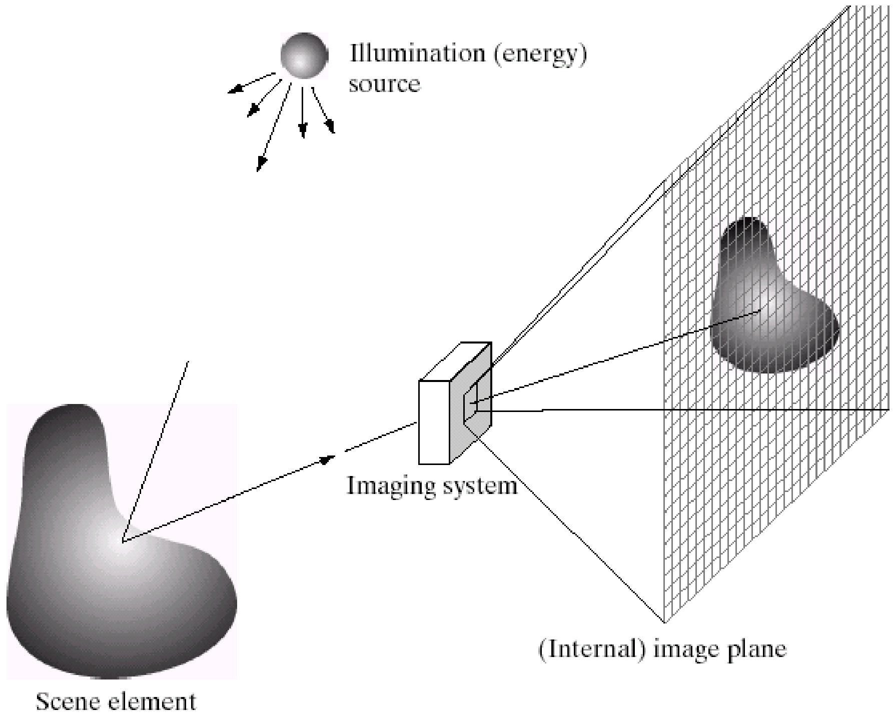
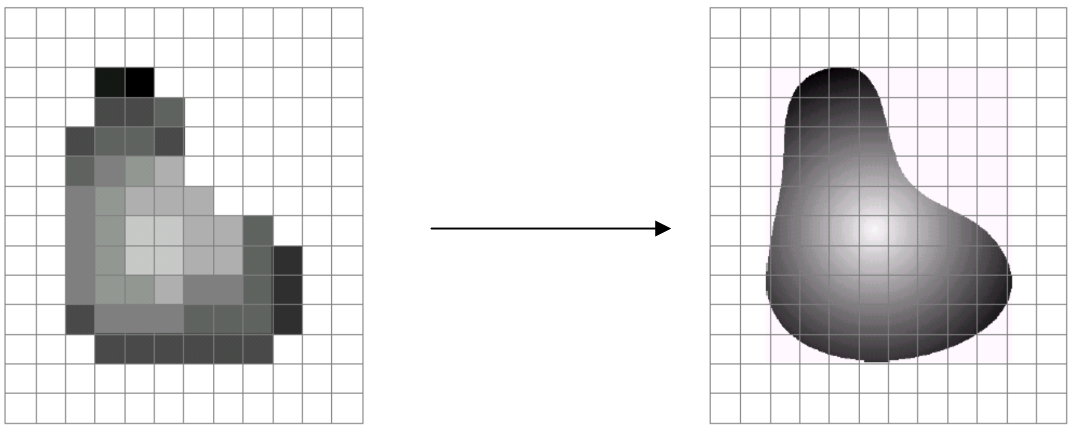
Sampling and Reconstruction

15-463: Computational Photography
Alexei Efros, CMU, Fall 2011

Many slides from
Steve Marschner

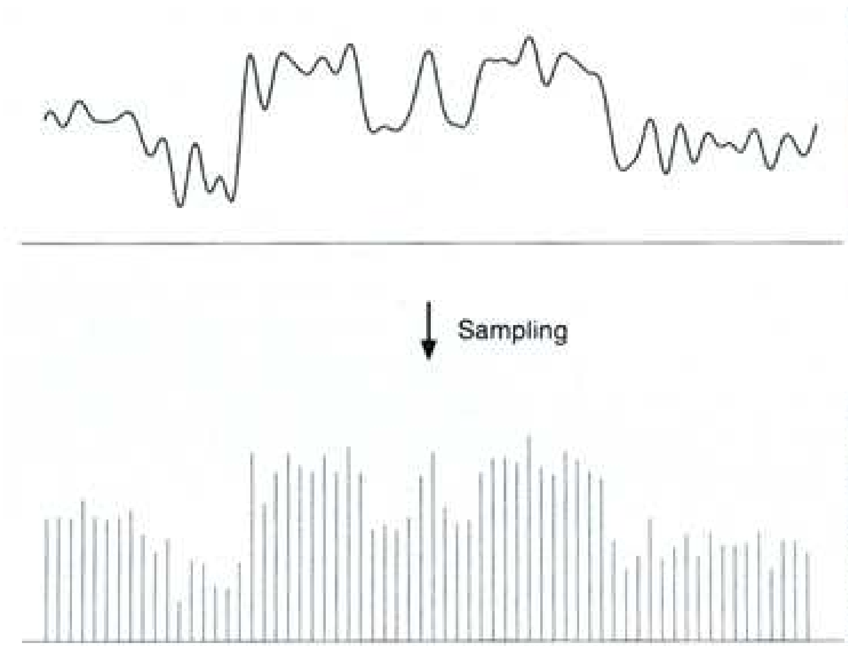
Sampling and Reconstruction





Sampled representations

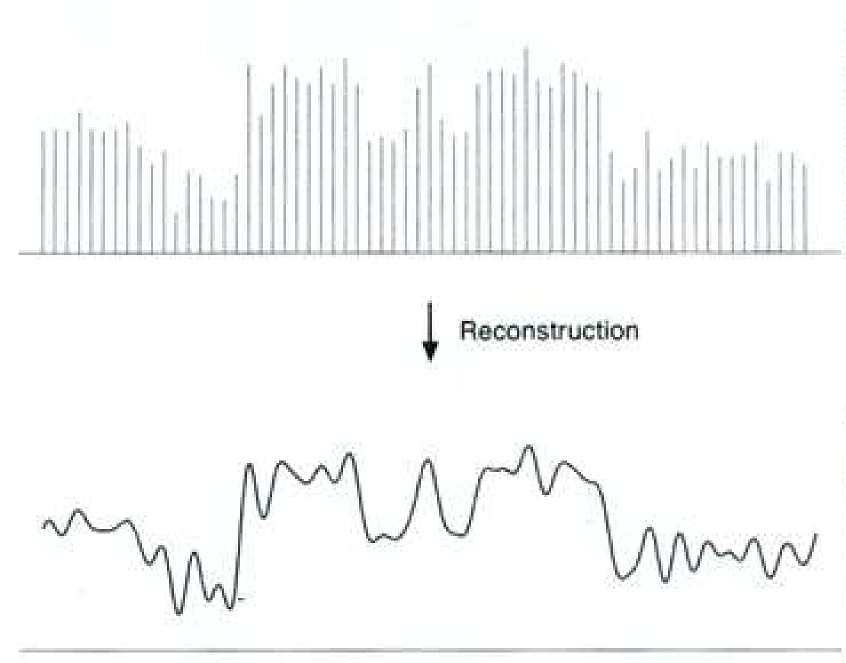
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function's values at many points



[FvDFH fig.14.14b / Wolberg]

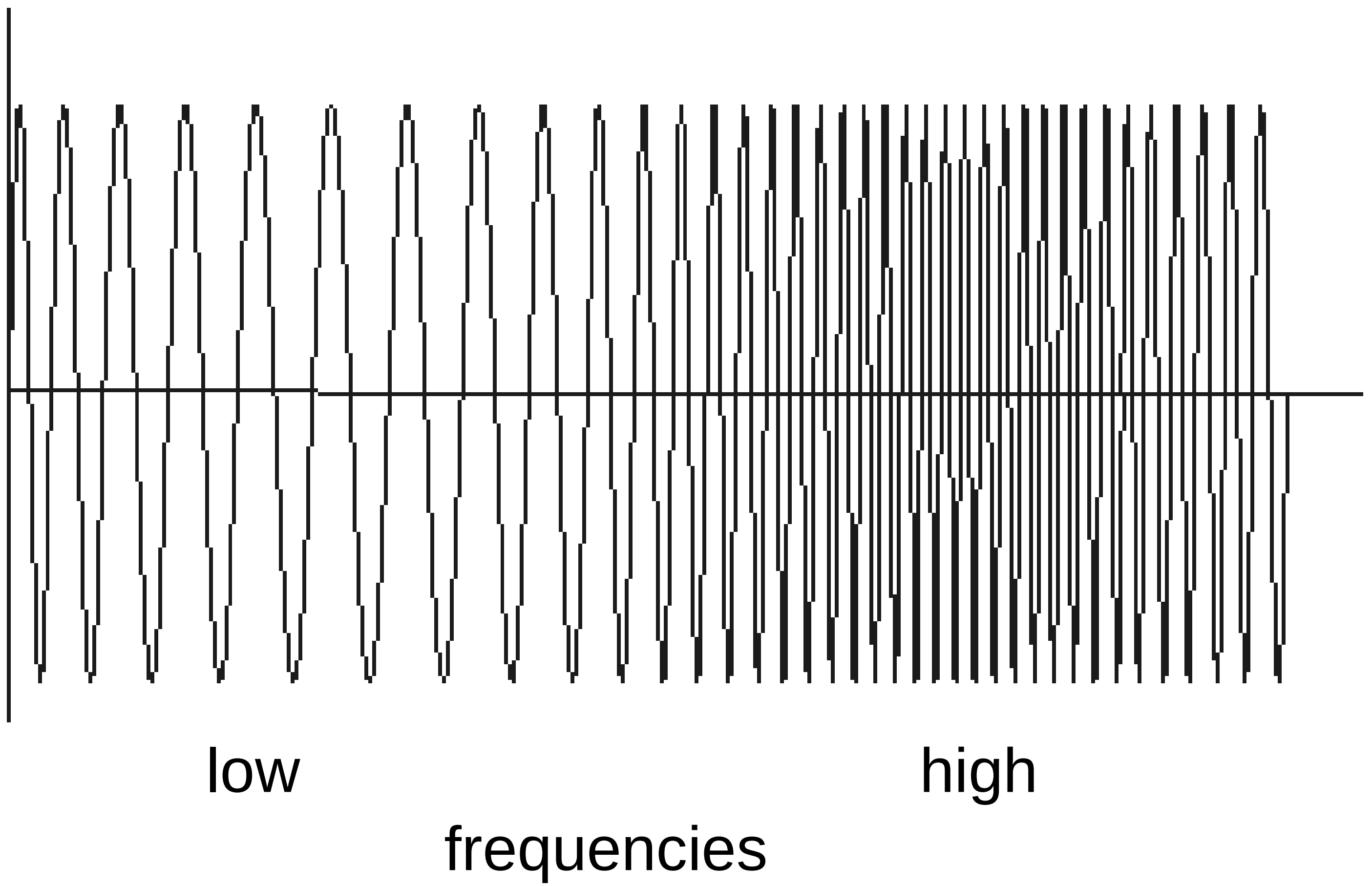
Reconstruction

- 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



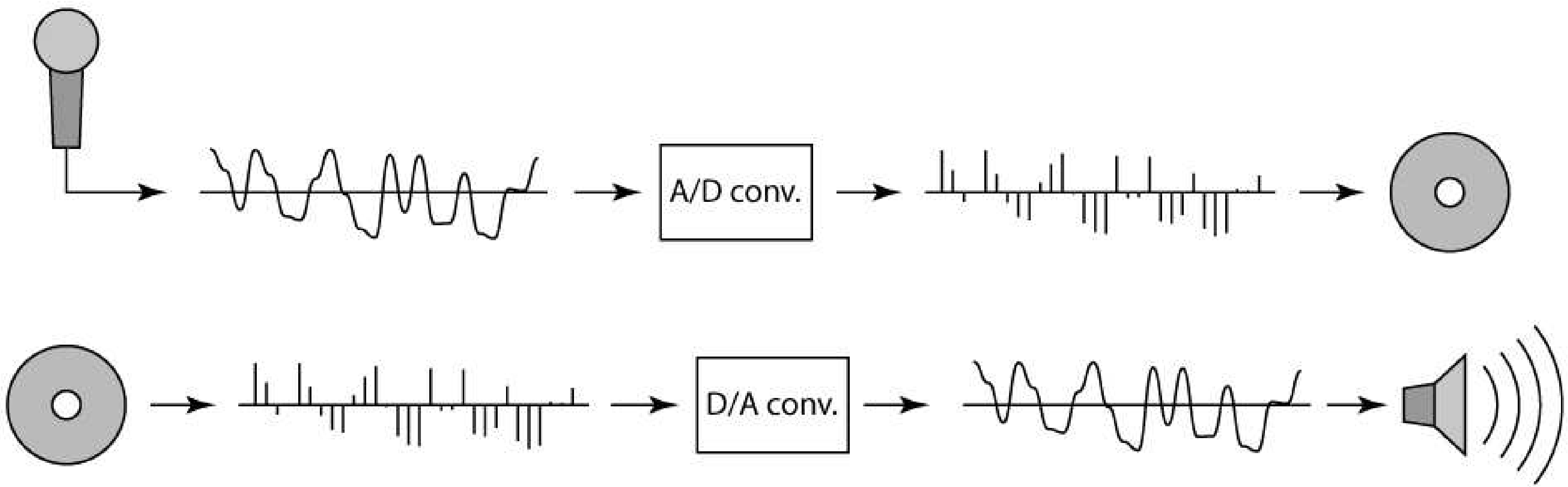
[FvDFH fig.14.14b / Wolberg]

1D Example: Audio



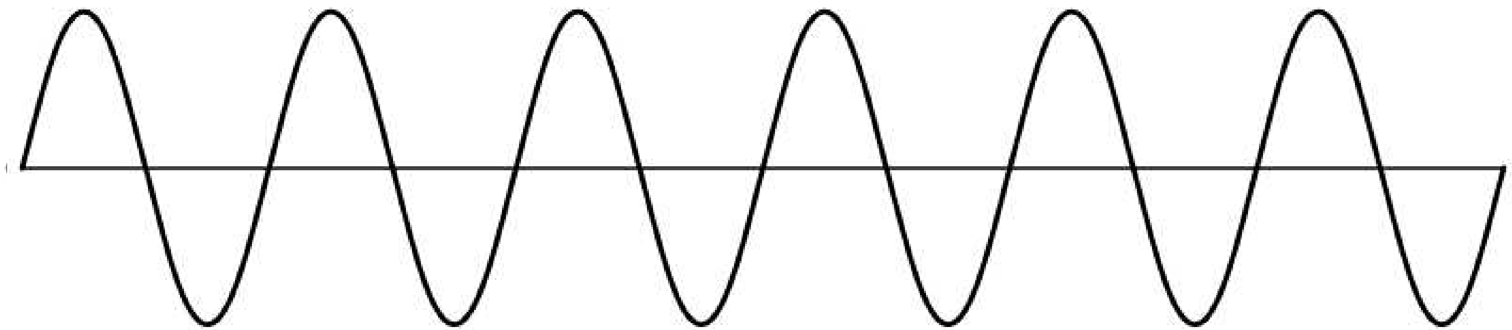
Sampling in digital audio

- 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?



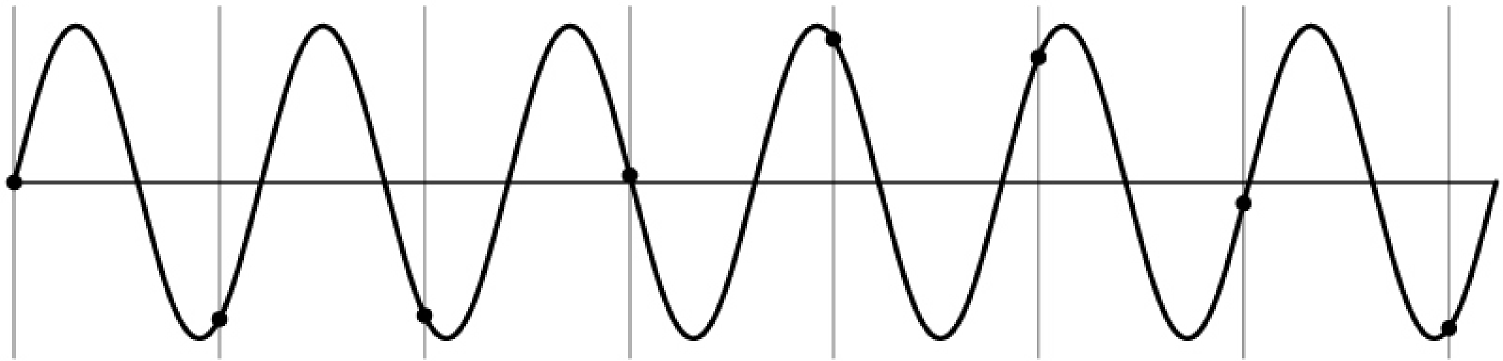
Sampling and Reconstruction

- Simple example: a sign wave



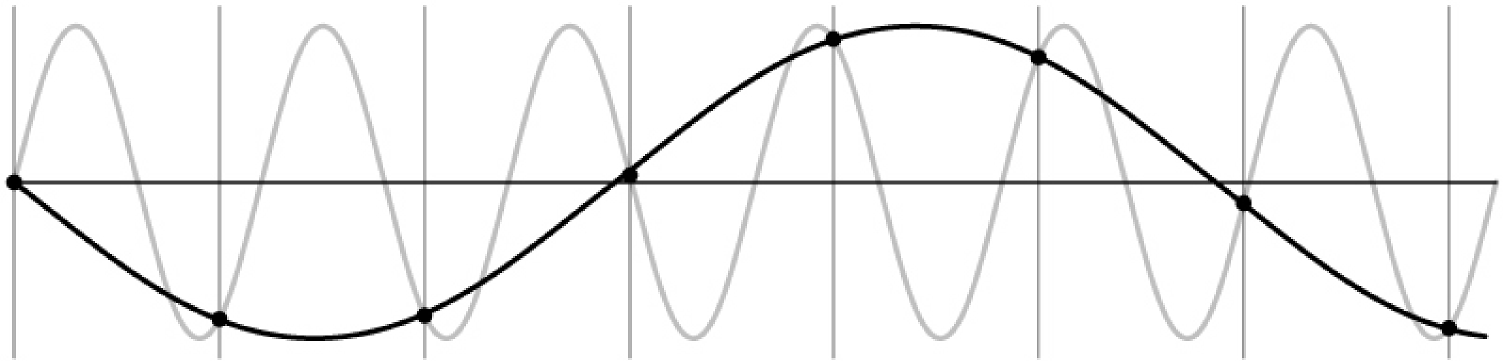
Undersampling

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



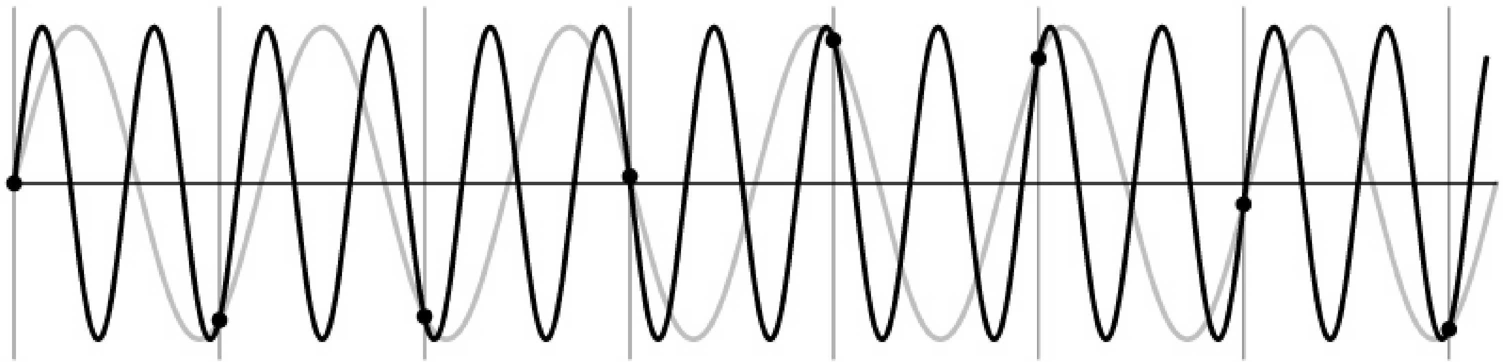
Undersampling

- 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

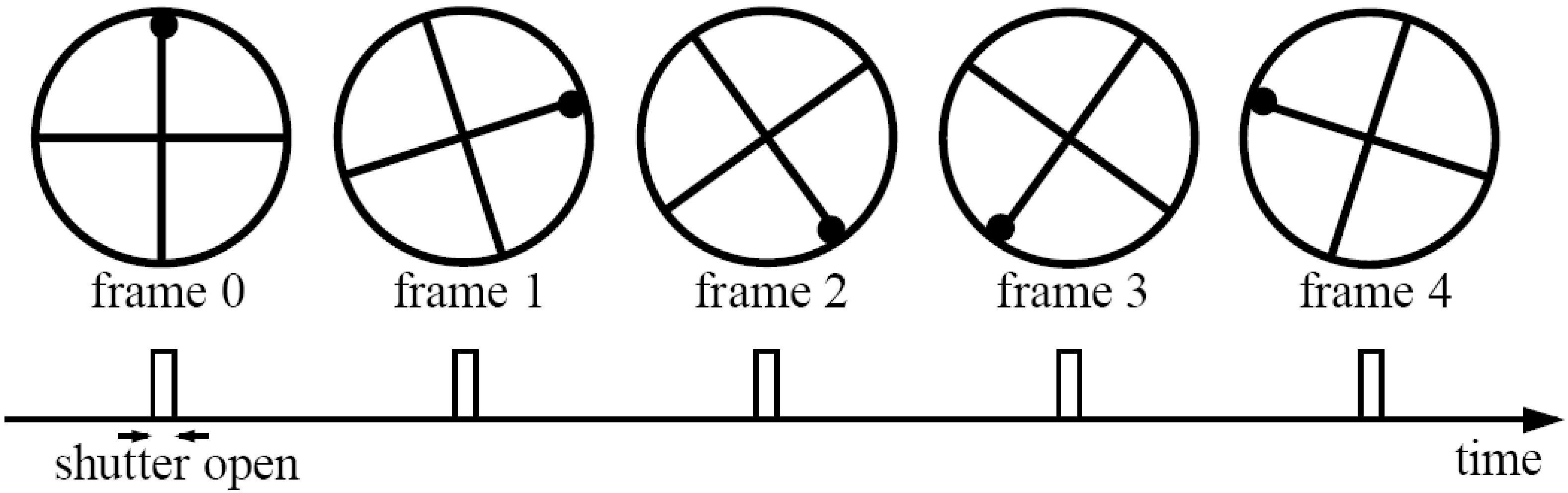
- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals “traveling in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



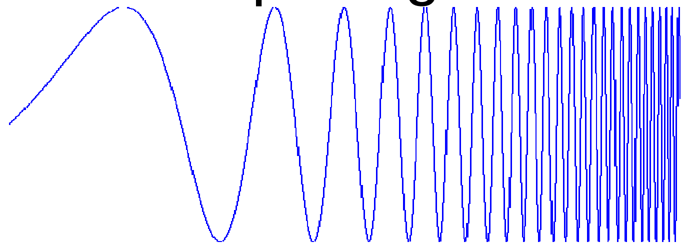
Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Slide by Steve Seitz

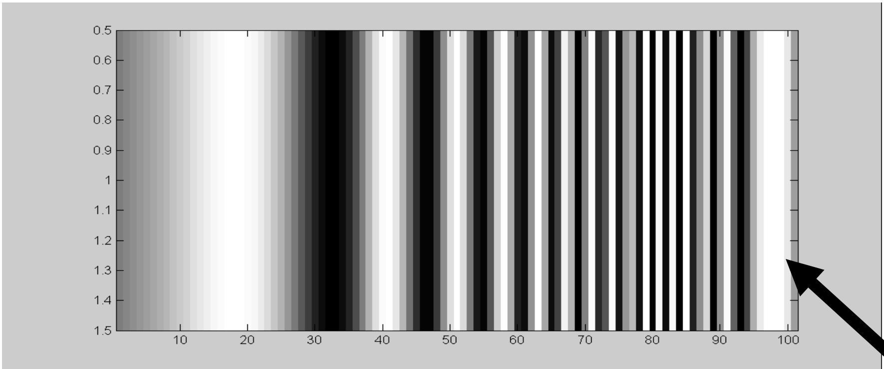
Aliasing in images

What's happening?

Input signal:

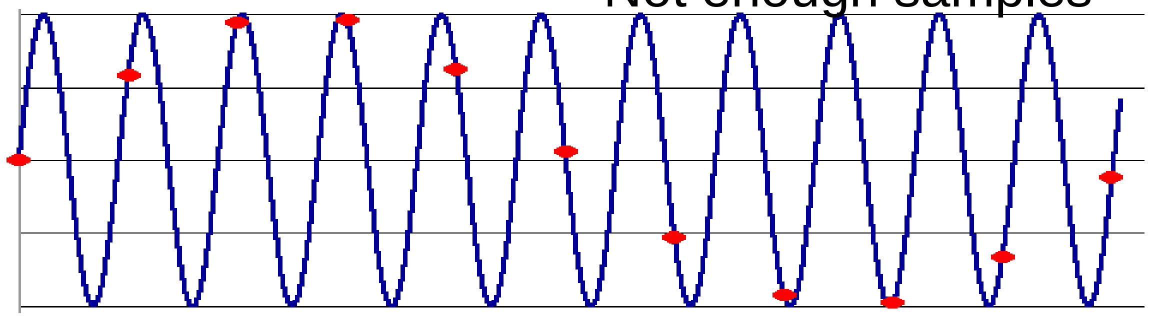


Plot as image:



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

Alias!
Not enough samples



Antialiasing

What can we do about aliasing?

Sample more often

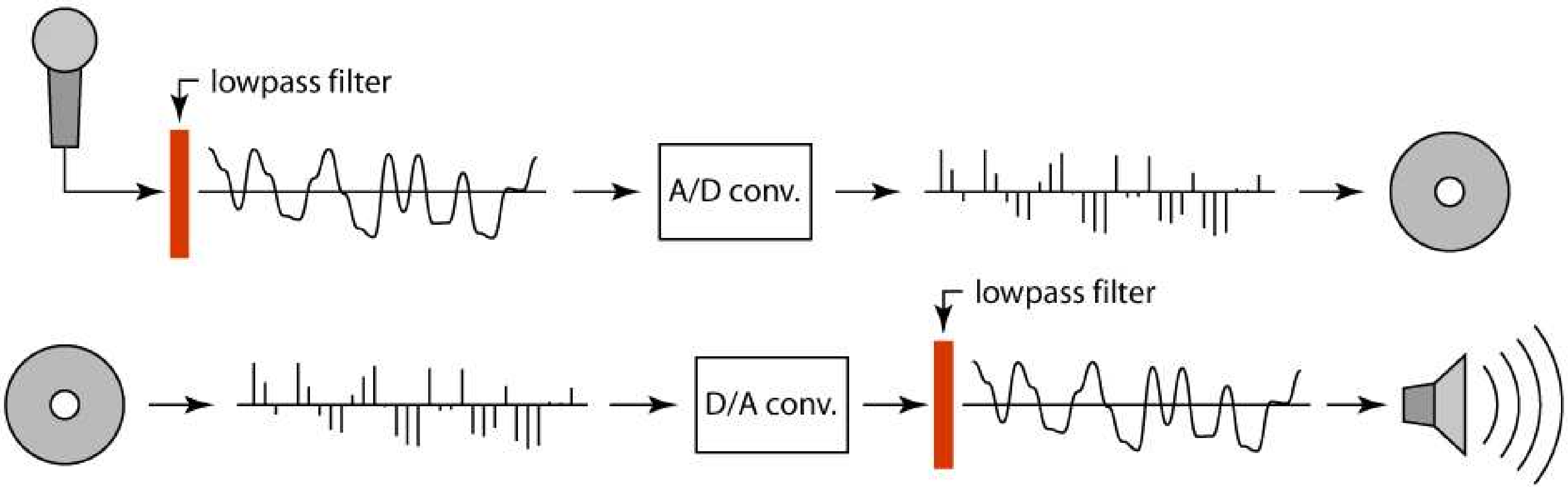
- Join the Mega-Pixel craze of the photo industry
- But this can’t go on forever

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)

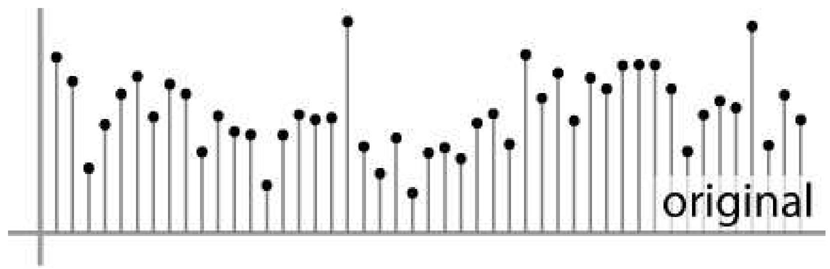


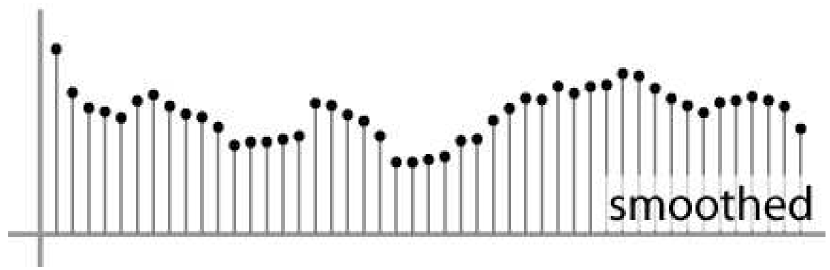
Linear filtering: a key idea

- 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

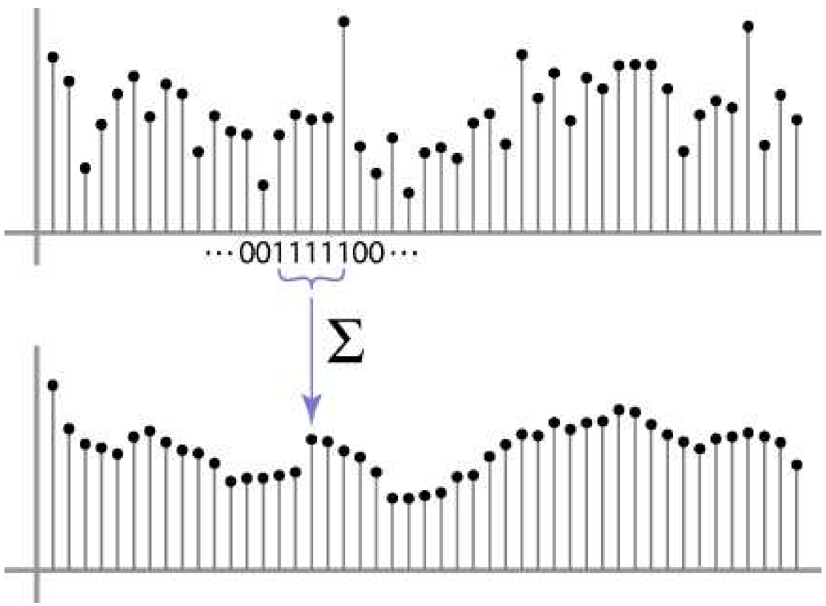
- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing





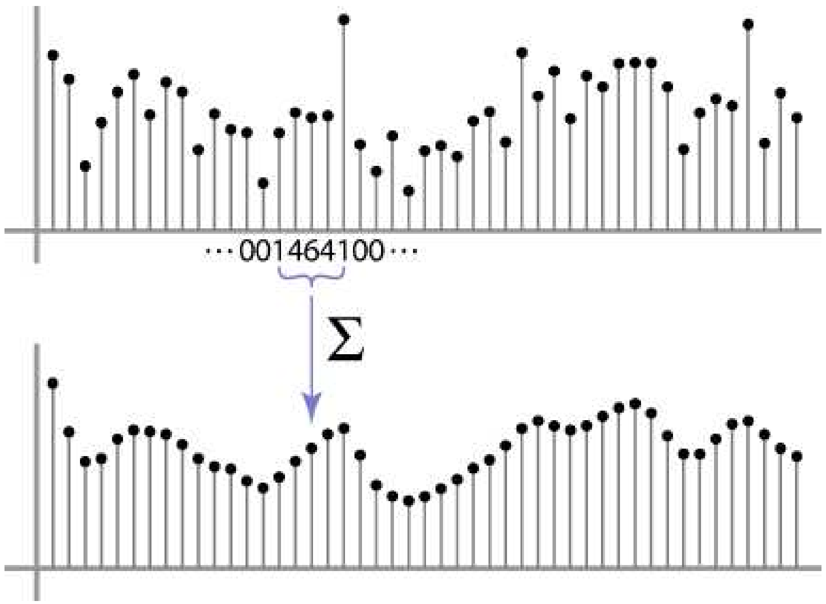
Weighted Moving Average

- 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

$$F[x, y]$$

$$H[u, v]$$

Slide by Steve Seitz

Cross-correlation filtering

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

  - We can generalize this idea by allowing different weights for different neighboring pixels:

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

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

$$G = H \otimes F$$

  - H is called the "filter," "kernel," or "mask."

Slide by Steve Seitz

Gaussian filtering

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

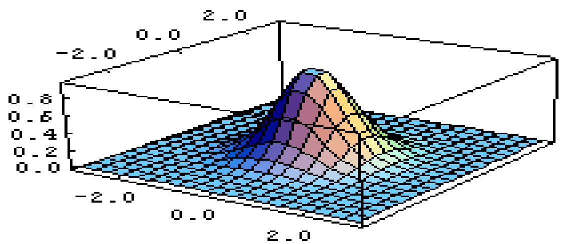
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

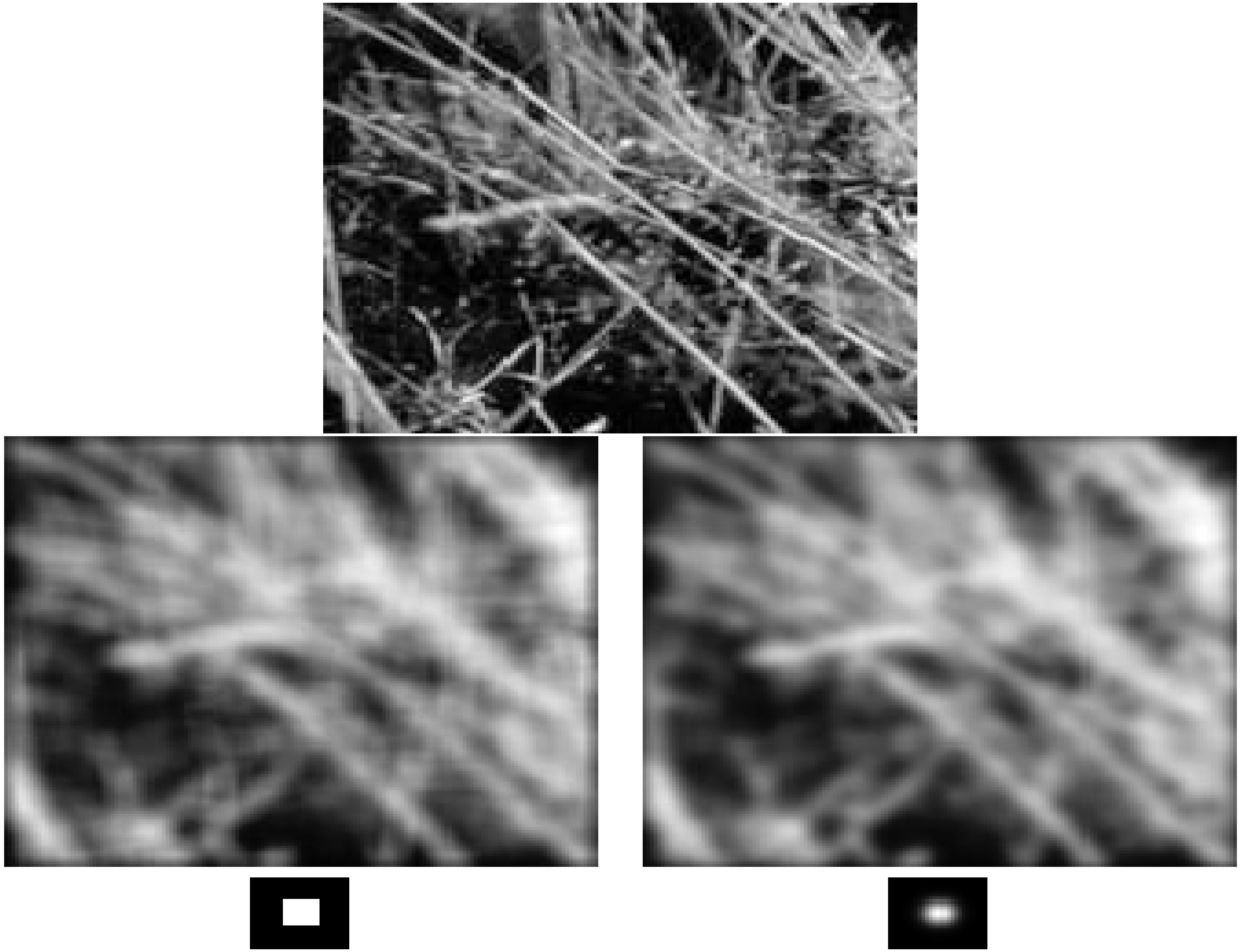
$H[u, v]$

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



Slide by Steve Seitz

Mean vs. Gaussian filtering



Slide by Steve Seitz

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:

$$G = H \star F$$

Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

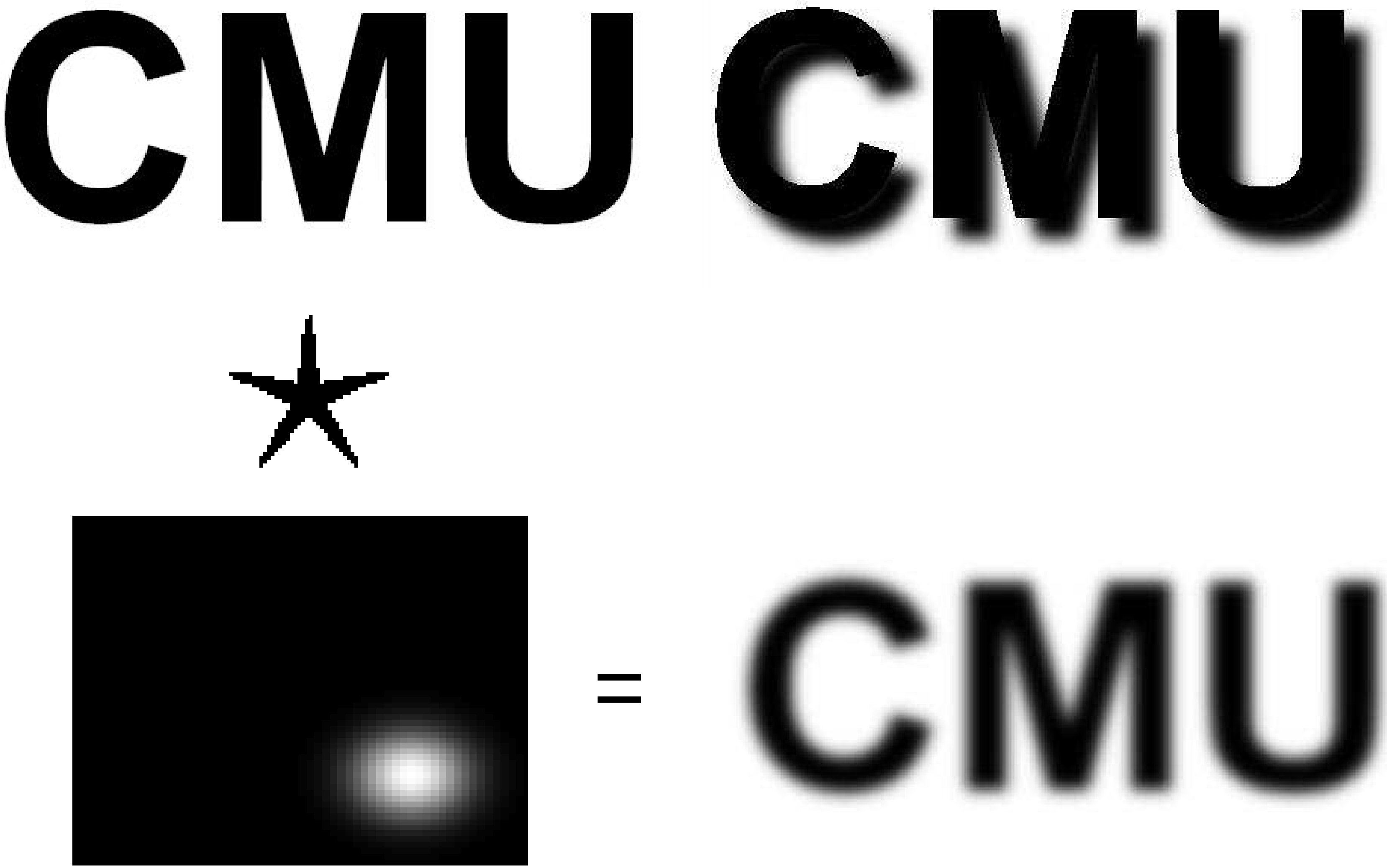
Slide by Steve Seitz

Convolution is nice!

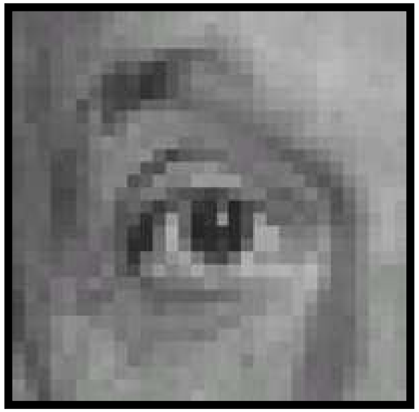
- Notation: $b = c \star a$
- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - 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: $a * (b_1 * b_2 * b_3)$

Tricks with convolutions



Practice with linear filters



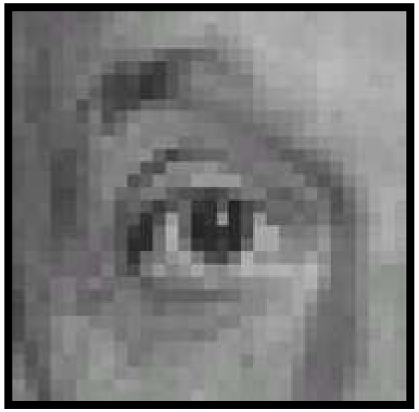
Original

0	0	0
0	1	0
0	0	0

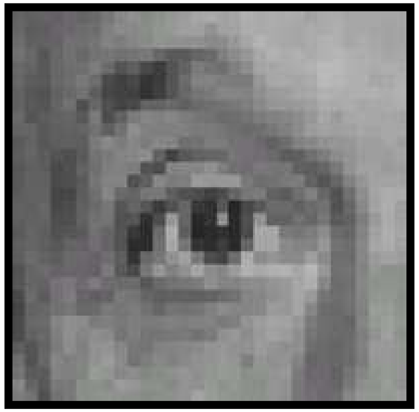
?

Source: D. Lowe

Practice with linear filters



0	0	0
0	1	0
0	0	0

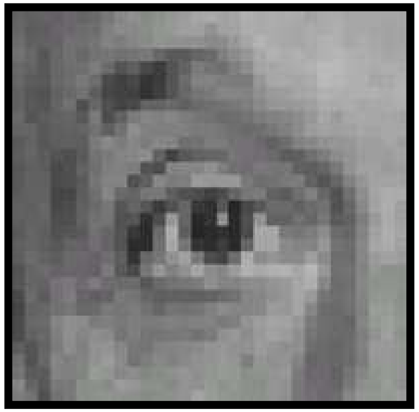


Original

Filtered
(no change)

Source: D. Lowe

Practice with linear filters



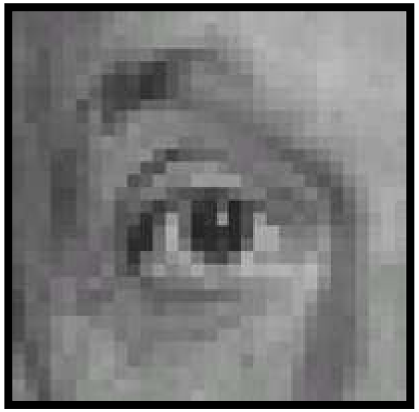
Original

0	0	0
0	0	1
0	0	0

?

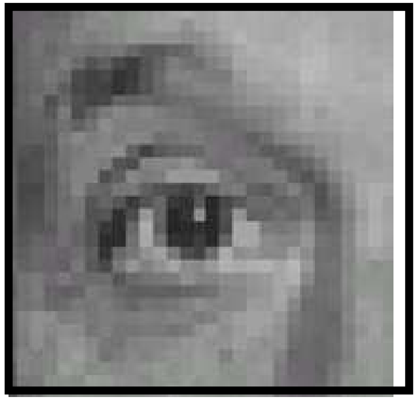
Source: D. Lowe

Practice with linear filters



Original

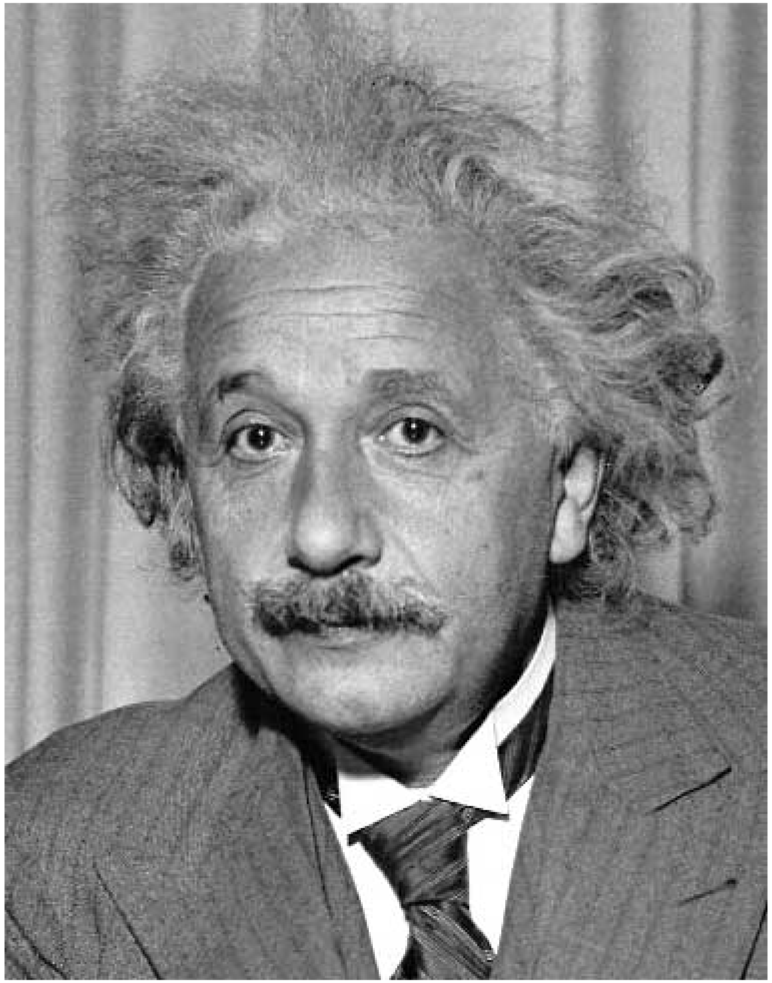
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

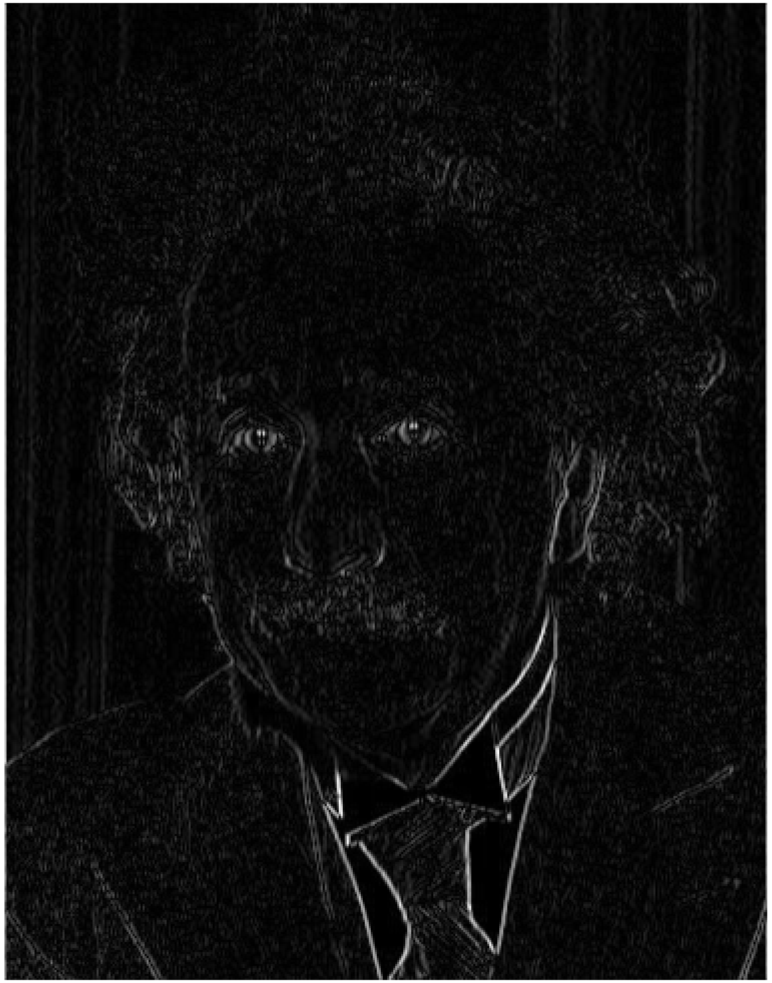
Source: D. Lowe

Other filters



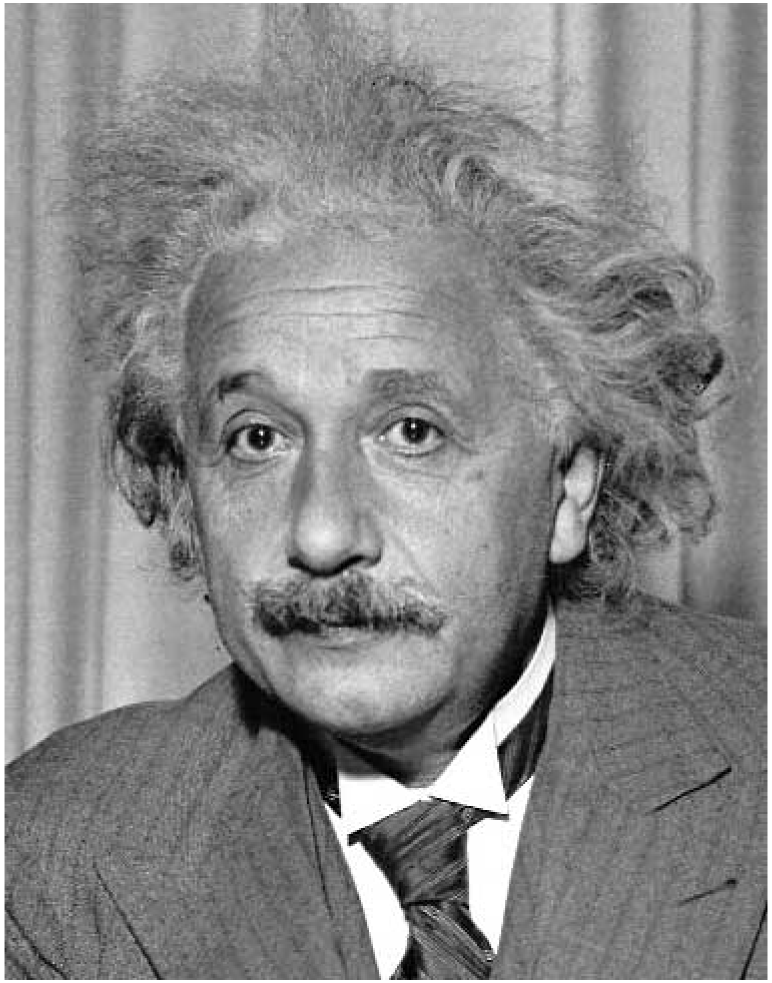
1	0	-1
2	0	-2
1	0	-1

Sobel



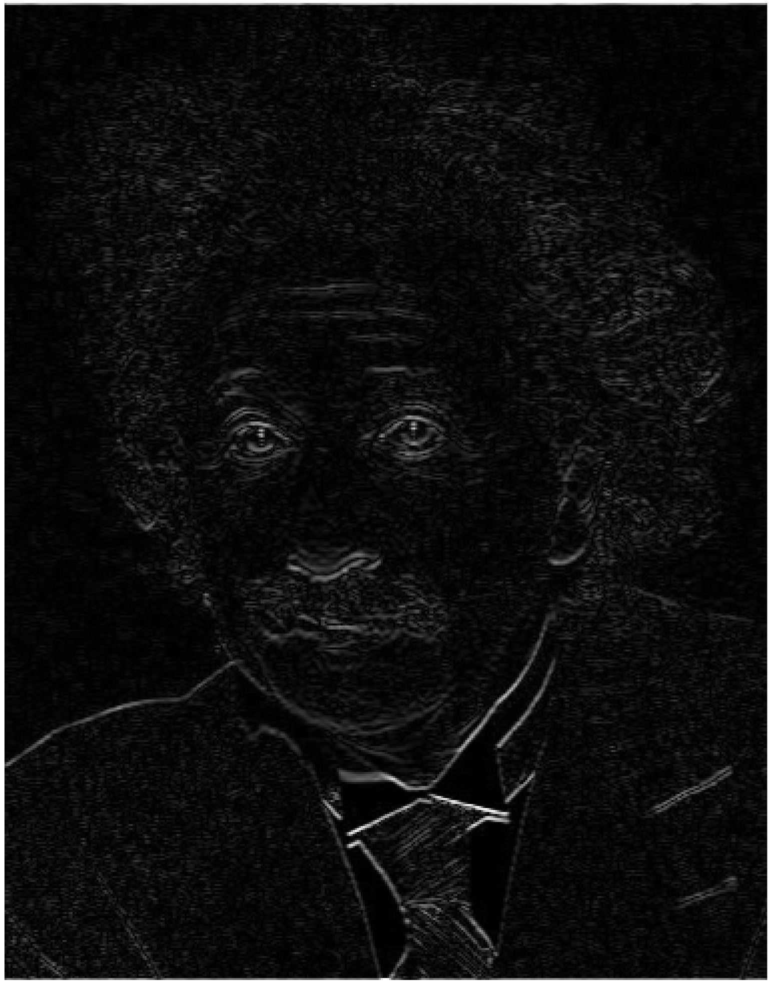
Vertical Edge
(absolute value)

Other filters



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

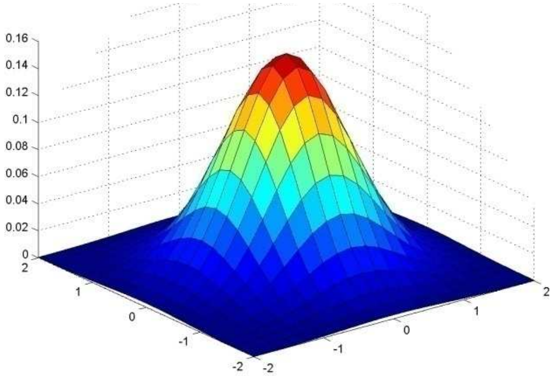
Sobel

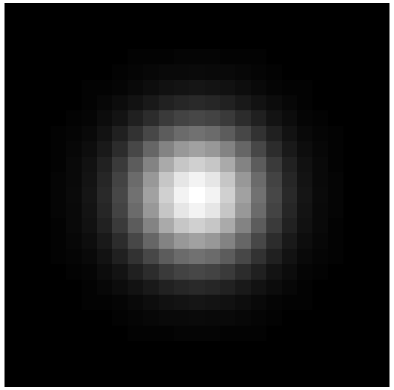


Horizontal Edge
(absolute value)

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

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

Slide credit: Christopher Rasmussen

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Convolution with self is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$

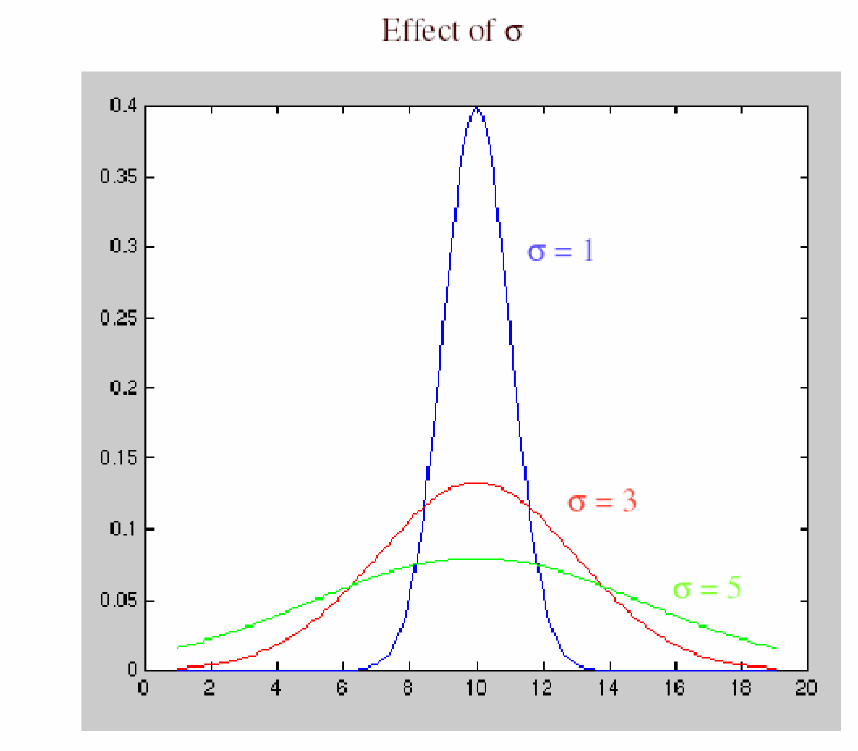
Source: K. Grauman

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 σ



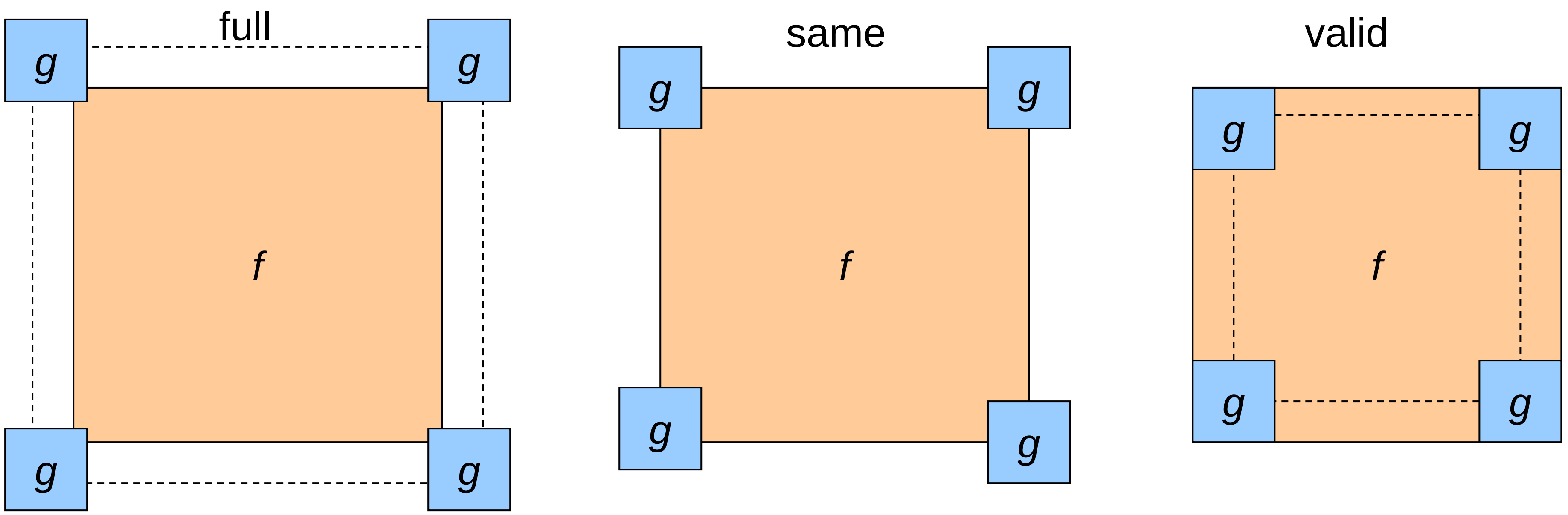
Side by Derek Hoiem

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



Source: S. Lazebnik

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

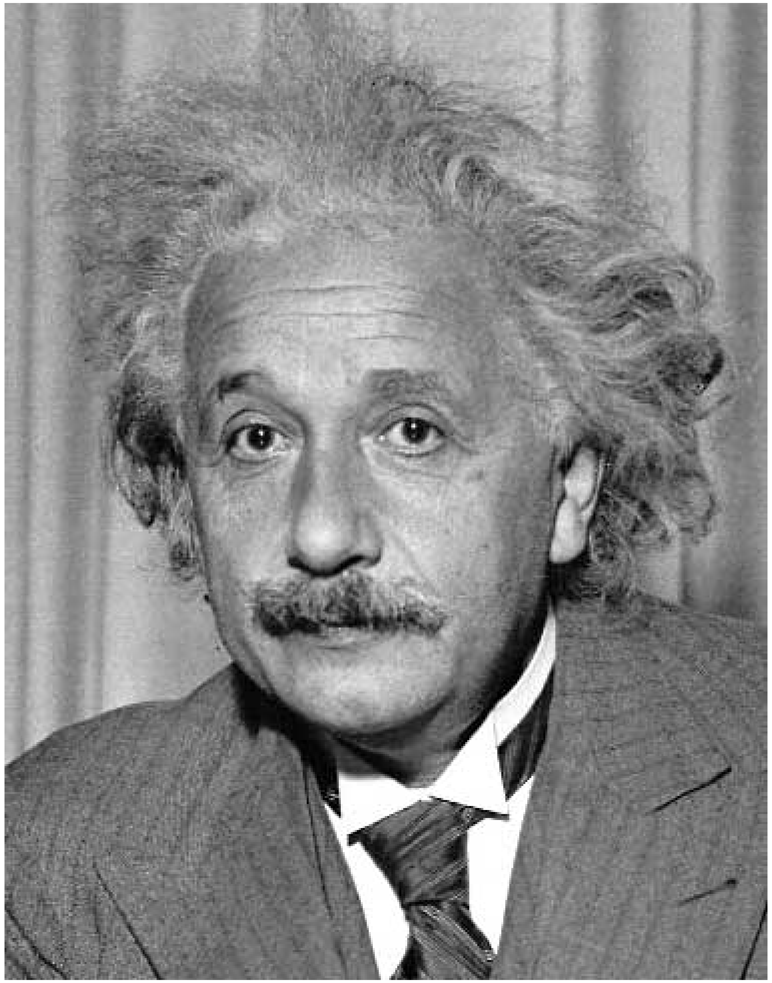
Practical matters

- 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

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

Side by Derek Hoiem

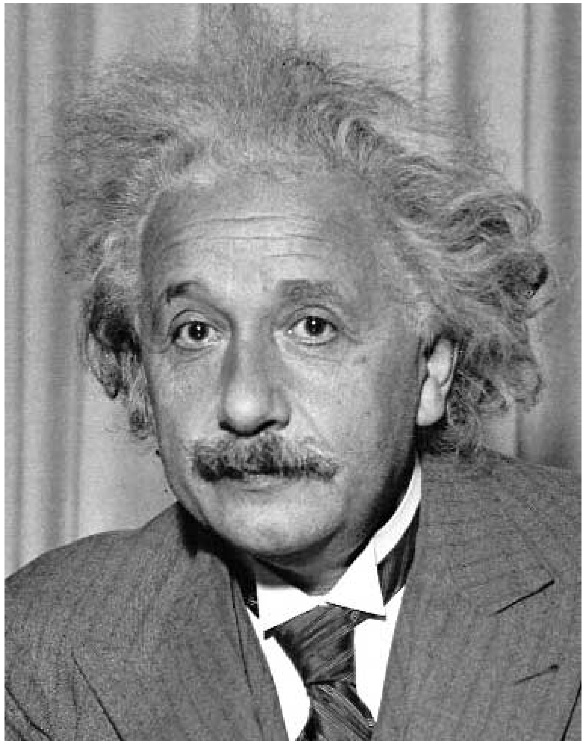
Matching with filters

Goal: find  in image

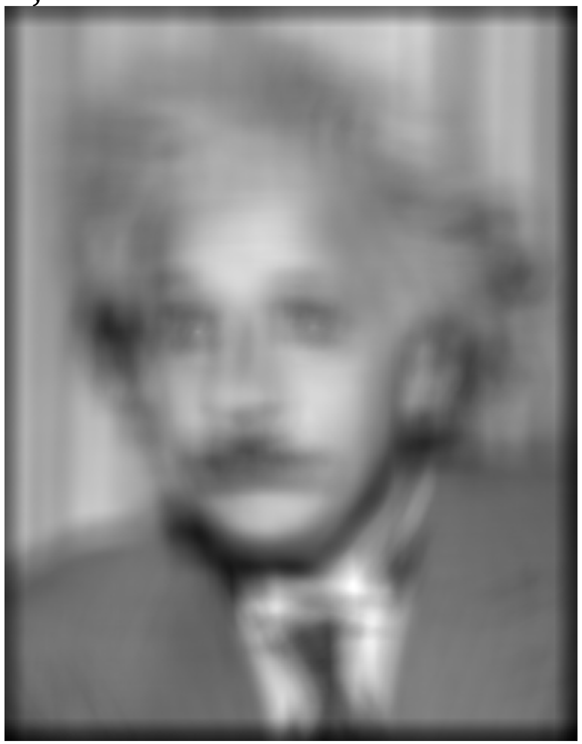
Method 0: filter the image with eye patch

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

f = image
g = filter



Input



Filtered Image

What went wrong?

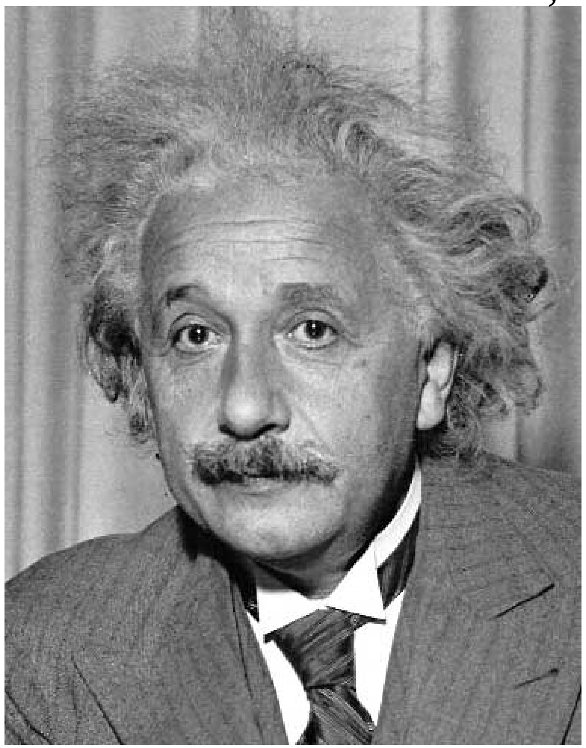
Side by Derek Hoiem

Matching with filters

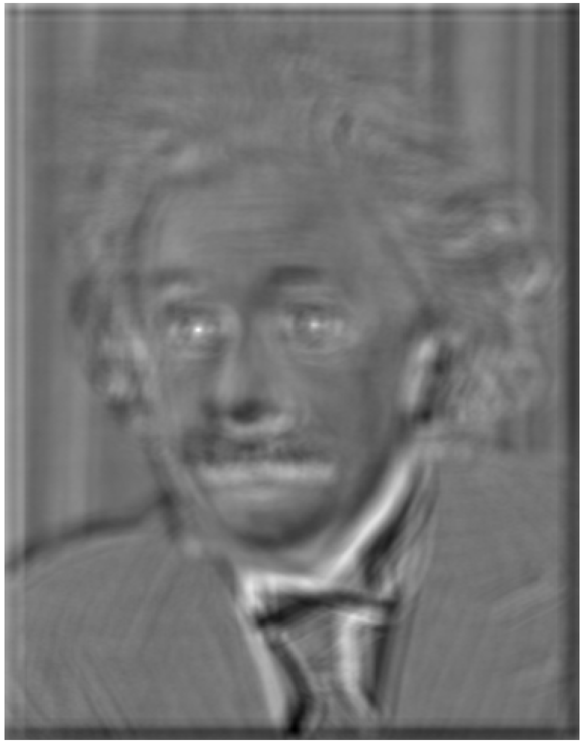
Goal: find  in image

Method 1: filter the image with zero-mean eye

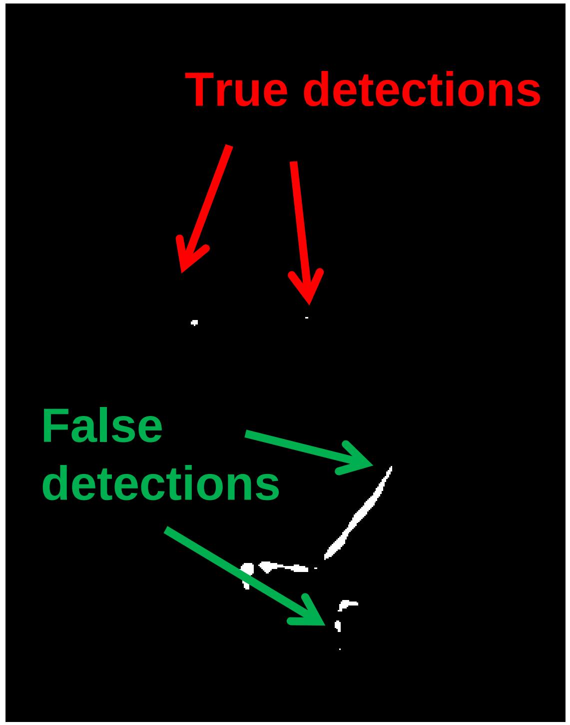
$$h[m,n]=\sum_{k,l}(f[k,l]-\underbrace{\bar{f}}_{\text{mean of f}})(g[m+k,n+l])$$



Input



Filtered Image (scaled)



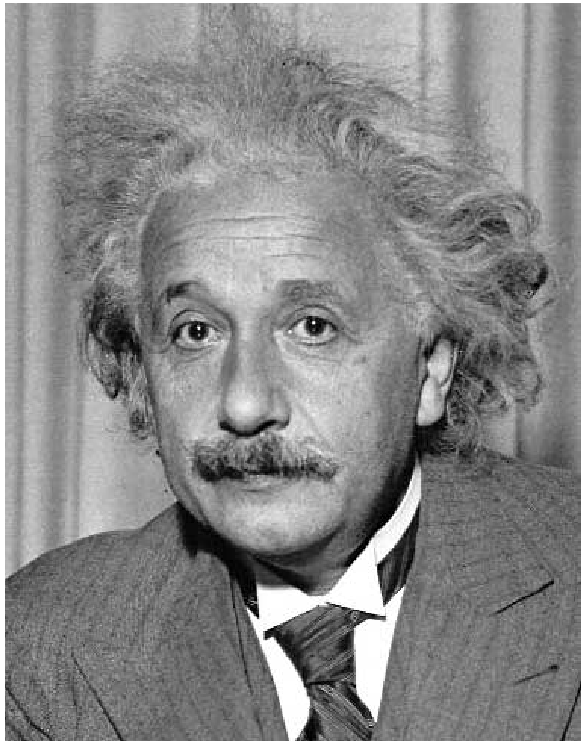
Thresholded Image

Matching with filters

Goal: find  in image

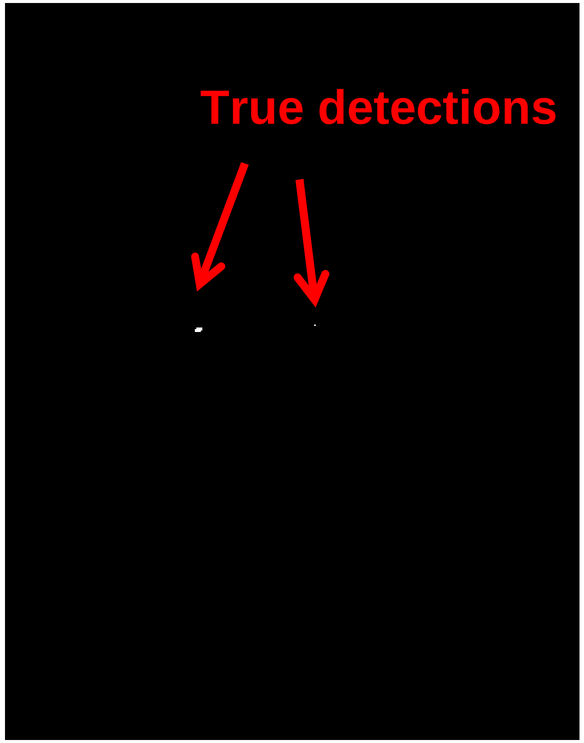
Method 2: SSD

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



Input

1- sqrt(SSD)



Thresholded Image

Matching with filters

Can SSD be implemented with linear filters?

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

Side by Derek Hoiem

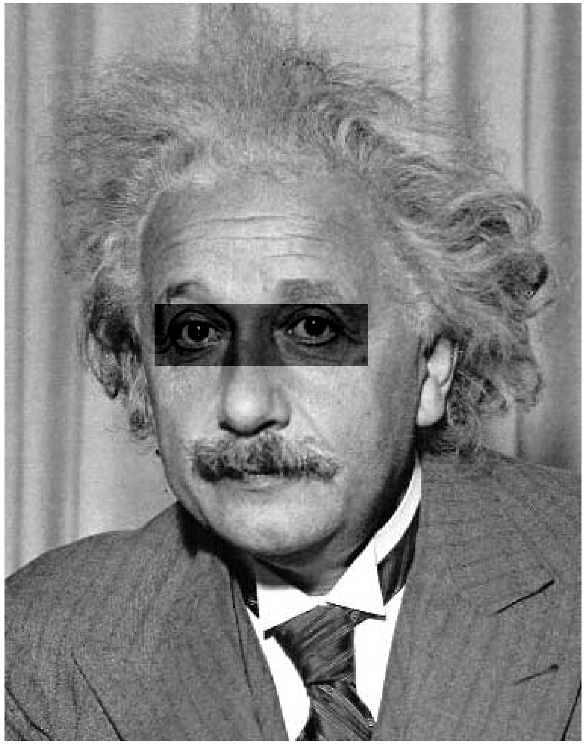
Matching with filters

Goal: find  in image

What's the potential downside of SSD?

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

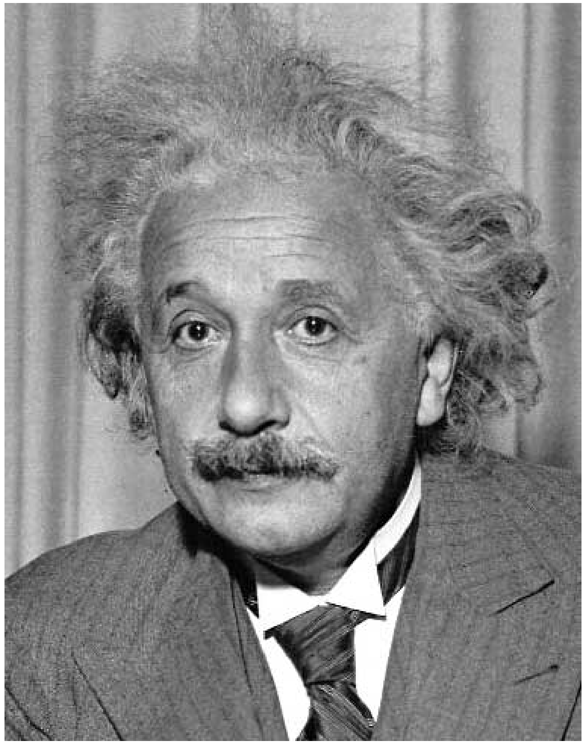
$$h[m,n] = \frac{\sum_{k,l} (g[k,l] - \overset{\substack{\text{mean template} \\ \downarrow}}{\bar{g}})(f[m+k,n+l] - \overset{\substack{\text{mean image patch} \\ \downarrow}}{\bar{f}_{m,n}})}{\left(\sum_{k,l} (g[k,l] - \bar{g})^2 \sum_{k,l} (f[m+k,n+l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

Side by Derek Hoiem

Matching with filters

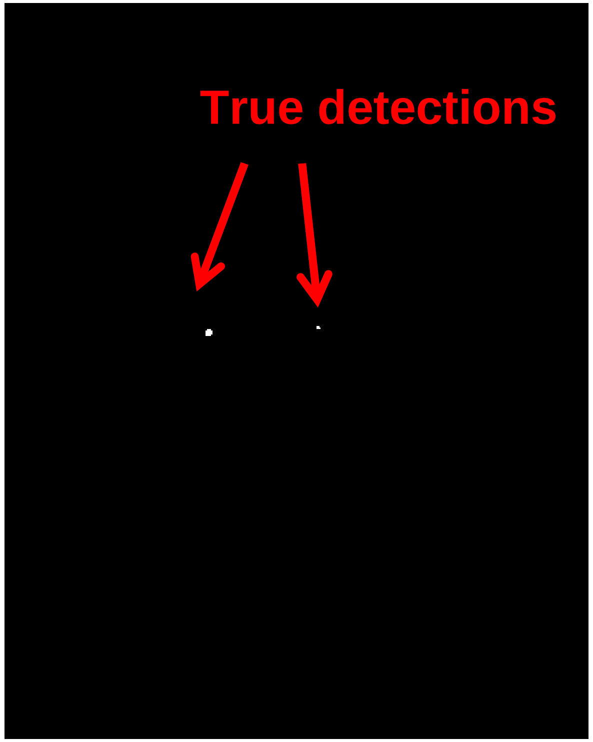
Goal: find  in image

Method 3: Normalized cross-correlation



Input

Normalized X-Correlation

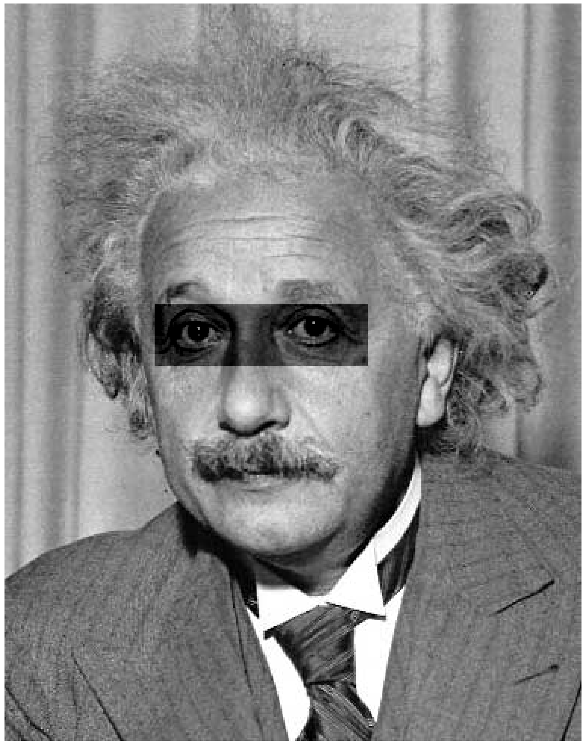


Thresholded Image

Matching with filters

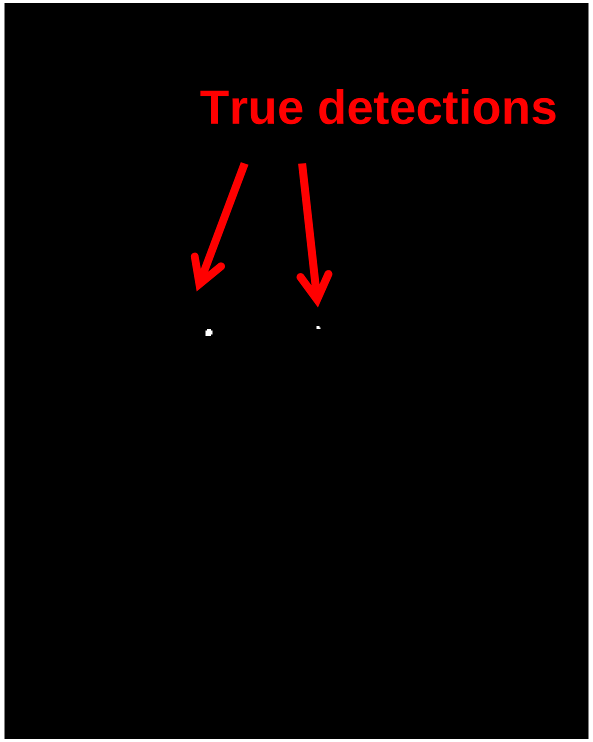
Goal: find  in image

Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



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

Side by Derek Hoiem

Image half-sizing

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

How to generate a half-sized version?

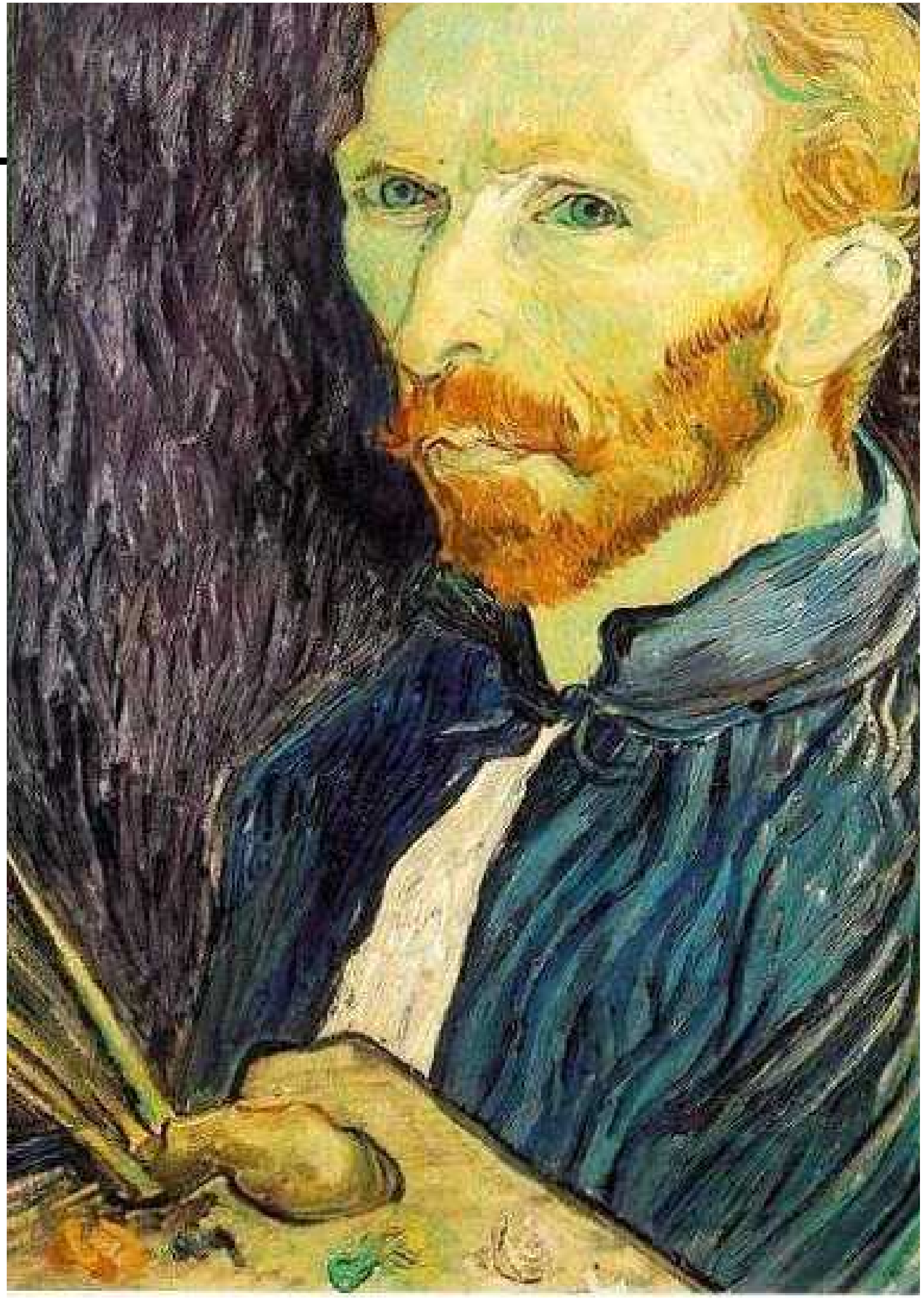
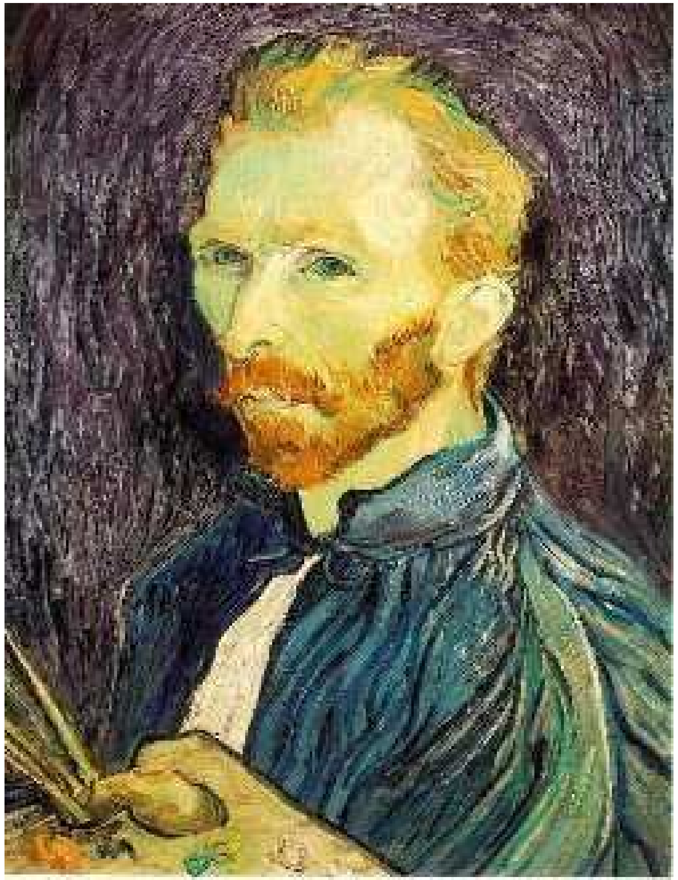
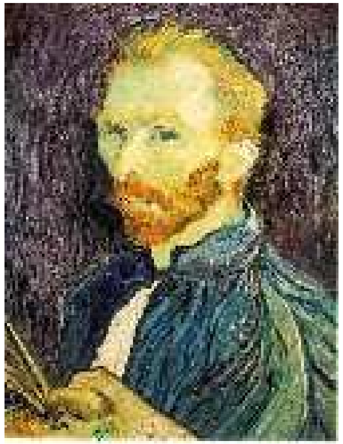


Image sub-sampling





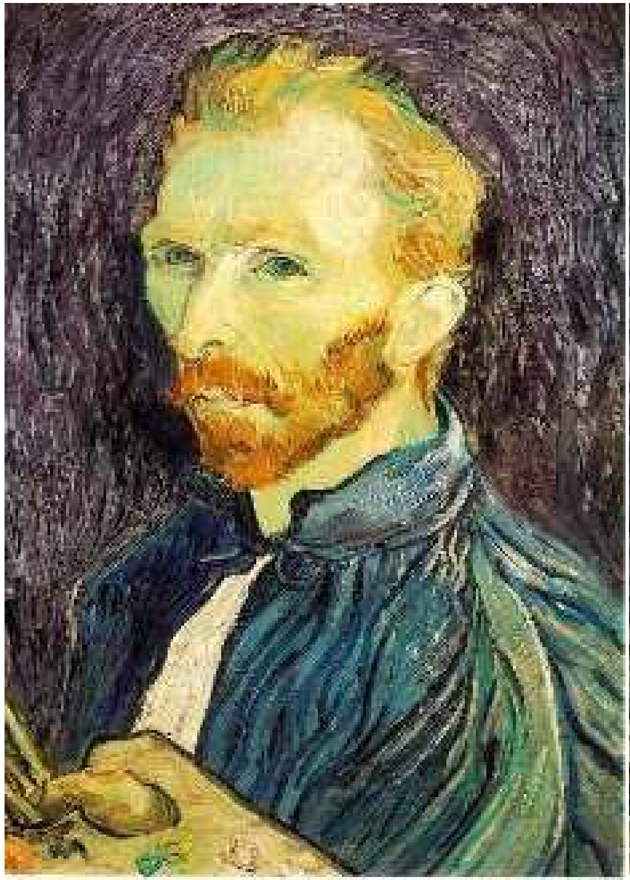
1/4

1/8

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

Slide by Steve Seitz

Image sub-sampling



1/2

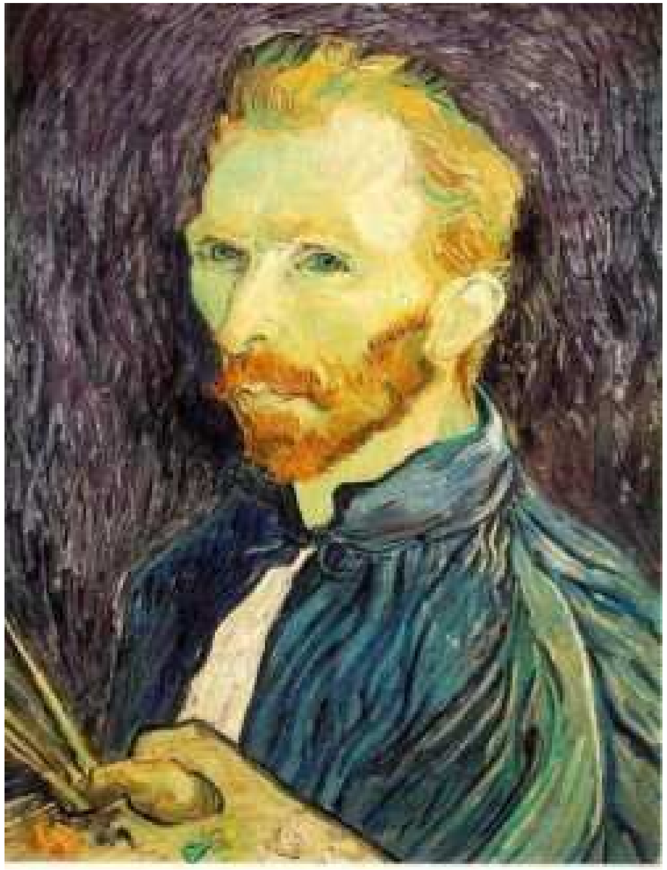
1/4 (2x zoom)

1/8 (4x zoom)

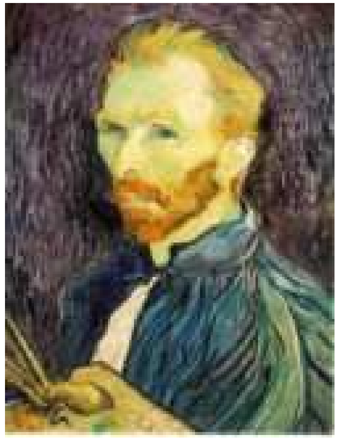
Aliasing! What do we do?

Slide by Steve Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2



G 1/4

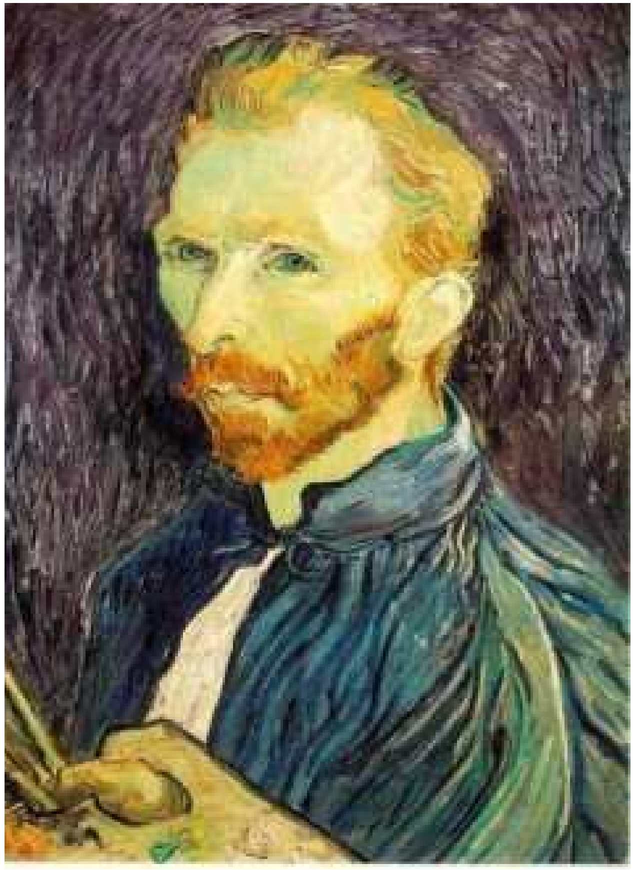
G 1/8

Solution: filter the image, *then* subsample

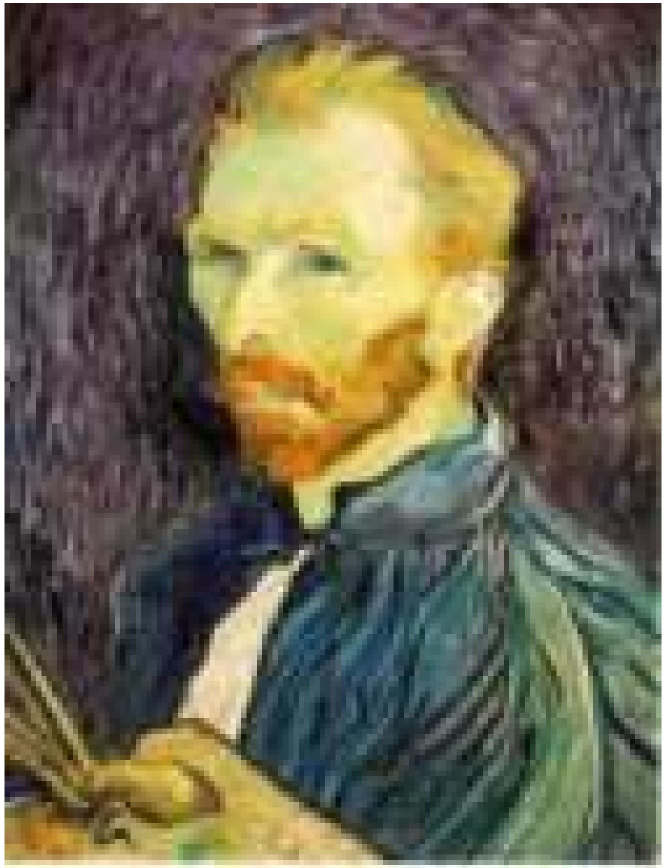
- Filter size should double for each ½ size reduction. Why?

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

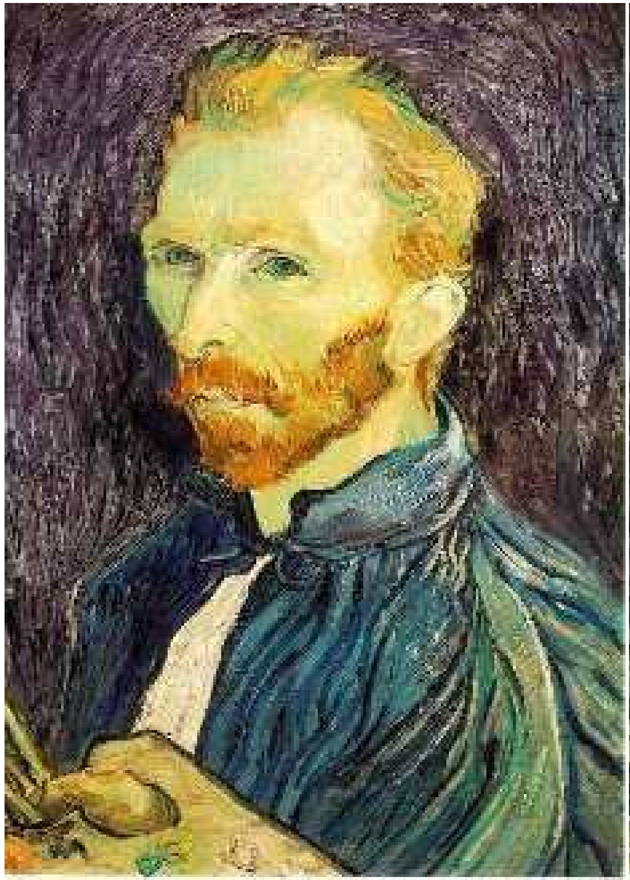


G 1/4

G 1/8

Slide by Steve Seitz

Compare with...



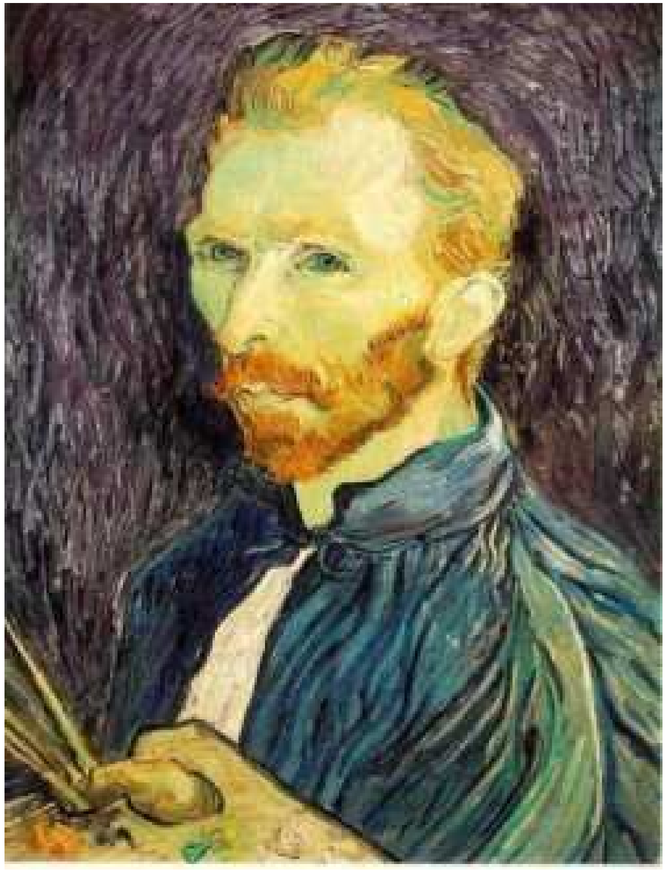
1/2

1/4 (2x zoom)

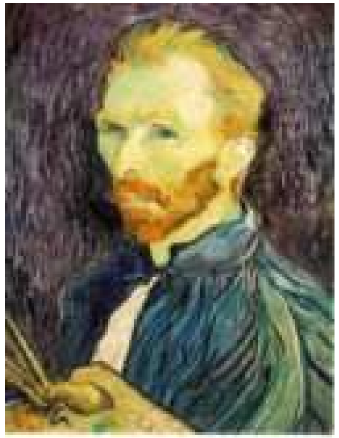
1/8 (4x zoom)

Slide by Steve Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2



G 1/4

G 1/8

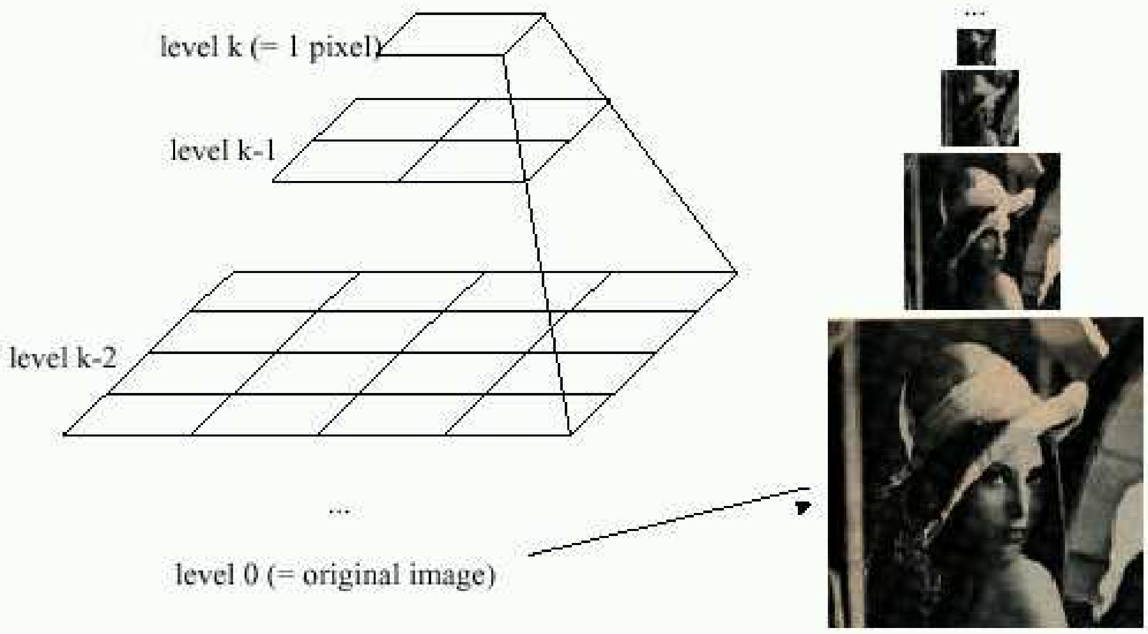
Solution: filter the image, *then* subsample

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

Slide by Steve Seitz

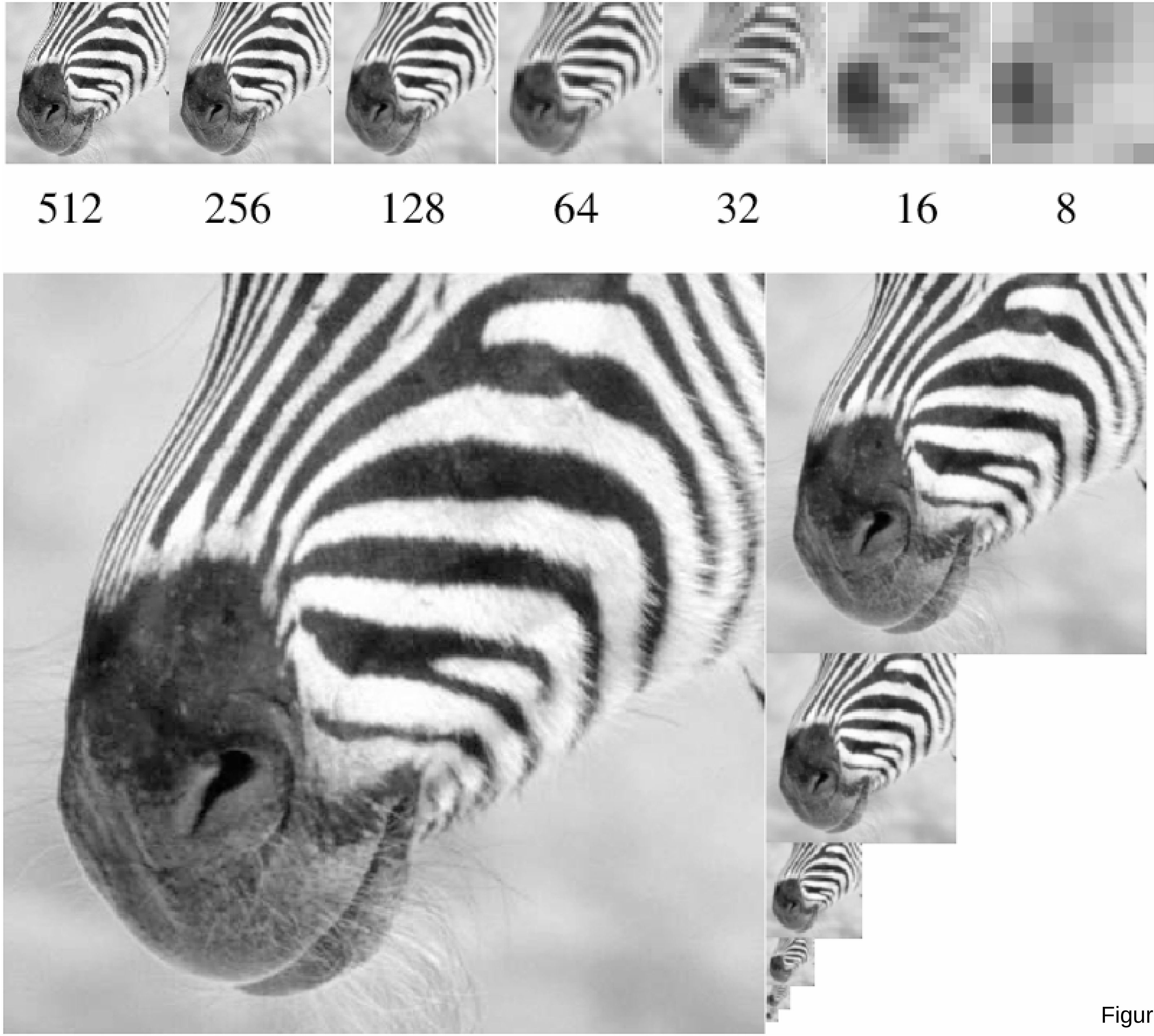
Image Pyramids

Idea: Represent NxN image as a “pyramid” of 1x1, 2x2, 4x4,…, 2^kx2^k images (assuming N=2^k)



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

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



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

What are they good for?

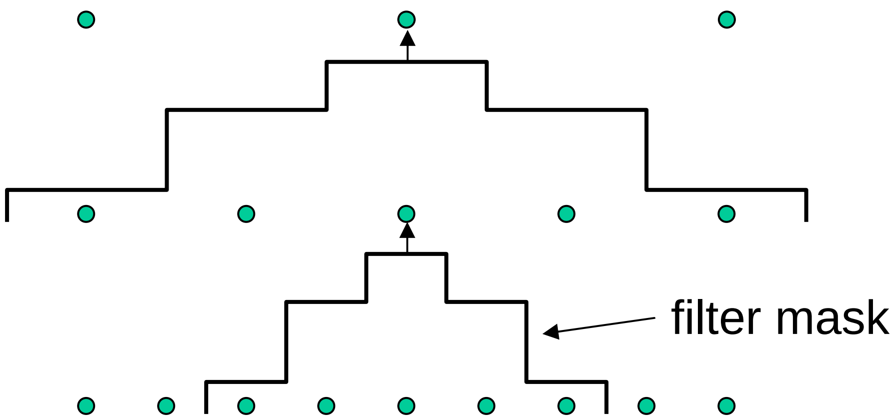
Improve Search

- Search over translations
  - Like project 1
  - Classic coarse-to-fine strategy
- Search over scale
  - Template matching
  - E.g. find a face at different scales

Pre-computation

- Need to access image at different blur levels
- Useful for texture mapping at different resolutions (called mip-mapping)

Gaussian pyramid construction



Repeat

- Filter
- Subsample

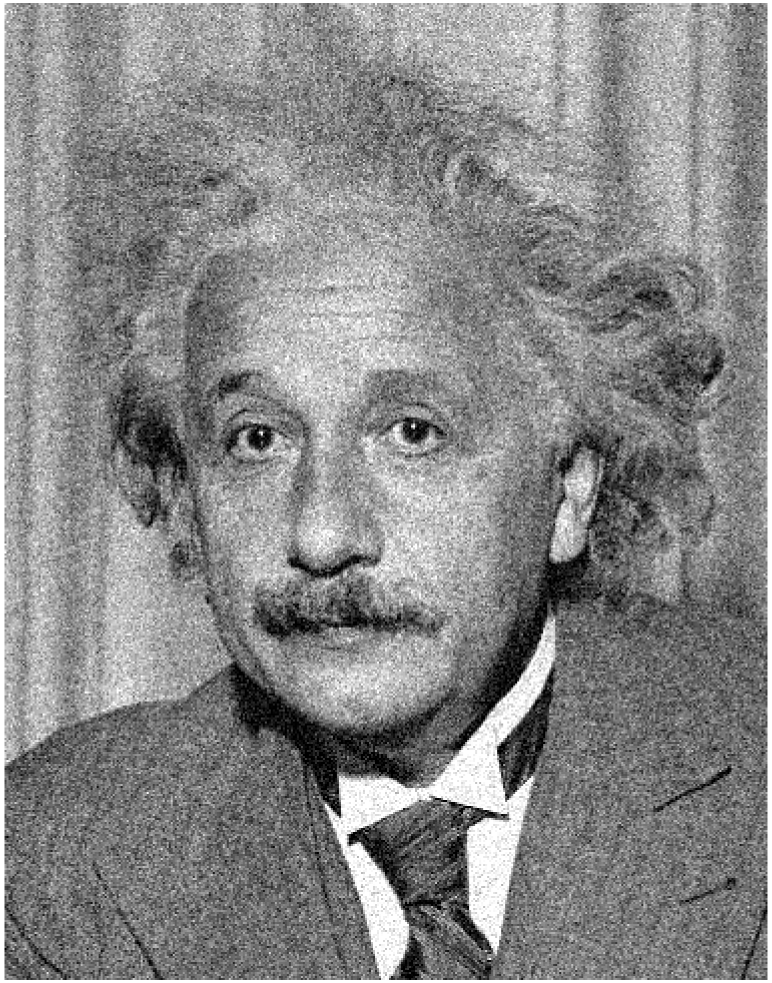
Until minimum resolution reached

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

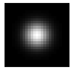
The whole pyramid is only 4/3 the size of the original image!

Slide by Steve Seitz

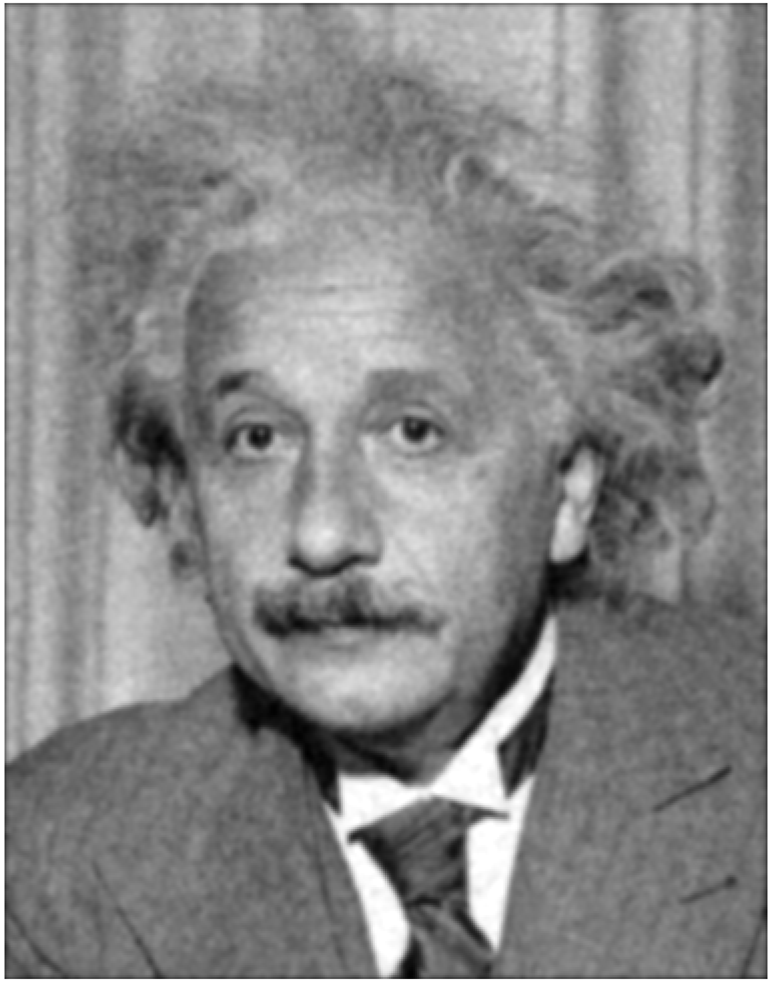
Denoising



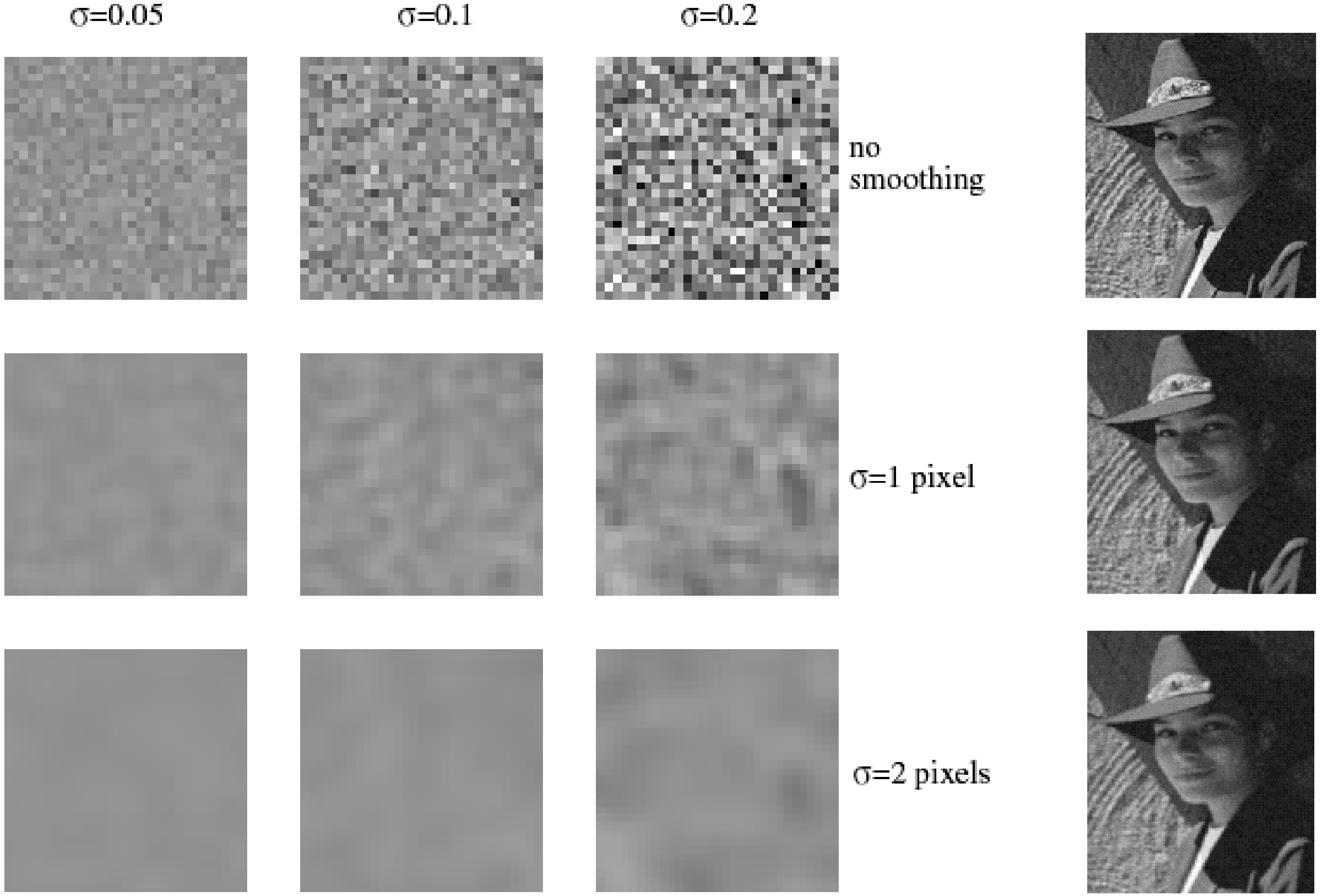
Additive Gaussian Noise



Gaussian
Filter



Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Source: S. Lazebnik

Reducing salt-and-pepper noise by Gaussian smoothing

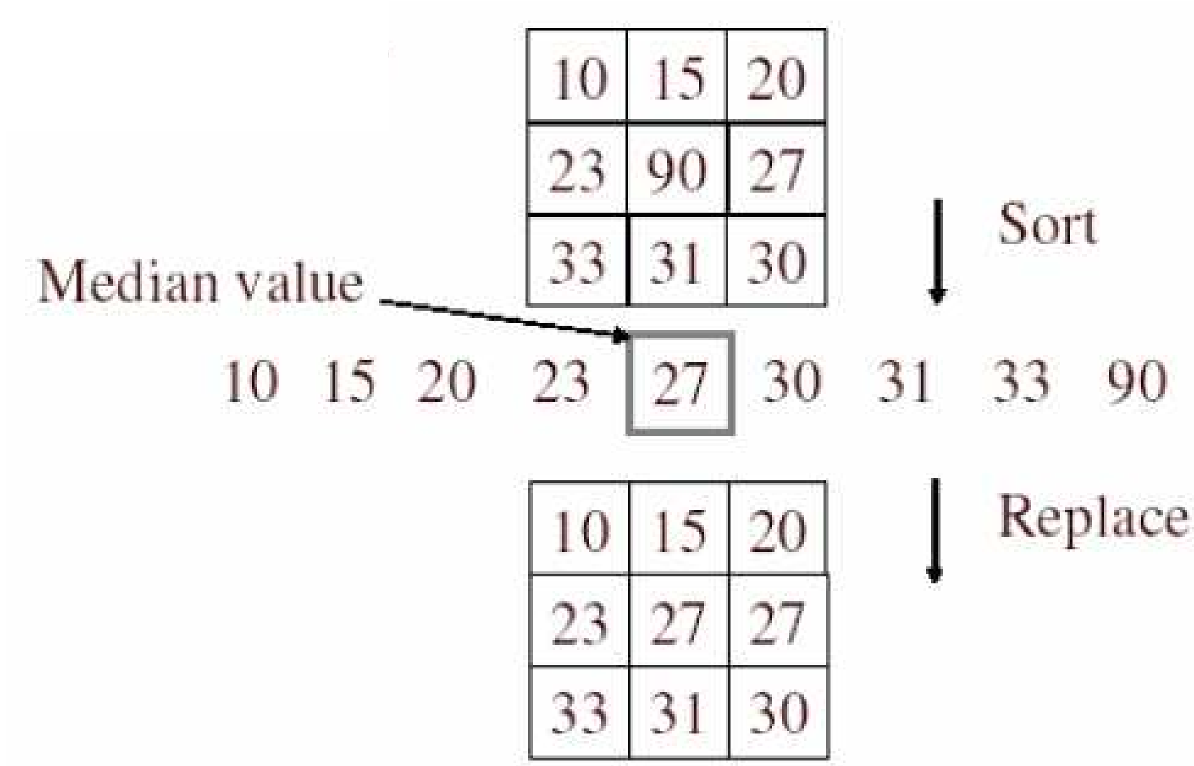
3x3

5x5

7x7

Alternative idea: Median filtering

A **median filter** operates over a window by selecting the median intensity in the window



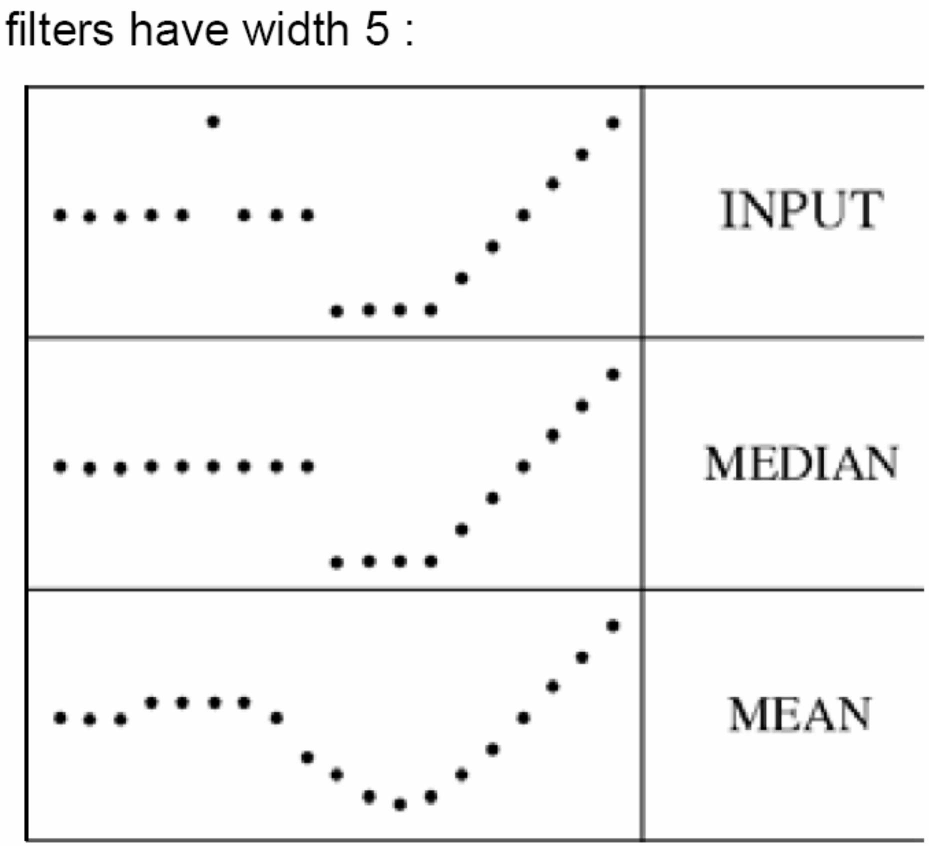
- Is median filtering linear?

Source: K. Grauman

Median filter

What advantage does median filtering have over Gaussian filtering?

- Robustness to outliers

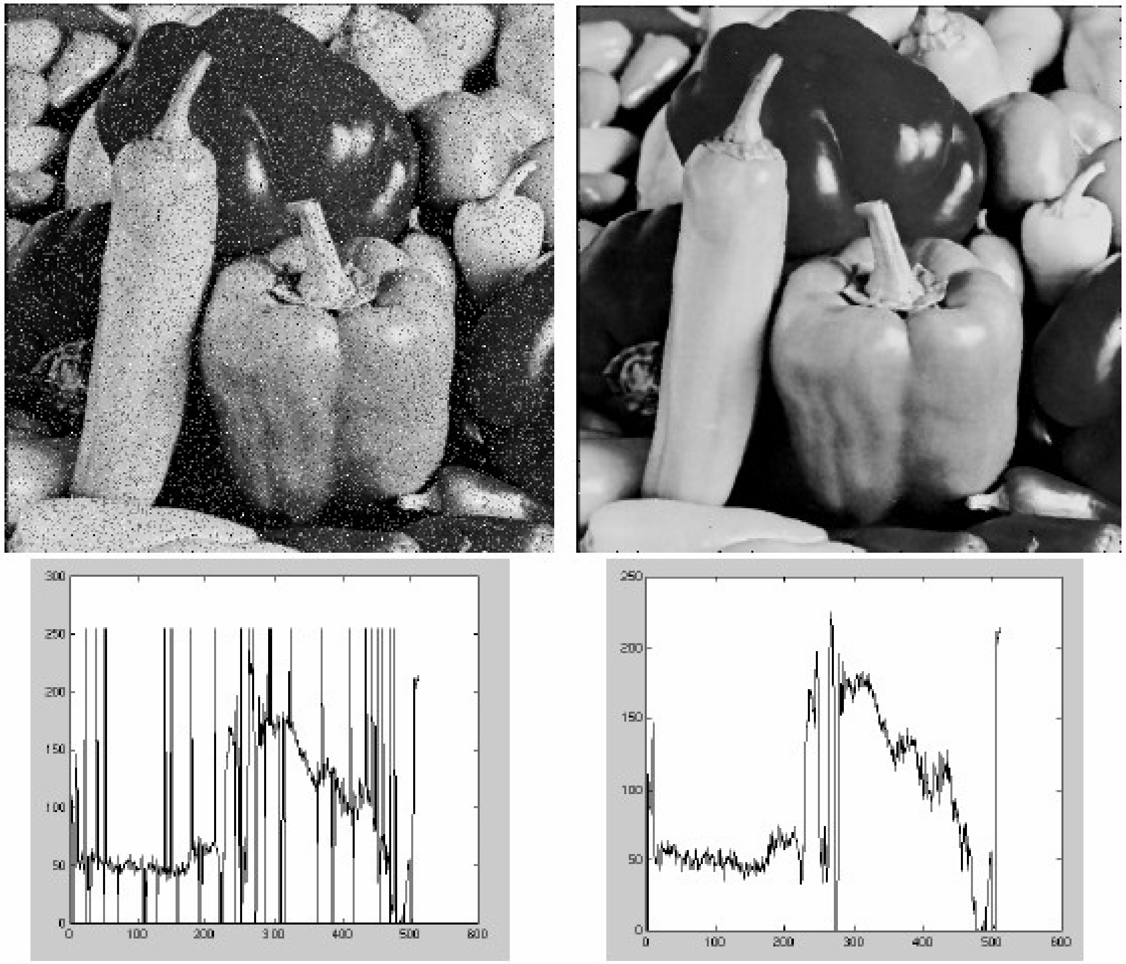


Source: K. Grauman

Median filter

Salt-and-pepper noise

Median filtered



MATLAB: medfilt2(image, [h w])

Source: M. Hebert

Median vs. Gaussian filtering

3x3 5x5 7x7

Gaussian

Median