Linear filtering

Subhransu Maji

CMPSCI 670: Computer Vision

September 27, 2016

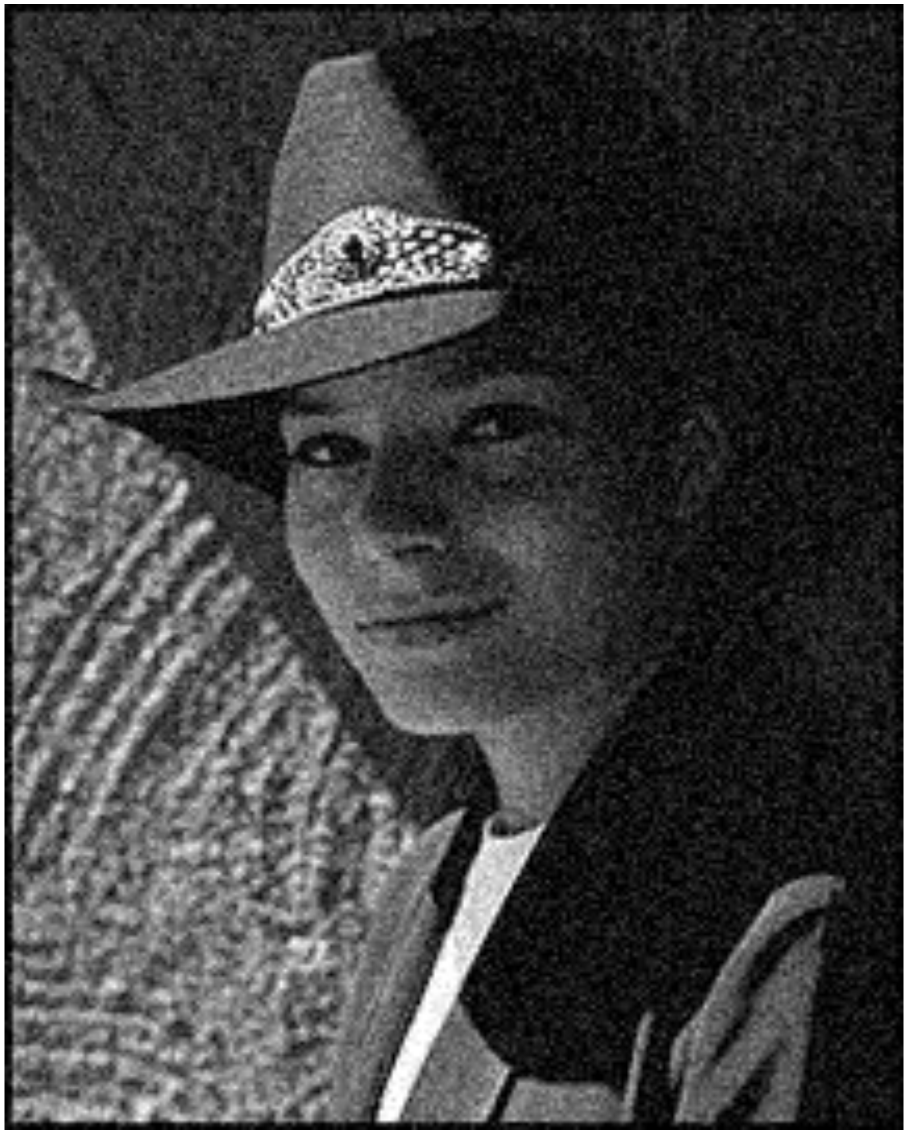
Slides credit: S. Lazebnik and others

Overview

- Linear filtering
  - Mathematical model
  - Implementation details
- Applications
  - De-noising
  - Sharpening
  - Edge detection
- Canny edge detector and recent advances

Motivation

- How can we reduce noise in a photograph?



Moving average

- Let's replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for the average of a 3x3 neighborhood?

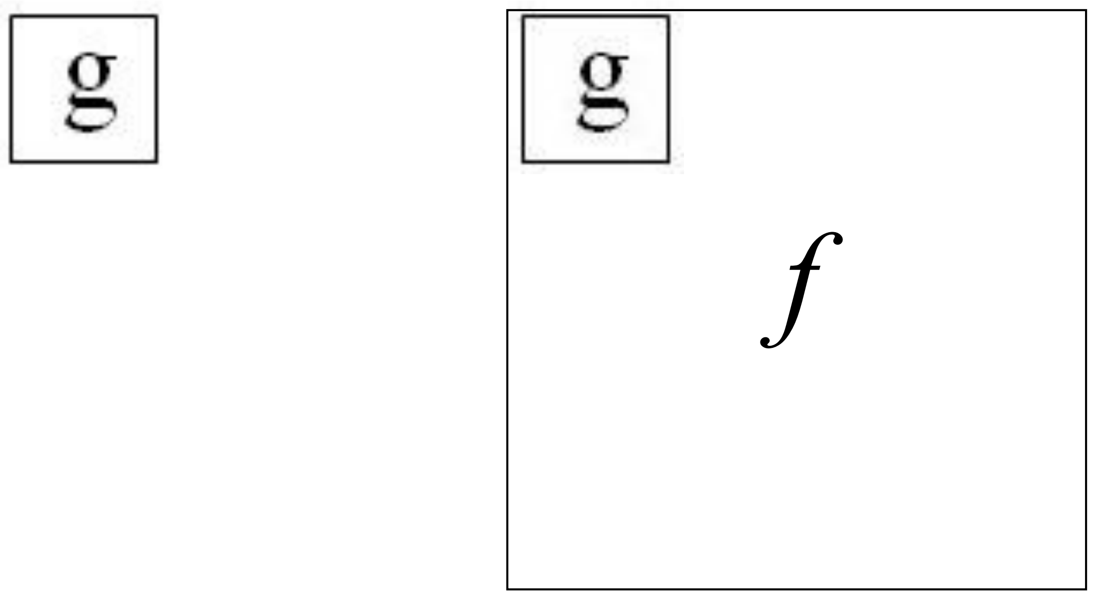
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

Filtering

- Let *f* be the image and *g* be the kernel. The output of filtering *f* with *g* denoted *f *g* is given by:

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



- Filtering computes the correlation between the *g* and *f* at each location
- Convolution is filtering with a flipped *g* (by notation)

Key properties

- **Linearity:** filter($f_1 + f_2$) = filter(f_1) + filter(f_2)
- **Shift invariance:** same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

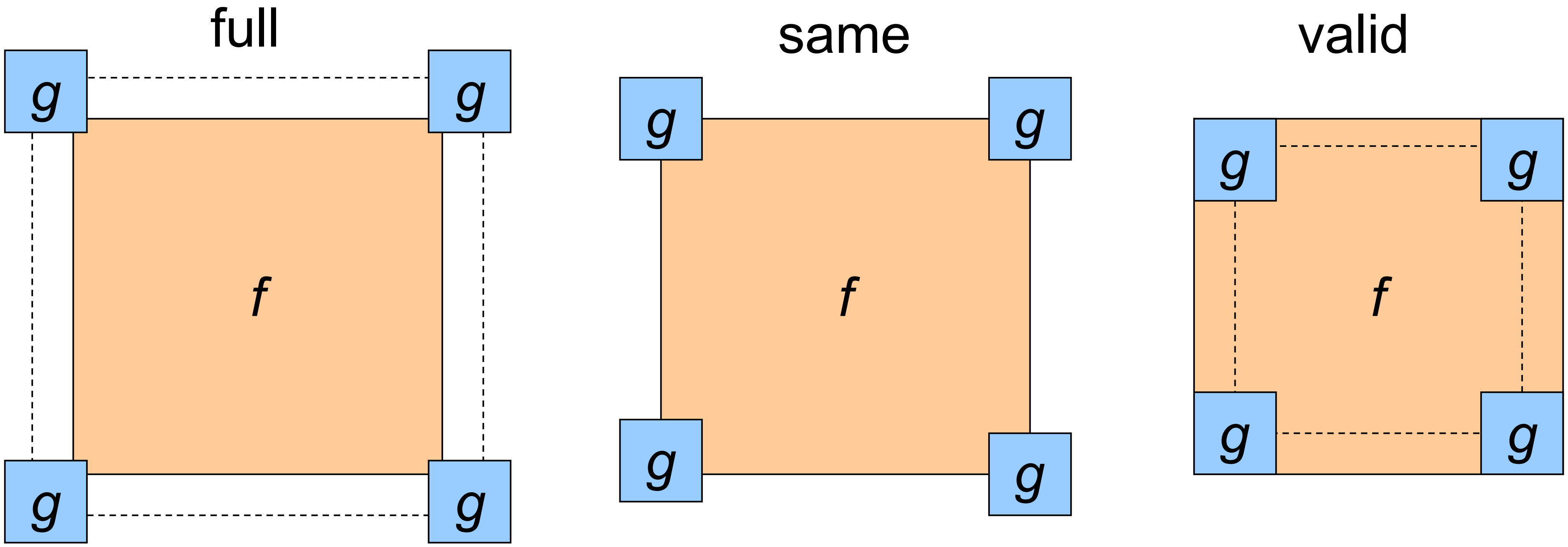
Properties in more detail

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Annoying details

What is the size of the output?

- MATLAB: filter2(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



Annoying details

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

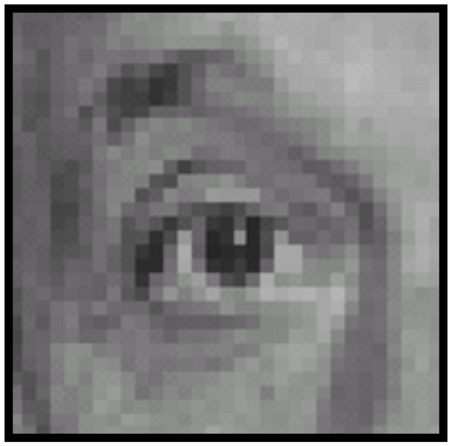
Annoying details

What about near the edge?

- the filter window falls off the edge of the image
- need to extrapolate
- 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

Practice with linear filters

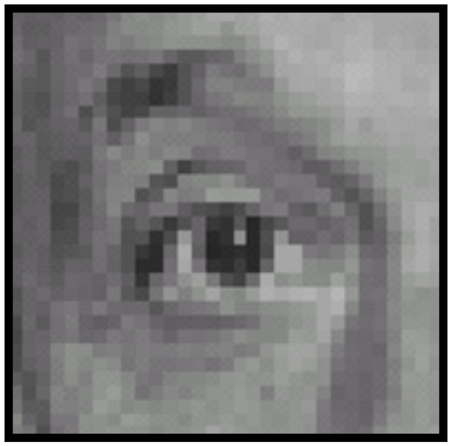

Original

0	0	0
0	1	0
0	0	0

?

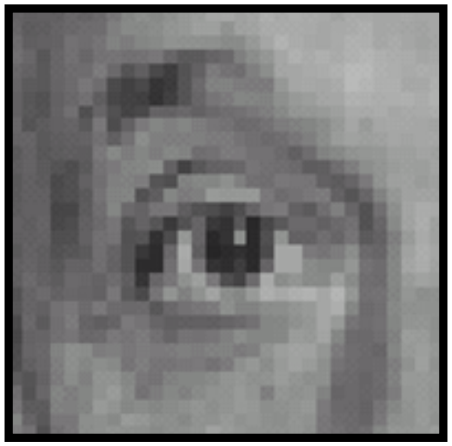
Source: D. Lowe

Practice with linear filters



Original

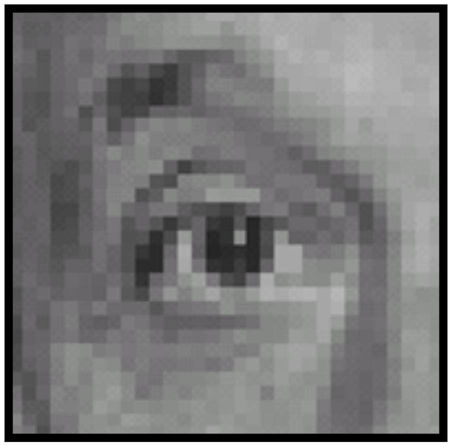
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



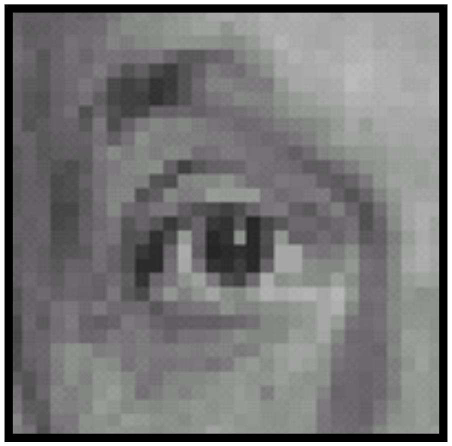
Original

0	0	0
0	0	1
0	0	0

?

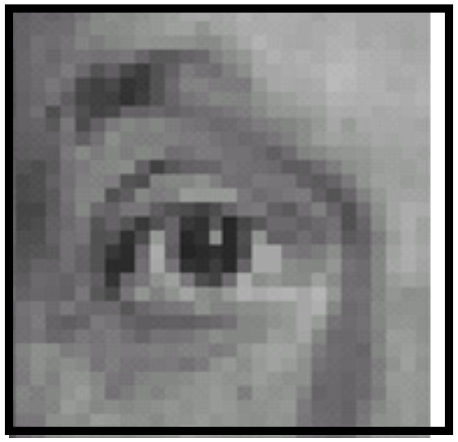
Source: D. Lowe

Practice with linear filters



Original

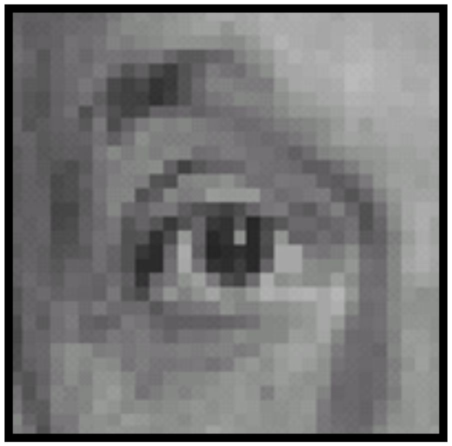
0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters



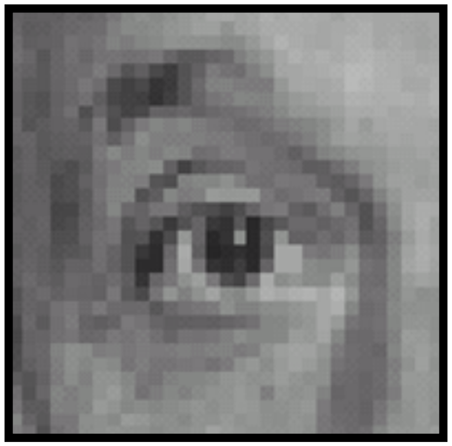
Original

$$\frac{1}{9}\ \boxed{\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}}$$

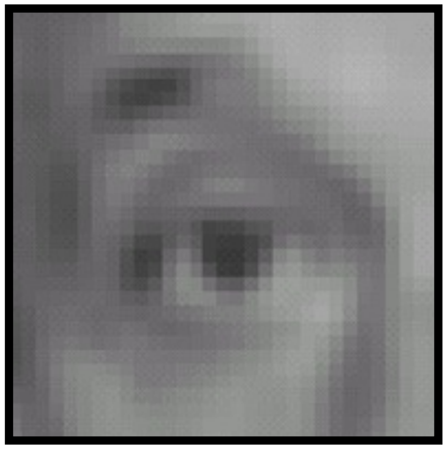
?

Source: D. Lowe

Practice with linear filters

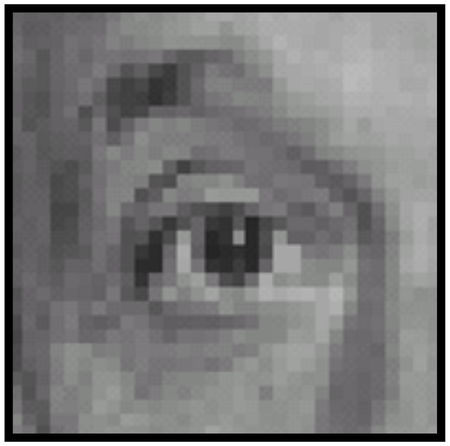

Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$


Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



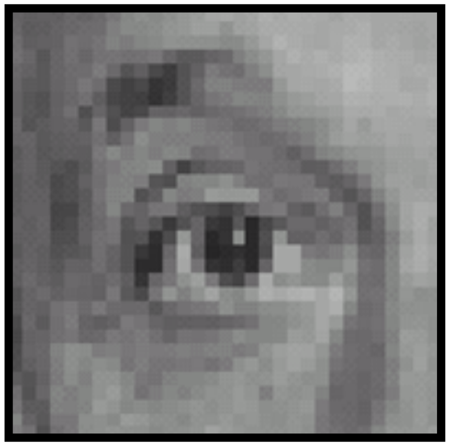
Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad ?$$

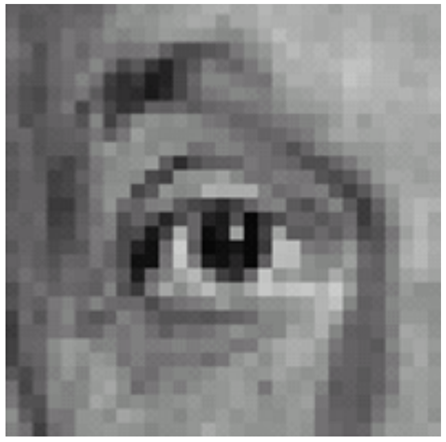
(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



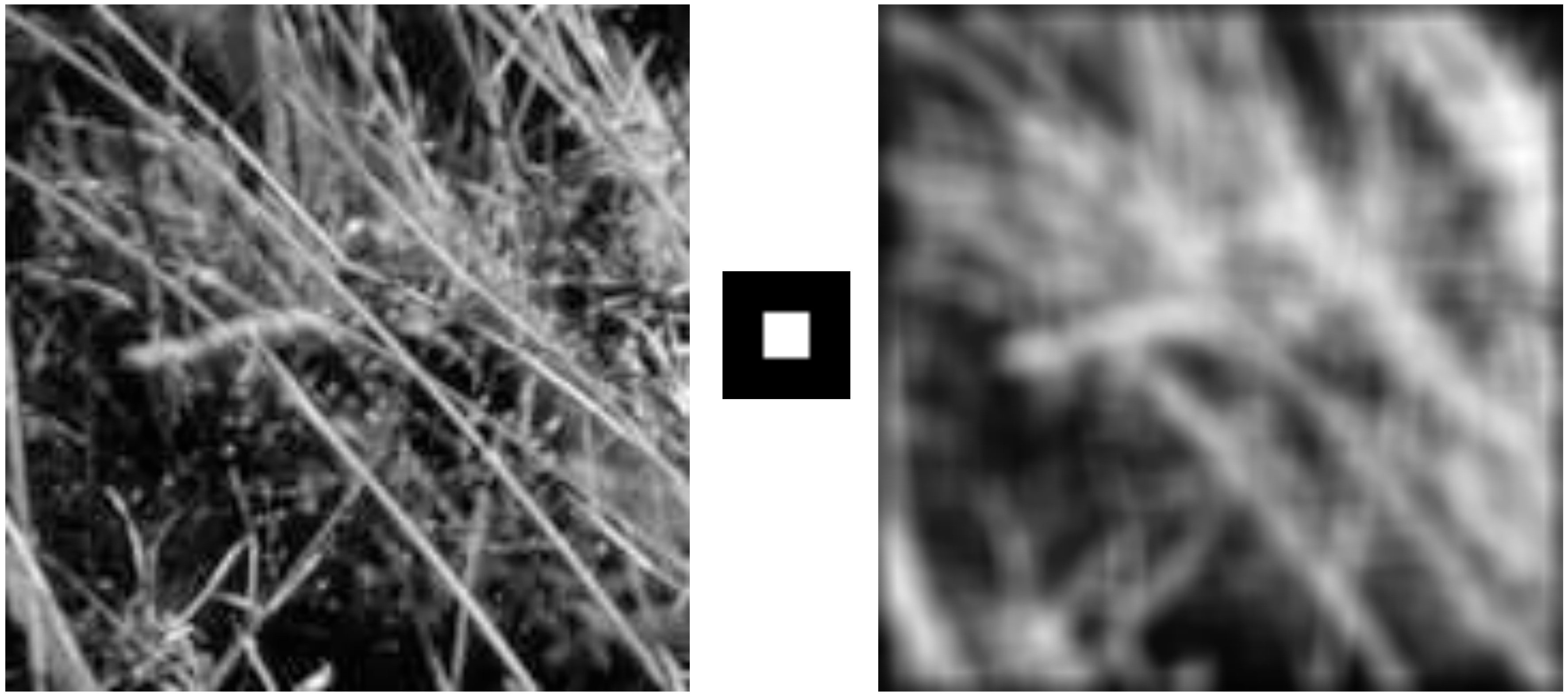
Original

Sharpening filter: accentuates differences with local average

Source: D. Lowe

Smoothing with box filter revisited

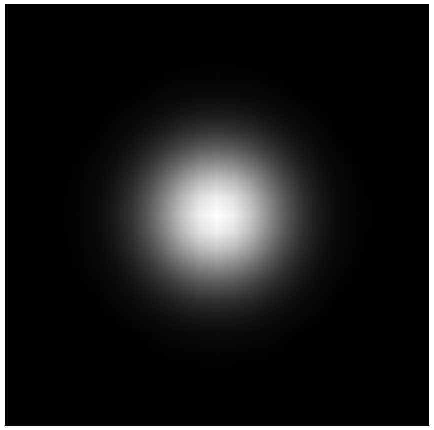
- What's wrong with this picture?
- What's the solution?



Source: D. Forsyth

Smoothing with box filter revisited

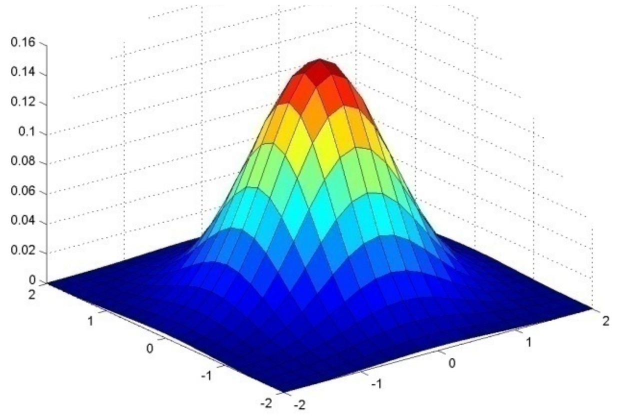
- What's wrong with this picture?
- What's the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

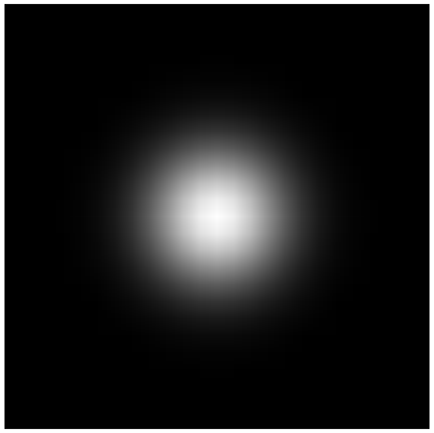


"fuzzy blob"

Gaussian Kernel

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





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

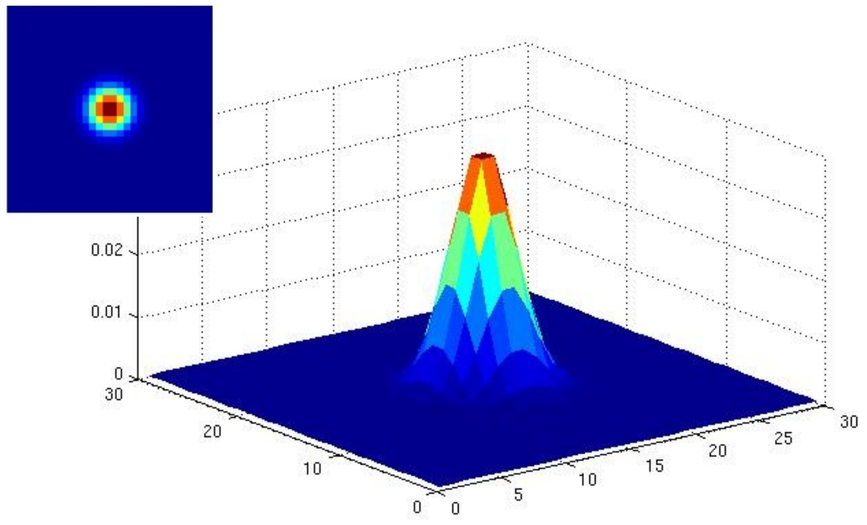
5 x 5, $\sigma = 1$

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should *renormalize* weights to sum to 1 in any case)

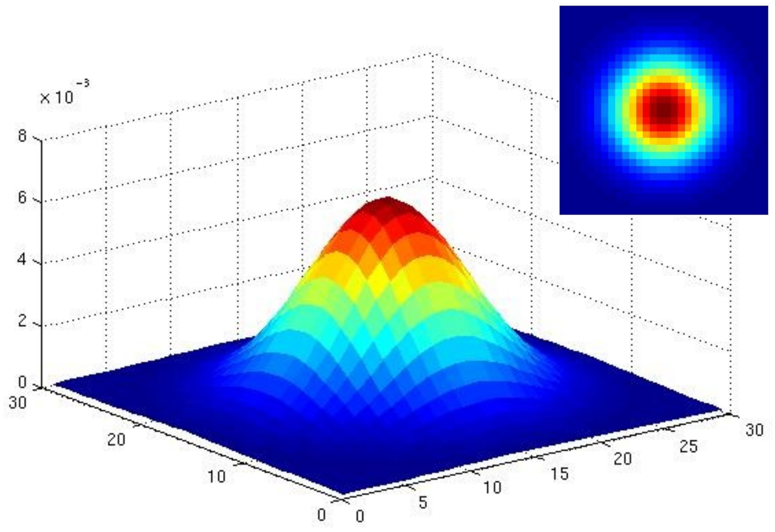
Source: C. Rasmussen

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

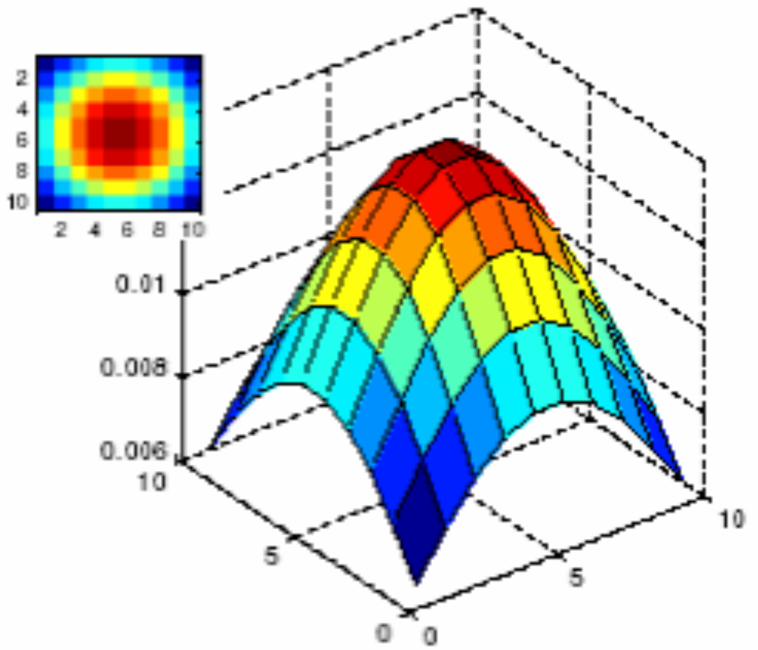


σ = 5 with 30 x 30 kernel

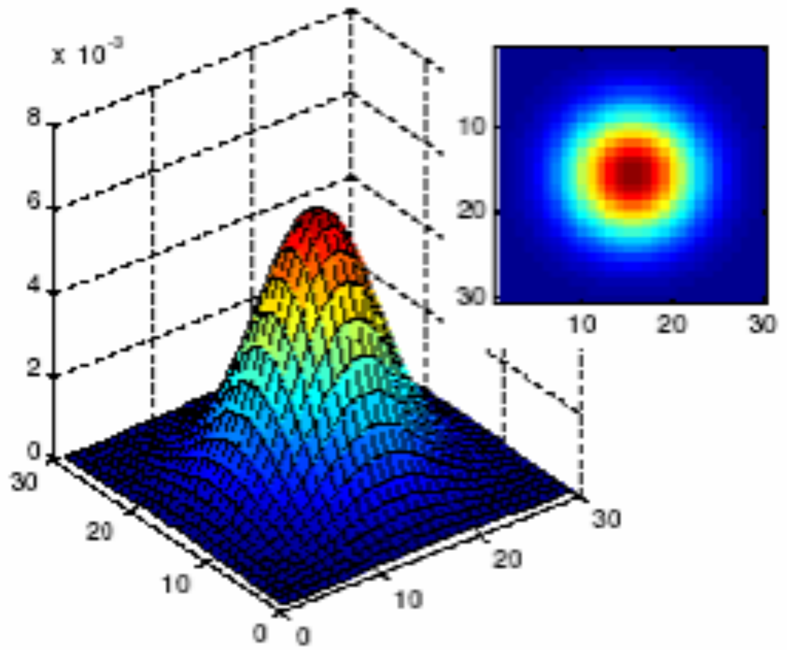
- Standard deviation σ: determines extent of smoothing

Choosing kernel width

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



σ = 5 with 10x10 kernel

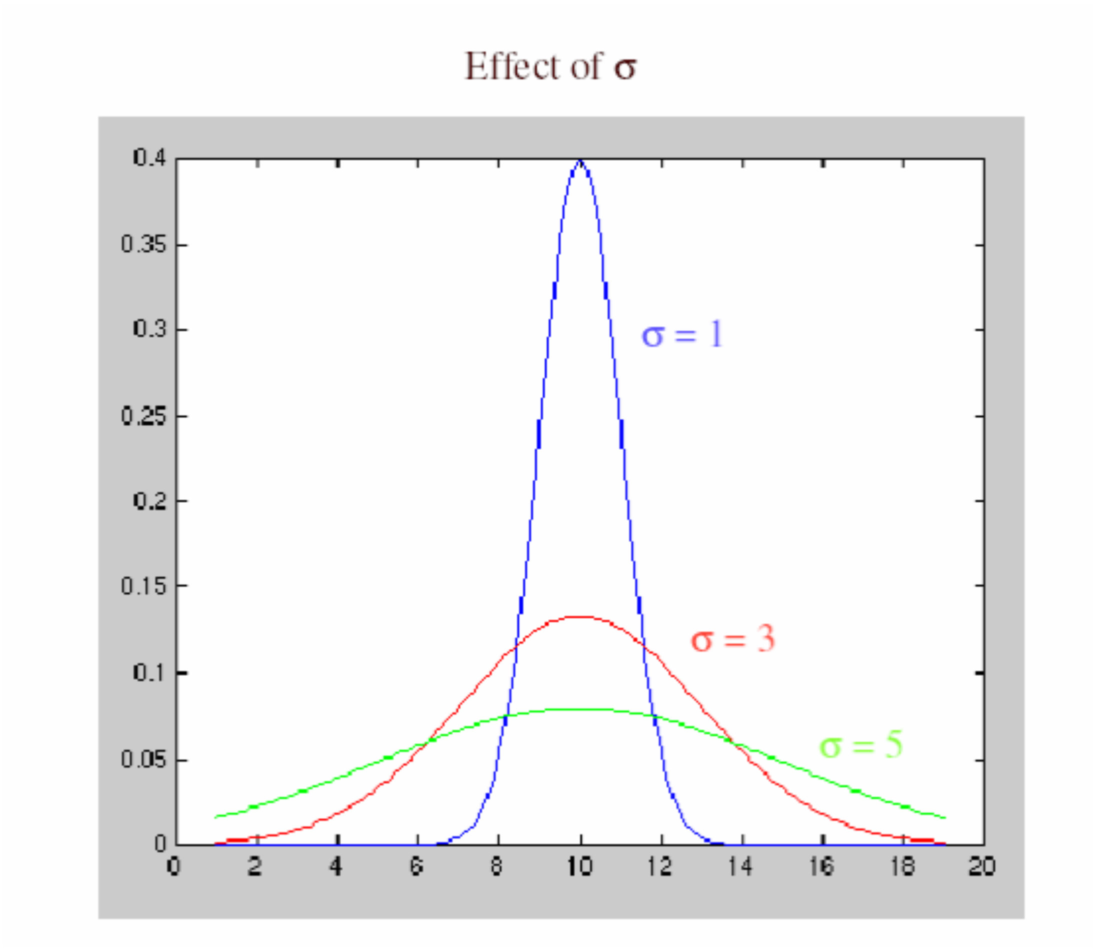


σ = 5 with 30x30 kernel

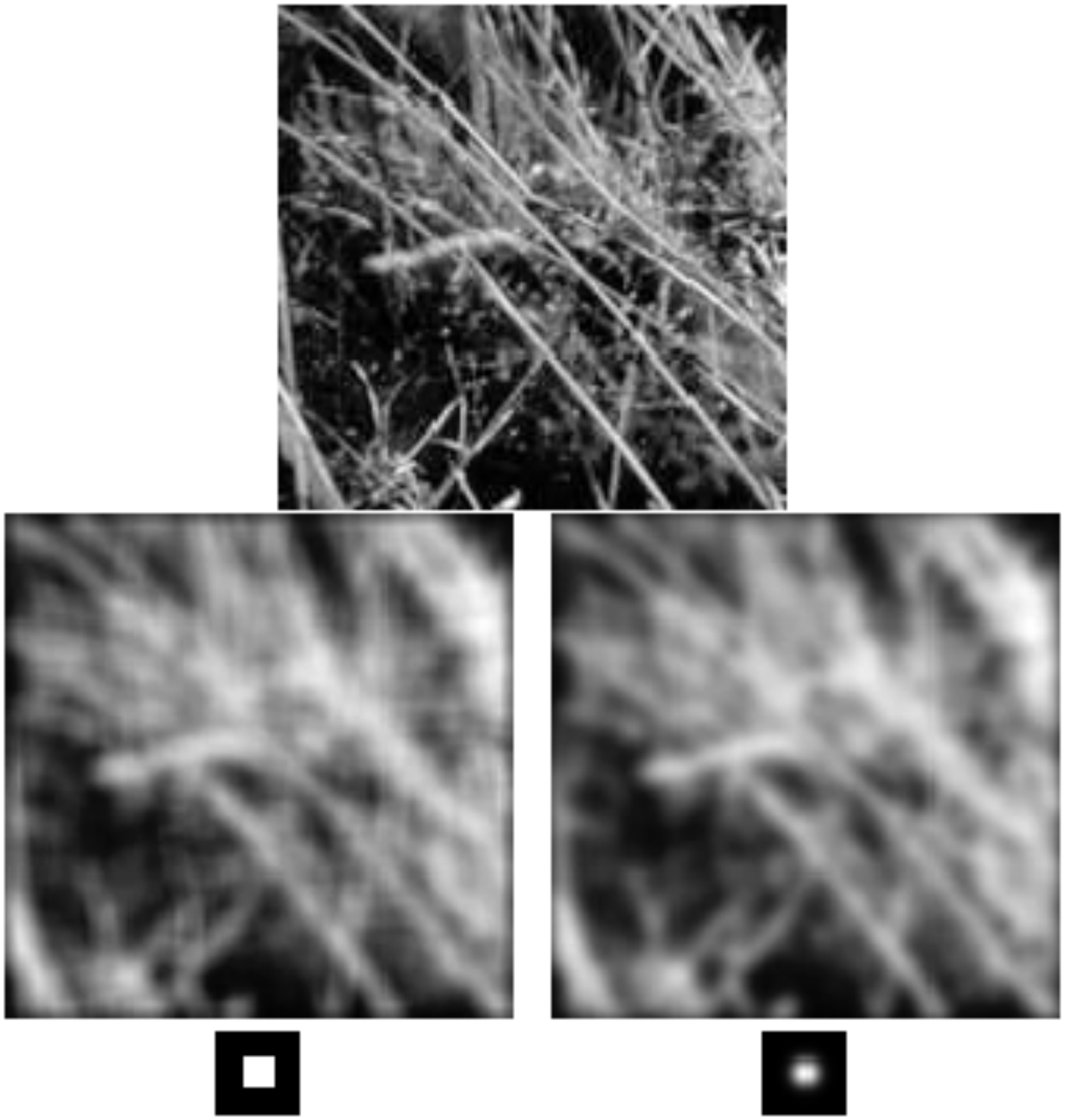
Source: K. Grauman

Choosing kernel width

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



Gaussian vs. box filtering



Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example:

$$\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} = \begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

Source: K. Grauman

Separability of the Gaussian filter

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

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

The 2D Gaussian can be expressed as the product of two functions, one a function of *x* and the other a function of *y*

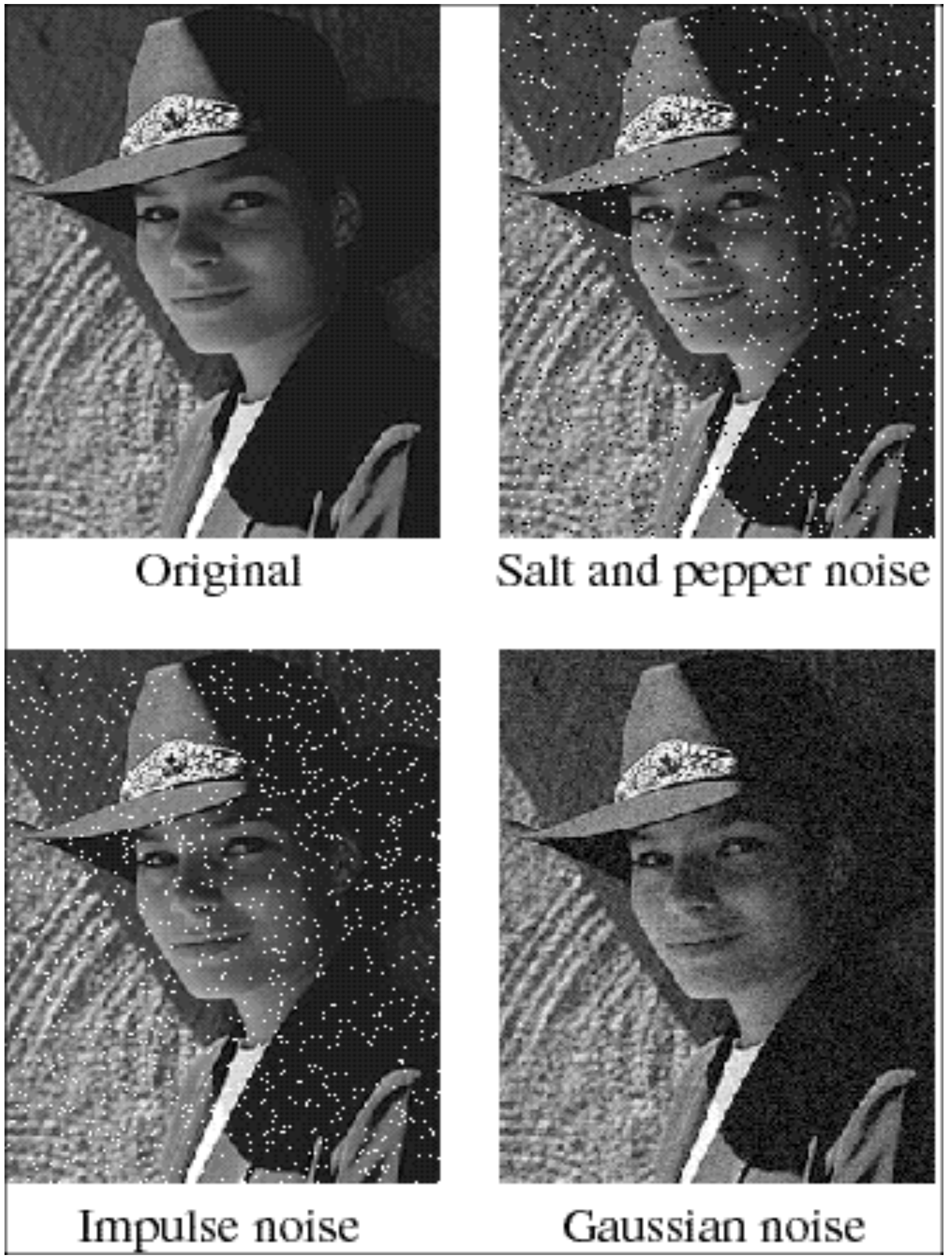
In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Types of noise

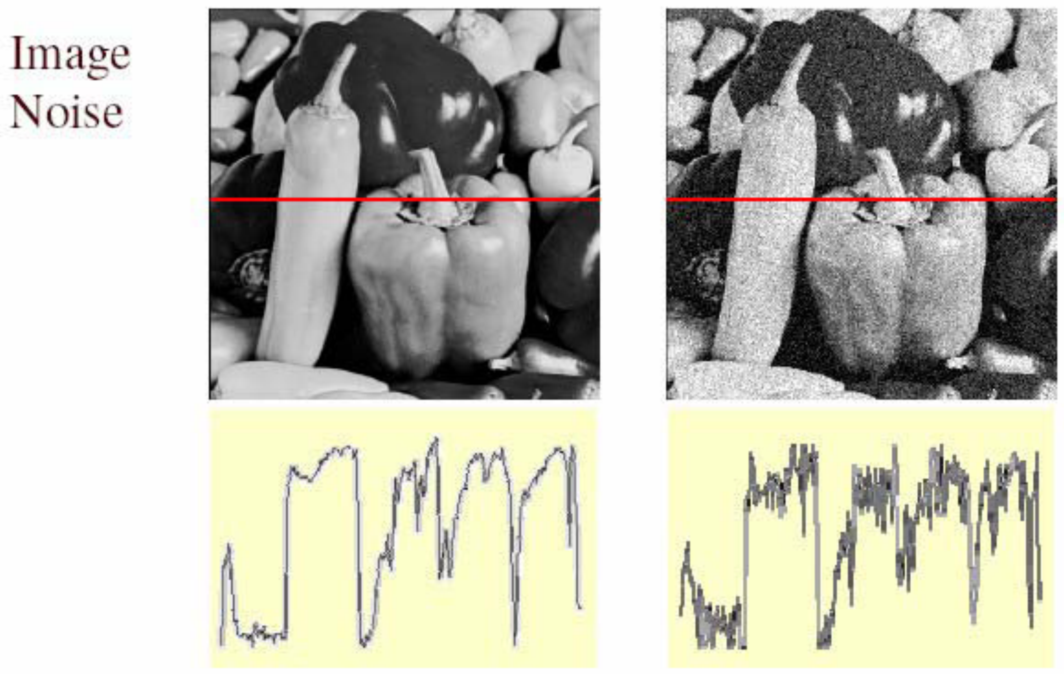


- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

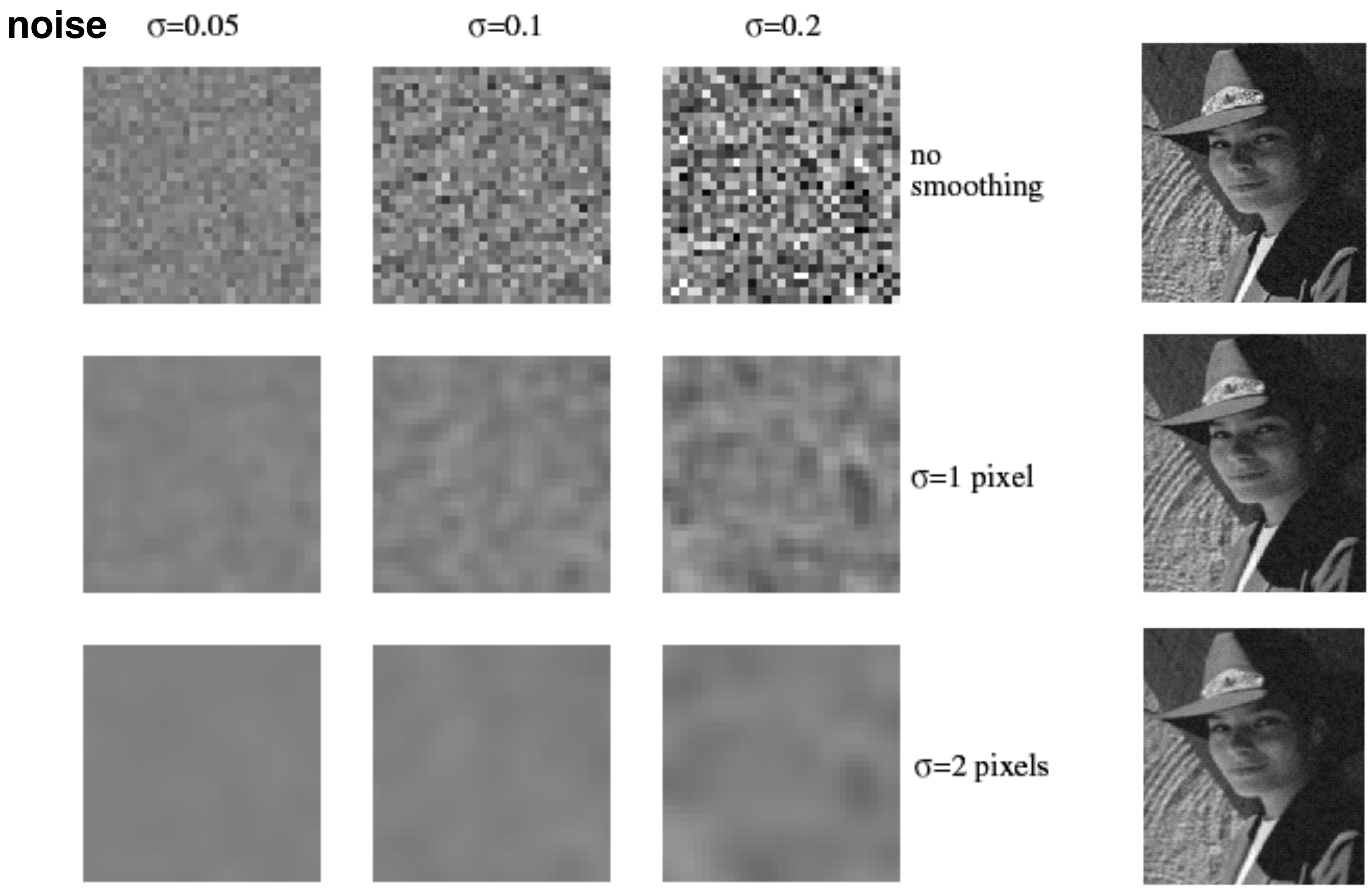


$$f(x,y) = \overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

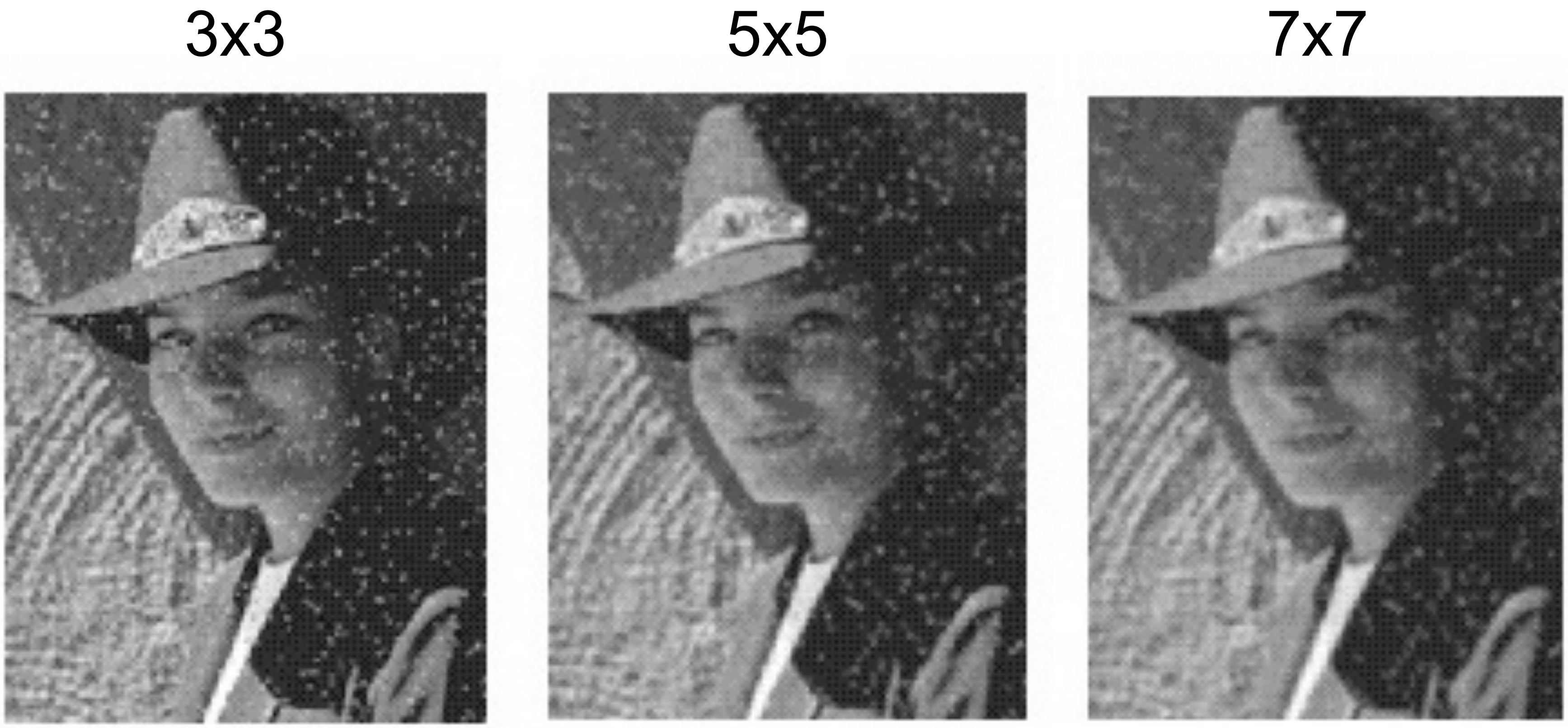
Source: M. Hebert

Reducing Gaussian noise



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

Reducing salt-and-pepper noise

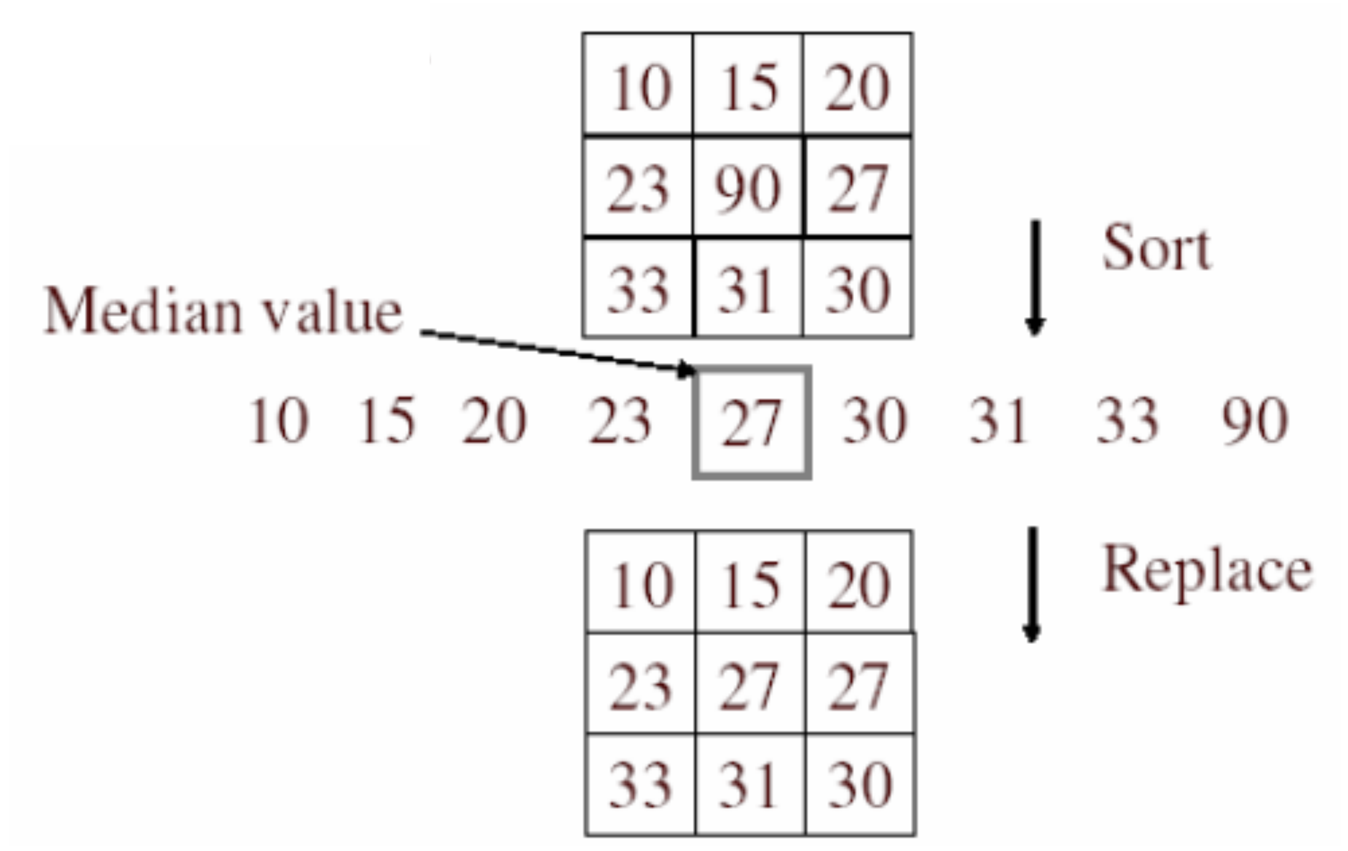


Gaussian smoothing with increasing standard deviation

What is wrong with these results?

Alternative idea: Median filtering

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



Question: is median filtering linear?

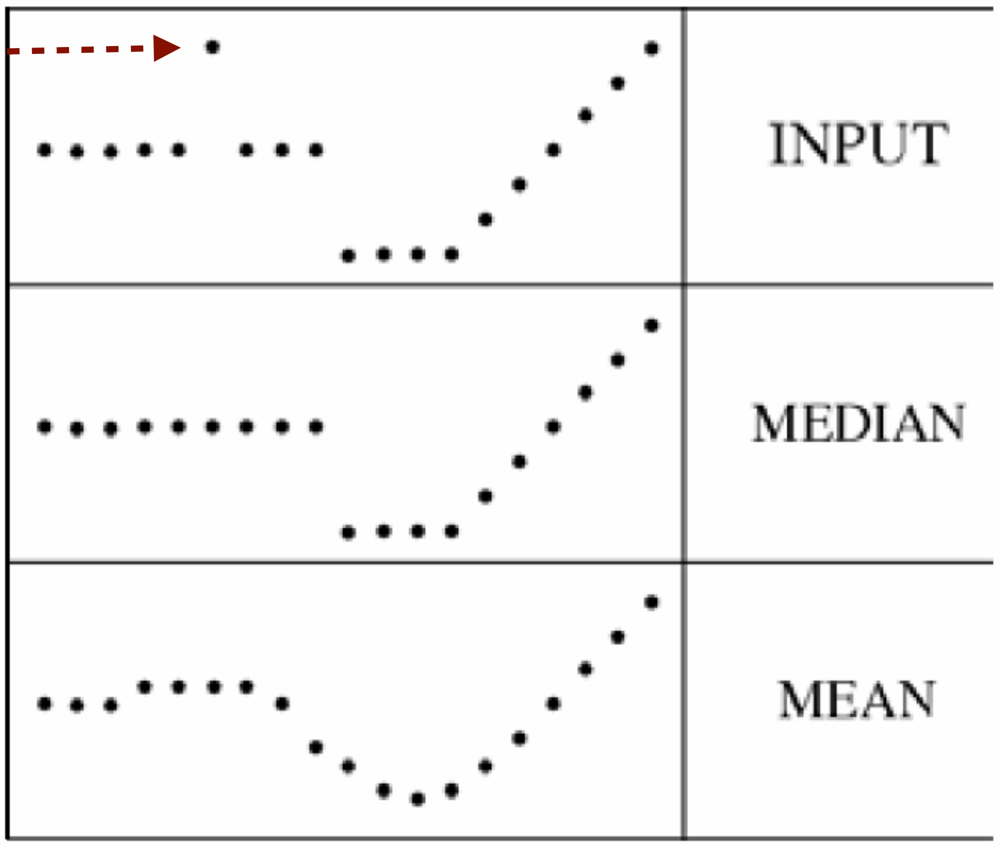
Source: K. Grauman

Median filter

- What advantage does median filtering have over Gaussian filtering?

Robustness to outliers

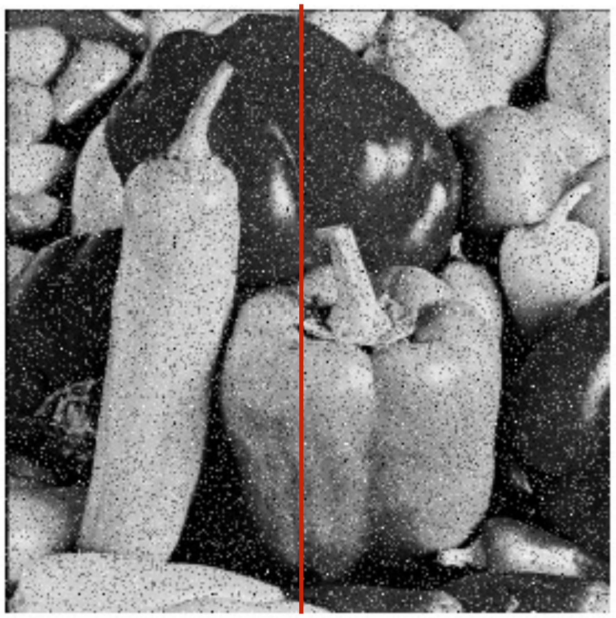
filters have width 5 :

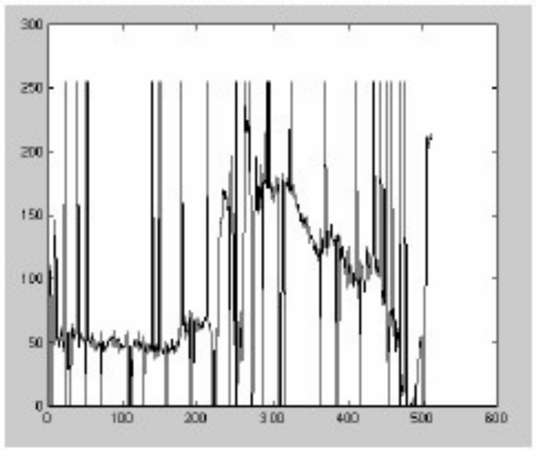


Source: K. Grauman

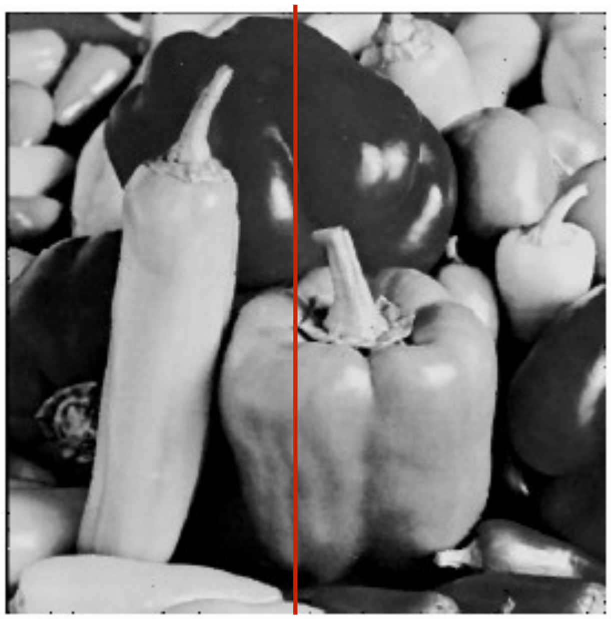
Median filter

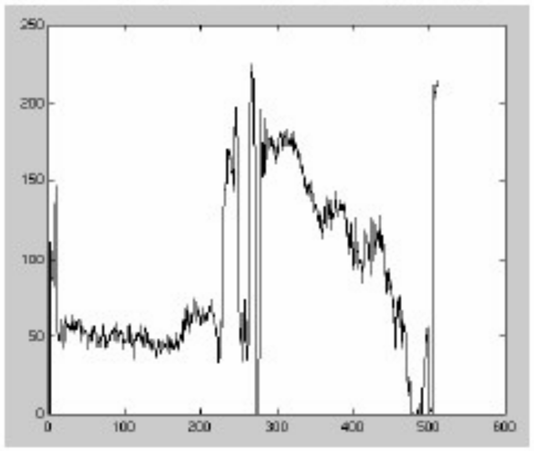
Salt-and-pepper noise





Median filtered





MATLAB: medfilt2(image, [h w])

Source: M. Hebert

Linear filtering

Subhransu Maji

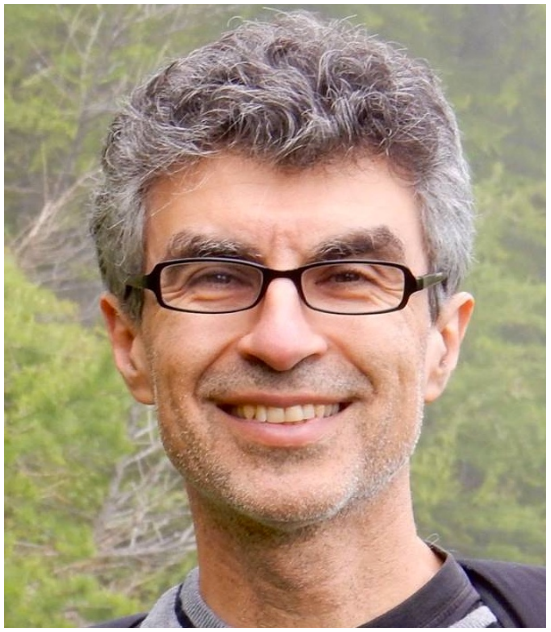
CMPSCI 670: Computer Vision

September 29, 2016

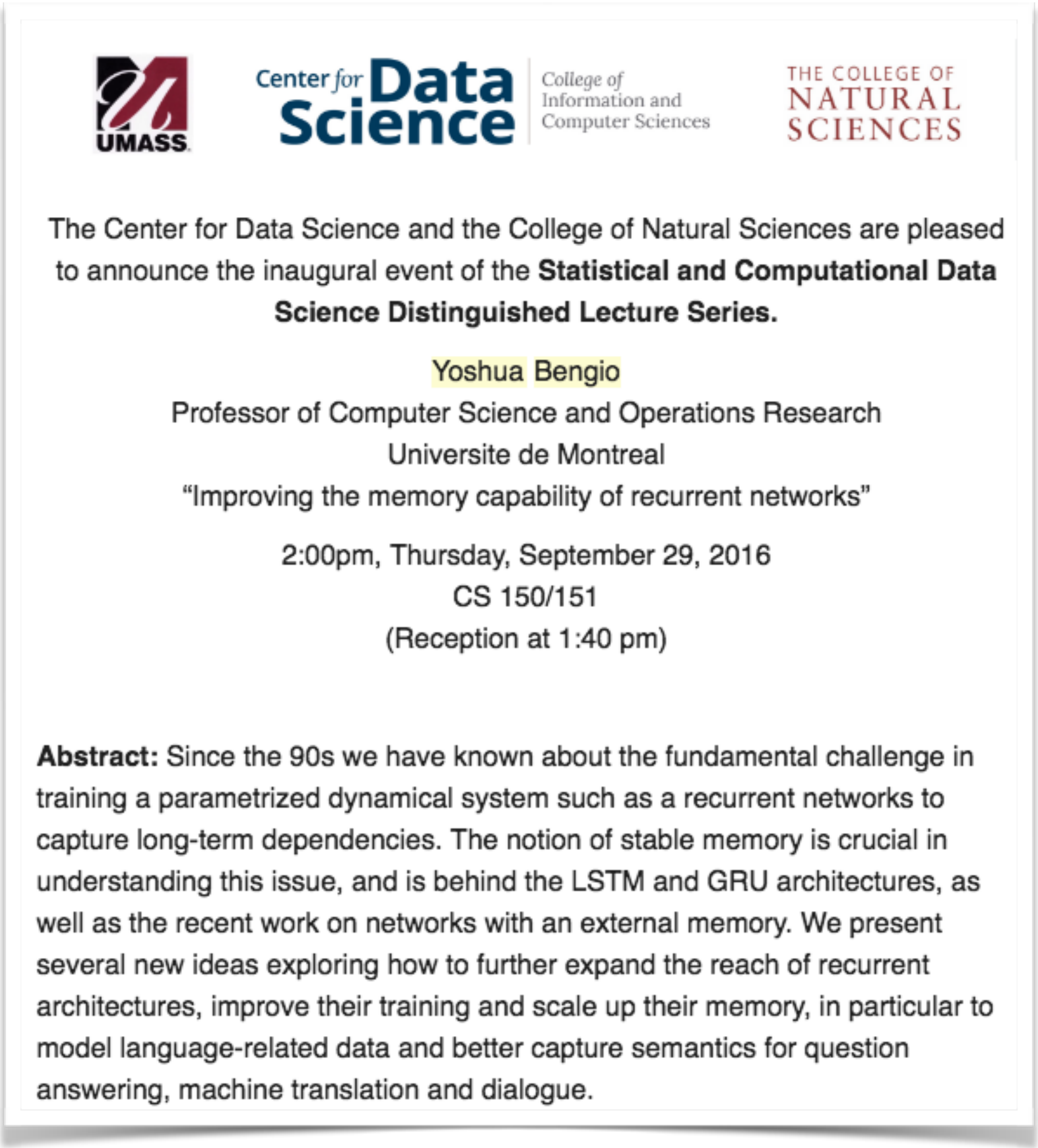
Slides credit: S. Lazebnik and others

Announcements

- Today's lecture ends at 1:55pm
  - Encourage you to attend Yoshua Bengio's talk @ 2



- Administrivia
  - Mini-project 1 due today
  - Mini-project 2 will be posted later today (due Oct 14)



UMASS | Center for Data Science | College of Information and Computer Sciences | THE COLLEGE OF NATURAL SCIENCES

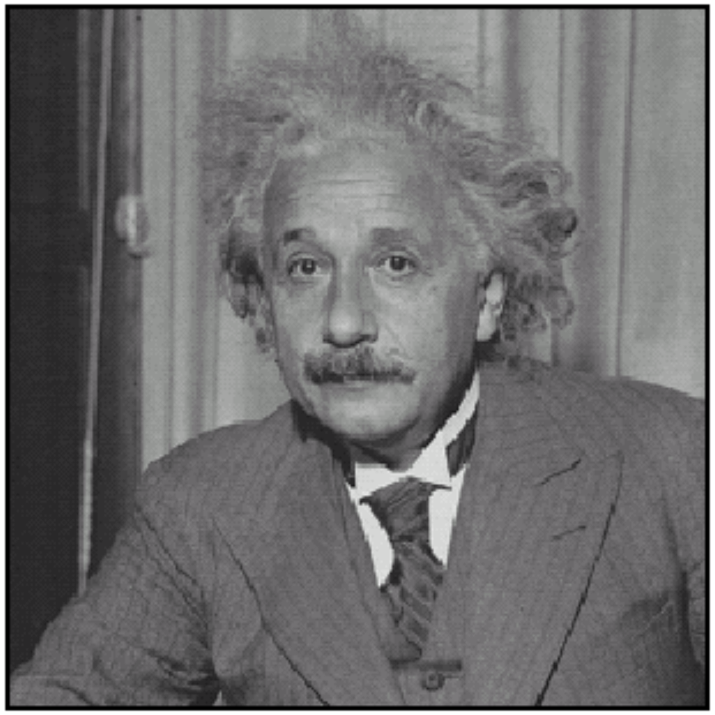
The Center for Data Science and the College of Natural Sciences are pleased to announce the inaugural event of the **Statistical and Computational Data Science Distinguished Lecture Series.**

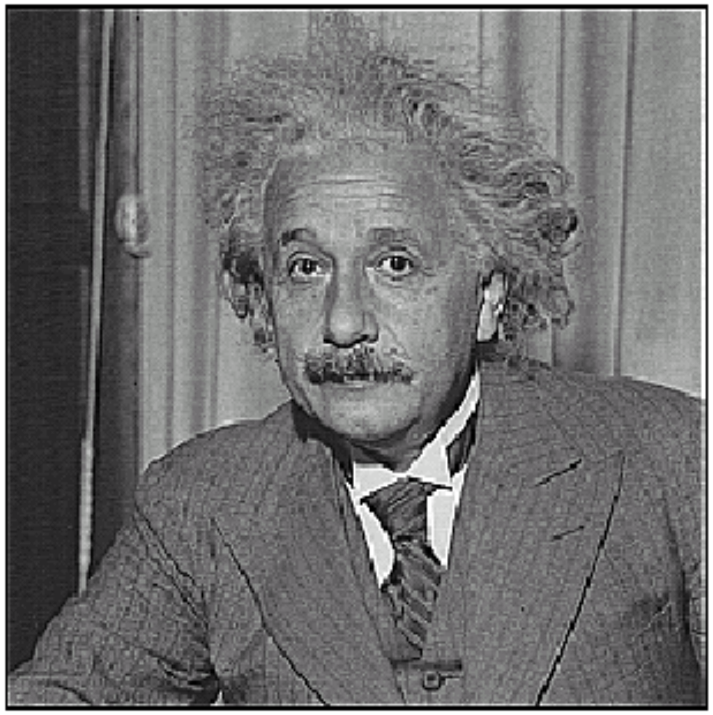
Yoshua Bengio
Professor of Computer Science and Operations Research
Universite de Montreal
"Improving the memory capability of recurrent networks"

2:00pm, Thursday, September 29, 2016
CS 150/151
(Reception at 1:40 pm)

Abstract: Since the 90s we have known about the fundamental challenge in training a parametrized dynamical system such as a recurrent networks to capture long-term dependencies. The notion of stable memory is crucial in understanding this issue, and is behind the LSTM and GRU architectures, as well as the recent work on networks with an external memory. We present several new ideas exploring how to further expand the reach of recurrent architectures, improve their training and scale up their memory, in particular to model language-related data and better capture semantics for question answering, machine translation and dialogue.

Sharpening

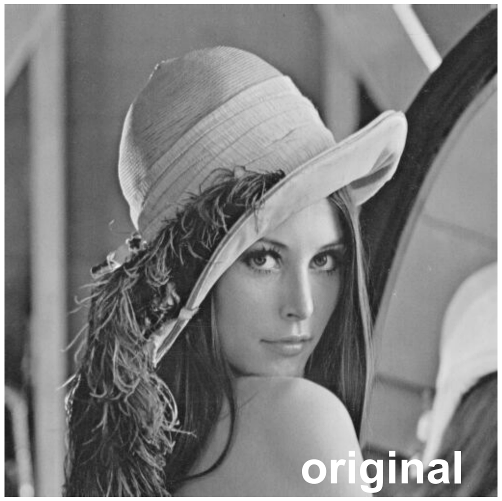

before


after

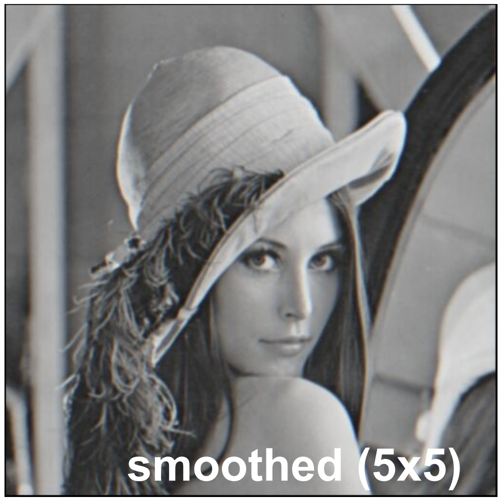
Source: D. Lowe

Sharpening

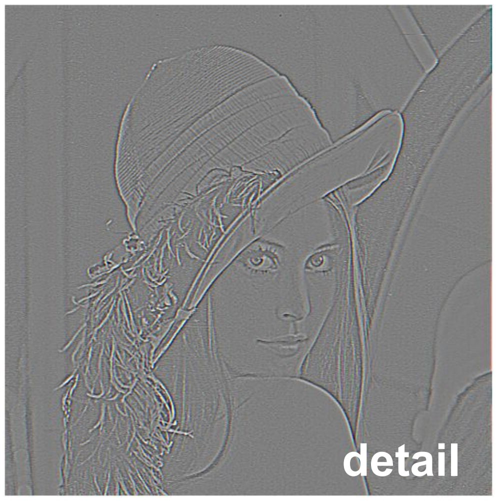
What does blurring take away?



–

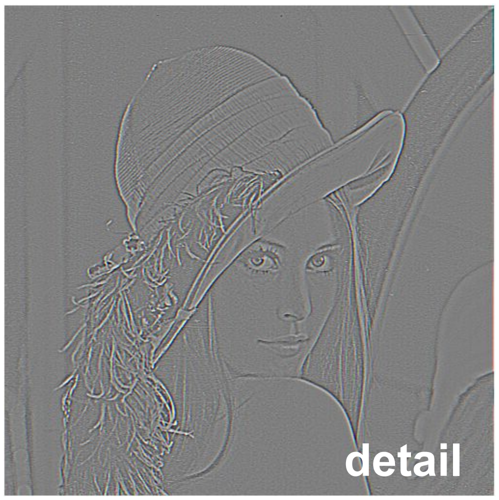


=

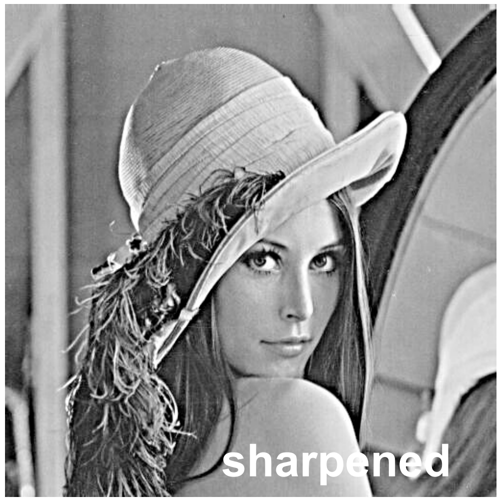


Let’s add it back:

$+ \alpha$

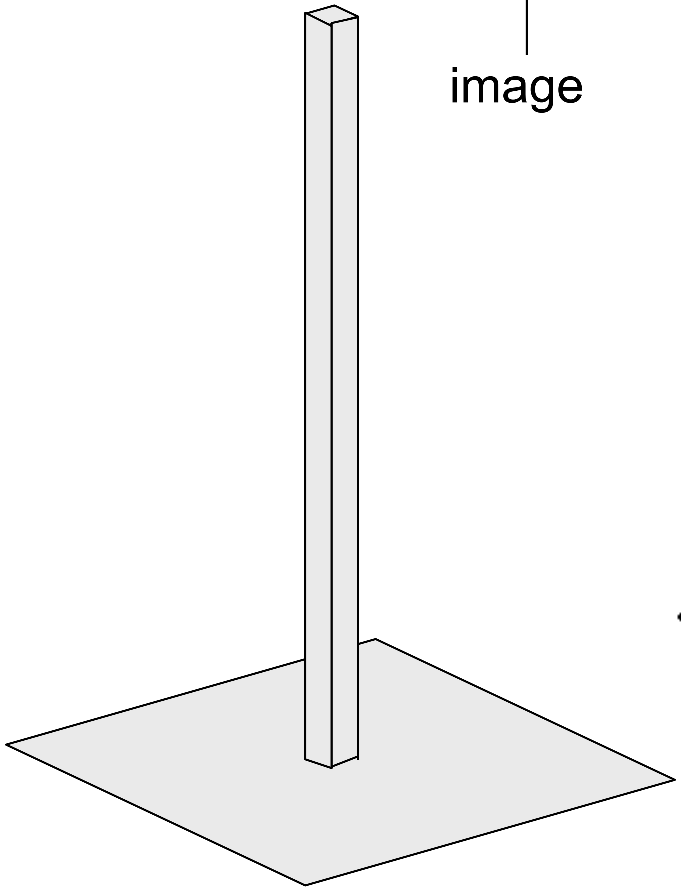


=

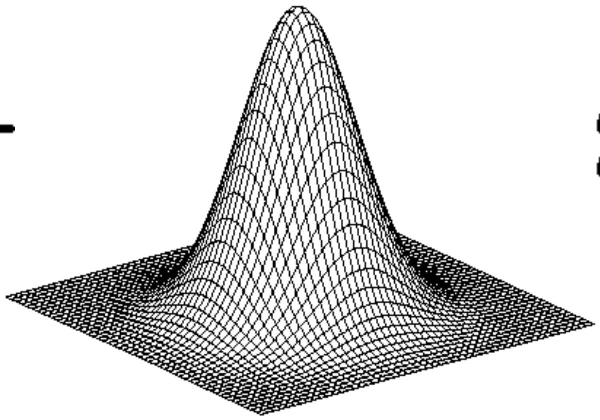


Unsharp mask filter

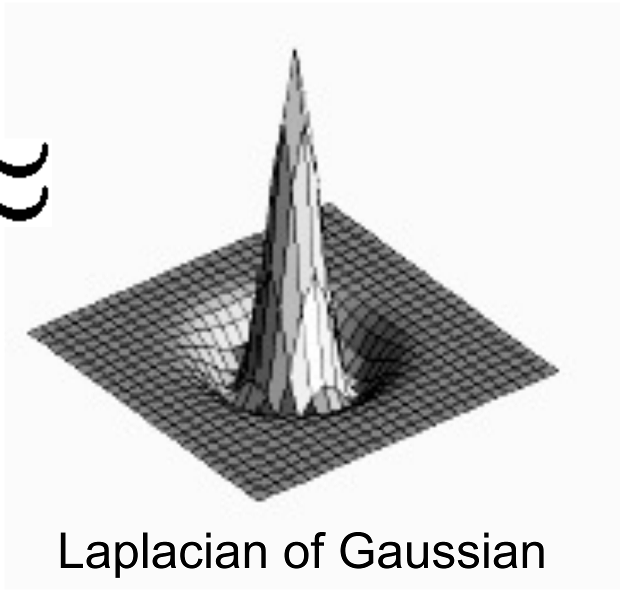
$$\underset{\text{image}}{f} + \alpha(f - \underset{\substack{\text{blurred}\\\text{image}}}{f * g}) = (1+\alpha)f - \alpha f * g = f * ((1+\alpha)\underset{\substack{\text{unit impulse}\\\text{(identity)}}}{e} - g)$$


unit impulse

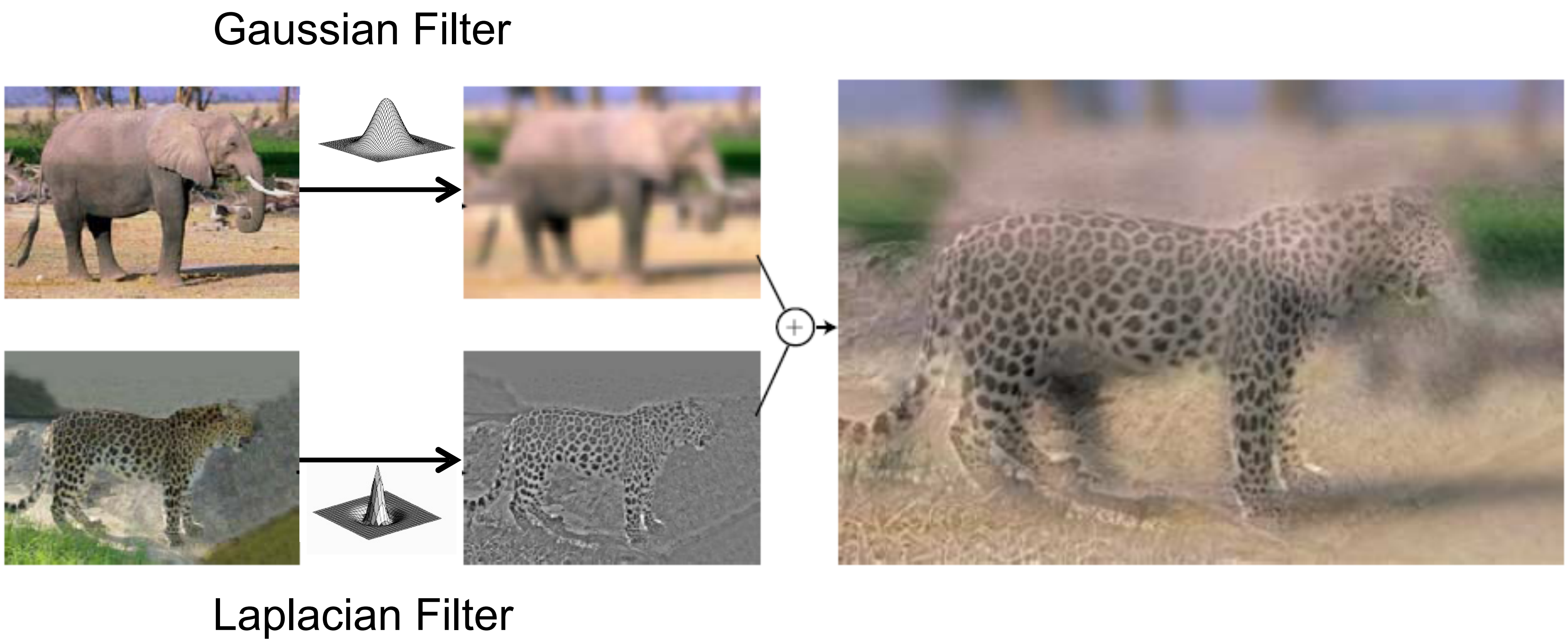
$-$


Gaussian

$\approx$


Laplacian of Gaussian

Hybrid Images

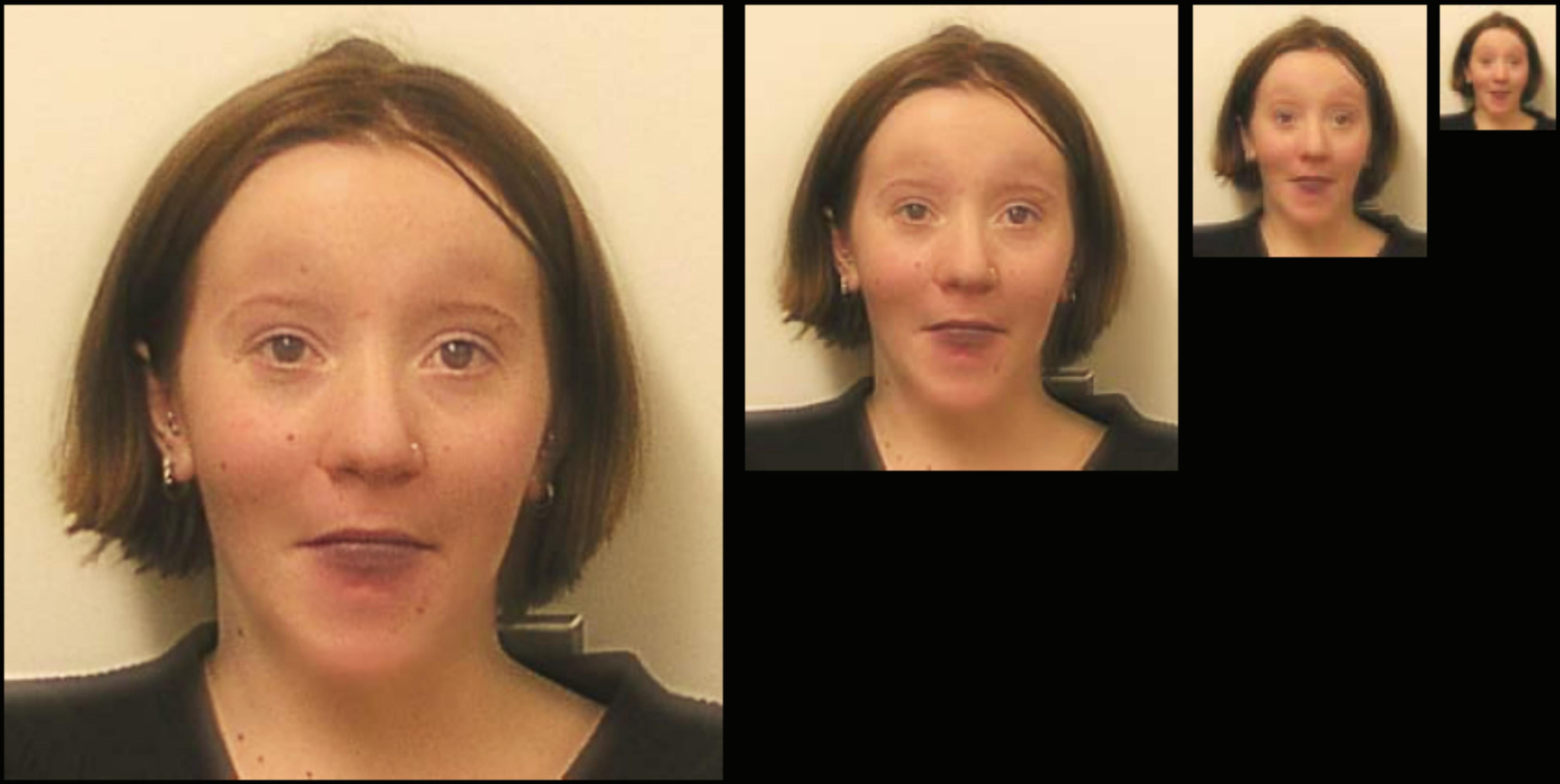


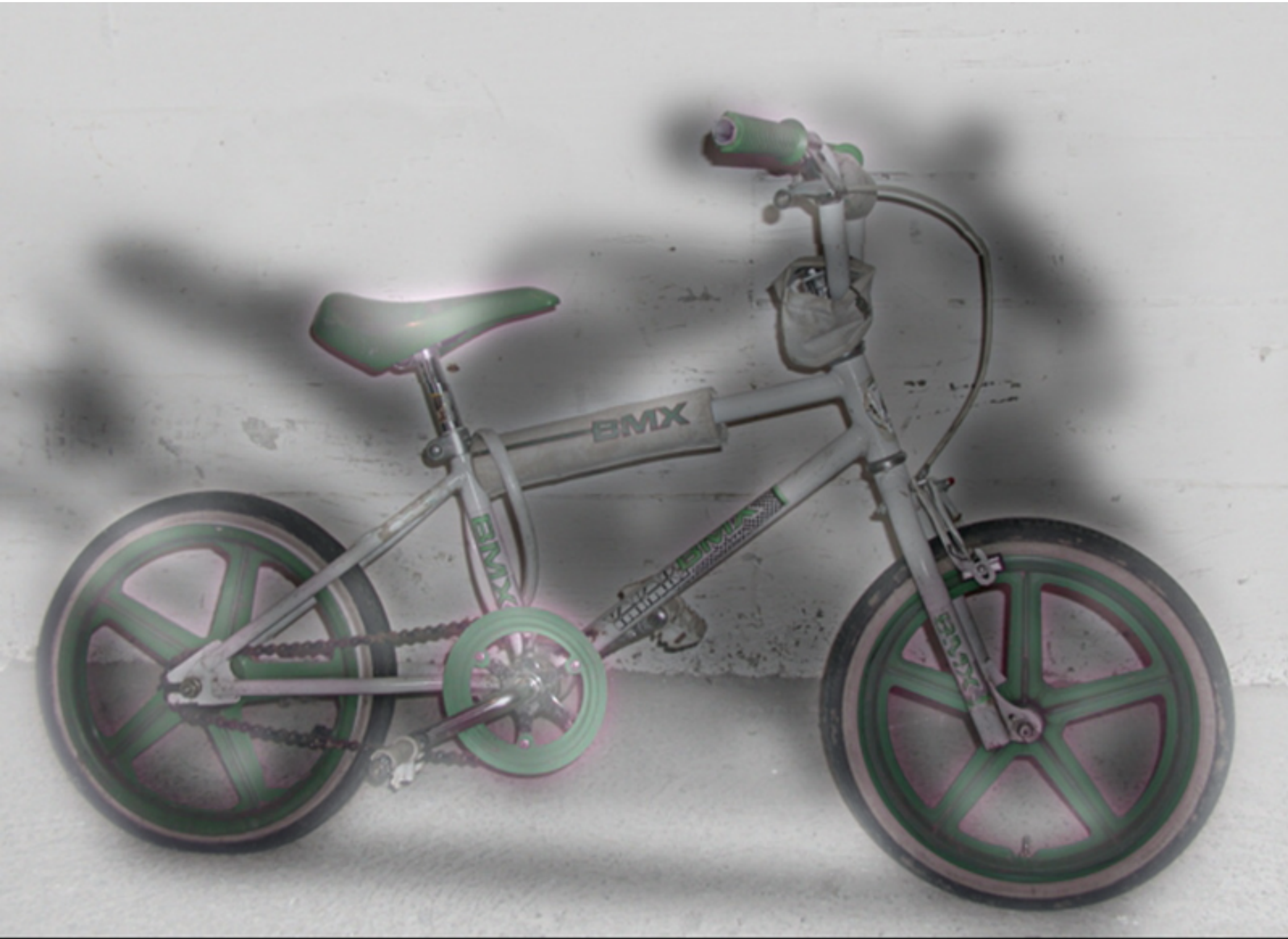
A. Oliva, A. Torralba, P.G. Schyns, “Hybrid Images,” SIGGRAPH 2006

Changing expression

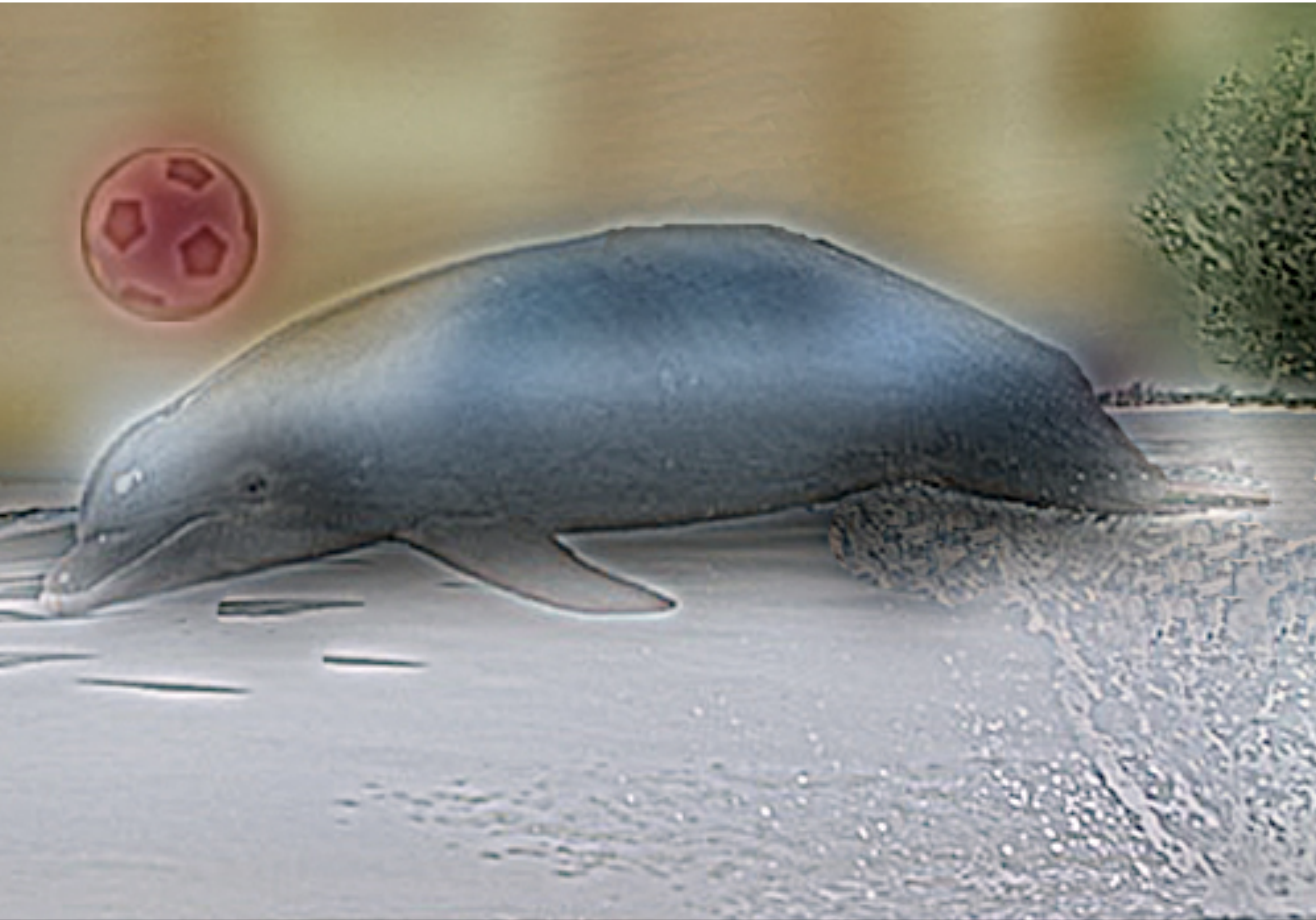
SIGGRAPH2006

Sad ⟷ Surprised





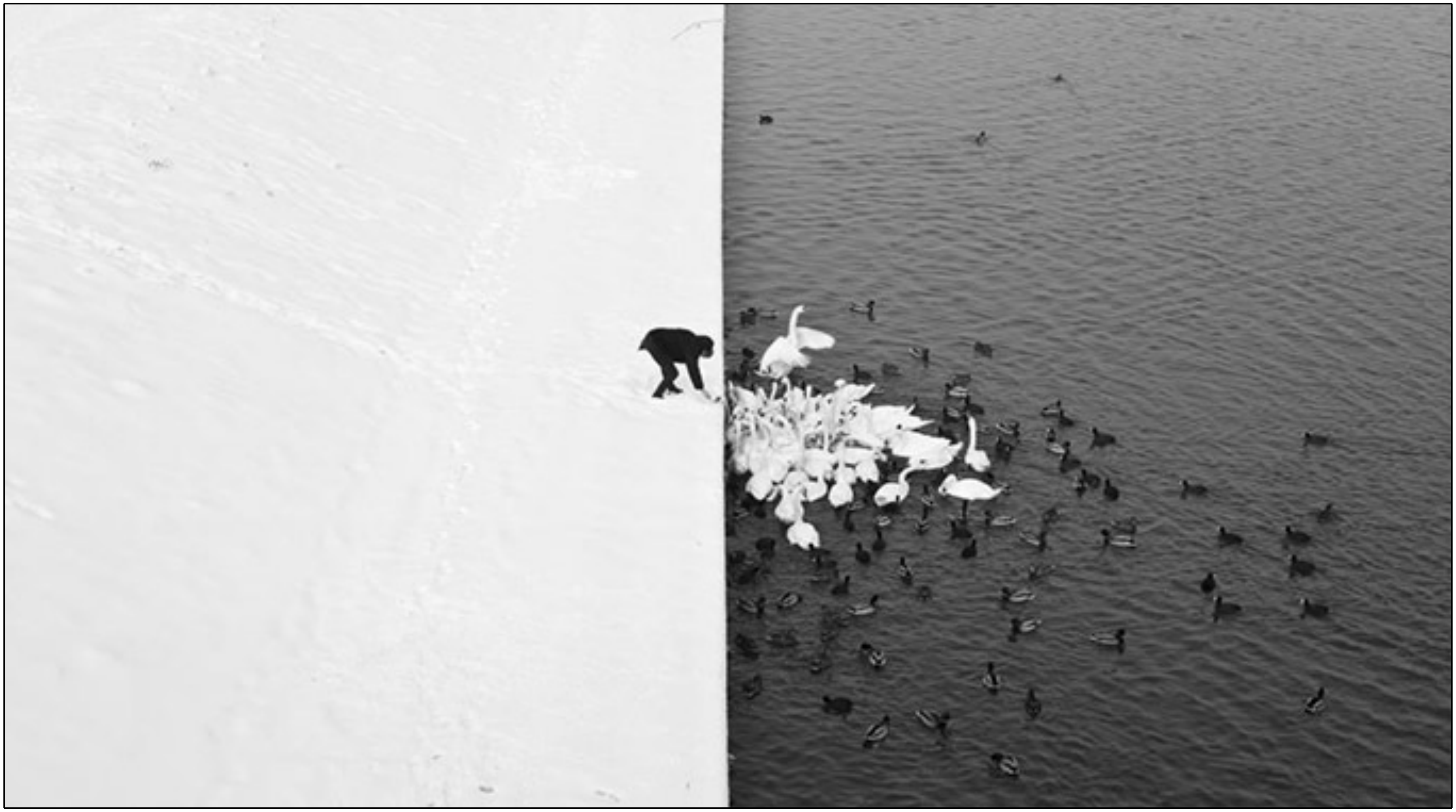
motorcycle and bicycle



dolphin and car

© 2006 Antonio Torralba and Aude Oliva

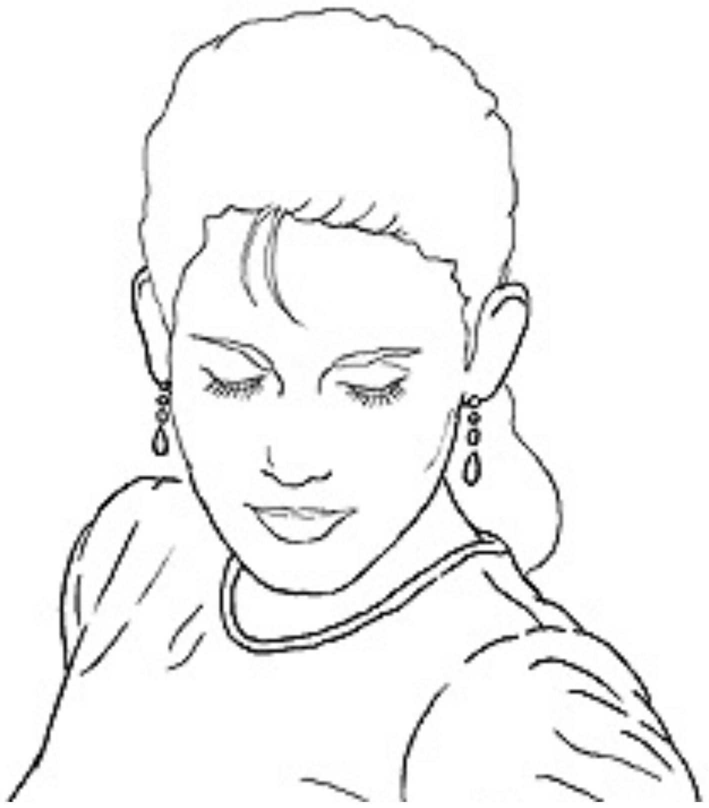
Edge detection



Winter in Kraków photographed by Marcin Ryczek

Edge detection

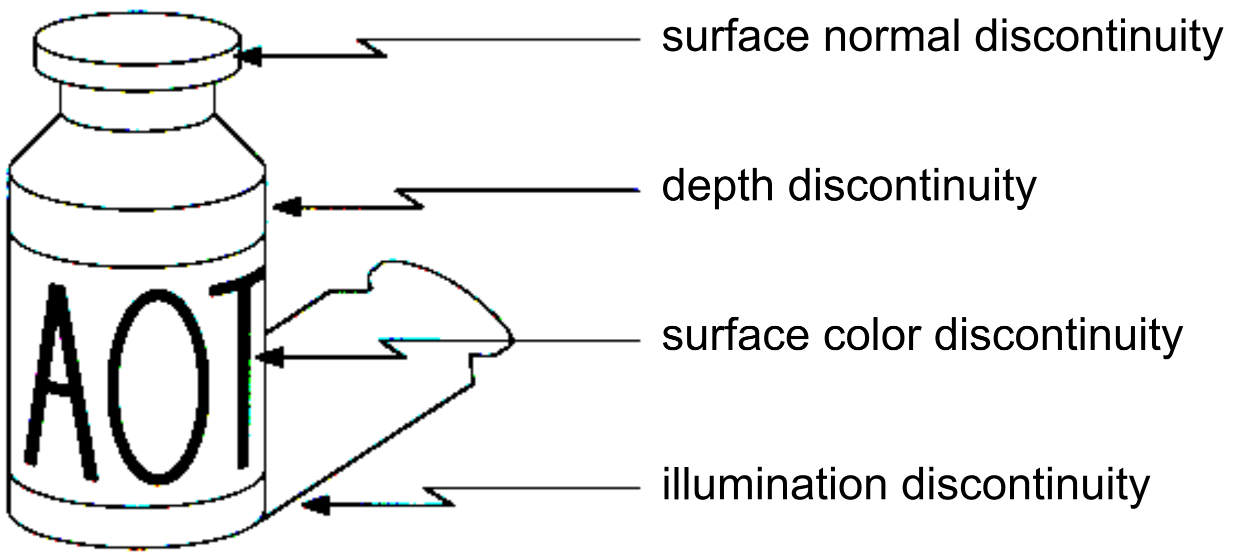
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



Source: D. Lowe

Origin of edges

Edges are caused by a variety of factors:

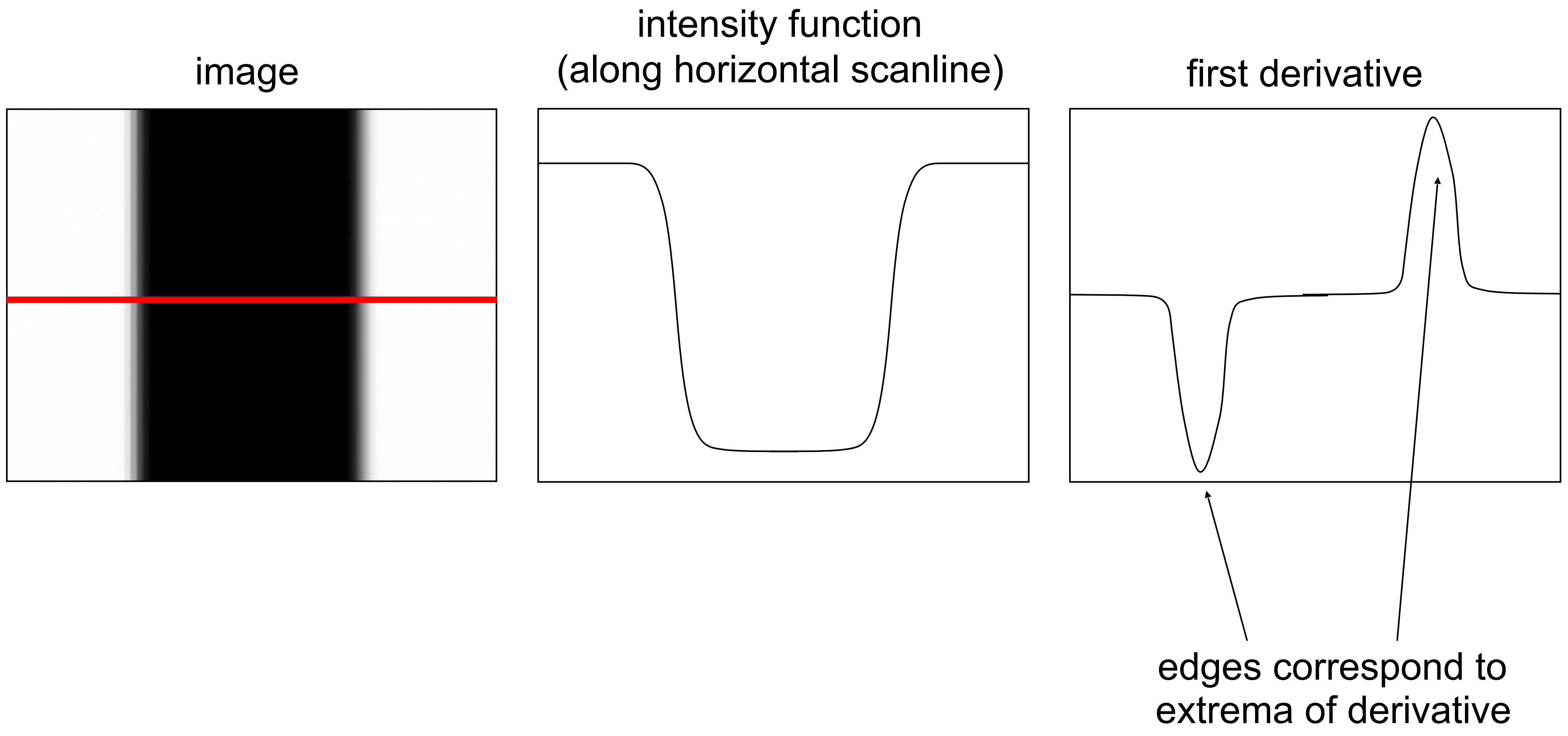


surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity
AOT

Source: Steve Seitz

Edge detection

- An edge is a place of rapid change in the image intensity function



Derivatives with convolution

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

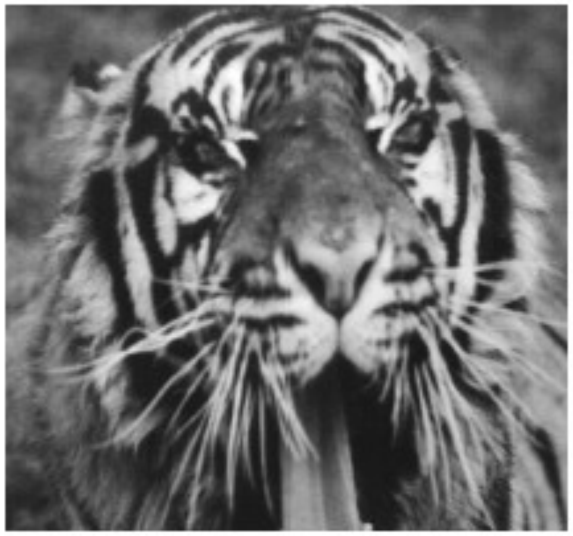
For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

To implement the above as convolution, what would be the associated filter?

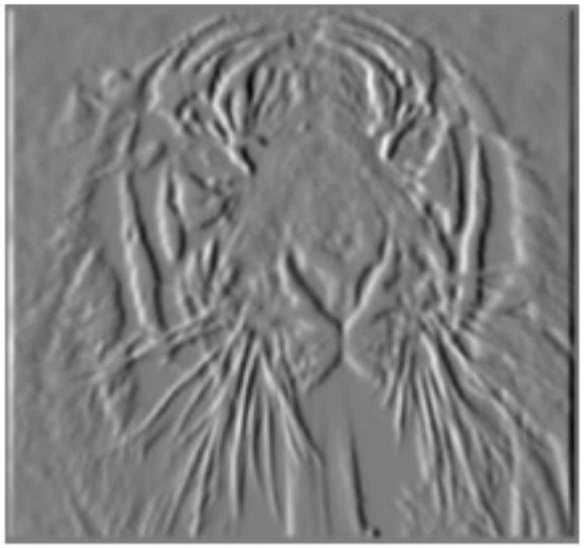
Source: K. Grauman

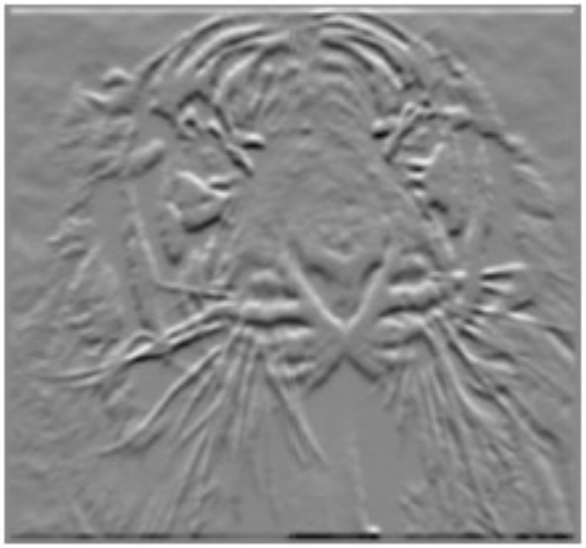
Partial derivatives of an image



$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





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

$$\begin{array}{|c|}\hline -1 \\ \hline 1 \\ \hline \end{array} \quad \textbf{or} \quad \begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array}$$

Which one shows changes with respect to x?

Finite difference filters

Other approximations of derivative filters exist:

$$\textbf{Prewitt:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$\textbf{Sobel:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

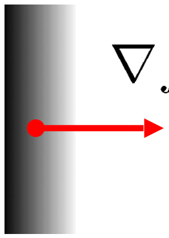
$$\textbf{Roberts:}\quad M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$$

Source: K. Grauman

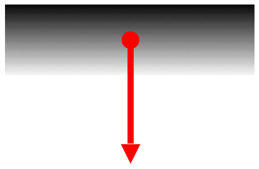
Image gradient

The gradient of an image:

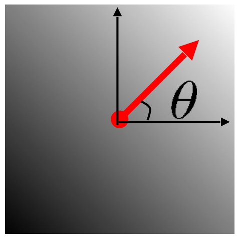
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

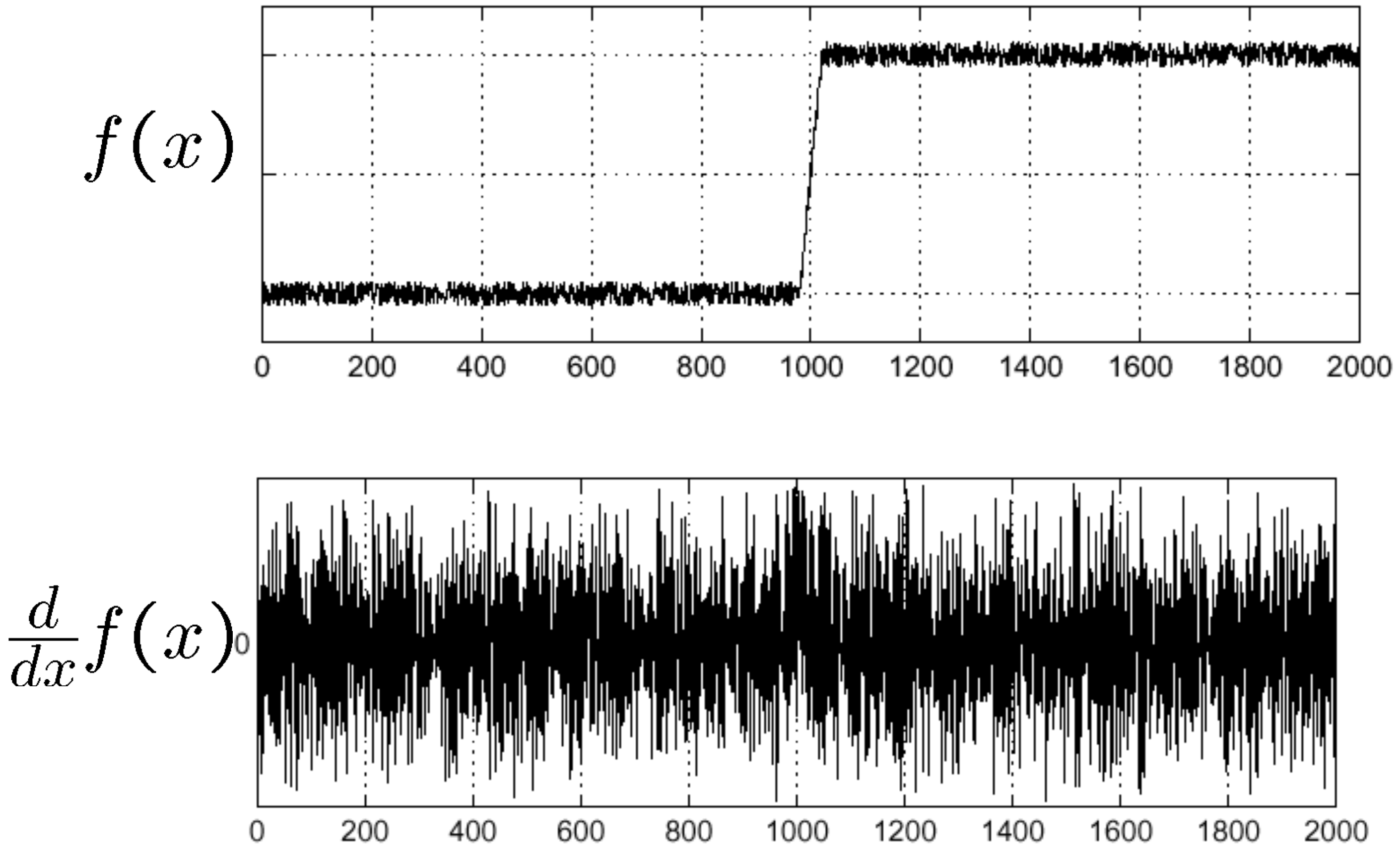
The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Effects of noise

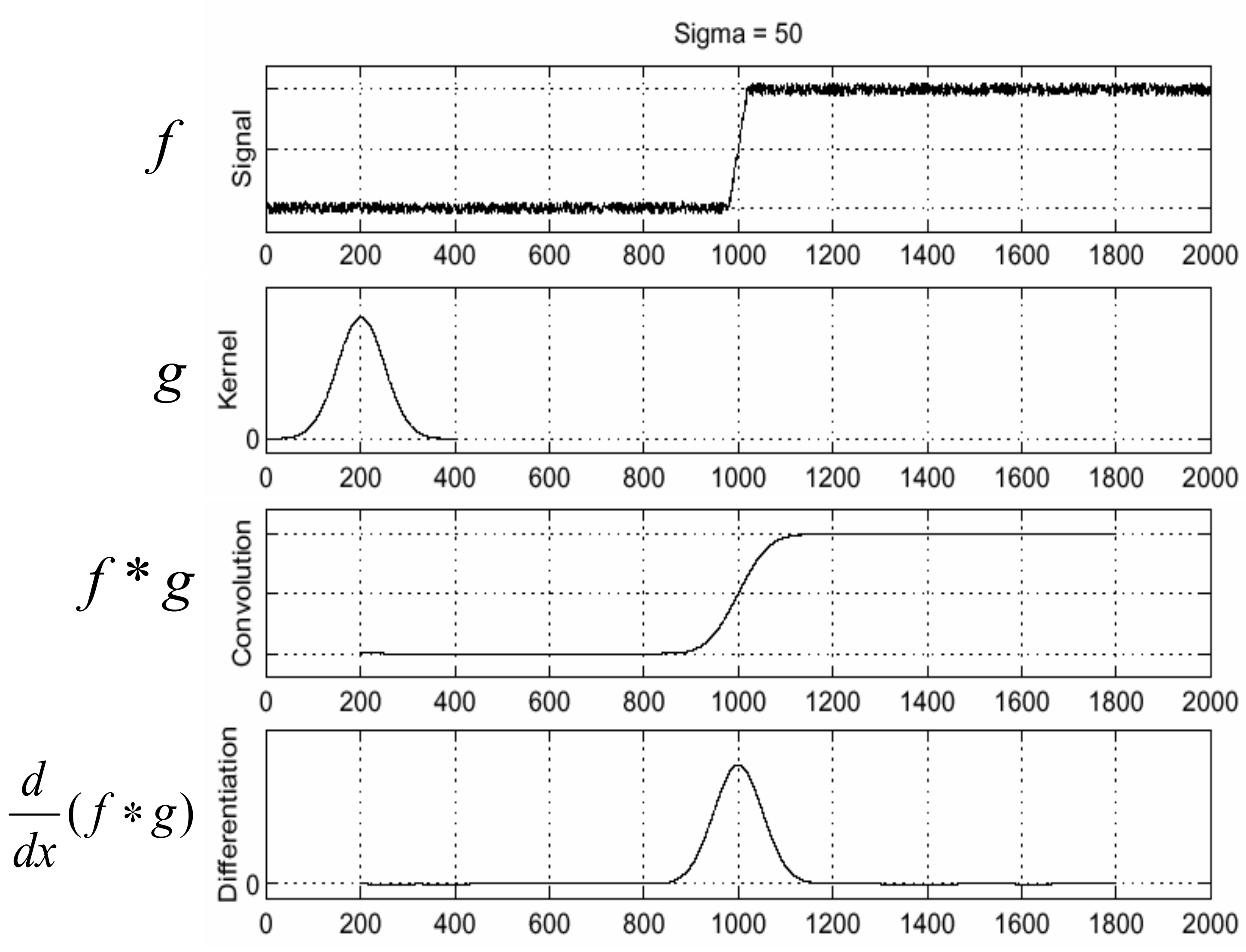
Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

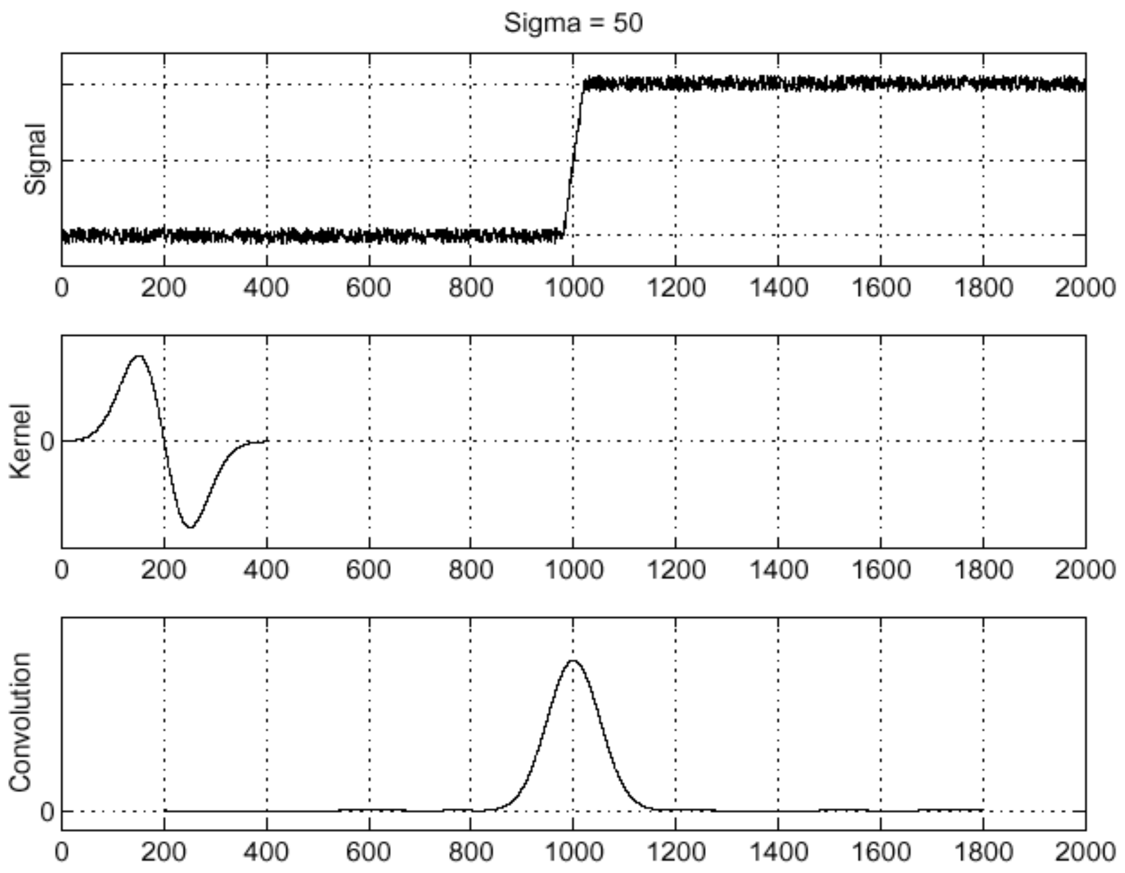
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

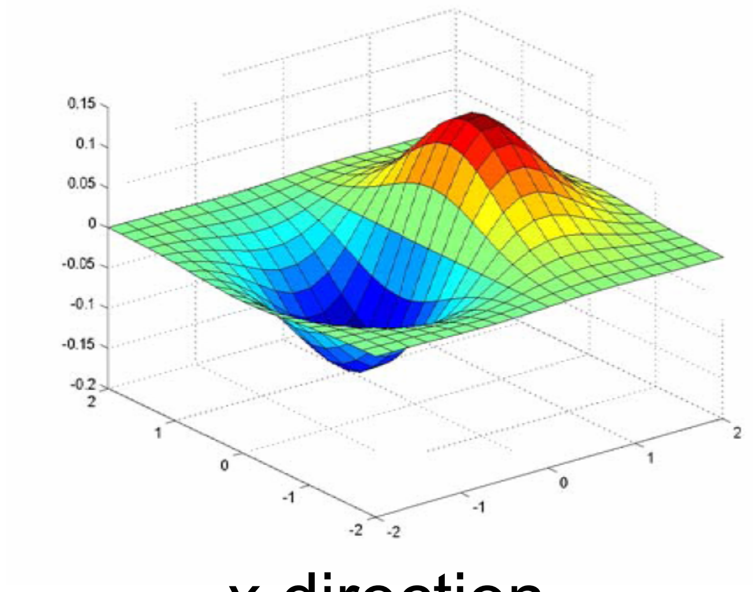
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

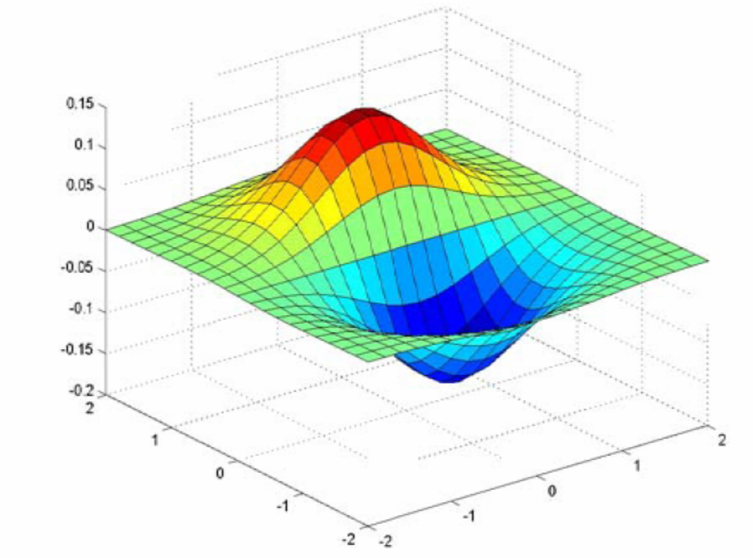


Source: S. Seitz

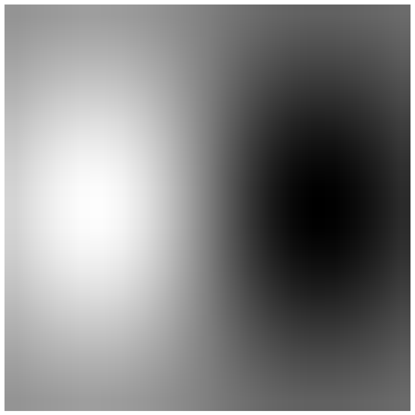
Derivative of Gaussian filters

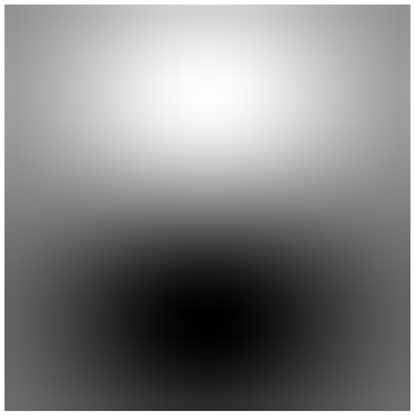


x-direction



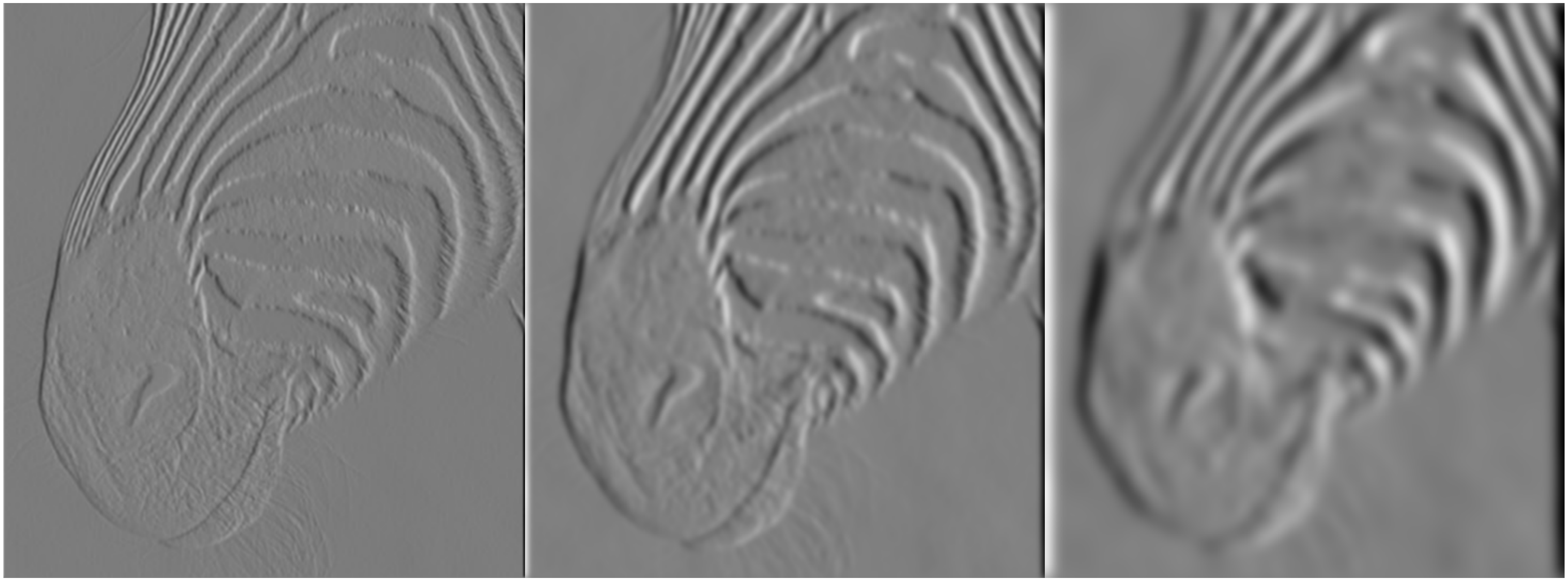
y-direction





1) Which one finds horizontal edges?
2) Are these filters separable?

Scale of Gaussian derivative filter



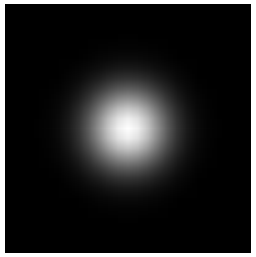
1 pixel   3 pixels   7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”

Source: D. Forsyth

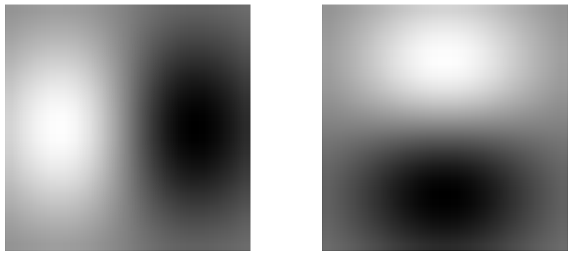
Smoothing and derivative filters

Smoothing filters



- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter

Derivative filters



- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions
- High absolute value at points of high contrast

The Canny edge detector

Filter image with derivative of Gaussian

Find magnitude and orientation of gradient

Non-maximum suppression:

Thin wide “ridges” down to single pixel width

Linking and thresholding (hysteresis):

Define two thresholds: low and high

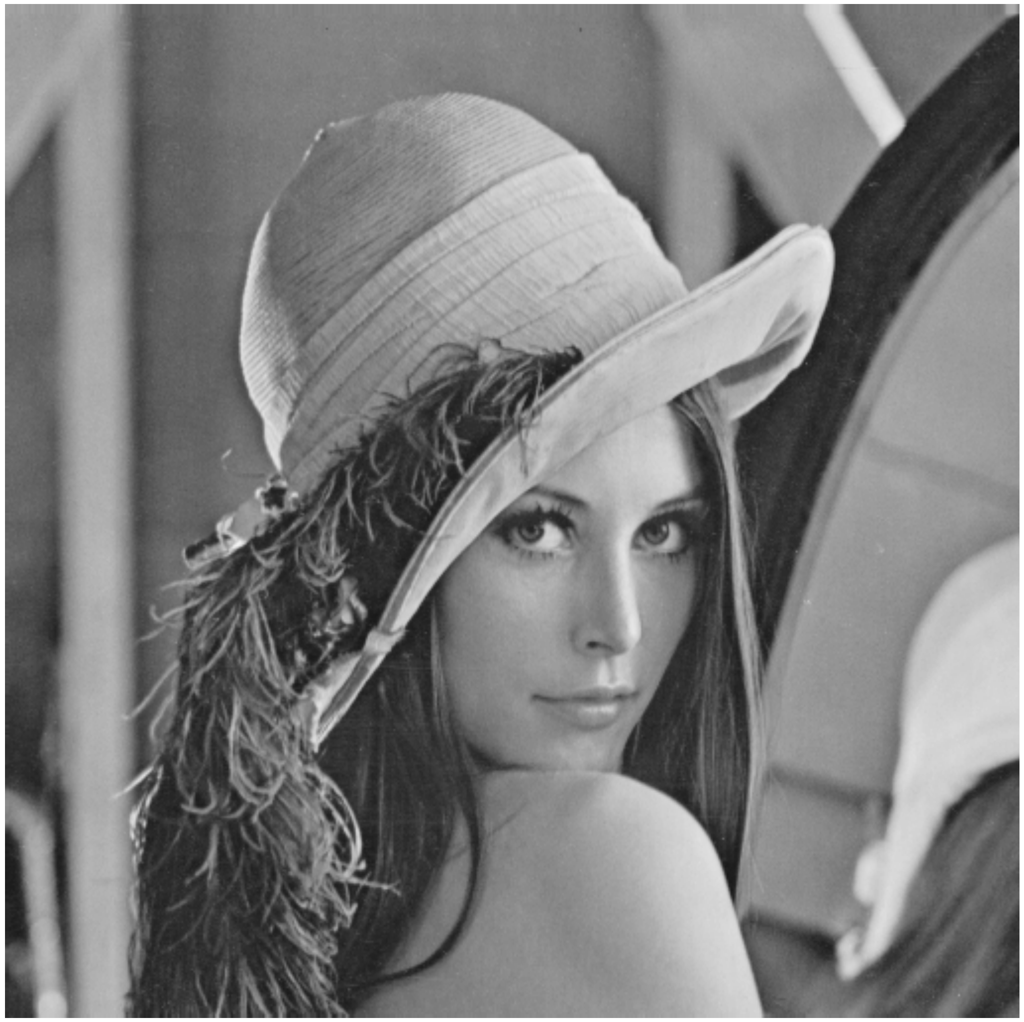
Use the high threshold to start edge curves and the low threshold to continue them

Matlab: `edge(image, ‘canny’);`

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

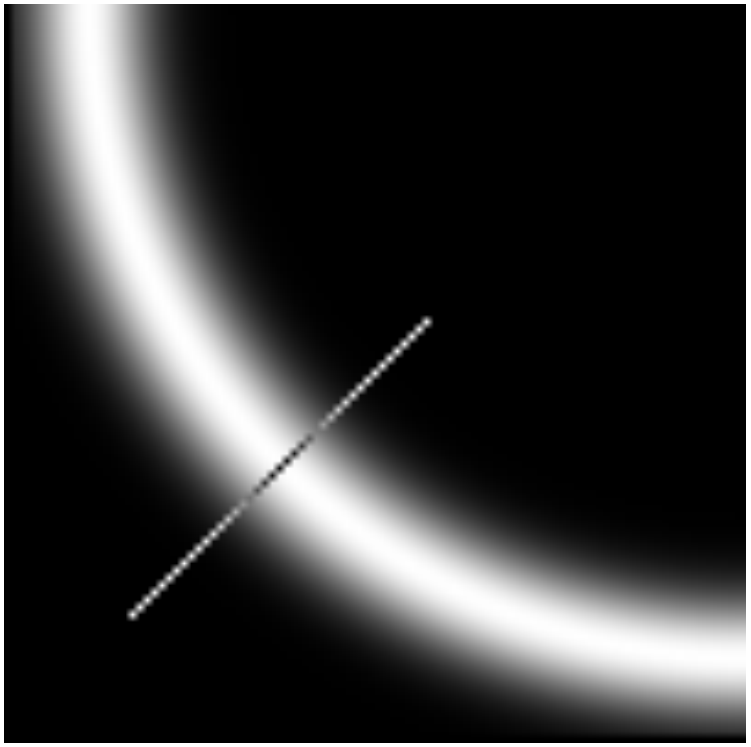
The Canny edge detector

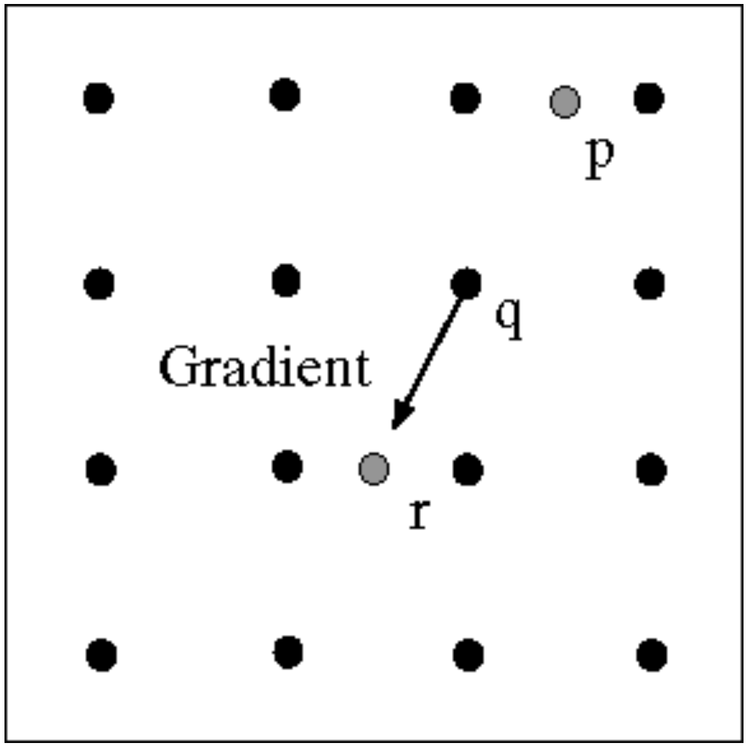
original image



Slide credit: Steve Seitz

Thinning (non-maximum suppression)

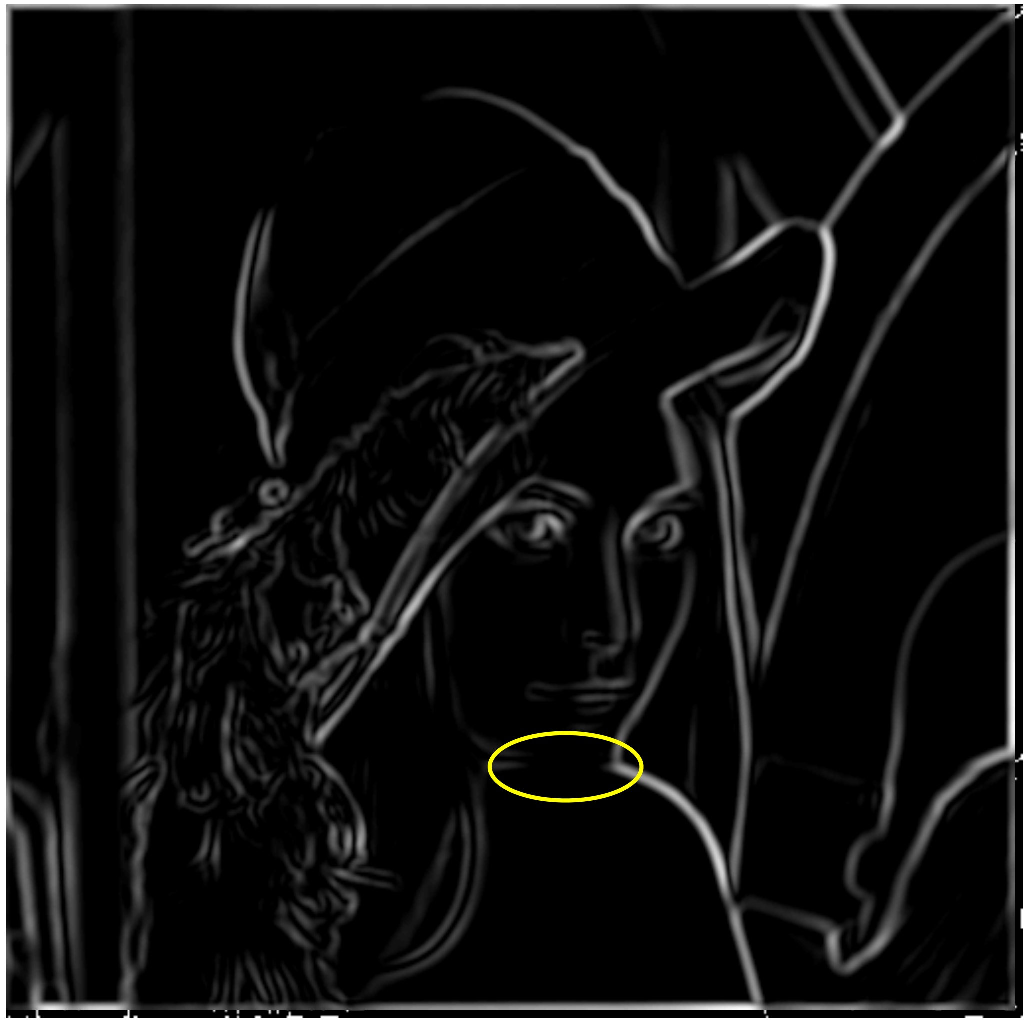




Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

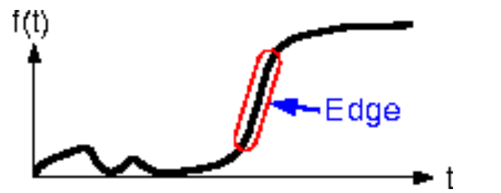
Thinning (non-maximum suppression)

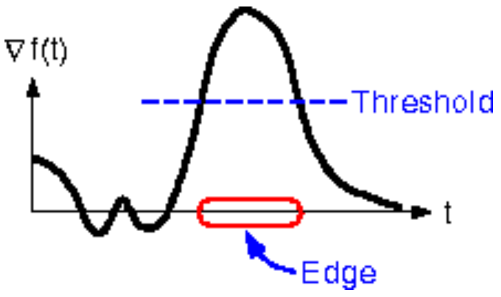


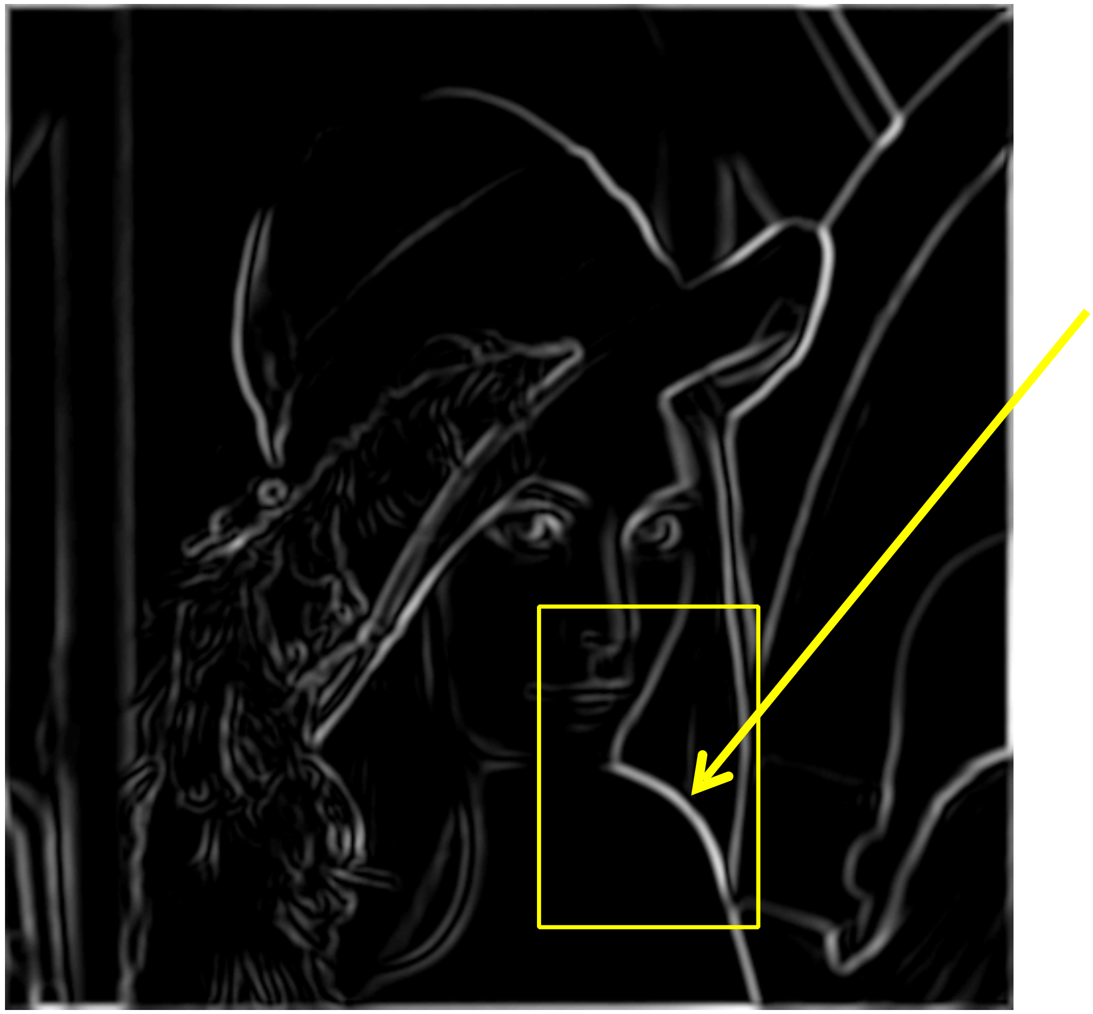
Problem: pixels along this edge didn’t survive the thresholding

thinning
(non-maximum suppression)

Hysteresis thresholding





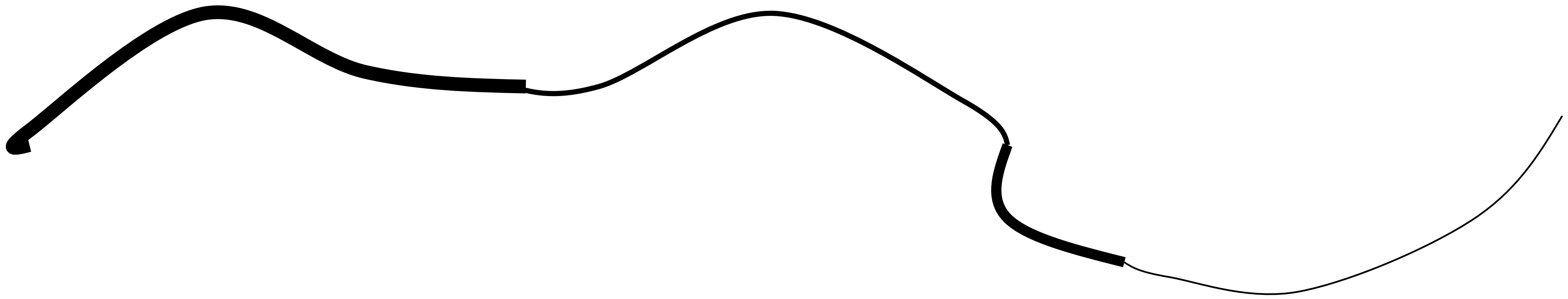


How to turn these thick regions of the gradient into curves?

thresholding

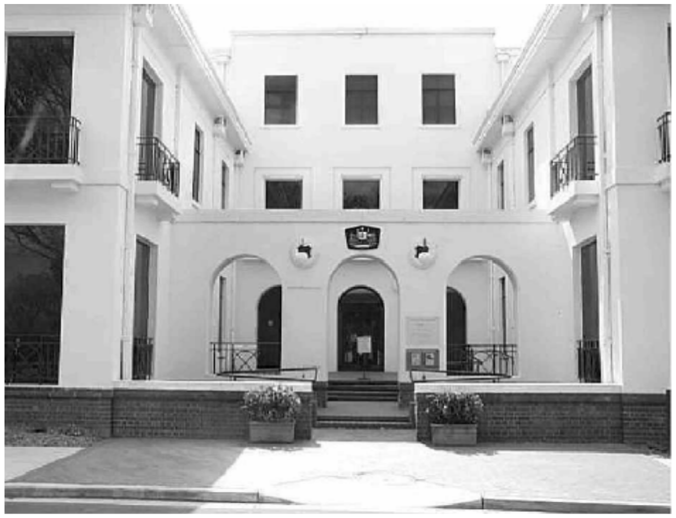
Hysteresis thresholding

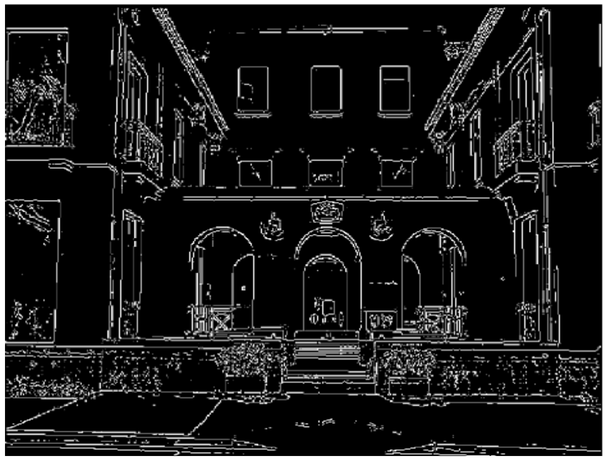
Use a high threshold to start edge curves, and a low threshold to continue them.

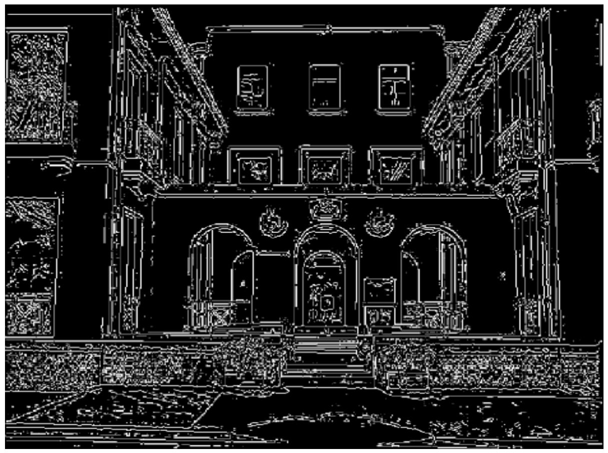


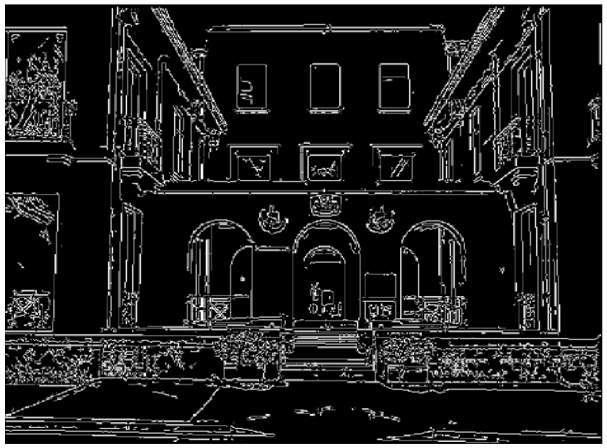
Source: Steve Seitz

Hysteresis thresholding


original image


high threshold (strong edges)


low threshold (weak edges)

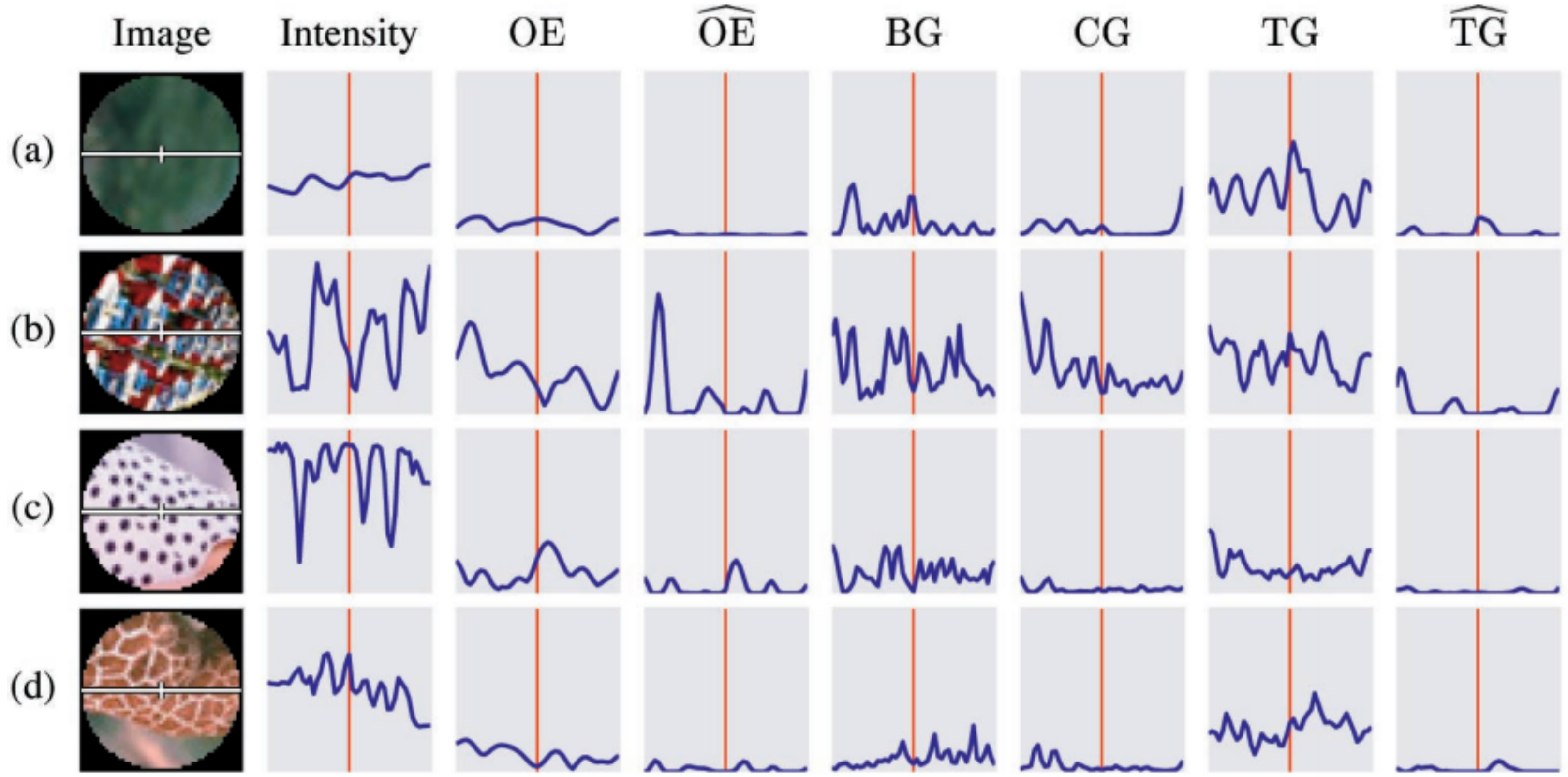

hysteresis threshold

Source: L. Fei-Fei

How do modern edge detectors work?

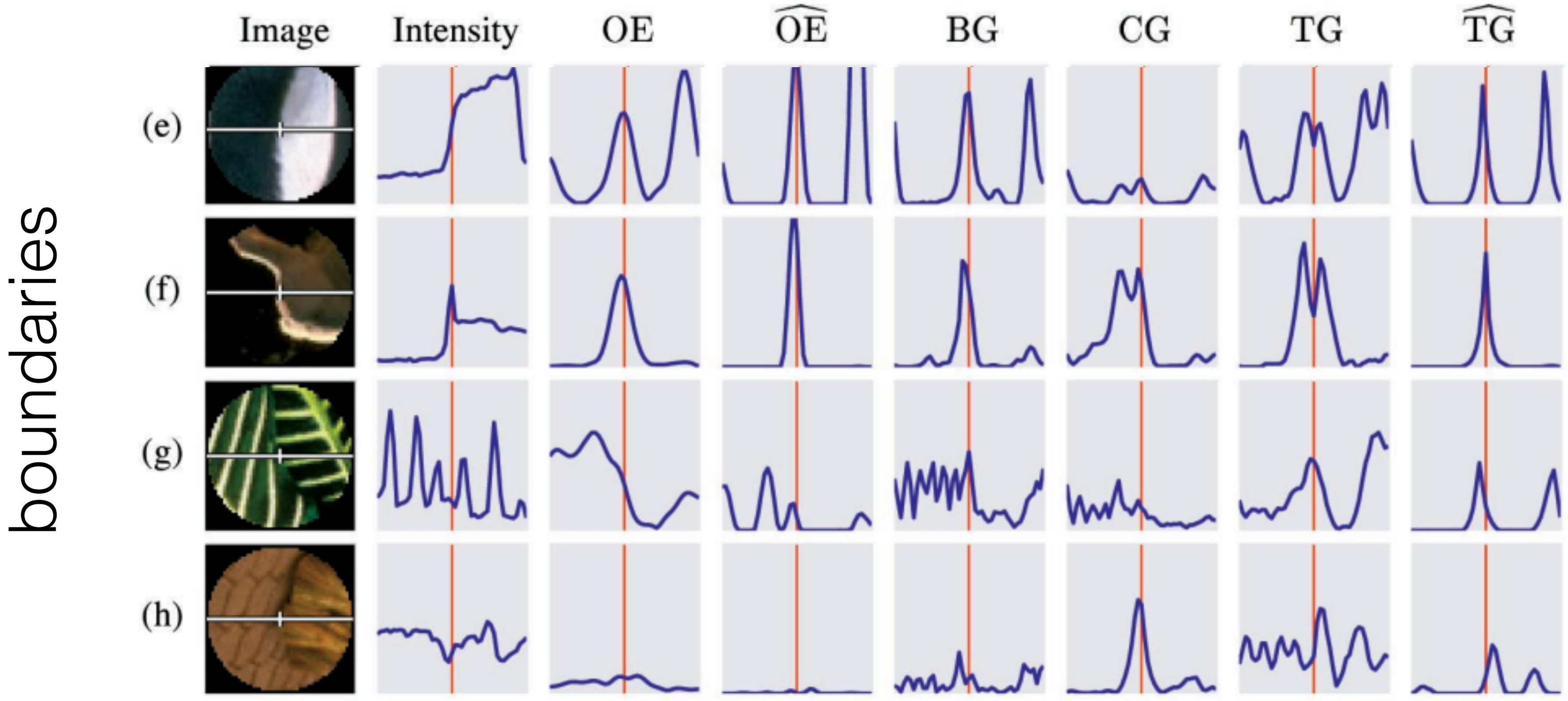
- Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues, *D. Martin, C. Flowkes, J. Malik, PAMI 2004*
  - Convert edge detection to classification problem: is there a vertical edge at the center of this patch?
  - Compare average brightness, color, and texture of half-disks
  - Do this for 8 different orientations

non-boundaries

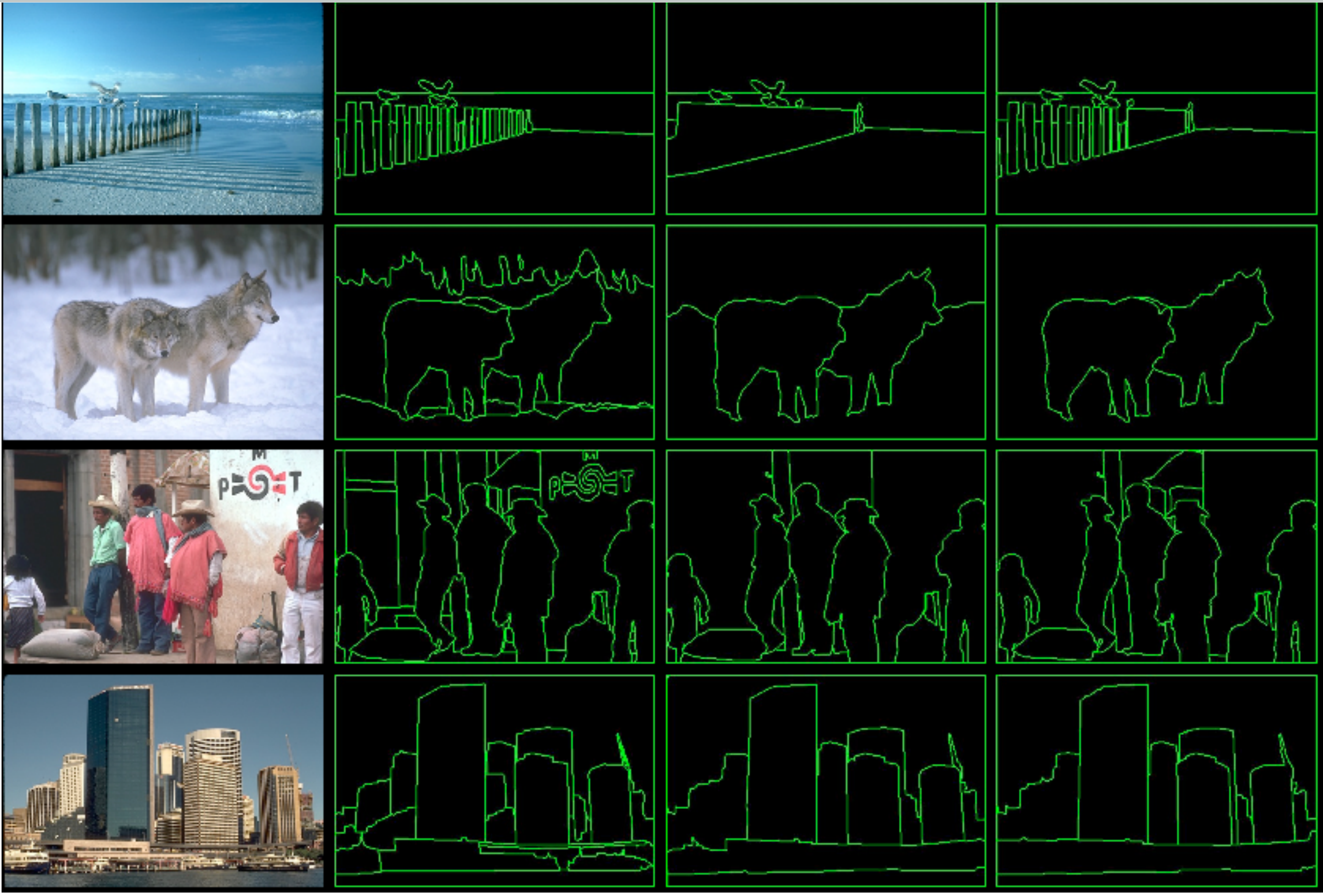


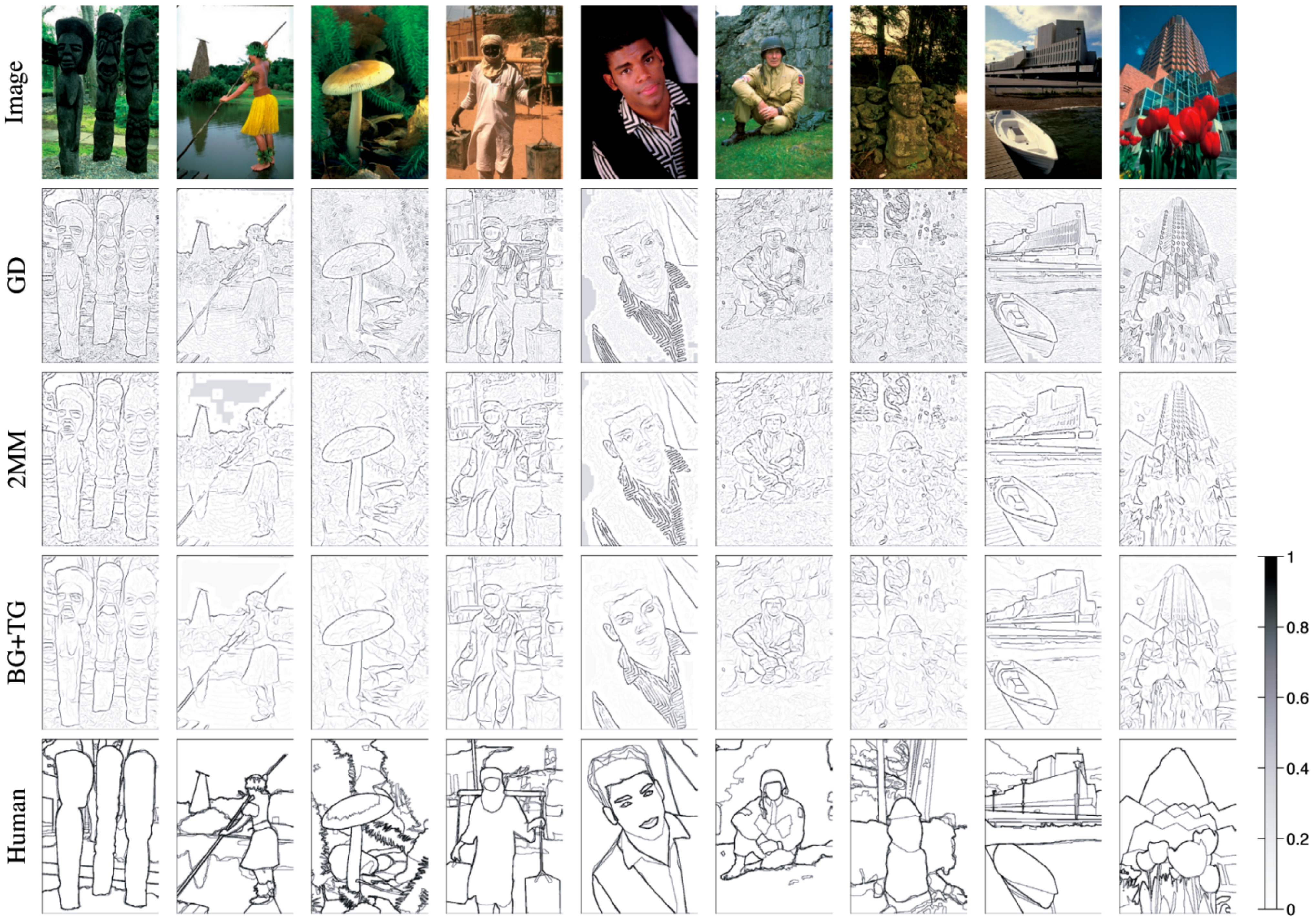
How do modern edge detectors work?

- Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues, *D. Martin, C. Flowkes, J. Malik, PAMI 2004*
  - Convert edge detection to classification problem: is there a vertical edge at the center of this patch?
  - Compare average brightness, color, and texture of half-disks
  - Do this for 8 different orientations



Berkeley segmentation database





example boundary detections

Further thoughts and readings …

- Hybrid images project
  - http://cvcl.mit.edu/hybridimage.htm
- Canny edge detector
  - www.limsi.fr/Individu/vezien/PAPIERS_ACS/canny1986.pdf
- Bilateral filtering for image de-noising (and other application)
  - http://people.csail.mit.edu/sparis/bf_course/
- Berkeley segmentation dataset and other resources
  - https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html