

Image Processing 1



CS180: Intro to Comp. Vision, and Comp. Photo
Alexei Efros, Ren Ng, UC Berkeley, Fall 2026

What is an image?

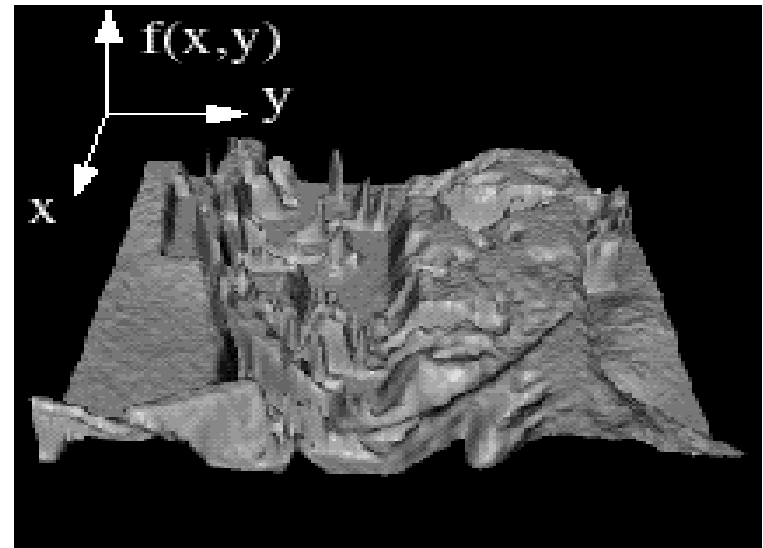
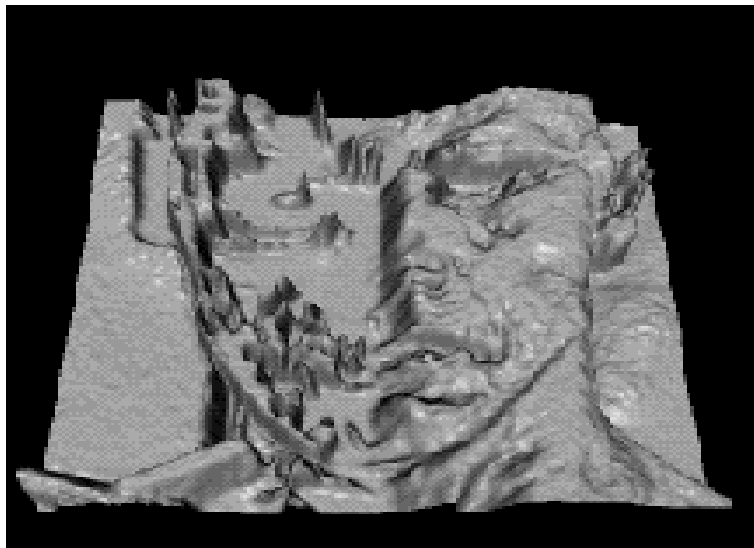
We can think of an **image** as a function, f , from $\mathbb{R}^2$ to $\mathbb{R}$:

- $f(x, y)$ gives the **intensity** at position (x, y)
- Realistically, we expect the image only to be defined over a rectangle, with a finite range:
 - $f: [a, b] \times [c, d] \rightarrow [0, 1]$

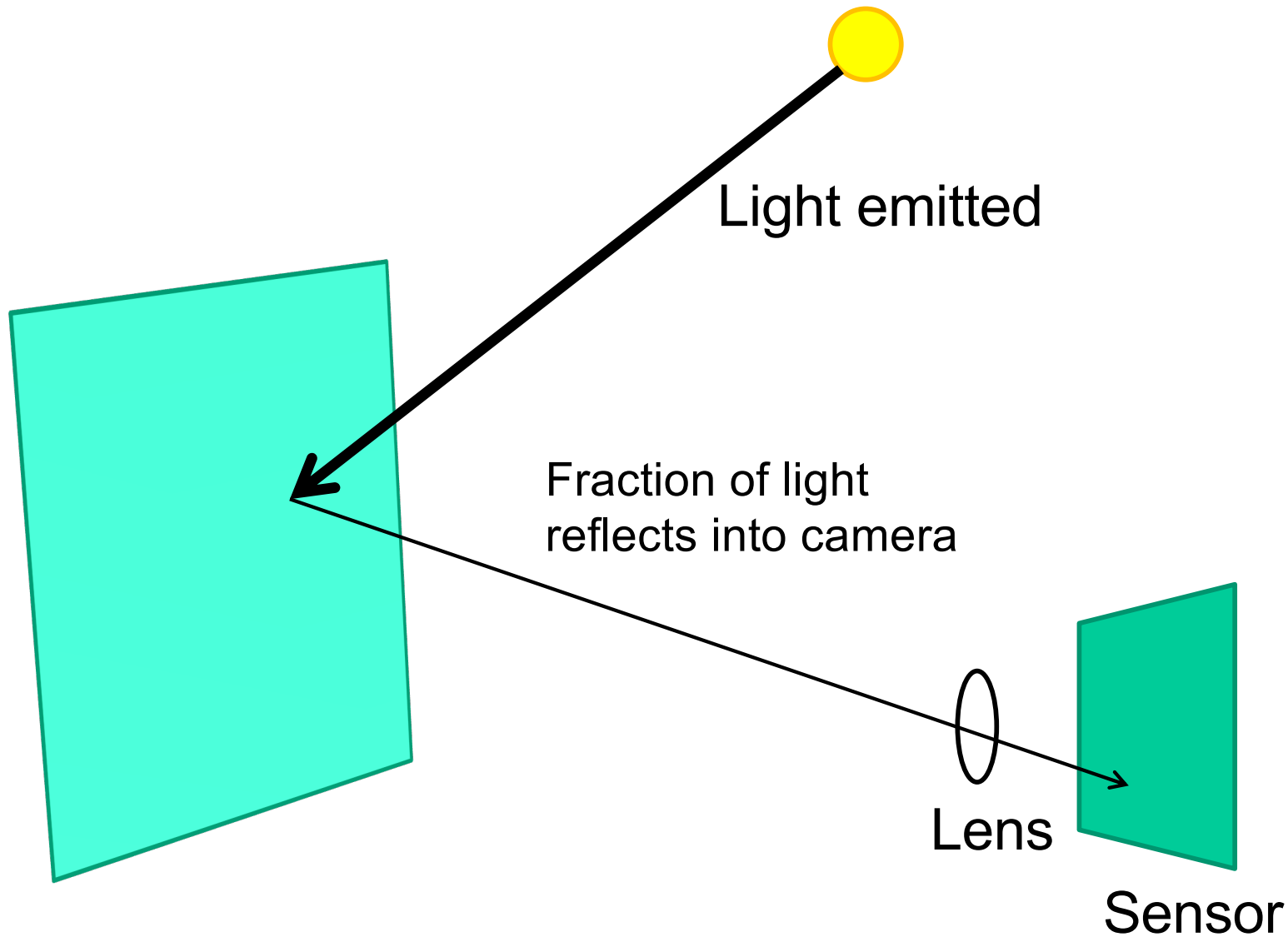
A color image is just three functions pasted together.
We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

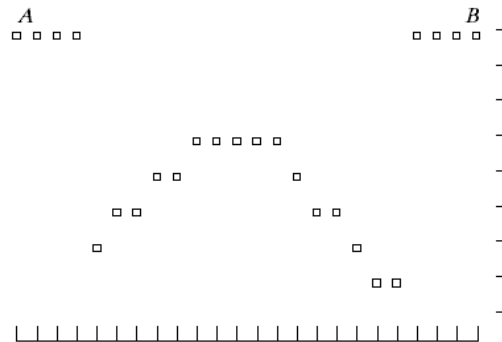
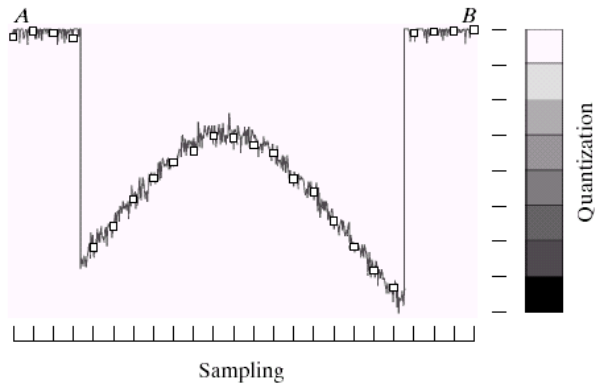
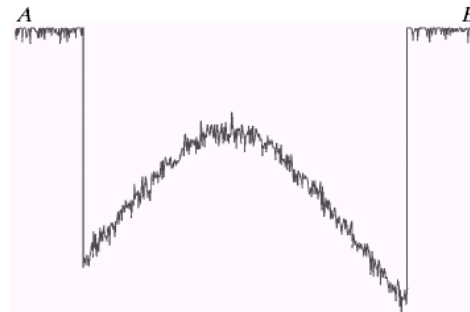
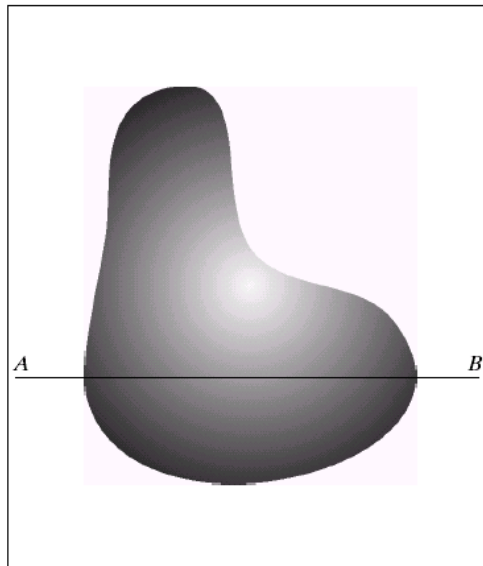
Images as functions



How does a pixel get its value?

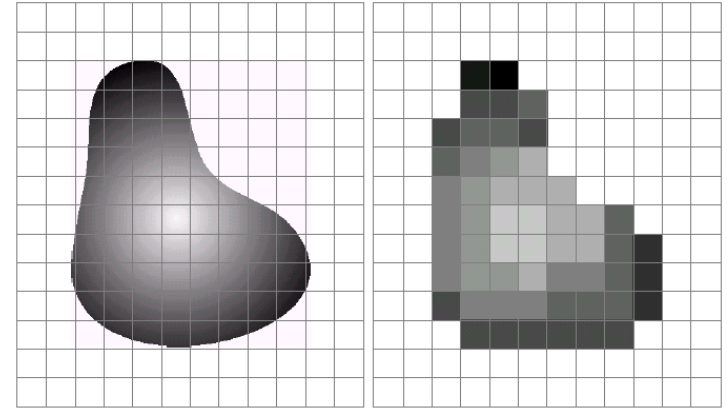


Sampling and Quantization



a b
c d

FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.



a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Image Formation

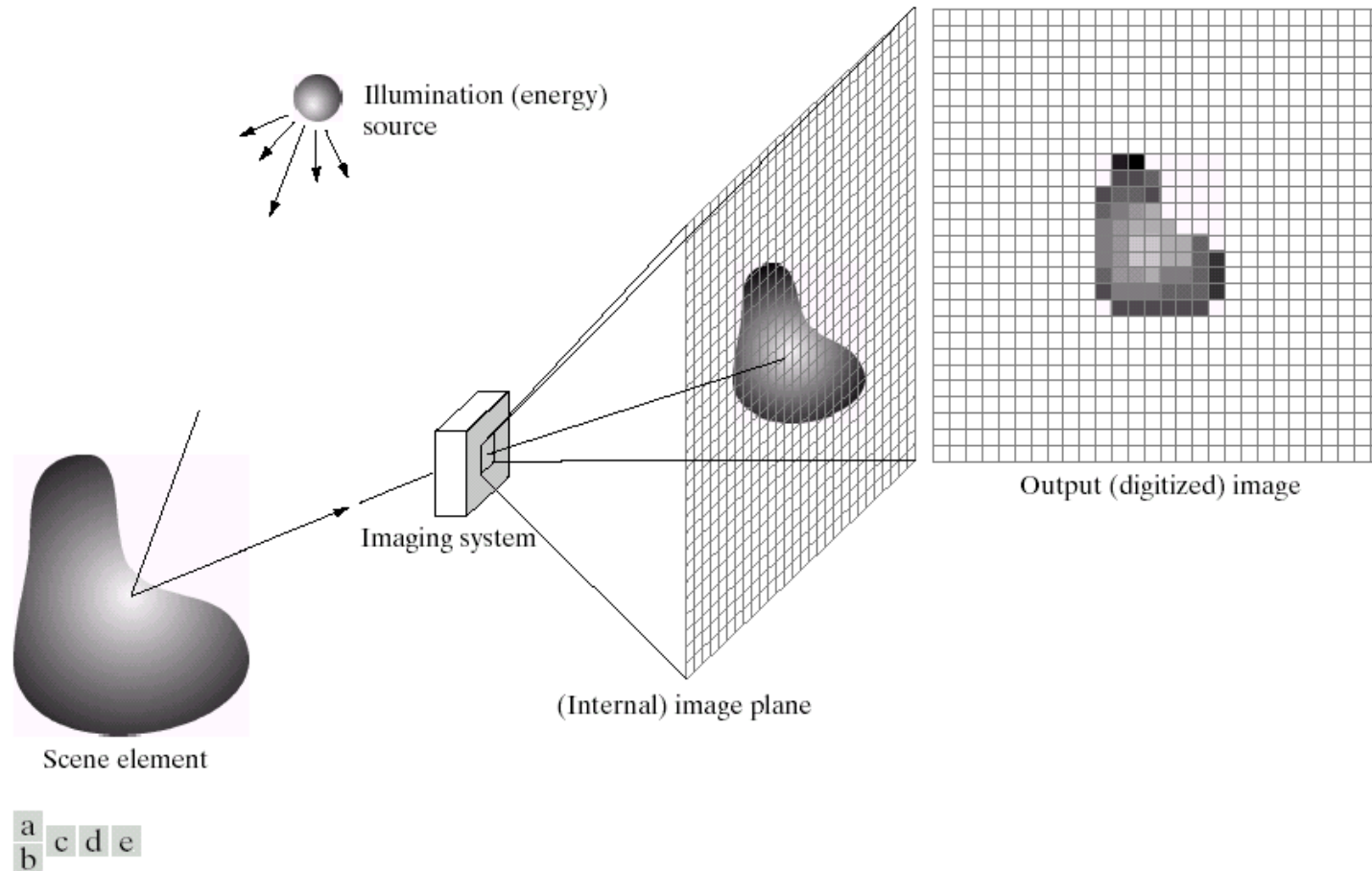


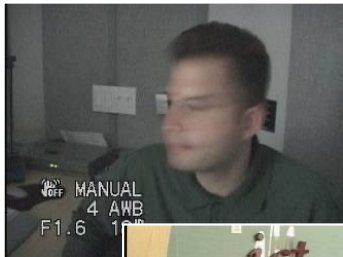
FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

$$f(x,y) = \text{reflectance}(x,y) * \text{illumination}(x,y)$$

Reflectance in $[0,1]$, illumination in $[0,\text{inf}]$

Problem: Dynamic Range

The real world is
High dynamic range



1



1500



25,000

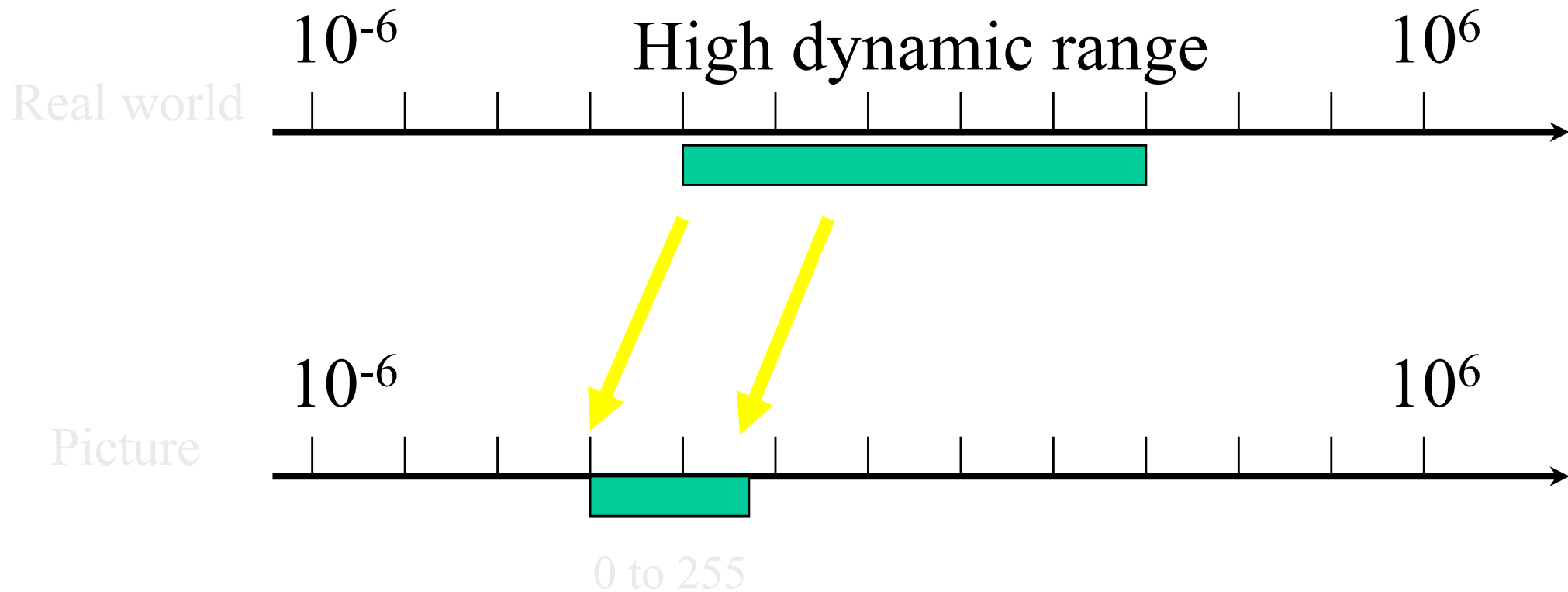


400,000



2,000,000,000

Long Exposure



Short Exposure

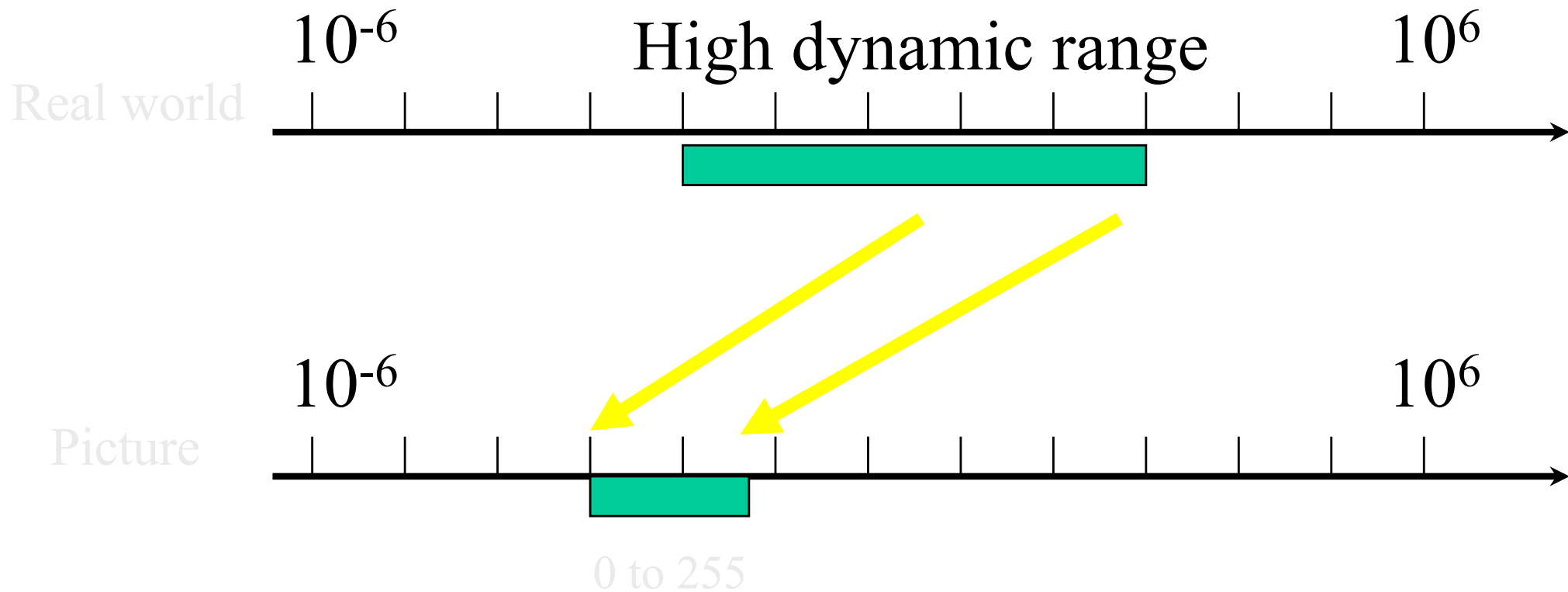
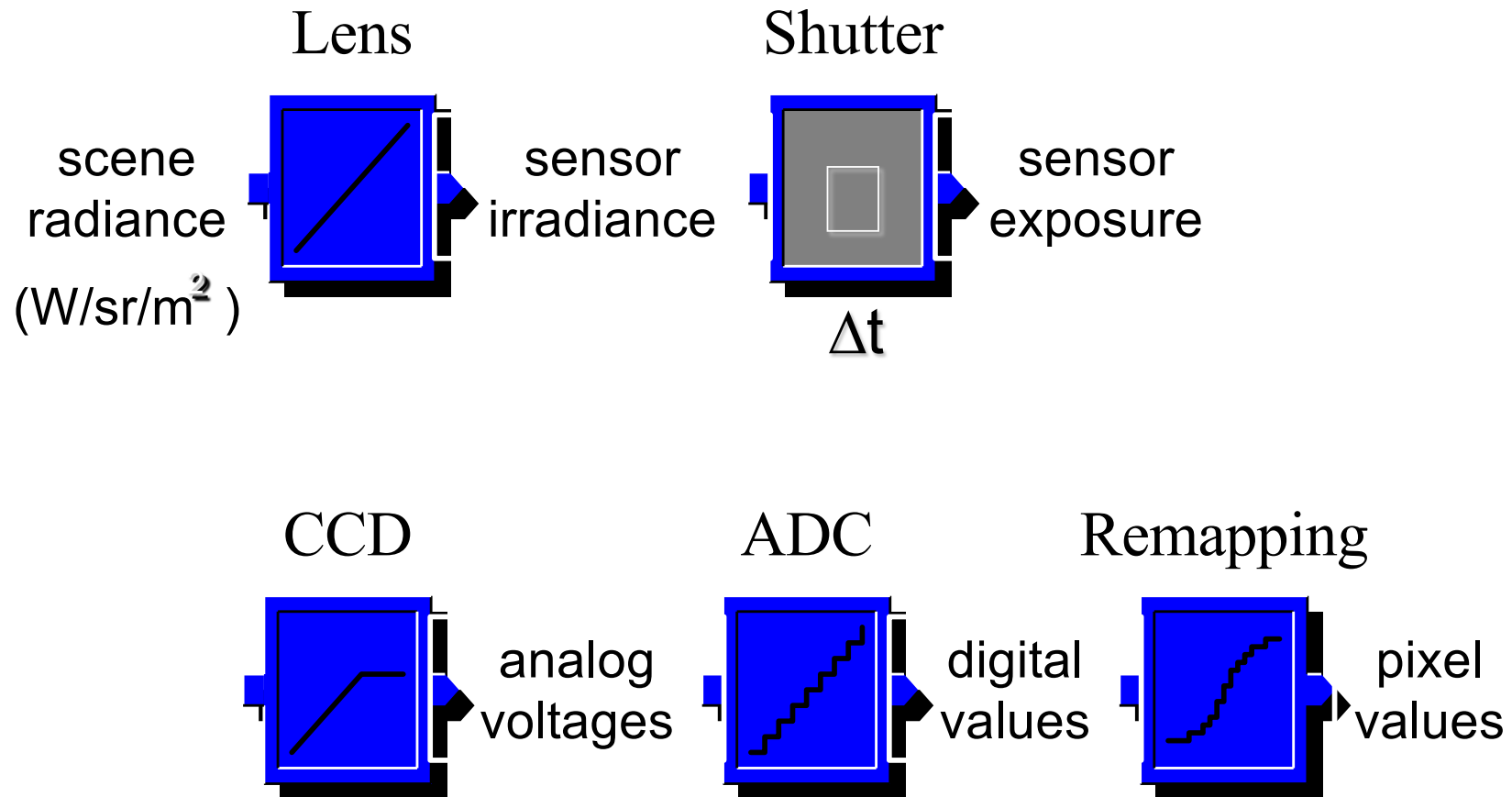


Image Acquisition Pipeline

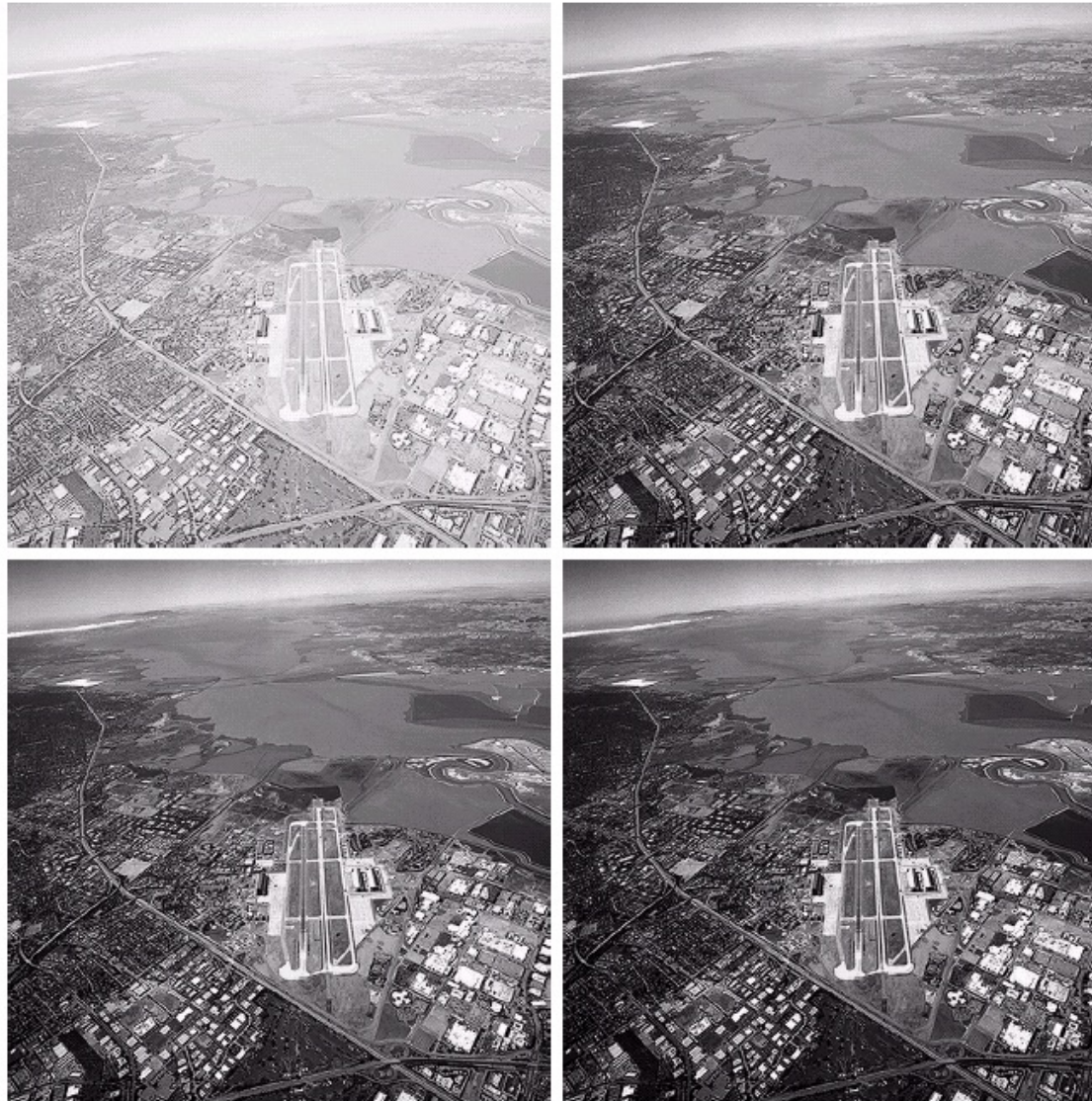


Simple Point Processing: Enhancement

a b
c d

FIGURE 3.9

(a) Aerial image.
(b)–(d) Results of
applying the
transformation in
Eq. (3.2-3) with
 $c = 1$ and
 $\gamma = 3.0, 4.0,$ and
 5.0 , respectively.
(Original image
for this example
courtesy of
NASA.)



Power-law transformations

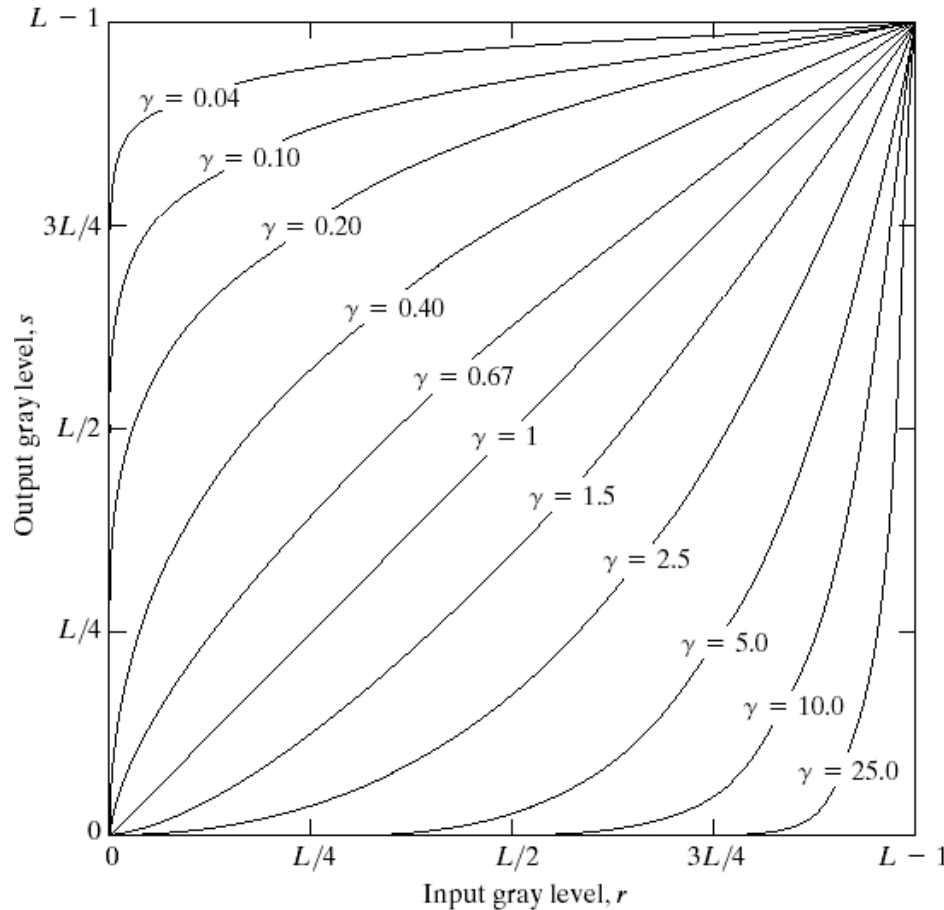
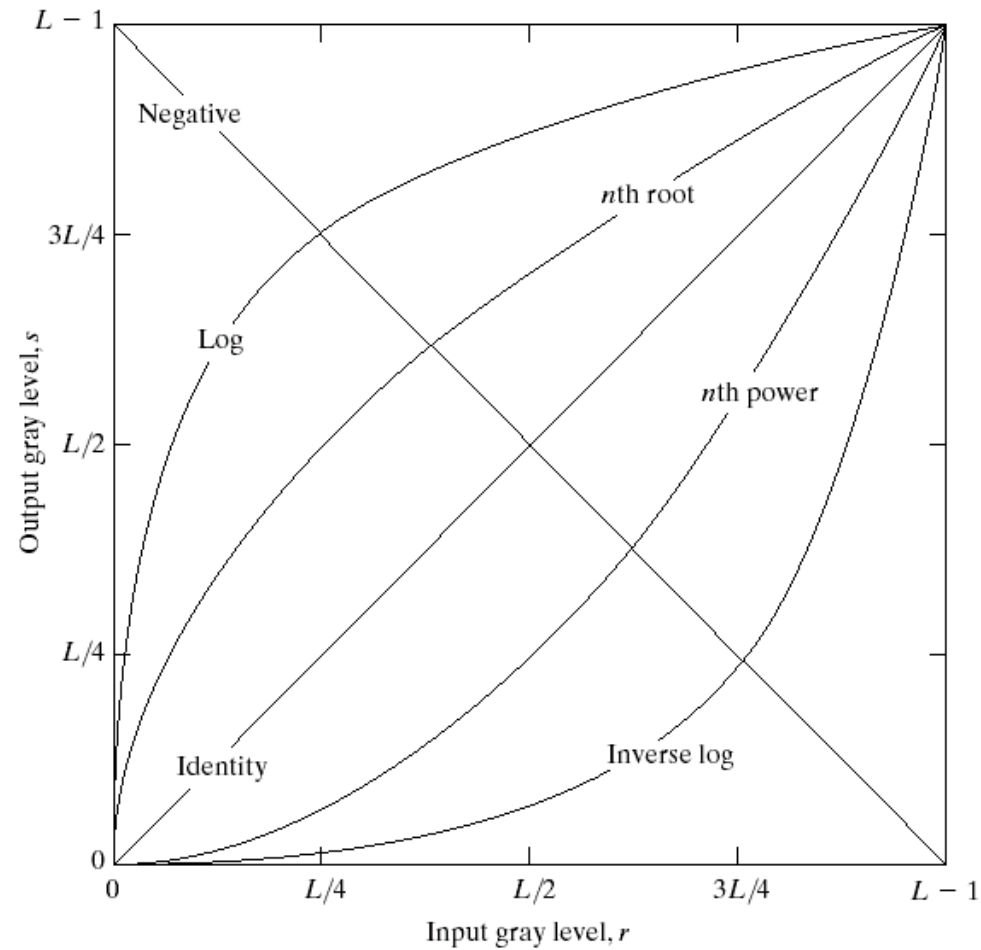


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

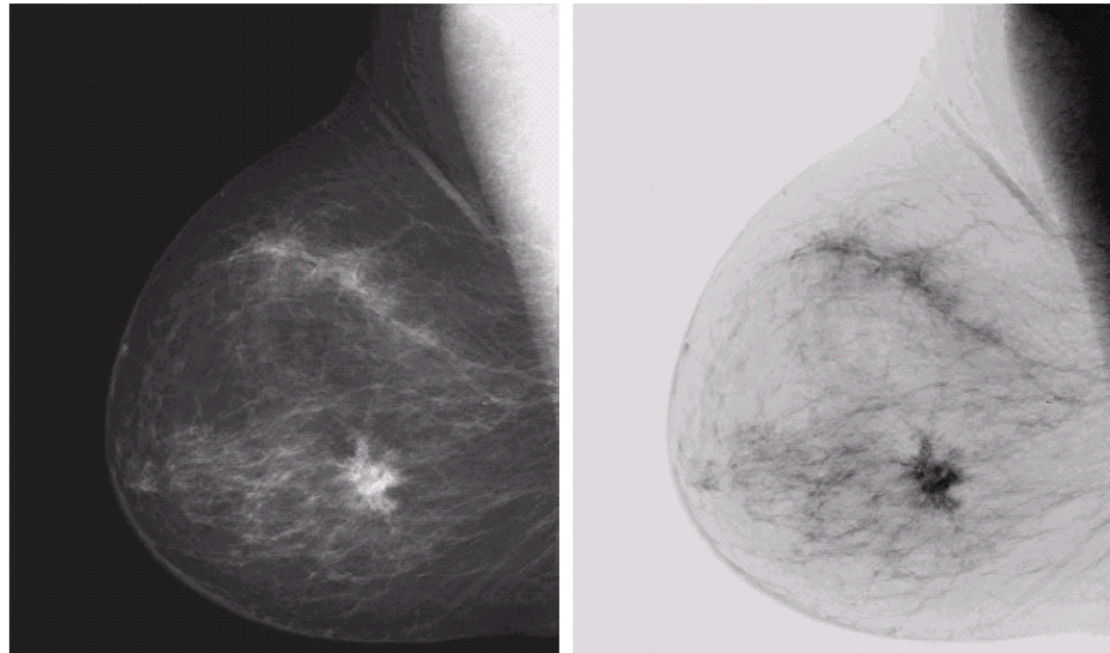
$$s = cr^\gamma$$

Basic Point Processing

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



Negative



a b

FIGURE 3.4

(a) Original digital mammogram.

(b) Negative image obtained using the negative transformation in Eq. (3.2-1).

(Courtesy of G.E. Medical Systems.)

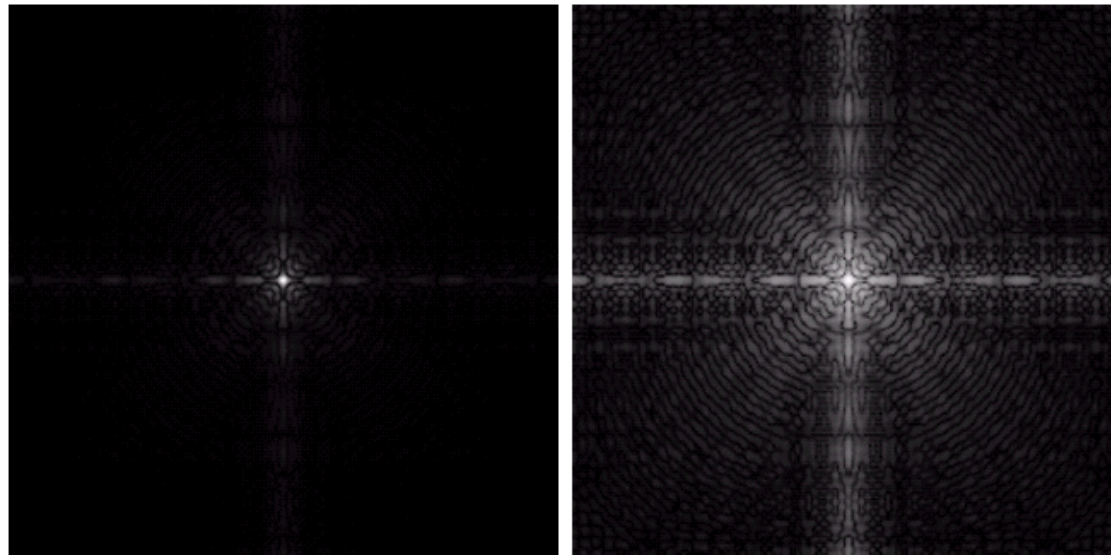
Log

a b

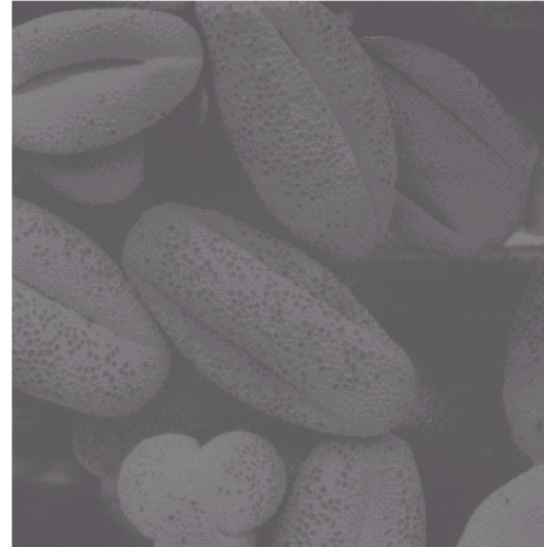
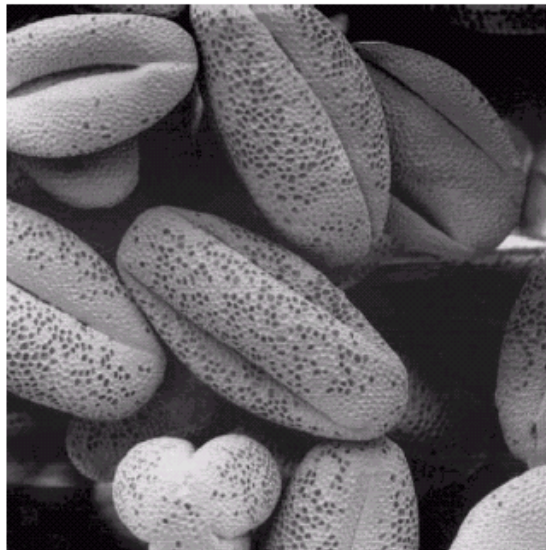
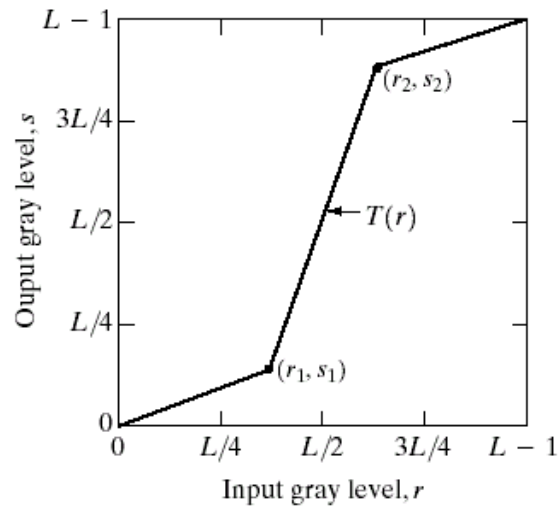
FIGURE 3.5

(a) Fourier spectrum.

(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



Contrast Stretching



a b
c d

FIGURE 3.10

Contrast stretching.

(a) Form of transformation function.

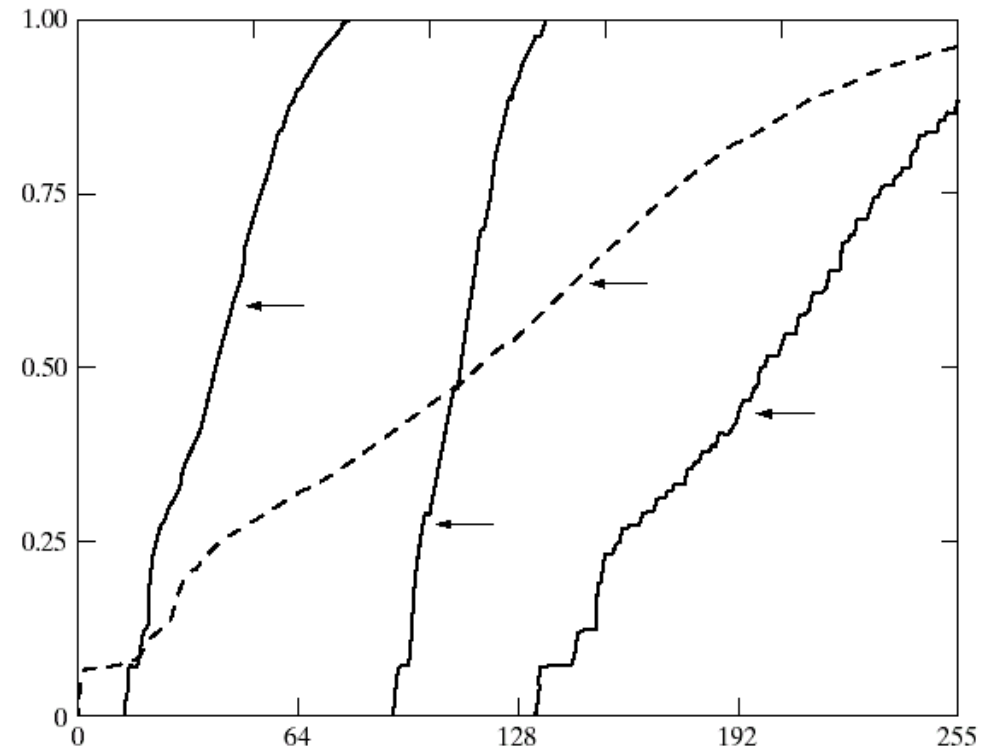
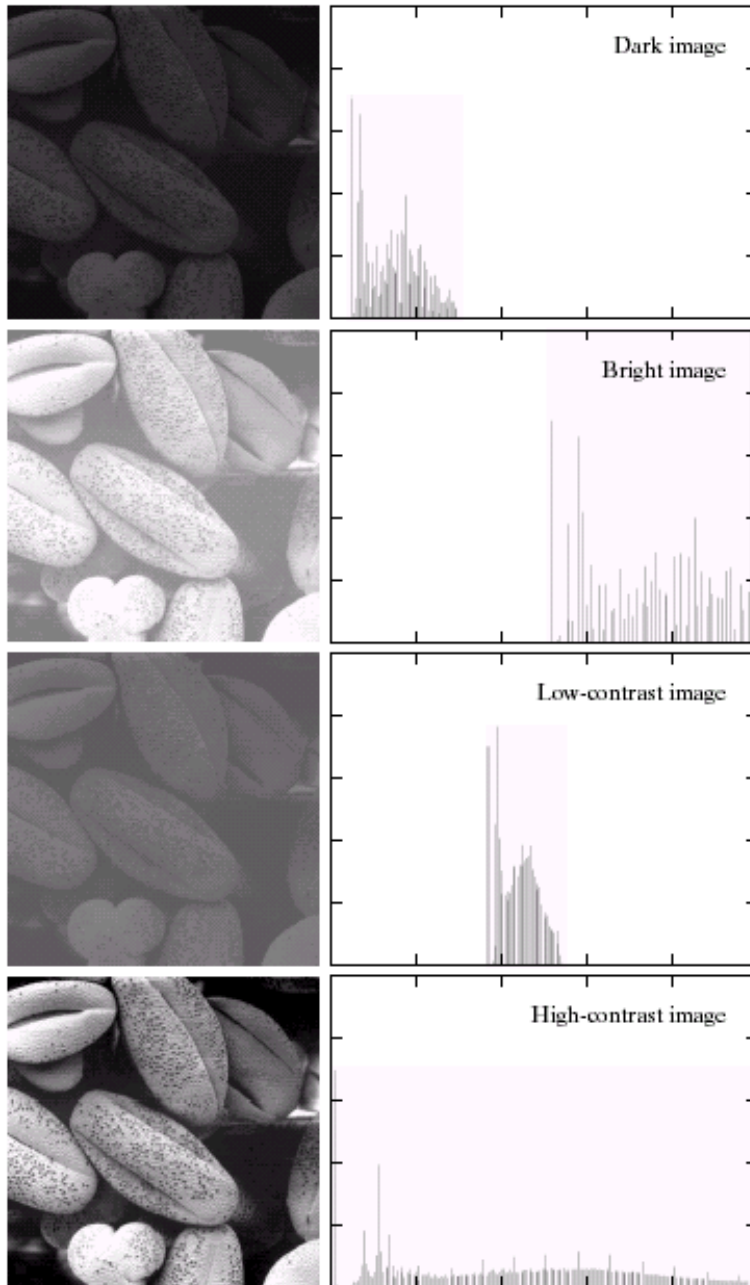
(b) A low-contrast image.

(c) Result of contrast stretching.

(d) Result of thresholding.

(Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Image Histograms



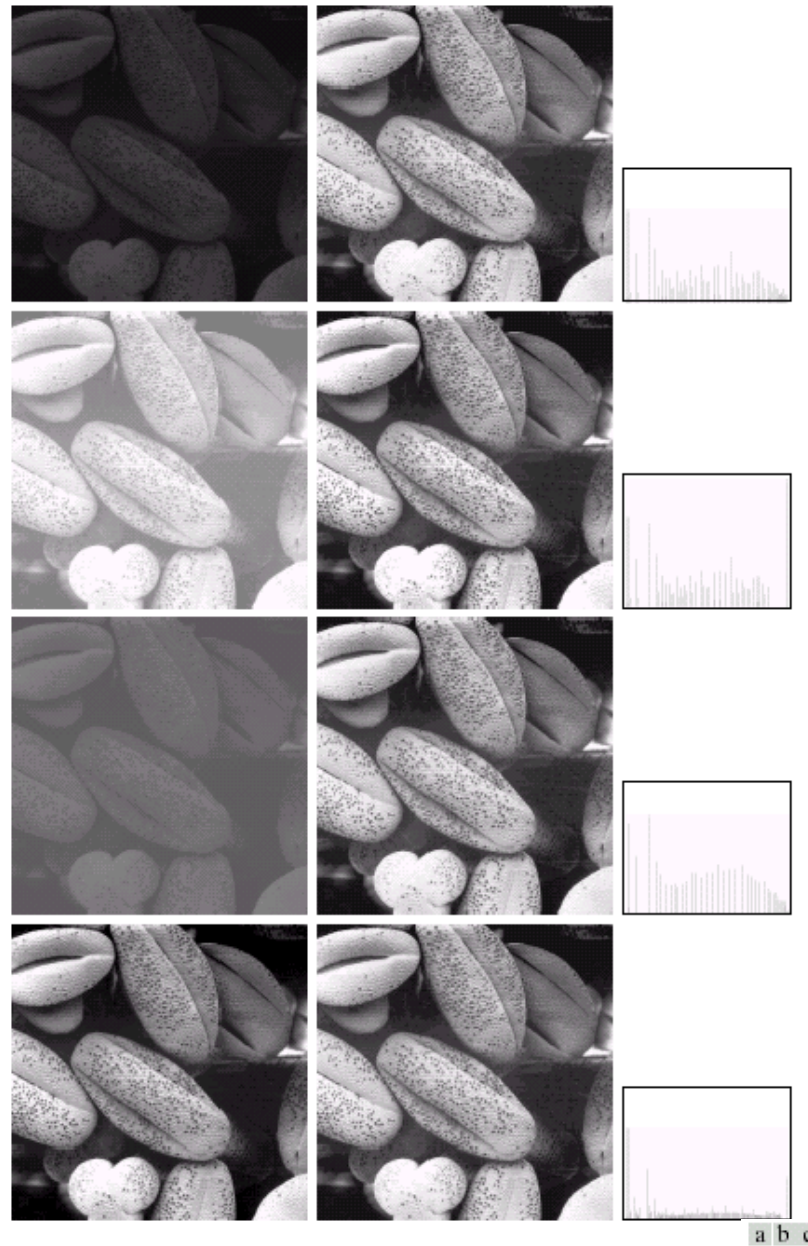
Cumulative Histograms

$$s = T(r)$$

a b

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Histogram Equalization



a b c

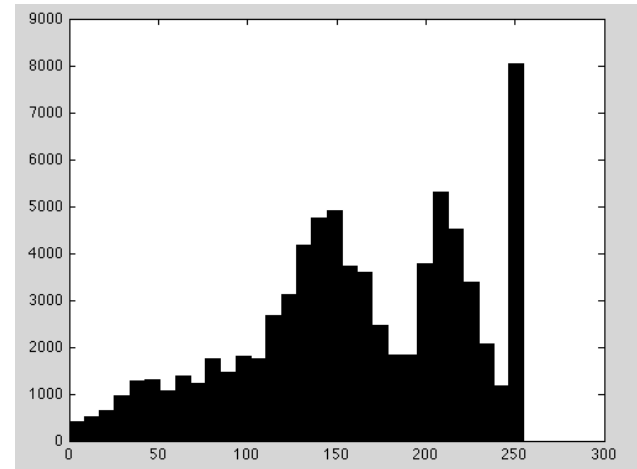
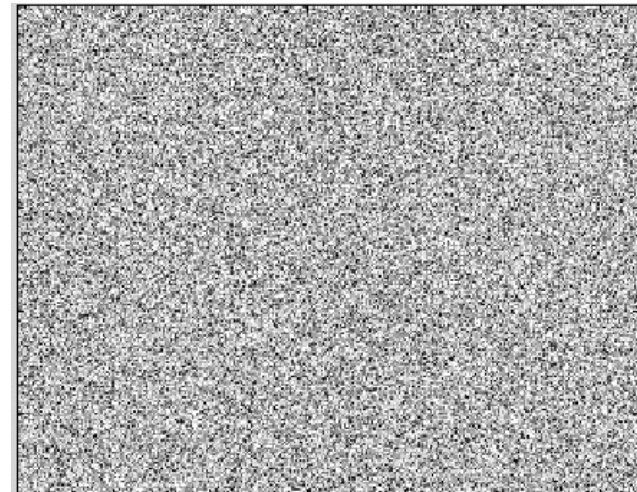
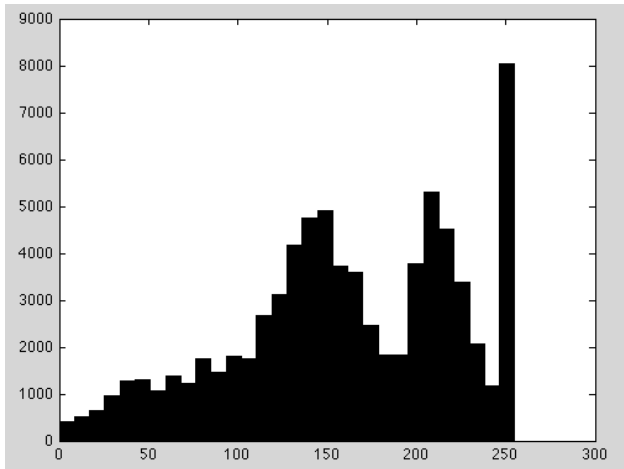
FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

Color Transfer [Reinhard, et al, 2001]

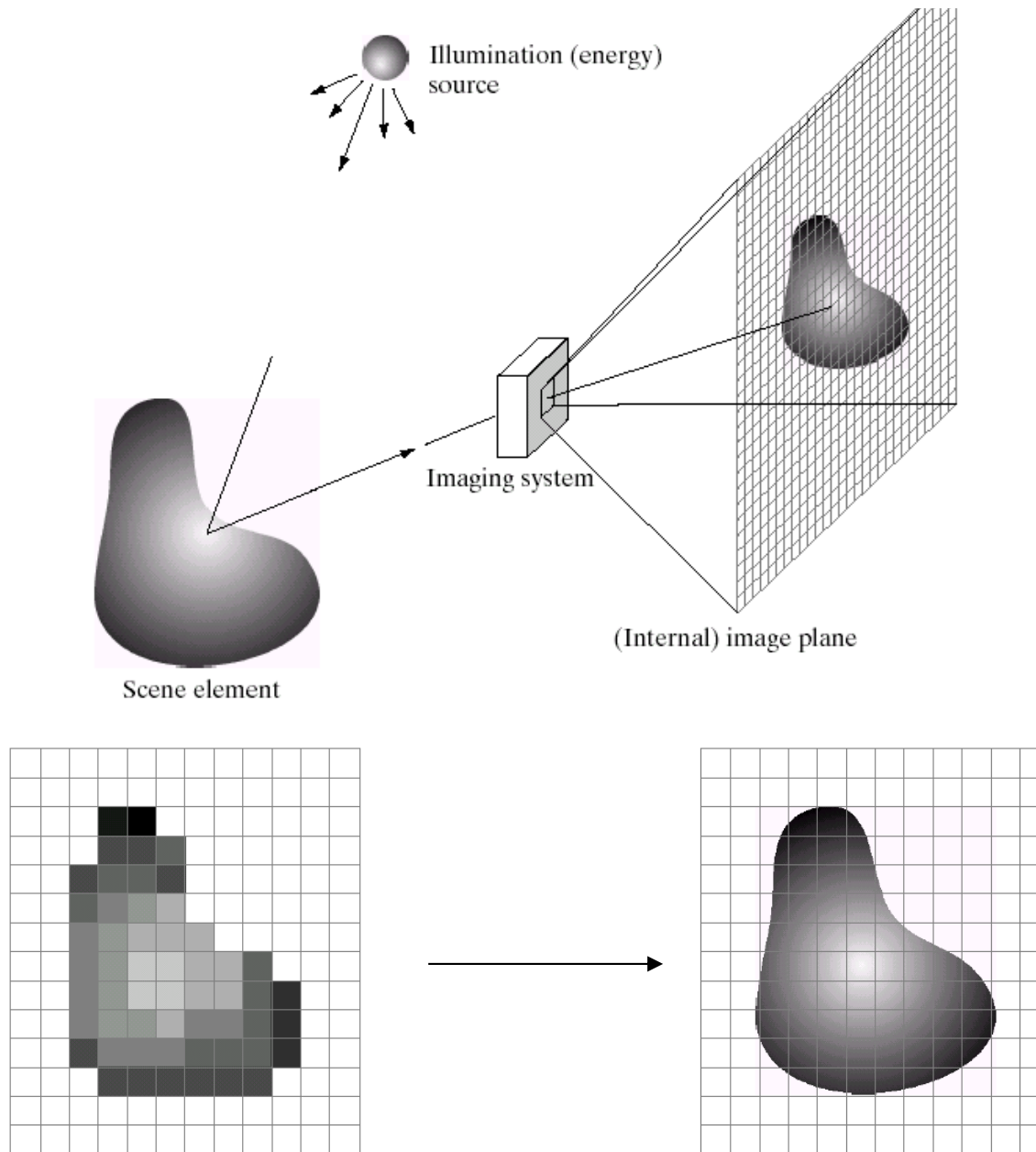


Erik Reinhard, Michael Ashikhmin, Bruce Gooch, Peter Shirley, [Color Transfer between Images](#). *IEEE Computer Graphics and Applications*, 21(5), pp. 34–41. September 2001.

Limitations of Point Processing...

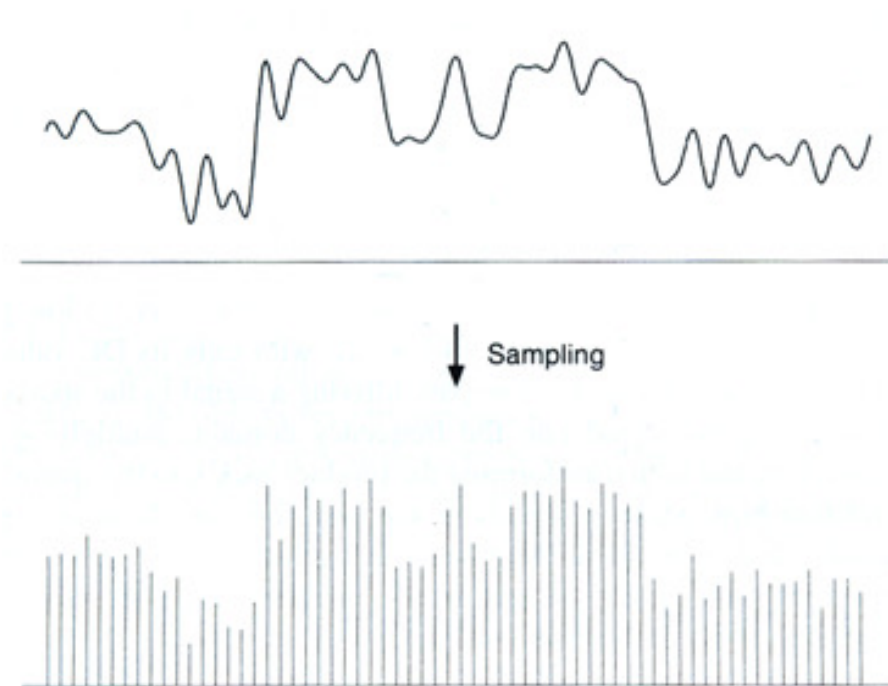


Sampling and Reconstruction



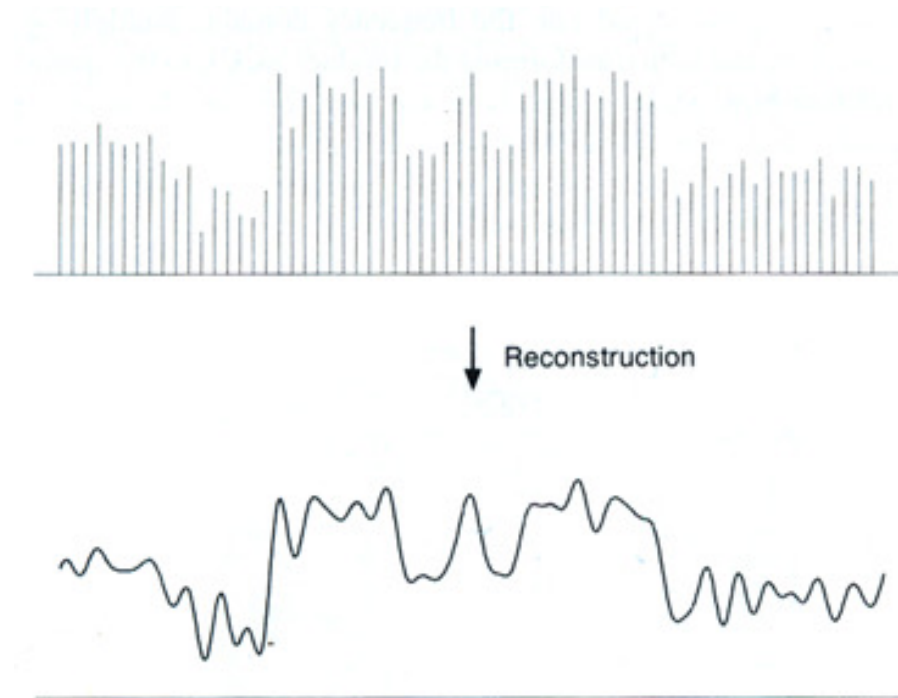
Sampled representations

- How to store and compute with continuous functions?
- Common scheme for representation: samples
 - write down the function's values at many points

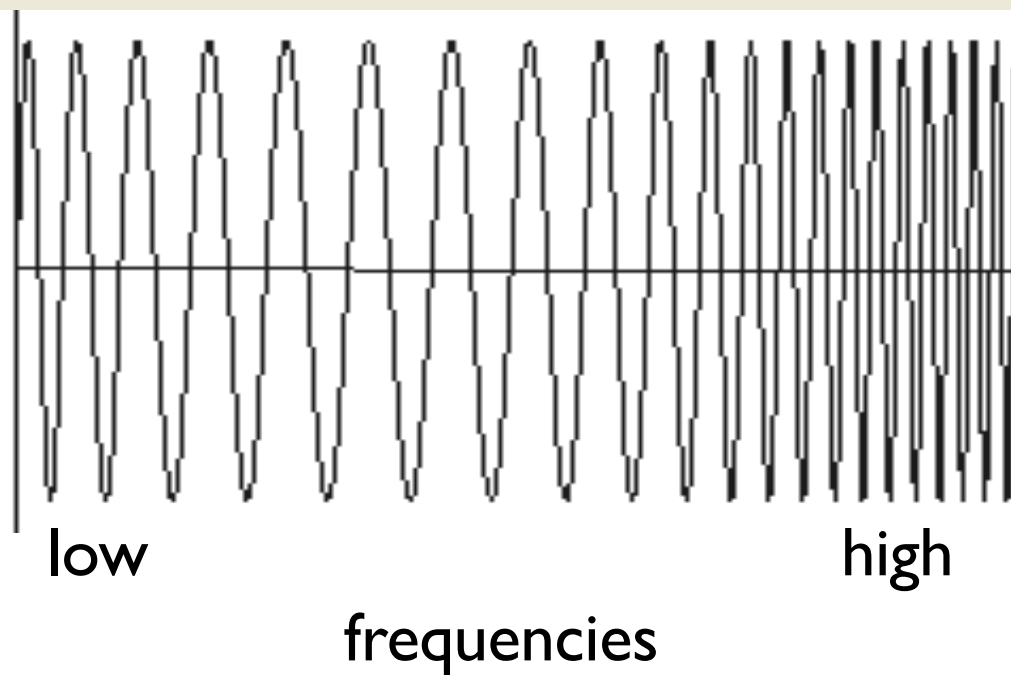
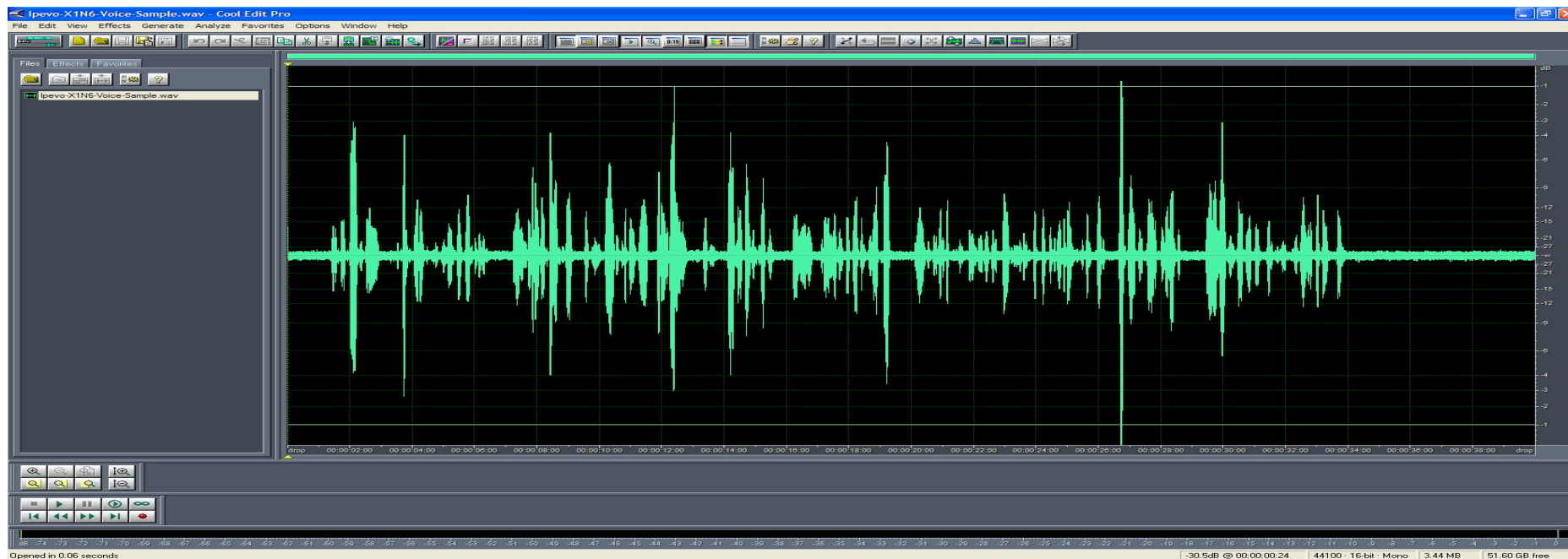


Reconstruction

- Making samples back into a continuous function
 - for output (need realizable method)
 - for analysis or processing (need mathematical method)
 - amounts to “guessing” what the function did in between

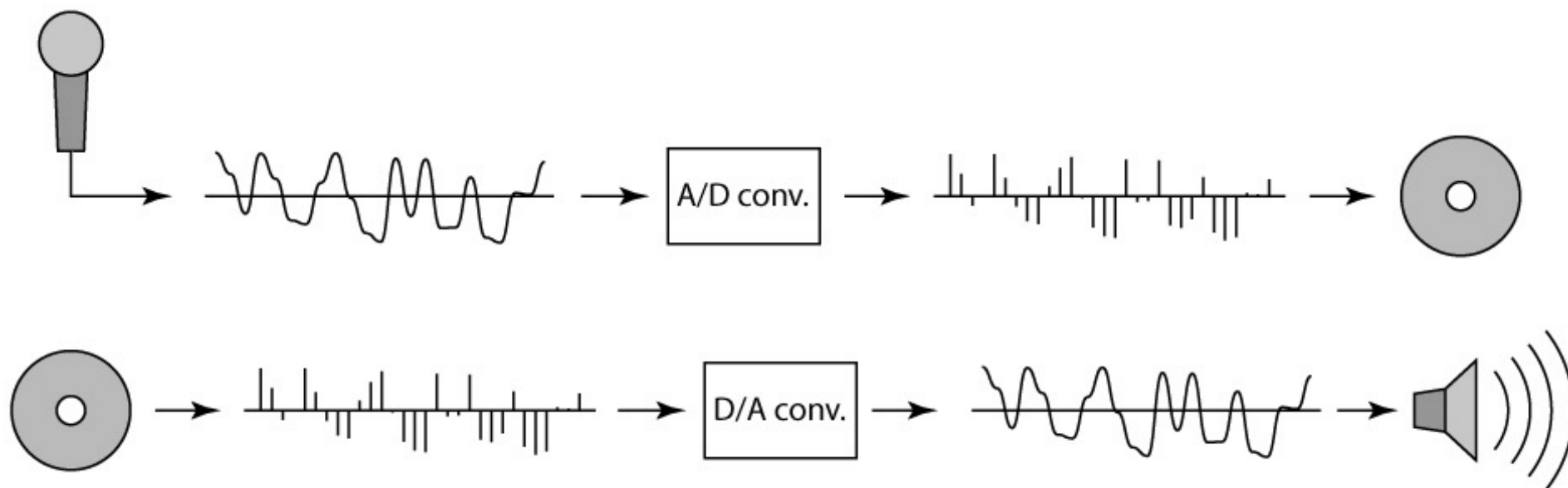


1D Example: Audio



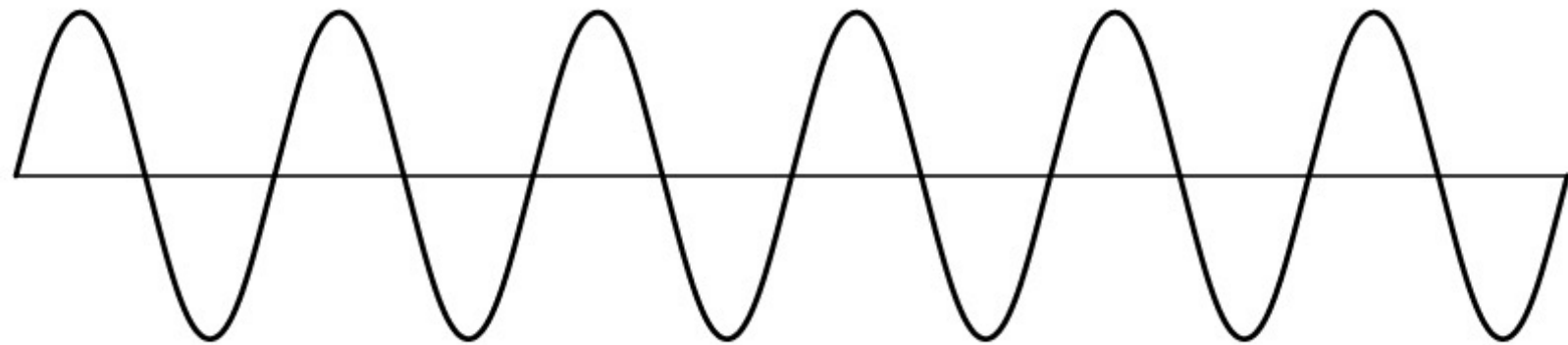
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
 - how can we be sure we are filling in the gaps correctly?



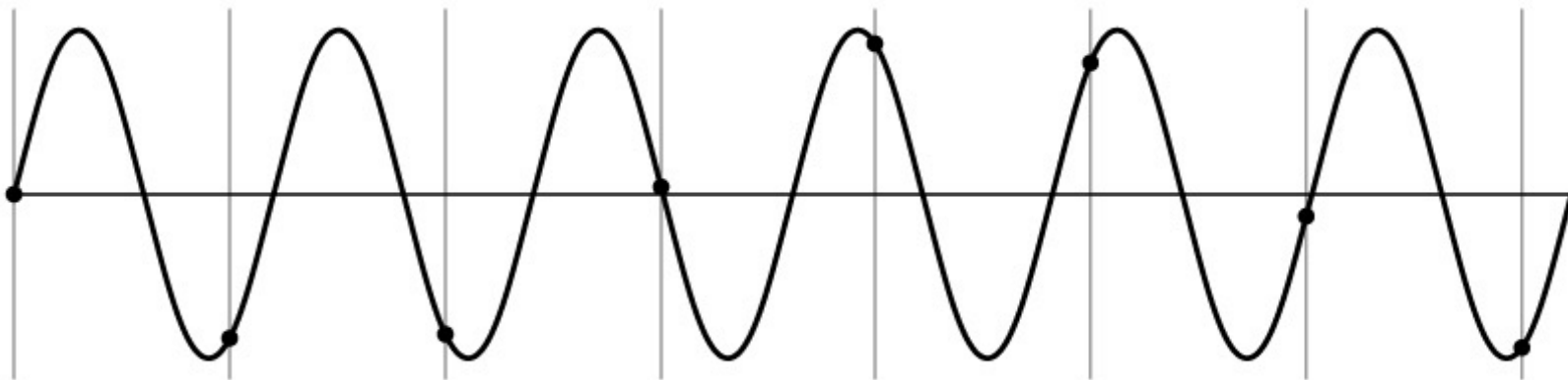
Sampling and Reconstruction

- Simple example: a sign wave



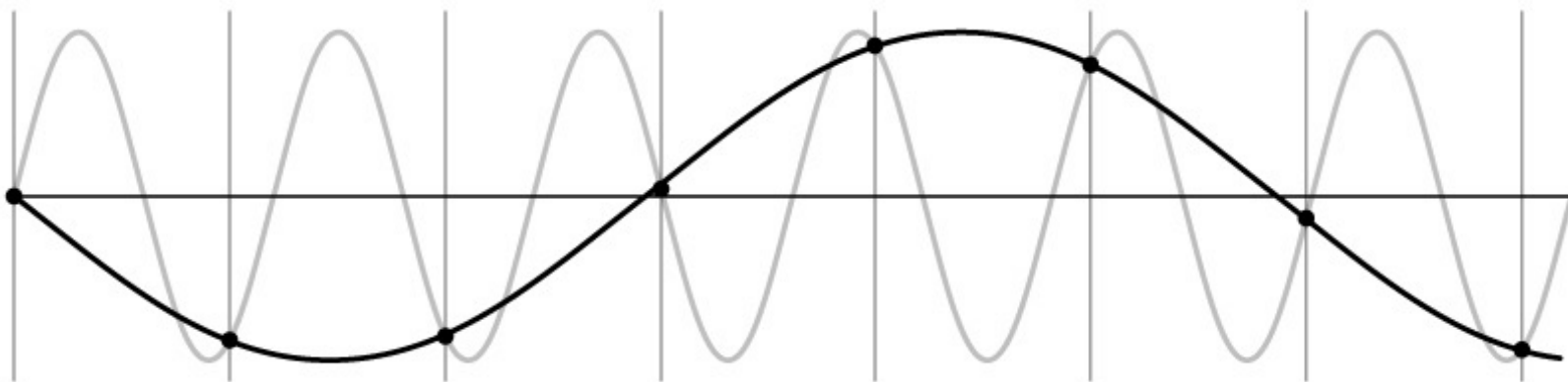
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
 - unsurprising result: information is lost



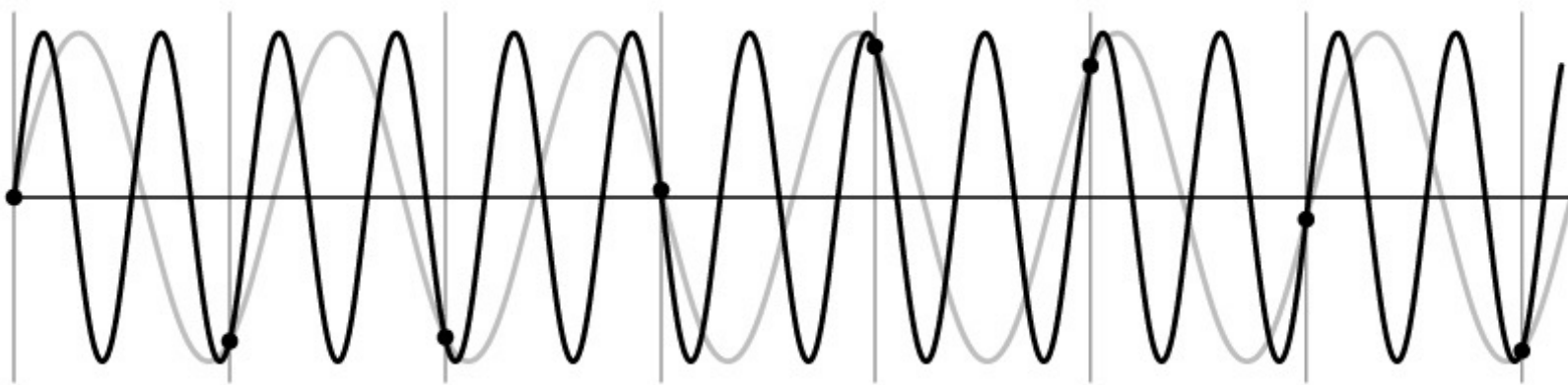
Undersampling

- 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



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

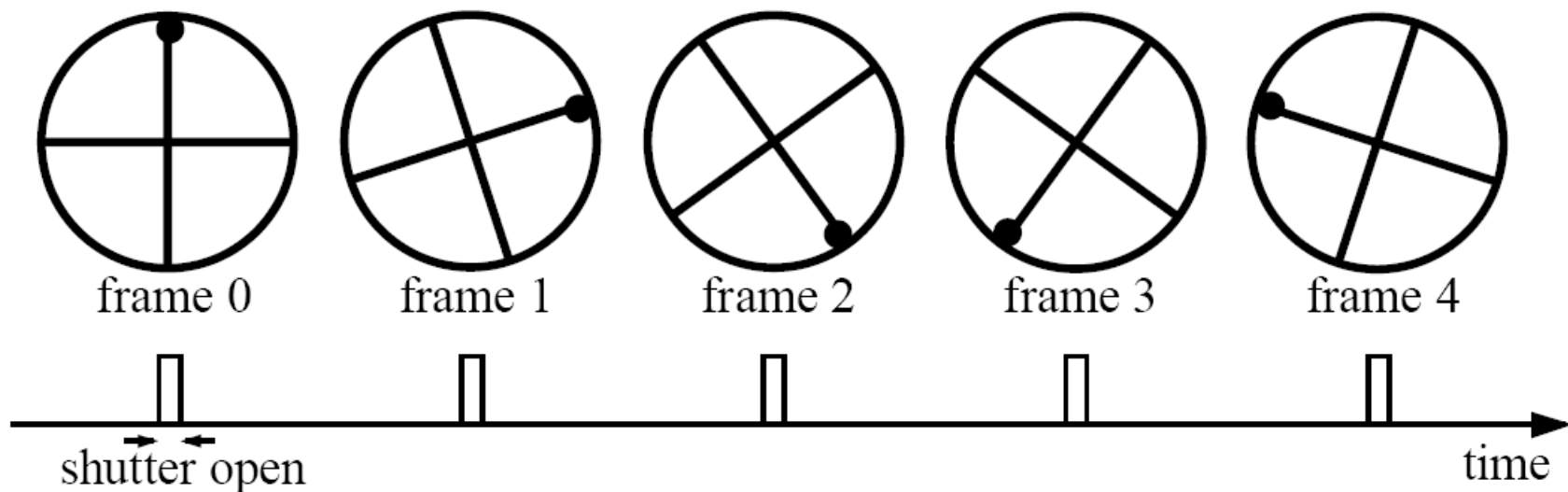


Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise).

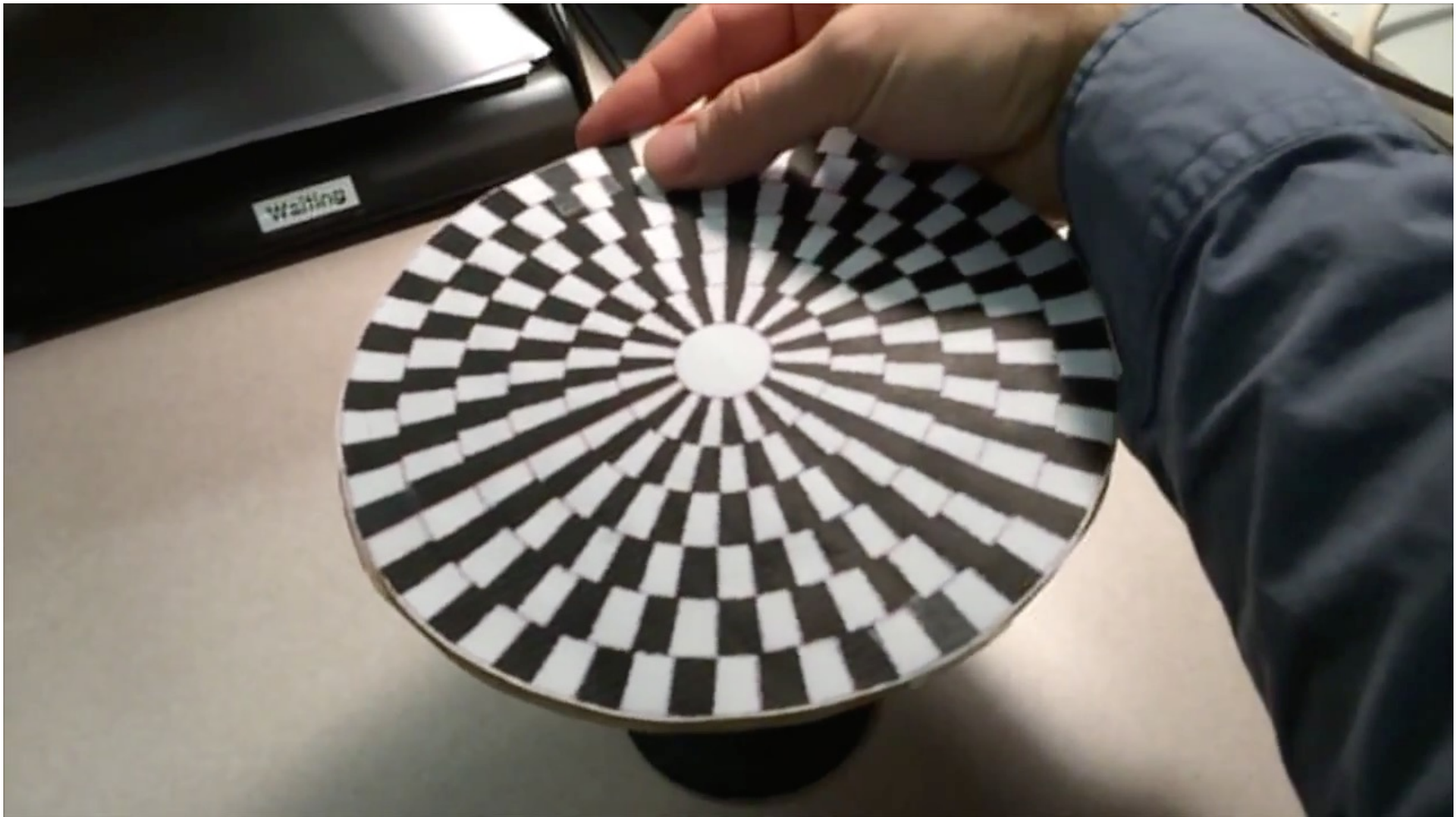
Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = $1/30$ sec. for video, $1/24$ sec. for film):



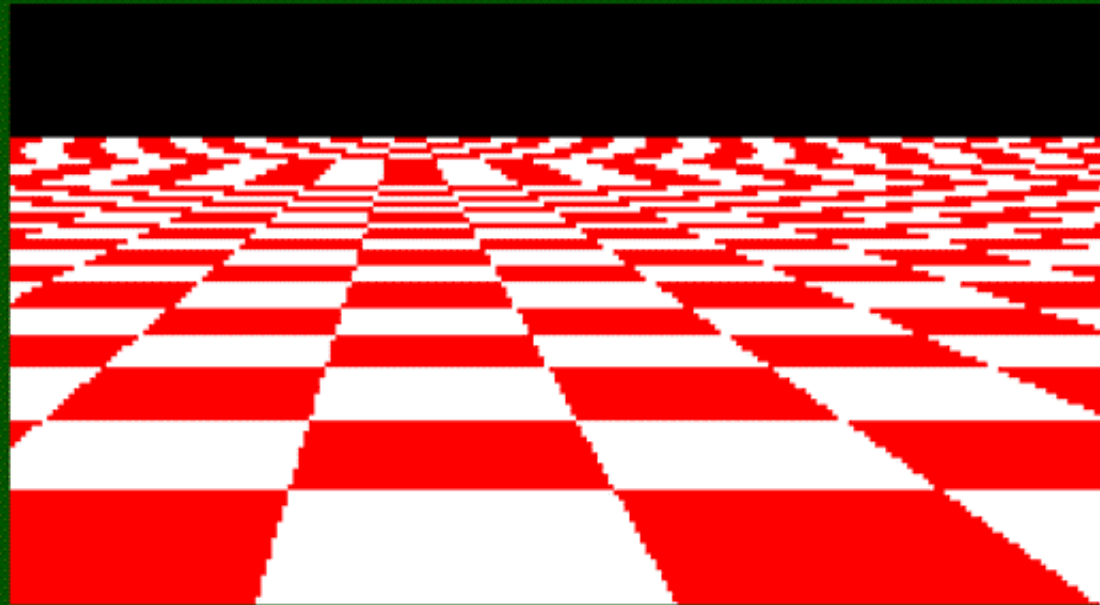
Without dot, wheel appears to be rotating slowly backwards!
(counterclockwise)

Wagon Wheel Illusion (False Motion)



Created by Jesse Mason, https://www.youtube.com/watch?v=QOwzkND_ooU

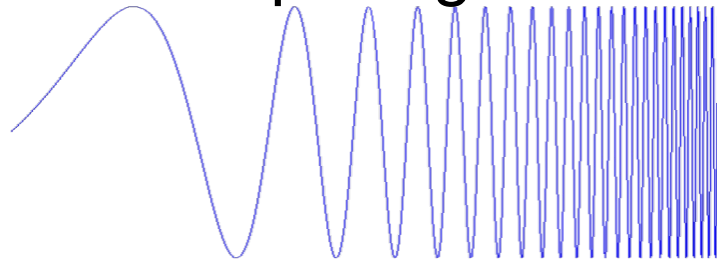
Aliasing in Synthetic Images



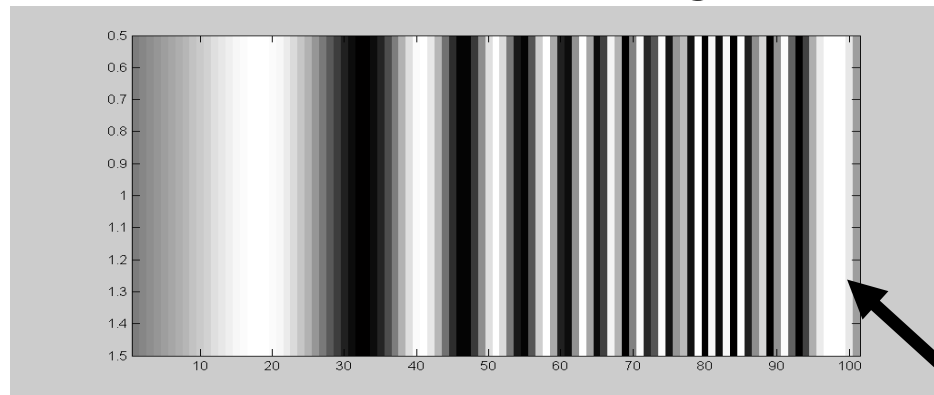
Disintegrating textures

What's Happening?

Input signal:



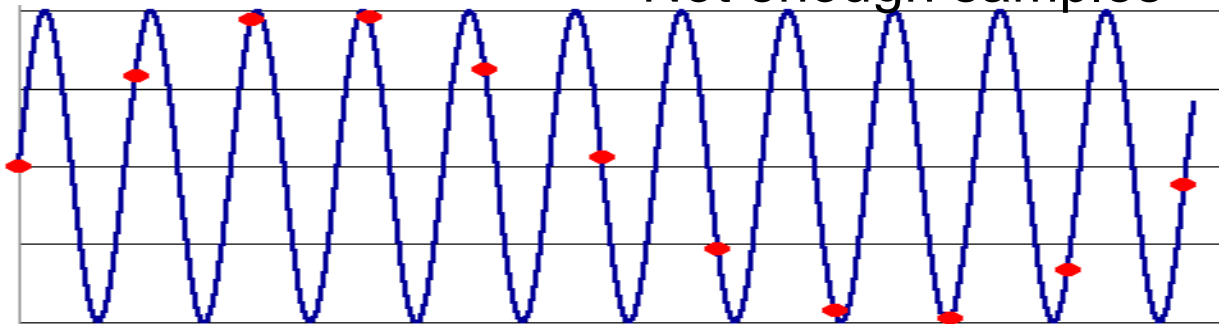
Plot as image:



`x = 0:.05:5; imagesc(sin((2.^x).*x))`

Alias!

Not enough samples



Aliasing in Natural Imagery



Aliasing in Image Processing



All sensor pixels



Delete odd rows and columns



512 x 512
original



32 x 32
thumbnail
(subsampled)

Antialiasing

What can we do about aliasing?

Sample more often

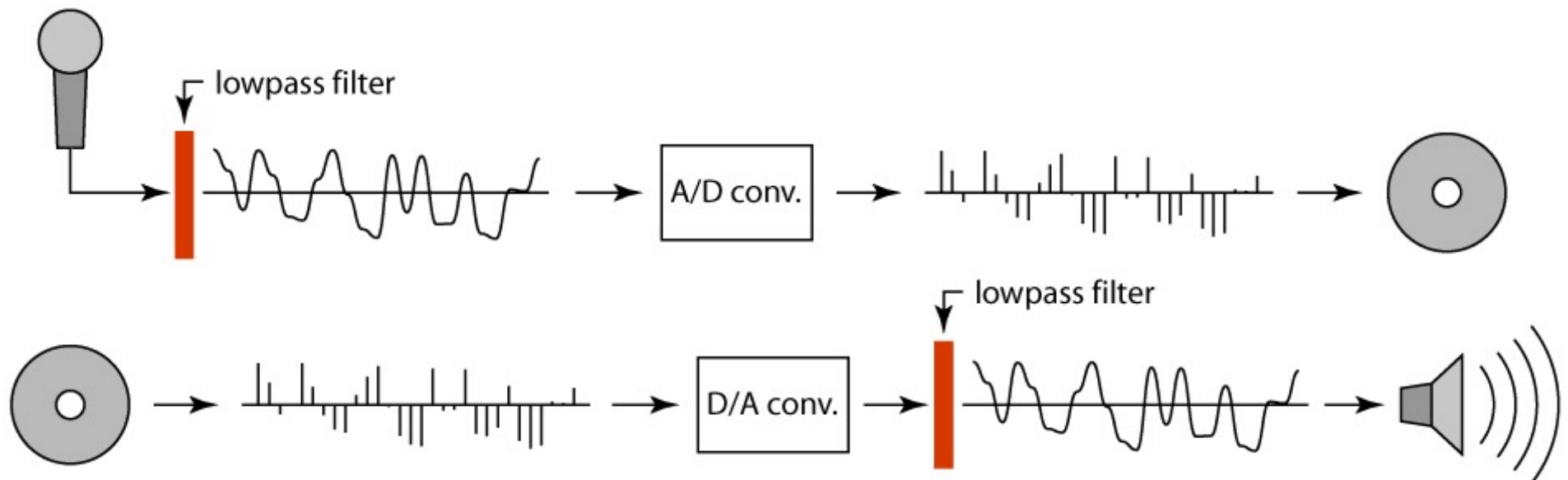
- Join the Mega-Pixel craze of the photo industry
- But this can't go on forever

Make the signal less “wiggly”

- Get rid of some high frequencies
- Will lose information
- But it's better than aliasing

Preventing aliasing

- Introduce lowpass filters:
 - remove high frequencies leaving only safe, low frequencies
 - choose lowest frequency in reconstruction (disambiguate)

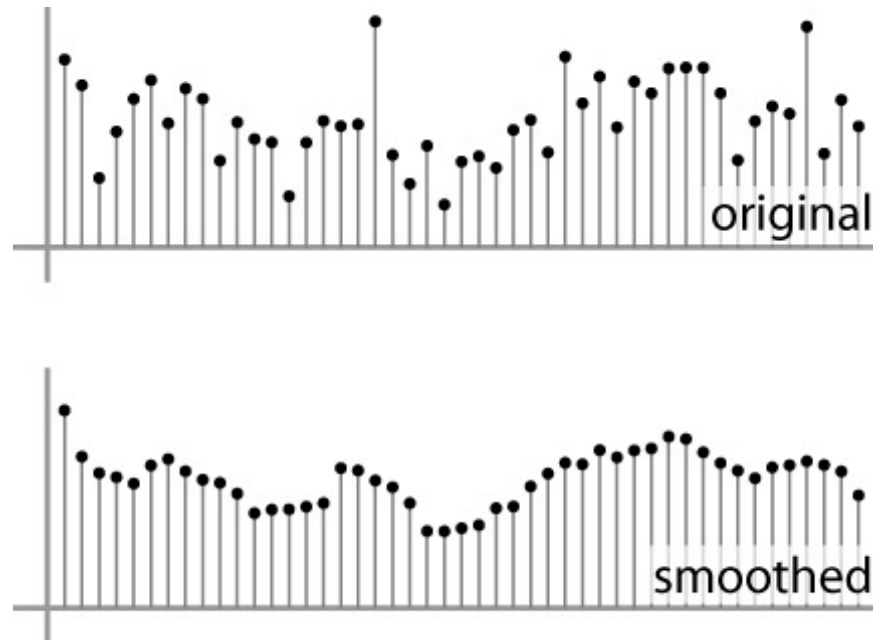


Linear filtering: a key idea

- Transformations on signals; e.g.:
 - bass/treble controls on stereo
 - blurring/sharpening operations in image editing
 - smoothing/noise reduction in tracking
- Key properties
 - linearity: $\text{filter}(f + g) = \text{filter}(f) + \text{filter}(g)$
 - shift invariance: behavior invariant to shifting the input
 - delaying an audio signal
 - sliding an image around
- Can be modeled mathematically by *convolution*

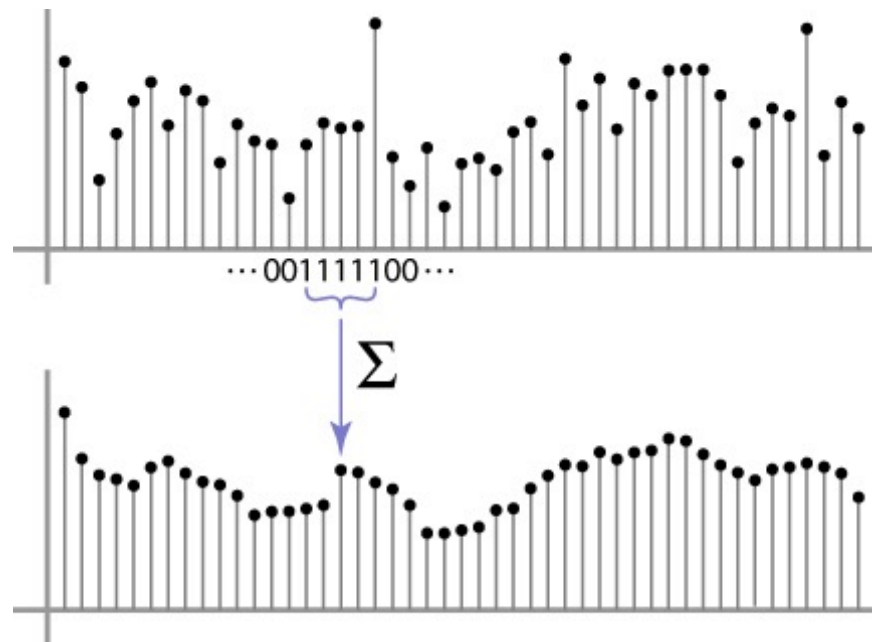
Moving Average

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing



Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5



In 2D: box filter

$$\frac{1}{9} h[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

$$g[m, n] = \sum_{k, l} h[k, l] f[m + k, n + l]$$

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10							

$$g[m, n] = \sum_{k, l} h[k, l] f[m + k, n + l]$$

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10	20						

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10	20	30					

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10	20	30	30				

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10	20	30	30				

Image filtering

$$h[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g[\cdot, \cdot]$

	0	10	20	30	30				
						?			
				50					

Image filtering

$$h[\cdot, \cdot] \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[\cdot, \cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$g[\cdot, \cdot]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Cross-correlation

Let F be the image, H be the kernel (of size $2k+1 \times 2k+1$), 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

Box Filter

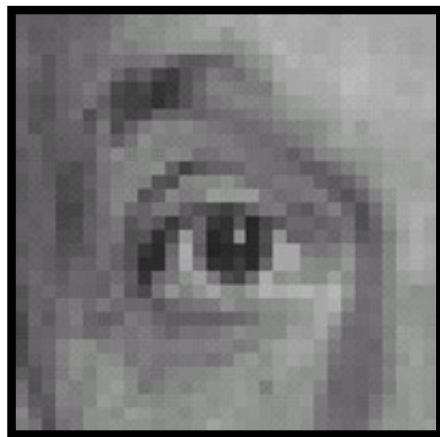
What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$$\frac{1}{9} h[\cdot, \cdot]$$

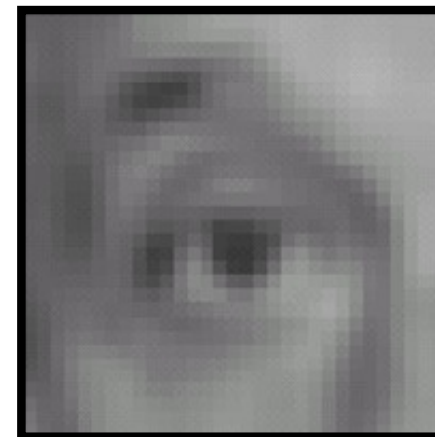
1	1	1
1	1	1
1	1	1

Linear filters: examples



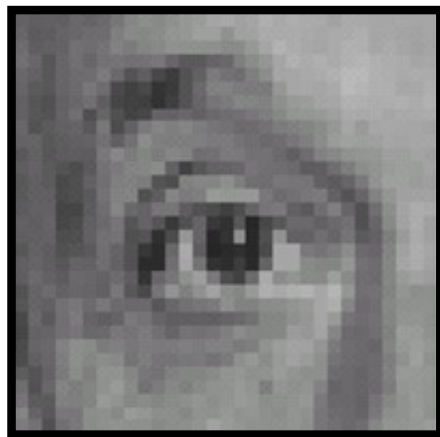
Original

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



Blur (with a mean filter)

Practice with linear filters

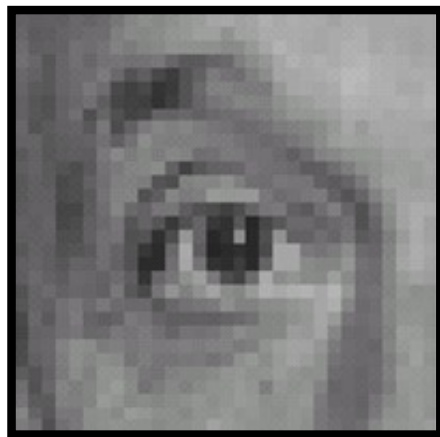


Original

0	0	0
0	1	0
0	0	0

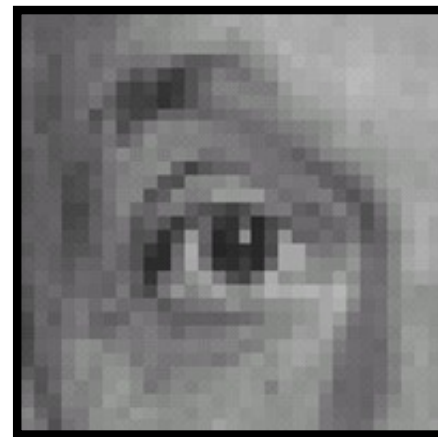
?

Practice with linear filters



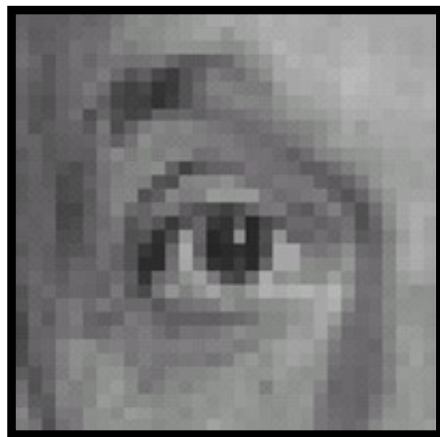
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

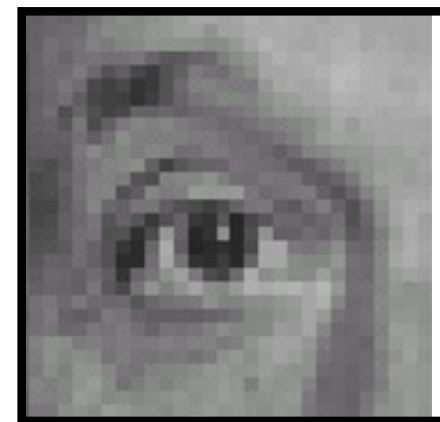
?

Practice with linear filters



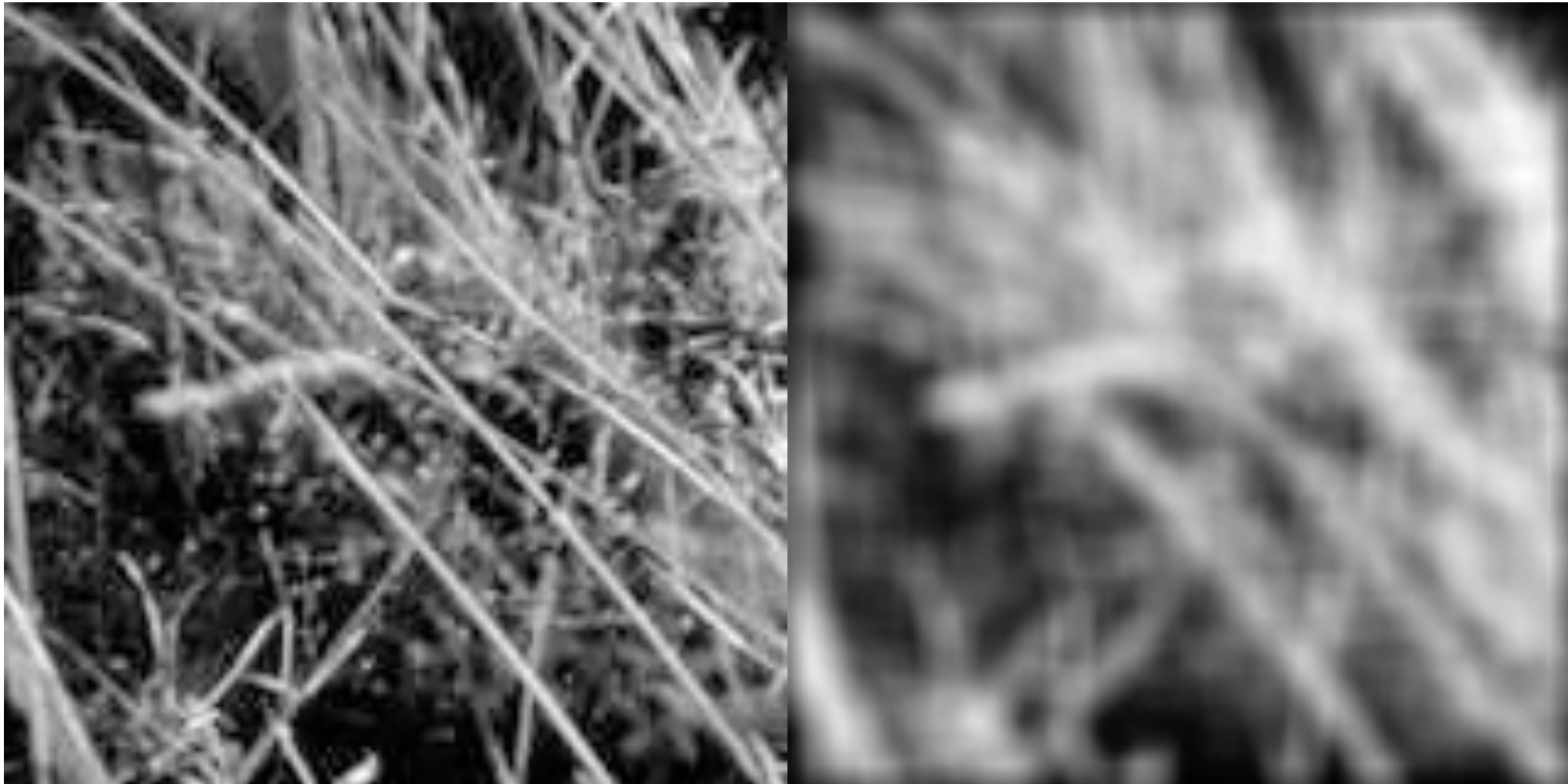
Original

0	0	0
0	0	1
0	0	0



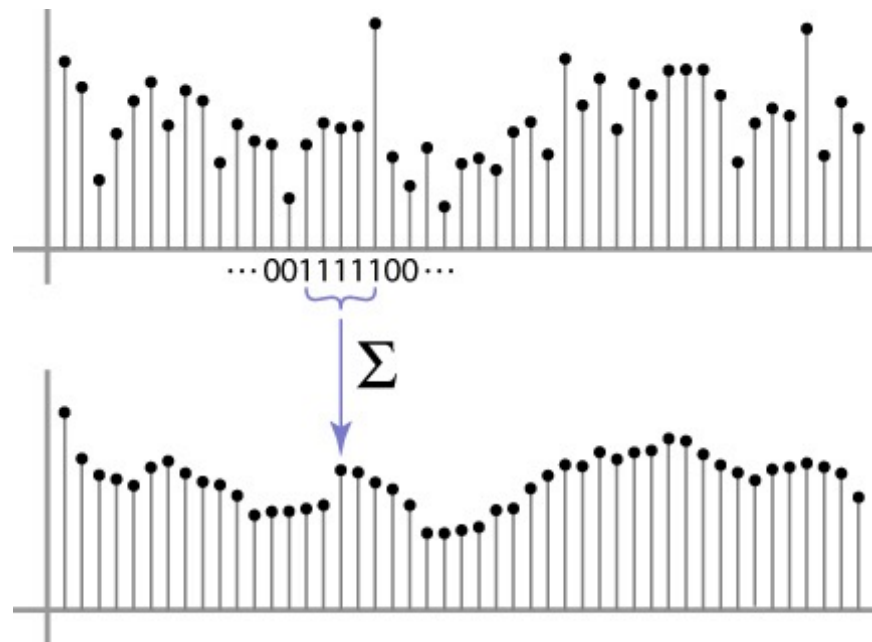
Shifted left
By 1 pixel

Back to the box filter



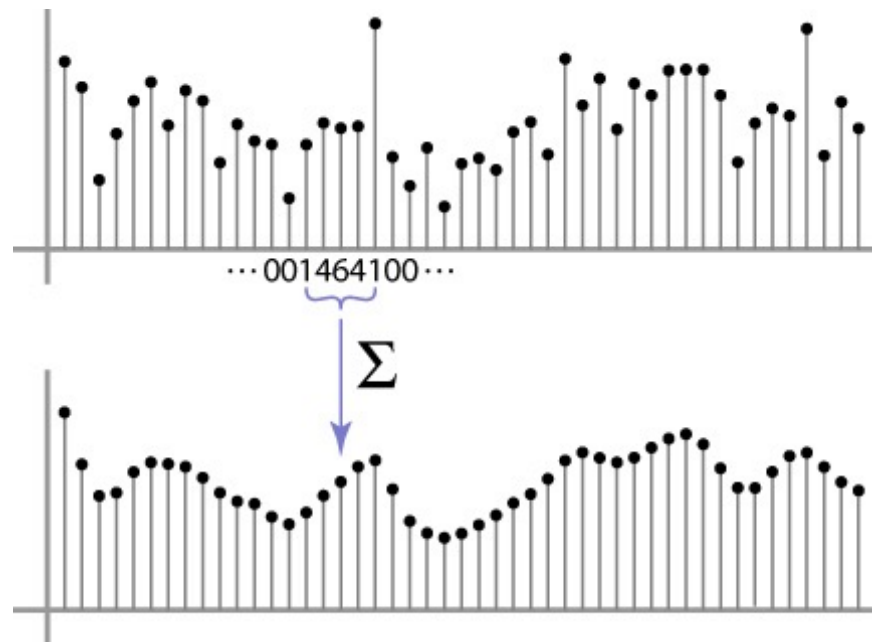
Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5



Weighted Moving Average

- bell curve (gaussian-like) weights [..., 1, 4, 6, 4, 1, ...]



Moving Average In 2D

What are the weights H ?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$H[u, v]$

Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

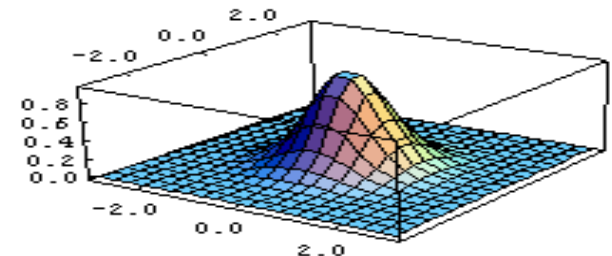
$F[x, y]$

$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

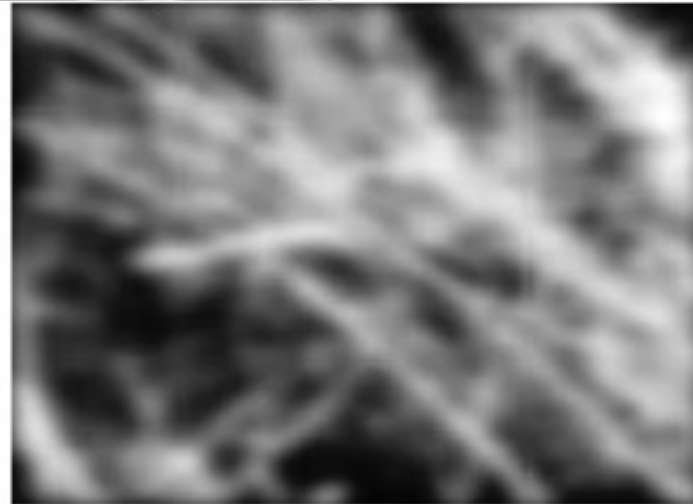
$H[u, v]$

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



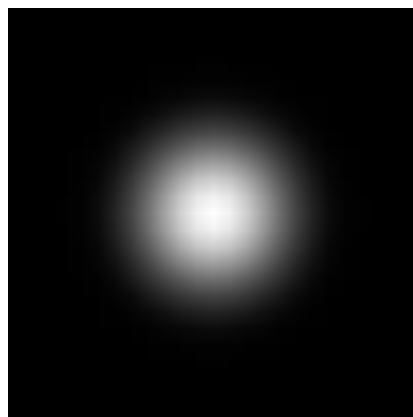
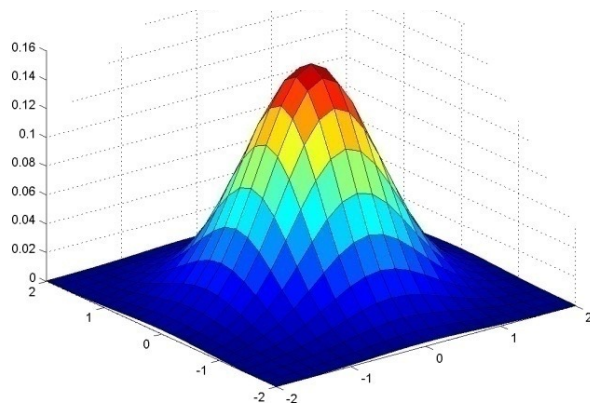
This kernel is an approximation of a Gaussian function:

Mean (Box) vs. Gaussian filtering



Important filter: Gaussian

Weight contributions of neighboring pixels by nearness



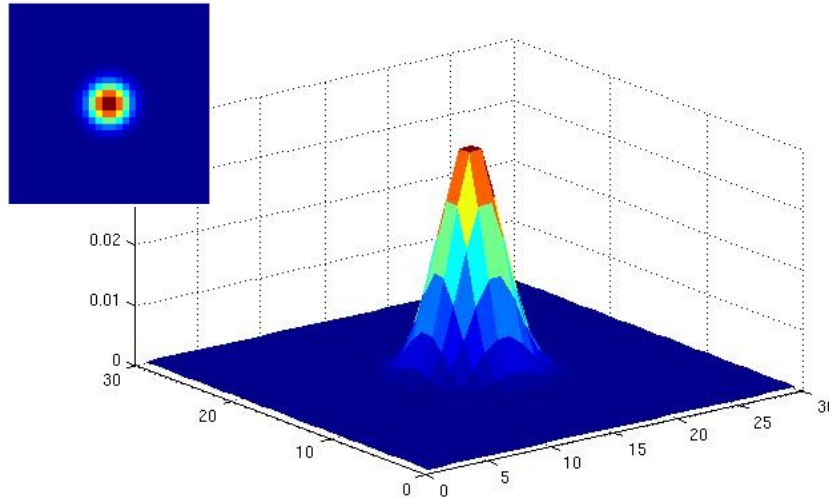
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

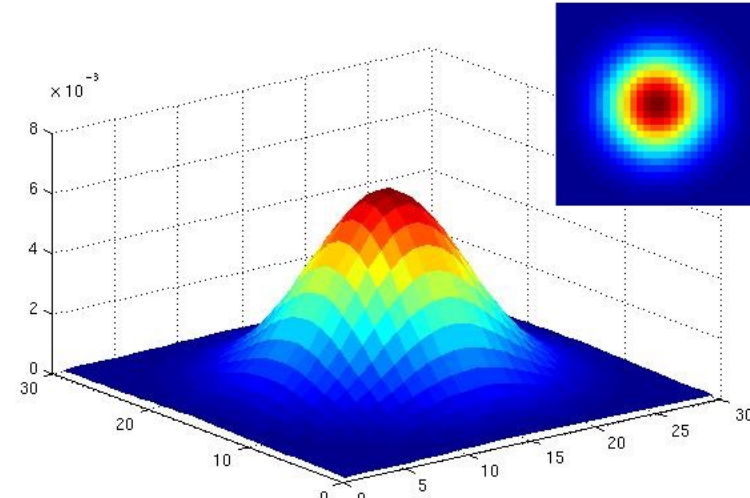
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian Kernel

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



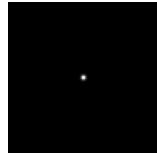
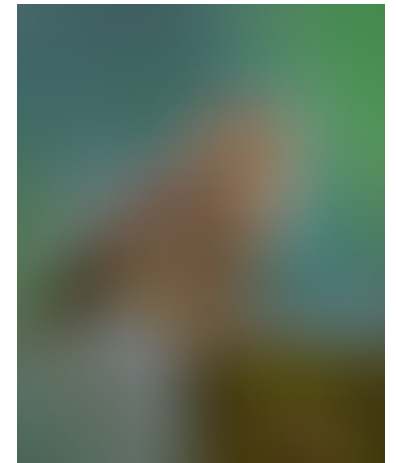
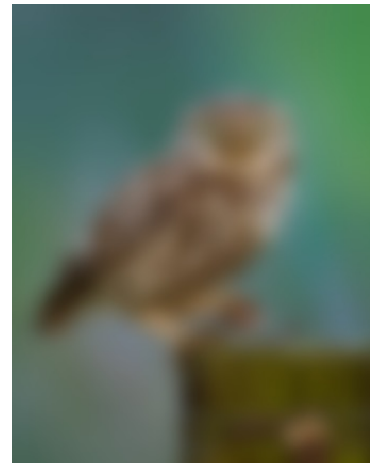
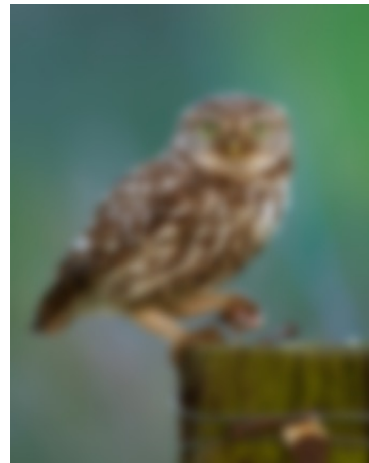
$\sigma = 2$ with 30×30
kernel



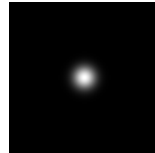
$\sigma = 5$ with 30×30
kernel

- Standard deviation σ : determines extent of smoothing

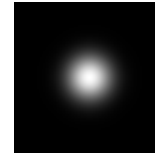
Gaussian filters



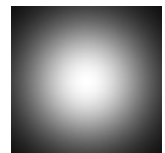
$\sigma = 1$ pixel



$\sigma = 5$ pixels



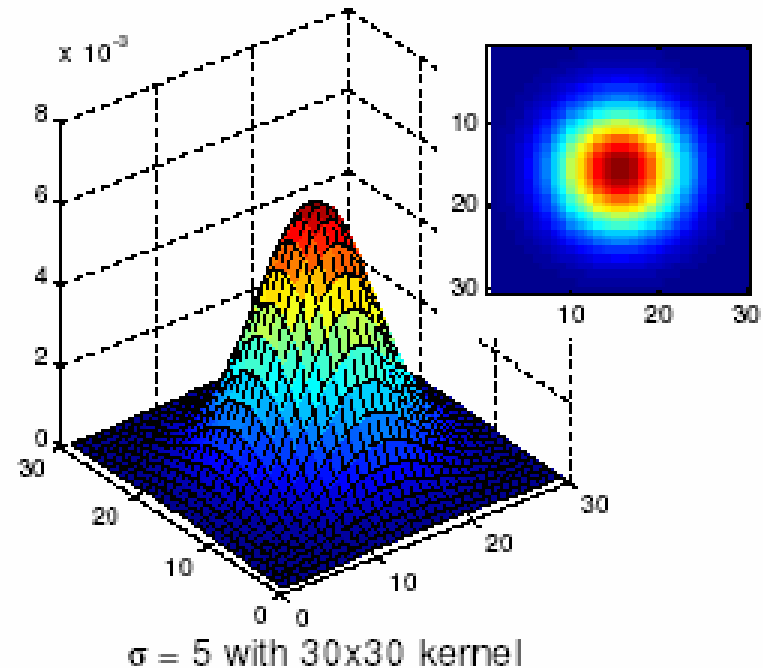
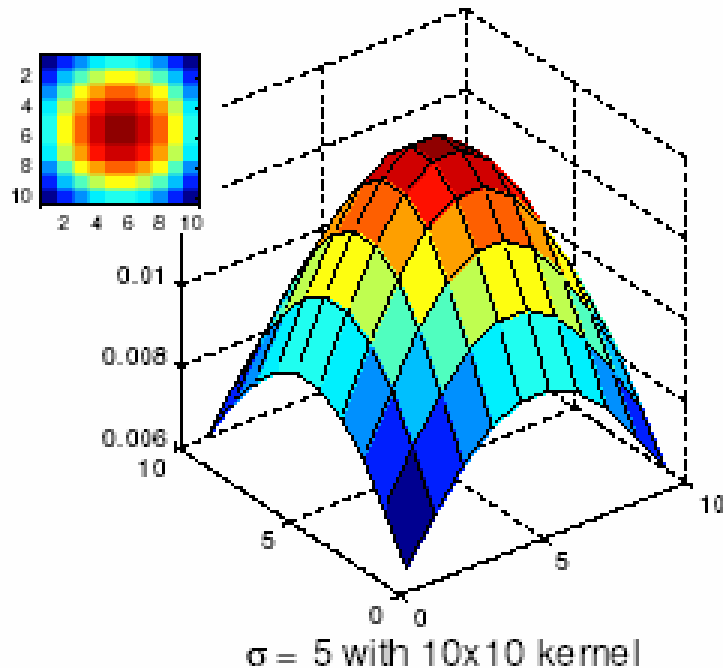
$\sigma = 10$ pixels



$\sigma = 30$ pixels

Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels



Practical matters

How big should the filter be?

Values at edges should be near zero

Rule of thumb for Gaussian: set filter half-width to about 3σ

