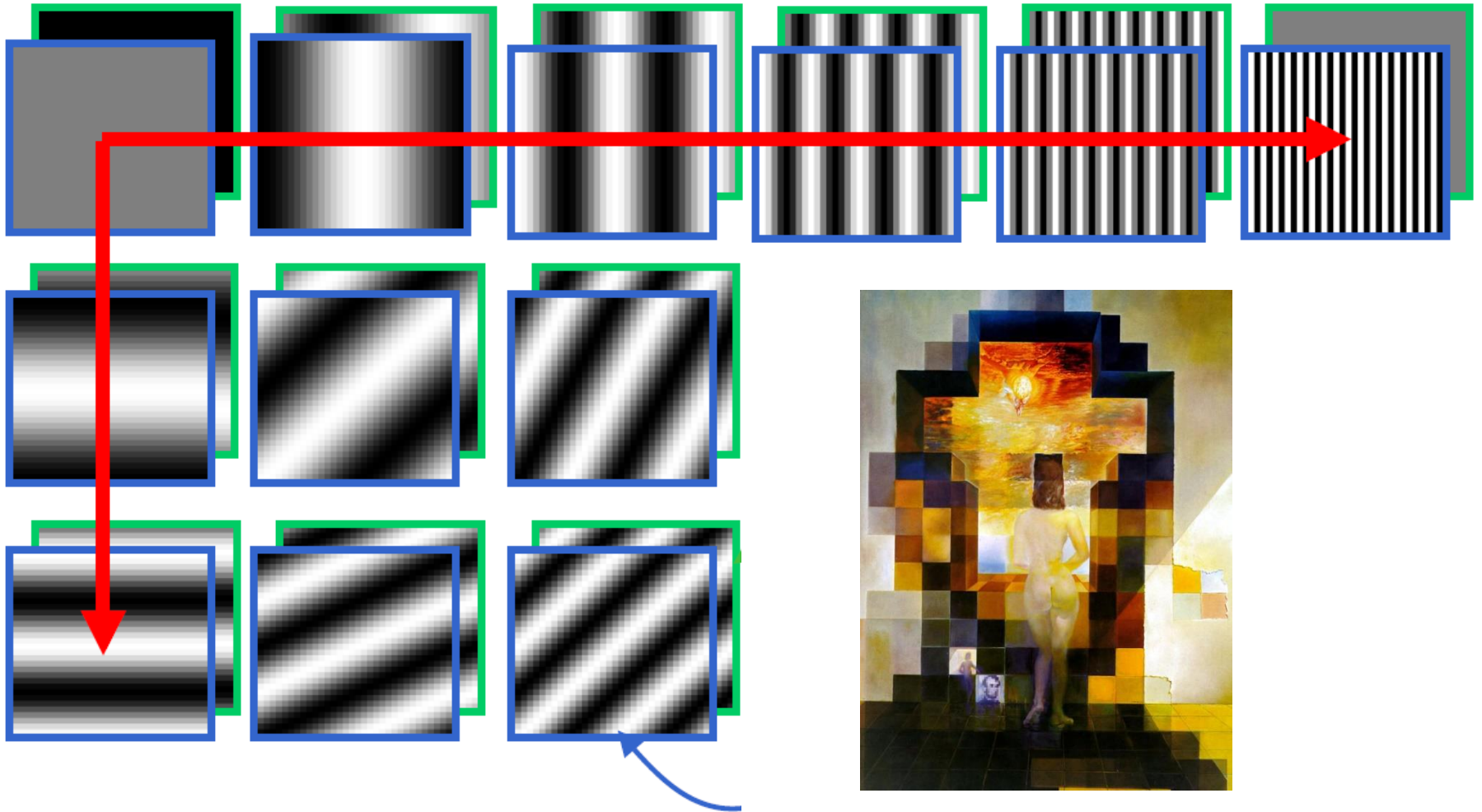


# Frequency Domain Processing

Dr. Mongkol Ekpanyapong

# A nice set of basis

Teases away fast vs. slow changes in the image.



This change of basis has a special name...

# Jean Baptiste Joseph Fourier (1768-1830)

- Had crazy idea (1807):
  - **Any** periodic function can be rewritten as a weighted sum of sines and cosines of different frequencies.
- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's true!
  - Called Fourier Series



# Joseph Fourier



Joseph's father was a tailor in Auxerre  
Joseph was the ninth of twelve children  
His mother died when he was nine and  
his father died the following year

Fourier demonstrated talent on math  
at the age of 14.

In 1787 Fourier decided to train for  
the priesthood - a religious life or a  
mathematical life?

In 1793, Fourier joined the local  
Revolutionary Committee

Born: 21 March 1768 in Auxerre, Bourgogne, France  
Died: 16 May 1830 in Paris, France

# Fourier's "Controversy" Work

- Fourier did his important mathematical work on the theory of heat (highly regarded memoir *On the Propagation of Heat in Solid Bodies* ) from 1804 to 1807
- This memoir received objection from Fourier's mentors (Laplace and Lagrange) and not able to be published until 1815

Napoleon awarded him a pension of 6000 francs, payable from 1 July, 1815. However Napoleon was defeated on 1 July and Fourier did not receive any money



# Fourier Domain

- Expresses an image as the sum of weighted sinusoids
  - Wavelengths are determined by image dimensions
  - Amplitudes are determined by sample values
- Fourier coefficients are **complex** rather than real values
- Given a one-dimensional sequence  $f$  of  $N$  samples, the one dimensional discrete Fourier transform is given as

$$\mathcal{F}(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \left[ \cos\left(\frac{2\pi ux}{N}\right) - j \sin\left(\frac{2\pi ux}{N}\right) \right]$$

- $N$  is the length of a row and hence  $u$  is in  $[0, N-1]$

$$\mathcal{F}(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \left[ \cos\left(\frac{2\pi ux}{N}\right) - j \sin\left(\frac{2\pi ux}{N}\right) \right]$$



# Complex complexities

- The symbol  $j$  denotes the imaginary unit  
 $-j$  satisfies the relation  $j^2 = -1$

$$\cos(x) - j \sin(x) = e^{-jx}, \quad (9.11)$$

- Usually written more compactly by using Euler's formula

$$\mathcal{F}(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi ux/N}. \quad (9.12)$$



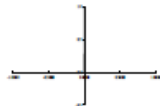

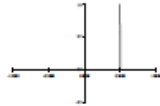
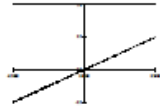
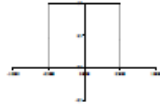
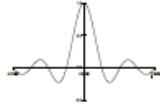

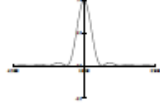
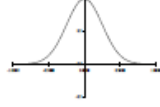

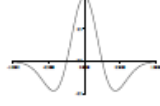

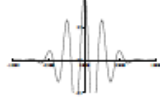
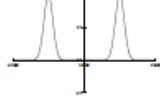


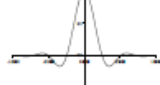

# Forward and Inverse

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

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



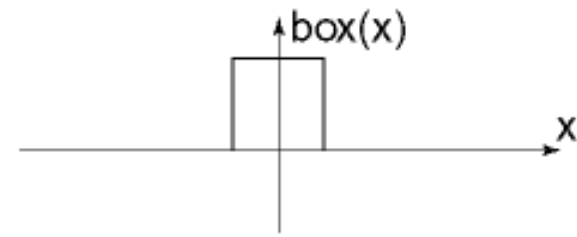
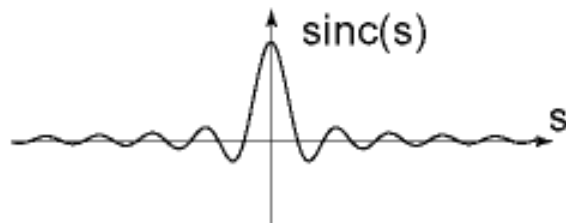
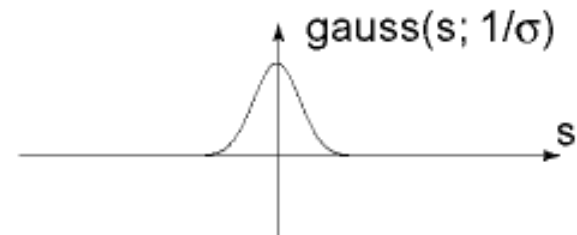
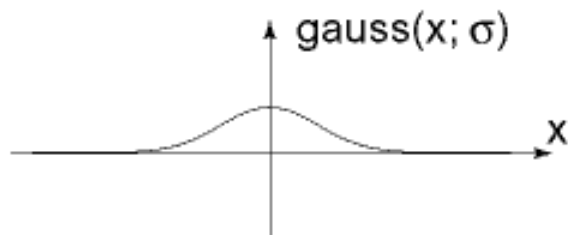
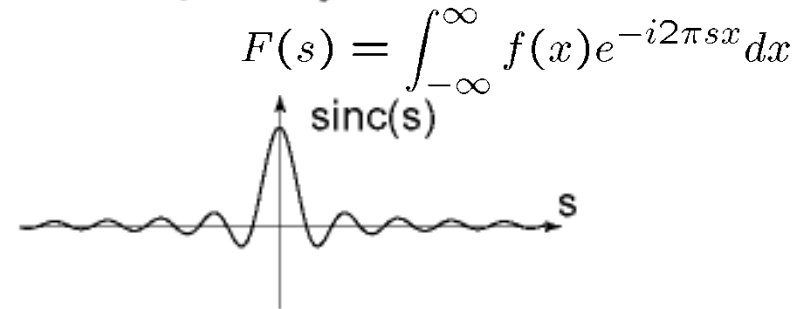
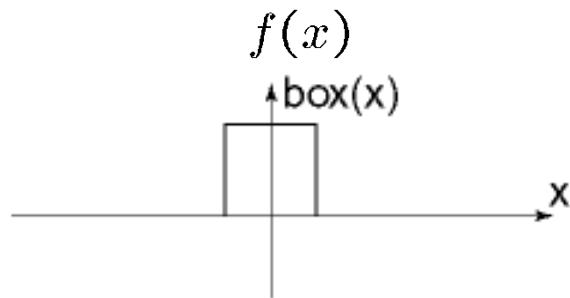
# Fourier Pairs (from Szeliski)

Name	Signal	Signal	Transform	Transform	
impulse		$\delta(x)$	$\Leftrightarrow$	1	
shifted impulse		$\delta(x - u)$	$\Leftrightarrow$	$e^{-j\omega u}$	
box filter		$\text{box}(x/a)$	$\Leftrightarrow$	$a\text{sinc}(a\omega)$	
tent		$\text{tent}(x/a)$	$\Leftrightarrow$	$a\text{sinc}^2(a\omega)$	
Gaussian		$G(x; \sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma} G(\omega; \sigma^{-1})$	
Laplacian of Gaussian		$(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})G(x; \sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma} \omega^2 G(\omega; \sigma^{-1})$	
Gabor		$\cos(\omega_0 x)G(x; \sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma} G(\omega \pm \omega_0; \sigma^{-1})$	
unsharp mask		$(1 + \gamma)\delta(x) - \gamma G(x; \sigma)$	$\Leftrightarrow$	$(1 + \gamma) - \frac{\sqrt{2\pi}\gamma}{\sigma} G(\omega; \sigma^{-1})$	
windowed sinc		$\text{rcos}(x/(aW)) \text{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29)	

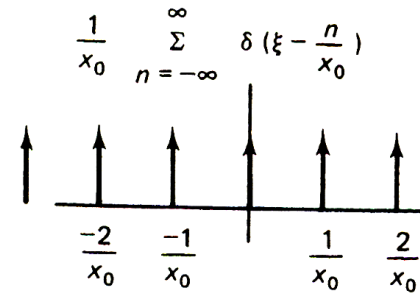
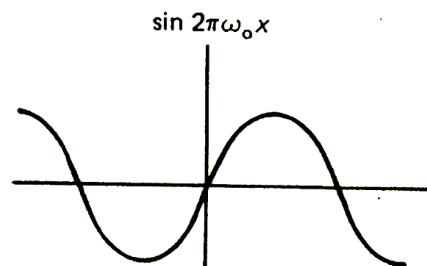
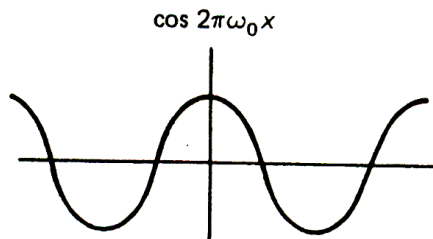
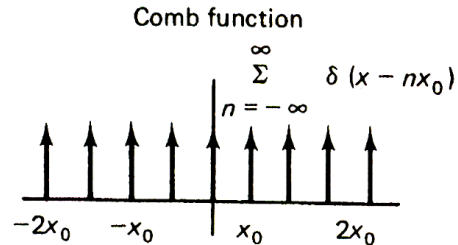
# Fourier Transform smoothing pairs

Spatial domain

Frequency domain



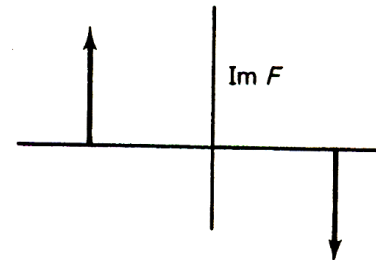
# Fourier Transform Sampling Pairs



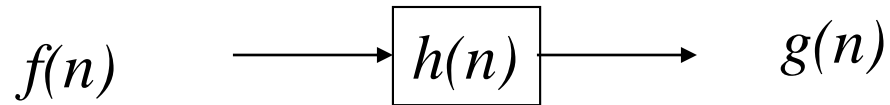
$$\frac{1}{2} [\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$$



$$\frac{1}{2} j [-\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$$



# 1D Linear Filtering



$$g(n) = \sum_{k=-\infty}^{\infty} h(k) f(n-k) = h(n) \otimes f(n) = f(n) \otimes h(n)$$

See review section

Linear convolution

- Linearity  $a_1 f_1(n) + a_2 f_2(n) \rightarrow a_1 g_1(n) + a_2 g_2(n)$
- Time-invariant property  $f(n - n_0) \rightarrow g(n - n_0)$

# Filter Examples

Low-pass (LP)

$$h(n)=[1,1]$$



$$|H(w)|=2\cos(w/2)$$

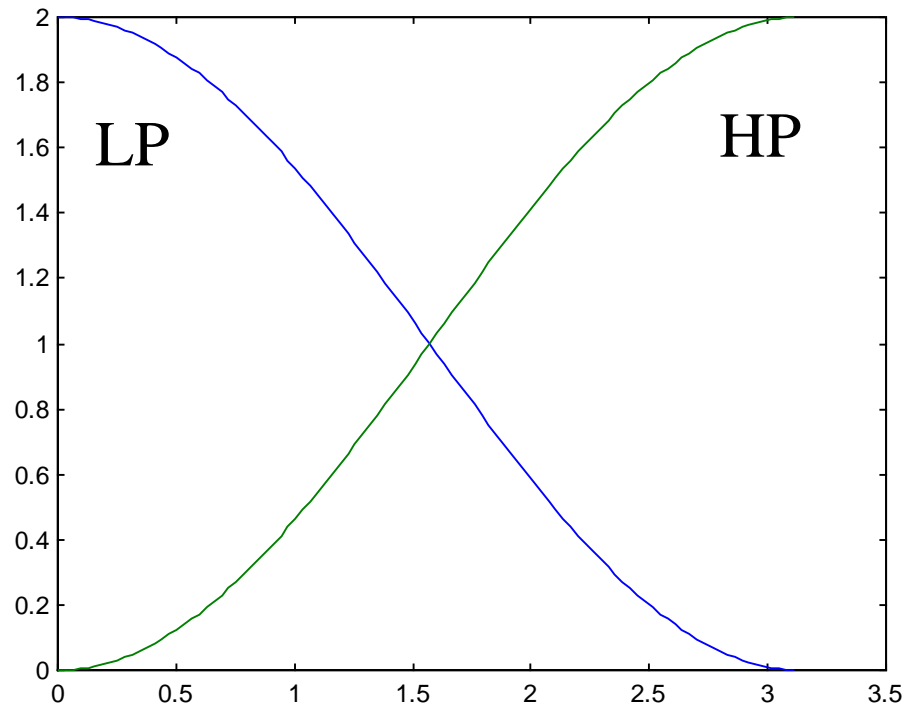
High-pass (LP)

$$h(n)=[1,-1]$$



$$|H(w)|=2\sin(w/2)$$

$|H(w)|$



$w$

# 2D DFT

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



# 2D DFT

- Each DFT coefficient is a complex value
  - There is a single DFT coefficient for each spatial sample
  - A complex value is expressed by two real values in either Cartesian or polar coordinate space.
    - Cartesian:  $R(u,v)$  is the *real* and  $I(u, v)$  the *imaginary* component
    - Polar:  $|F(u,v)|$  is the *magnitude* and  $\phi(u,v)$  the *phase*

$$\mathcal{F}(u, v) = R(u, v) + jI(u, v)$$

$$\mathcal{F}(u, v) = |F(u, v)|e^{j\phi(u, v)}$$

# 2D DFT

- Representing the DFT coefficients as magnitude and phase is a more useful for processing and reasoning.
  - The magnitude is a measure of strength or length
  - The phase is a direction and lies in  $[-\pi, +\pi]$
- The magnitude and phase are easily obtained from the real and imaginary values

$$|\mathcal{F}(u, v)| = \sqrt{R^2(u, v) + I^2(u, v)}$$
$$\phi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right].$$

# Magnitude Spectrum and Phase Spectrum

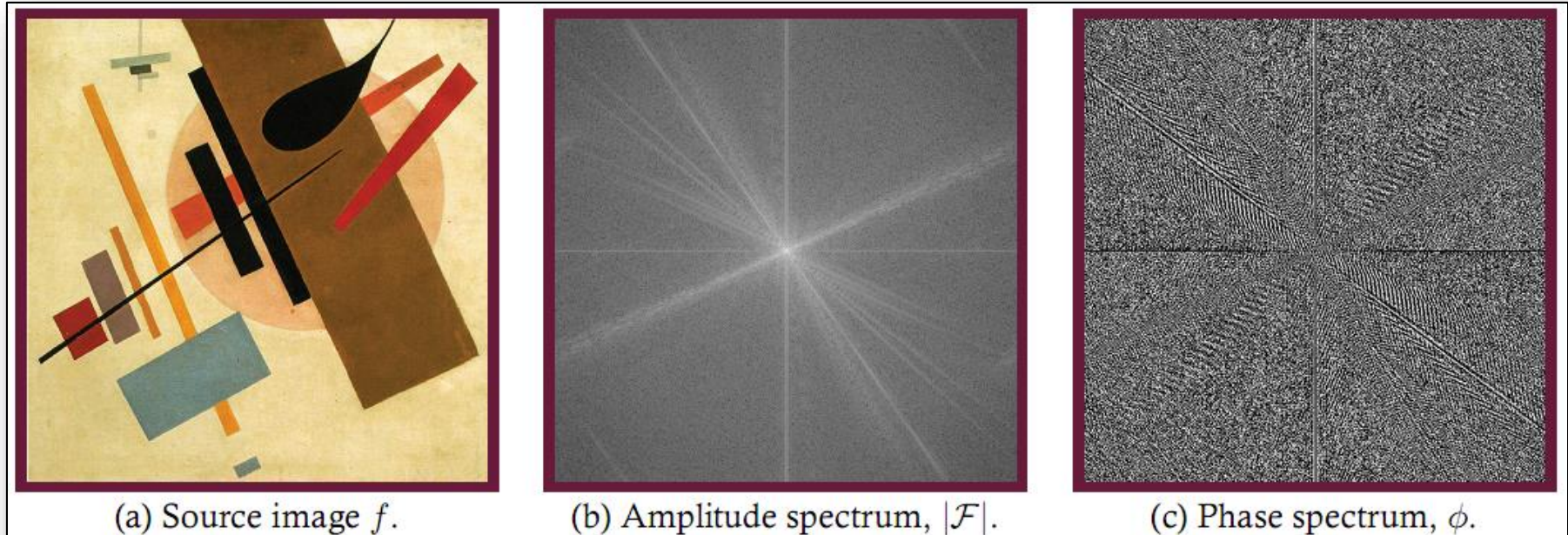


Figure 9.7. DFT Spectrum.

A decorative image in the top-left corner consisting of a blue square above a grid of smaller squares in various colors.

# Magnitude Spectrum and Phase Spectrum

- Notes on the magnitude spectrum:
  - Magnitudes are generally referred to as the “spectrum” but this should be understood as the magnitude spectrum.
  - Typically has an extremely large dynamic range and it is typical to log-compress those values for display (as in the previous slide)
  - For presentation, the DC component,  $F(0,0)$ , is placed at the center. Low frequency components are shown near the center and frequency increases with distance from center.



# Magnitude Spectrum and Phase Spectrum

- The magnitude spectrum contains information about the shape of objects. A strong edge in the source will generate a strong edge in the magnitude spectrum (rotated 90 degrees)
- The phase spectrum contains information about their actual location in the source. An image of lots of 'Q's will have the same magnitude spectra but not the same phase spectra.

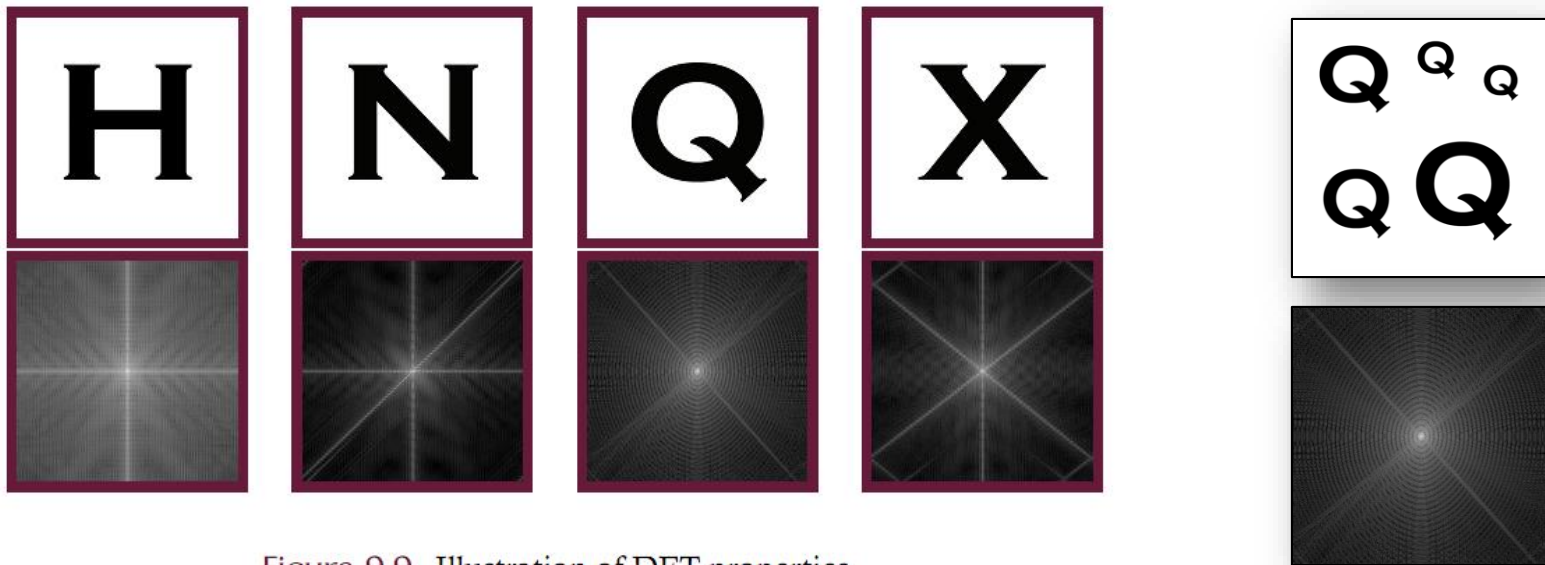
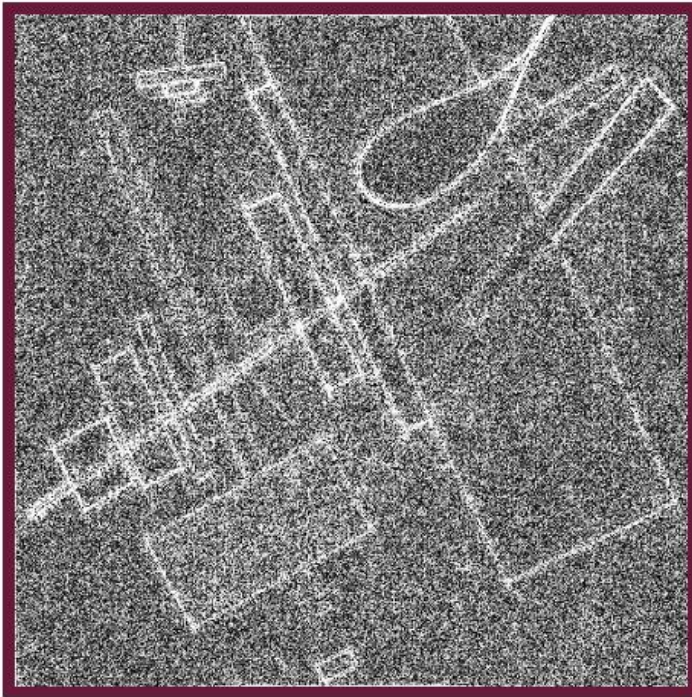


Figure 9.9. Illustration of DFT properties.



# Magnitude Spectrum and Phase Spectrum



(a) Reconstructed from phase information only.



(b) Reconstruction from amplitude information only.

Figure 9.8. Comparison of the contribution of the amplitude and phase spectrum.



# DFT Example

- Given a row profile, compute the Fourier coefficients

Index	0	1	2	3	4	5	6	7
Value	20	12	18	56	83	10	104	114

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

# Translation, Rotation, Distributivity

- **Translation** of the source will cause the phase spectrum to change but leave the magnitude spectrum unchanged since the phase spectrum encodes location information while the magnitude spectrum encodes shape information.
- **Rotation** of the source corresponds to an identical rotation of the magnitude and phase spectra.
- **Distributivity.** The Fourier transform is distributive over addition (not multiplication):

$$\vec{\mathcal{F}}(f + g) = \vec{\mathcal{F}}(f) + \vec{\mathcal{F}}(g)$$

# Translation, Rotation, Distributivity

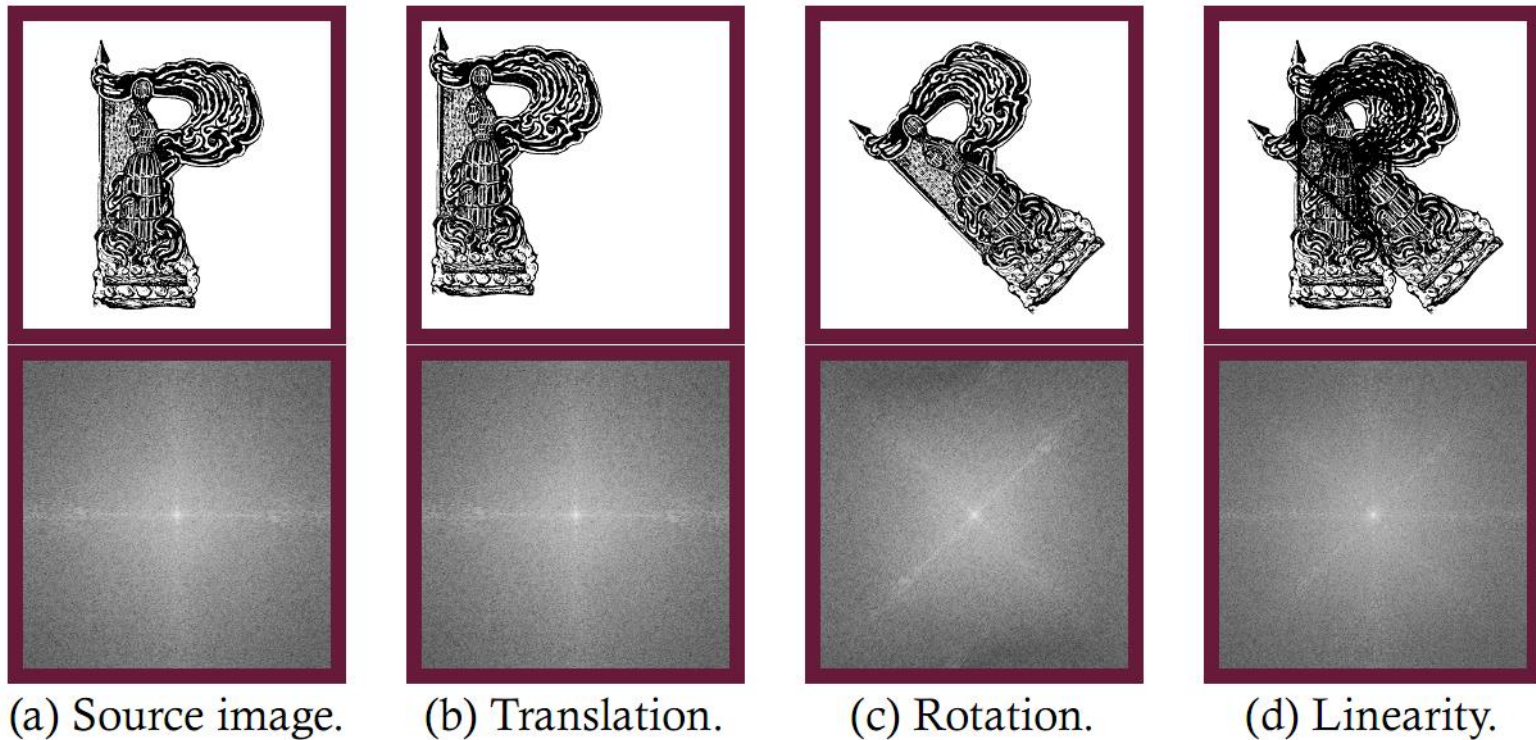
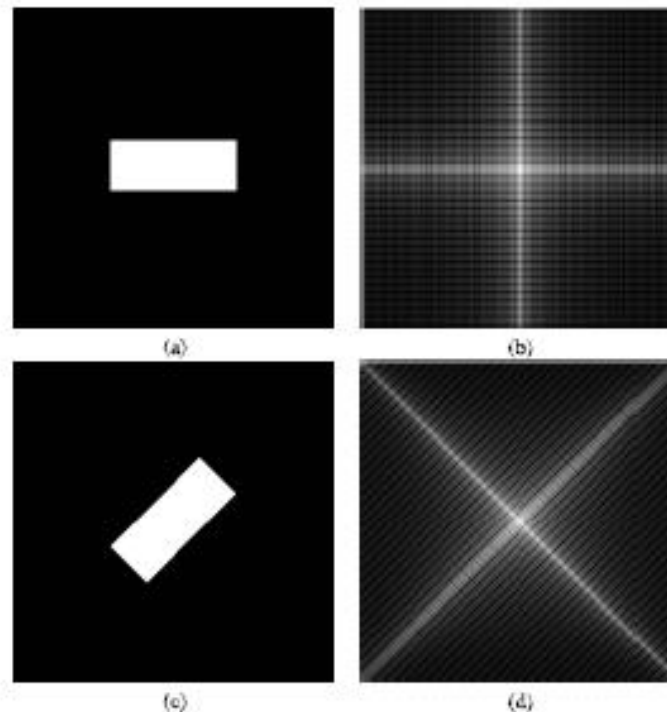


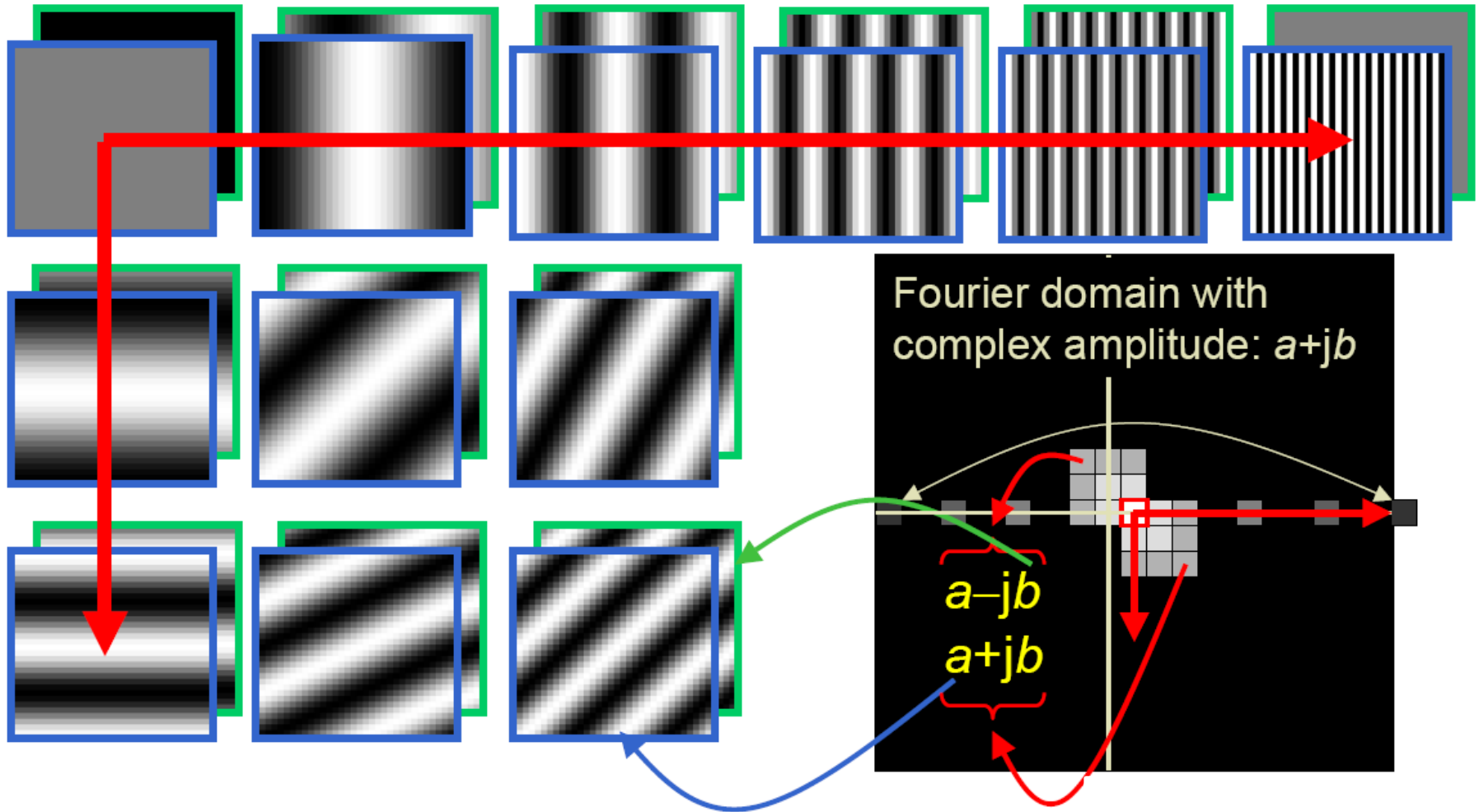
Figure 9.10. Properties of the DFT under translation, rotation, and linear combination.

# Fourier Transform Properties

- Rotation: if an image is rotated by a certain angle, its 2D FT will be rotated by the same angle.



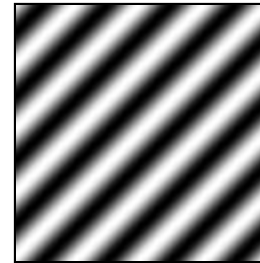
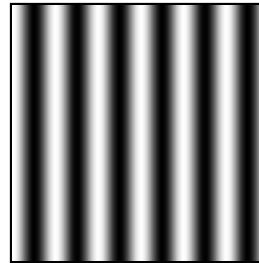
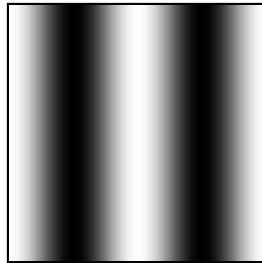
# Extension to 2D



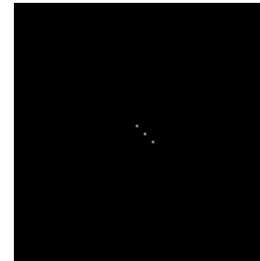
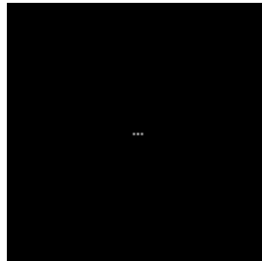
In matlab, check out: `imagesc(log(abs(fftshift(fft2(im)))));`  
`imagesc` Scale data and display as image to use full color map

# Fourier analysis in images

Intensity Image

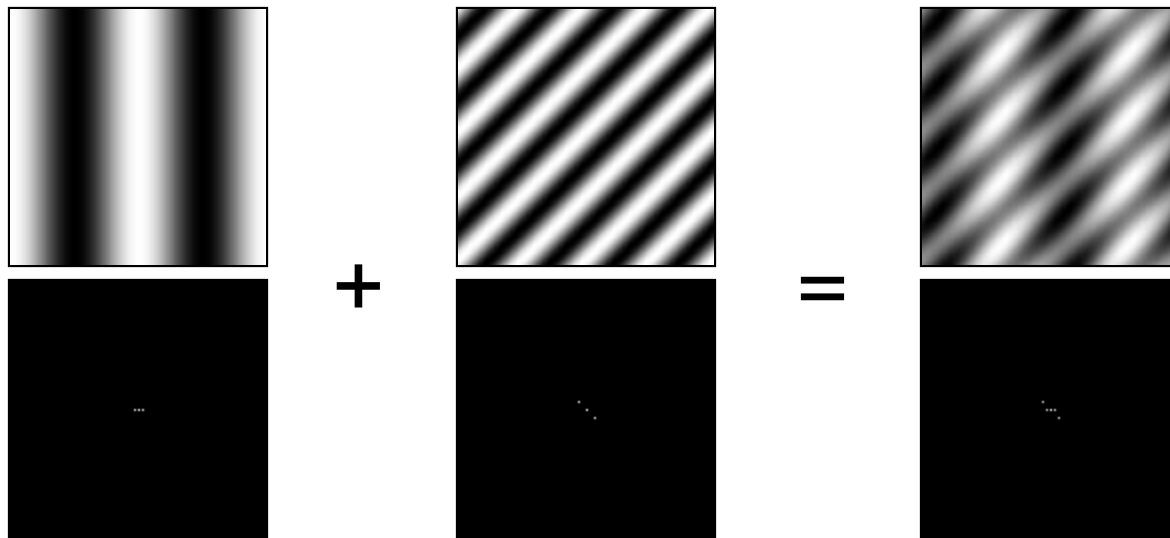


Fourier Image



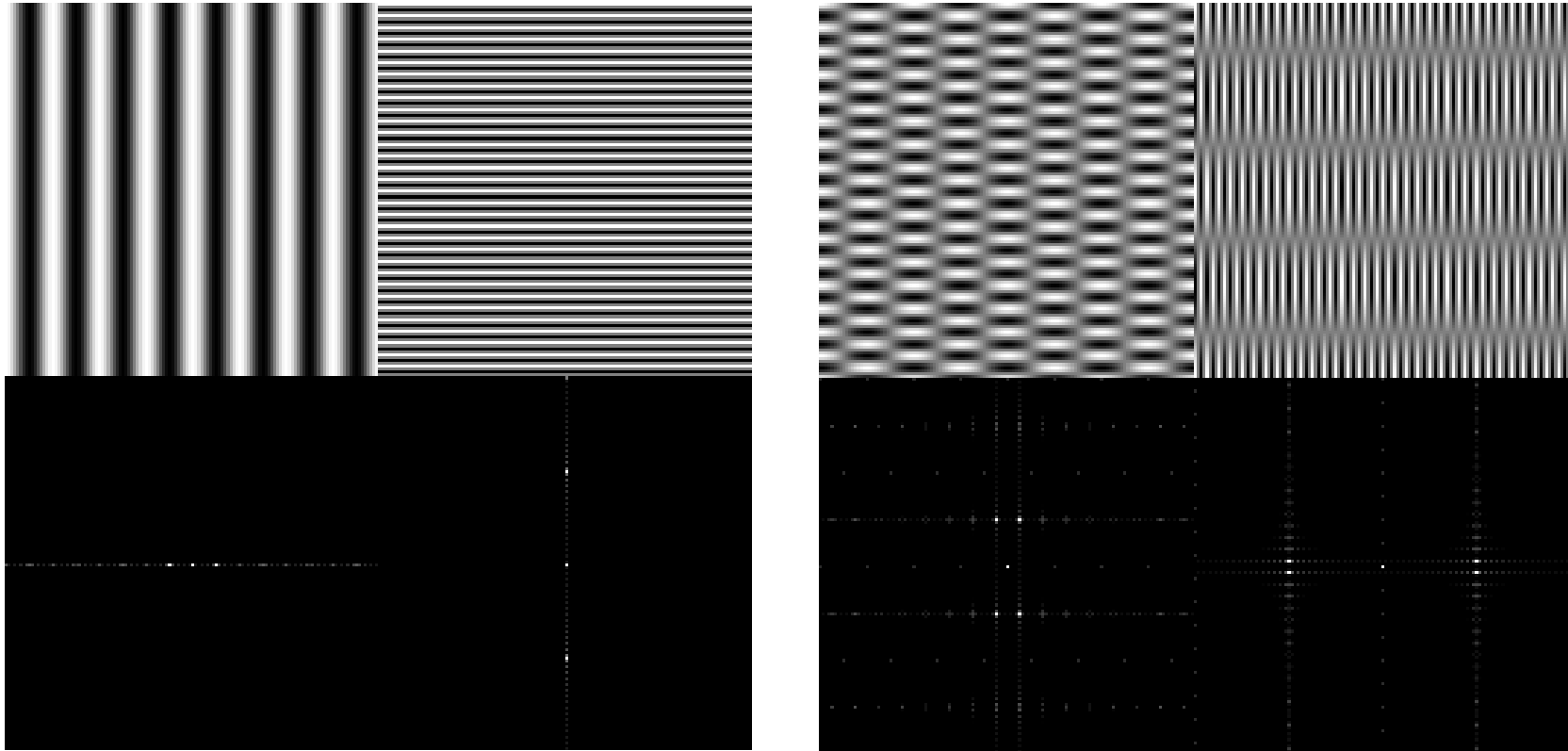


# Signals can be composed



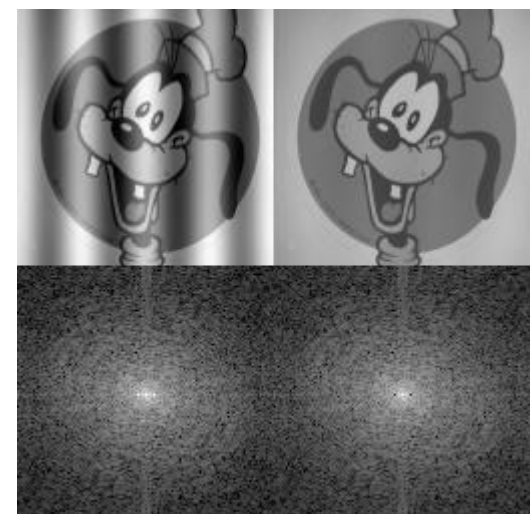
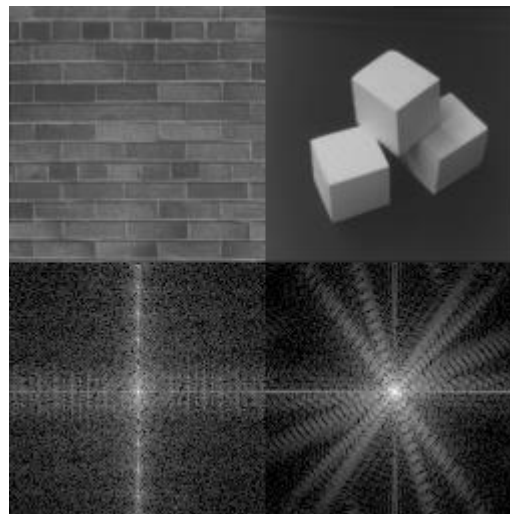
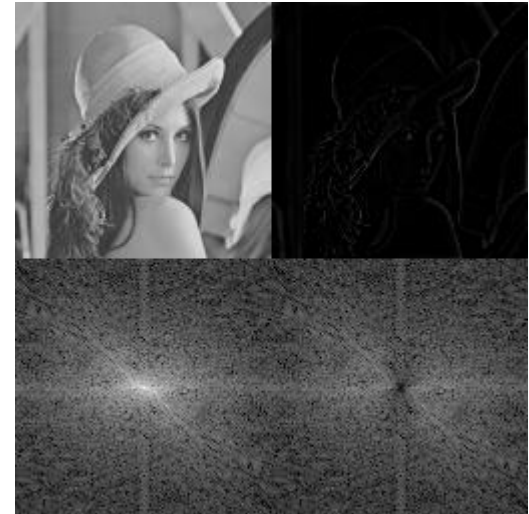
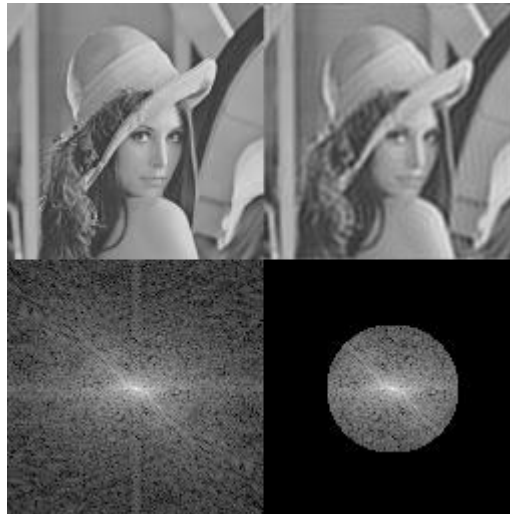
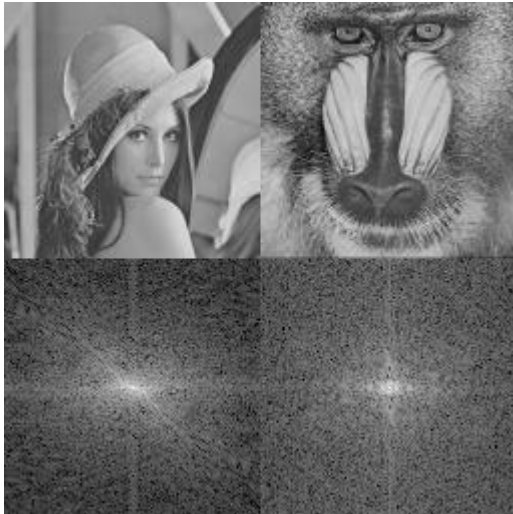
<http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering>  
More: <http://www.cs.unm.edu/~brayer/vision/fourier.html>

# Examples



Magnitudes are shown

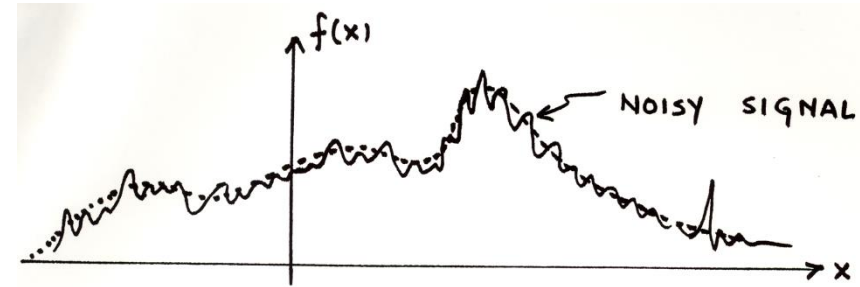
# Examples



# Example use: Smoothing/Blurring

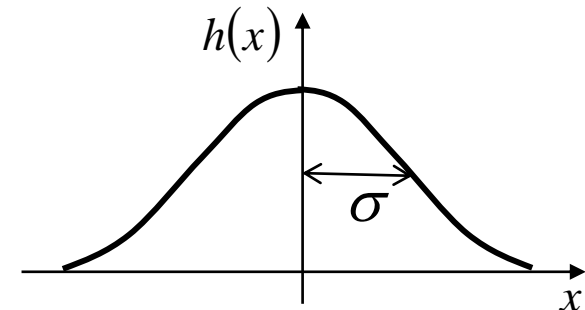
- We want a smoothed function of  $f(x)$

$$g(x) = f(x) * h(x)$$



- Let us use a Gaussian kernel

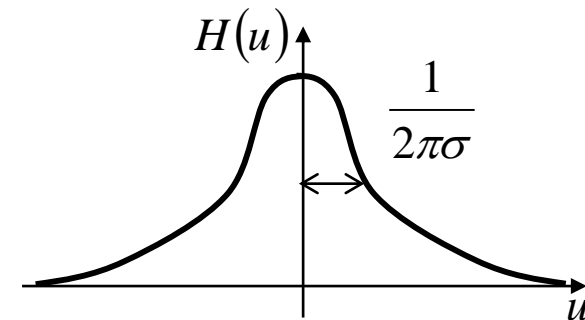
$$h(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2} \frac{x^2}{\sigma^2}\right]$$



- Then

$$H(u) = \exp\left[-\frac{1}{2} (2\pi u)^2 \sigma^2\right]$$

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



$H(u)$  attenuates high frequencies in  $F(u)$  (Low-pass Filter)!

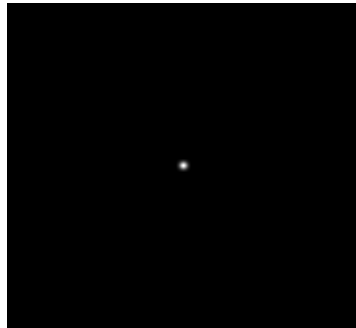
# 2D convolution theorem example

$f(x,y)$



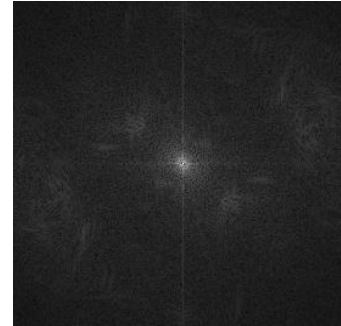
\*

$h(x,y)$



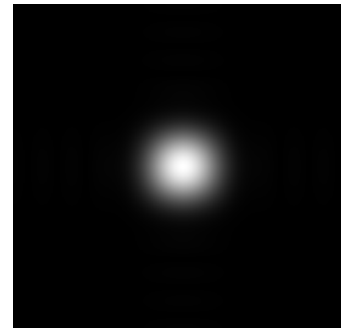
$\Downarrow$

$g(x,y)$

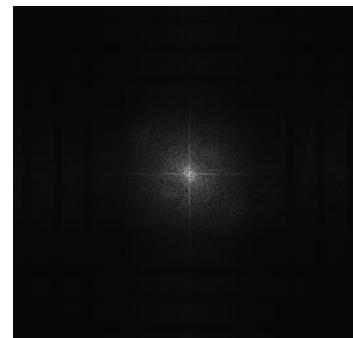


$\times$

$|F(s_x, s_y)|$   
( or  $|F(u, v)|$  )



$\Downarrow$

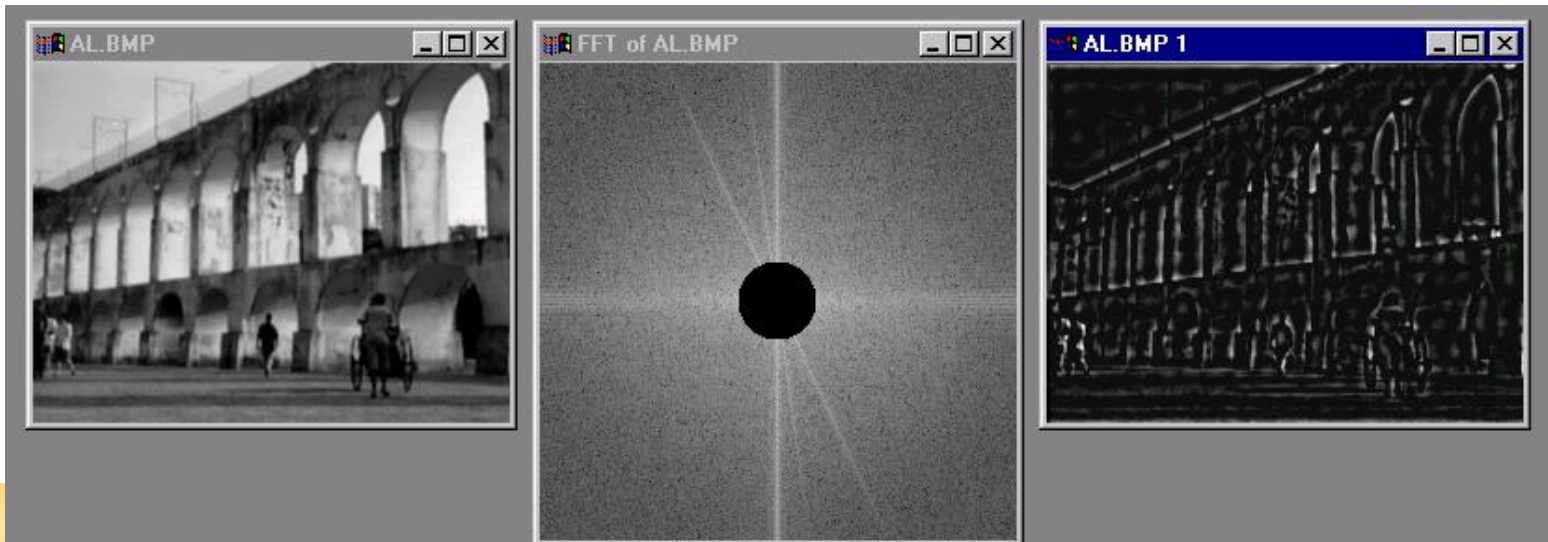
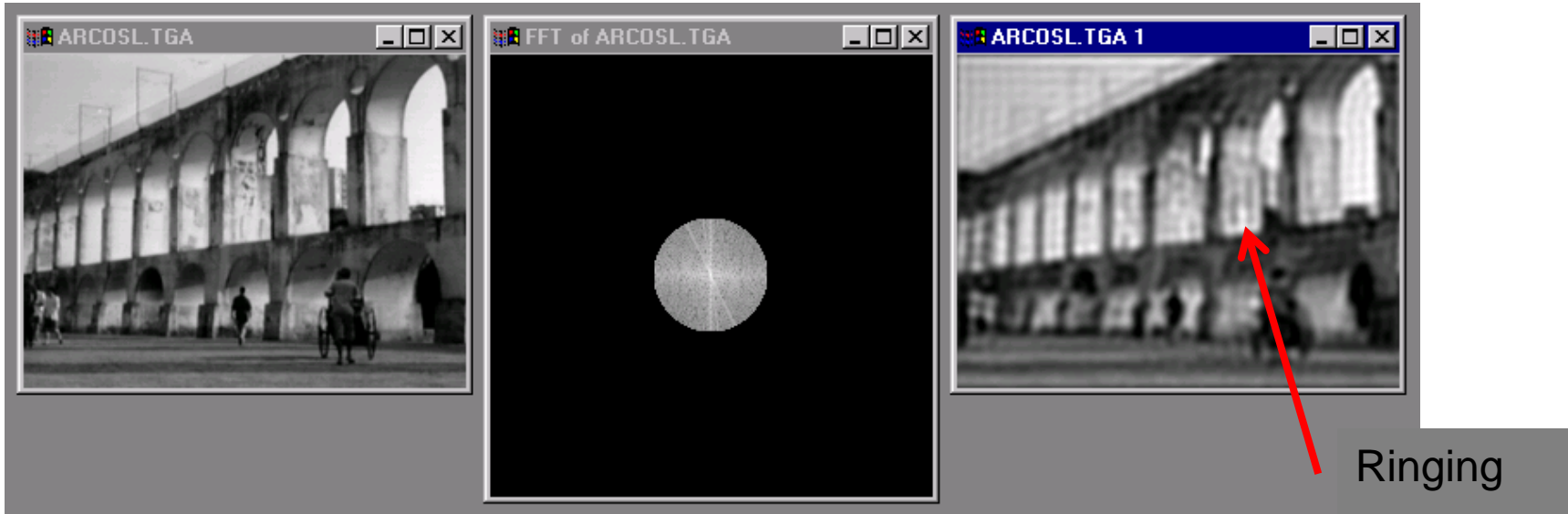


$|H(s_x, s_y)|$

$|G(s_x, s_y)|$



# Low and High Pass filtering





# Properties of Fourier Transform

Spatial Domain ( $x$ )

Frequency Domain ( $u$ )

**Linearity**

$$c_1 f(x) + c_2 g(x)$$

$$c_1 F(u) + c_2 G(u)$$

**Scaling**

$$f(ax)$$

$$\frac{1}{|a|} F\left(\frac{u}{a}\right)$$

**Shifting**

$$f(x - x_0)$$

$$e^{-i2\pi u x_0} F(u)$$

**Symmetry**

$$F(x)$$

$$f(-u)$$

**Conjugation**

$$f^*(x)$$

$$F^*(-u)$$

**Convolution**

$$f(x) * g(x)$$

$$F(u)G(u)$$

**Differentiation**

$$\frac{d^n f(x)}{dx^n}$$

$$(i2\pi u)^n F(u)$$

# Periodicity

- The DFT is periodic
  - Sinusoids have infinite, repeating extent and so the DFT ‘image’ is infinite and repeated (tiled)

$$\mathcal{F}(u, v) = \mathcal{F}(u + M, v) = \mathcal{F}(u, v + N) = \mathcal{F}(u + M, v + N), \quad (9.16)$$

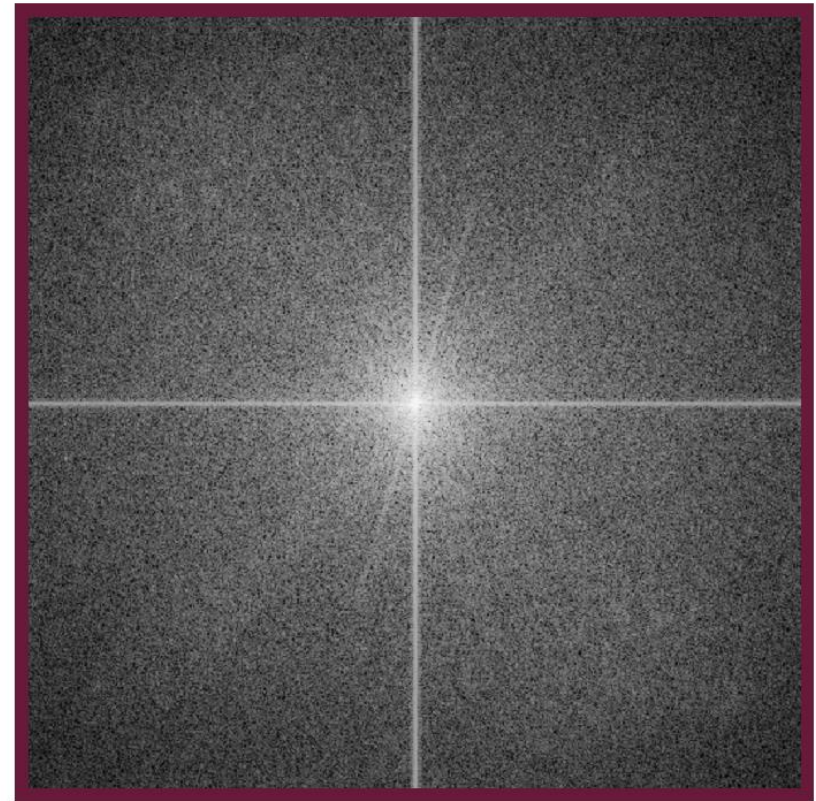
- While this property of the DFT may initially seem to be of little consequence, it carries important implications.
  - Since the source image wraps around itself, the source image will appear to have edges where no edges actually exist.
  - For example, the left side of an image is bright and the right side is dark, there will be a false edge created when the image is tiled since the brightness of the left side is placed adjacent to the darkness of the right side.
  - These artificial edges will then be reflected in the amplitude spectrum and give a false characterization of the power spectrum of the source image.



# Example



(a) Tiled source.



(b) Amplitude spectrum of the source.

Figure 9.11. Periodicity effects of the DFT.

# Windowing

- Windowing is a technique to minimize the artificial distortions injected into the spectrum of an image due to tiling discontinuities.
- Windowing is an image processing filter that is applied prior to conversion into the Fourier domain and an inverse windowing filter must be applied when recovery of the source image is desired from the DFT coefficients.
- Windowing forces continuity (or near continuity) at tile edges. Smoothly force the samples to be zero (or near zero) at the image edges by scaling image samples.



# Windowing

- Windowing is multiplication
  - Each sample is multiplied by a windowing function
  - Windowing function is parameterized on distance from center
  - Windowing function is unity at the center and falls to zero at image borders
- Choose a windowing function
  - Bartlett
  - Hanning
  - Blackman
  - Hamming




A decorative image in the top-left corner showing a blue square above a colorful, abstract pattern.

# Windowing Functions

- The Bartlett window is a cone
  - Discontinuous at edges and center
  - Computationally simple

$$w(r) = \begin{cases} 1 - \frac{r}{r_{max}} & r \leq r_{max} \\ 0 & r > r_{max} \end{cases}$$

- Hanning window
  - Continuous

$$w(r) = \frac{1}{2} - \frac{1}{2} \cos \left[ \pi \left( 1 - \frac{r}{r_{max}} \right) \right]$$
A decorative image in the bottom-left corner showing three small, colorful, abstract patterns.



# Bartlett

- Blackman window
  - Continuous
  - Steeper dropoff than Hanning

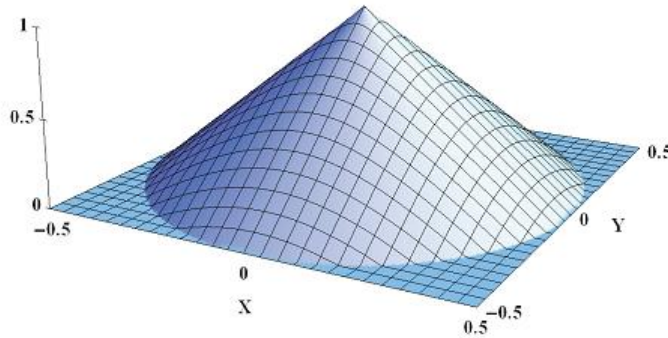
$$w(r) = .42 - \frac{1}{2} \cos \left[ \pi \left( 1 - \frac{r}{r_{\max}} \right) \right] + .08 \cos \left[ 2\pi \left( 1 - \frac{r}{r_{\max}} \right) \right]$$

- Hamming (not Hanning) window
  - Discontinuous at edges
  - Steeper dropoff than Hanning

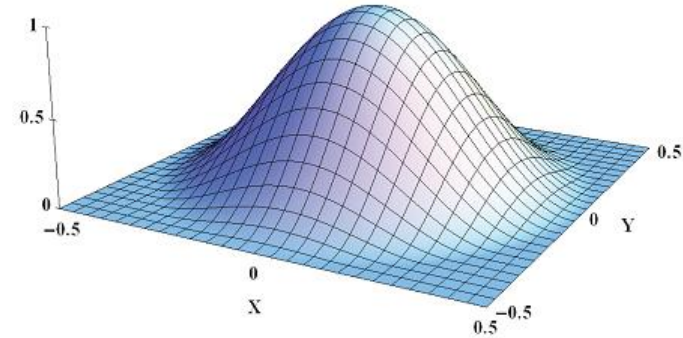
$$w(r) = 0.53836 + 0.46164 \cos \left( \pi \frac{r}{r_{\max}} \right)$$



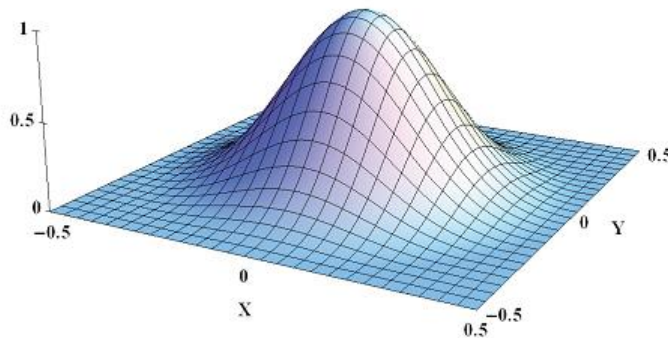
# Common Windowing Functions



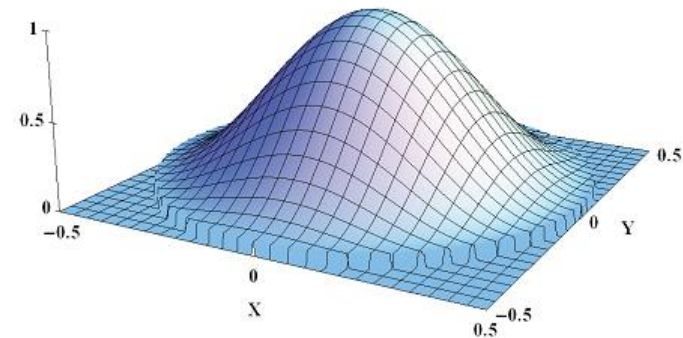
(a) Bartlett Window.



(b) Hanning Window.



(c) Blackman Window.



(c) Hamming Window.

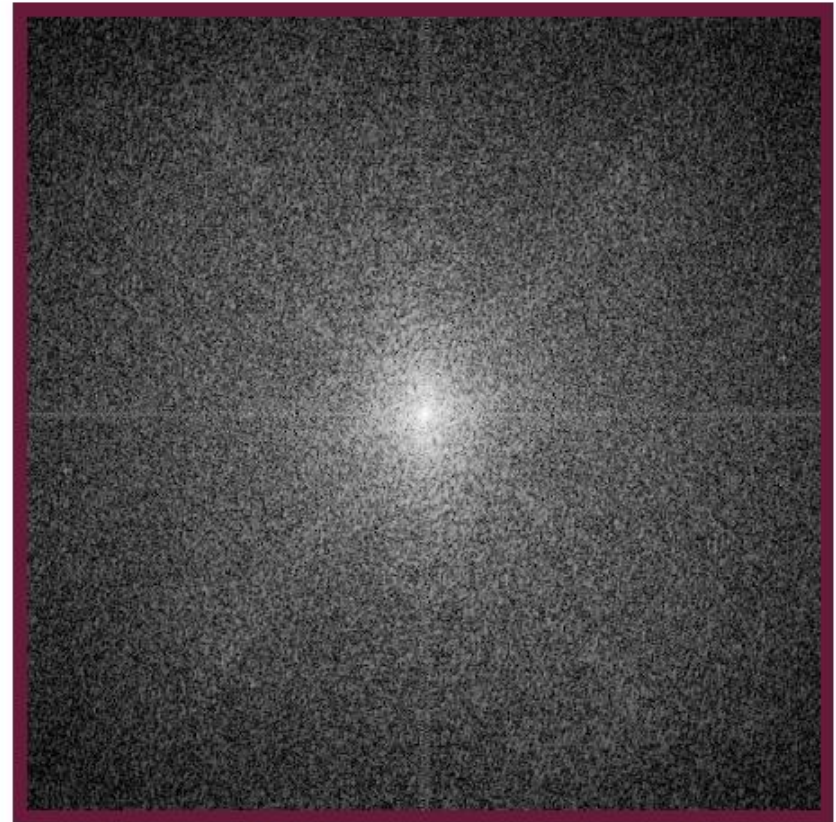
Figure 9.12. Normalized windowing functions covering a  $1 \times 1$  region.



# Effect of Windowing on Amplitude Spectrum



(a) Tiled source.



(b) Amplitude spectrum of the source.

Figure 9.13. The effect of windowing on the spectrum.





# Frequency Domain Filtering

- Images can be processed in either the spatial or frequency domain
  - Spatial domain filters have already been discussed
  - Frequency domain filters can achieve the same results by altering the DFT coefficients directly
  - Frequency domain filtering can be generalized as the multiplication of the spectrum  $F$  of an image by a transfer function  $H$ .

$$\mathcal{G}(u, v) = \mathcal{F}(u, v) \cdot \mathcal{H}(u, v), \quad (9.21)$$



# Frequency Domain Filtering

- Recall that the DFT coefficients are complex-valued.
  - Multiplication of  $F$  with  $H$  will possibly change both the amplitude and phase spectrum
  - In practice, however, most frequency domain filters are zero-phase. This means that the phase spectrum is not changed; only the amplitude spectrum is changed.

$$\mathcal{G}(u, v) = \mathcal{F}(u, v) \cdot \mathcal{H}(u, v), \quad (9.21)$$



# Convolution

- Convolution in the spatial domain corresponds to multiplication in the Fourier domain.

$$f \otimes h \Leftrightarrow \mathcal{F} \cdot \mathcal{H}$$

- This has strong computational implications
  - Recall that convolution of an  $N \times N$  image with  $M \times M$  kernel is  $O(M^2 N^2)$  in the spatial domain.
  - What is the performance in the Fourier domain?
    - $O(N^2)$

# Convolution

- Convolution of an image  $f$  with kernel  $h$  can be performed using point-by-point multiplication

1. Compute the DFT coefficients of  $f$  to obtain  $\mathcal{F}$
2. Compute the DFT coefficients of  $h$  to obtain  $\mathcal{H}$
3. Compute the product of these two transforms to obtain  $\mathcal{F} \cdot \mathcal{H}$
4. Compute the inverse DFT of the product to obtain  $\vec{\mathcal{F}}^{-1}(\mathcal{F} \cdot \mathcal{H})$

- This may seem inefficient since the DFT must be computed twice (forward and inverse).
  - The brute-force technique is  $O(n^4)$
  - The FFT technique (presented later) is  $O(N \log N)$

# 2D Filtering=Two Sequential 1D Filtering

- Just as we have observed with 2D transform, 2D (separable) filtering can be viewed as two sequential 1D filtering operations: one along row direction and the other along column direction
- The order of filtering does not matter

$$h(m,n) = h^1(m) \otimes h^1(n) = h^1(n) \otimes h^1(m)$$

$h^1$  : 1D filter



# Numerical Example

1D filter

$$h^1(m)=[1,1], h^1(n)=[1,-1]$$

$$h^1(m) \otimes h^1(n)$$

$$= \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

$$h^1(n) \otimes h^1(m)$$

$$= \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

MATLAB command:

```
>h1=[1,1];h2=[1,-1];  
>conv2(h1,h2)  
>conv2(h2,h1)
```

# Fourier Series (2D case)

$$F(w_1, w_2) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m, n) e^{-j(w_1 m + w_2 n)}$$

spatial-domain convolution

$$f(m, n) \otimes h(m, n)$$

frequency-domain multiplication

$$F(w_1, w_2) H(w_1, w_2)$$

Note that the input signal is **discrete**  
while its FT is a **continuous** function

# Filter Examples

## Low-pass (LP)

1D

$$h^1(n)=[1,1]$$



$$|H^1(w)|=2\cos(w/2)$$

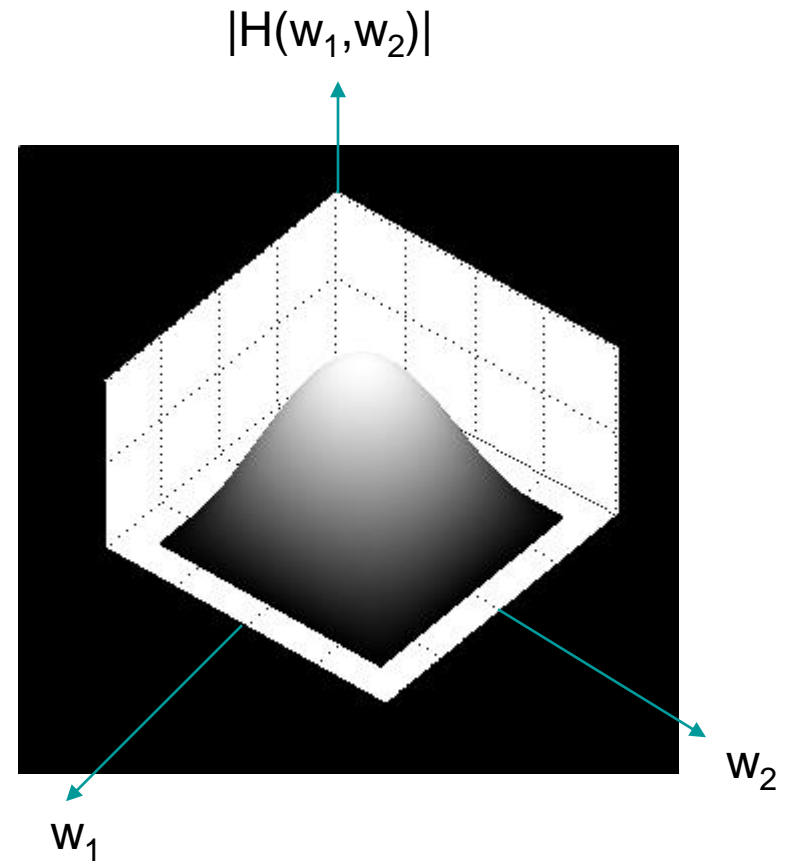


2D

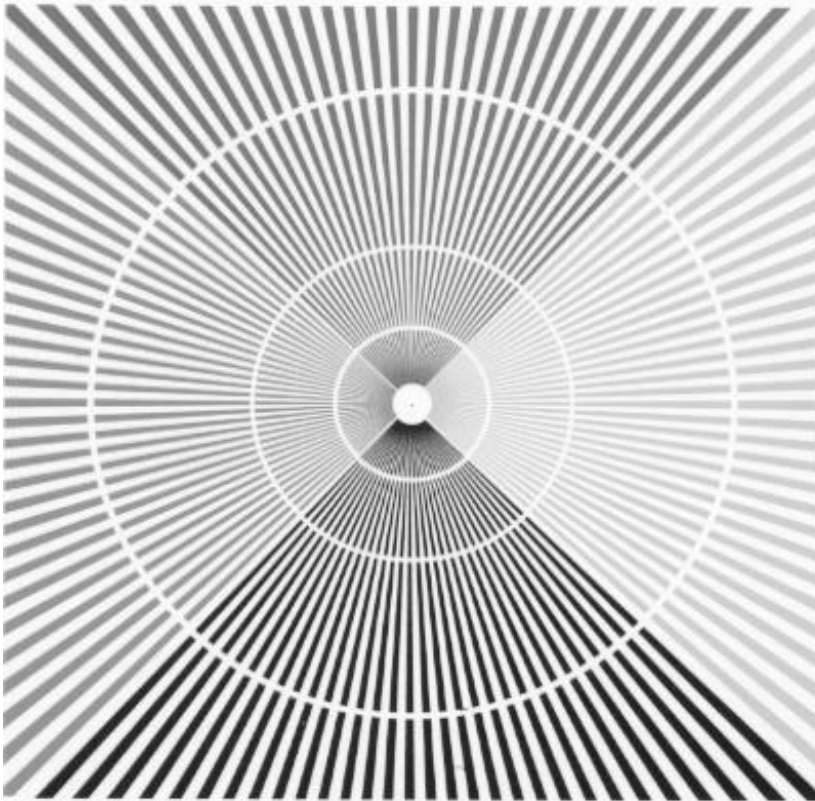
$$h(n)=[1,1;1,1]$$



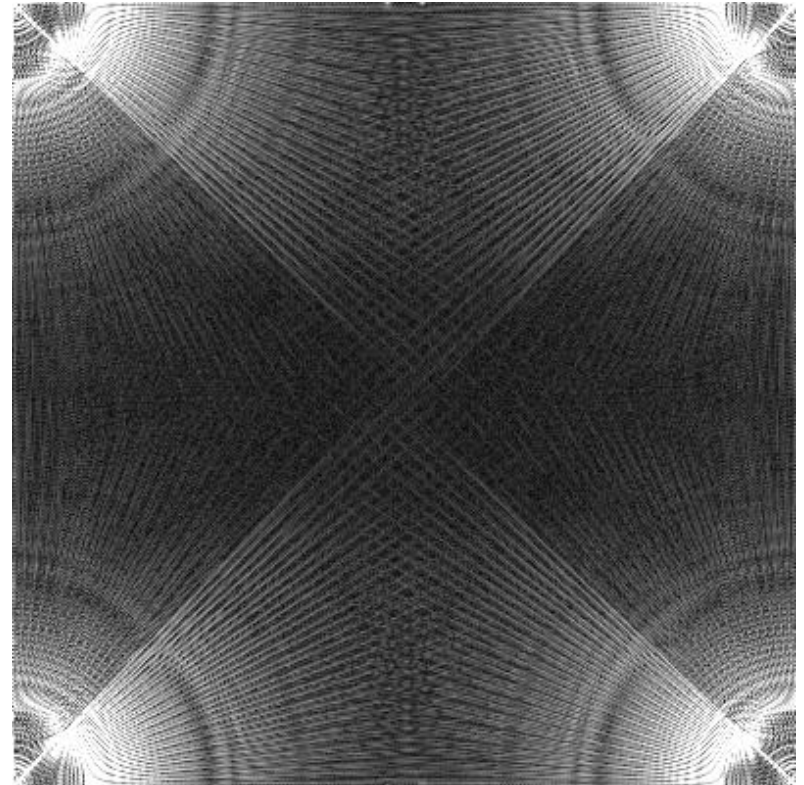
$$|H(w_1,w_2)|=4\cos(w_1/2)\cos(w_2/2)$$



# Image DFT Example

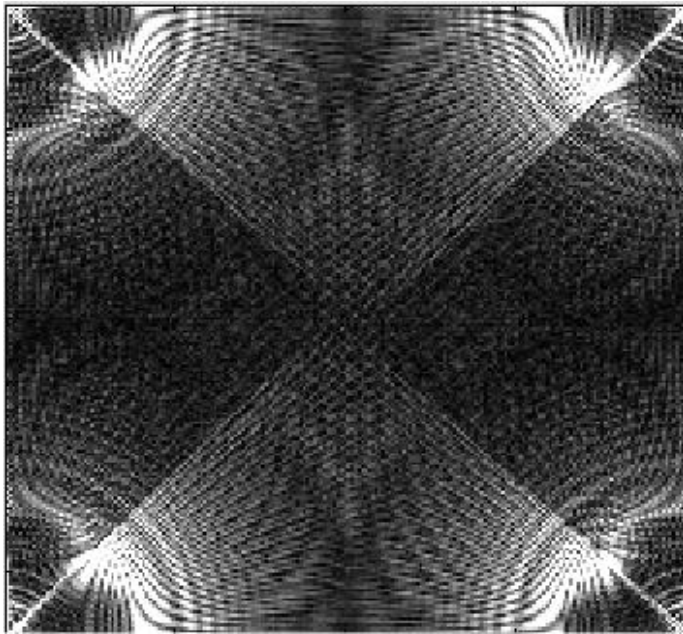


Original ray image  $\mathbf{X}$

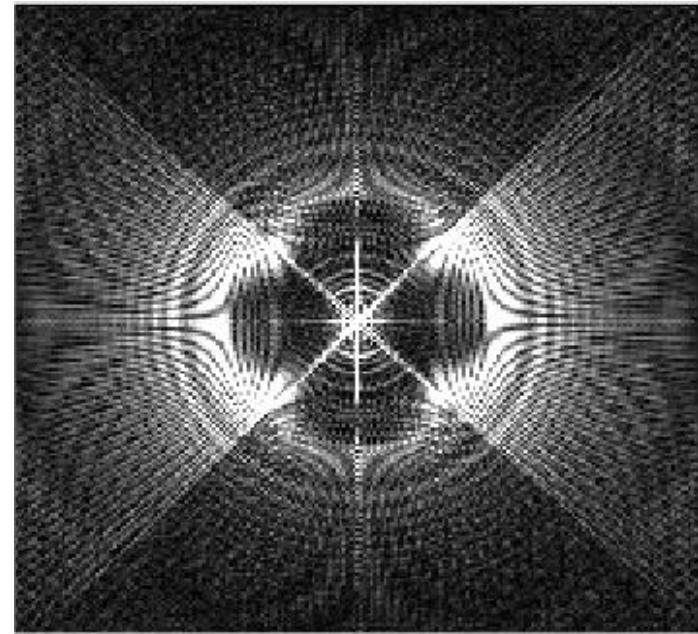


choice 1:  $\mathbf{Y}=\text{fft2}(\mathbf{X})$

# Image DFT Example (Con't)



choice 1:  $\mathbf{Y} = \text{fft2}(\mathbf{X})$   
Low-frequency at four corners

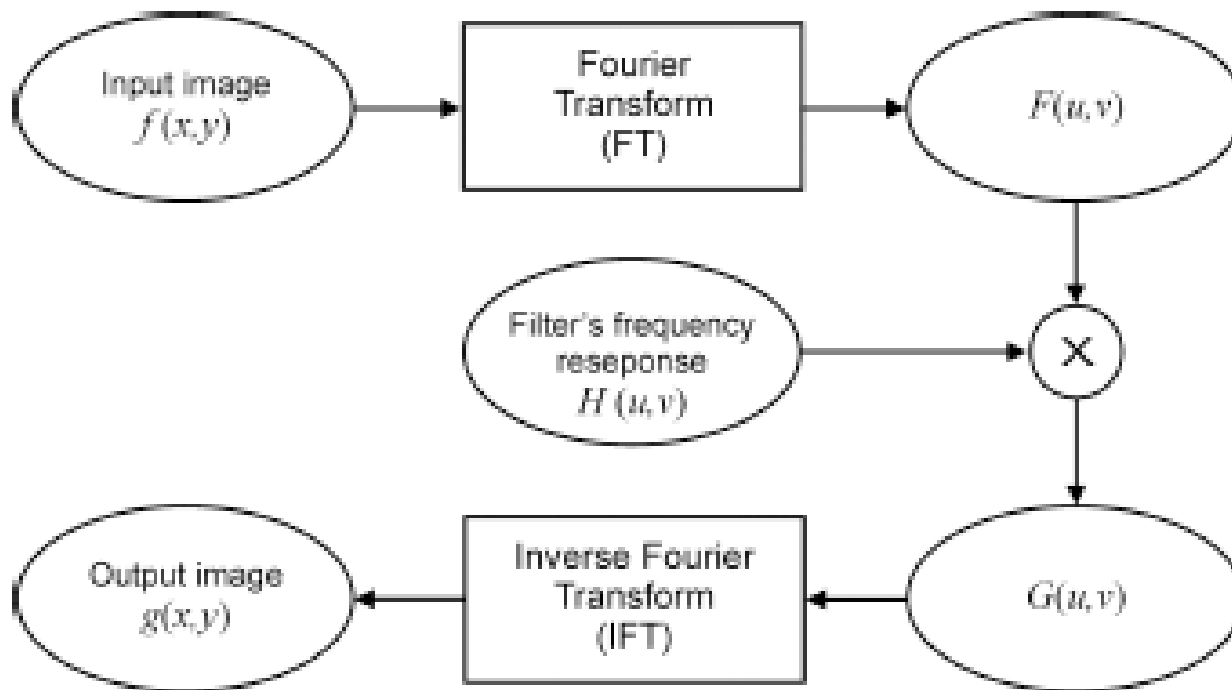


choice 2:  $\mathbf{Y} = \text{fftshift}(\text{fft2}(\mathbf{X}))$   
Low-frequency at the center

FFTSHIFT Shift zero-frequency component to center of spectrum.



# Frequency Domain Operation

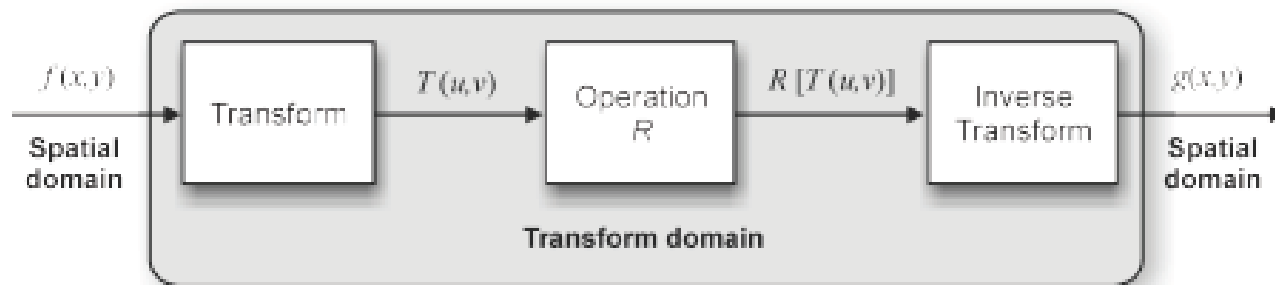




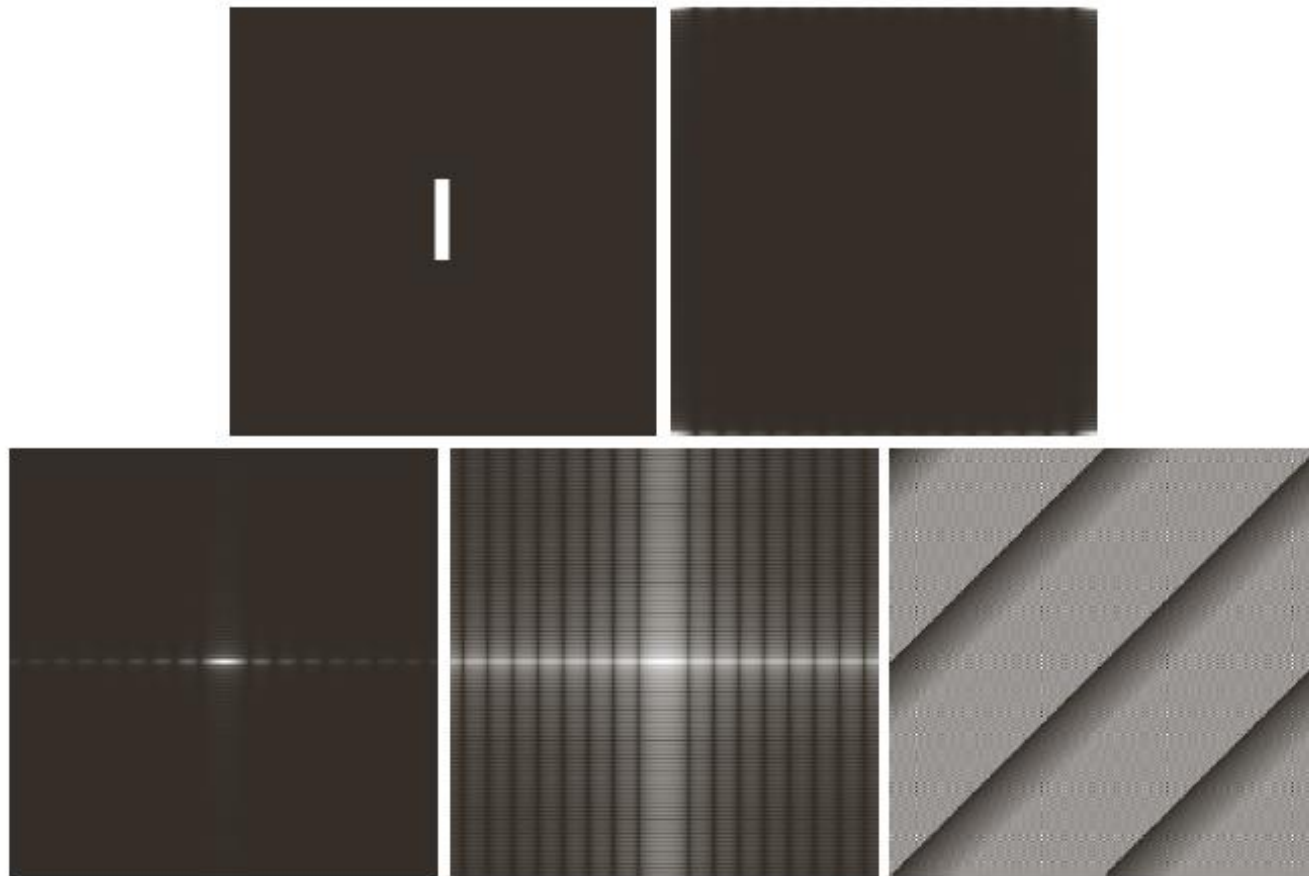
# Mathematical Foundation

- Transform: a mathematical tool that allows the conversion of a set of values to another set of values, creating a new way of representing the same information
- Some tasks are best performed by applying selected algorithm in the transformed domain

# Mathematical Foundation

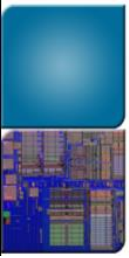


# Output

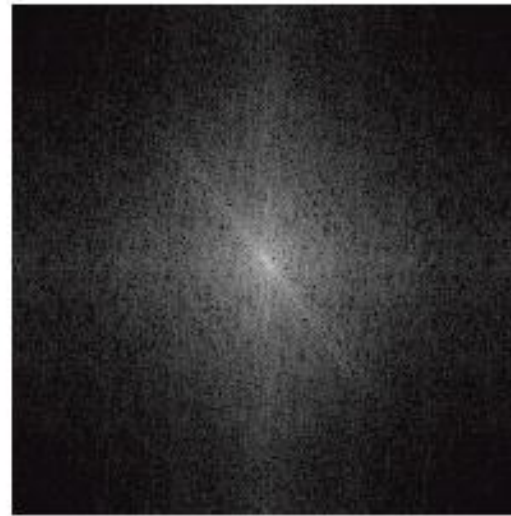


a b  
c d e

**FIGURE 4.3** (a) Image. (b) Fourier spectrum. (c) Centered spectrum. (d) Spectrum visually enhanced by a log transformation. (e) Phase angle image.



# Example of Image and FT



A decorative image in the top-left corner consisting of a blue square above a colorful, abstract pattern.

# Filtering in Frequency Domain

- The relation between spatial domain convolution and frequency domain filtering

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

- And vice versa

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



# Low Pass Filter

- Low-pass filters attenuate the high frequency components of the Fourier Transform of an image, while leaving the low frequency components unchanged
- The typical overall effect of applying a low-pass filter to an image is a controlled degree of blurring





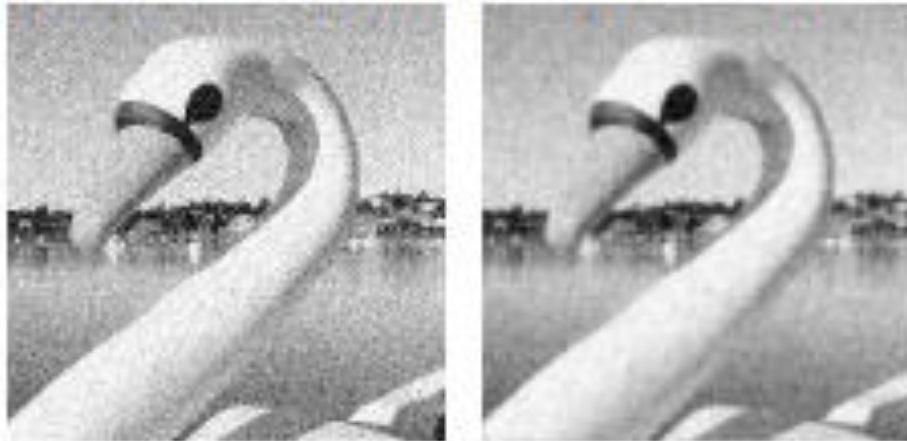
# Example of LPF for smoothing of false contours

- Example of LPF for smoothing of false contours

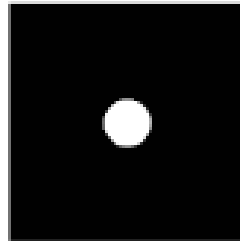
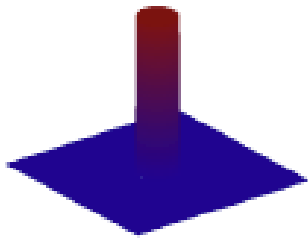


# Example of LPF for smoothing of false contours

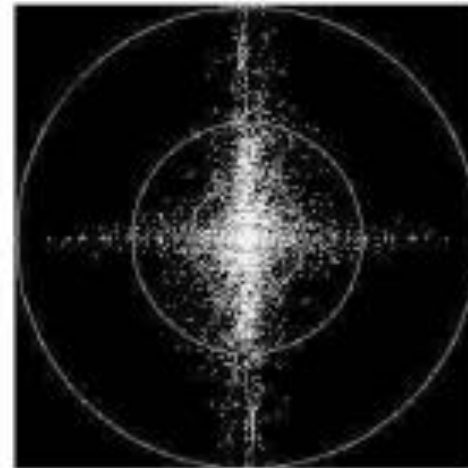
- Example of LPF for noise reduction



# Ideal LPF



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



# Ideal LPF

- b) radi 8
- c) radi 16
- d) radi 32
- e) radi 64
- f) radi 128



(a)



(b)



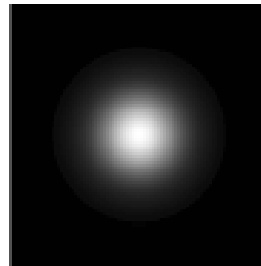
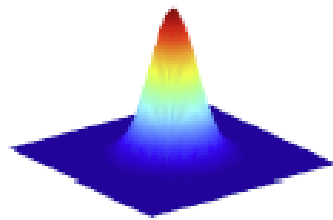
(c)



(d)



# Gaussian LPF



$$H_G(u, v) = e^{-\frac{D(u, v)^2}{2\sigma^2}}$$

# Gaussian LPF

- b)  $\sigma = 5$
- c)  $\sigma = 10$
- d)  $\sigma = 20$
- e)  $\sigma = 30$
- f)  $\sigma = 75$



(a)



(b)



(c)



(d)



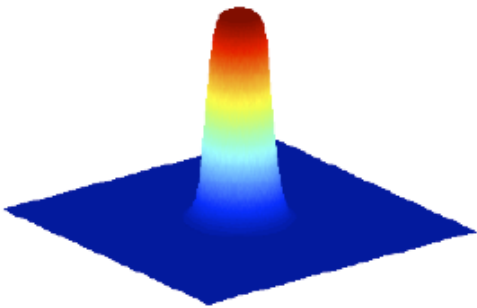
(e)



(f)



# Butterworth LPF



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

# Butterworth LPF

- b) radi 8
- c) radi 16
- d) radi 32
- e) radi 64
- f) radi 128



(a)



(b)



(c)



(d)

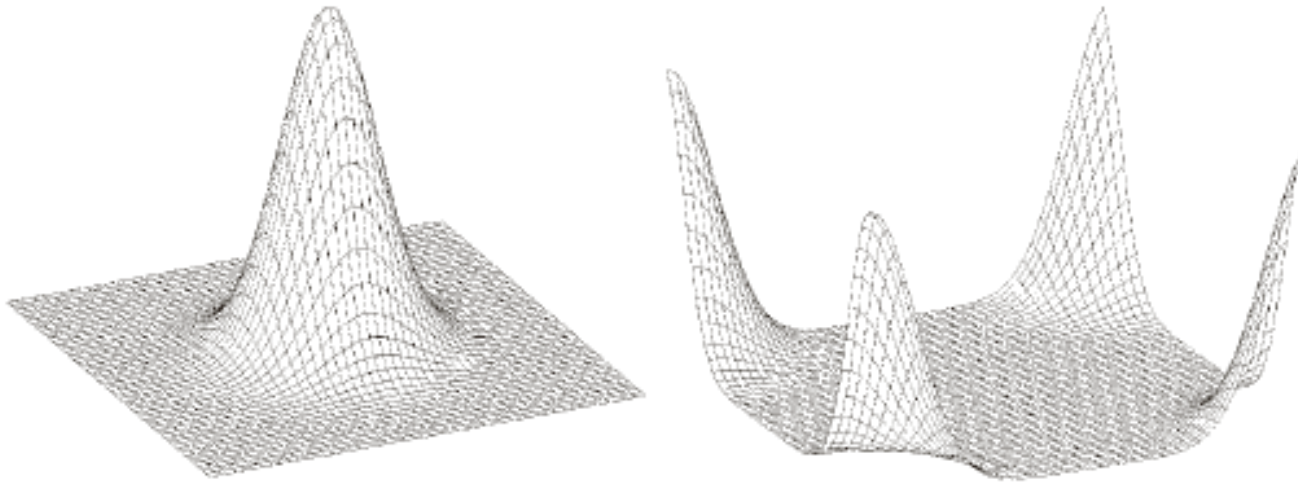


(e)



(f)

# Frequency Domain Low Pass Filter

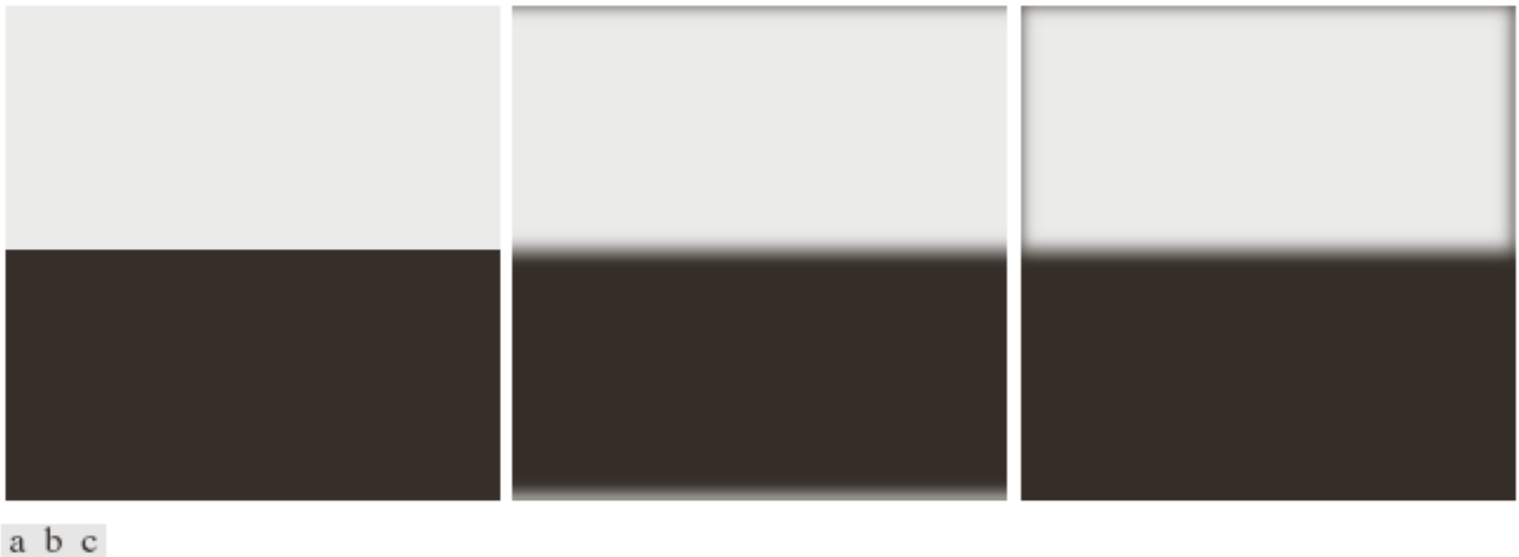


a b

**FIGURE 4.4**

Transfer functions of (a) a centered lowpass filter, and (b) the format used for DFT filtering. Note that these are frequency domain filters.

# Output



**FIGURE 4.5** (a) An image of size  $256 \times 256$  pixels. (b) Image lowpass-filtered in the frequency domain without padding. (c) Image lowpass-filtered in the frequency domain with padding. Compare the upper portion of the vertical edges in (b) and (c).

A decorative image in the top-left corner consisting of a blue square above a colorful, abstract pattern.

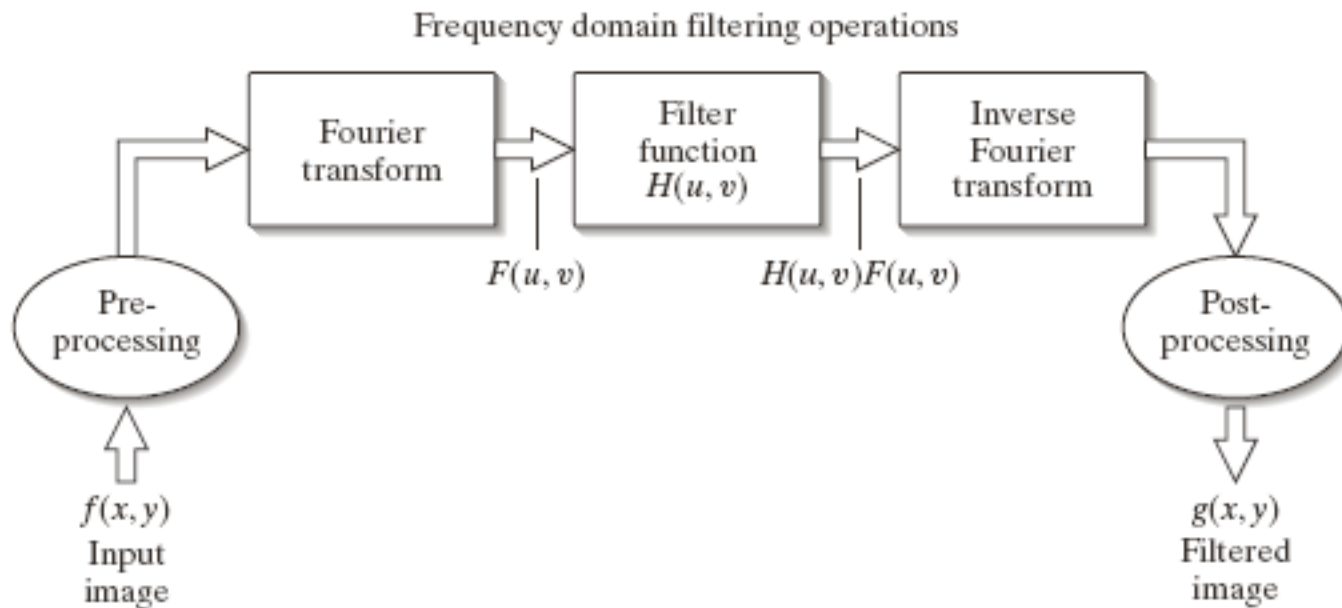
# Basic Step in DFT Filtering

1. Obtain the padding parameters
2. Obtain Fourier Transform with padding
3. Generate a filter function  $H$
4. Multiply the transform with the filter
5. Obtain the real part of the inverse FFT
6. Crop the top left rectangle to the original size



# Frequency Domain Filtering

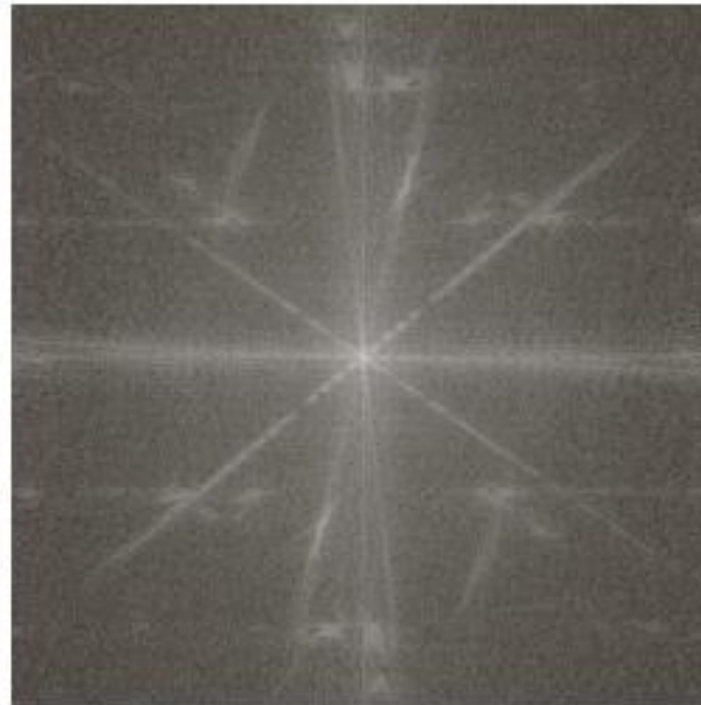
## Basic Step



**FIGURE 4.8**  
Basic steps for  
filtering in the  
frequency  
domain.



# Example Image



a b

**FIGURE 4.9**  
(a) A gray-scale  
image. (b) Its  
Fourier spectrum.

# Low Pass Filtering

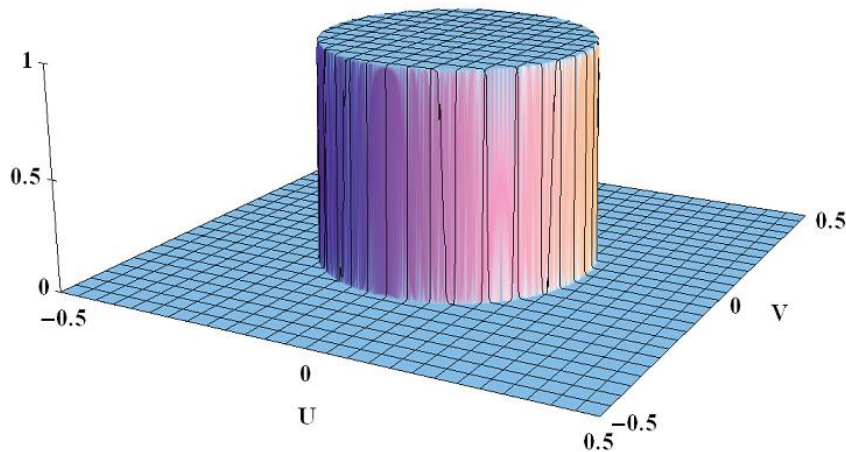
- Low pass filtering is convolution. It attenuates high frequency components of an image while leaving low frequency components intact.
- Low pass filtering can be naturally accomplished in the frequency domain since the frequency components are explicitly isolated in the spectrum.
- The DFT coefficients correspond to frequency components of the source and that their frequency increases with increasing distance from the center of the shifted spectrum.
  - Low pass filtering is then accomplished by zeroing the amplitude of the high frequency DFT coefficients. Determining which DFT coefficients should be zeroed amounts to determining how far the coefficients is from the center of the shifted spectrum.
  - Typically, a threshold radius is chosen such that all DFT coefficients outside of this threshold radius have their magnitude set to zero while all DFT coefficients falling inside of the threshold are unchanged (passed through).

# Low Pass Filtering

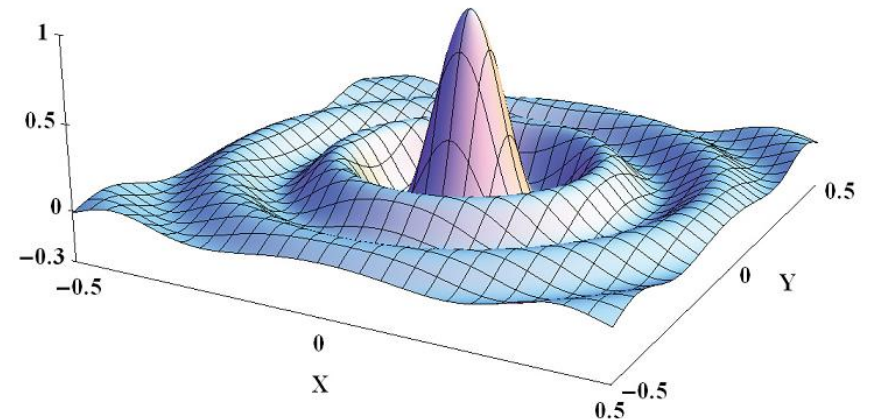
- Representing low pass filtering as multiplication of DFT coefficients, we must identify a transfer function ( $H$ ) that corresponds to setting high-frequency components to zero.
- An “ideal” low pass filter is given in 9.23 where  $r_c$  is the cutoff frequency.
  - The term *ideal* as it is used of an *ideal* low pass filter should not be taken to mean that this filter is optimal for low pass filtering.
  - An ideal low pass filter is ideal in the sense that it has an exact cutoff above which all terms are exactly zero and below which all terms are set to unity.

$$\mathcal{H}(u, v) = \begin{cases} 1 & \sqrt{u^2 + v^2} \leq r_c, \\ 0 & \text{otherwise,} \end{cases} \quad (9.23)$$

# Ideal low pass filter



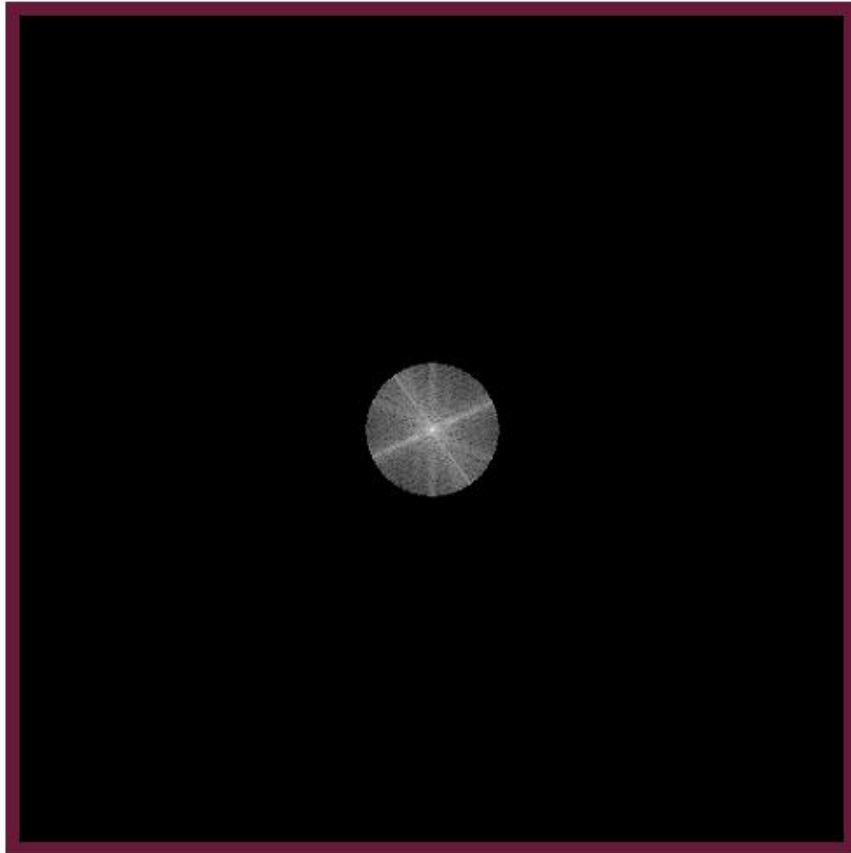
(a)  $\mathcal{H}(u, v)$ .



(b)  $h(x, y)$ .

Figure 9.15. Ideal low-pass filter in (a) the frequency domain and (b) the spatial domain.

# Ideal Low-Pass filtering example



(a) Filtered magnitude spectrum.



(b) Reconstructed spatial domain image.

Figure 9.14. Ideal low-pass filtering in the frequency domain.

# Low Pass Filtering

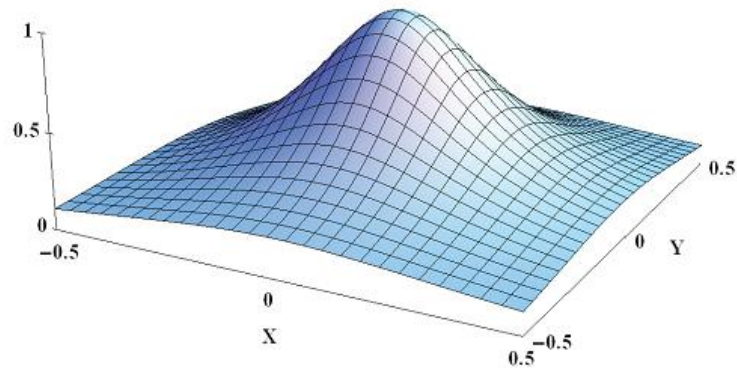
- Butterworth filter is a low-pass filter with smooth edges such that there is no (or minimal) ringing in the spatial domain
  - Parameterized on “order” – defines the sharpness of the filter

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

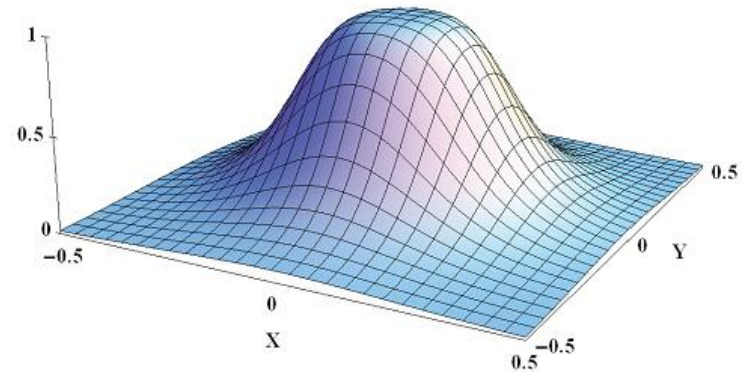
- N is the order of the filter. Larger N represent stronger cutoffs
- $R_c$  is the cutoff frequency. Smaller  $R_c$  is stronger blur



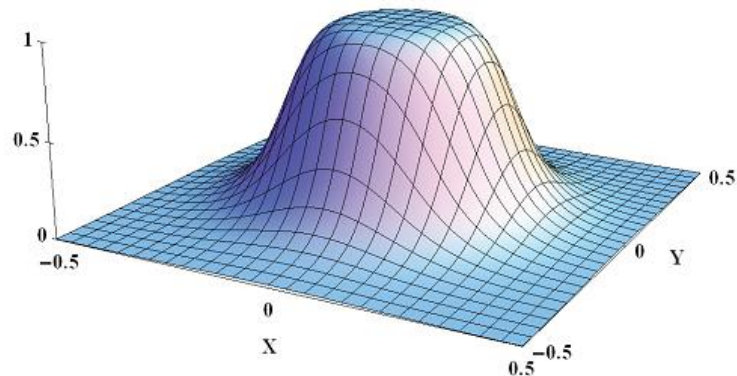
# Butterworth Filters



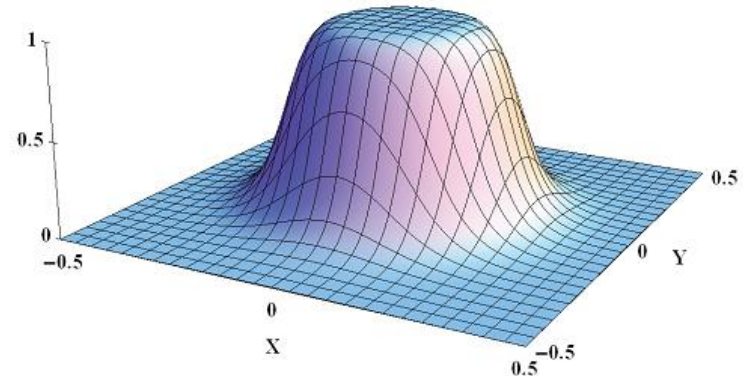
(a) First order,  $n = 1$ .



(b) Second order,  $n = 2$ .



(c) Third order,  $n = 3$ .



(d) Fourth order,  $n = 4$ .

Figure 9.16. Butterworth filters.

# Ideal vs. Butterworth (5<sup>th</sup> order)



(a)



(b)

Figure 9.17. (a) Low pass filtering in the frequency domain using an ideal filter and (b) 5th order

# Gaussian low pass filter

- Other well-known frequency domain low pass filters include the Chebyshev and the Gaussian transfer functions.
  - The Gaussian low pass filter has the very interesting property of having the same form in both the Fourier and spatial domains. In other words, the DFT of a Gaussian function is itself a Gaussian function.
  - A Gaussian low pass filter introduces no ringing when applied either in the spatial or frequency domains. The Gaussian transfer function is given as

$$\mathcal{H}(u, v) = e^{-\frac{1}{2}[r(u, v)/r_c]^2}. \quad (9.25)$$

A decorative image in the top-left corner consisting of a blue square above a circuit board pattern.

# High-pass and band-pass filtering

- High pass attenuates low frequency components while leaving high frequency components intact.
- The technique for performing high pass filtering is identical to that of low-pass filtering; however, the transfer functions are inverses to the low pass functions.



# High Pass

- Ideal – Butterworth - Gaussian

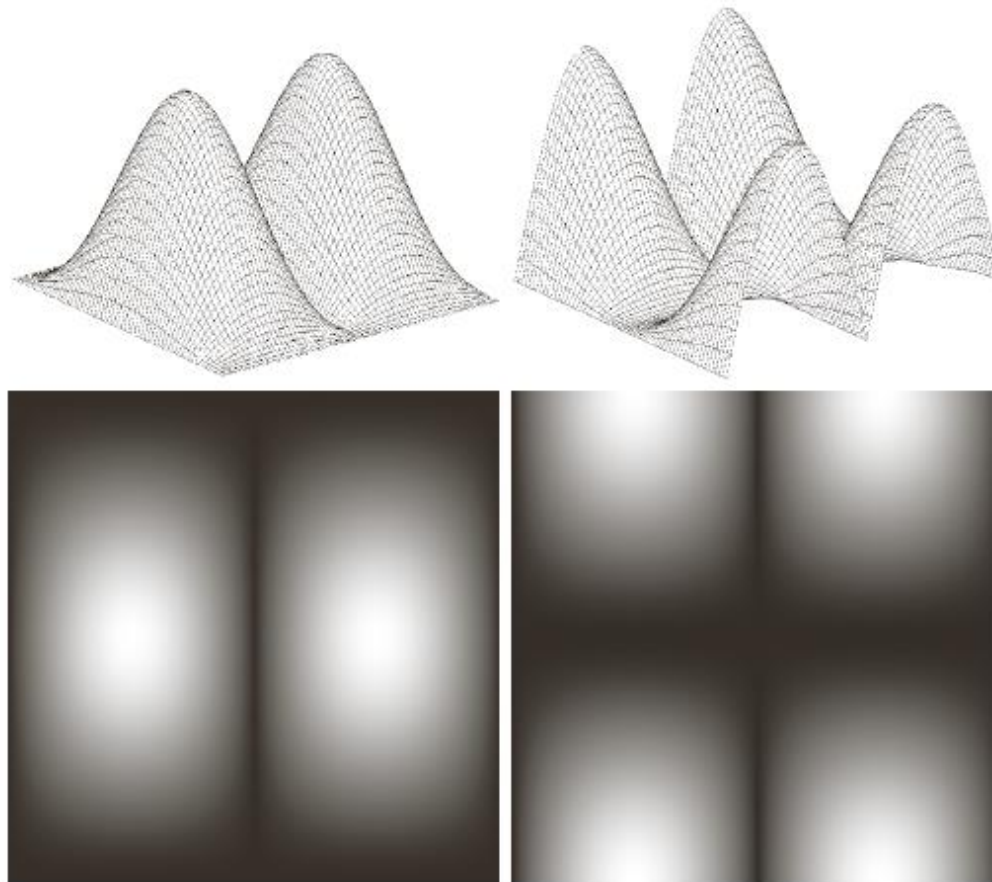
$$\mathcal{H}(u, v) = \begin{cases} 0 & \sqrt{u^2 + v^2} \leq r_c, \\ 1 & \text{otherwise.} \end{cases} \quad (9.26)$$

$$\mathcal{H}(u, v) = \frac{1}{1 + [r_c/r(u, v)]^{2n}}, \quad (9.27)$$

$$\mathcal{H}(u, v) = 1 - e^{-\frac{1}{2}[r(u, v)/r_c]^2}. \quad (9.28)$$



# High Pass Filter



a b  
c d

**FIGURE 4.10**

(a) Absolute value of the frequency domain filter corresponding to a vertical Sobel spatial filter.  
(b) The same filter after processing with function `ifftshift`.  
Figures (c) and (d) show the filters as images.



# Result

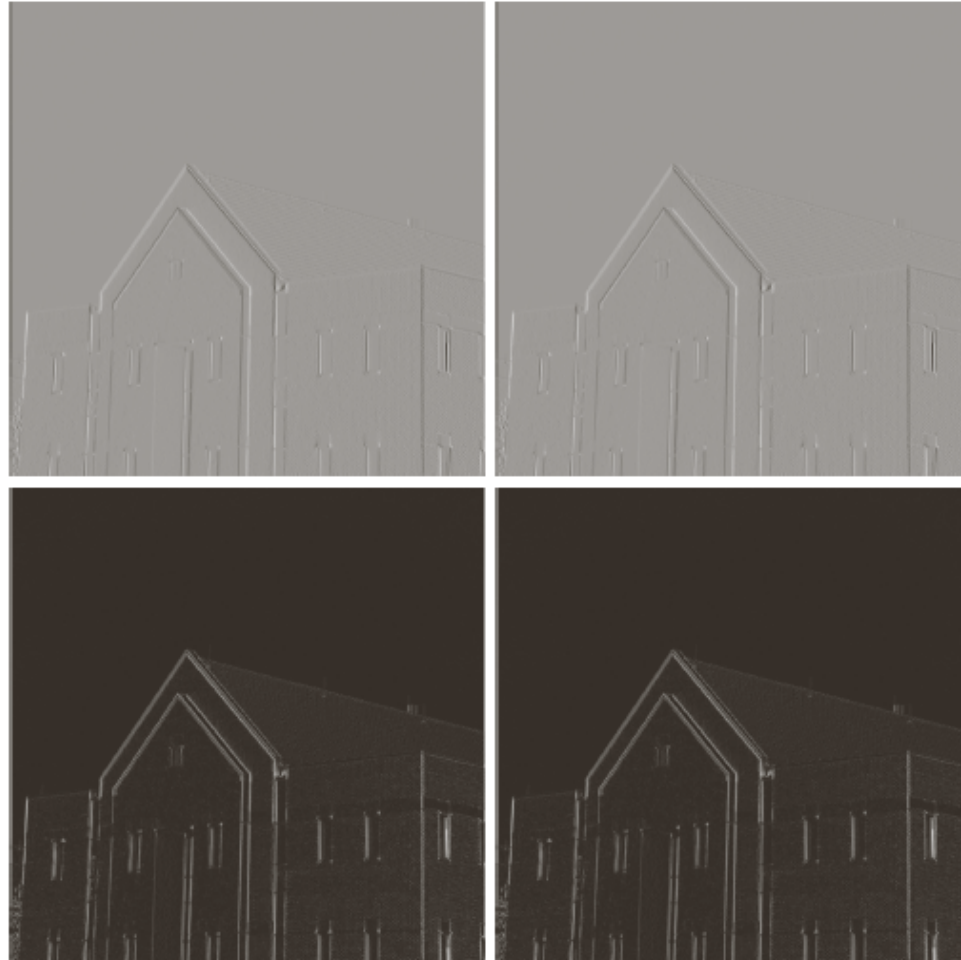
a b  
c d

**FIGURE 4.11**

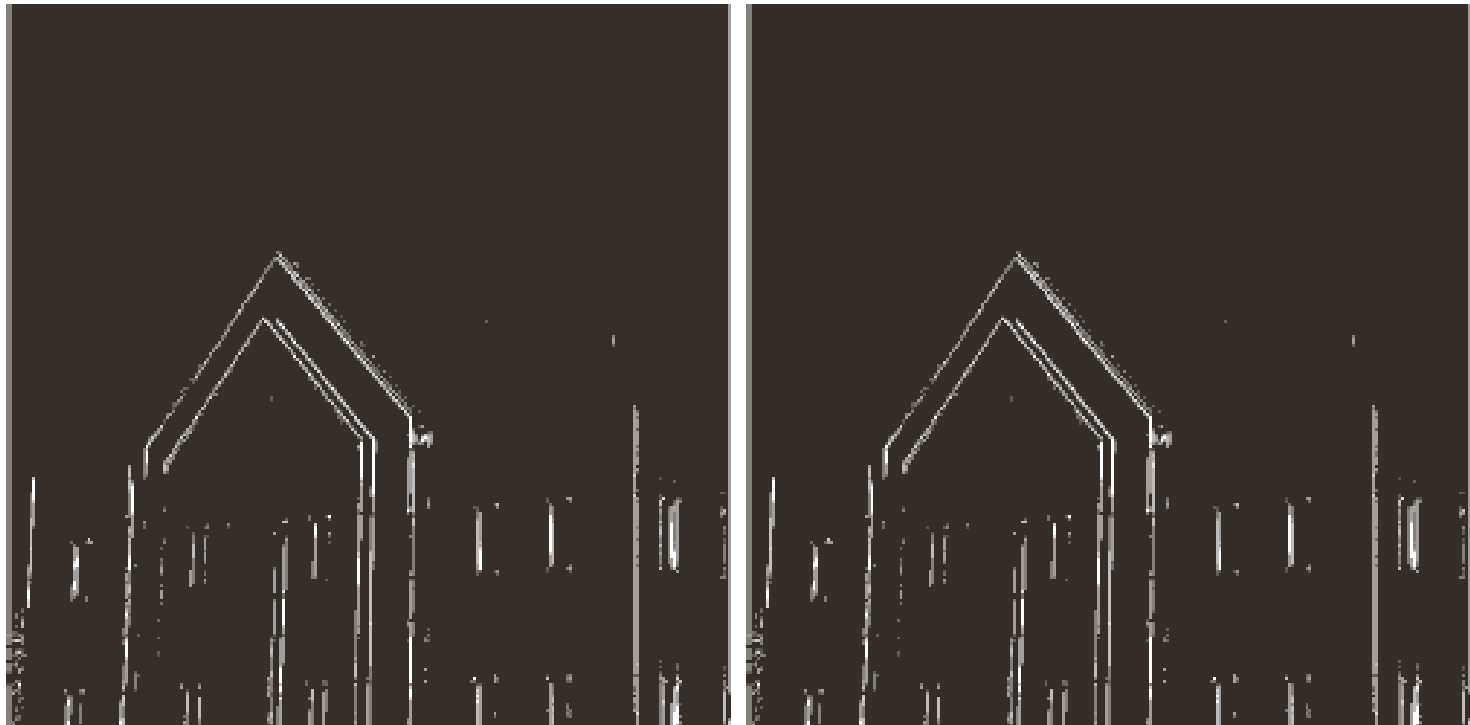
(a) Result of filtering Fig. 4.9(a) in the spatial domain with a vertical Sobel mask.

(b) Result obtained in the frequency domain using the filter shown in Fig. 4.10(b).

Figures (c) and (d) are the absolute values of (a) and (b), respectively.



# Result2

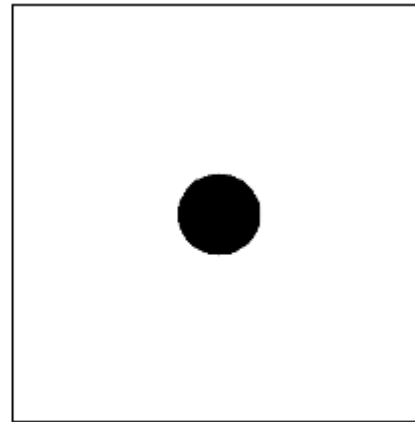
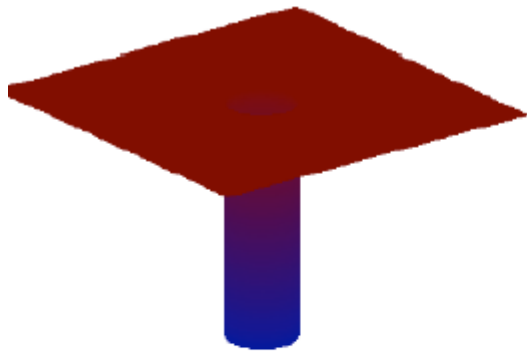


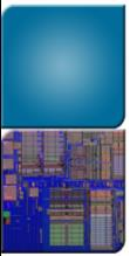
a b

**FIGURE 4.12** Thresholded versions of Figs 4.11(c) and (d), respectively, to show the principal edges more clearly.

# Ideal High-Pass Filter

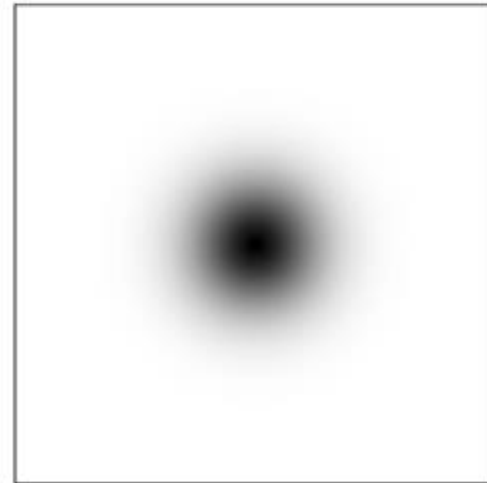
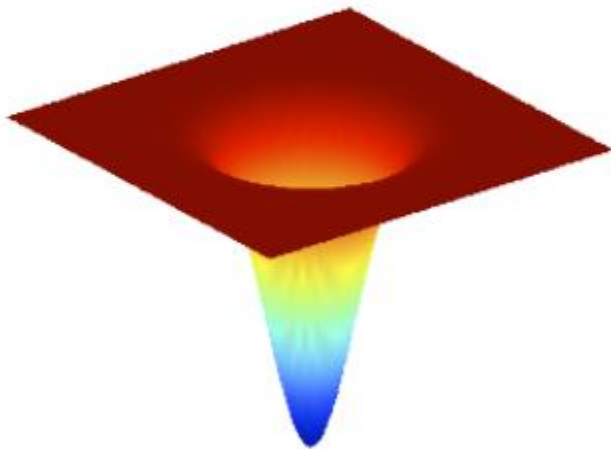
$$H_I(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$





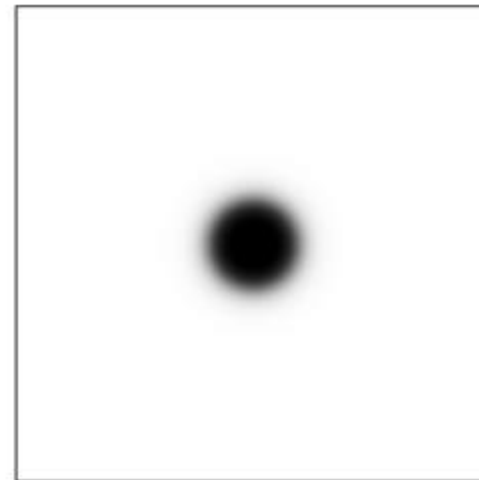
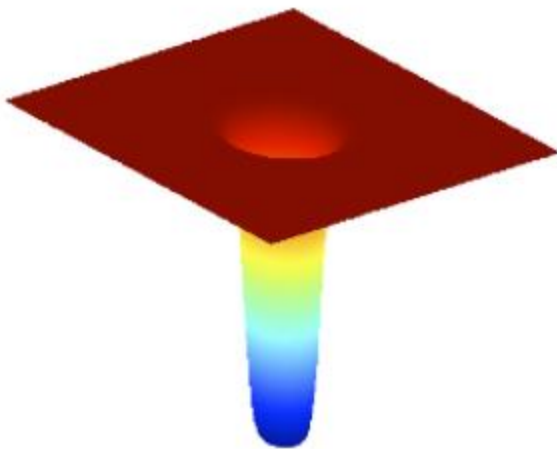
# Gaussian High Pass Filter

$$H_G(u, v) = 1 - e^{-\frac{D(u, v)^2}{2\sigma^2}}$$



# Butterworth High Pass Filter

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





# High Frequency Emphasis



(a)



(b)



(c)

a) ori

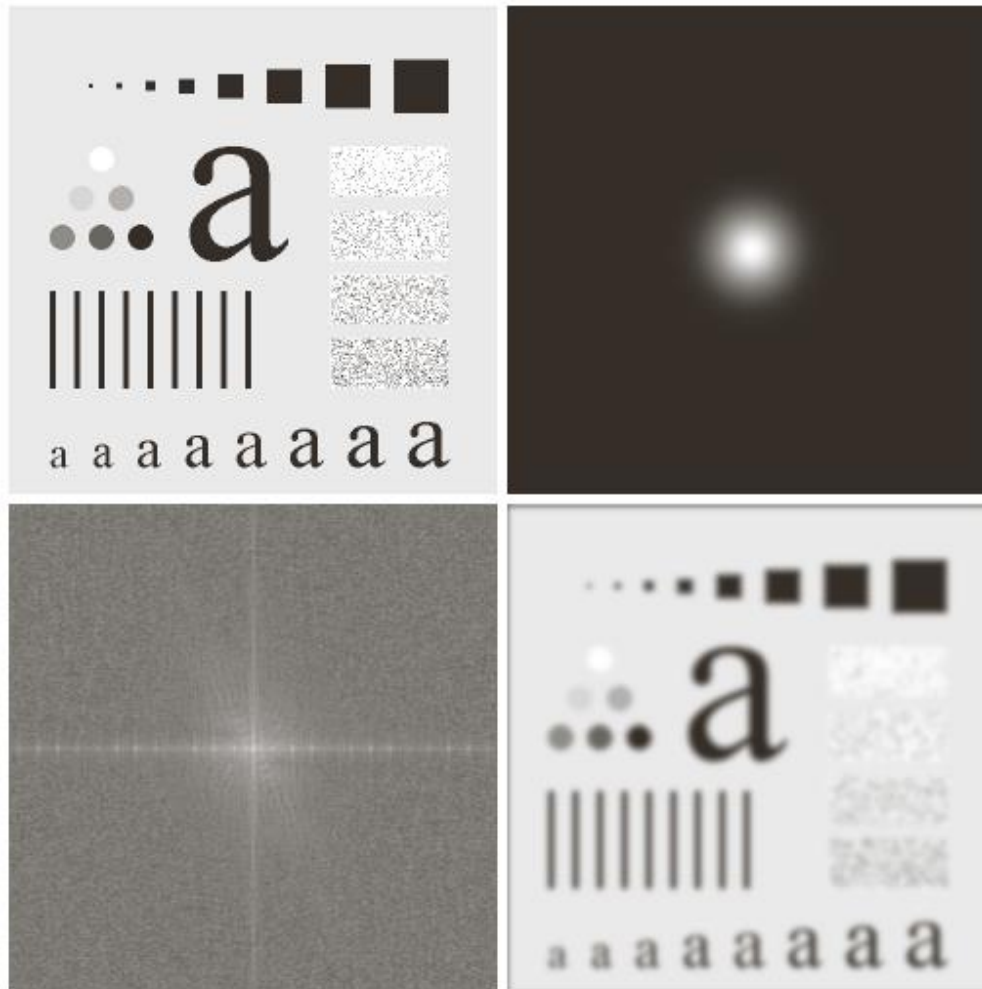
b) horizontal edge

c) high pass





# Filter Example



a b  
c d

**FIGURE 4.13**

Lowpass  
filtering.

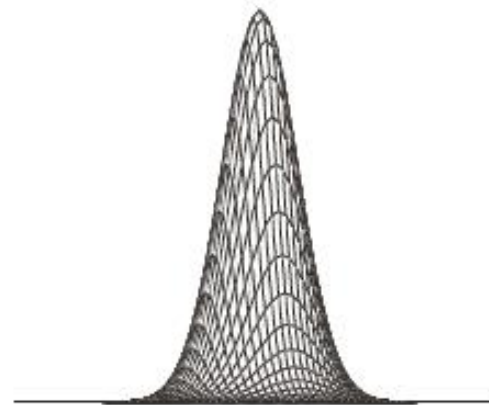
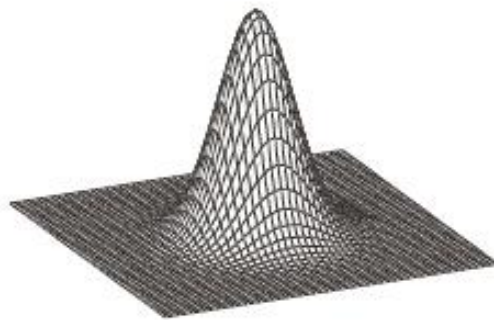
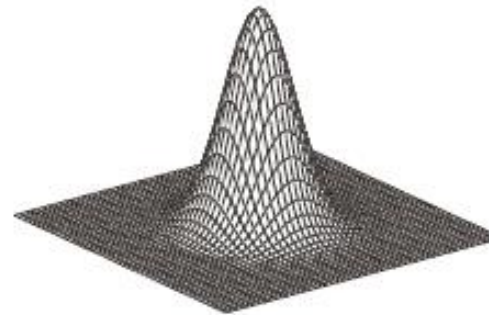
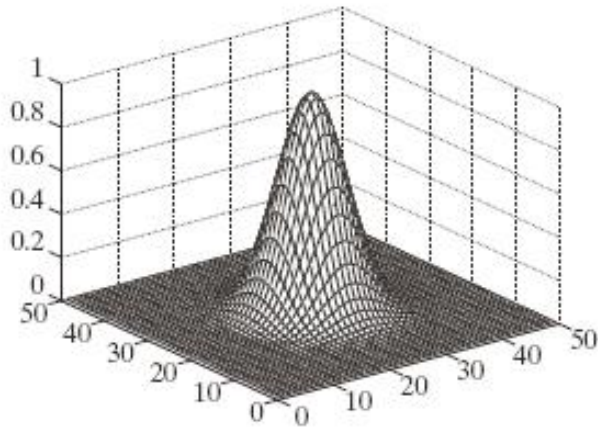
(a) Original  
image.

(b) Gaussian  
lowpass filter  
shown as an  
image.

(c) Spectrum of  
(a).

(d) Filtered  
image.

# Low Pass Filter



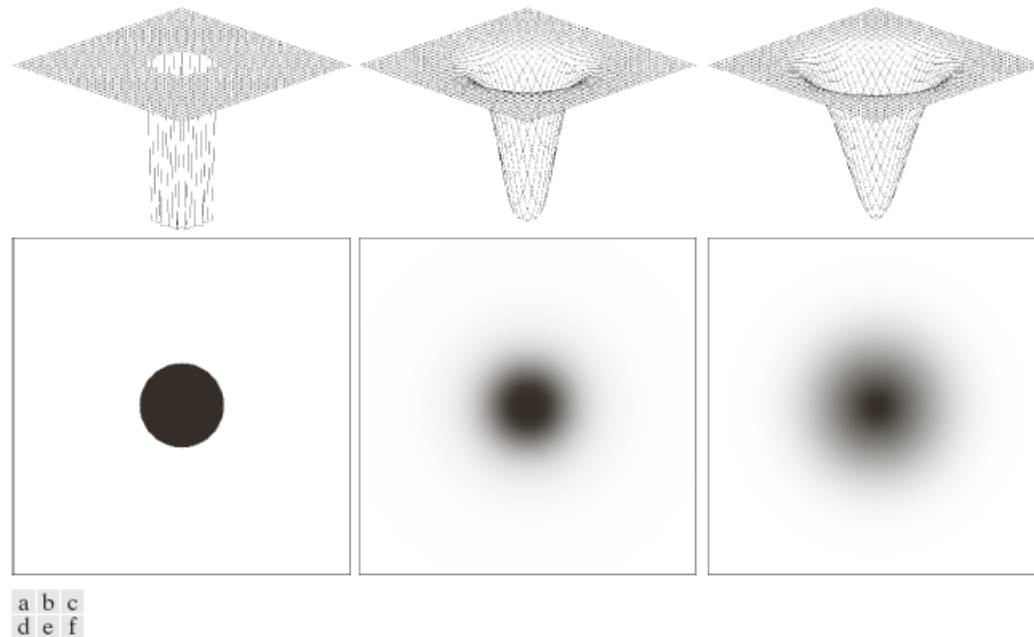
a	b
c	d

**FIGURE 4.15**

(a) A plot obtained using function mesh.  
 (b) Axes and grid removed. (c) A different perspective view obtained using function view.  
 (d) Another view obtained using the same function.

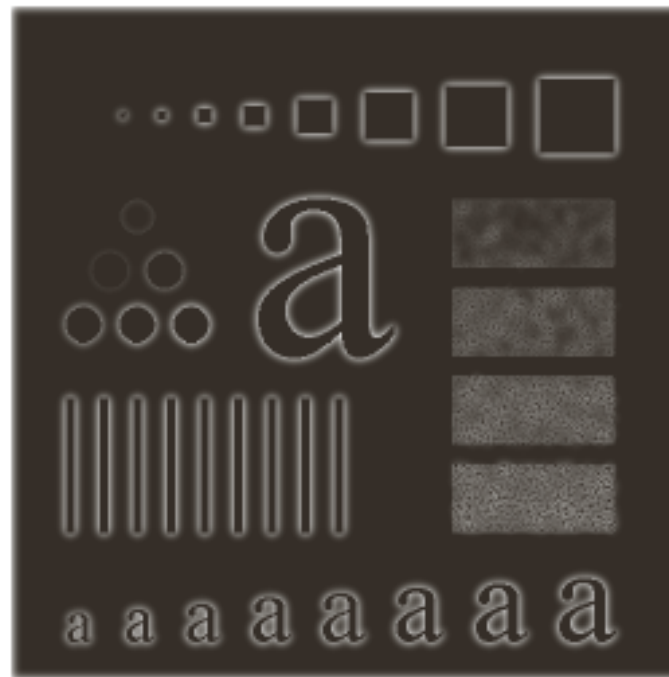
# High-pass Filter

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



**FIGURE 4.17** Top row: Perspective plots of ideal, Butterworth, and Gaussian highpass filters. Bottom row: Corresponding images. White represents 1 and black is 0.

# Output



a b

**FIGURE 4.18**

(a) Original image. (b) Result of Gaussian high-pass filtering.

# Band-pass filtering

## Gaussian Pyramid (low-pass images)



# Band Filters

- Low pass filtering is useful for reducing noise but may produce an image that is overly blurry.
- High pass filtering is useful for sharpening edges but also accentuates image noise.
- Band filtering seeks to retain the benefits of these techniques while reducing their undesirable properties.
- Band filtering isolates the mid-range frequencies from both the low-range and high-range frequencies.
  - A band stop (or notch) filter attenuates mid-level frequencies while leaving the high and low frequencies unaltered.
  - A band pass filter is the inverse of a band stop; leaving the mid-level frequencies unaltered while attenuating the low and high frequencies in the image.
- A band of frequencies may be specified by giving the center frequency and the width of the band. The band width determines the range of frequencies that are included in the band.





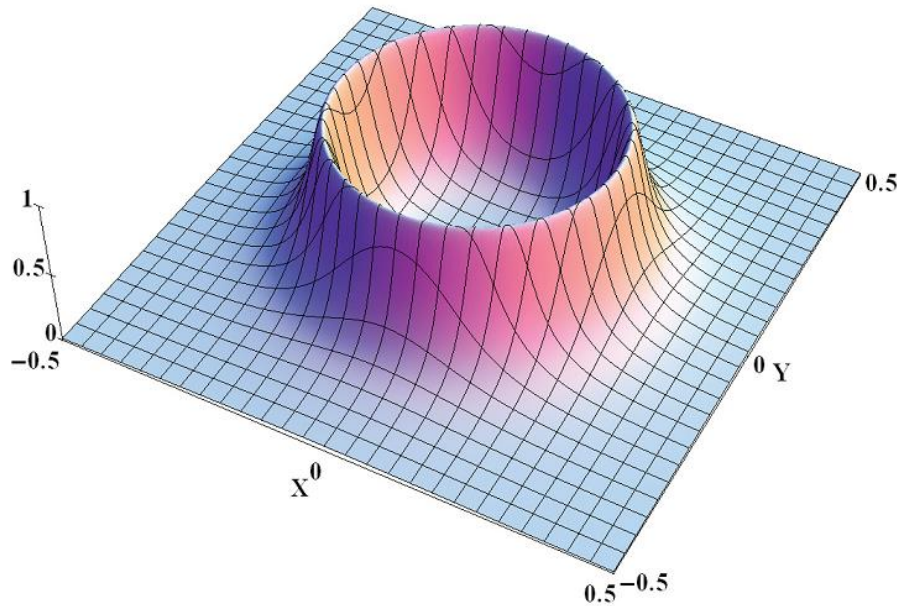
# Band filters

- A band stop filter is essentially a combination of a low and high pass filter, which implies that ideal, Butterworth, and Gaussian band stop filters can be defined.
- Equation 9.29 is the Butterworth band pass while Equation 9.30 is the Butterworth band stop.

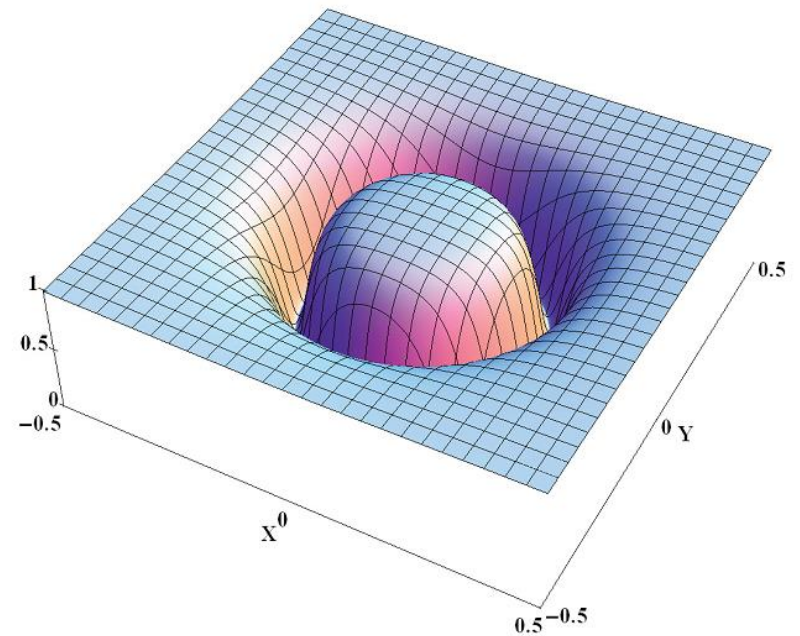
$$\mathcal{H}(u, v) = \frac{1}{1 + [\Omega * r(u, v) / (r(u, v)^2 - r_c^2)]^{2n}}, \quad (9.29)$$

$$\mathcal{H}(u, v) = 1 - \frac{1}{1 + [\Omega * r(u, v) / (r(u, v)^2 - r_c^2)]^{2n}}. \quad (9.30)$$

# Band pass and band stop



(a) Butterworth band pass.

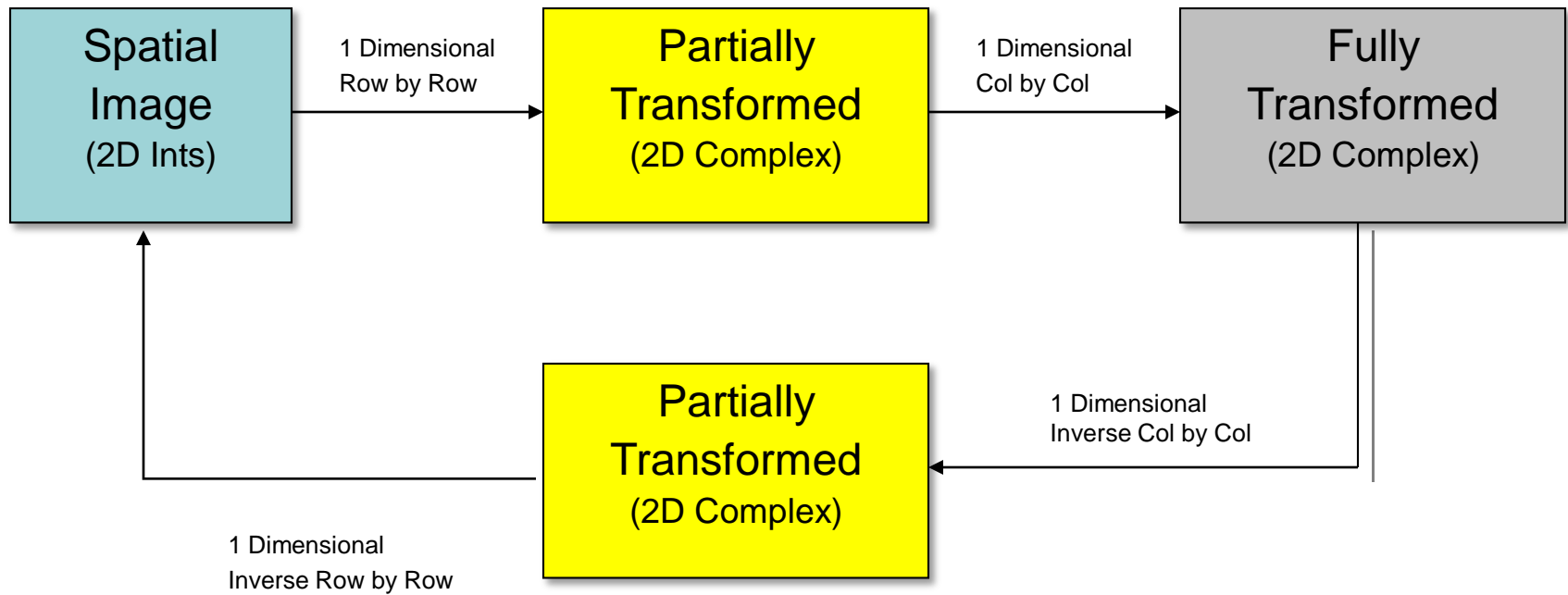


(b) Butterworth band stop.

**Figure 9.18.** Normalized Butterworth second-order band filters having a center frequency of .4 and a band width of .3.



# Separability

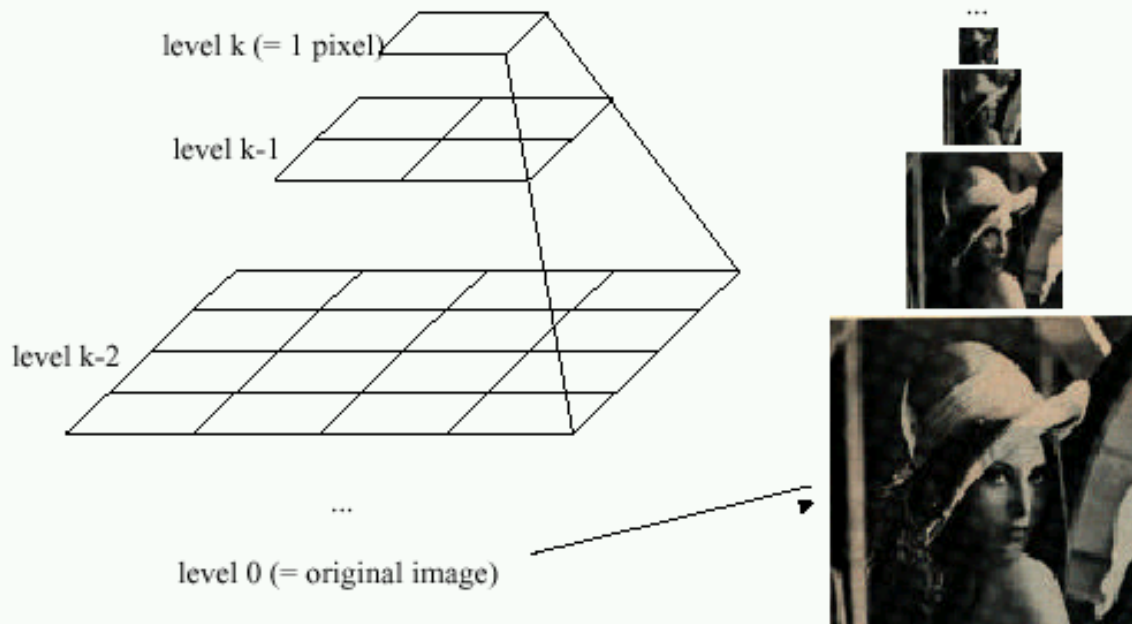


The Fourier Transform is separable. We can compute the 2d DFT (and inverse) by repeated applications of the 1D DFT

```
int main(int argc, char** argv){
    Mat A = imread("d:/image/cameraman2.tif", IMREAD_GRAYSCALE);
    Size patchSize(100, 100);
    Point topleft(A.cols / 2, A.rows / 2);
    Rect roi(topleft.x, topleft.y, patchSize.width, patchSize.height);
    Mat B = A(roi);
    int dft_M = getOptimalDFTSize(A.rows + B.rows - 1);
    int dft_N = getOptimalDFTSize(A.cols + B.cols - 1);
    Mat dft_A = cv::Mat::zeros(dft_M, dft_N, CV_32F);
    Mat dft_B = cv::Mat::zeros(dft_M, dft_N, CV_32F);
    Mat dft_A_part = dft_A(Rect(0, 0, A.cols, A.rows));
    Mat dft_B_part = dft_B(Rect(0, 0, B.cols, B.rows));
    A.convertTo(dft_A_part, dft_A_part.type(), 1, -mean(A)[0]);
    B.convertTo(dft_B_part, dft_B_part.type(), 1, -mean(B)[0]);
    dft(dft_A, dft_A, 0, A.rows);
    dft(dft_B, dft_B, 0, B.rows);
    mulSpectrums(dft_A, dft_B, dft_A, 0, true);
    idft(dft_A, dft_A, DFT_SCALE, A.rows + B.rows - 1);
    Mat corr = dft_A(Rect(0, 0, A.cols + B.cols - 1, A.rows + B.rows - 1));
    normalize(corr, corr, 0, 1, NORM_MINMAX, corr.type());
    pow(corr, 3., corr);
    B ^= cv::Scalar::all(255);
    imshow("Image", A);
    imshow("Correlation", corr);
    waitKey();
    return 0;
```

# Image Pyramids

Idea: Represent  $N \times N$  image as a “pyramid” of  $1 \times 1, 2 \times 2, 4 \times 4, \dots, 2^k \times 2^k$  images (assuming  $N = 2^k$ )

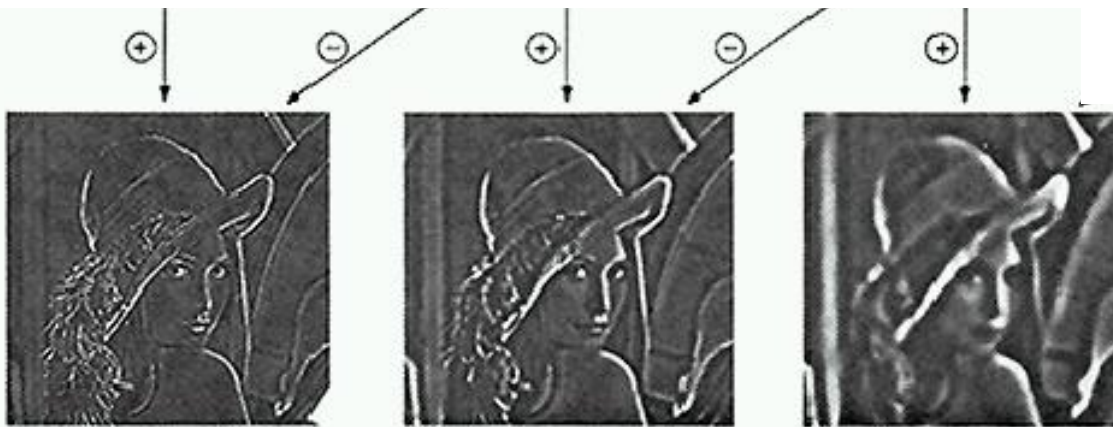


Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

# Laplacian Pyramid

Original  
image



- How can we reconstruct (collapse) this pyramid into the original image?





# What can you do with band limited imaged?



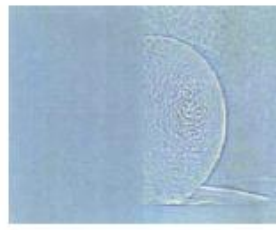


# Apples and Oranges in bandpass

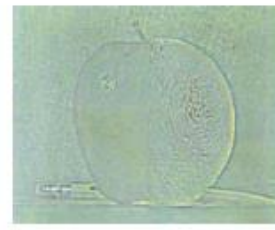
$L_0$



(a)



(b)

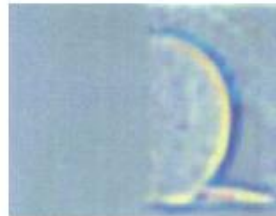


(c)

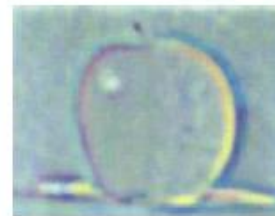
$L_2$



(d)



(e)

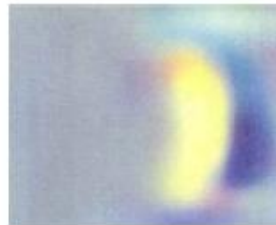


(f)

$L_4$



(g)



(h)



(i)

Reconstructed



(j)



(k)



(l)



# Applications of Fourier Transform

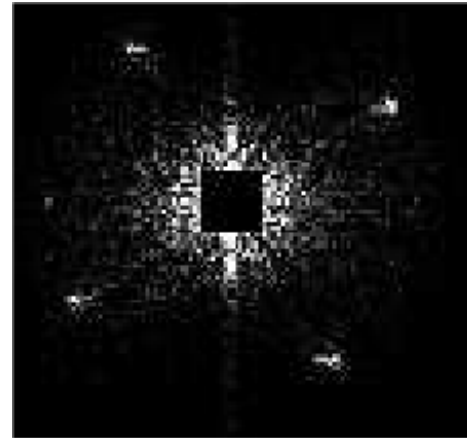
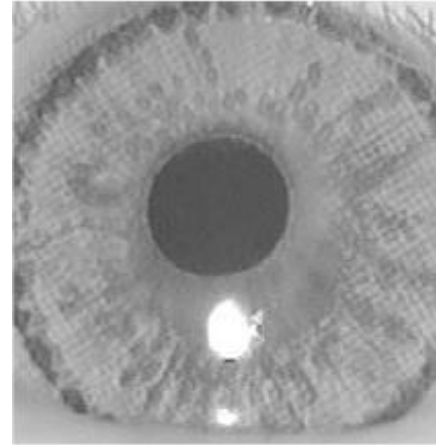
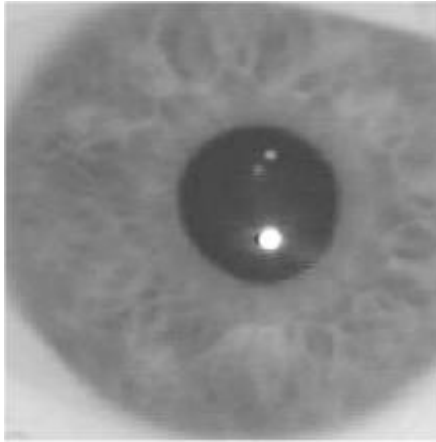
- Physics
  - Solve linear PDEs (heat conduction, Laplace, wave propagation)
- Antenna design
  - Seismic arrays, side scan sonar, GPS, SAR
- Signal processing
  - 1D: speech analysis, enhancement ...
  - 2D: image restoration, enhancement ...



# Not Just for EE

- Just like Calculus invented by Newton, Fourier analysis is another mathematical tool
- BIOM: fake iris detection
- CS: anti-aliasing in computer graphics
- CpE: hardware and software systems

# FT in Biometrics



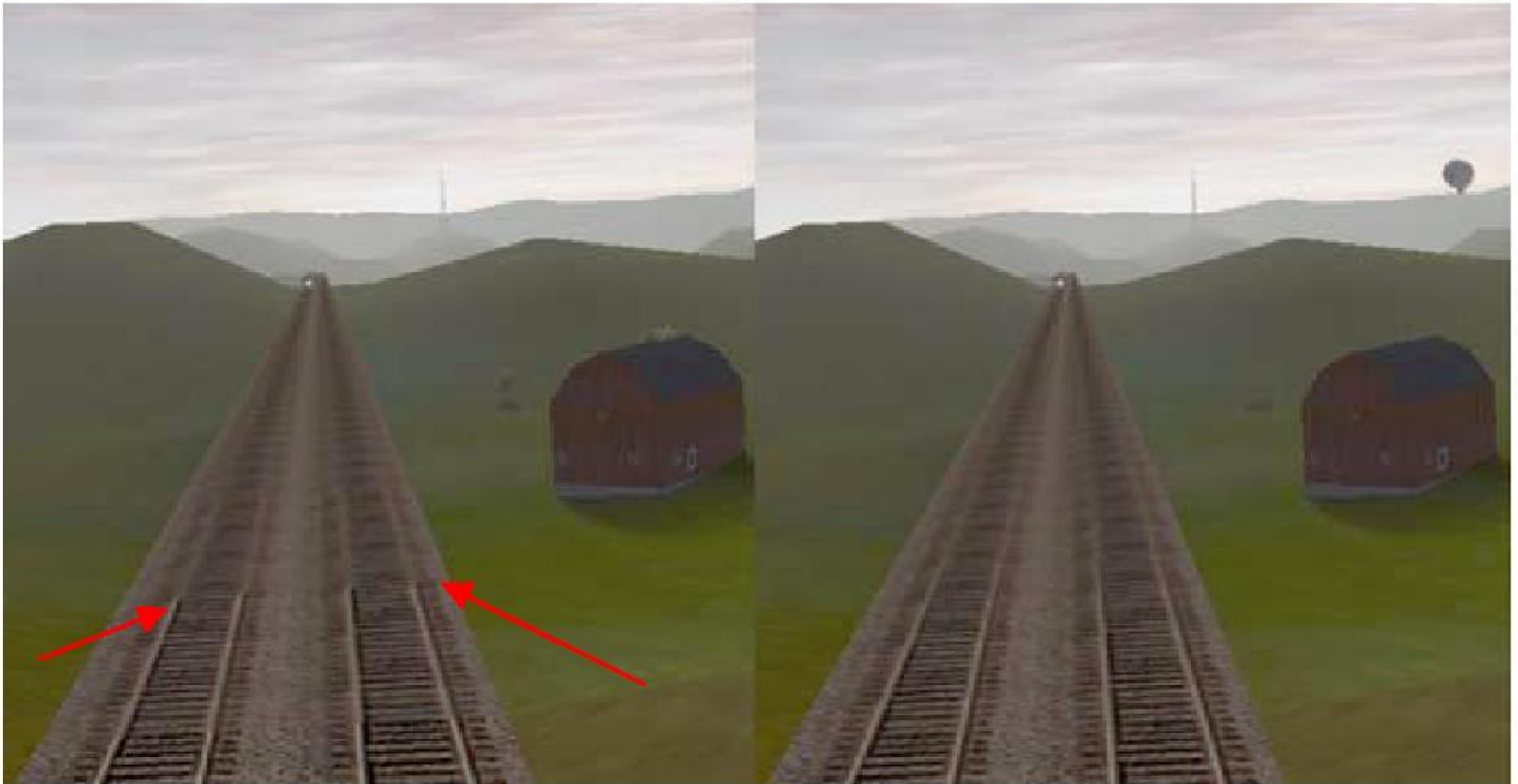
natural

fake





# FT in CS



Anti-aliasing in 3D graphic display

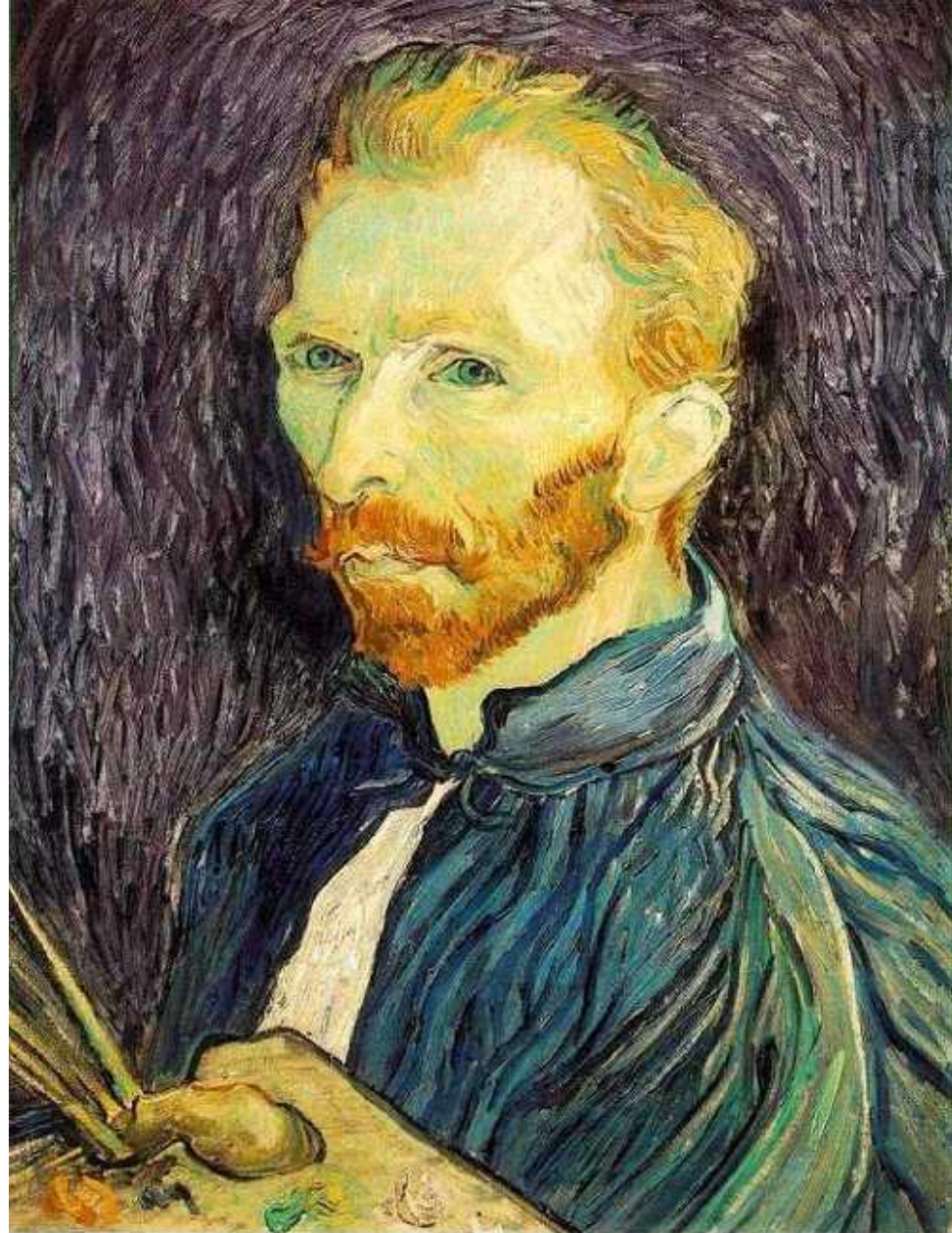




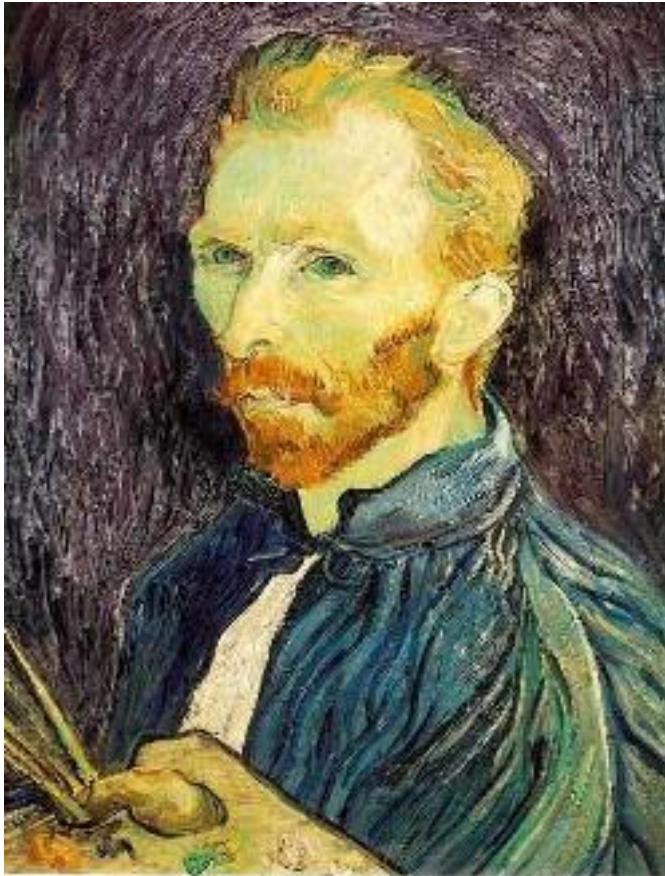
# Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?



# Image sub-sampling



1/4



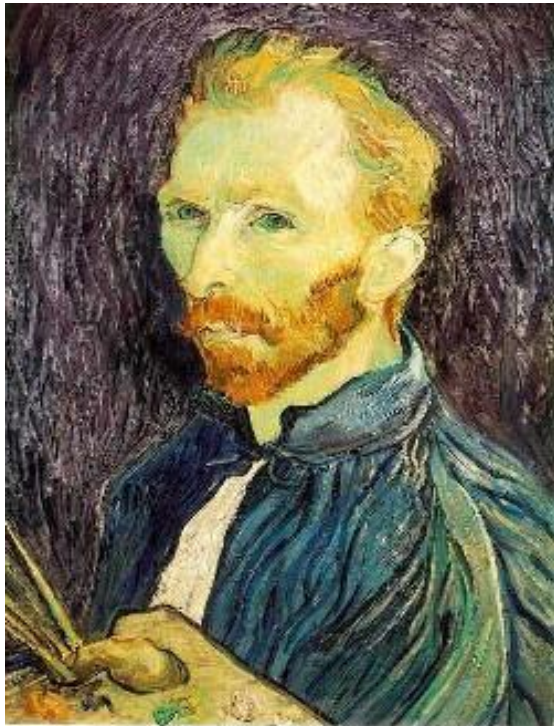
1/8

Throw away every other row and column to create a 1/2 size image called *image sub-sampling*





# Image sub-sampling



$1/2$



$1/4$  (2x zoom)

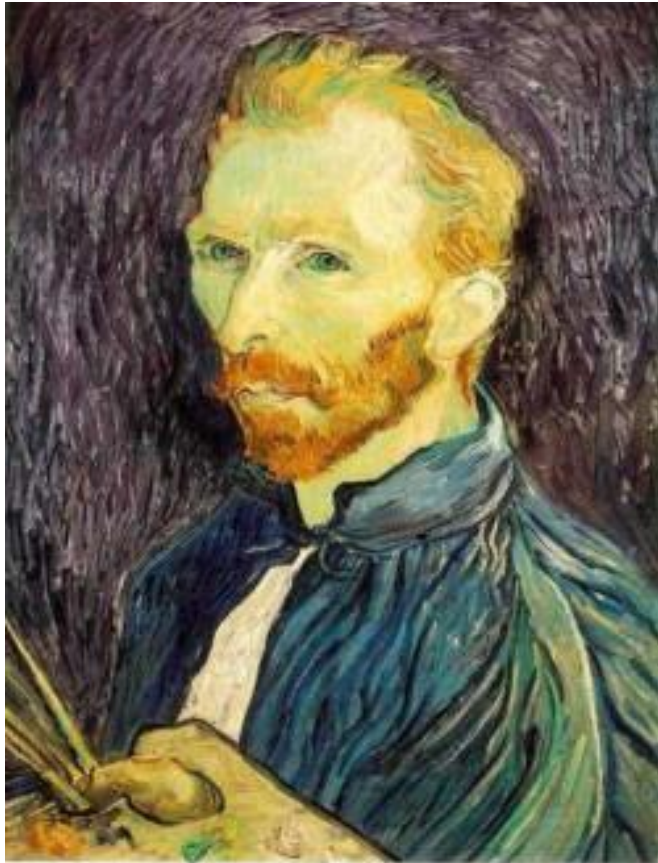


$1/8$  (4x zoom)

Aliasing! What do we do?



# Gaussian (lowpass) pre-filtering



Gaussian 1/2



G 1/4



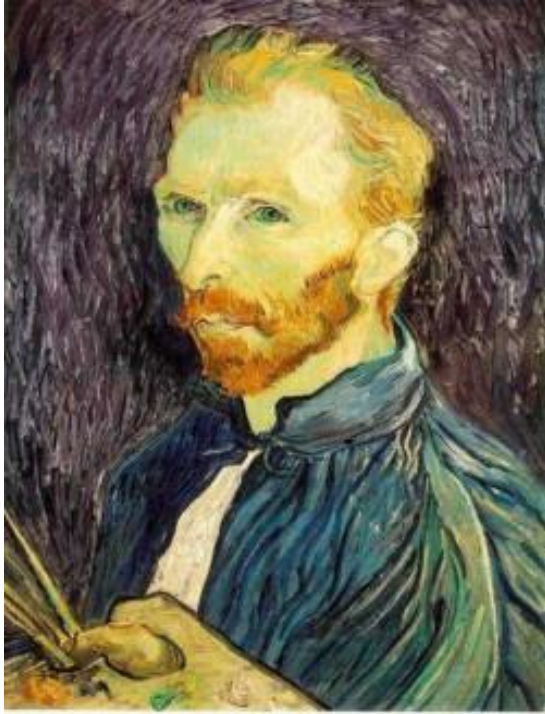
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each  $\frac{1}{2}$  size reduction. Why?



# Subsampling with Gaussian pre-filtering



Gaussian  $1/2$



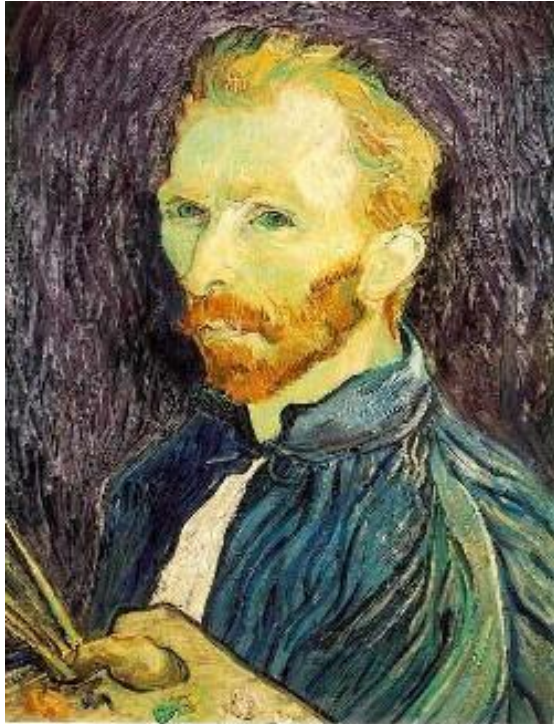
G  $1/4$



G  $1/8$



# Compare with...



$1/2$



$1/4$  (2x zoom)



$1/8$  (4x zoom)





# Questions?

