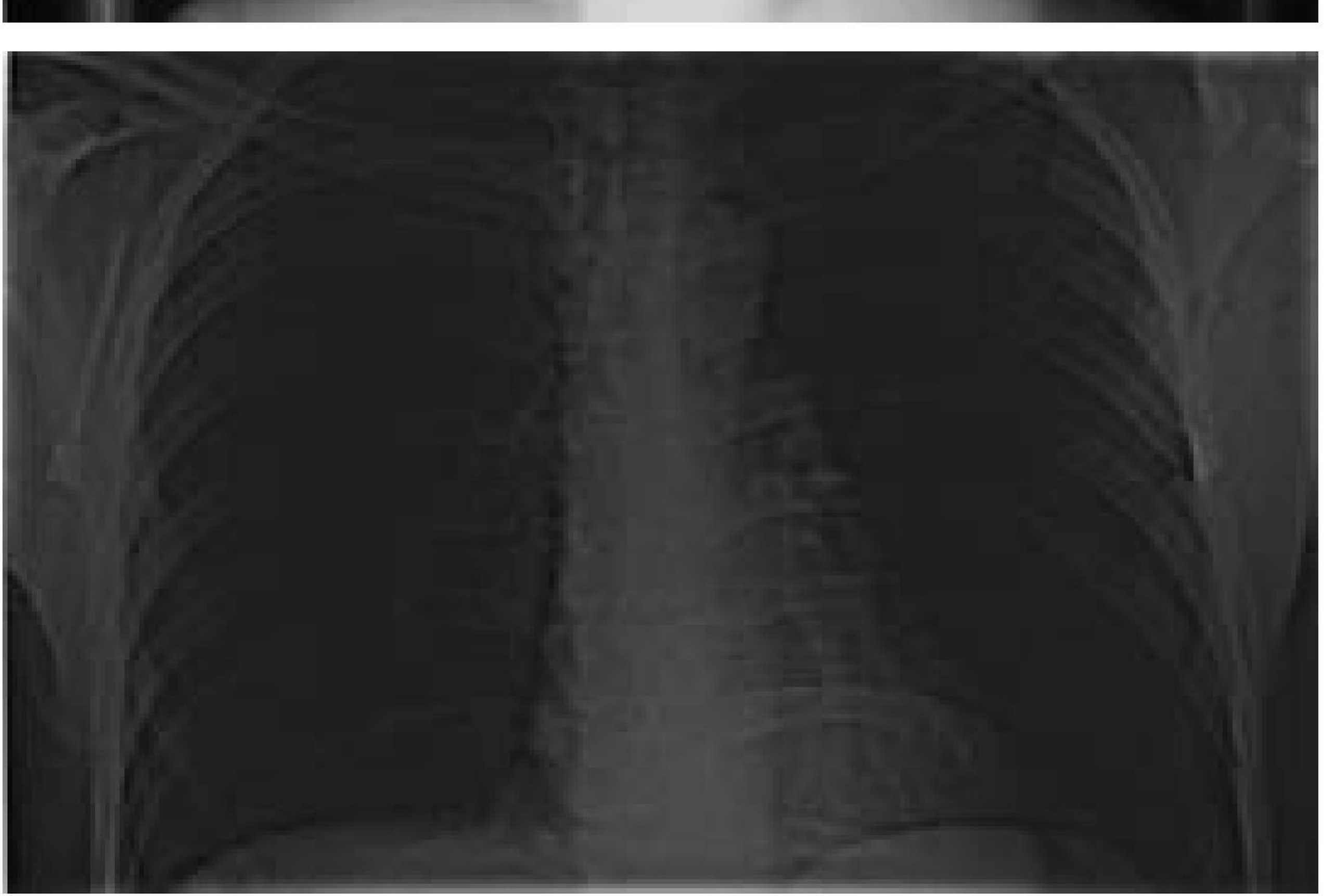
Highpass filtering result

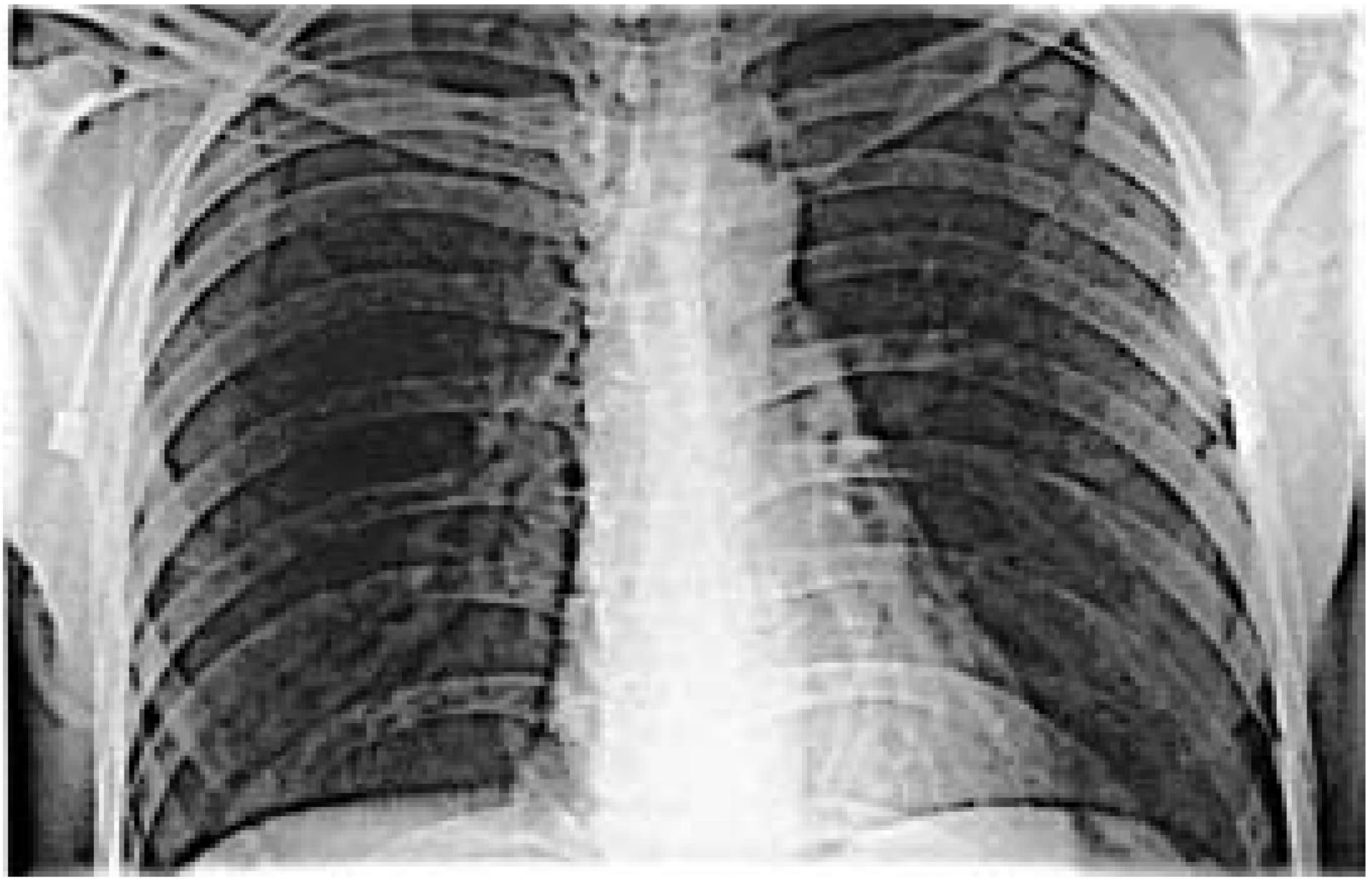
High frequency emphasis result



After histogram equalisation

Images taken from Gonzalez & Woods, Digital Image Processing (2002)





1-D Hadamard Transform

- **Hadamard Kernel**

$$g(x,u)=\frac{1}{N}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

Where $b_k(z)$ is the kth bit in the binary representation of z

- **Hadamard Transform**

$$H(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

- **$N=2^n$**
- **Inverse 1-D Hadamard transform**

$$f(x)=\sum_{x=0}^{N-1} H(u,v)(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

2-D Hadamard Transform

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)(-1)^{\sum_{i=0}^{n-1}(b_i(x)b_i(u)+b_i(x)b_i(u))}$$

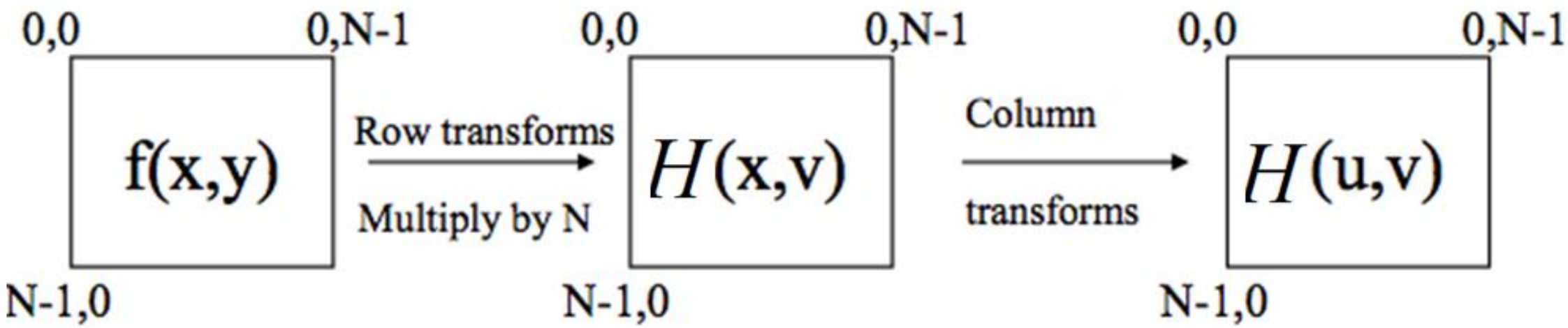
- **Inverse 2-D Hadamard transform**

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}H(u,v)(-1)^{\sum_{i=0}^{n-1}(b_i(x)b_i(u)+b_i(x)b_i(u))}$$

Properties of Hadamard Transform

- Hadamard kernel forms a matrix having orthogonal rows and columns ($A^{-1} = A^T$)
- Hadamard Transform are
  - Real, Symmetric, and orthogonal: $\mathbf{H}=\mathbf{H}^*=\mathbf{H}^T=\mathbf{H}^{-1}$
  - Ultra Fast Transform (± 1)
  - Good-Very Good energy compactness

- 2D Hadamard kernels are separable and symmetric



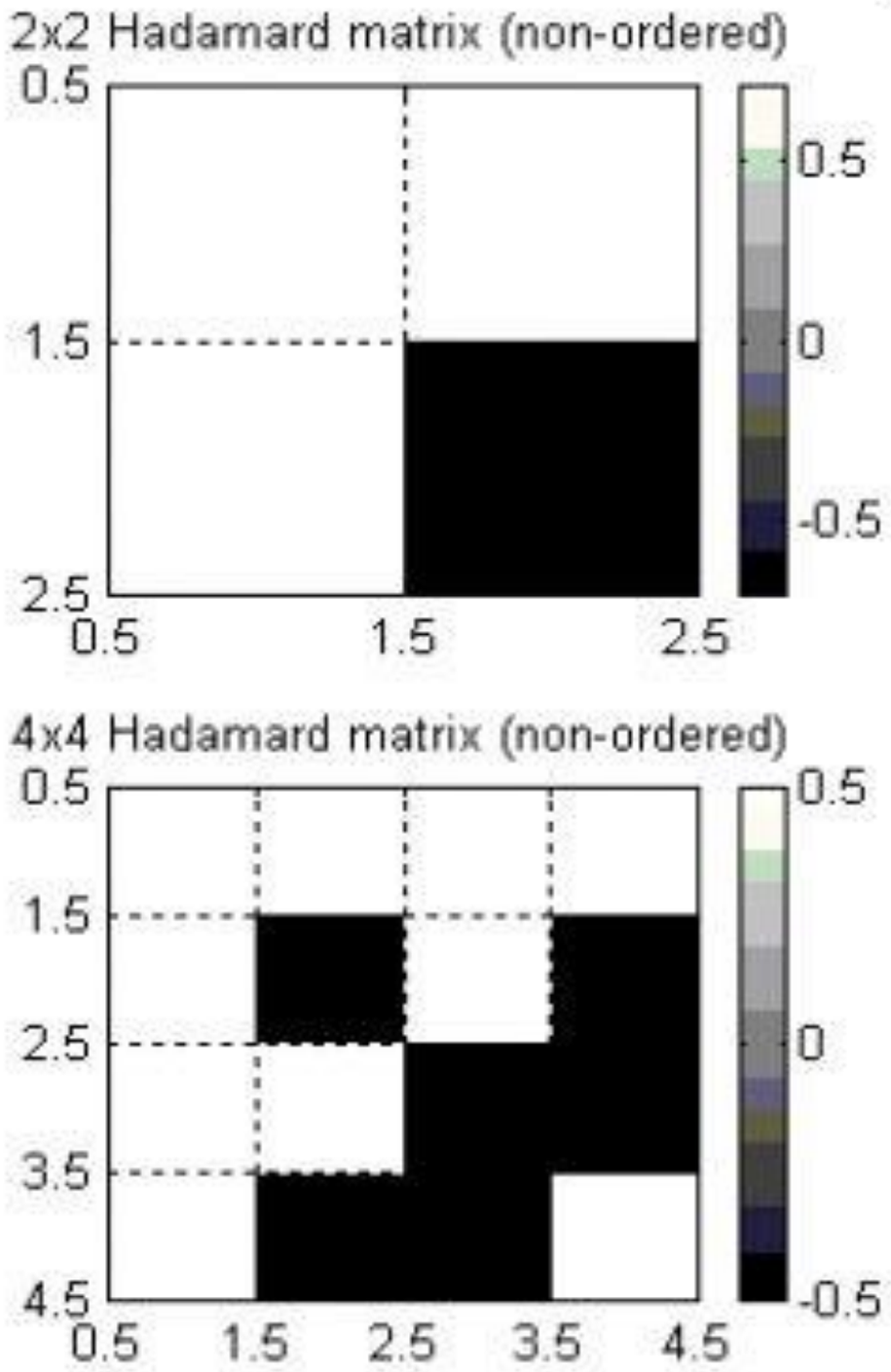
Recursive Relationship of the Hadamard Transform

- **An important property of Hadamard transform is that, letting H_N represent the Hadamard matrix of order N the recursive relationship holds :**

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

- **Therefore, starting from a small Hadamard matrixwe can compute a Hadamard matrix of any size.**
- **This is a good reason to use the Hadamard transform**

Image of 1-D Hadamard Matrices



Haar Transform

- Haar function is defined as

$$h_k(x), \quad x \in [0,1], \quad k = 0,\ldots,N-1$$

$$k = 2^p + q - 1$$

$$\begin{cases} 0 \le p \le n-1; q = 0,1 & p = 0 \\ 1 \le q \le 2^p & p \ne 0 \end{cases}$$

$$h_0(x) \triangleq h_{0,0}(x) = \frac{1}{\sqrt{N}}, \quad x \in [0,1]$$

$$h_k(x) \triangleq h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & \frac{q-1}{2^p} \le x \le \frac{q-1/2}{2^p} \\ -2^{p/2} & \frac{q-1/2}{2^p} \le x \le \frac{q}{2^p} \\ 0 & \text{O.W.} \end{cases}$$

$$x = \frac{m}{N}, \quad m = 0,1,\ldots,N-1$$

- Generates N*N matrix

Haar Transform

- With N=2 k=0,1

k	0	1
p	0	0
q	0	1

x=0/2,1/2

$h_0(x) = h_{0,0}(x) = 1/\sqrt{2}$

$h_1(x) = h_{0,1}(x) = 1/\sqrt{2}$

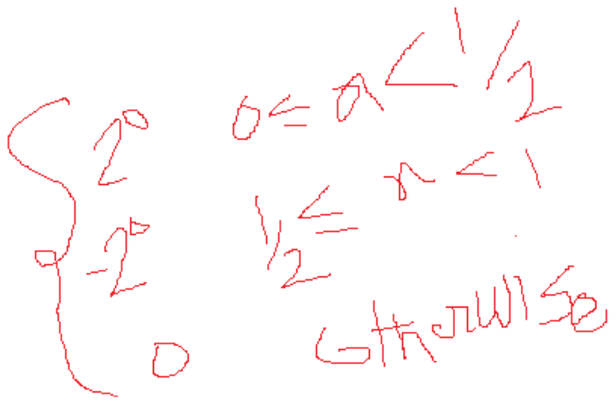
$h_1(0/2) = 1/\sqrt{2}$

$h_1(1/2) = -1/\sqrt{2}$

$$h_0(x) \triangleq h_{0,0}(x) = \frac{1}{\sqrt{N}}, \quad x \in [0,1]$$

$$h_k(x) \triangleq h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & \frac{q-1}{2^p} \le x \le \frac{q-1/2}{2^p} \\ -2^{p/2} & \frac{q-1/2}{2^p} \le x \le \frac{q}{2^p} \\ 0 & \text{O.W.} \end{cases}$$

$$x = \frac{m}{N}, \quad m = 0,1,\ldots,N-1$$



$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Haar Transform

- With N=2 k=0,1 ,2,3

k	0	1	2	3
p	0	0	1	1
q	0	1	1	2

x=0/4,1/4,2/4,3/4

$h_0(x) = h_{0,0}(x) = 1/\sqrt{2}$

$h_1(x) = h_{0,1}(x)$= criteria

$h_2(x) = h_{1,1}(x)$= criteria

$h_2(x) = h_{1,2}(x)$= criteria

$$h_0(x) \triangleq h_{0,0}(x) = \frac{1}{\sqrt{N}}, \quad x \in [0,1]$$

$$h_k(x) \triangleq h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2} & \frac{q-1}{2^p} \le x \le \frac{q-1/2}{2^p} \\ -2^{p/2} & \frac{q-1/2}{2^p} \le x \le \frac{q}{2^p} \\ 0 & \text{O.W.} \end{cases}$$

$$x = \frac{m}{N}, \quad m = 0,1,\ldots,N-1$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

Haar Transform

- With N=8 k=0,1……7
  - 1D Cases:

Sign change

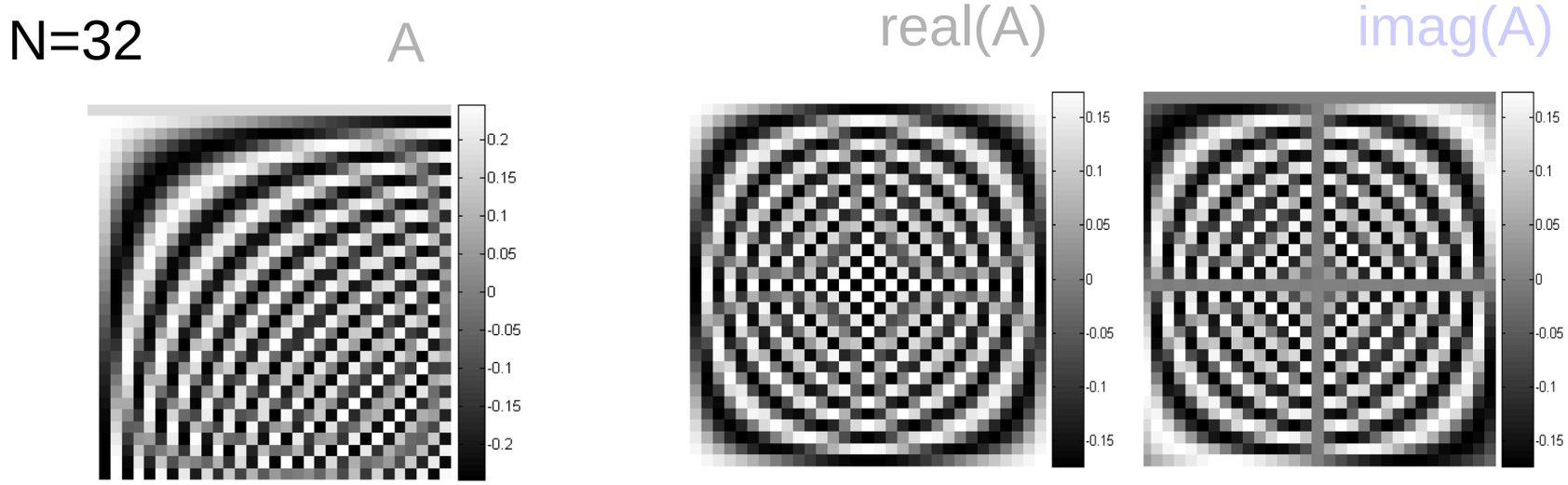
$$\mathbf{H}_r = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{matrix}$$

Properties of Haar Transform

- Real and Orthogonal: $\mathbf{Hr}=\mathbf{Hr}^{*}$, $\mathbf{Hr}^{-1}=\mathbf{Hr}^{T}$
- Fast Transform
- Poor energy compactness
- The basis vectrs of the haar matrix are sequency ordered

Discrete Cosine Transform (DCT)

- Discrete Cosine Transform (DCT) is a fourier related transform similar to DFT but it is only concerned with real numbers only
- inverse and forward transformation channel are identical.



- This is used in image compression

DFT vs. DCT

$$y = Ax$$

1D-DCT

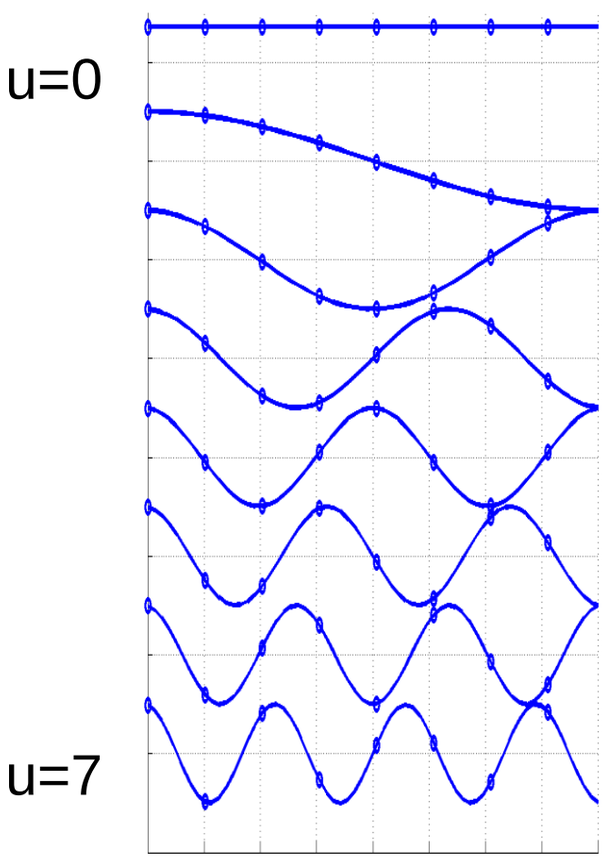
$$a(0,n) = \sqrt{\frac{1}{N}} \qquad u = 0$$

$$a(u,n) = \sqrt{\frac{2}{N}} cos\frac{\pi(2n+1)u}{2N} \qquad u = 1, 2, \ldots, N-1$$

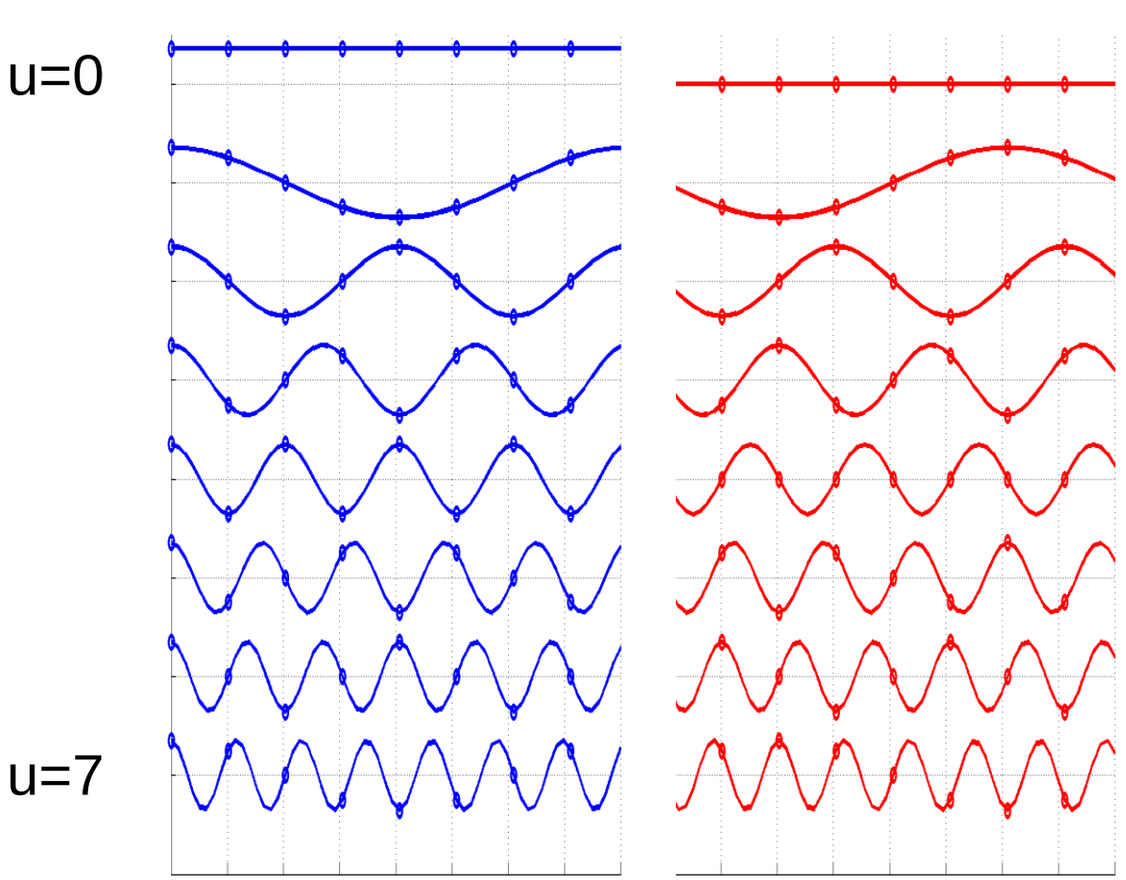
1D-DFT

$$a(u,n) = e^{-j2\pi\frac{un}{N}}$$
$$= cos(2\pi\frac{un}{N}) + jsin(2\pi\frac{un}{N})$$

a



real(a)



imag(a)

1-D Discrete Cosine Transform (DCT)

$$\begin{cases} Z(k) = \sum_{n=0}^{N-1} z(n) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \\ z(n) = \sum_{k=0}^{N-1} Z(k) \cdot \alpha(k) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \end{cases}$$

$$\alpha(0) = \frac{1}{\sqrt{N}}, \alpha(k) = \sqrt{\frac{2}{N}}$$

- Transform matrix *A*
  - a(k,n) = $\alpha(0)$ for k=0
  - a(k,n) = $\alpha(k)$ cos[$\pi(2n+1)/2N$] for k>0

- A is real and orthogonal
  - rows of A form orthonormal basis
  - A is not symmetric!
  - DCT is <u>not</u> the real part of unitary DFT!

2-D Discrete Cosine Transform (DCT)

Forward DCT:

$$C(u, v) = \alpha(u).\alpha(v) \sum_{x, y = 0}^{N-1} \sum \{f(x, y) \cos[(2x+1)u\pi/2N]. \cos[(2y+1)v\pi/2N]\}$$

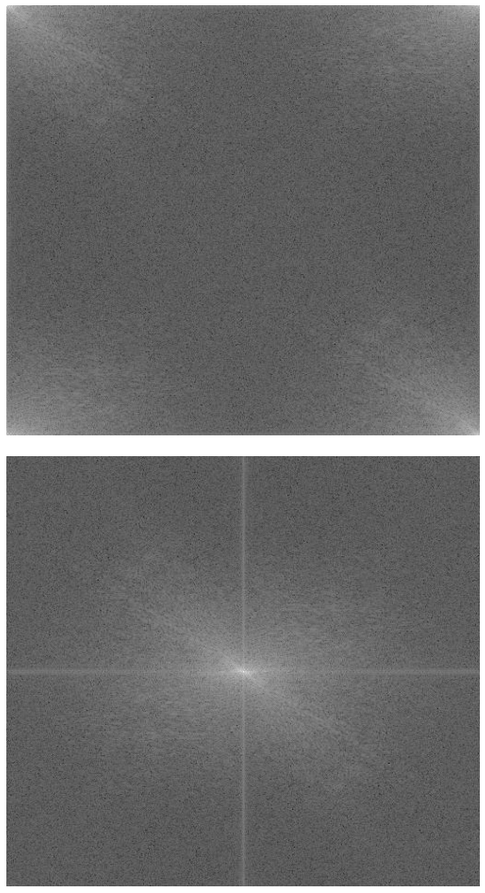
Reverse DCT:

$$f(x, y) = \sum_{u, v = 0}^{N-1} \sum \{\alpha(u).\alpha(v)\, C(u, v) \cos[(2x+1)u\pi/2N]. \cos[(2y+1)v\pi/2N]\}$$

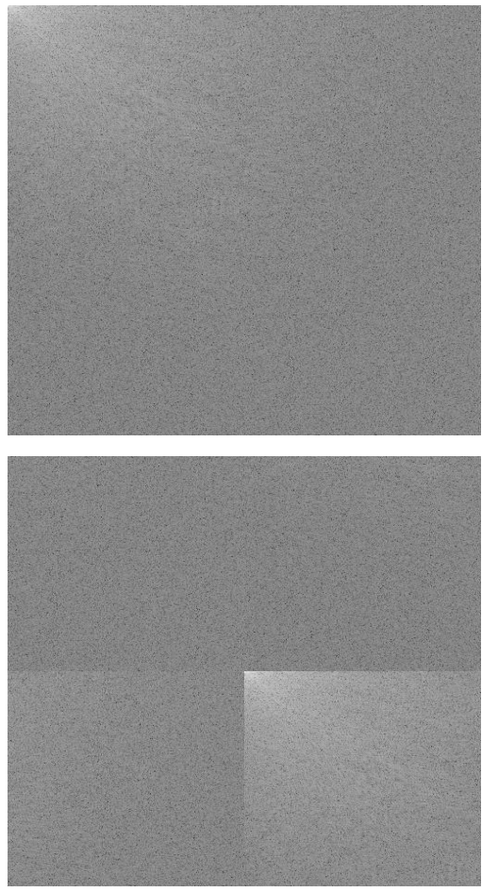
In DCT, energy is concentrated in a small region. It helps in image compression.

DFT and DCT on Lena

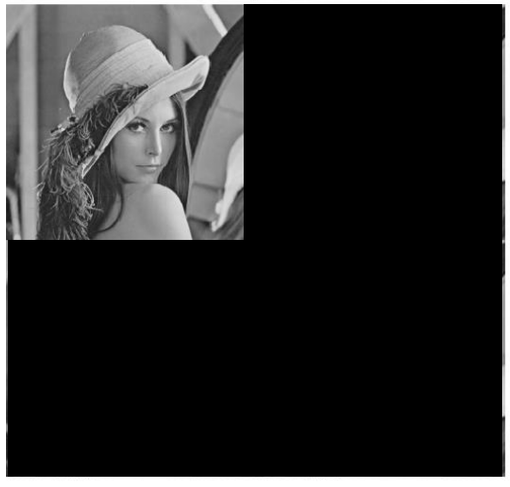
DFT2



DCT2



Shift low-freq
to the center



Assume periodic and zero-padded …

Assume reflection …

Properties of DCT

- Real and Orthogonal: $\mathbf{C}=\mathbf{C}^{*}$, $\mathbf{C}^{-1}=\mathbf{C}^{T}$
- Fast Transform
- It can be calculated in Nlog2N operations via N-point transform.

- It is good for energy compaction for highly correlated data.
- It is used for JPEG compression.

Fast Fourier Transform

The reason that Fourier based techniques have become so popular is the development of the *Fast Fourier Transform (FFT)* algorithm

Allows the Fourier transform to be carried out in a reasonable amount of time

Reduces the amount of time required to perform a Fourier transform by a factor of 100 – 600 times!

The Fast Fourier Transform (FFT)

Discrete Fourier Transform - Revisited

The Discrete Fourier Transform is

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x)\ e^{-i2\pi ux/M}$$

- This takes $O(M^2)$ to compute
- Can we do it faster?

The Fast Fourier Transform (FFT)

If we let

$$W_M = e^{-i2\pi/M}$$

the Discrete Fourier Transform can be written as

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x)\ W_M{}^{ux}$$

If M is a multiple of 2, $M = 2K$ for some positive number K.

Substituting $2K$ for M gives

$$F(u) = \frac{1}{2K} \sum_{x=0}^{2K-1} f(x)\ W_{2K}{}^{ux}$$

The Fast Fourier Transform (FFT)

Separating out the K even terms and the K odd terms,

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x)\,W_{2K}{}^{u(2x)} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)\,W_{2K}{}^{u(2x+1)}\right\}$$

Notice that

$$W_{2K}^{2u} = W_K^u$$

$$W_{2K}^{2(u+1)} = W_{2K}^{2u}\,W_{2K}^{u} = W_K^u W_{2K}^u$$

So,

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x)\,W_K^{ux} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)\,W_K^{ux}\,W_{2K}^{u}\right\}$$

The Fast Fourier Transform (FFT)

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x)W_K^{ux} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)W_K^{ux}W_{2K}^{u}\right\}$$

$$F(u) = \frac{1}{2}\left\{F_{\text{even}}(u) + F_{\text{odd}}(u)W_{2K}^{u}\right\}$$

Use this to get the first K terms ($u = 0..K-1$), then re-use these parts to get the last K terms ($u = K..2K-1$):

$$F(u+K) = \frac{1}{2}\left\{F_{\text{even}}(u) - F_{\text{odd}}(u)W_{2K}^{u}\right\}$$

Only have to do $u = 0..M/2-1$, $x = 0..M/2-1$ and a little extra math.

$$To\ derive\ F(u+K) \quad use \quad W_K^{u+K} = W_K^u \ and\ W_{2K}^{u+K} = -W_{2K}^u$$

The Fast Fourier Transform (FFT)

Computational Complexity:

Discrete Fourier Transform	$O(N^2)$
Fast Fourier Transform	$O(N \log N)$

Remember:
The Fast Fourier Transform is just a faster *algorithm* for computing the Discrete Fourier Transform—it is *not* any different in result.

The Fast Fourier Transform (FFT)

What About 2-D?

Remember, we can use separability of the Fourier Transform to break a 2-D transform into $2N$ 1-D transforms:

DFT	$O(N^4)$
DFT using separability	$O(N^3)$
FFT using separability	$O(N^2 \log N)$

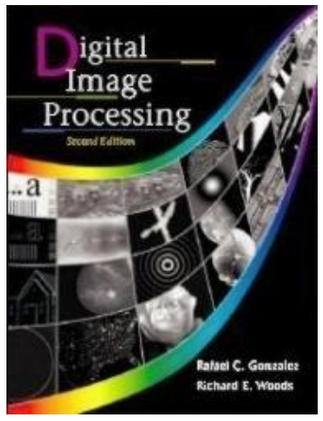
Frequency Domain Filtering & Spatial Domain Filtering

Similar jobs can be done in the spatial and frequency domains

Filtering in the spatial domain can be easier to understand

Filtering in the frequency domain can be much faster – especially for large images

References



"Digital Image Processing", Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002

–Much of the material that follows is taken from this book

–Image Processing and Pattern Recognition Slides of Dr. Sanjeeb Prasad Panday

Thank you !!!