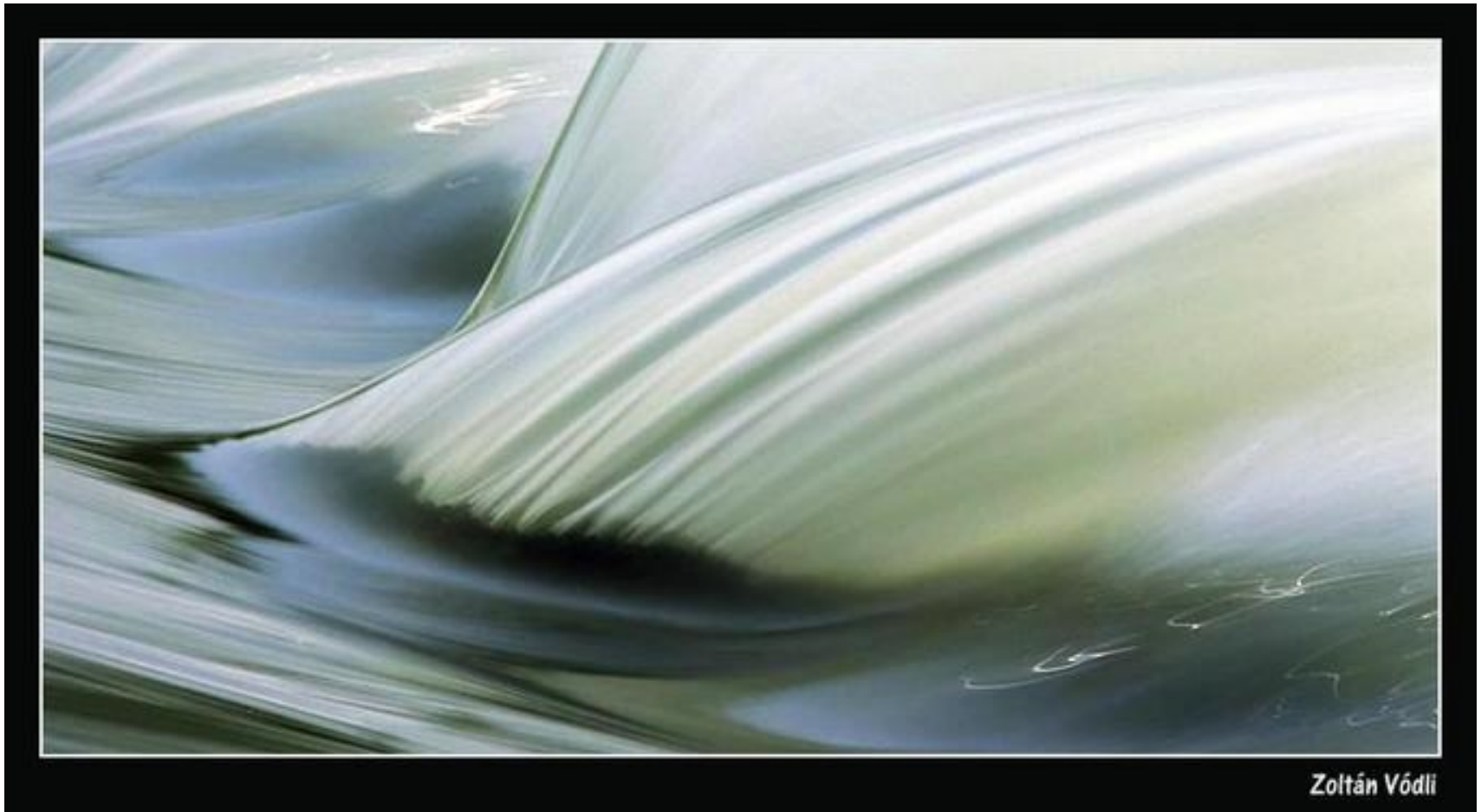# Point Processing & Filtering



Zoltán Vódli

CS194: Image Manipulation & Computational Photography
Alexei Efros, UC Berkeley, Fall 2014
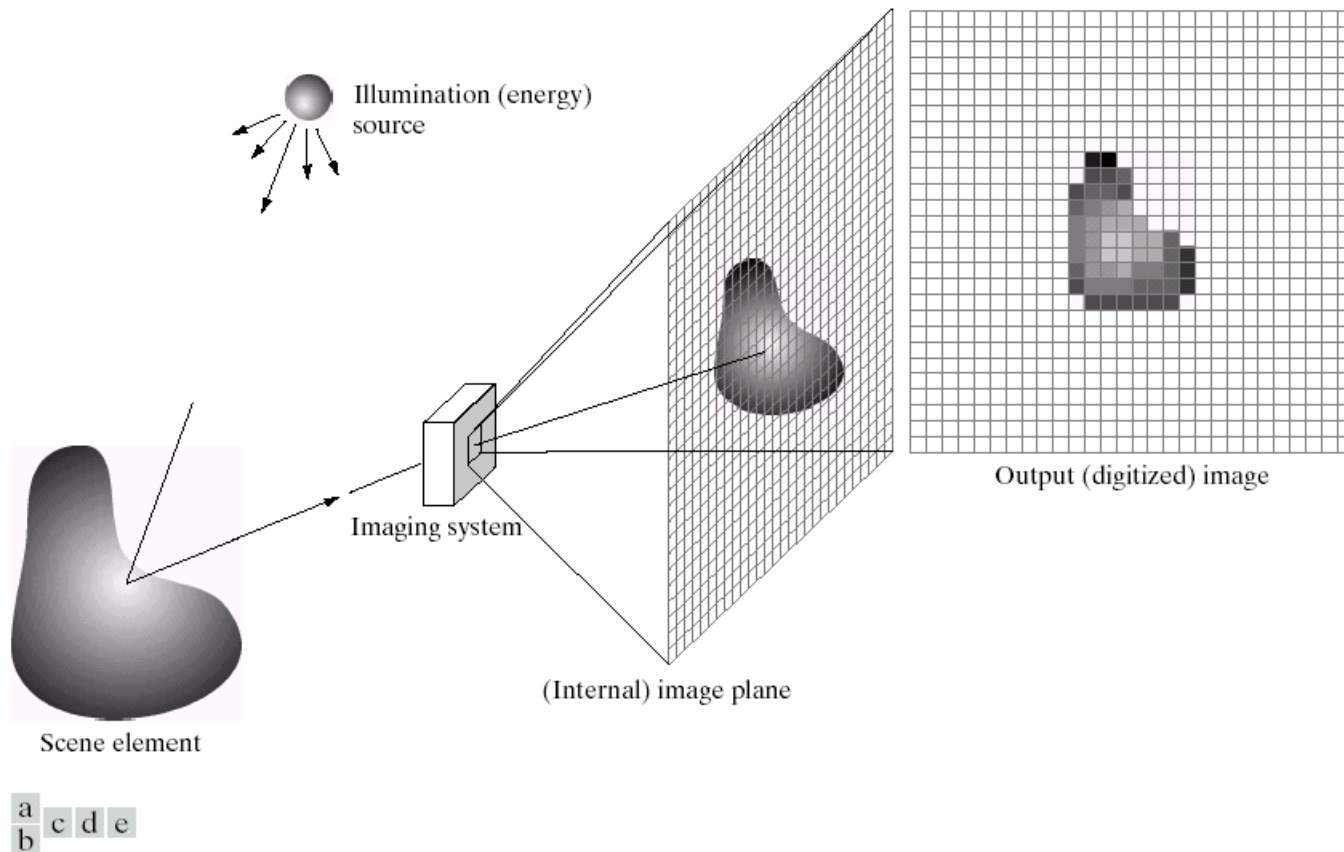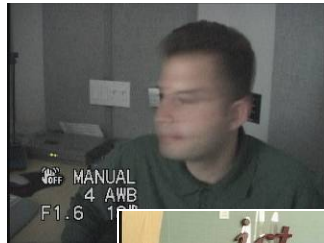
# Image Formation



**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.

$$f(x,y) = reflectance(x,y) * illumination(x,y)$$

*Reflectance in [0,1], illumination in [0,inf]*

# Problem: Dynamic Range



1

1500

25,000

400,000

2,000,000,000

The real world is
High dynamic range

# Long Exposure

$10^{-6}$ High dynamic range $10^6$

Real world

$10^{-6}$ $10^6$

Picture

0 to 255

# Short Exposure

$10^{-6}$  High dynamic range  $10^6$

Real world

$10^{-6}$  $10^6$

Picture

0 to 255
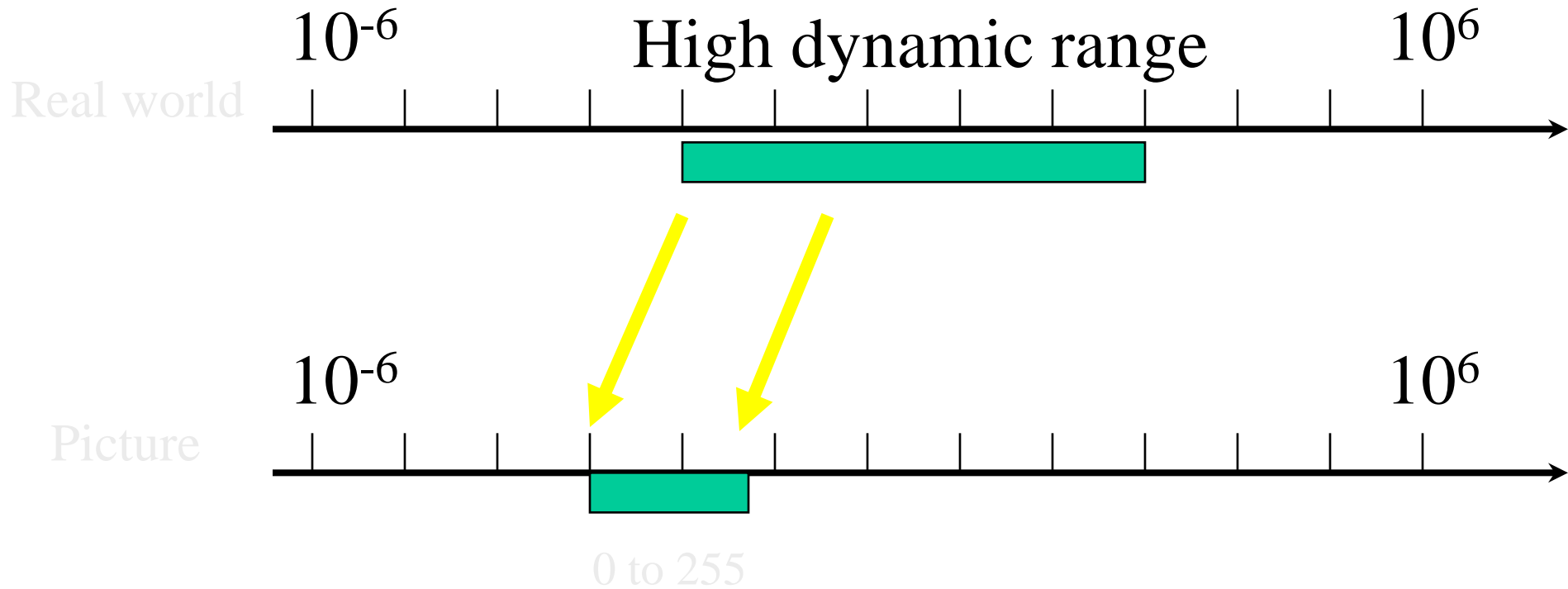
# Image Acquisition Pipeline

**Lens**

scene
radiance

$(W/sr/m^2)$

sensor
irradiance

**Shutter**

$\int$

$\Delta t$

sensor
exposure

**CCD**
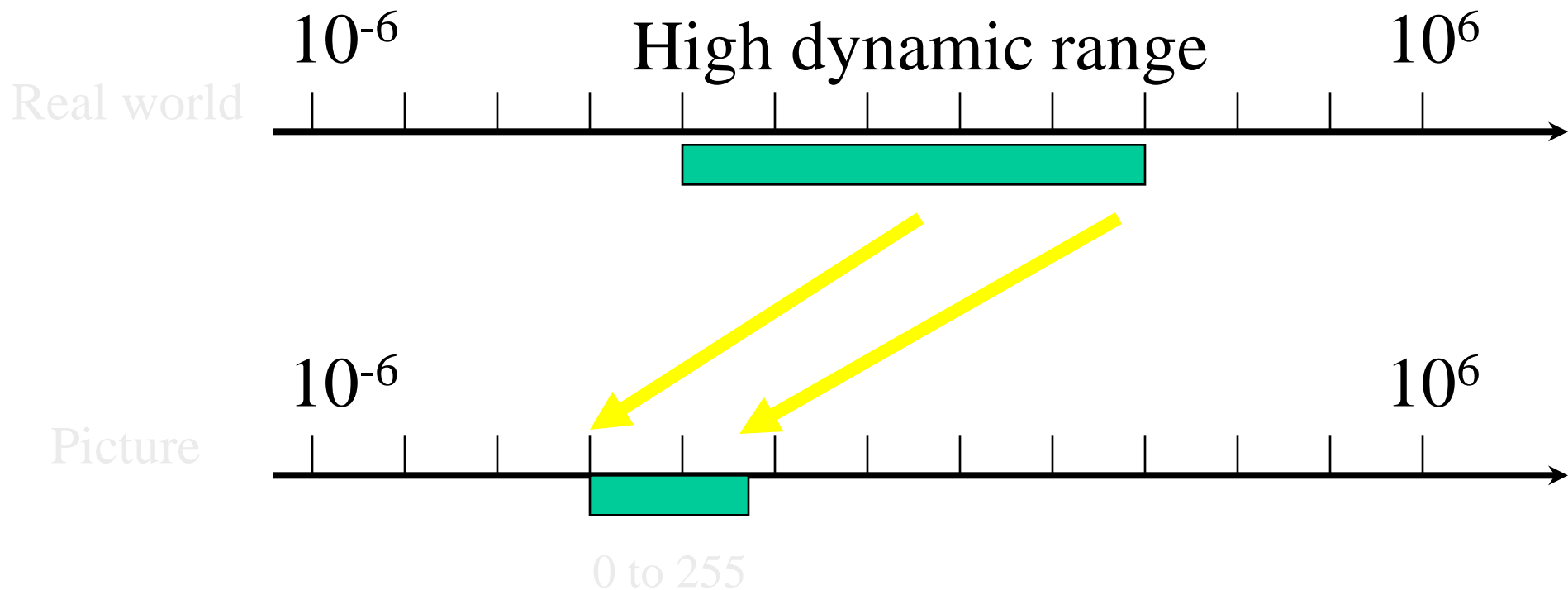
analog
voltages

**ADC**

digital
values

**Remapping**

pixel
values
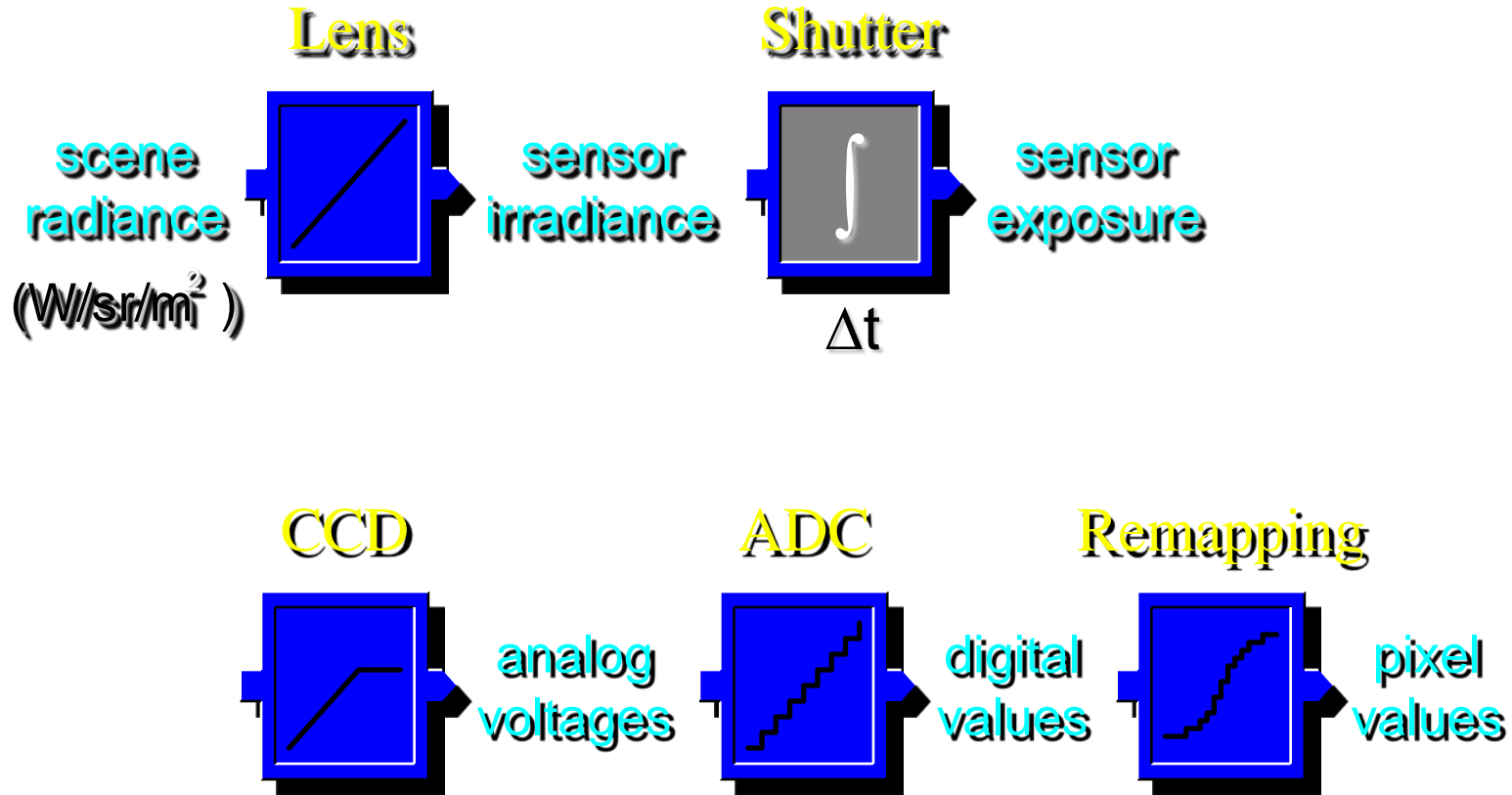
# Simple Point Processing: Enhancement



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.)
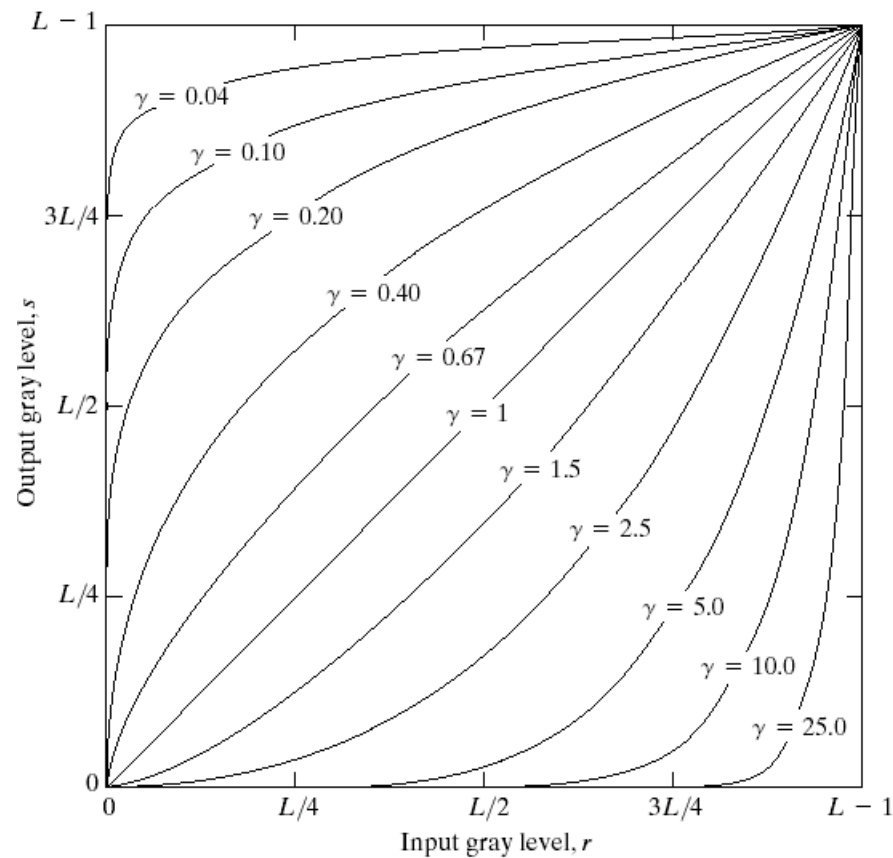
# Power-law transformations
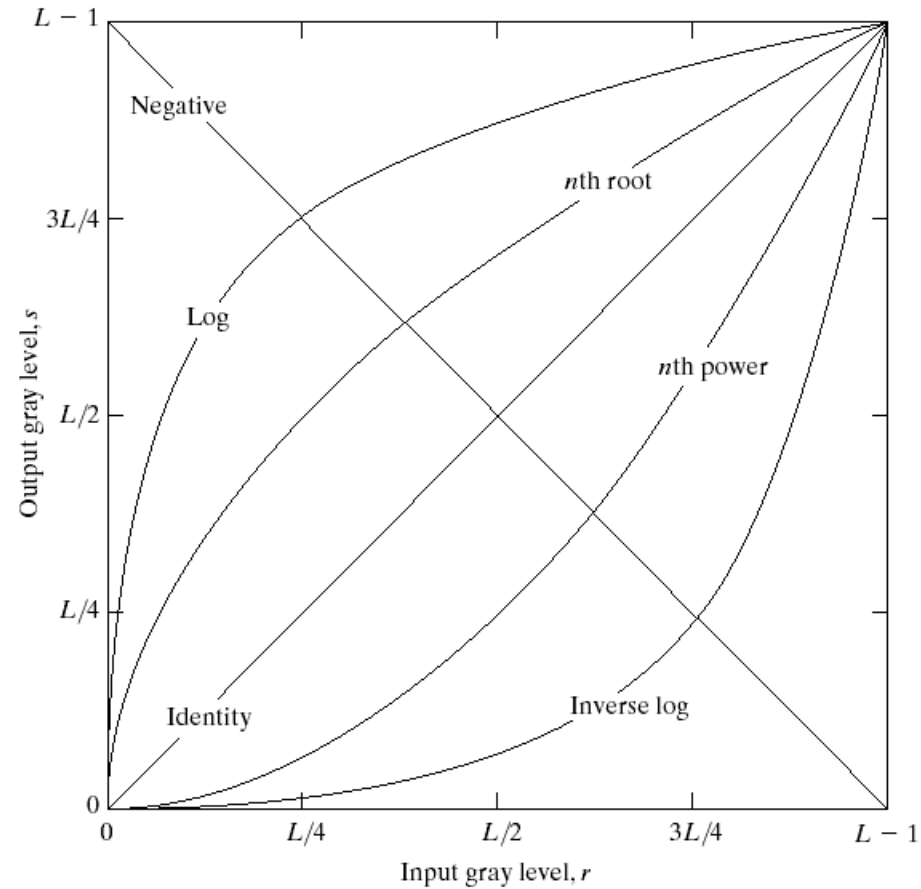


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of $\gamma$ ($c = 1$ in all cases).
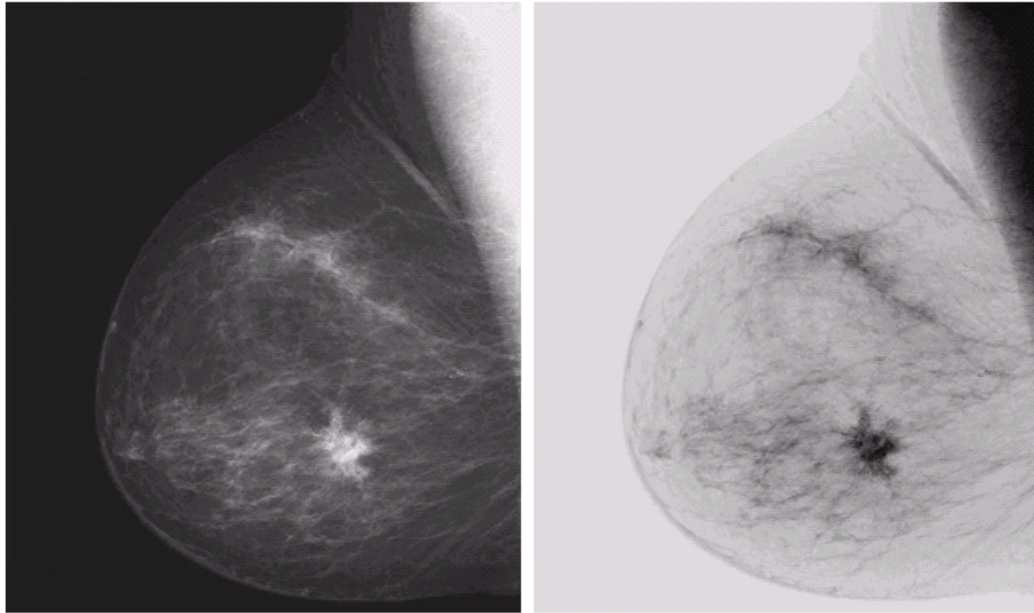
$$s = cr^\gamma$$

# 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.)
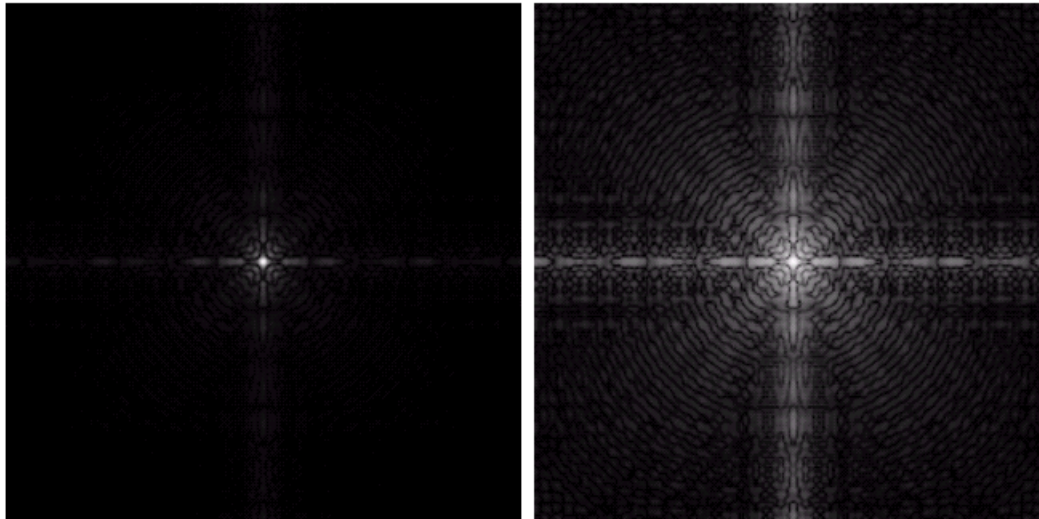
# Log

**FIGURE 3.5**
(a) Fourier
spectrum.
(b) Result of
applying the log
transformation
given in
Eq. (3.2-2) with
$c = 1$.

# Contrast Stretching
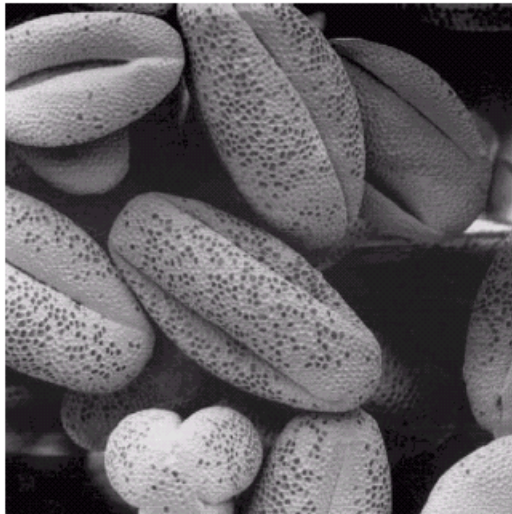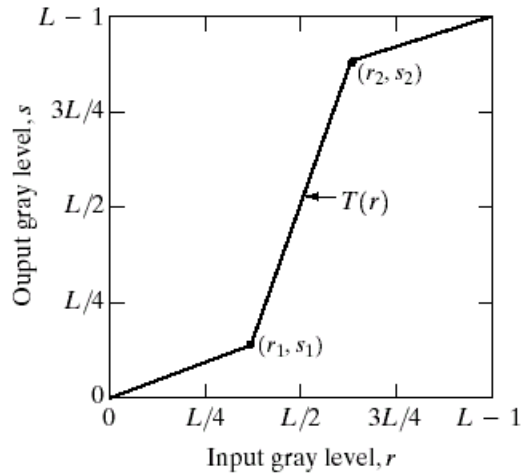
**FIGURE 3.10** Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

# Image Histograms



Dark image

Bright image

Low-contrast image

High-contrast image

Cumulative Histograms

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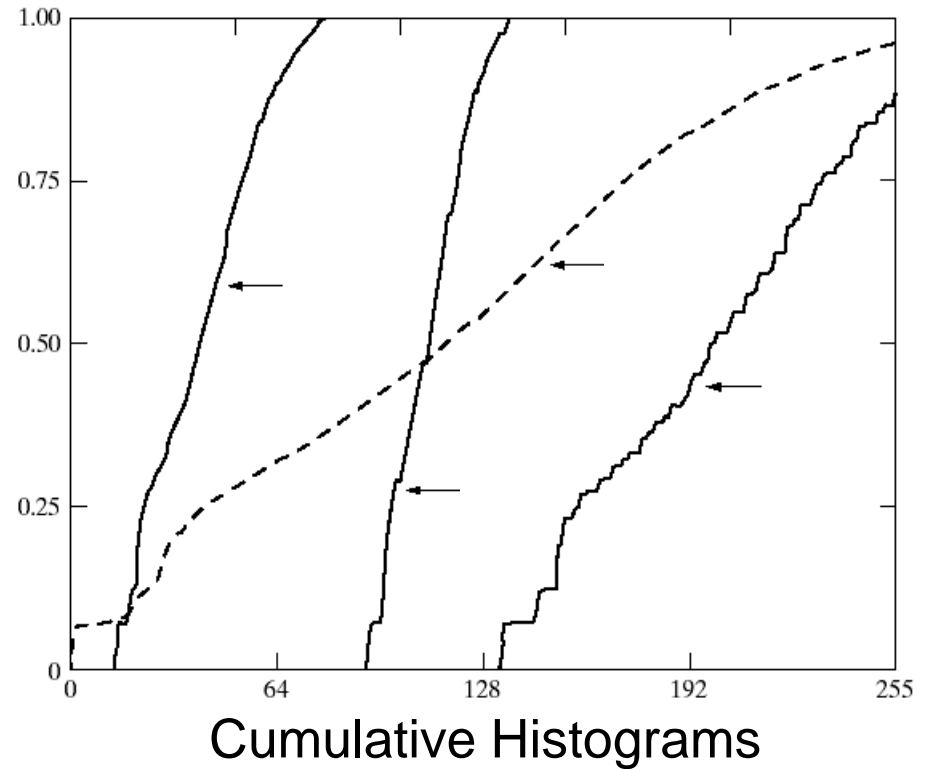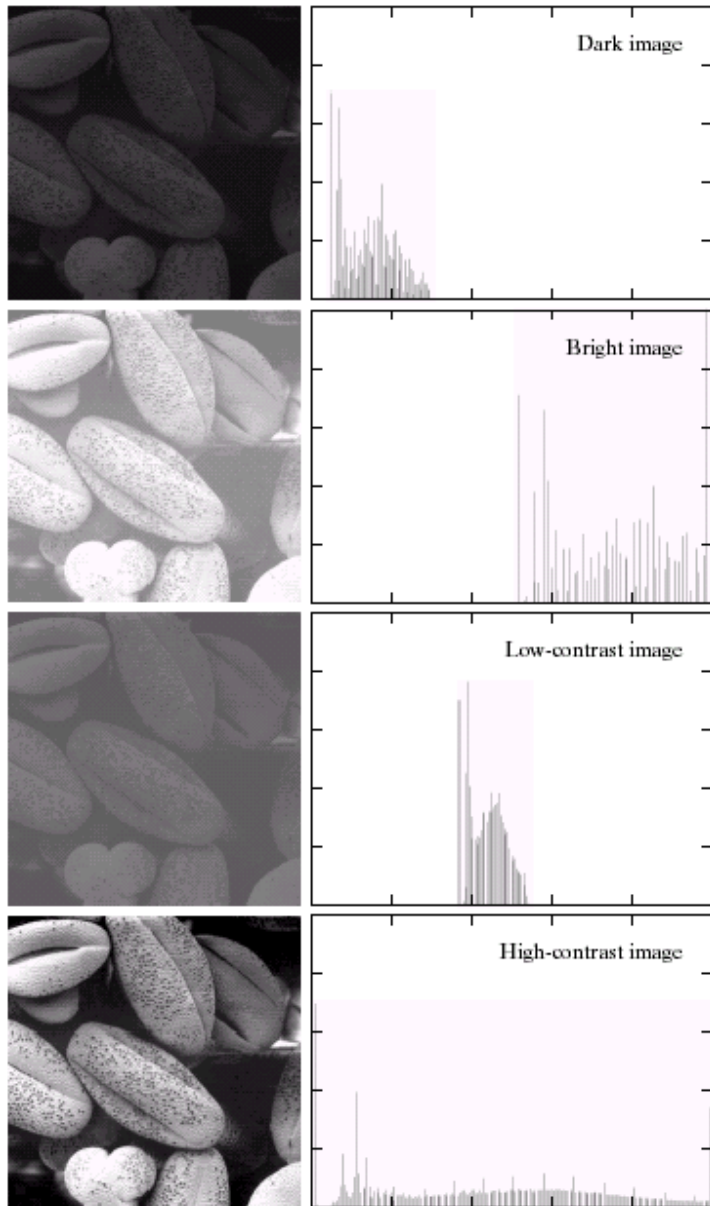
**FIGURE 3.15** Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

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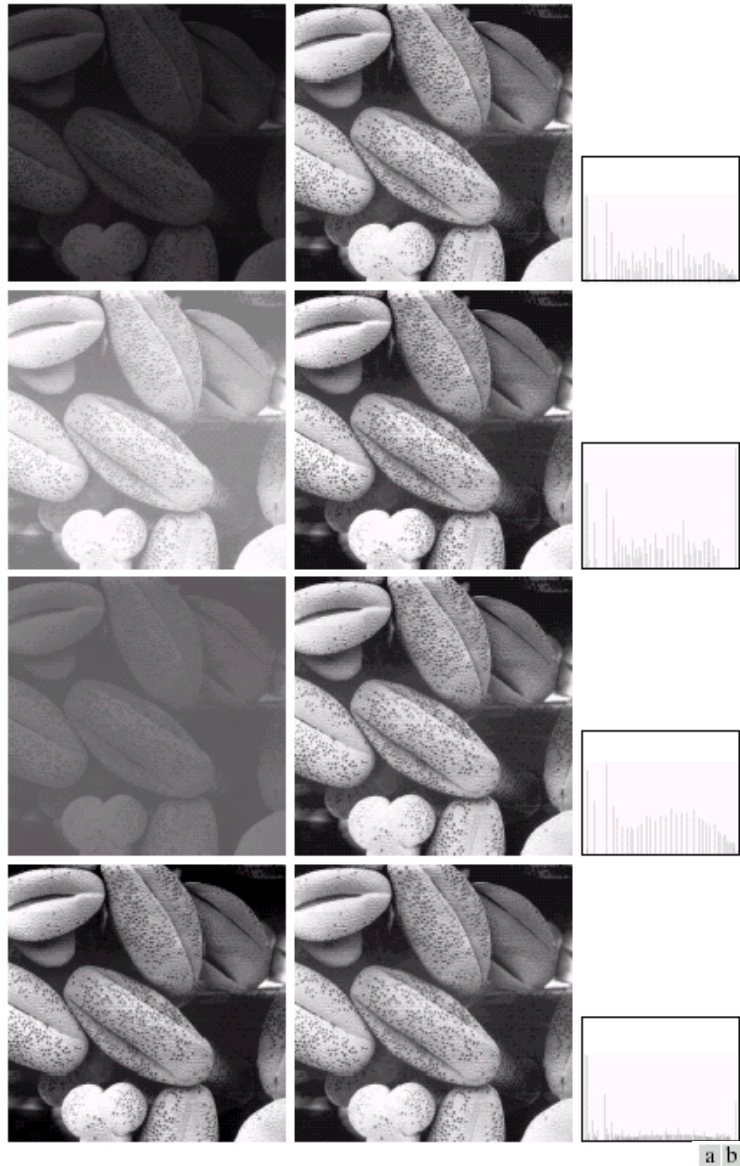
**FIGURE 3.17** (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

# 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.
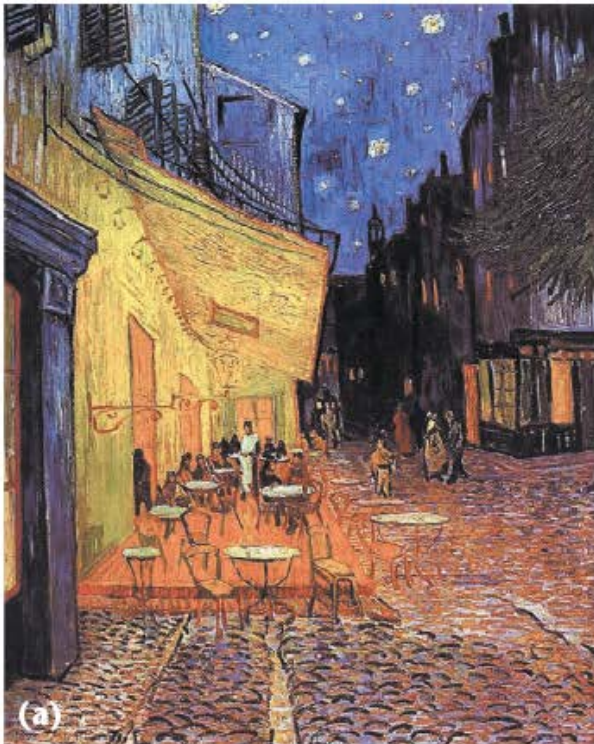
# Limitations of Point Processing

Q: What happens if I reshuffle all pixels within
   the image?
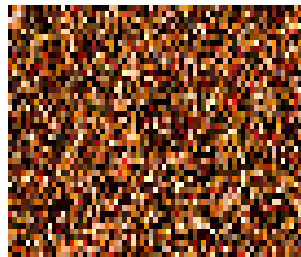


A: It's histogram won't change.  No point
   processing will be affected…

# What is an image?

We can think of an **image** as a function, *f*, from $R^2$ to 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*]x[*c,d*] → [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

# Sampling and Reconstruction



Illumination (energy) source

Scene element

Imaging system

(Internal) image plane
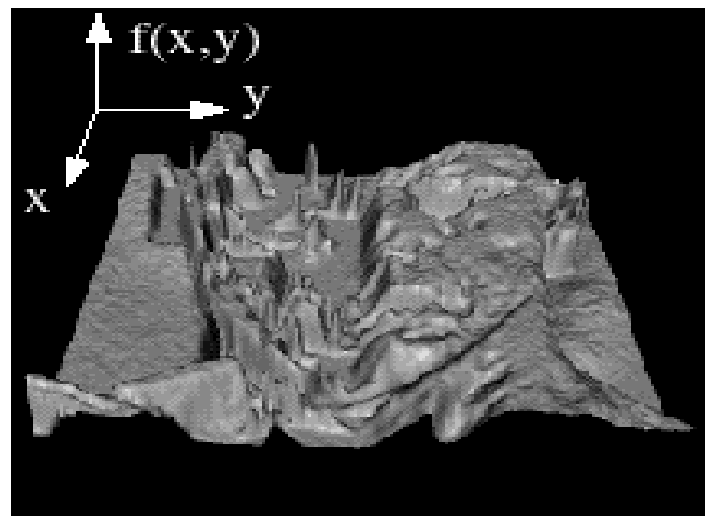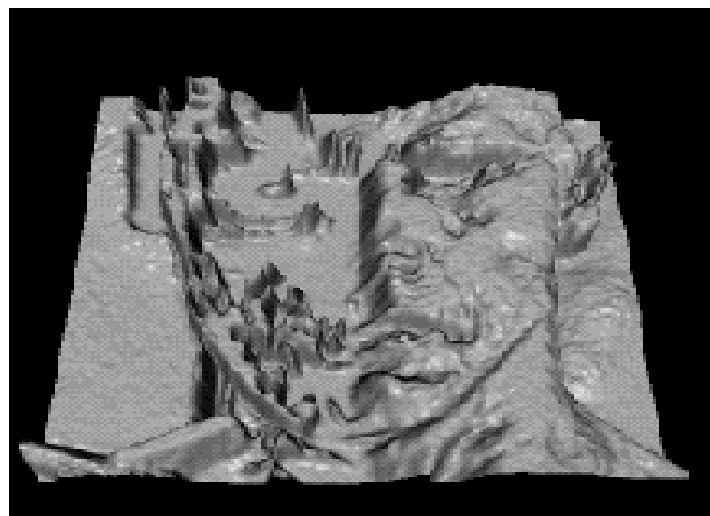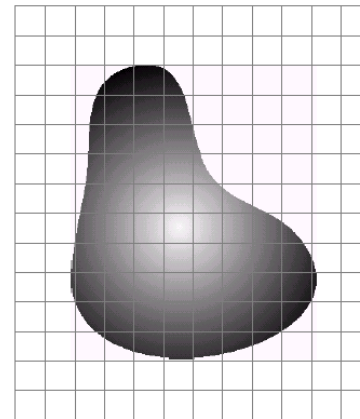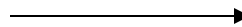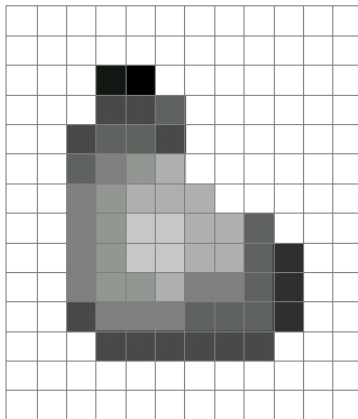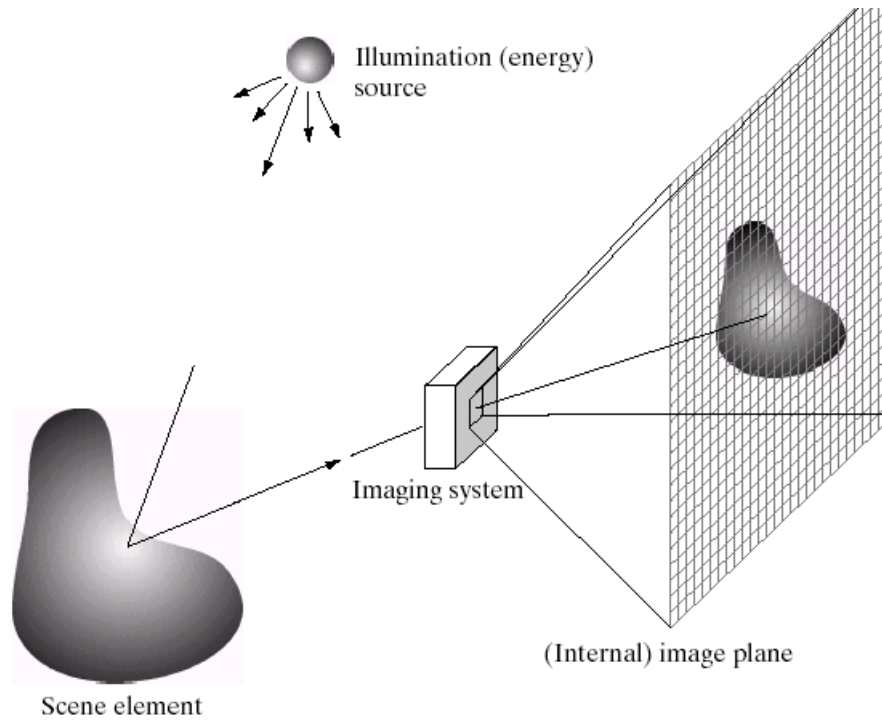
# Sampled representations

- How to store and compute with continuous functions?

- Common scheme for representation: samples
  - write down the function's values at many points
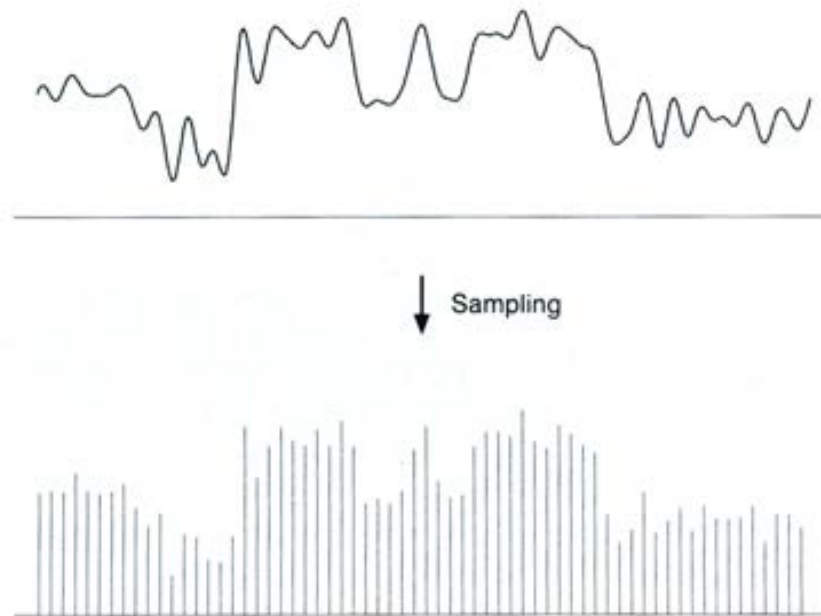


Sampling

# 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



Reconstruction

# 1D Example: Audio



low                              high

frequencies
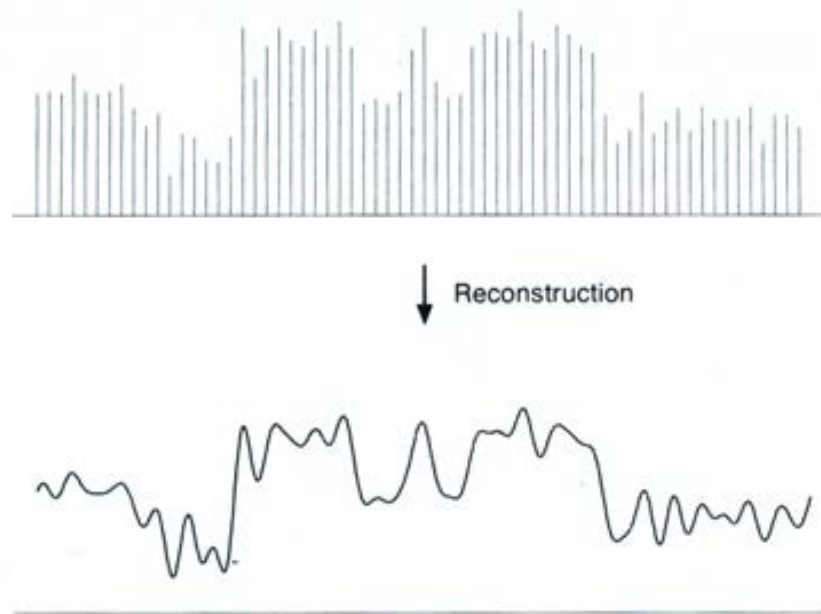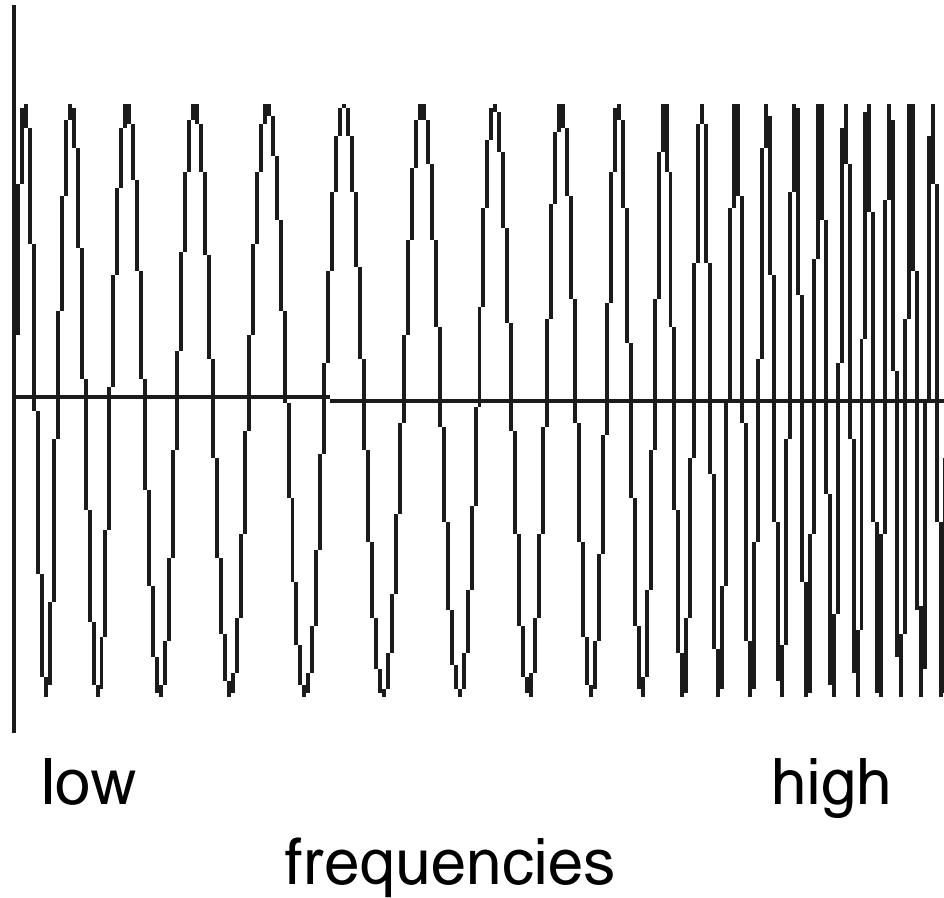
# Sampling in digital audio

- 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?
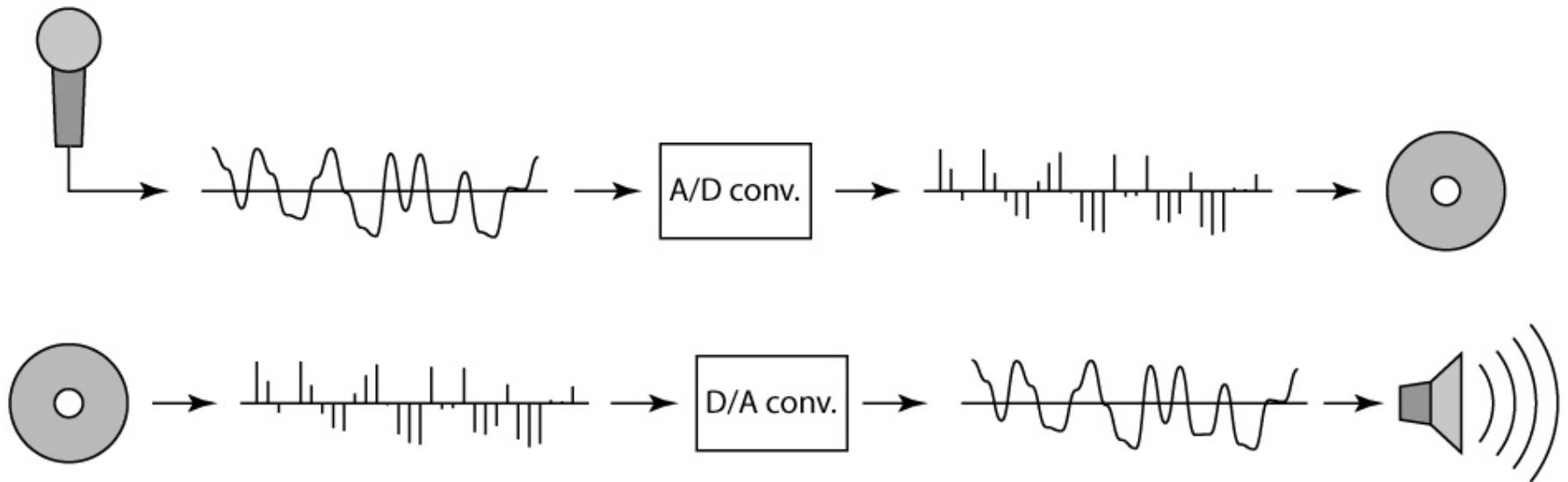
# Sampling and Reconstruction

- Simple example: a sign wave
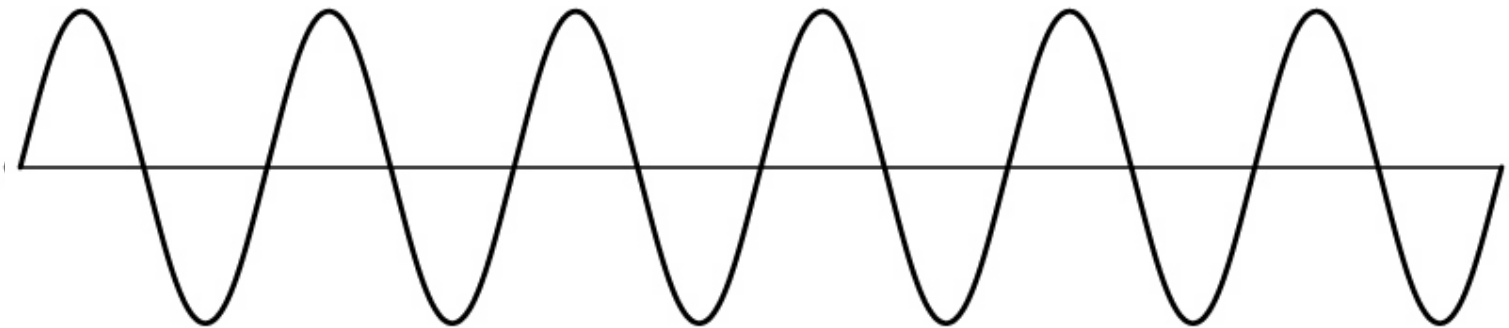
# **Undersampling**

- What if we "missed" things between the samples?

- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
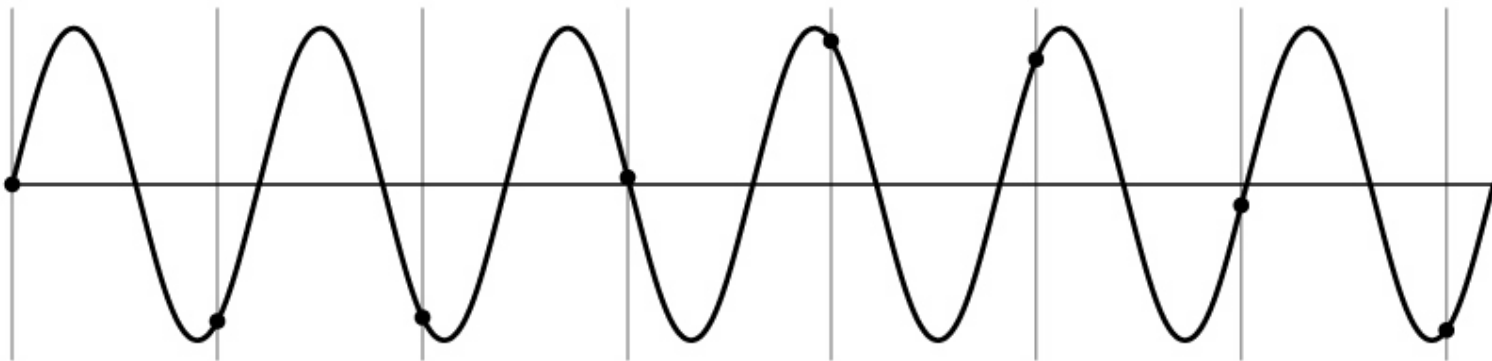
# Undersampling

- 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
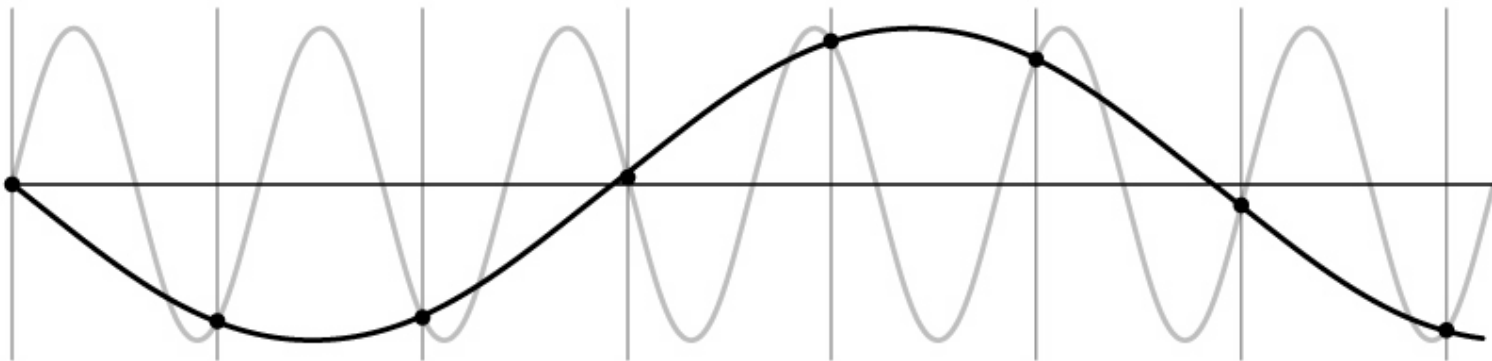
# Undersampling

- 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
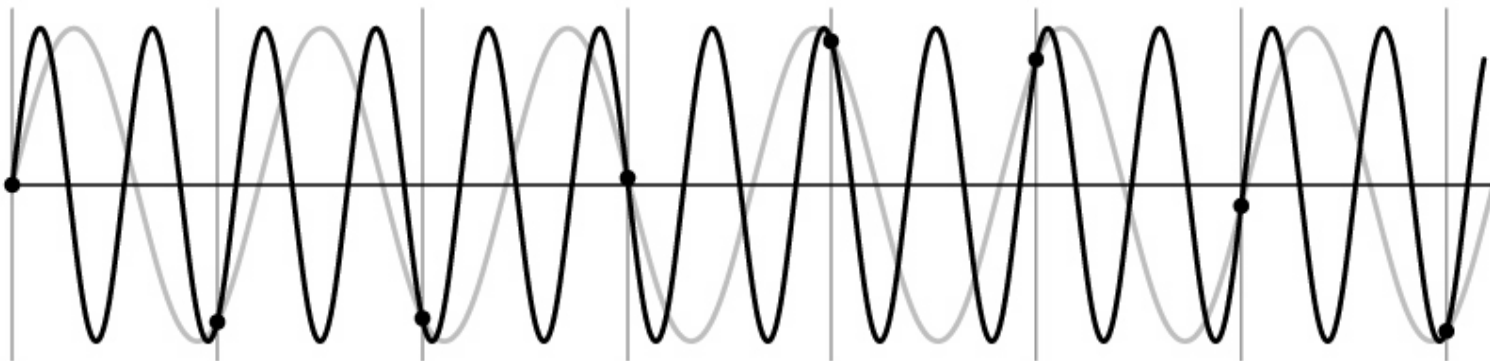
# Aliasing in video

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):



frame 0     frame 1     frame 2     frame 3     frame 4

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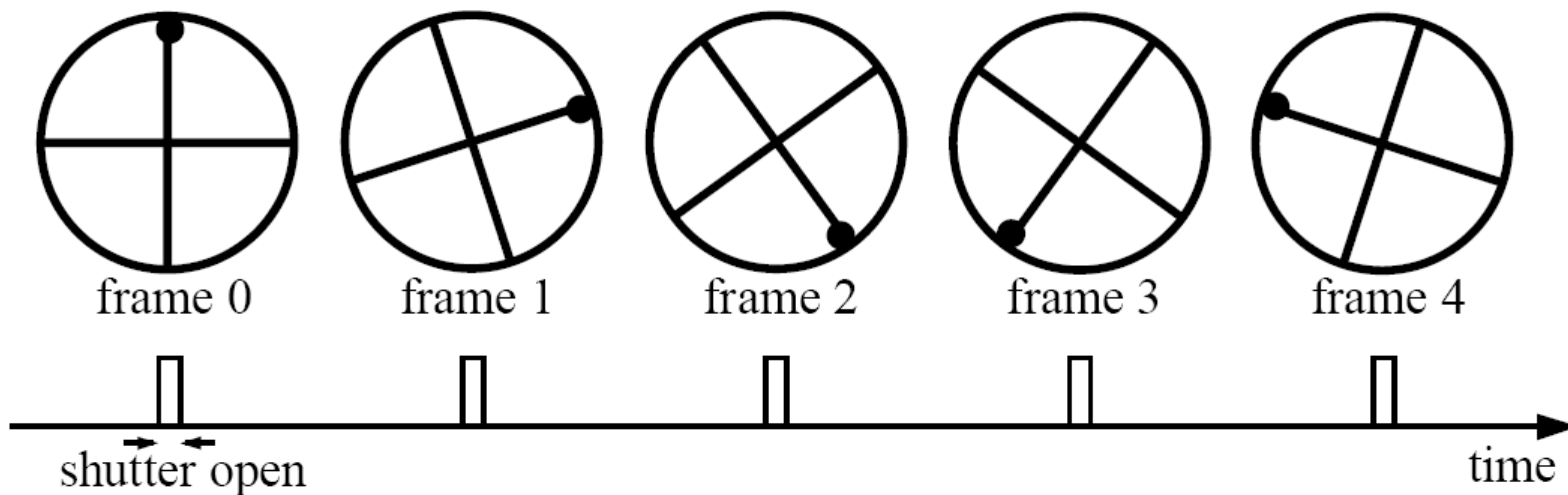
Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

# Aliasing in images
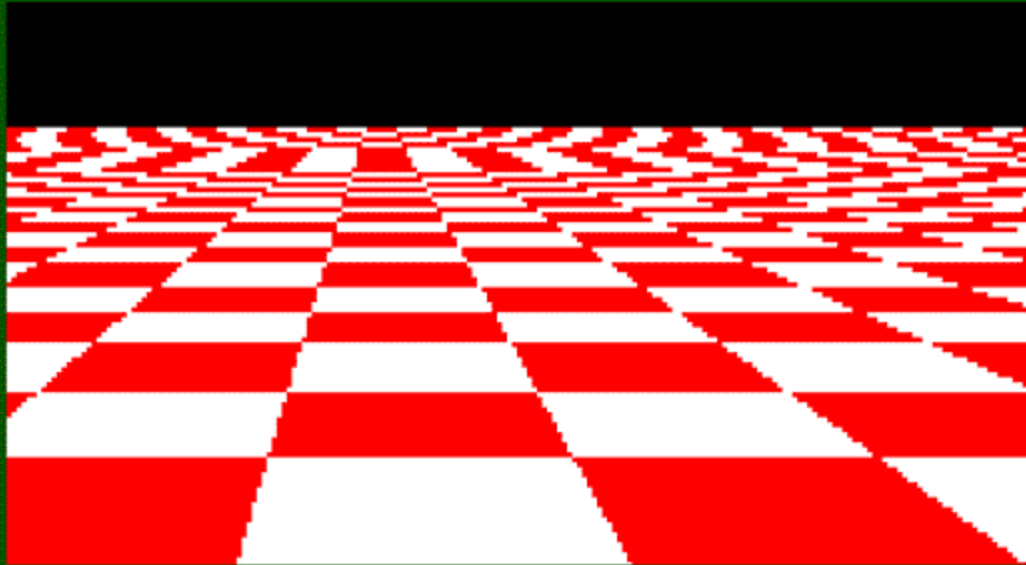


Disintegrating textures

# What's happening?

Input signal:
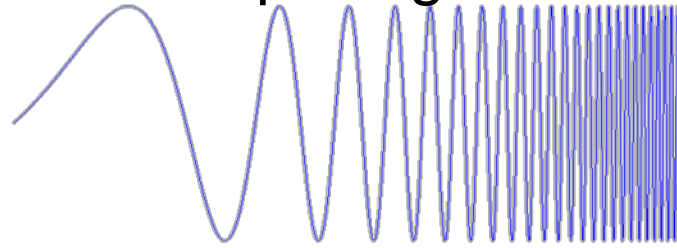
Plot as image:
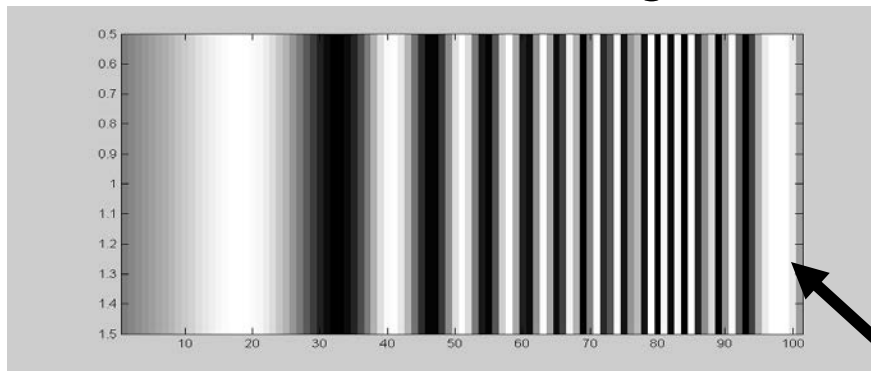
x = 0:.05:5;  imagesc(sin((2.^x).*x))

Alias!
Not enough samples
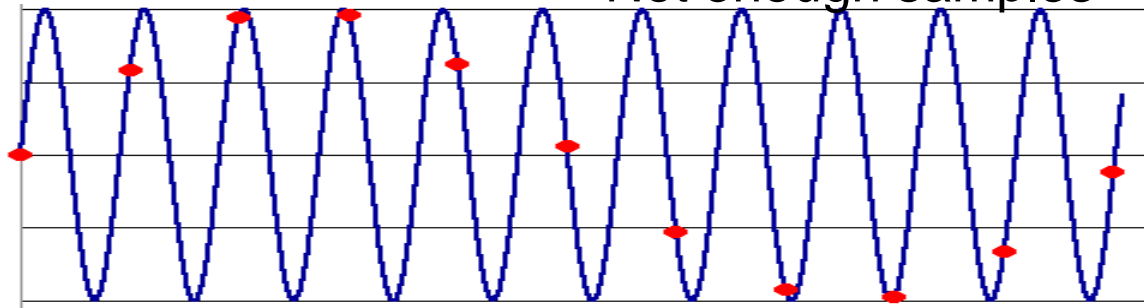
# Antialiasing

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 loose 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)
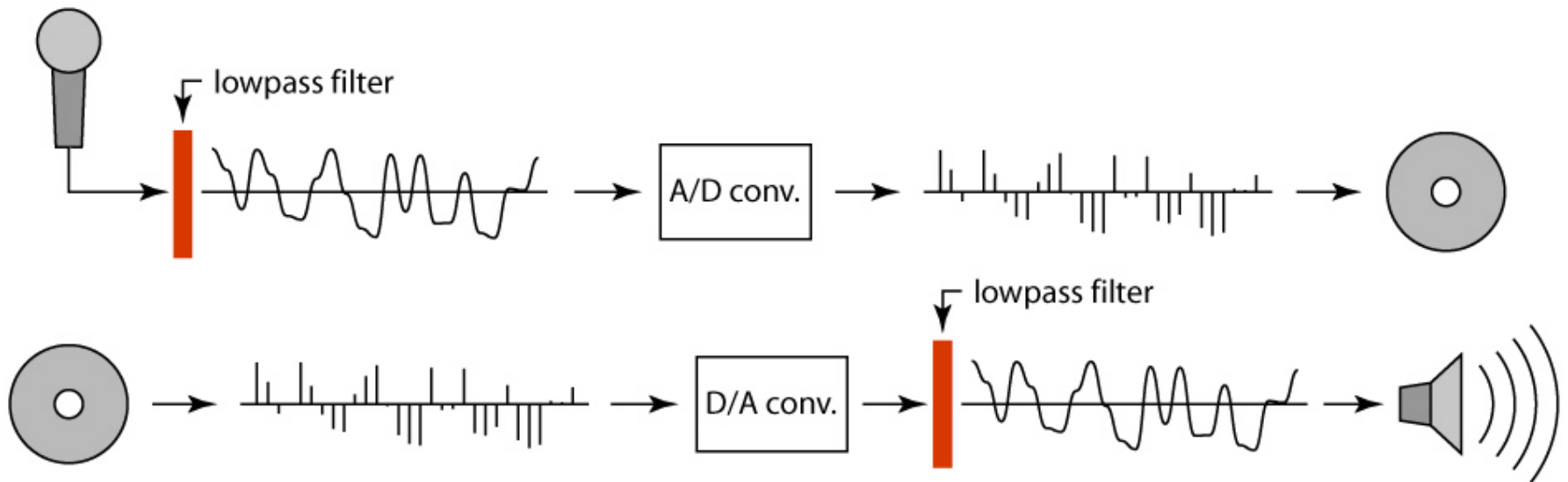
# Linear filtering: a key idea

- Transformations on signals; e.g.:
  - bass/treble controls on stereo
  - blurring/sharpening operations in image editing
  - smoothing/noise reduction in tracking

- Key properties
  - linearity: filter($f + g$) = filter($f$) + 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



original

smoothed

# Moving Average

- Can add weights to our moving average

- *Weights* […, 0, 1, 1, 1, 1, 1, 0, …]  / 5



···001111100···

$\Sigma$
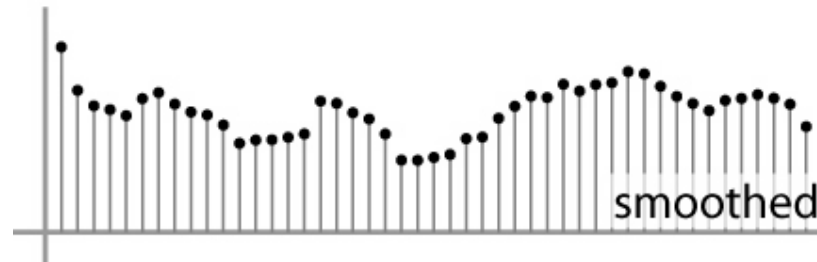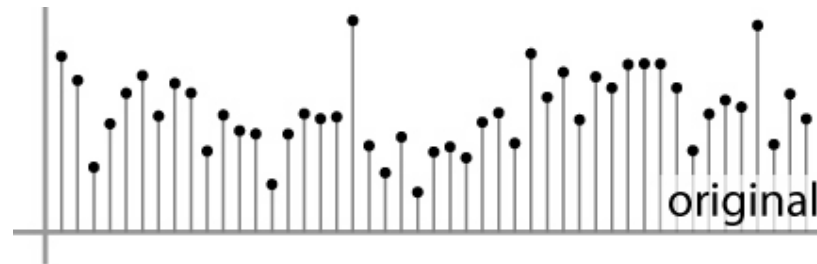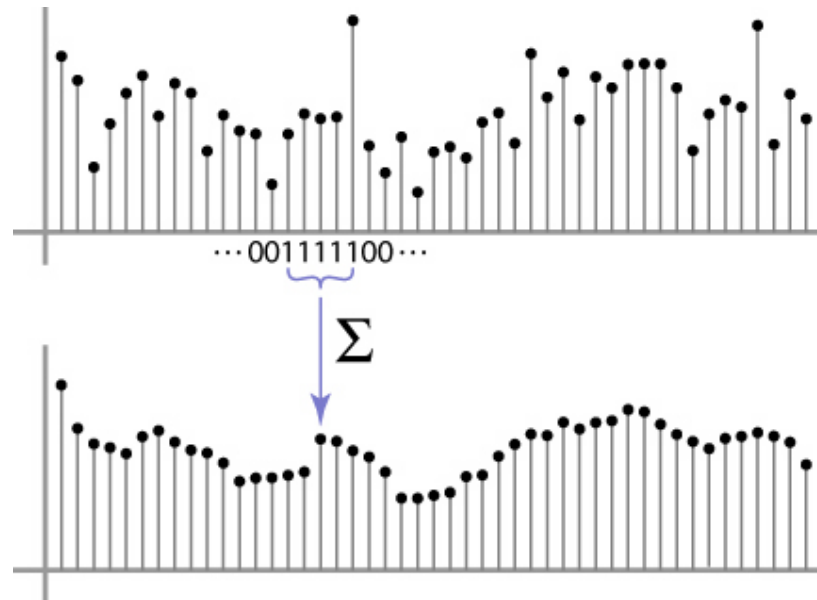
# Cross-correlation

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

# In 2D: box filter

$$h[\cdot,\cdot]$$

$\dfrac{1}{9}$

| 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 |

$$f[\,.,.]$$

| 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 |

$$g[\,.,.]$$

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

# Image filtering

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

| 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$$f[.,.]$$

| 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 |

$$g[.,.]$$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 10 | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

# Image filtering

$$h[\cdot,\cdot] \; \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$f[.,.]$$

| 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 |

$$g[.,.]$$

|   | 0 | 10 | 20 |   |   |   |   |   |   |
|---|---|----|----|---|---|---|---|---|---|
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |
|   |   |    |    |   |   |   |   |   |   |

# Image filtering

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

| 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$$f[.,.]$$

| 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 |

$$g[.,.]$$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 10 | 20 | 30 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

# Image filtering

$$h[\cdot,\cdot] \; \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

## $f[.,.]$

| 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 |

## $g[.,.]$

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
|   | 0 | 10 | 20 | 30 | 30 |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |

# Image filtering

$$h[\cdot,\cdot] \ \frac{1}{9} \ \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$$f[.,.]$$

| 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 |

$$g[.,.]$$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 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

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

| 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

$f[.,.]$

| 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 |

$g[.,.]$

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 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 | |
| | | | | | | | | | |

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

# Box Filter

## What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$h[\cdot\,,\cdot\,]$

$$\frac{1}{9}$$

| 1 | 1 | 1 |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |

# Linear filters: examples



Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} =$$
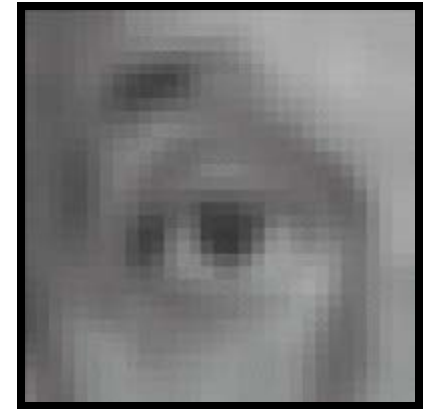
Blur (with a mean filter)

# Practice with linear filters



| 0 | 0 | 0 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 0 |

**?**

Original

# Practice with linear filters



Original

| 0 | 0 | 0 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 0 |



Filtered
(no change)

Source: D. Lowe

# Practice with linear filters



Original

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

?

# Practice with linear filters



Original

| 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 0 |



Shifted left
By 1 pixel

# Other filters
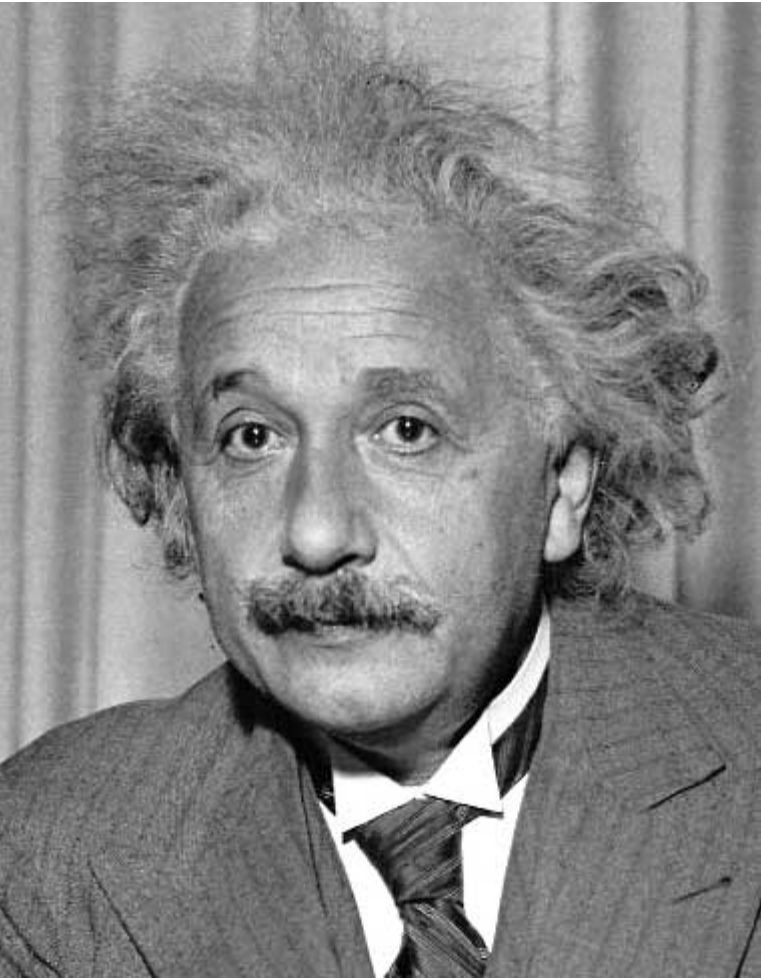


| 1 | 0 | -1 |
|---|---|----|
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel
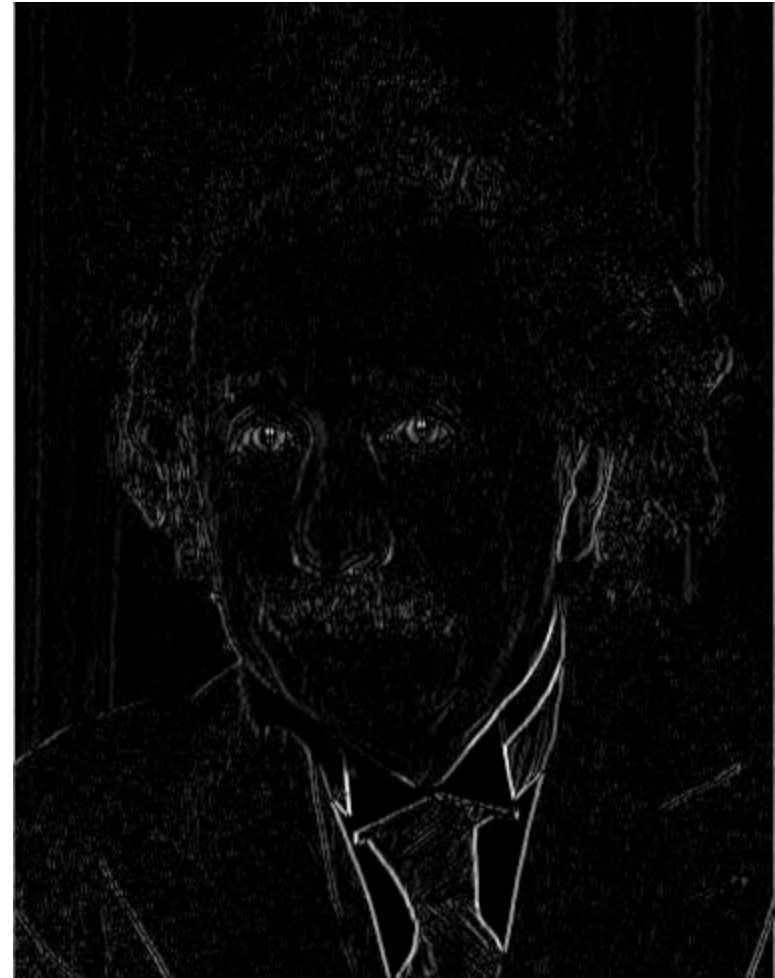


Vertical Edge
(absolute value)

# Other filters
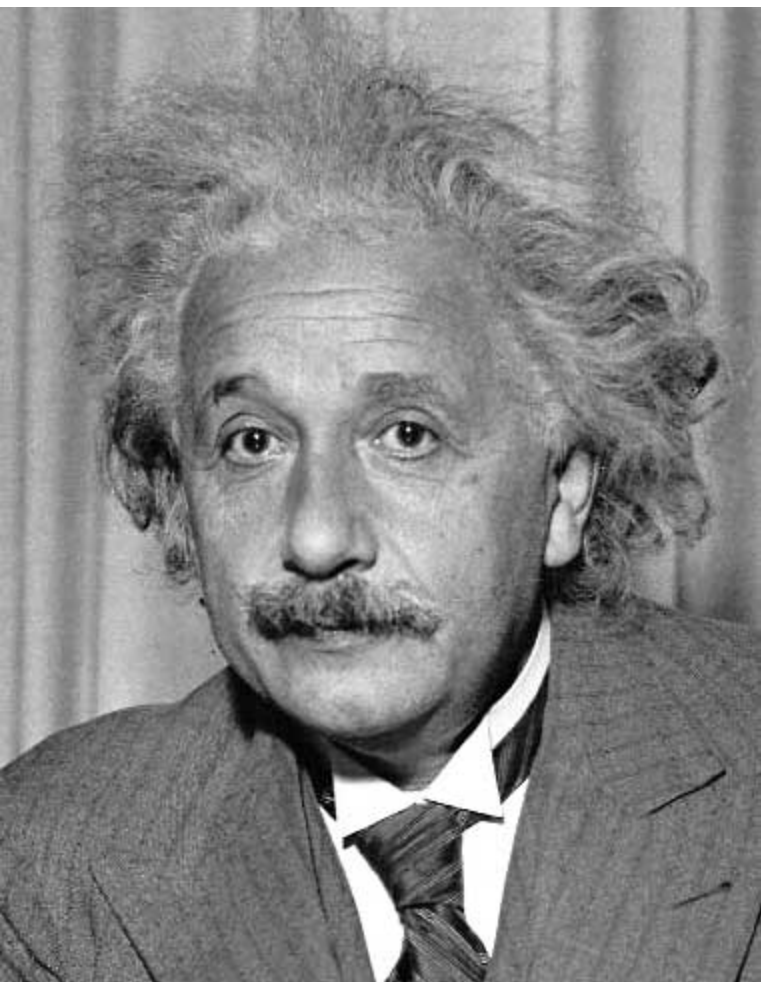


| 1 | 2 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Sobel
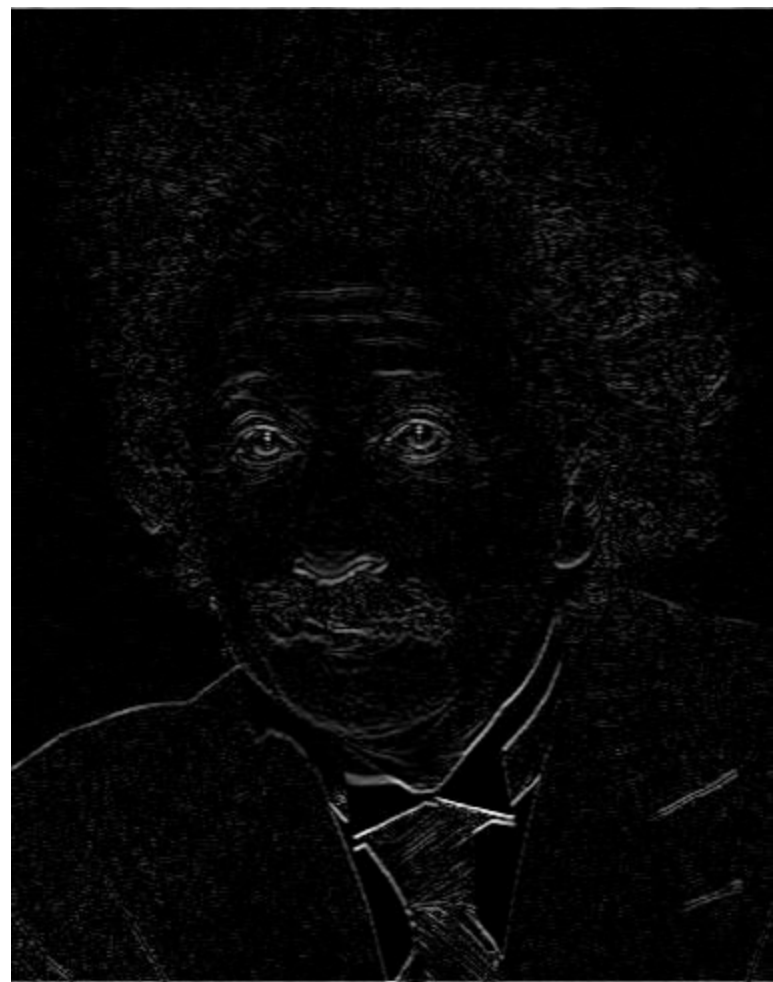


Horizontal Edge
(absolute value)

# Back to the box filter

# Moving Average

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



··· 001111100 ···

$\Sigma$
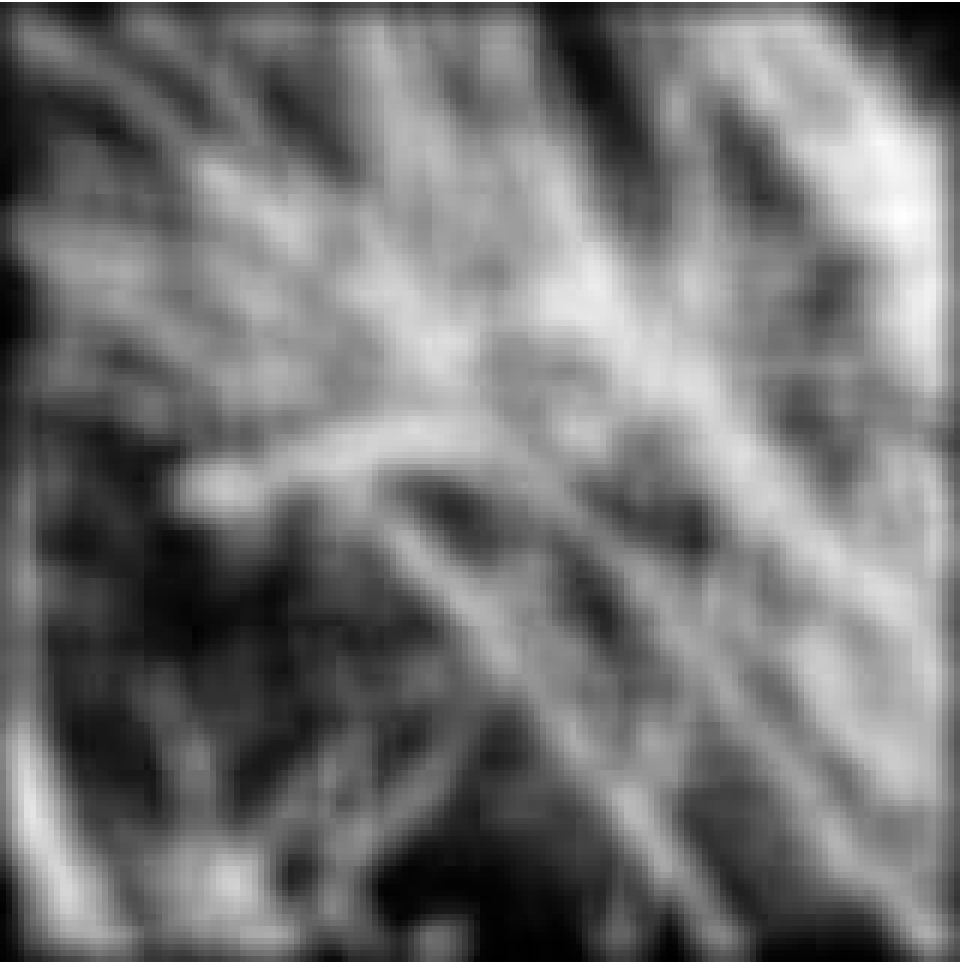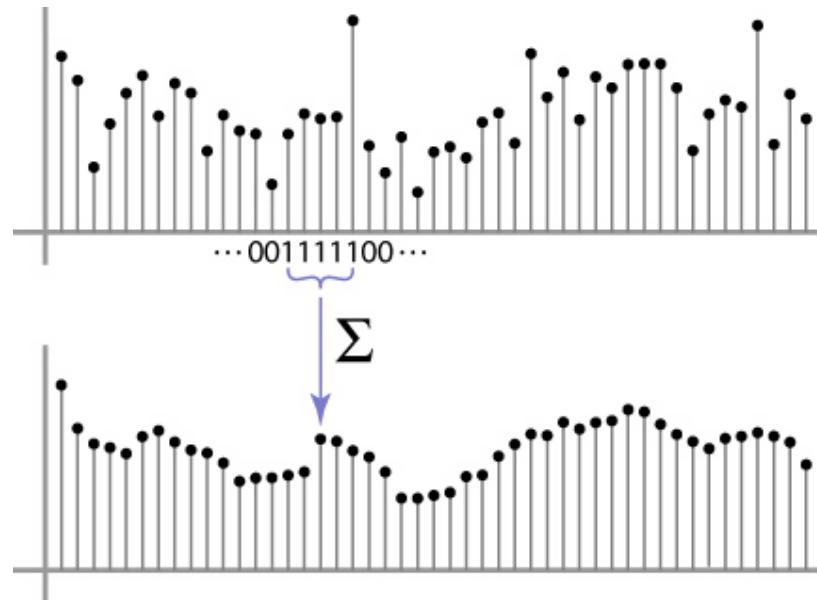
# Weighted Moving Average

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



···001464100···

$\Sigma$
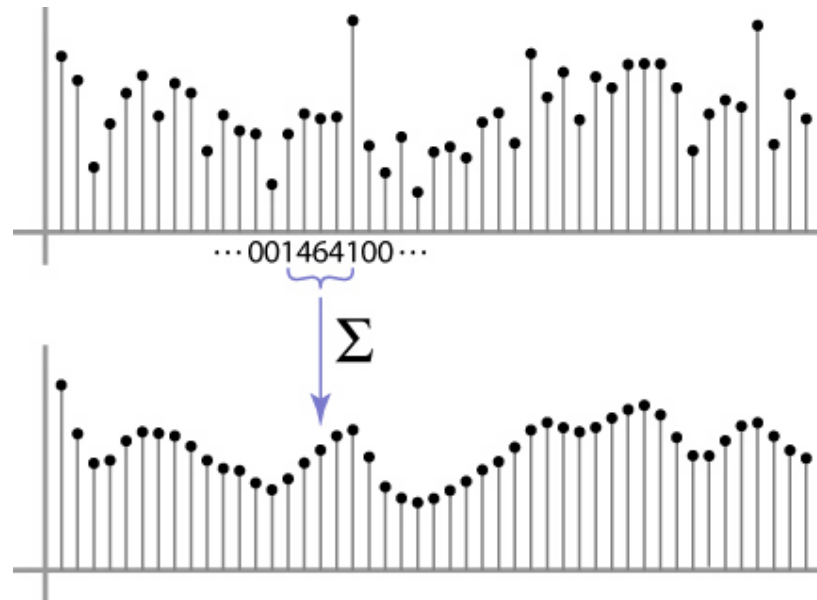
# Moving Average In 2D

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 |

$$F[x, y]$$

$$H[u, v]$$

Slide by Steve Seitz

# 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 |

$$F[x, y]$$

$$\frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$H[u, v]$$

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

This kernel is an approximation of a Gaussian function:

# Mean vs. Gaussian filtering

Slide by Steve Seitz
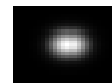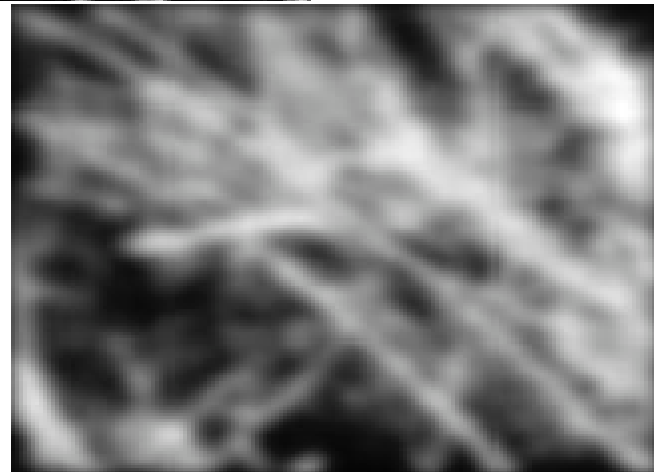
# Important filter: Gaussian

Weight contributions of neighboring pixels by nearness
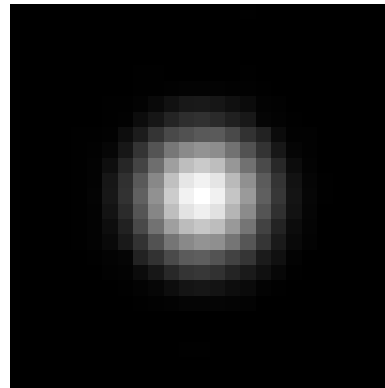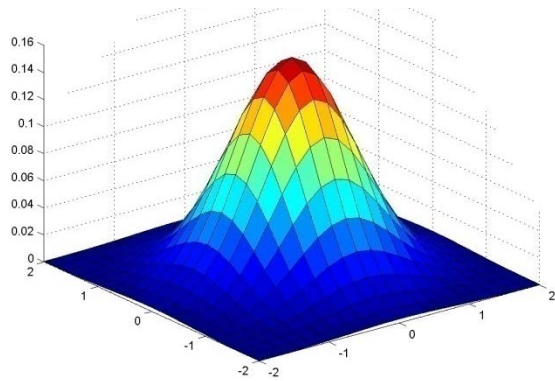


| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.022 | 0.097 | 0.159 | 0.097 | 0.022 |
| 0.013 | 0.059 | 0.097 | 0.059 | 0.013 |
| 0.003 | 0.013 | 0.022 | 0.013 | 0.003 |

5 x 5, σ = 1

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

# Gaussian Kernel

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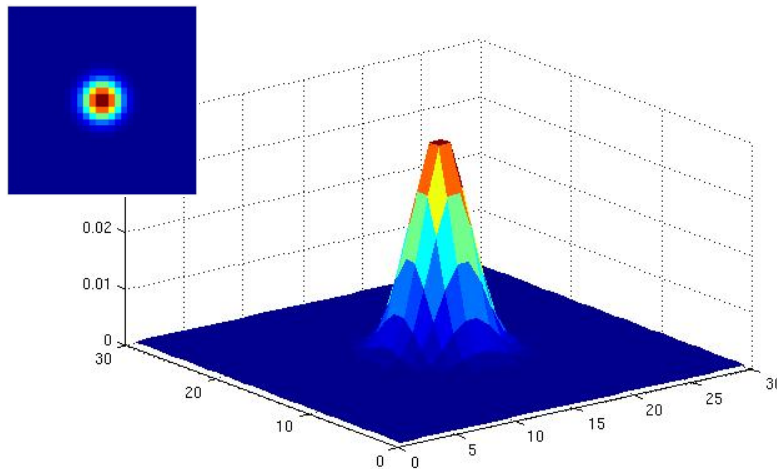


σ = 2 with 30 x 30 kernel
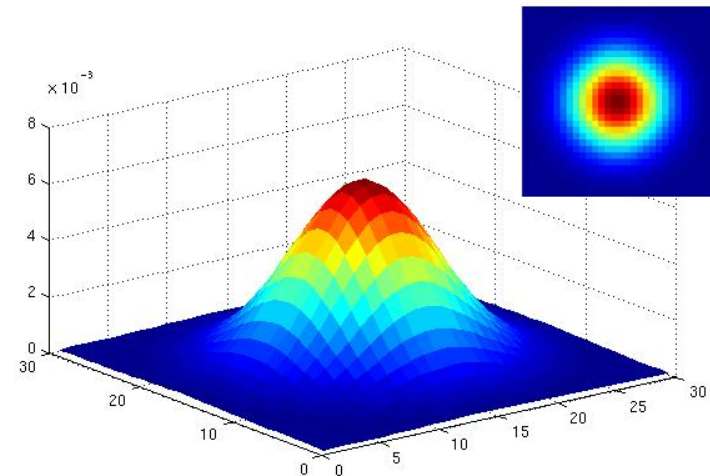
σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

Source: K. Grauman
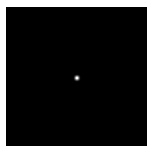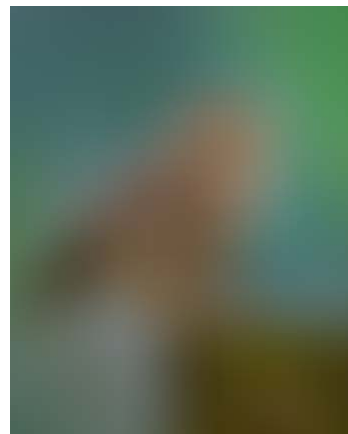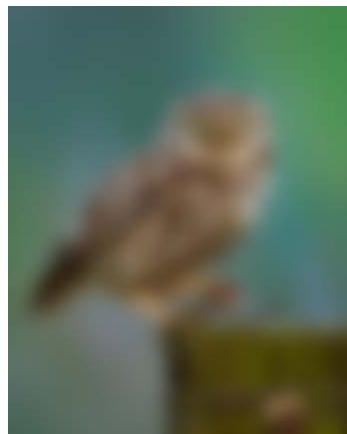
# Gaussian filters
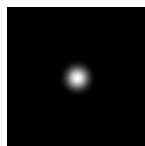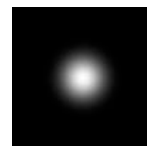


$\sigma$ = 1 pixel    $\sigma$ = 5 pixels    $\sigma$ = 10 pixels    $\sigma$ = 30 pixels

# Choosing kernel width

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



σ = 5 with 10x10 kernel

σ = 5 with 30x30 kernel

Source: K. Grauman
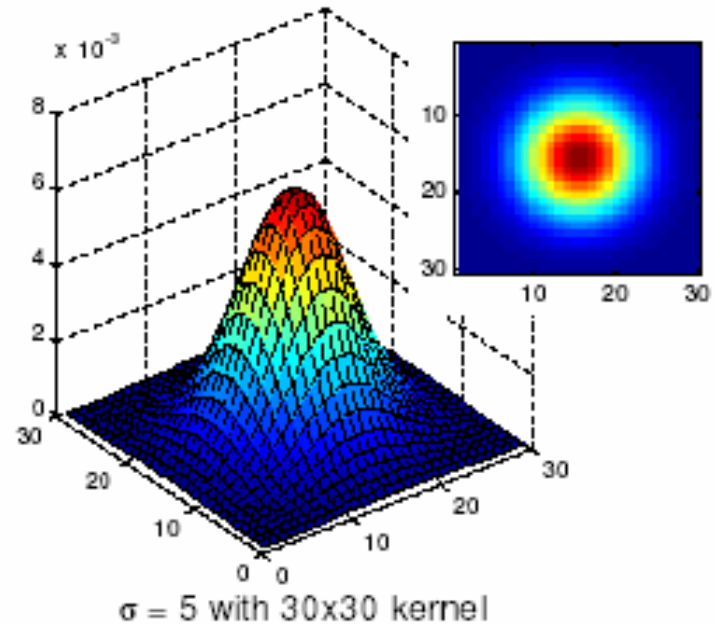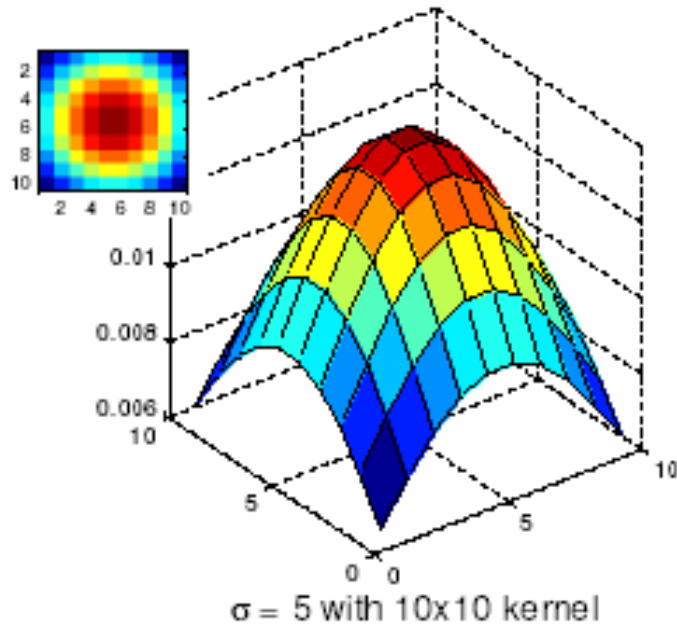
# Practical matters

## How big should the filter be?

Values at edges should be near zero

Rule of thumb for Gaussian: set filter half-width to about 3 $\sigma$



Effect of $\sigma$

Side by Derek Hoiem
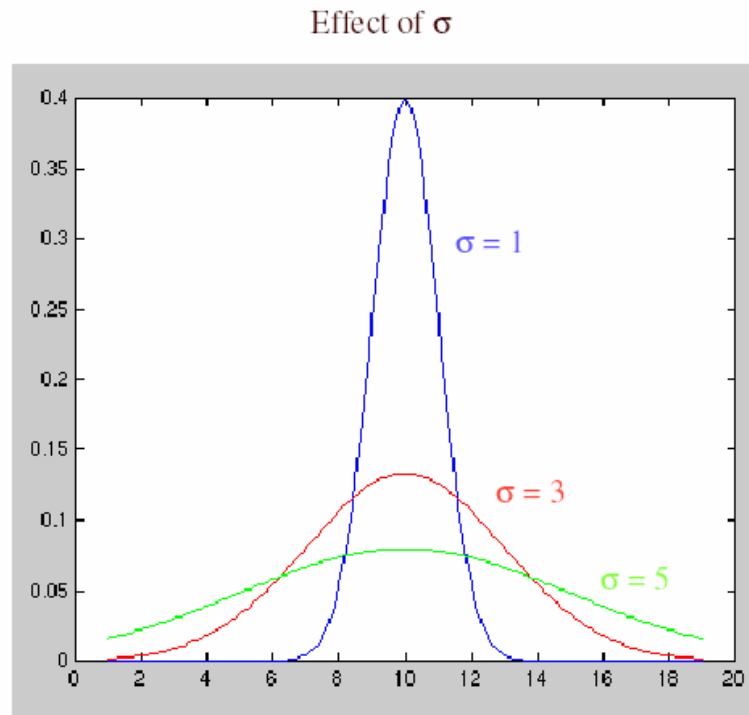
# Cross-correlation vs. Convolution

**cross-correlation**: $G = H \otimes F$

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

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

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

It is written:

$$G = H \star F$$

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

# Convolution
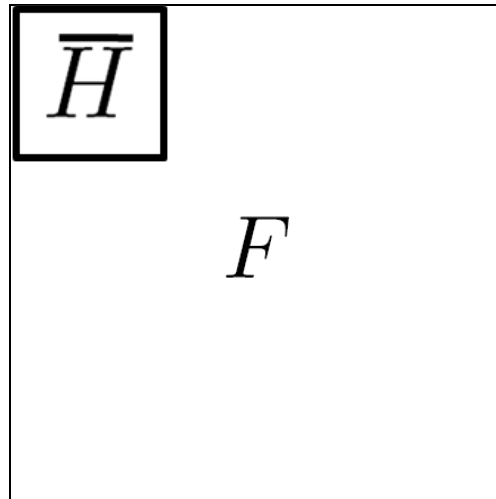
# Convolution is nice!

- Notation: $b = c \star a$

- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - identity: unit impulse $e$ = […, 0, 0, 1, 0, 0, …]
    $$a \star e = a$$

- Conceptually no distinction between filter and signal

- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

# Gaussian and convolution

- Removes "high-frequency" components from the image (low-pass filter)

- Convolution with self is another Gaussian
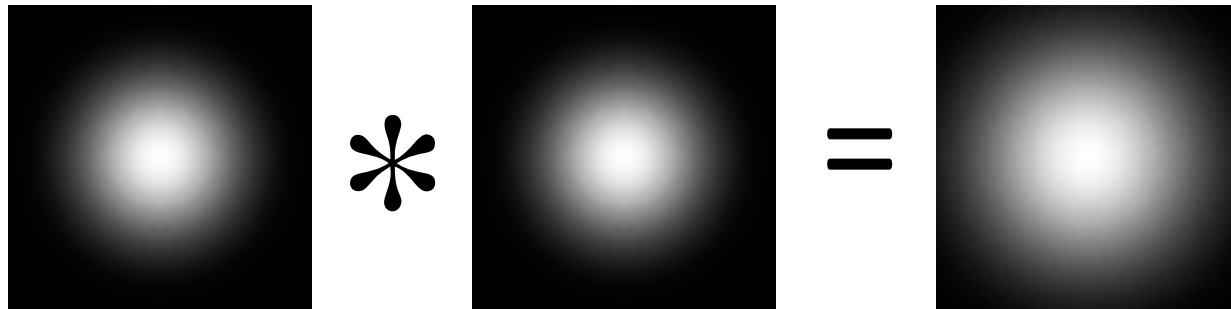


  – Convolving twice with Gaussian kernel of width $\sigma$ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

# Practical matters

What is the size of the output?

MATLAB: filter2(g, f, *shape*) or conv2(g,f,shape)

- *shape* = 'full': output size is sum of sizes of f and g
- *shape* = 'same': output size is same as f
- *shape* = 'valid': output size is difference of sizes of f and g
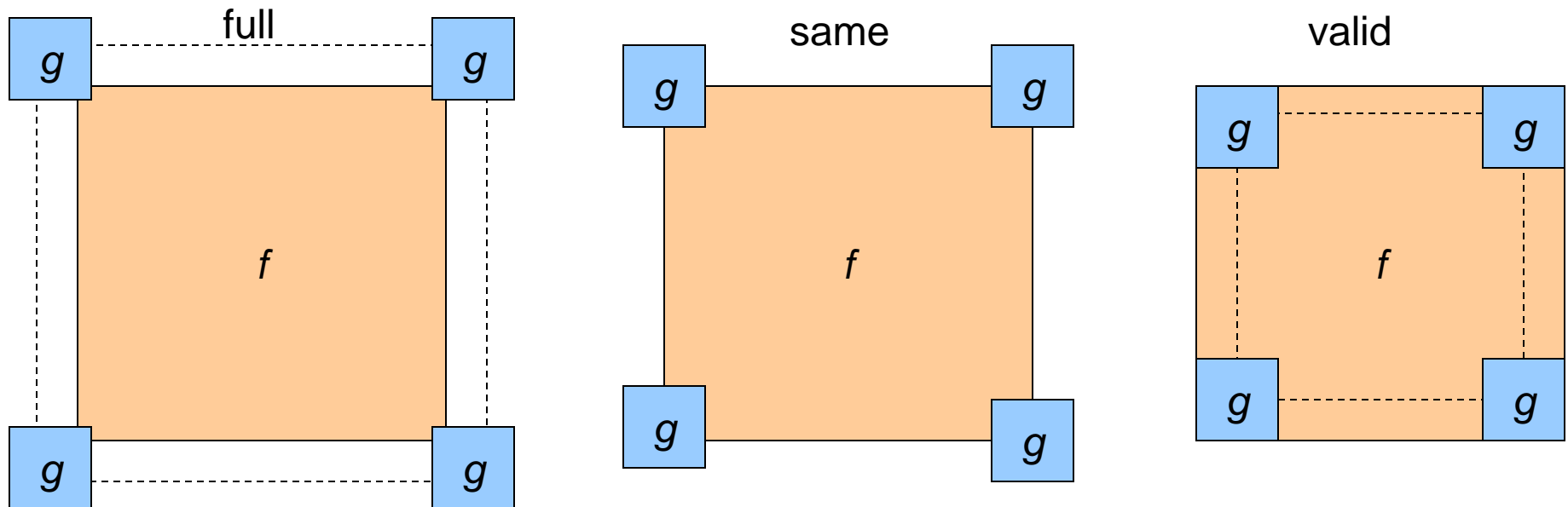
# Practical matters

## What about near the edge?

- the filter window falls off the edge of the image

- need to extrapolate

- methods:
  - clip filter (black)
  - wrap around
  - copy edge
  - reflect across edge



Source: S. Marschner

# Practical matters

- methods (MATLAB):
  - clip filter (black):      imfilter(f, g, 0)
  - wrap around:  imfilter(f, g, 'circular')
  - copy edge:          imfilter(f, g, 'replicate')
  - reflect across edge:    imfilter(f, g, 'symmetric')

Source: S. Marschner

# 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



1/8

1/4
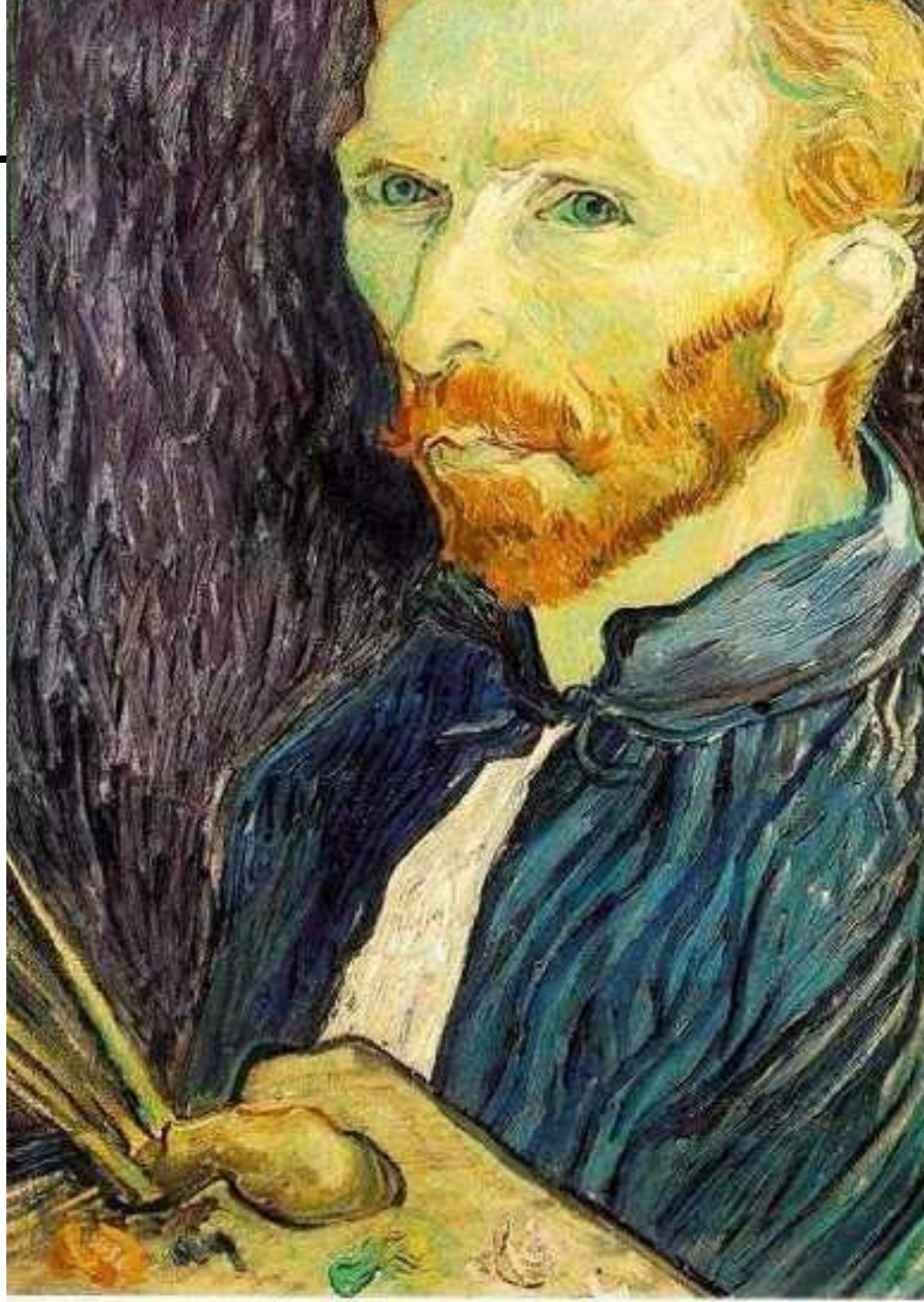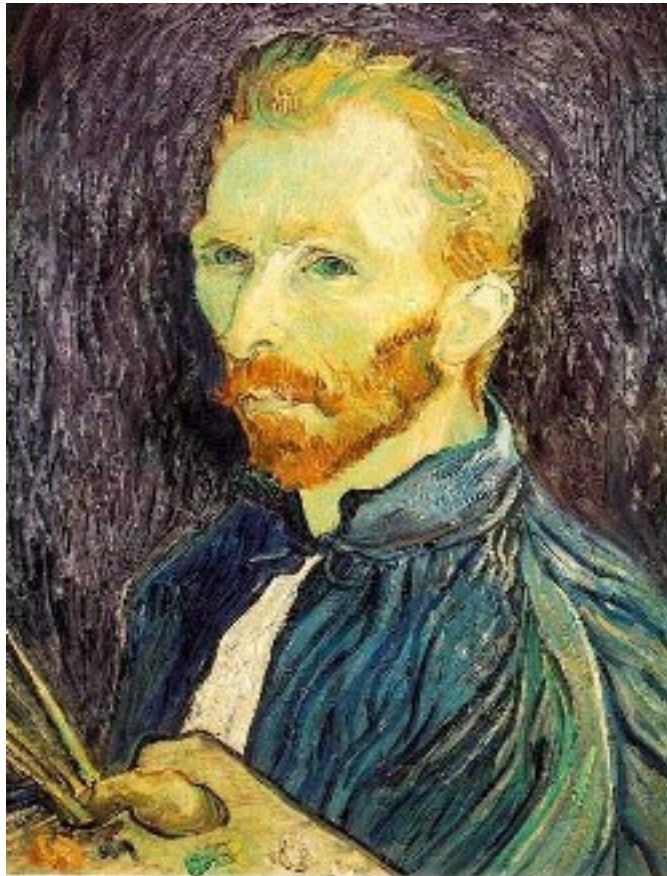
Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

# Image sub-sampling
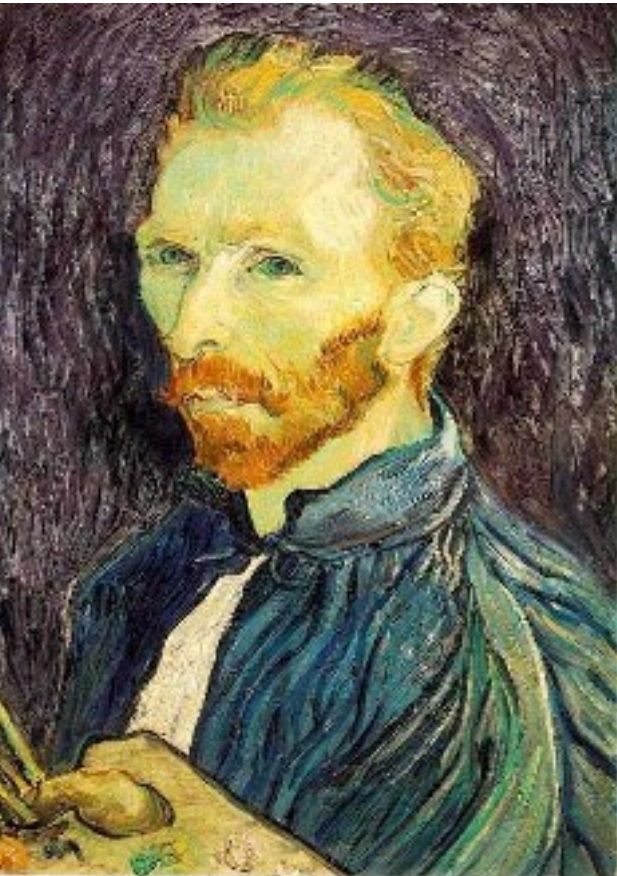


1/2            1/4  (2x zoom)            1/8  (4x zoom)

Aliasing!  What do we do?

# Sampling an image
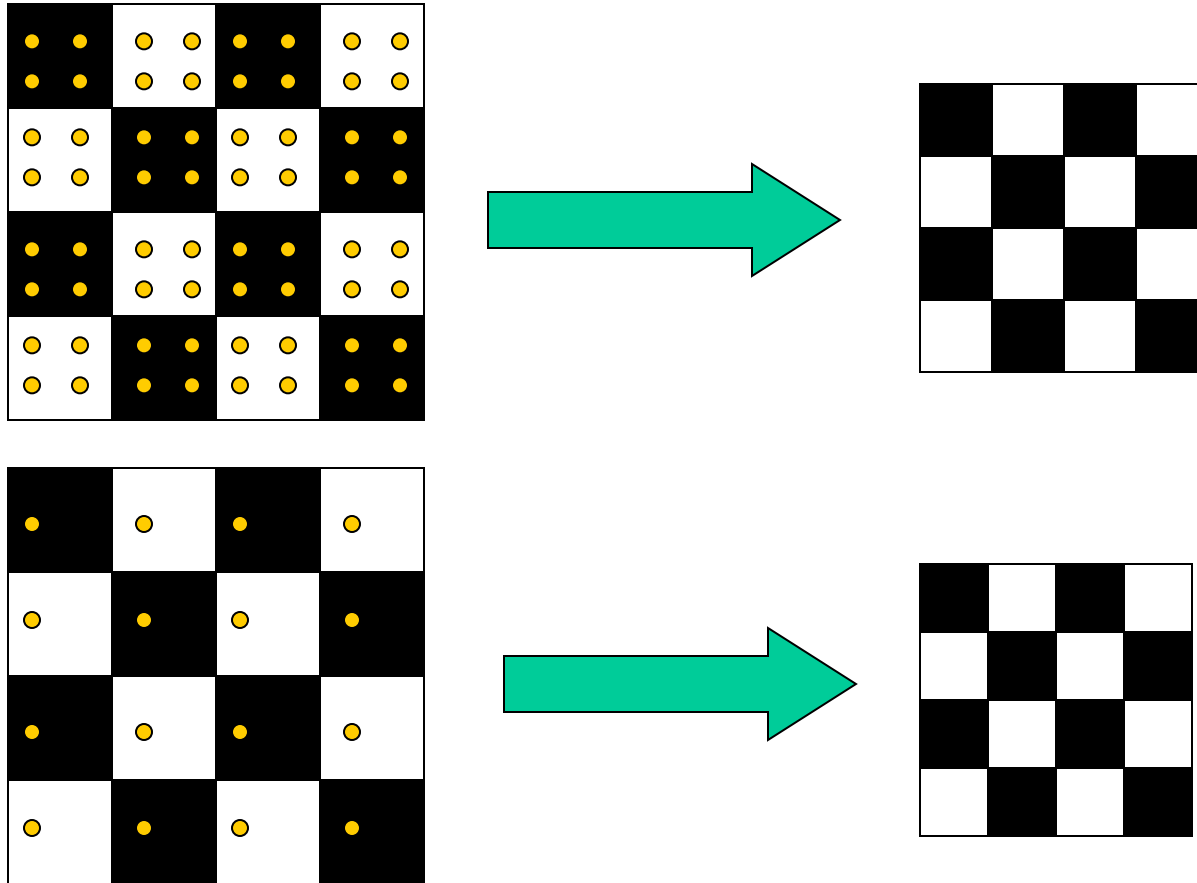


Examples of GOOD sampling

# Undersampling
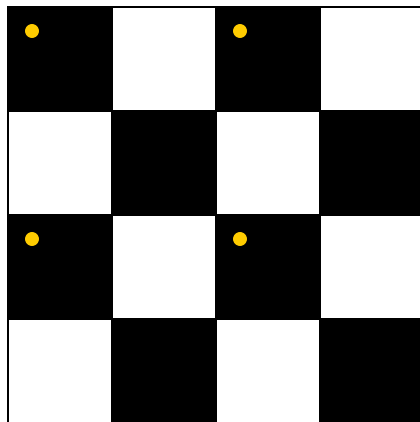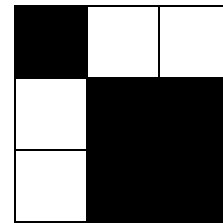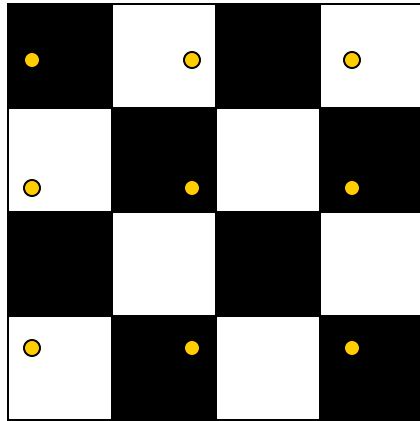


Examples of BAD sampling -> Aliasing

# Gaussian (lowpass) pre-filtering
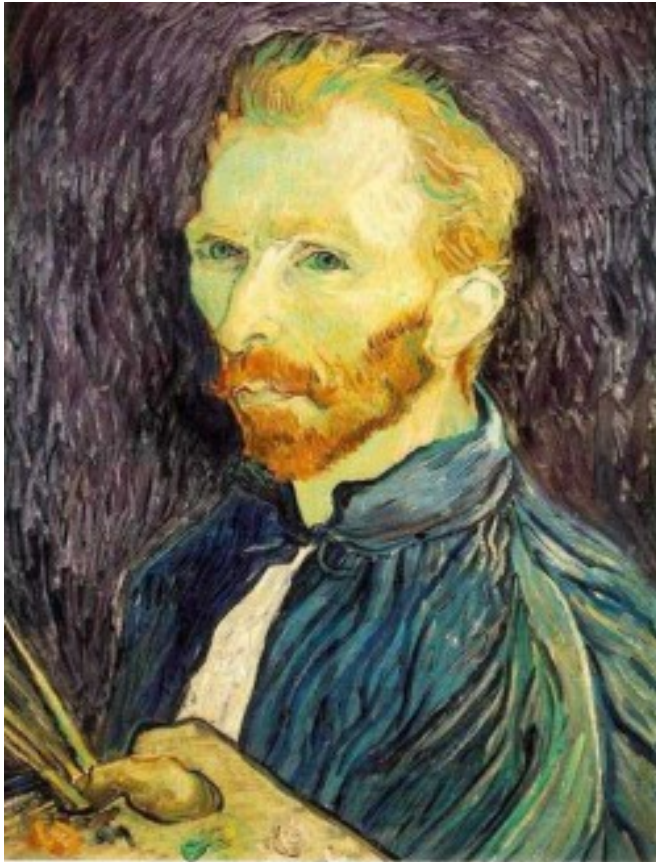


Gaussian 1/2

G 1/4

G 1/8

# Solution:  filter the image, *then* subsample

- Filter size should double for each ½ size reduction.  Why?

# Subsampling with Gaussian pre-filtering
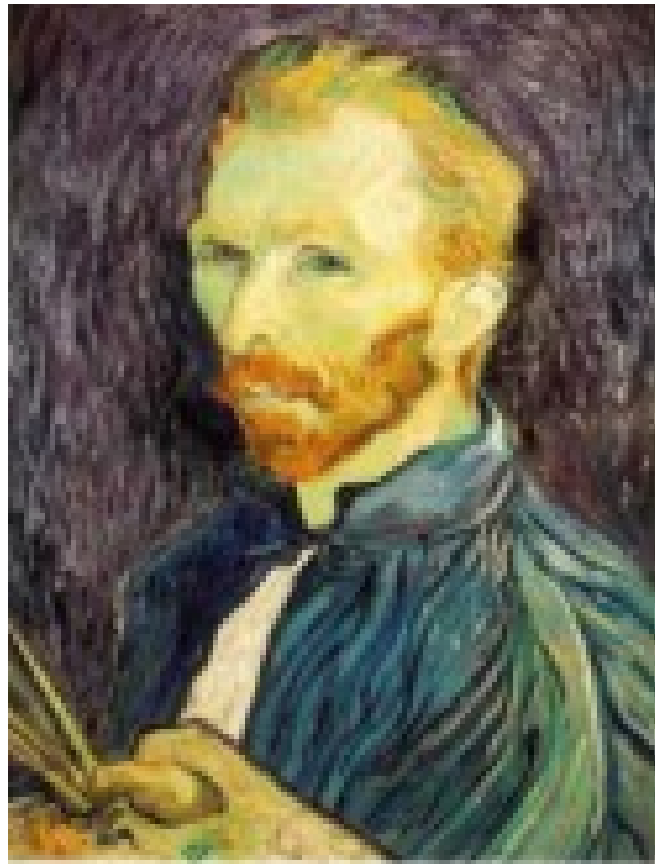


Gaussian 1/2          G 1/4          G 1/8

# Compare with...
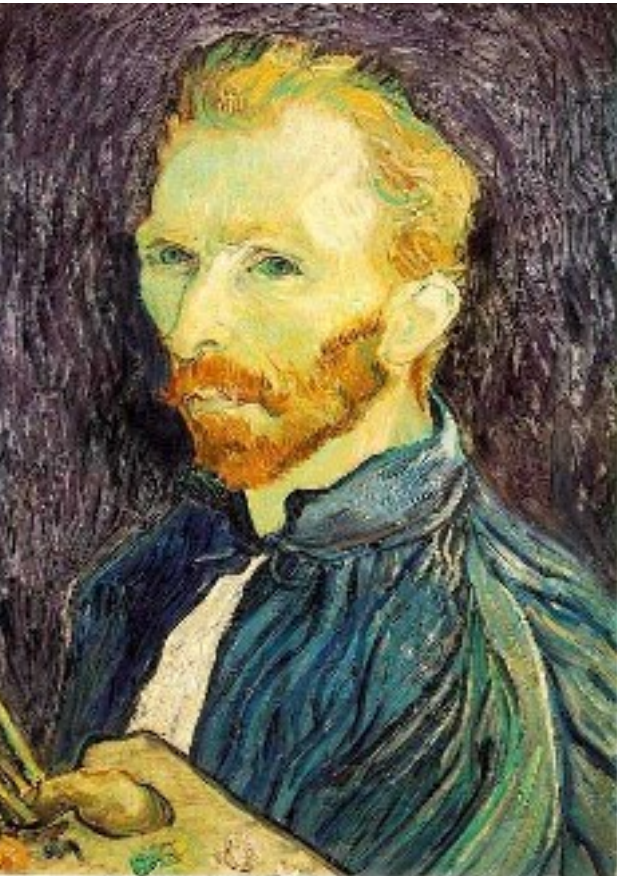


1/2          1/4  (2x zoom)          1/8  (4x zoom)

# Gaussian (lowpass) pre-filtering
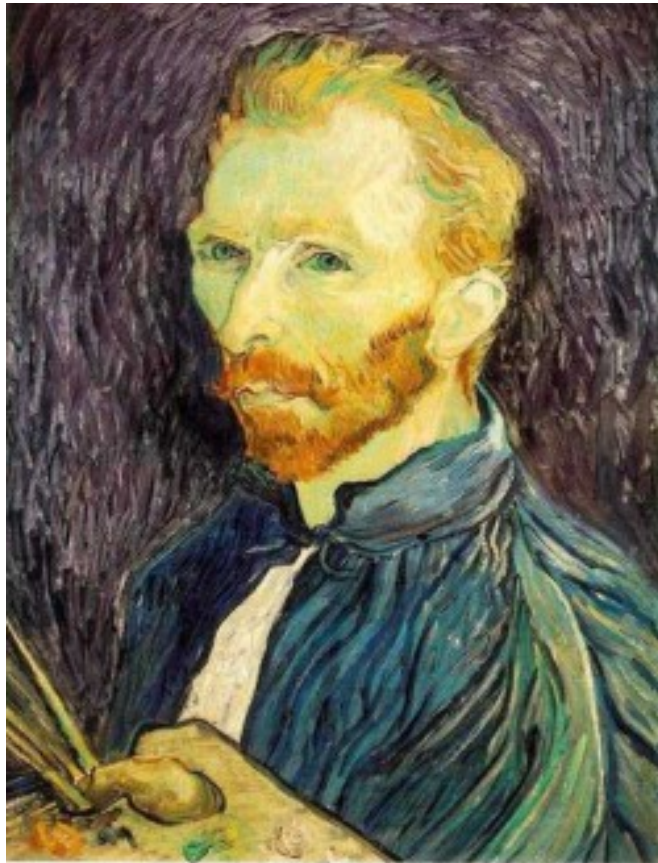


Gaussian 1/2

G 1/4

G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

# Image Pyramids



Idea: Represent NxN image as a "pyramid" of $1 \times 1$, $2 \times 2$, $4 \times 4$, ..., $2^k \times 2^k$ images (assuming $N = 2^k$)

level k (= 1 pixel)

level k-1

level k-2

...

level 0 (= original image)
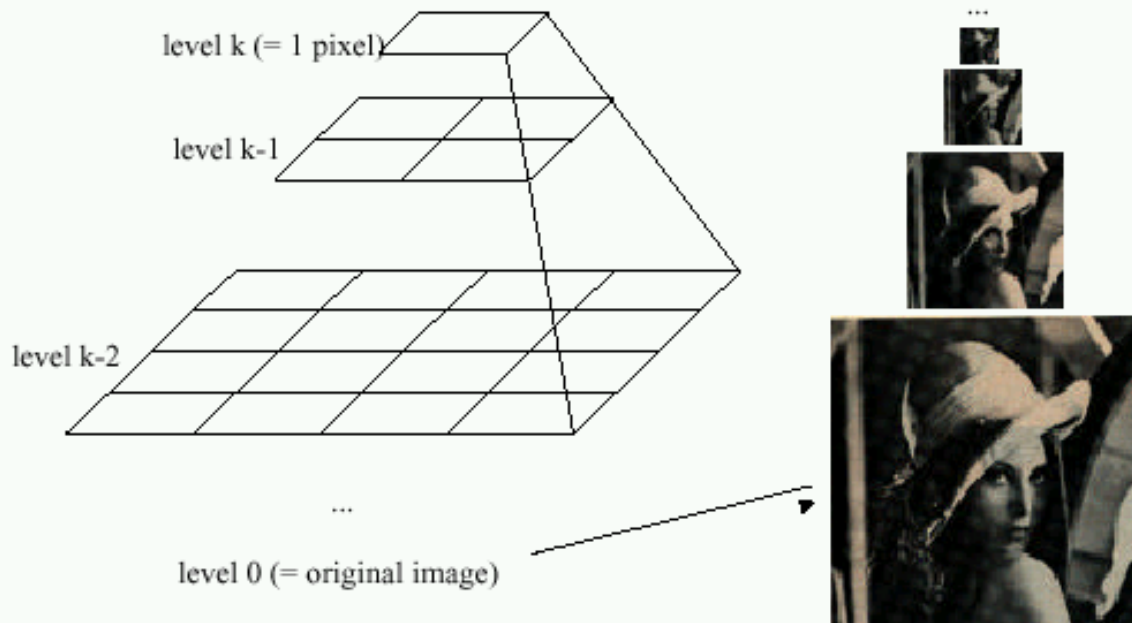
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]
- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

512    256    128    64    32    16    8

A bar in the big images is a hair on the zebra's nose; in smaller images, a stripe; in the smallest, the animal's nose

Figure from David Forsyth

# Gaussian pyramid construction



filter mask
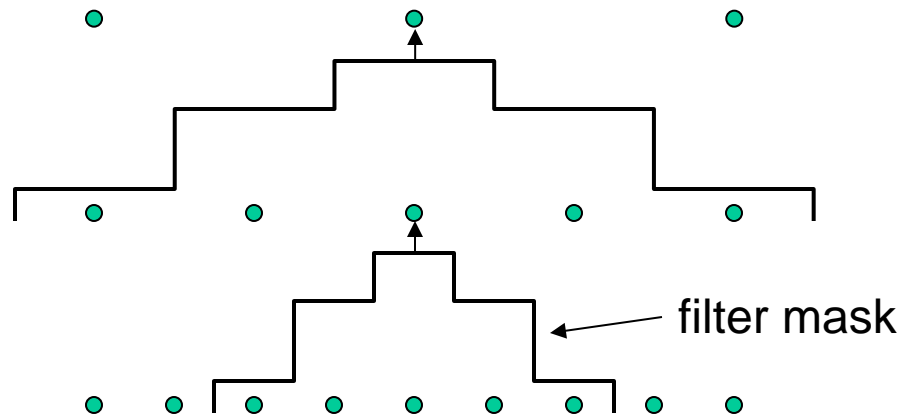
Repeat
- Filter
- Subsample

Until minimum resolution reached
- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

# What are they good for?

## Improve Search

- Search over translations
  - Classic coarse-to-fine strategy
- Search over scale
  - Template matching
  - E.g. find a face at different scales

# Template matching

Goal: find  in image

Main challenge: What is a good similarity or distance measure between two patches?

- Correlation
- Zero-mean correlation
- Sum Square Difference
- Normalized Cross Correlation
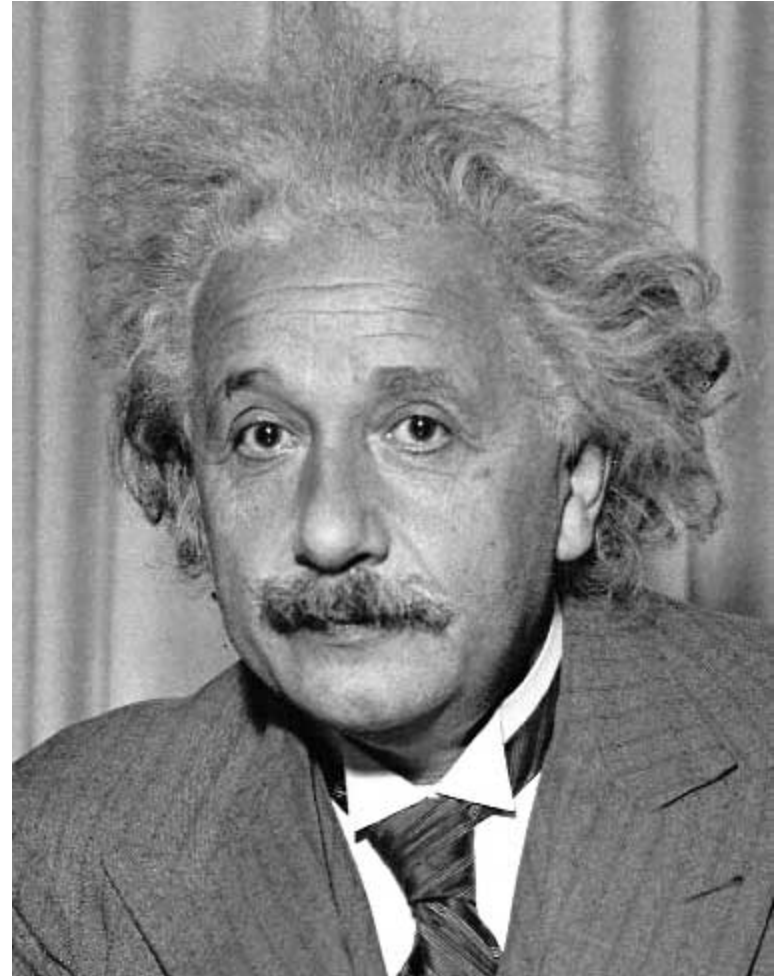
# Matching with filters

Goal: find 👁 in image

Method 0: filter the image with eye patch

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

f = image
g = filter
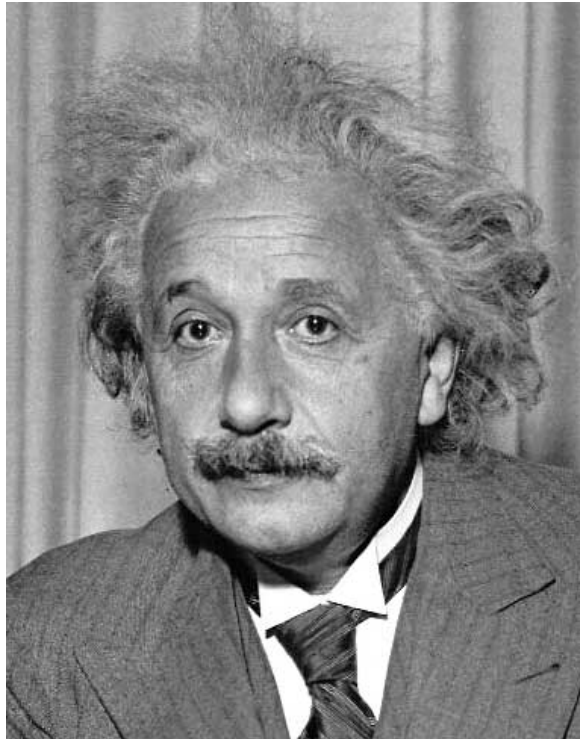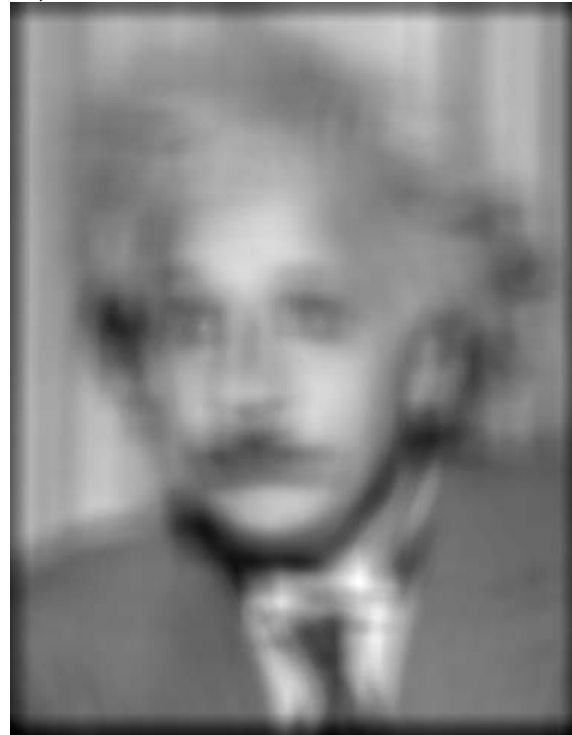
What went wrong?

Input

Filtered Image

# Matching with filters

Goal: find 👁 in image

Method 1: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (f[k,l] - \bar{f}) (g[m+k, n+l])$$
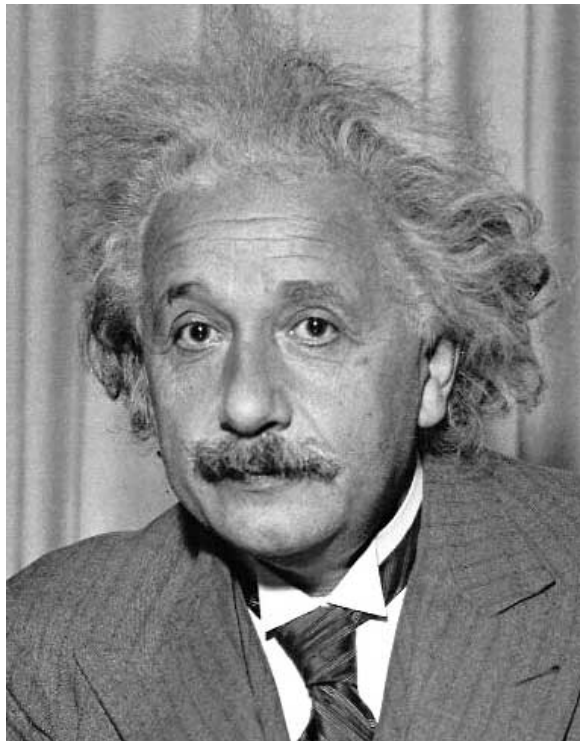
← mean of f
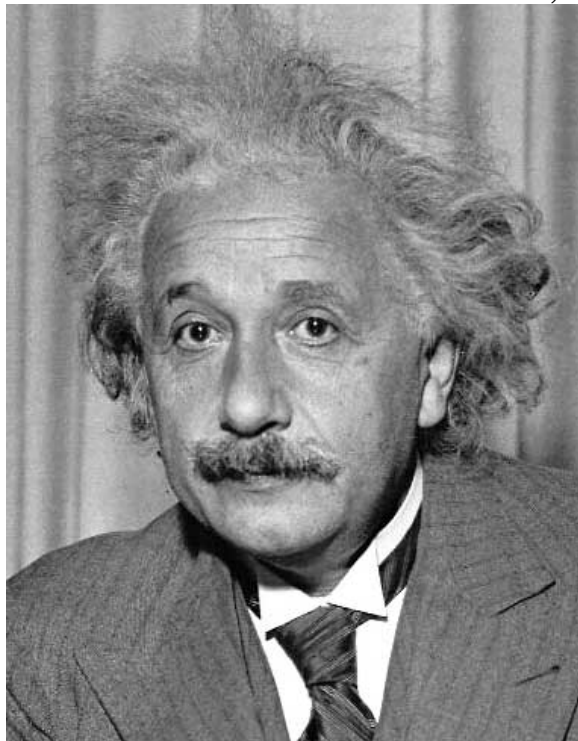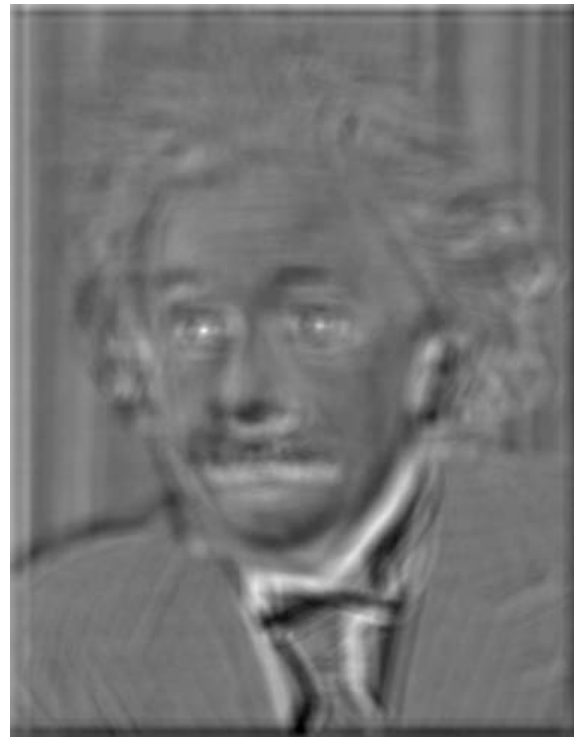


Input



Filtered Image (scaled)



**True detections**

**False detections**

Thresholded Image

# Matching with filters

Goal: find  in image

Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



**True detections**

| Input | 1- sqrt(SSD) | Thresholded Image |

# Matching with filters

Can SSD be implemented with linear filters?

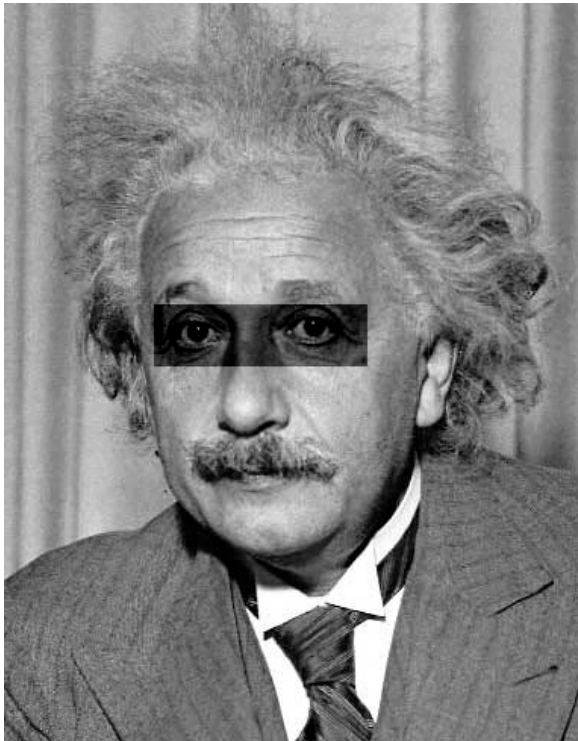$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

# Matching with filters

Goal: find 👁 in image

Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input



1- sqrt(SSD)

Side by Derek Hoiem

# Matching with filters

Goal: find  in image

Method 3: Normalized cross-correlation

mean template

mean image patch

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\overline{g})(f[m+k,n+l]-\overline{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\overline{g})^2 \sum_{k,l}(f[m+k,n+l]-\overline{f}_{m,n})^2\right)^{0.5}}$$
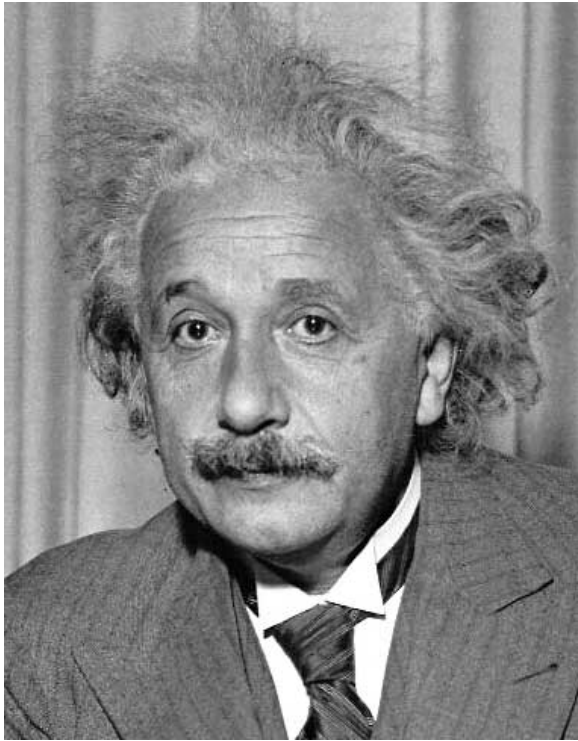
# Matching with filters
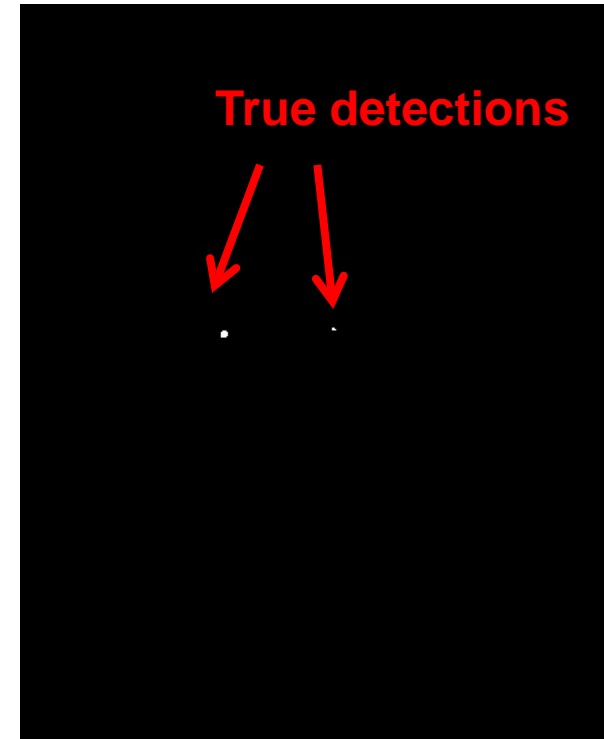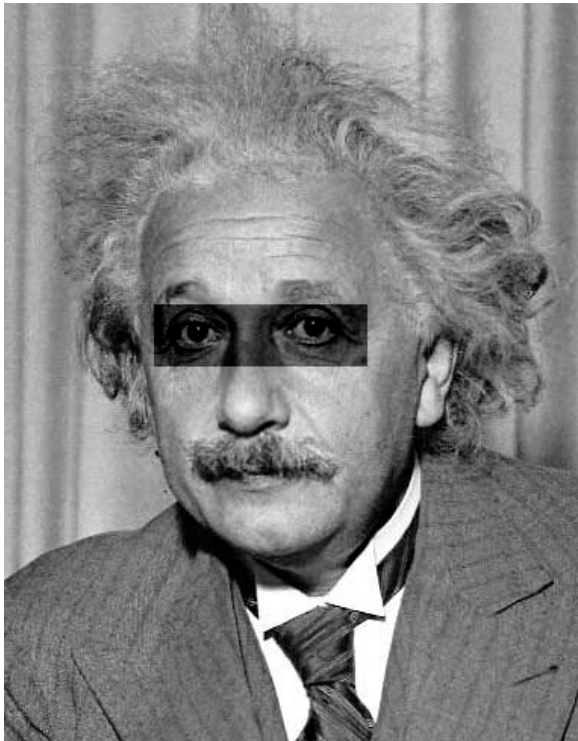
Goal: find  in image

Method 3: Normalized cross-correlation



Input

Normalized X-Correlation

Thresholded Image

**True detections**

# Matching with filters
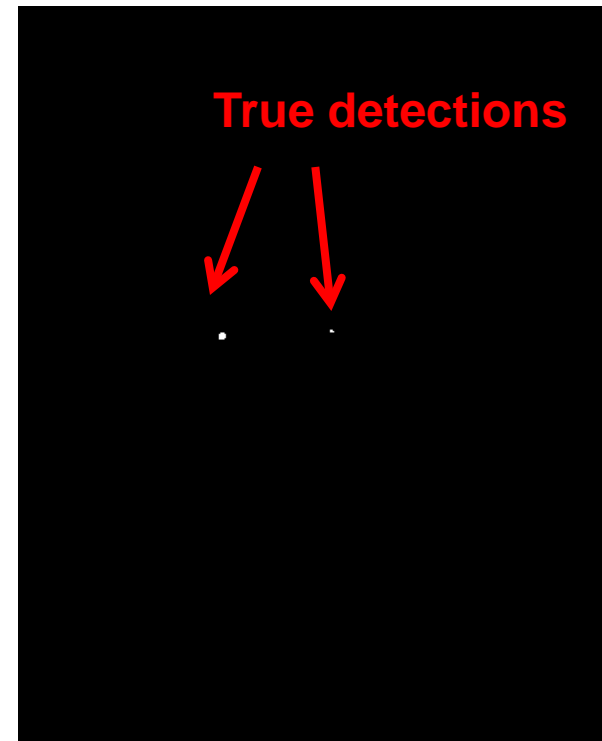
Goal: find  in image

Method 3: Normalized cross-correlation



Input



Normalized X-Correlation



True detections

Thresholded Image

# Q: What is the best method to use?

A: Depends

Zero-mean filter: fastest but not a great matcher

SSD: next fastest, sensitive to overall intensity

Normalized cross-correlation: slowest, invariant to local average intensity and contrast