CS4670/5670: Computer Vision

Kavita Bala

Lecture 4: Frequency Analysis

Announcements

- PA 1 will be out later this week
  - Will be split into 2 parts
  - due in 1 week each
  - to be done in groups of two – please form your groups ASAP
- Piazza: make sure you sign up
  - Post on piazza privately to staff
- CMS: mail to Megan Gatch (mlg34@cornell.edu)

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Discrete filtering in 2D

- Same equation, one more index

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

  – now the filter is a rectangle you slide around over a grid of numbers
- Commonly applied to images
  – blurring (using box, gaussian, …)
  – sharpening
- Usefulness of associativity
  – often apply several filters one after another:
    - $(((a * b_1) * b_2) * b_3)$
  – this is equivalent to applying one filter:
    - $a * (b_1 * b_2 * b_3)$

And in pseudocode…

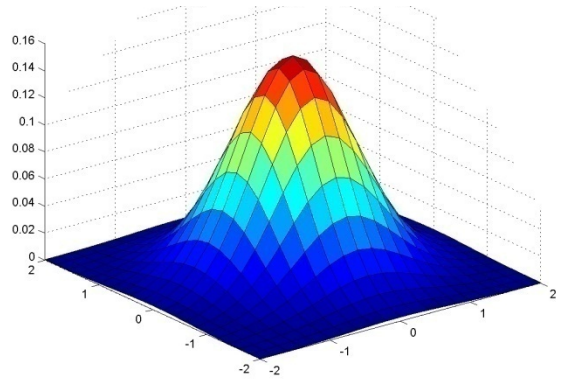
function convolve2d(filter2d a, filter2d b, int i, int j)
$s = 0$
$r = a$.radius
for $i' = -r$ to r **do**
  for $j' = -r$ to r **do**
    $s = s + a[i'][j']b[i - i'][j - j']$
return s

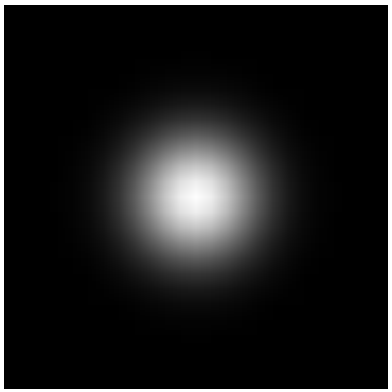
A gallery of filters

- Box filter
  - –Simple and cheap
- Tent filter
  - –Linear interpolation
- Gaussian filter
  - –Very smooth antialiasing filter
- B-spline cubic
  - –Very smooth
- ...

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$





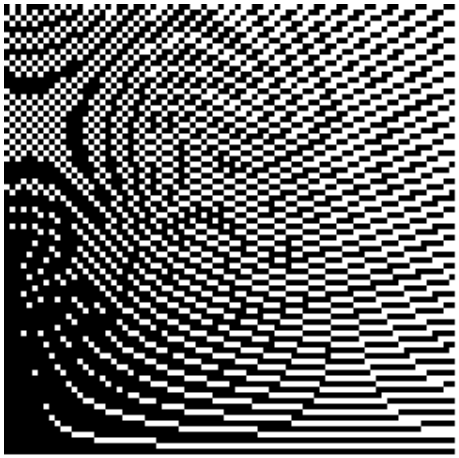
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

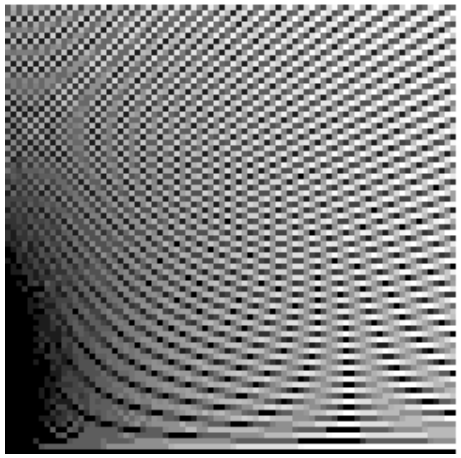
- Constant factor at front makes volume sum to 1 (can be ignored, as we should re-normalize weights to sum to 1 in any case)

Source: C. Rasmussen

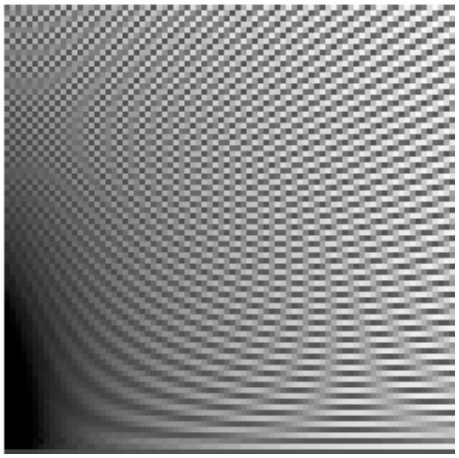
Filter comparison



Point sampling

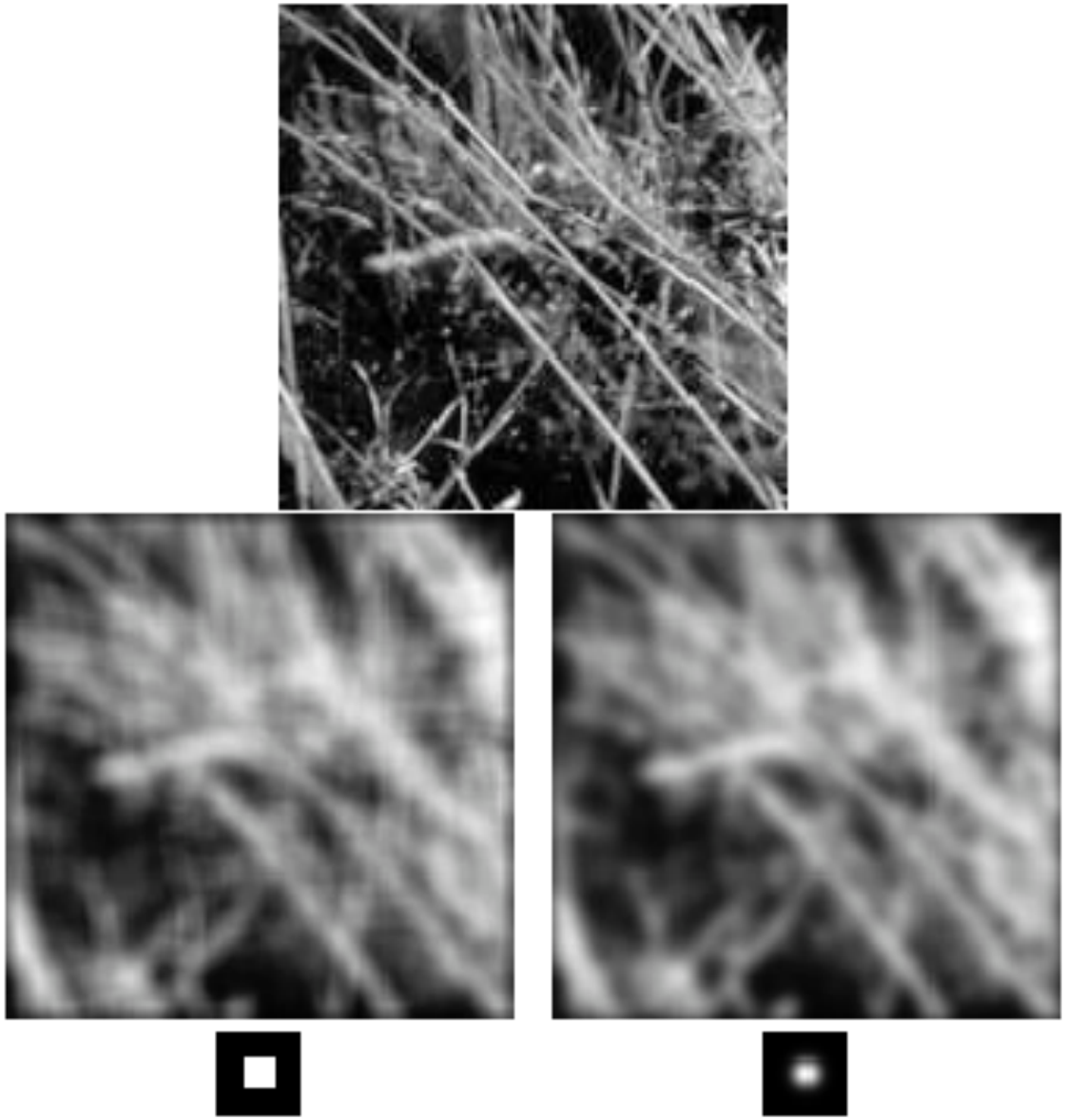


Box filtering



Gaussian filtering

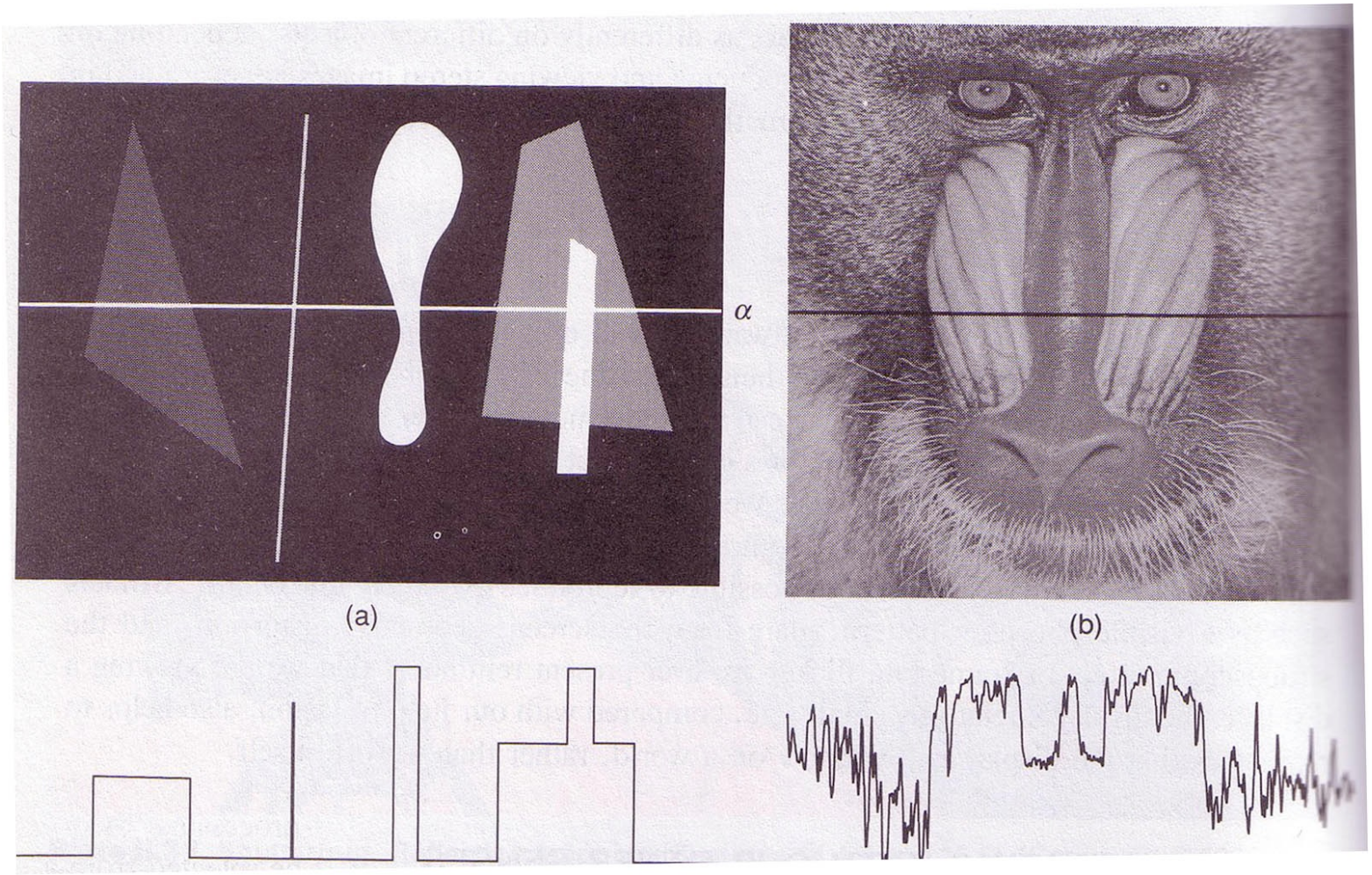
Mean vs. Gaussian filtering



But what are these filters?

- Theory that explains the why and what
- Fourier analysis

Examples

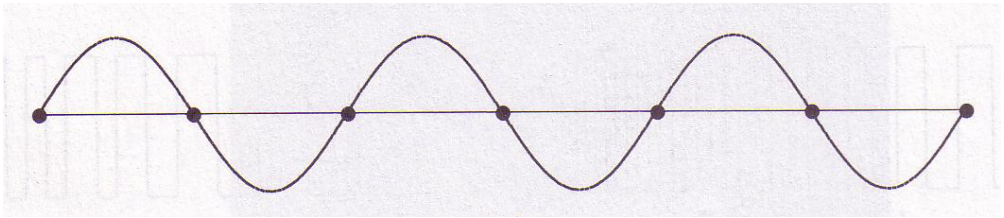


Fourier Analysis

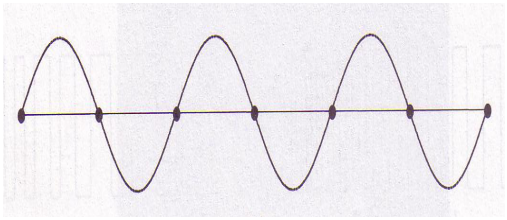
- Every signal has some frequency associated with it
- Fourier analysis finds the frequencies associated with a signal

Periodic Signal

- Consider a (co)sine wave



- A (co)sine wave has a frequency

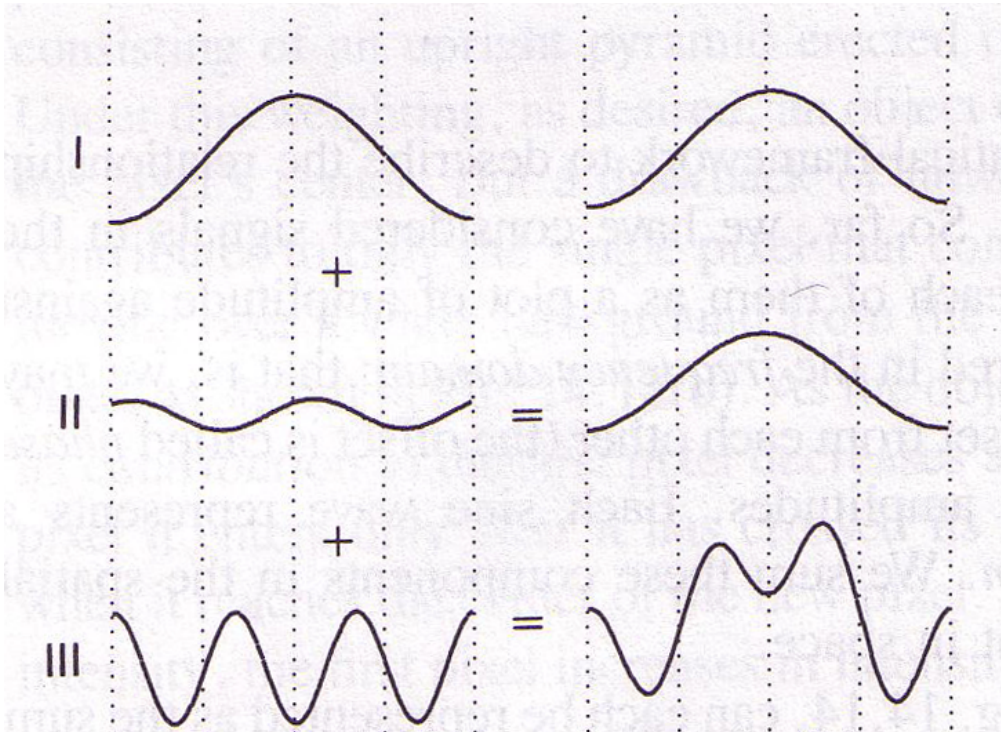


Fourier Analysis

- Every signal has some frequency associated with it
- Fourier analysis finds the frequencies associated with a signal
- For example, for the (co)sine waves
  - 4 pixels/cycle (frequency: 0.25 cycles/pixel)
  - 2 pixels/cycle (frequency: 0.5 cycles/pixel)

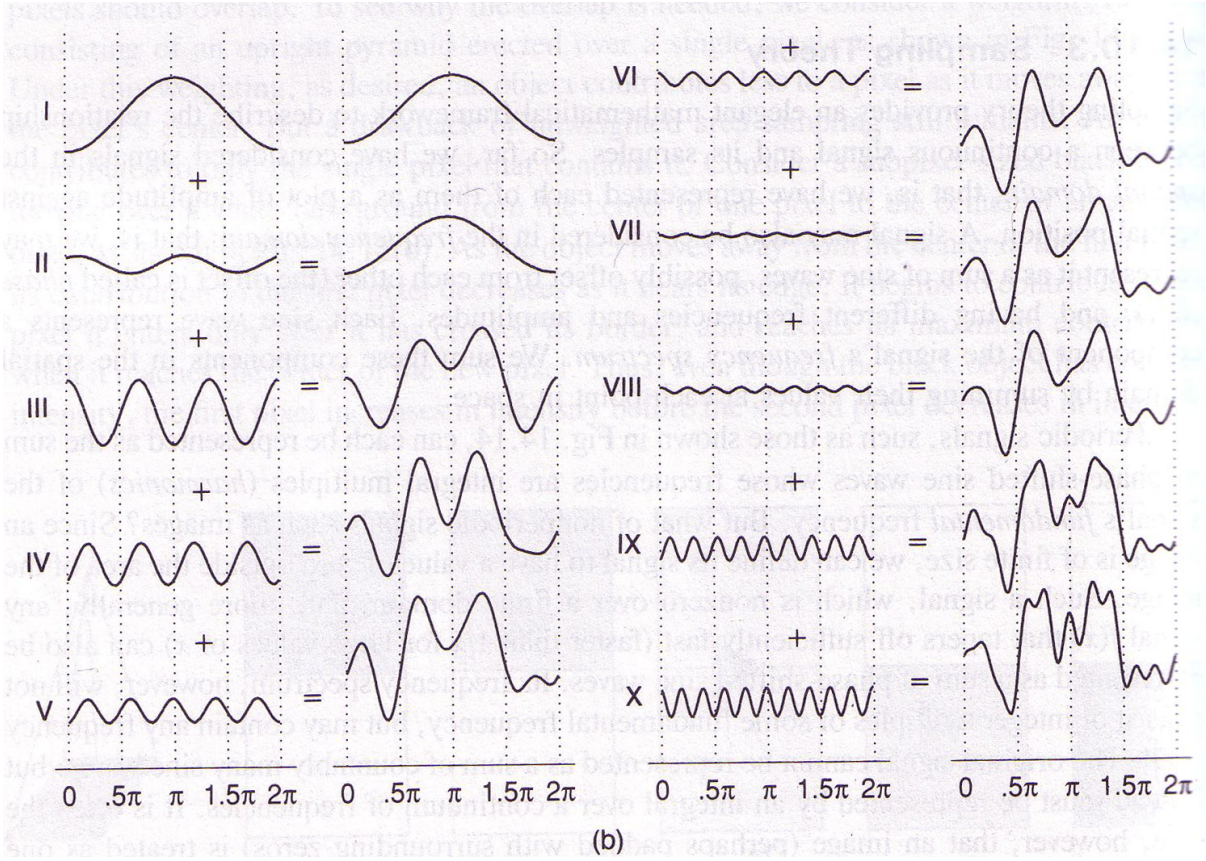
Idea of Fourier Analysis

- Every signal (doesn't matter what it is)
  - Sum of sine/cosine waves

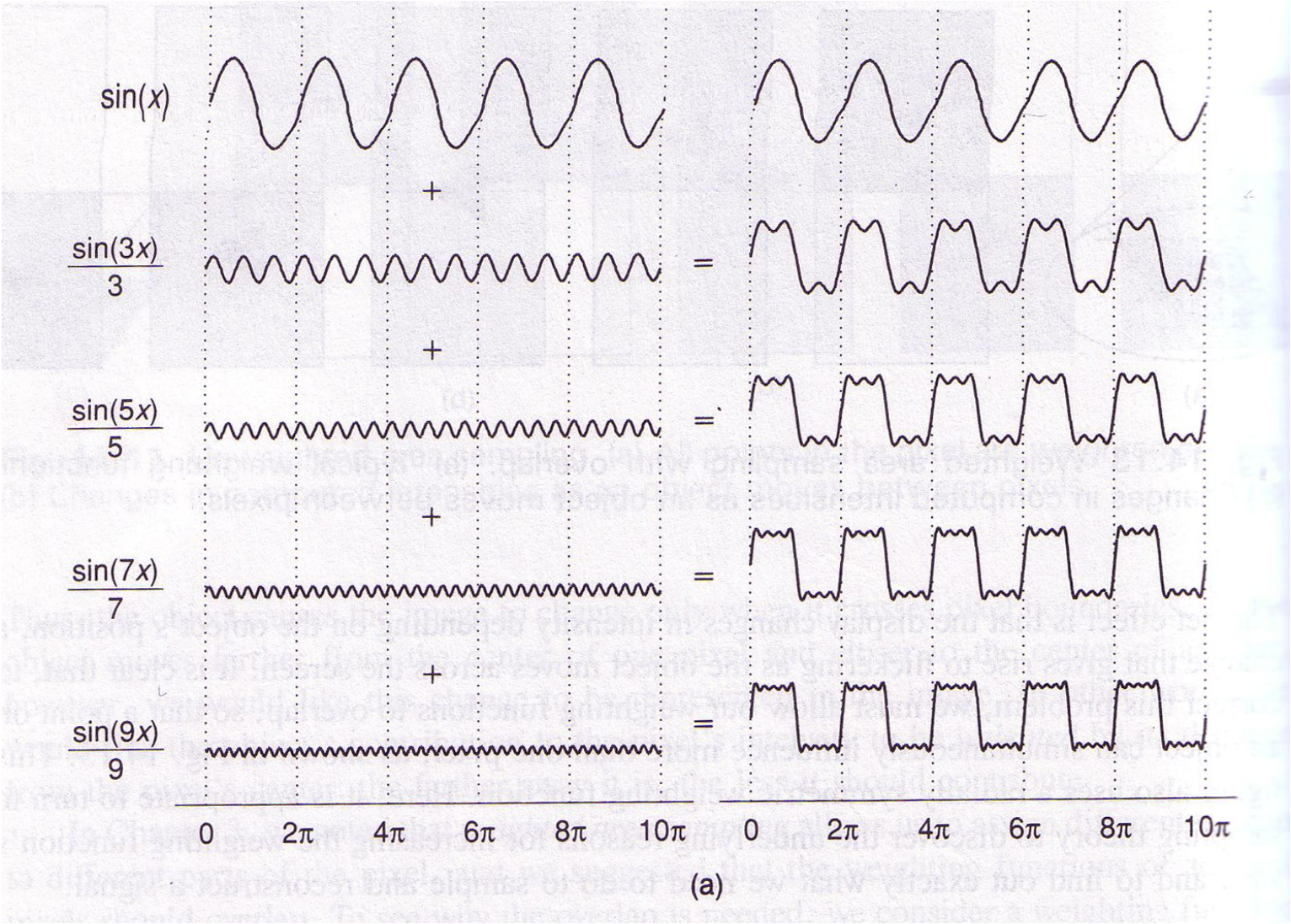


Idea of Fourier Analysis

- Every signal (doesn't matter what it is)
  - Sum of sine/cosine waves



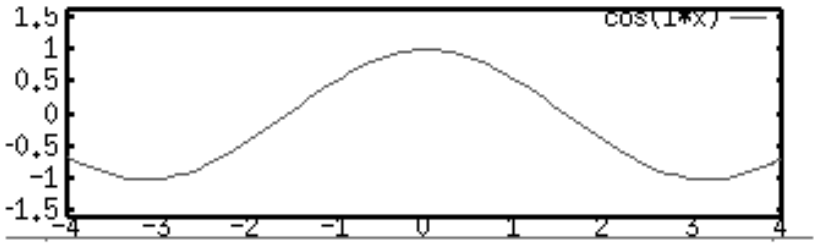
A box-like example

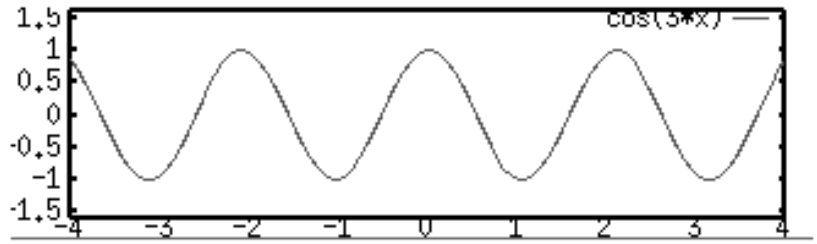


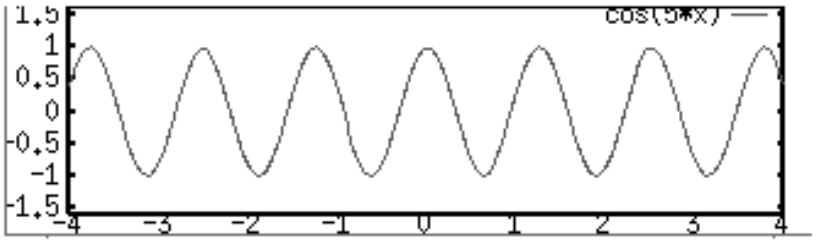
Signal as sum of (co)sines

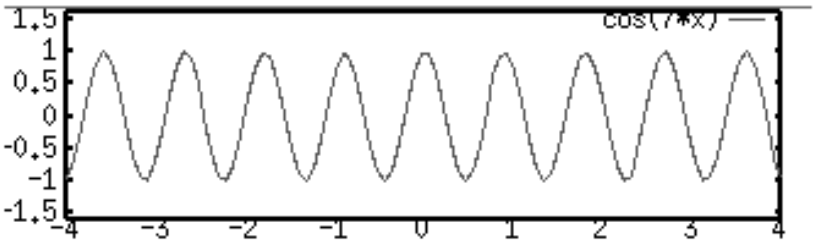
- Could be an infinite sum of sine/cosine waves
  - In fact, has to be infinite if signal is not periodic

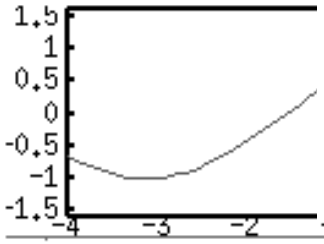
- Fourier Analysis finds out what these frequencies are
  - The more angular/discontinuous the function is, the more frequencies we need


cos(1*x)
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

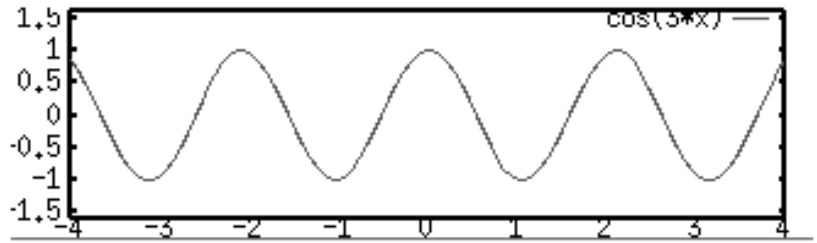

cos(3*x)
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

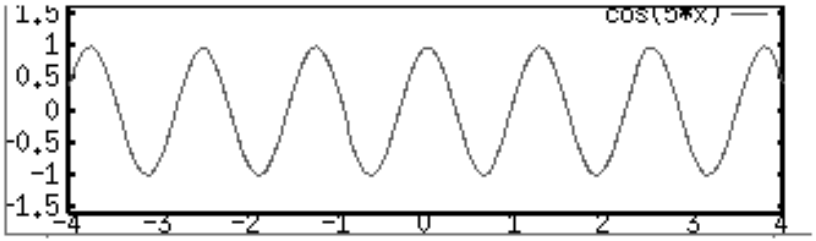

cos(5*x)
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

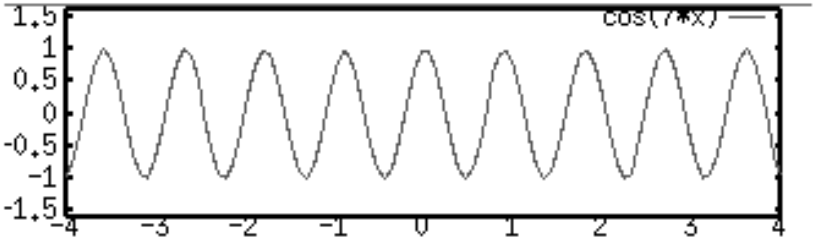

cos(7*x)
1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

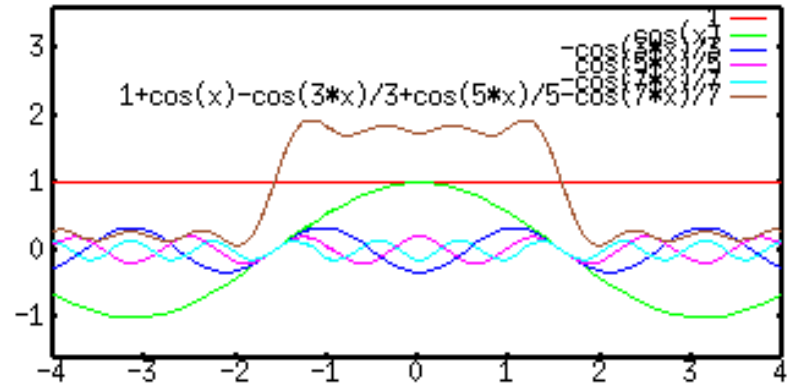


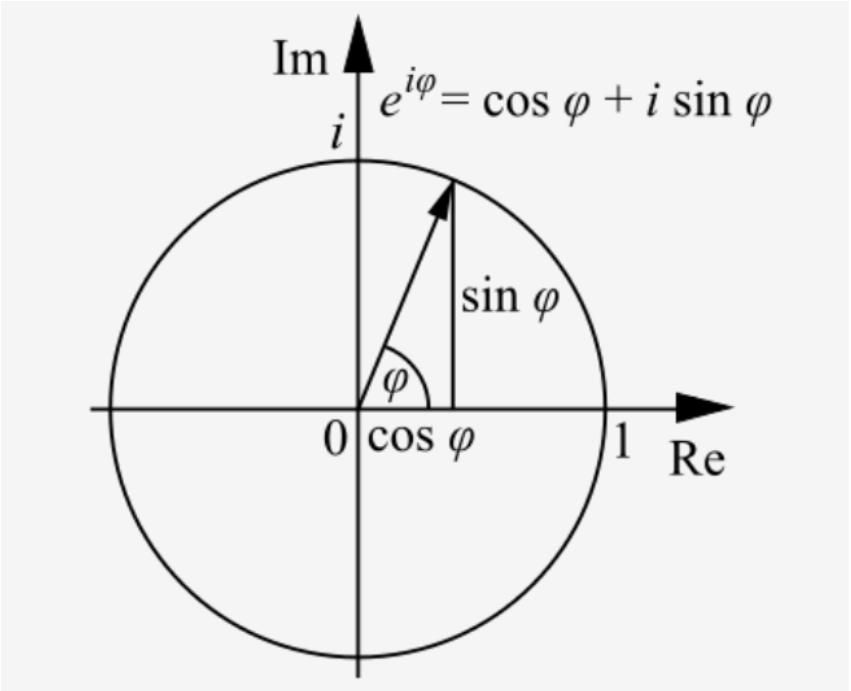
$$x(t) = cos(0t) + cos(t) - \frac{1}{3}cos(3t) + \frac{1}{5}cos(5t) - \frac{1}{7}cos(7t)$$











Euler's formula, named after Leonhard Euler, is a mathematical formula in complex analysis that establishes the fundamental relationship between the trigonometric functions and the complex exponential function. Euler's formula states that, for any real number x:

$$e^{ix} = \cos x + i\sin x$$

$$x(t) = cos(0t) + cos(t) - \frac{1}{3}cos(3t) + \frac{1}{5}cos(5t) - \frac{1}{7}cos(7t)$$

$$\begin{cases} cos(\omega_0 t) = [e^{j\omega_0 t} + e^{-j\omega_0 t}]/2 \\ sin(\omega_0 t) = [e^{j\omega_0 t} - e^{-j\omega_0 t}]/2j \end{cases}$$

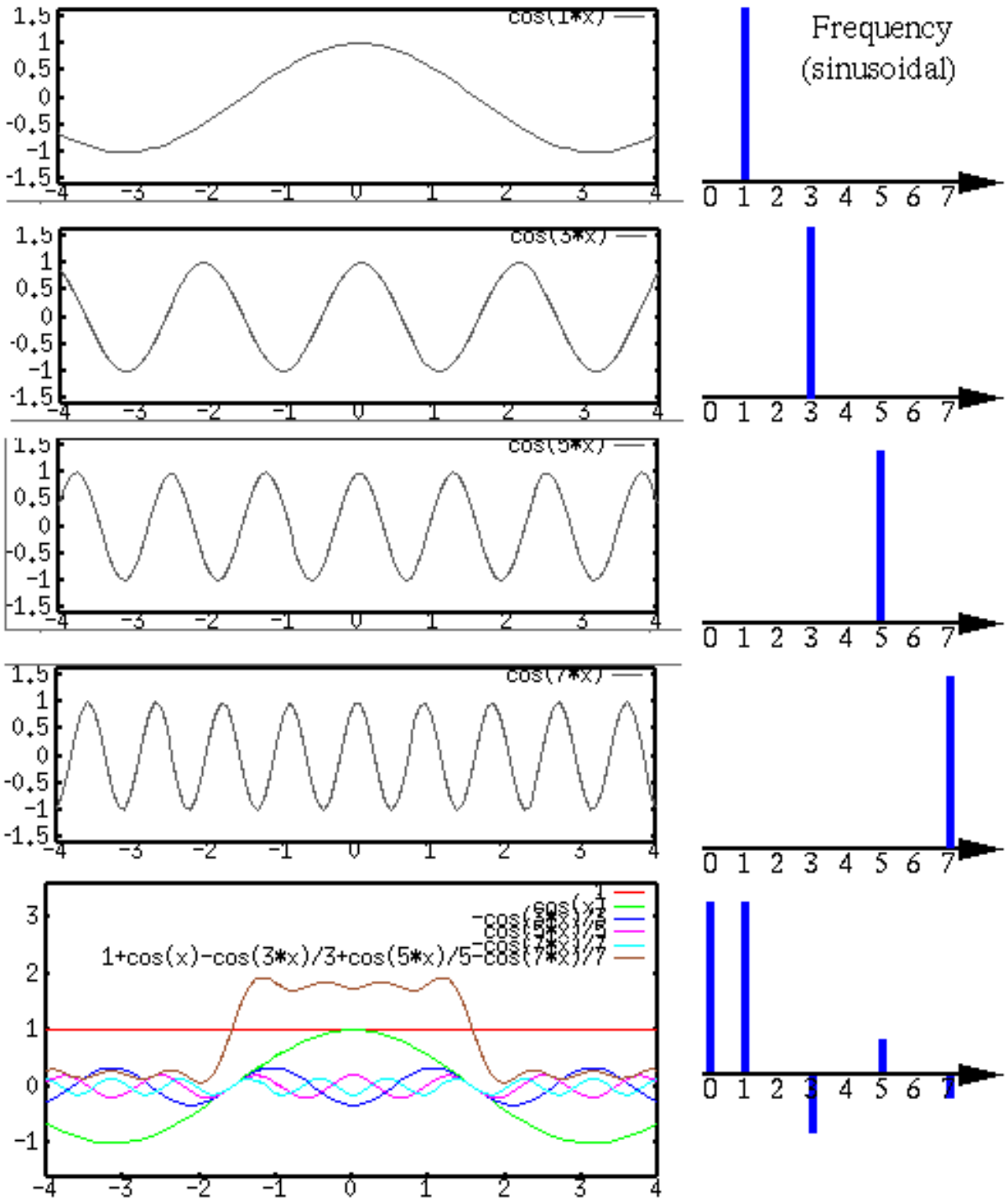
i is same as j

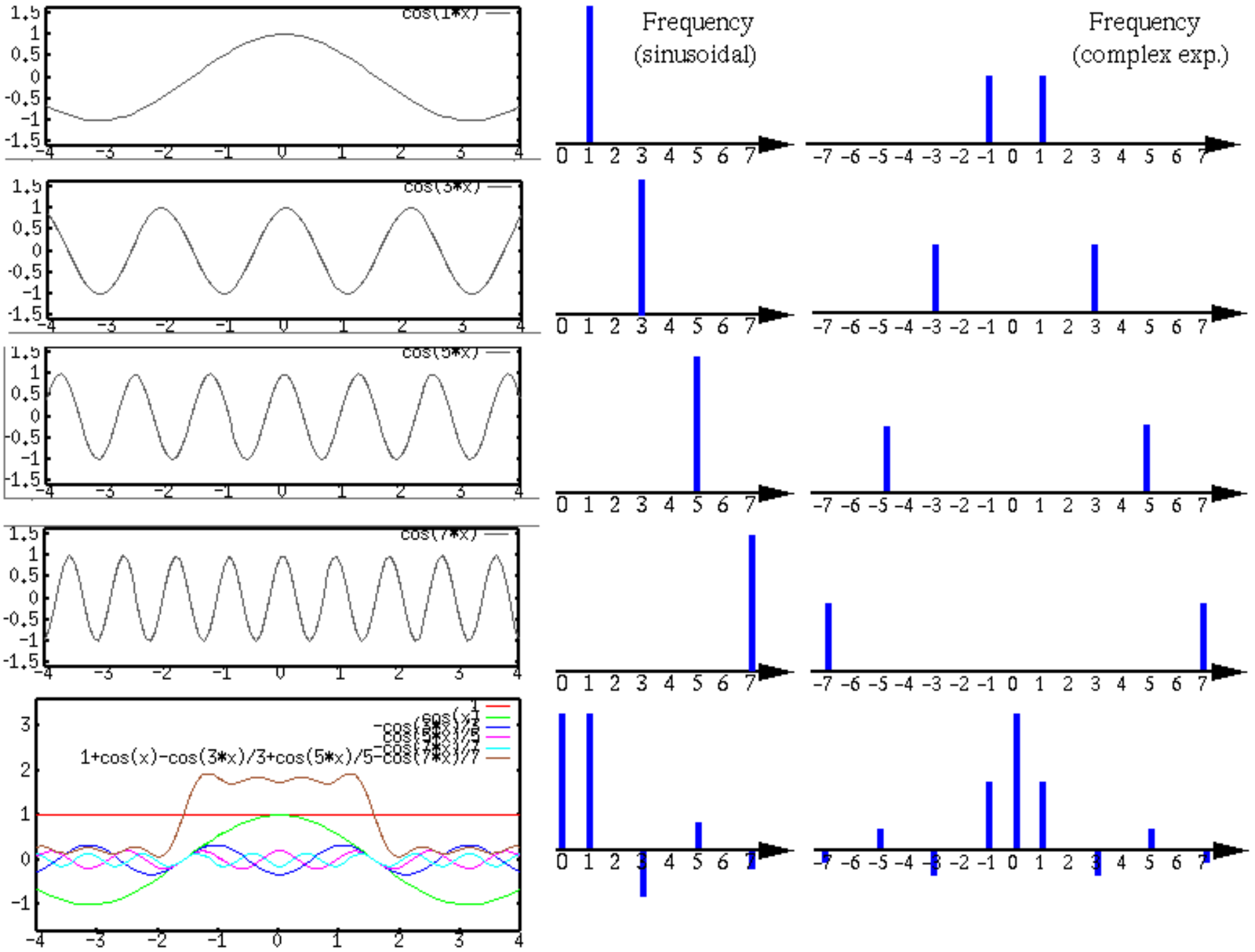
x = ω t

$$x(t) = e^{0t} + \frac{1}{2}[(e^{jt} + e^{-jt}) - \frac{1}{3}(e^{j3t} + e^{-j3t}) + \frac{1}{5}(e^{j5t} + e^{-j5t}) - \frac{1}{7}(e^{j7t} + e^{-j7t})] = \sum_{k=-7}^{7} X[k]e^{jk\omega_0 t}$$

$$X[0] = 0; \;\; X[1] = X[-1] = 1/2, \;\; X[3] = X[-3] = 1/6, \;\; X[5] = X[-5] = 1/10,$$

$$X[7] = X[-7] = 1/14, \;\; X[2] = X[-2] = X[4] = X[-4] = X[6] = X[-6] = 0$$

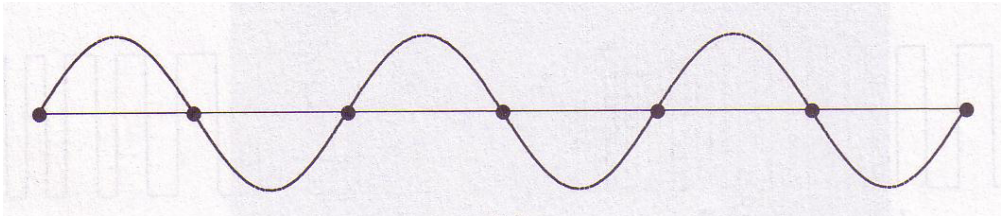

cos(1*x)
cos(3*x)
cos(5*x)
cos(7*x)
1+cos(x)-cos(3*x)/3+cos(5*x)/5-cos(7*x)/7
Frequency
(sinusoidal)
0 1 2 3 4 5 6 7

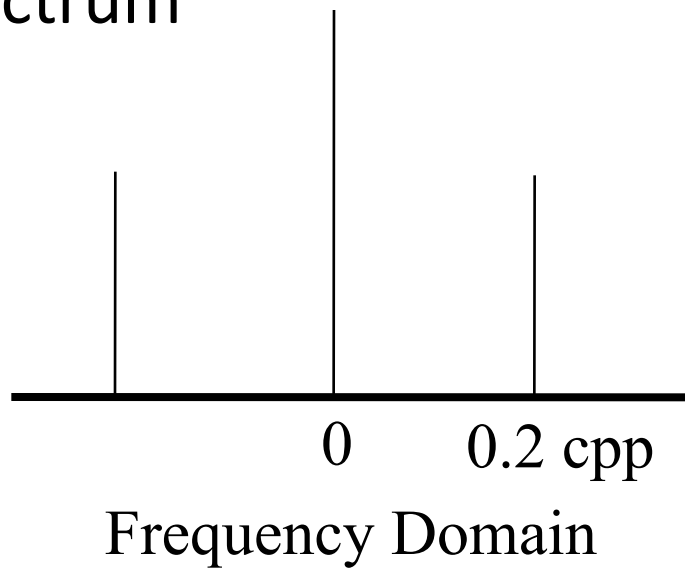

cos(1*x)
cos(3*x)
cos(5*x)
cos(7*x)
1+cos(x)-cos(3*x)/3+cos(5*x)/5-cos(7*x)/7
Frequency
(sinusoidal)
Frequency
(complex exp.)

Fourier Analysis: Dual Spaces

- Signal: spatial domain

- Fourier Transform: Frequency domain
  - Amplitudes are called spectrum





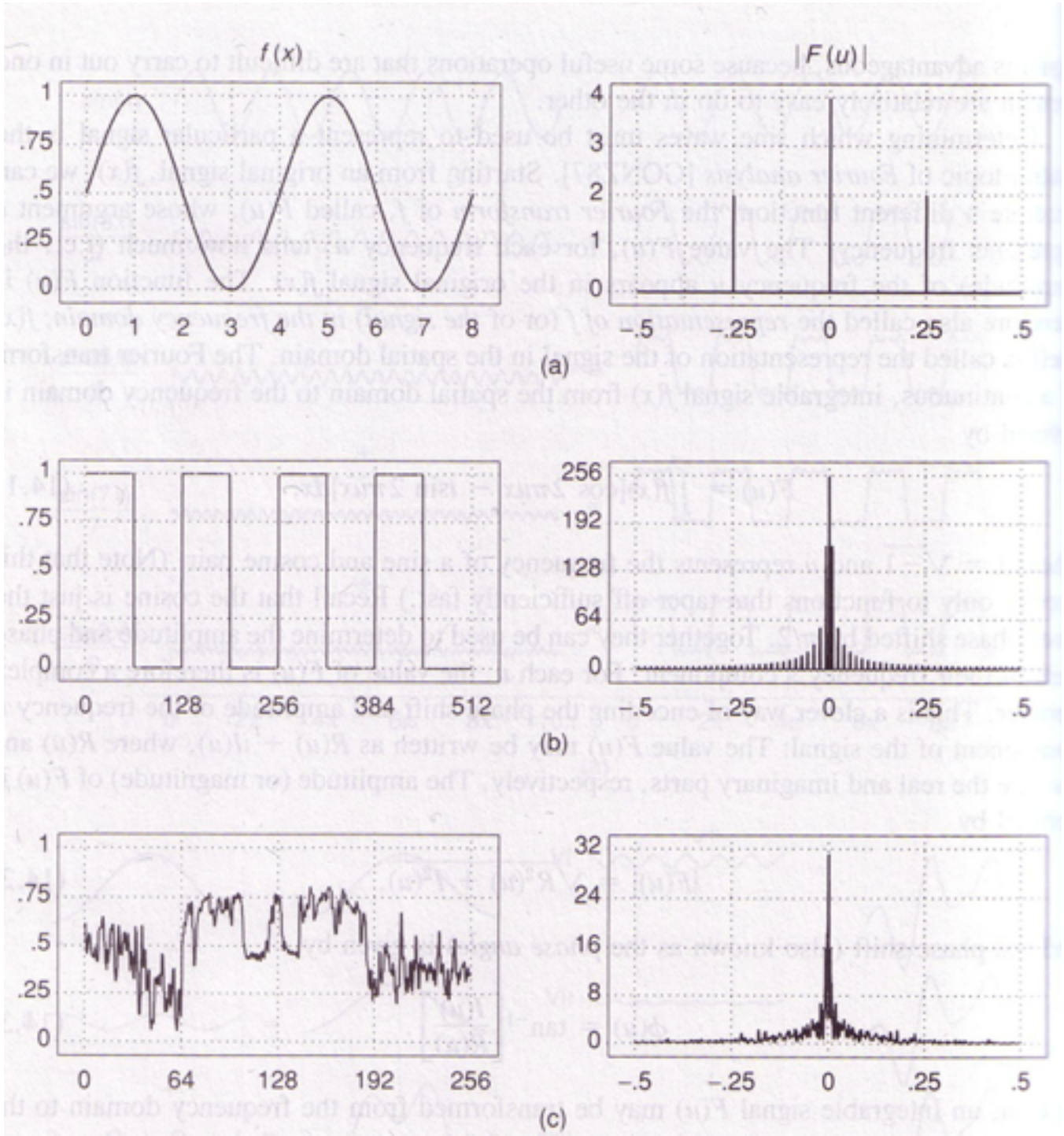
Fourier Transform

- Given a signal compute (co)sine waves that contribute to signal
  - For each frequency k, contribution of that frequency to signal: X[k]

$$x_T(t) = \sum_{k=0}^{\infty} X'[k] cos(k\omega_0 t) = \sum_{k=-\infty}^{\infty} X[k] e^{jk\omega_0 t}$$

$$X[k] = \frac{1}{T} \int_T x_T(t) e^{-jk\omega_0 t} dt = \frac{1}{T} \int_T x_T(t) e^{-j2\pi k f_0 t} dt$$

Space



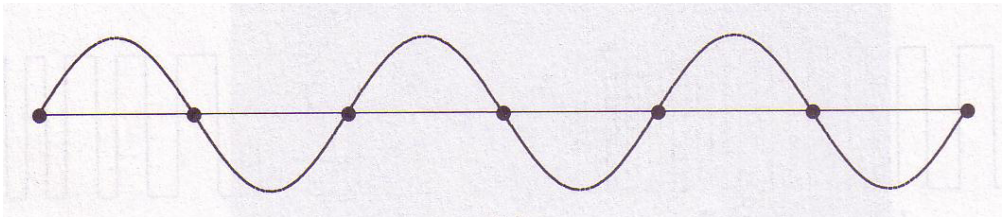
Frequency

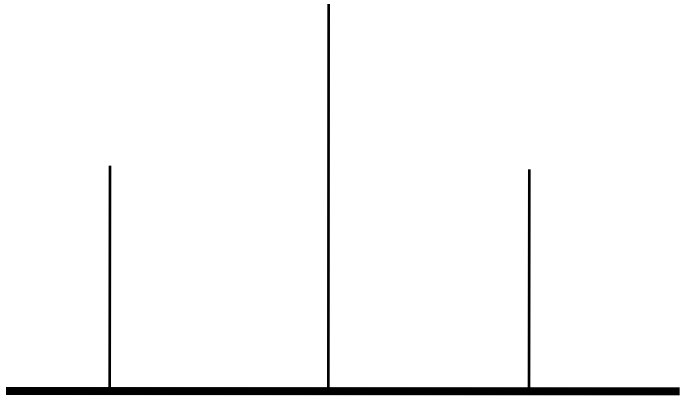
A gallery of filters

- Box filter
  - –Simple and cheap
- Tent filter
  - –Linear interpolation
- Gaussian filter
  - –Very smooth antialiasing filter
- B-spline cubic
  - –Very smooth
- ...

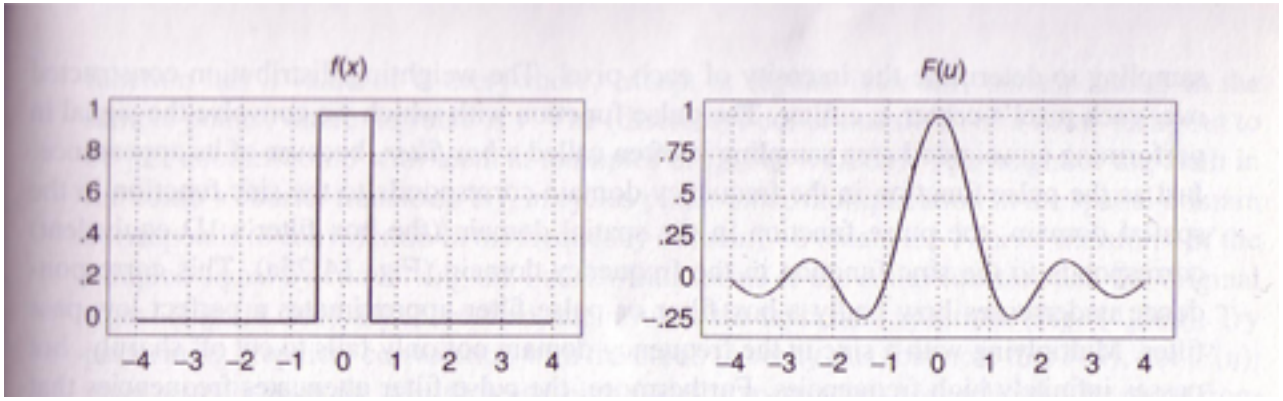
Types of Filters

Spatial	⟺	Frequency
• Sine		• Impulse
• Gaussian		• Gaussian
• Box		• Sinc





Frequency Domain


f(x)
1
.8
.6
.4
.2
0
–4 –3 –2 –1 0 1 2 3 4
F(u)
1
.75
.5
.25
0
–.25
–4 –3 –2 –1 0 1 2 3 4

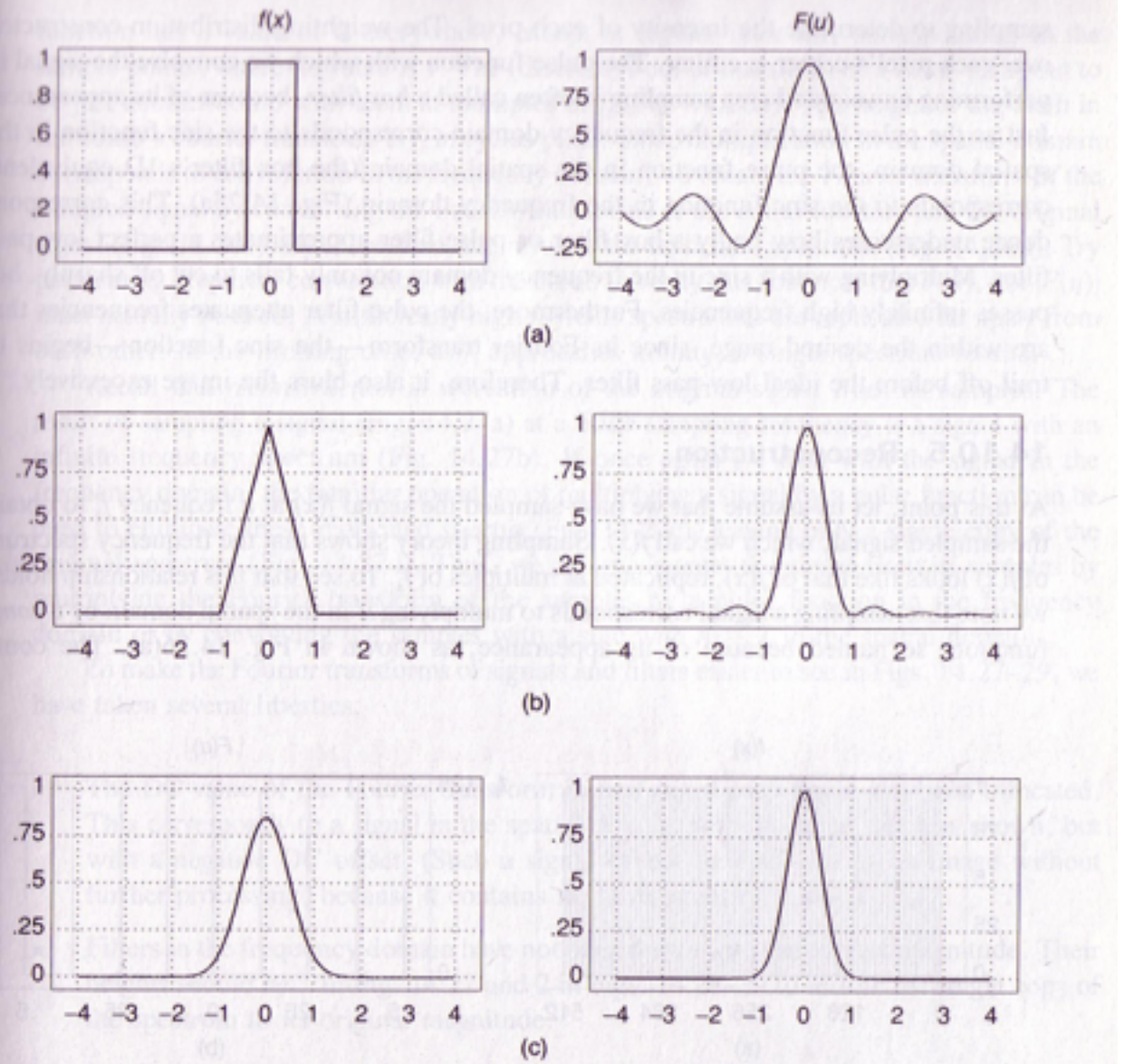
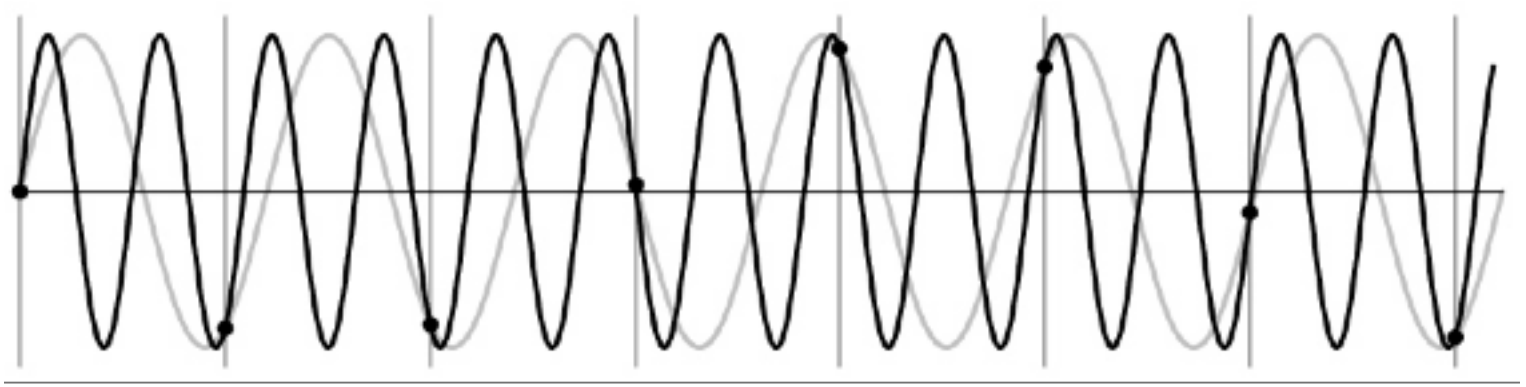


Fig. 14.25 Filters in spatial and frequency domains. (a) Pulse—sinc. (b) Triangle—sinc^2. (c) Gaussian—Gaussian. (Courtesy of George Wolberg, Columbia University.)

How to sample an image?

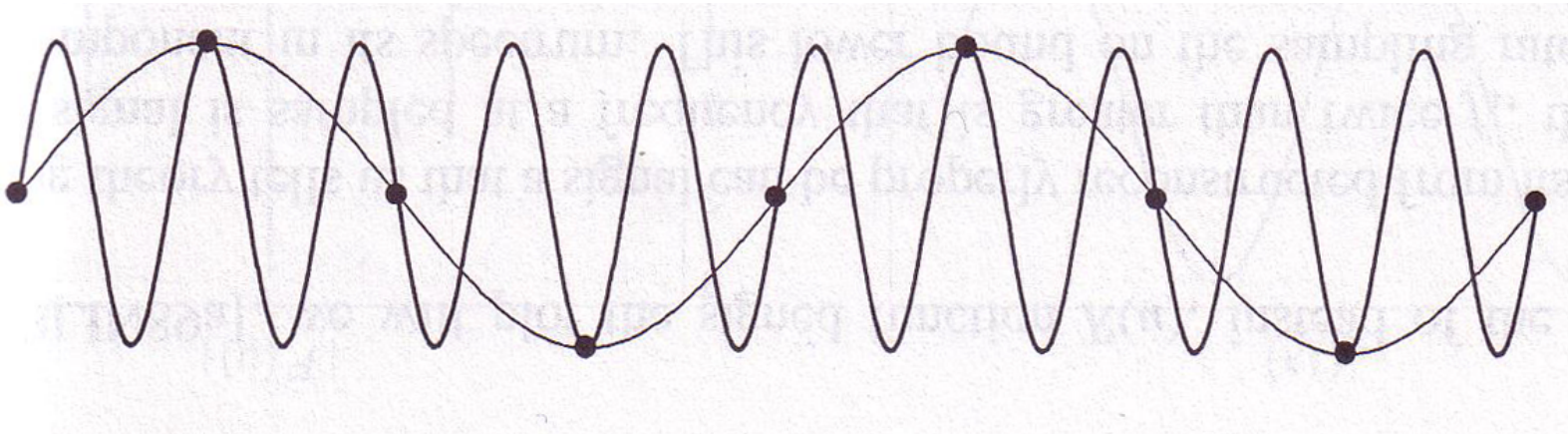
- An image is some set of samples of function
- How many samples do you need to accurately capture original image?
- Nyquist says that we should sample at $2f_{max}$

Undersampling

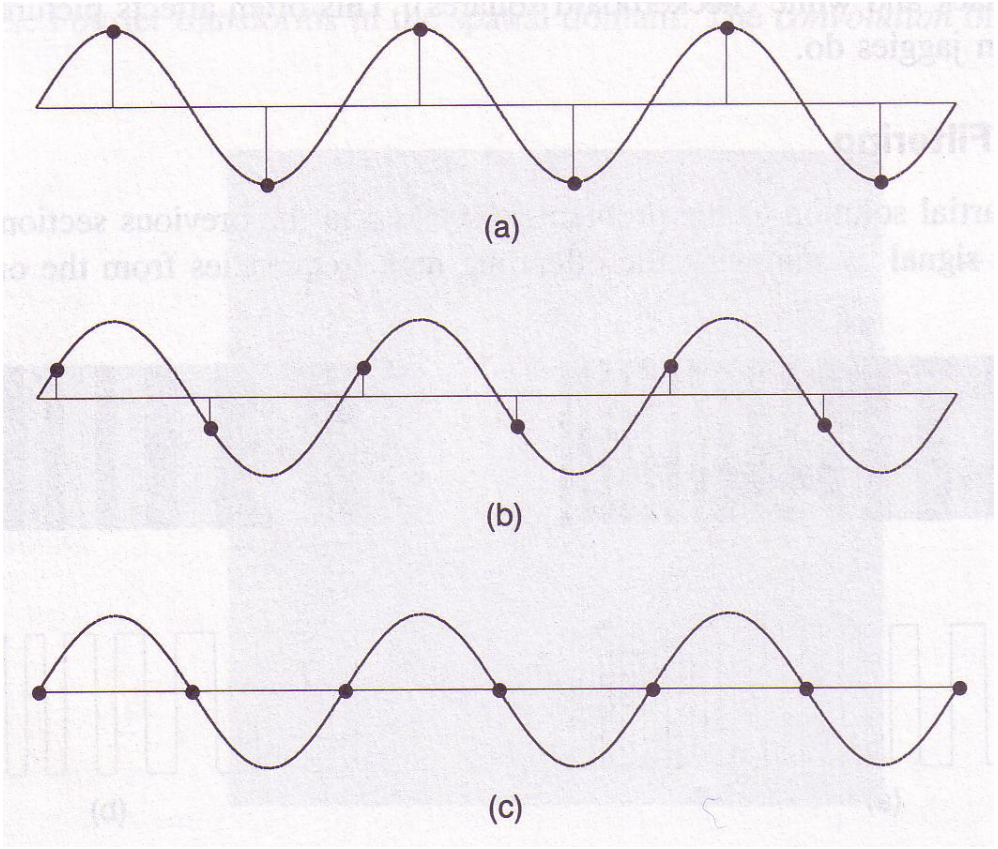


Why does Aliasing happen?

- Because sampling is not adequate



Nyquist Sampling at Work



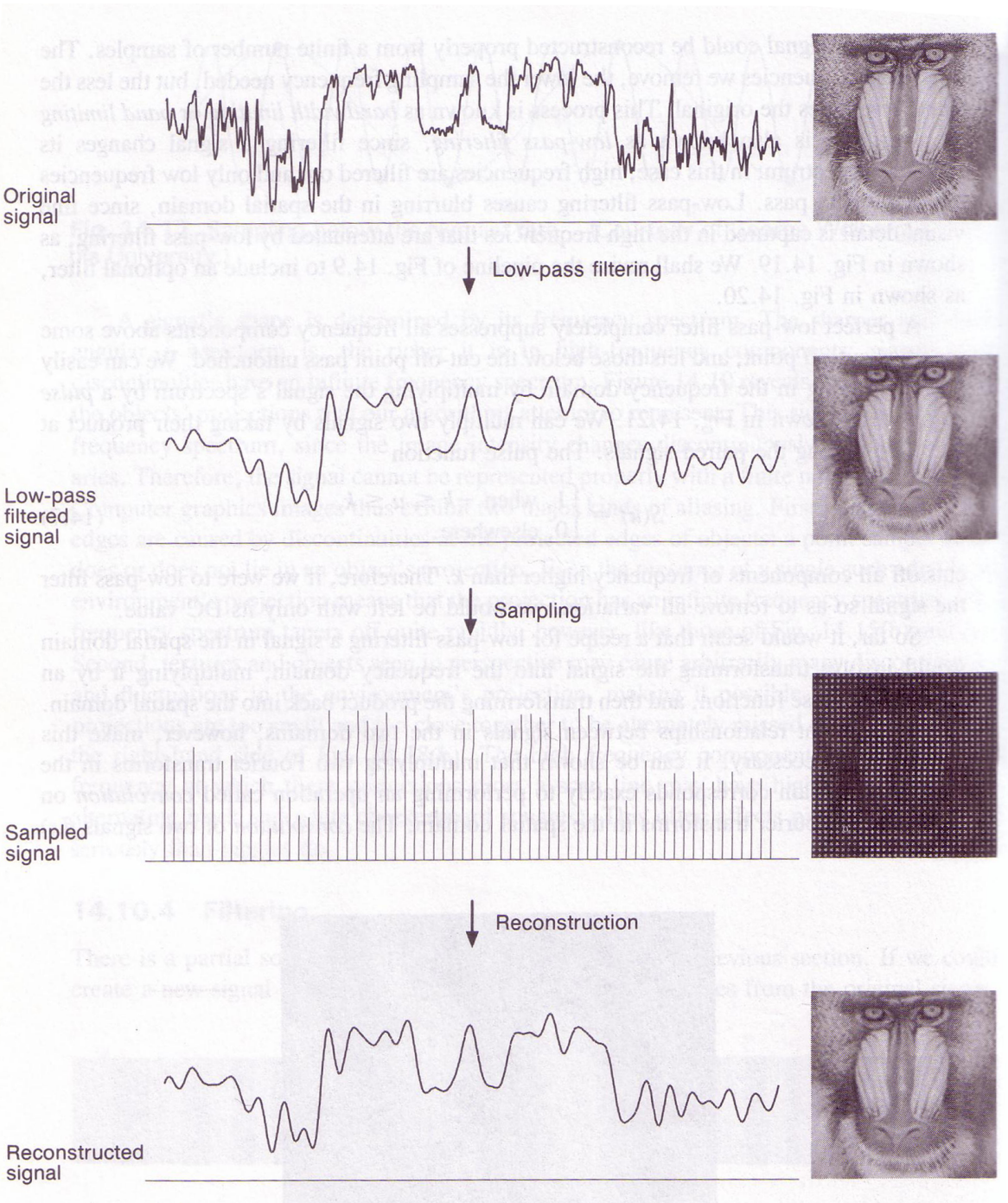
- (a) at peaks
- (b) other place
- (c) at zero crossings

Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals “traveling in disguise” as other frequencies

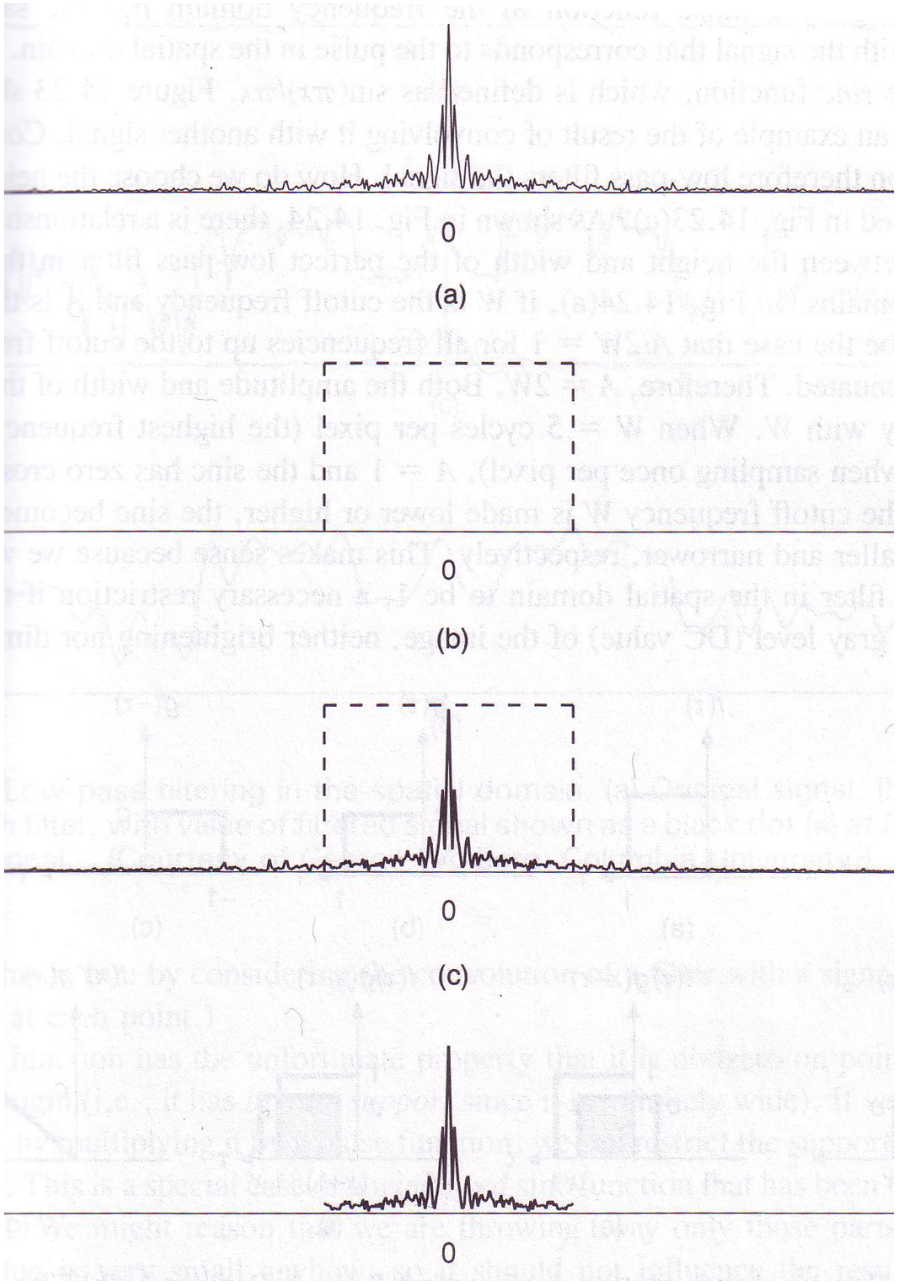
What sampling frequency is good?

- Frequency domain has infinite spectrum
- If you pick some random cutoff
  - Aliasing
- Instead, use pre-filtering to remove high frequencies
  - Then you can ensure that your sampling frequency is good enough


Original signal
Low-pass filtering
Low-pass filtered signal
Sampling
Sampled signal
Reconstruction
Reconstructed signal

Process

- Convert to frequency domain
- Multiply by low pass filter
  - Eliminate high frequencies
- Sample
- Reconstruct



Filtering

- Lost some detail in this process
- But ensure that you eliminate objectionable high frequency artifacts

Filtering

- Lost some detail in this process
- But ensure that you eliminate objectionable high frequency artifacts
- Have to transform to frequency domain?
  - Expensive
  - Not necessary because of relationship between spatial and frequency domain

Convolution

Multiplication in frequency domain

==

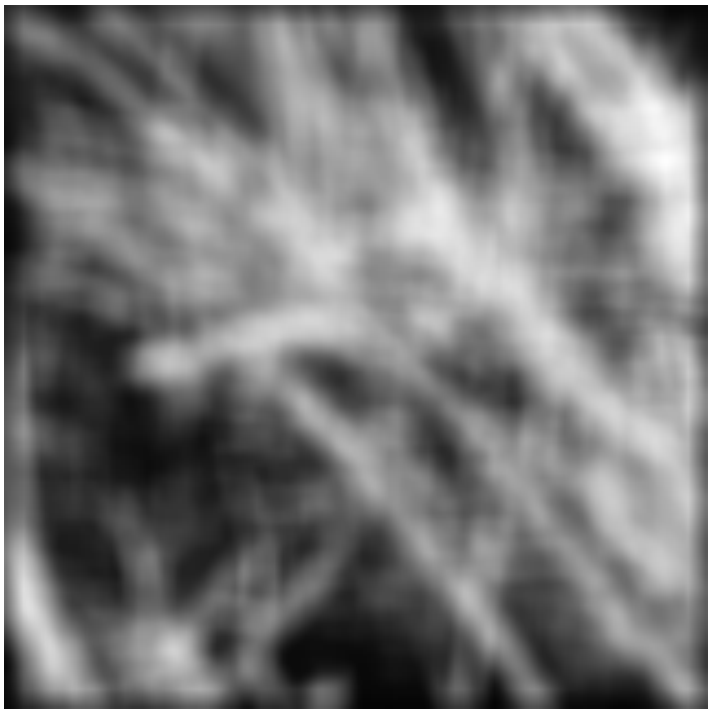
Convolution in spatial domain

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau)\, g(x-\tau)\, d\tau$$

Types of Filters

Spatial	⟺	Frequency
• Sine		• Impulse
• Gaussian		• Gaussian
• Box		• Sinc
• Sinc		• Box

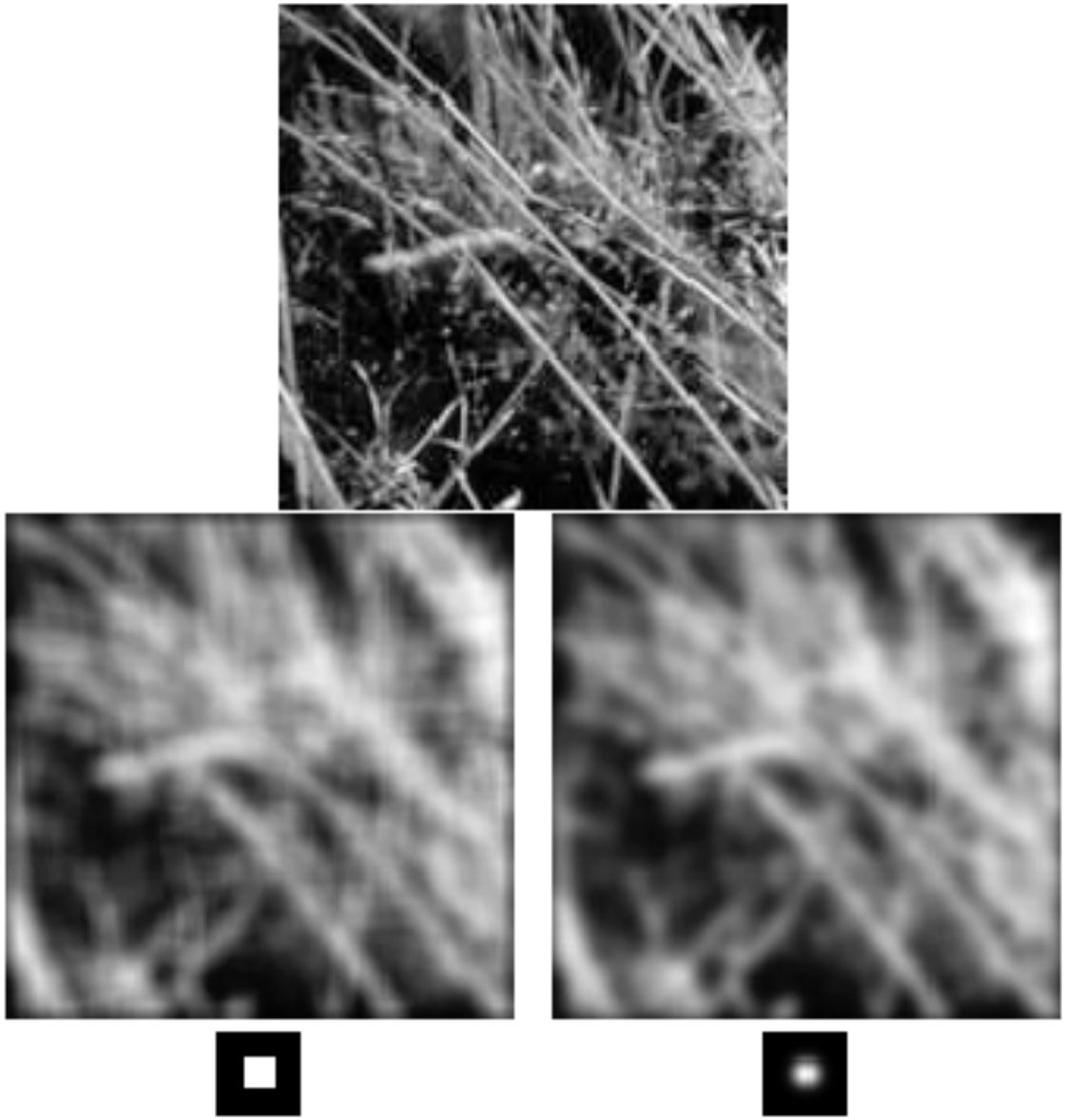
Smoothing with box filter



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Source: D. Forsyth

Mean vs. Gaussian filtering



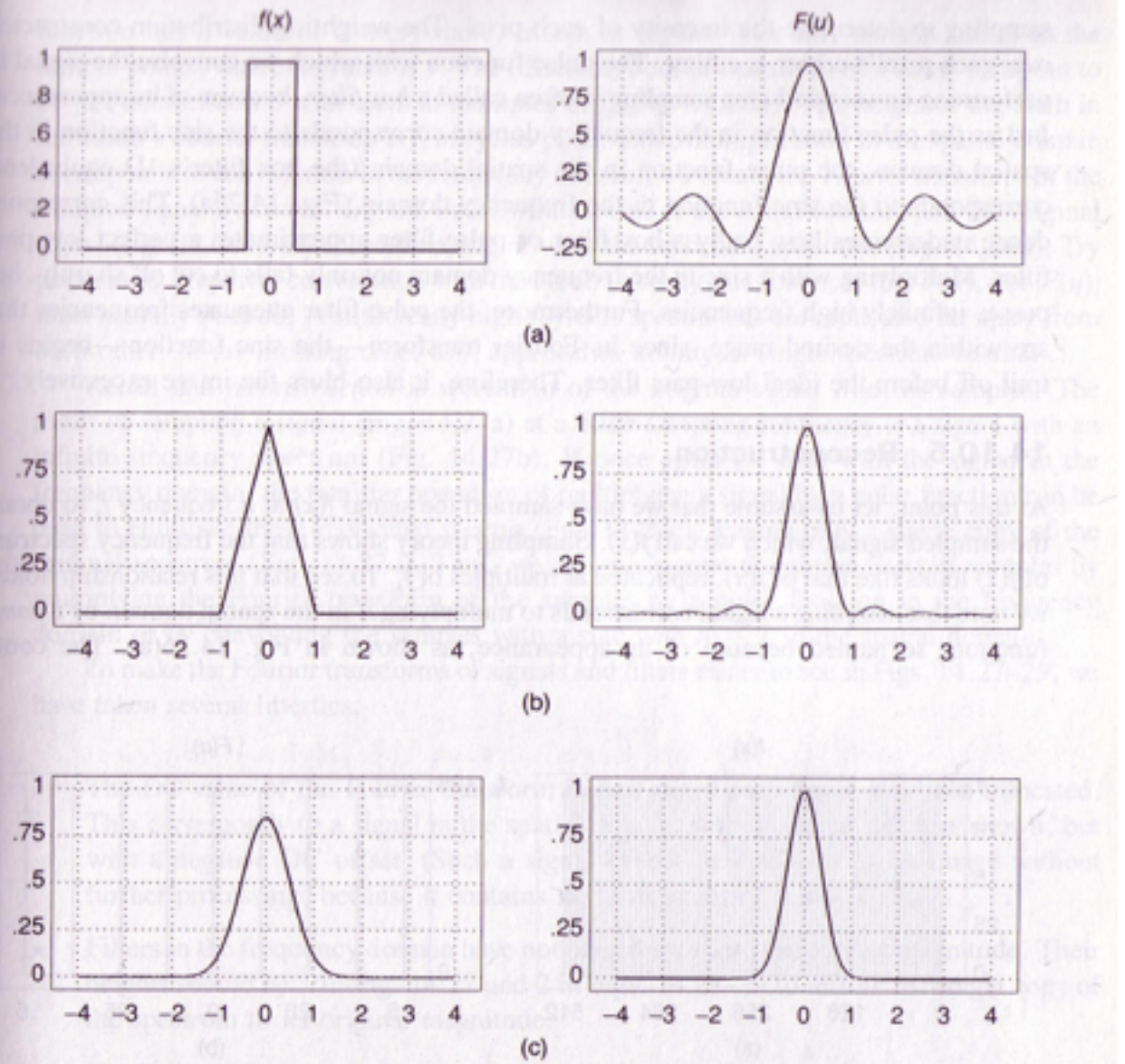
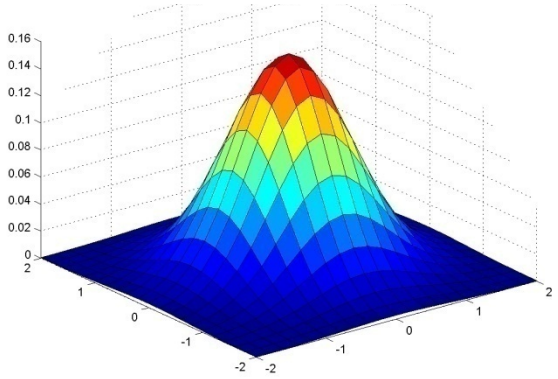
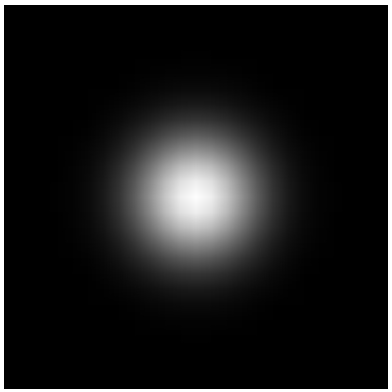


Fig. 14.25 Filters in spatial and frequency domains. (a) Pulse—sinc. (b) Triangle—sinc^2. (c) Gaussian—Gaussian. (Courtesy of George Wolberg, Columbia University.)

Gaussian Kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

Source: C. Rasmussen

Gaussian Filter

- Gaussian is preferred because it is same in frequency and spatial domain

Pre-Filtering

- Ideal is box/pulse in frequency space
- Sinc in spatial domain
- But infinite spread

Truncated sinc

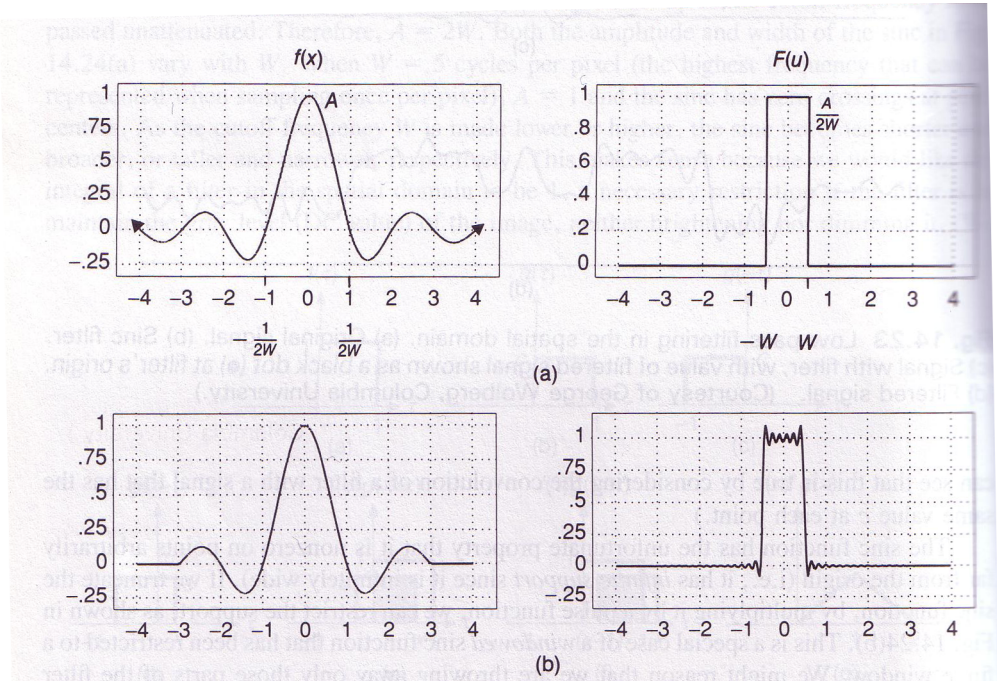


Fig. 14.24 (a) Sinc in spatial domain corresponds to pulse in frequency domain. (b) Truncated sinc in spatial domain corresponds to ringing pulse in frequency domain. (Courtesy of George Wolberg, Columbia University.)

Putting it together

- Get input image
- Pre-filter with appropriate filter
  - sinc, gaussian
  - Eliminate high frequencies
- Sample
- Reconstruct

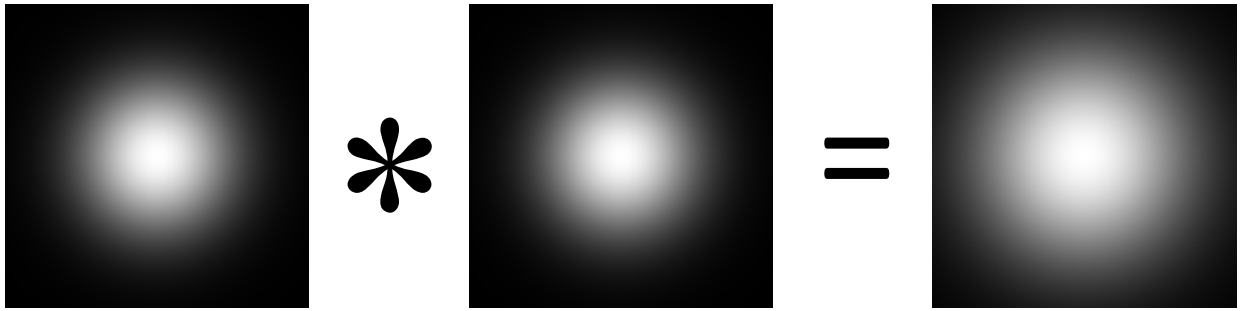
Convolution is special

- Convolution in image space
  - Multiplication in Fourier space

- Box filter -> sinc in Fourier space
- Gaussian filter -> Gaussian in Fourier space

Gaussian filter

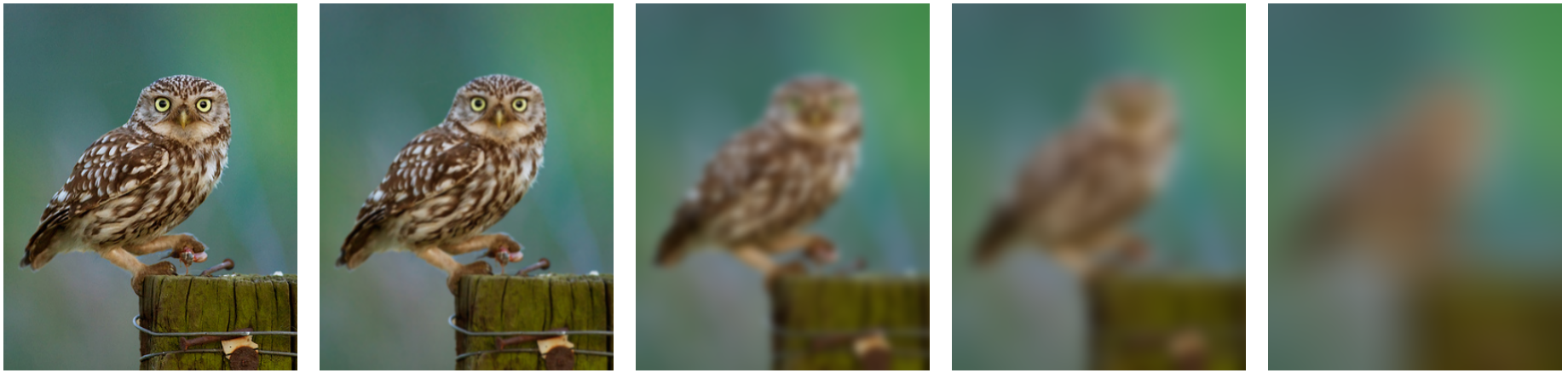
- Removes “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian



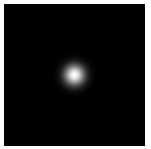
  - Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

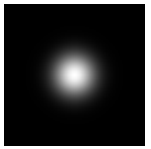
Gaussian filters



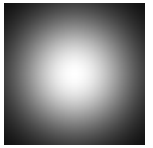
σ = 1 pixel



σ = 5 pixels



σ = 10 pixels



σ = 30 pixels