Lecture 3: Convolution and filtering

COS 429: Computer Vision

PRINCETON UNIVERSITY

Slides adapted from: Szymon Rusinkiewicz, Jia Deng

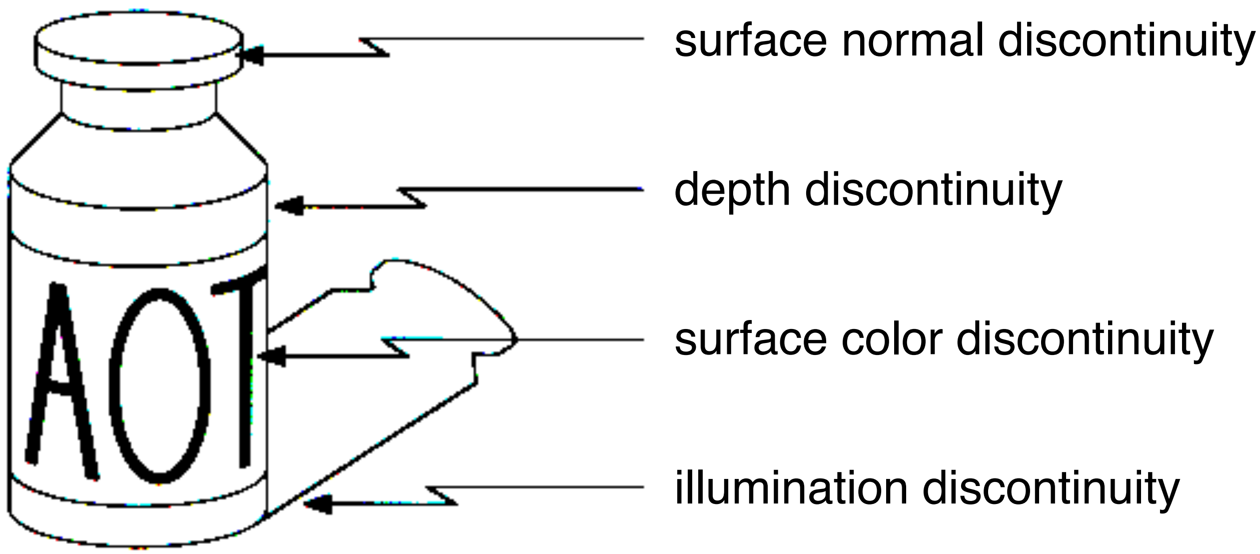
Image processing

90	92	92	93	93	94	94	95	95	96
94	95	96	96	97	98	98	99	99	99
98	99	99	100	101	101	102	102	102	103
103	103	104	104	105	107	106	106	111	121
108	108	109	110	112	111	112	119	123	117
113	113	110	111	113	112	122	120	117	106
118	118	109	96	106	113	112	108	117	114
116	132	120	111	109	106	101	106	117	118
111	142	112	111	101	106	104	109	113	110
114	139	109	108	103	106	107	108	108	108
115	139	117	114	101	104	103	105	114	110
115	129	103	114	101	97	109	116	117	118
120	130	104	111	116	104	107	109	110	99
125	130	103	109	108	98	104	109	119	105
119	128	123	138	140	133	139	120	137	145
164	138	143	163	155	133	145	125	133	155
174	126	123	122	102	106	108	62	62	114
169	134	133	127	92	102	94	47	52	118
125	132	117	122	102	103	98	51	53	120
109	99	113	116	111	98	104	82	99	116

What's the basic structure we may want to detect?

Origin of Edges

- Edges are caused by a variety of factors:



Credit: Steve Seitz

Edge Detection

- Intuitively, much of semantic and shape information is available in the edges
- Ideal: artist's line drawing (but artist is also using object-level knowledge)
- But what, mathematically, is an edge?

Credit: D. Lowe

What is an Edge?

Edge easy to find

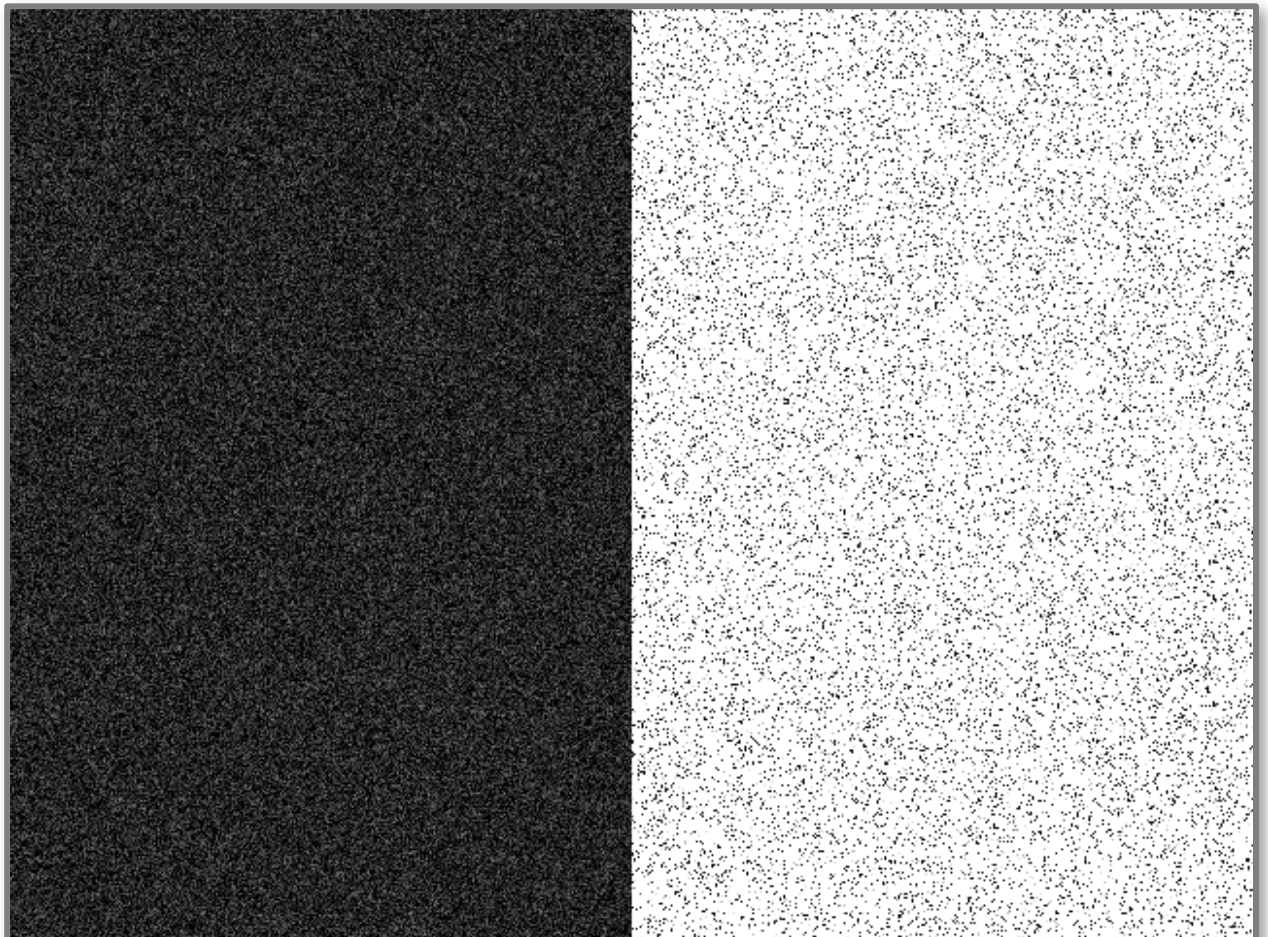
Source: S. Rusinkiewicz

What is an Edge?

Where is edge? Single pixel wide or multiple pixels?

Source: S. Rusinkiewicz

What is an Edge?



Noise: have to distinguish noise from actual edge

Source: S. Rusinkiewicz

Linear filtering: basics

Motivation: Image denoising

- How can we reduce noise in a photograph?

Source: S. Lazebnik

Idea #1: moving average

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

$$\frac{1}{9}\boxed{\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}}$$

"box filter"

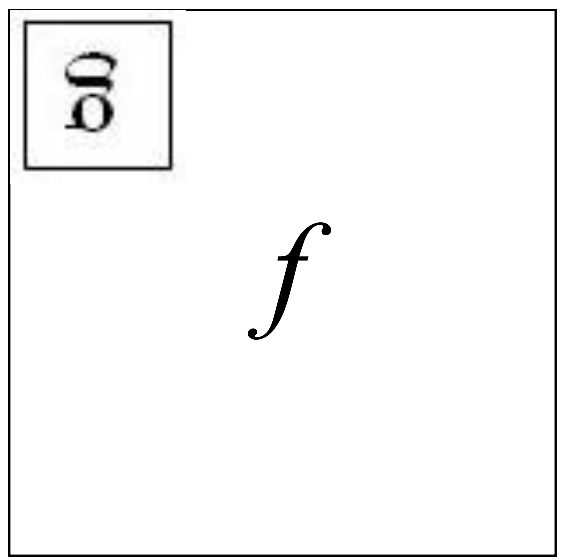
90	92	92	93	93	94	94	95	95	96
94	95	96	96	97	98	98	99	99	99
98	99	99	100	101	101	102	102	102	103
103	103	104	104	105	107	106	106	111	121
108	108	109	110	112	111	112	119	123	117
113	113	110	111	113	112	122	120	117	106
118	118	109	96	106	113	112	108	117	114
116	132	120	111	109	106	101	106	117	118
111	142	112	111	101	106	104	109	113	110
114	139	109	108	103	106	107	108	108	108
115	139	117	114	101	104	103	105	114	110
115	129	103	114	101	97	109	116	117	118
120	130	104	111	116	104	107	109	110	99
125	130	103	109	108	98	104	109	119	105
119	128	123	138	140	133	139	120	137	145
164	138	143	163	155	133	145	125	133	155

Defining convolution

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted *f* * *g*.

$$(f * g)[i,j] = \sum_{k,l} f[i-k, j-l]\,g[k,l]$$

Convention: kernel is "flipped"



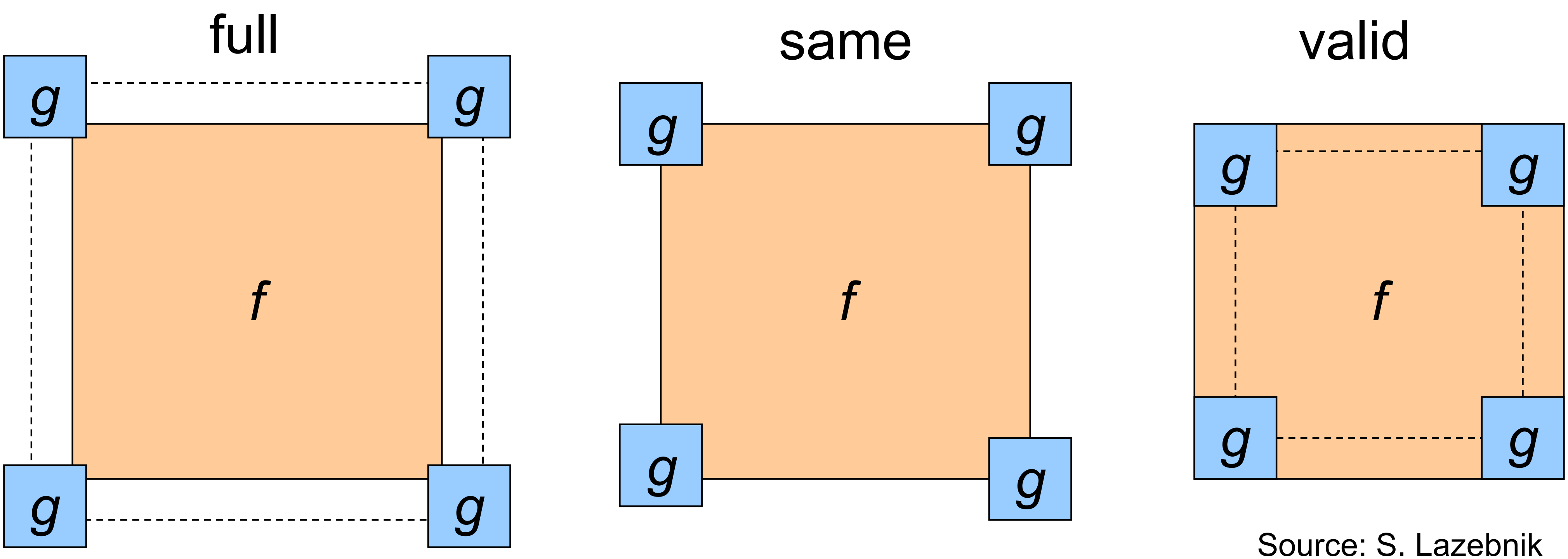
- Kernel center is positioned at [i,j]
  - for a 3x3 filter, k and l range between -1 and 1

Source: F. Durand

Annoying details

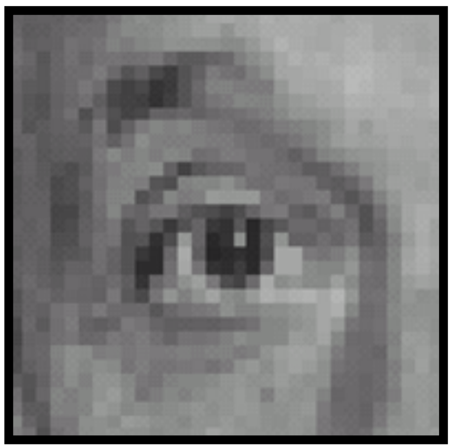
What is the size of the output?

- MATLAB: 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

Convolution with linear filters



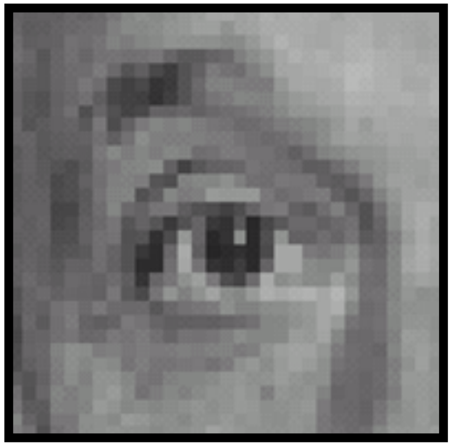
Original

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

?

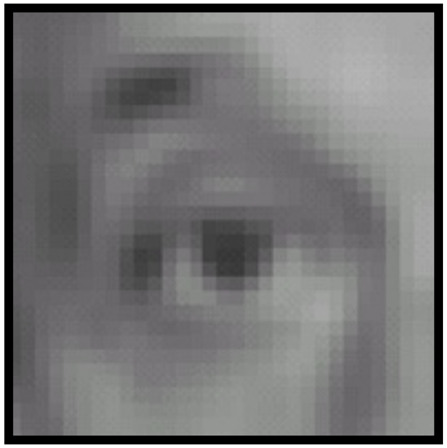
Source: D. Lowe

Convolution with linear filters



Original

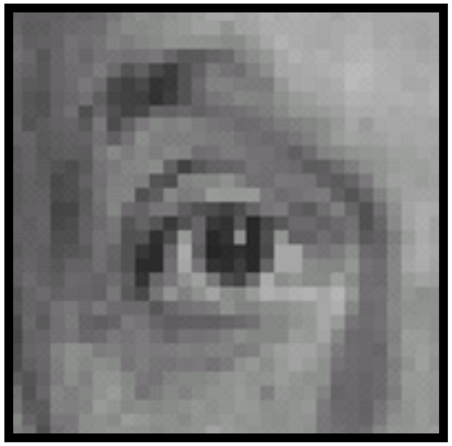
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Convolution with linear filters



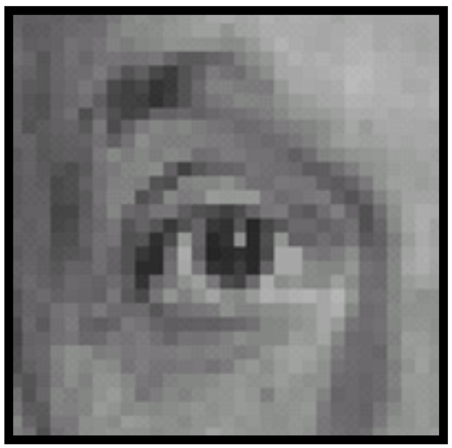
Original

0	0	0
0	1	0
0	0	0

?

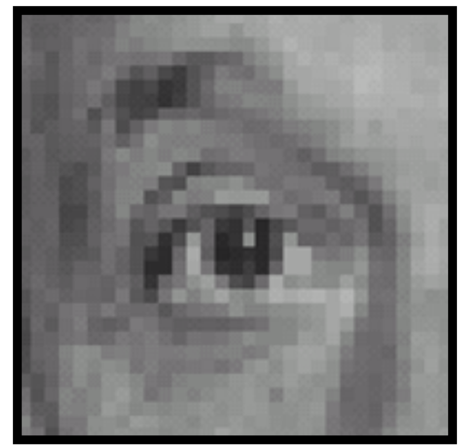
Source: D. Lowe

Convolution with linear filters



Original

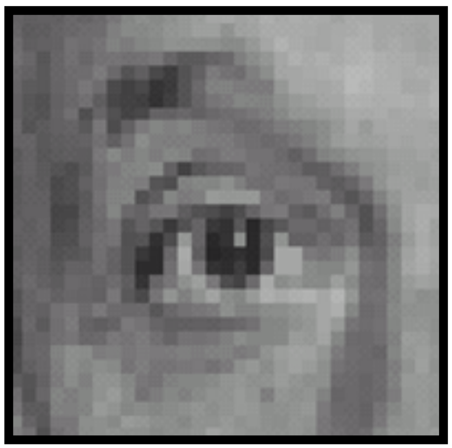
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Convolution with linear filters



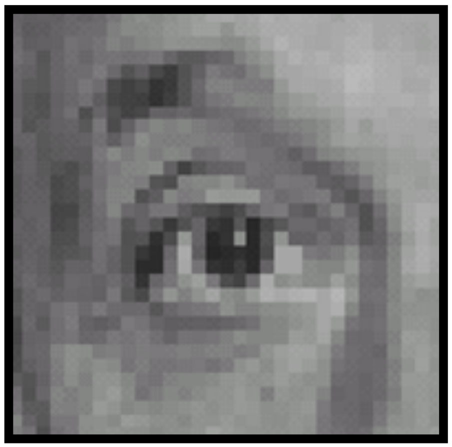
Original

0	0	0
1	0	0
0	0	0

?

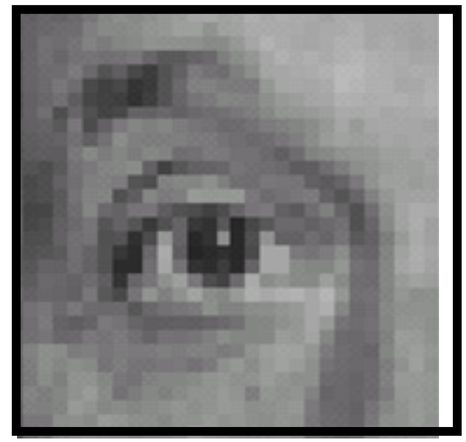
Source: D. Lowe

Convolution with linear filters



Original

0	0	0
1	0	0
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Proof that convolution is commutative

- **Claim**: $f * g = g * f$
  - Consider a 1-D convolution for simplicity (2-D proof similar)

$$(f * g)[i] = \sum_k f[i-k]g[k] \qquad\qquad m = i - k$$

$$= \sum_m f[m]g[i-m]$$

$$= (g * f)[i]$$

Key properties of convolutions

$$(f * g)[i] = \sum_k f[i-k]g[k]$$

- Commutative: $f * g = g * f$
  - Conceptually no difference between filter and signal
- Associative: $f * (g * h) = (f * g) * h$
  - Often apply several filters one after another: $(((f * g_1) * g_2) * g_3)$
  - This is equivalent to applying one filter: $f * (g_1 * g_2 * g_3)$
- Distributes over addition: $f * (g + h) = (f * g) + (f * h)$
- Scalars factor out: $kf * g = f * kg = k\,(f * g)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $f * e = f$

Source: S. Lazebnik

Convolution vs cross-correlation

Convolution

$$(f * g)[i] = \sum_k f[i-k]g[k]$$

- Preserves **associativity** and **commutativity**, unlike cross-correlation (exercise: check)

- Matlab: conv, conv2, imfilter

Cross-correlation

$$(f \otimes g)[i] = \sum_k f[i+k]g[k]$$

- Intuitively **simpler**

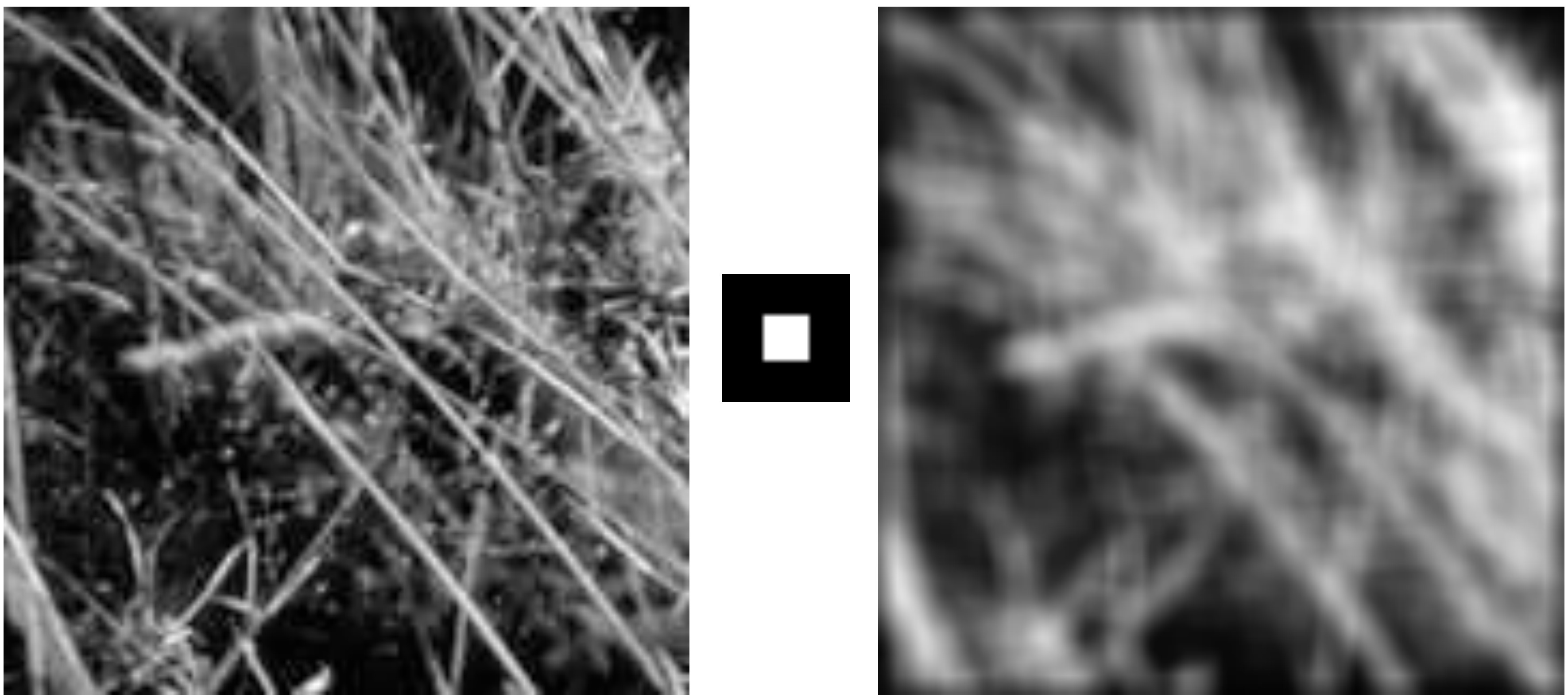
- Matlab: filter2, imfilter

- Used somewhat interchangeably in practice

Sharpening and Gaussian filters

Smoothing with box filter revisited

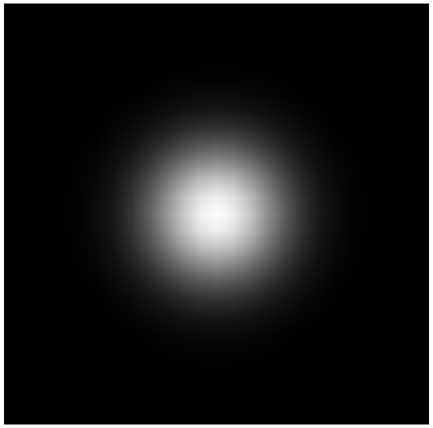
- What’s wrong with this picture?
- What’s the solution?



Source: D. Forsyth

Smoothing with box filter revisited

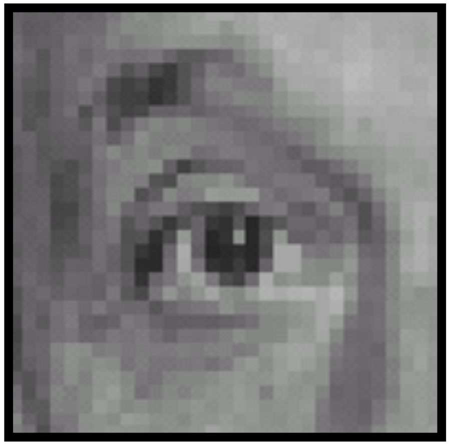
- What's wrong with this picture?
- What's the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center



"fuzzy blob"

Source: S. Lazebnik

Convolution with linear filters



Original

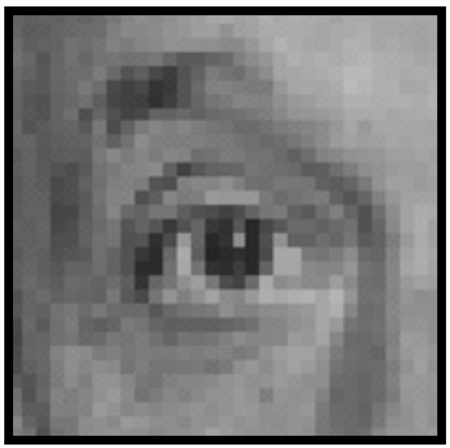
$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

?

(Note that filter sums to 1)

Source: D. Lowe

Convolution with linear filters



Original

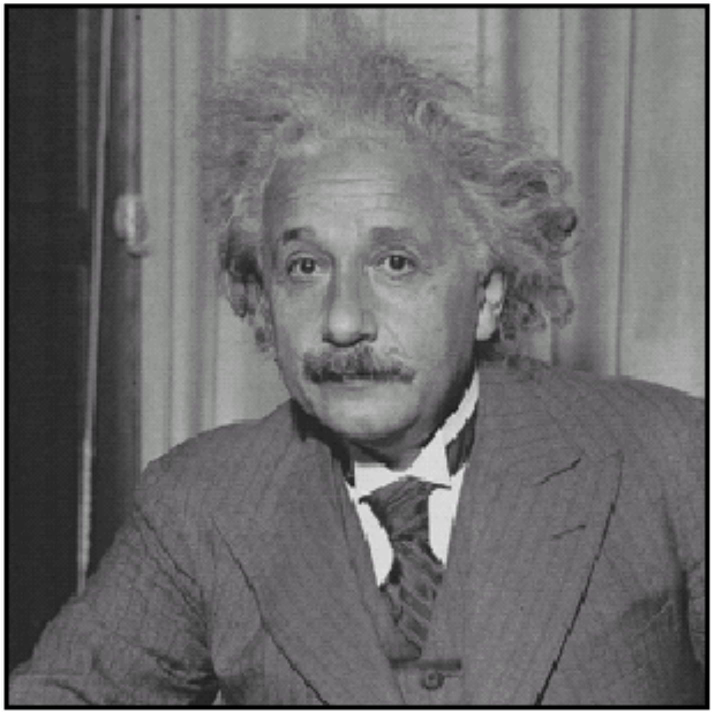
$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

Sharpening filter

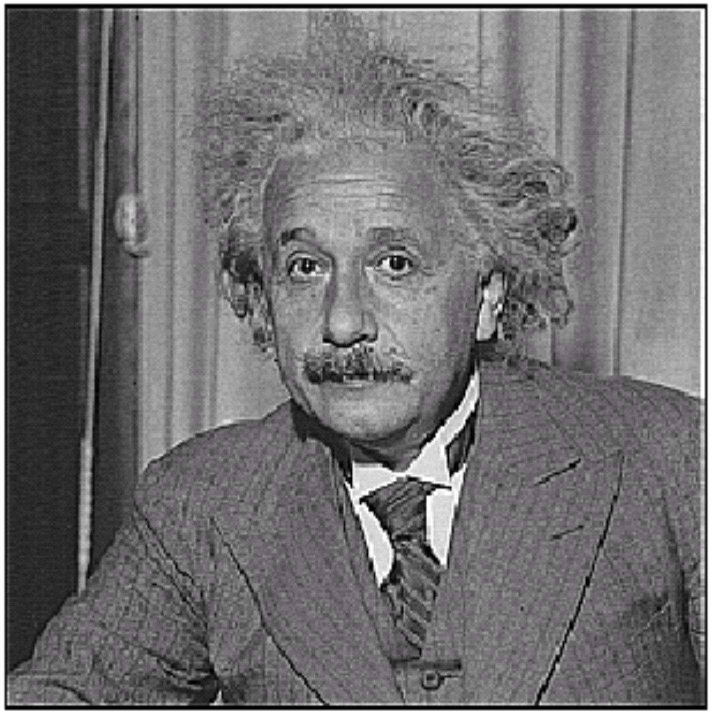
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before

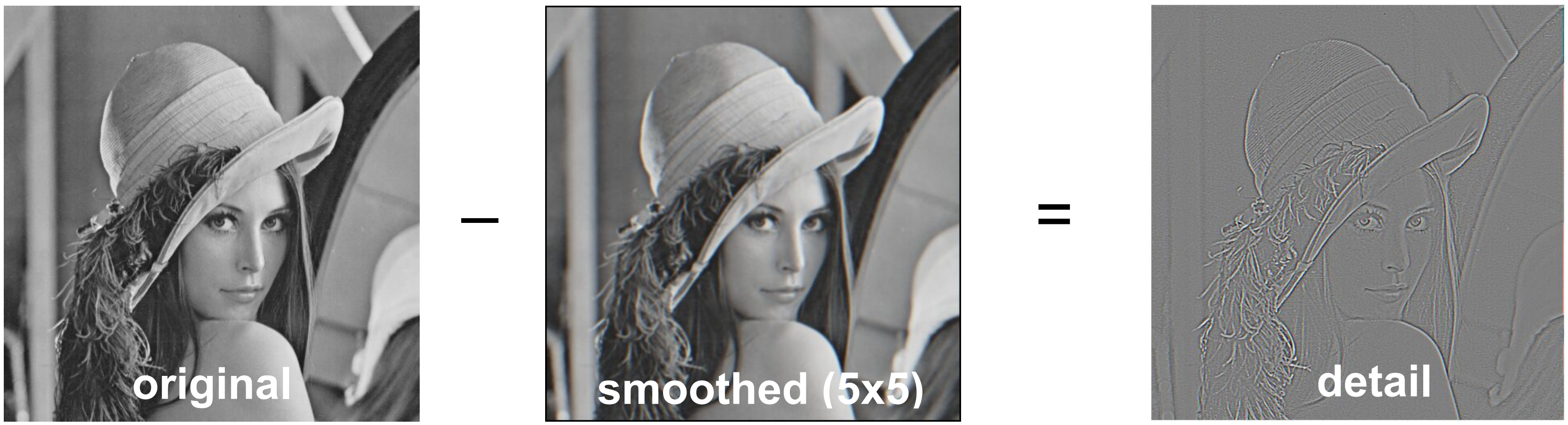


after

Source: D. Lowe

Sharpening

What does blurring take away?



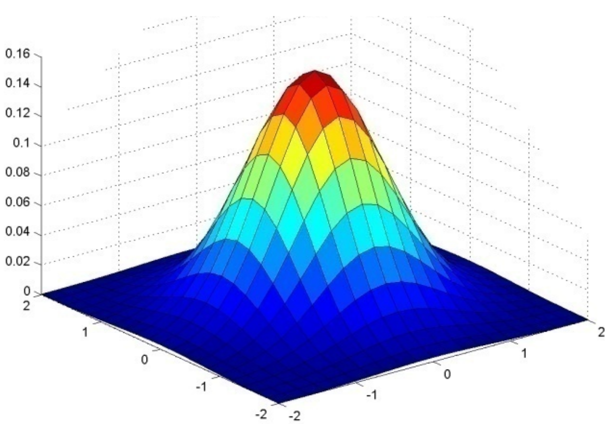
Let’s add it back:

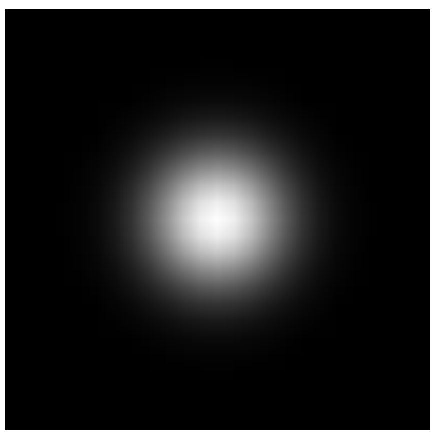
Source: S. Lazebnik

Gaussian Kernel

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

- 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)





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

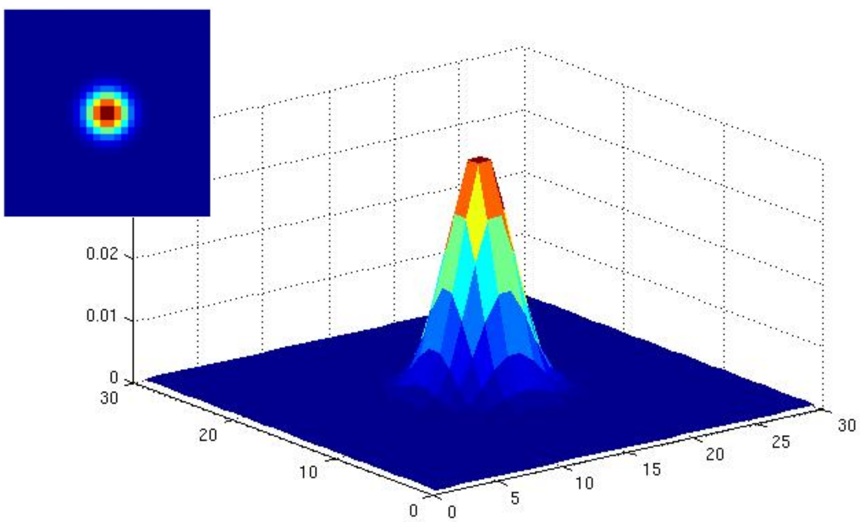
5 x 5, $\sigma = 1$

Source: C. Rasmussen

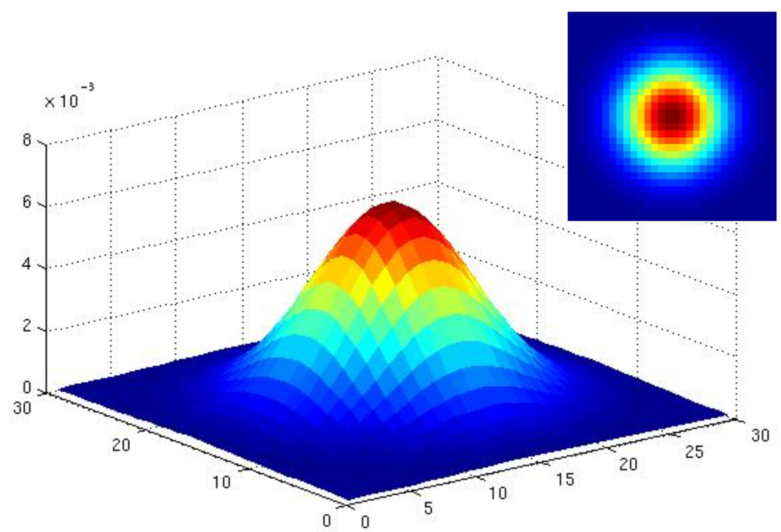
Gaussian Kernel

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

- Standard deviation σ: determines extent of smoothing



σ = 2 with 30 x 30 kernel

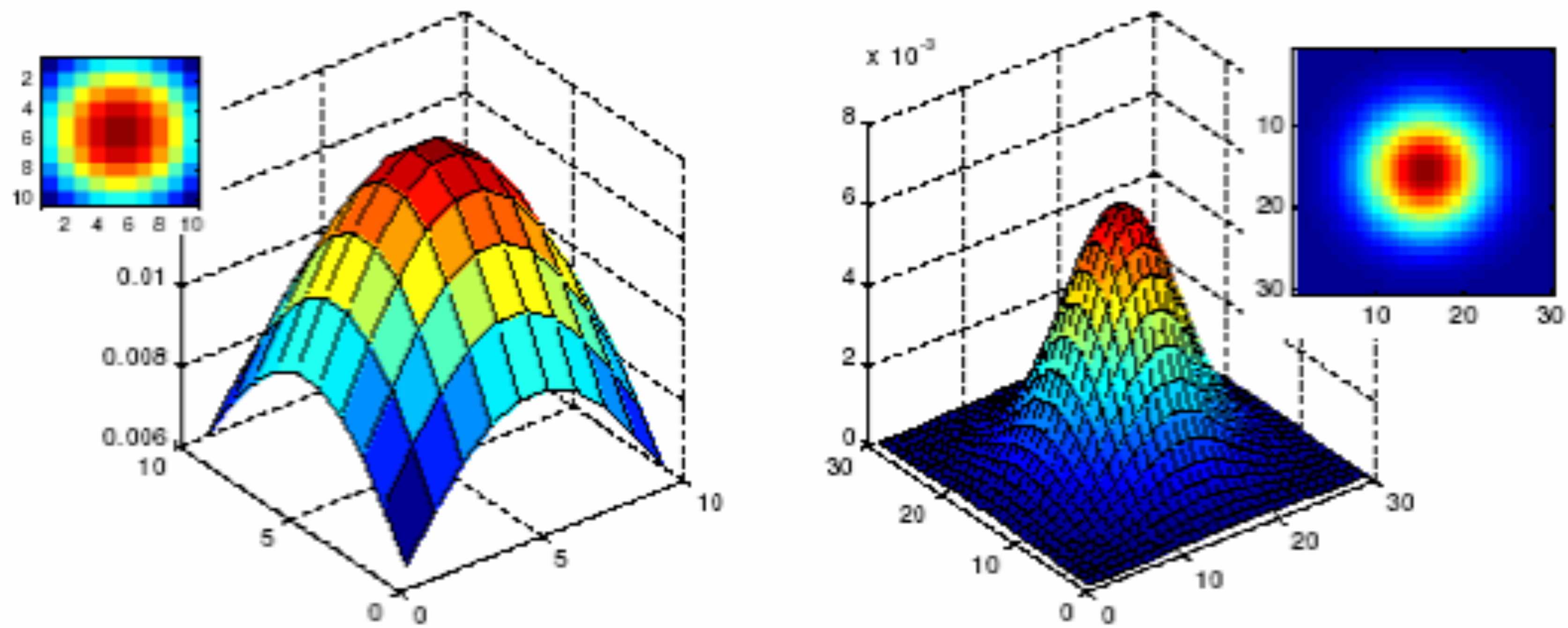


σ = 5 with 30 x 30 kernel

Source: K. Grauman

Choosing kernel width

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



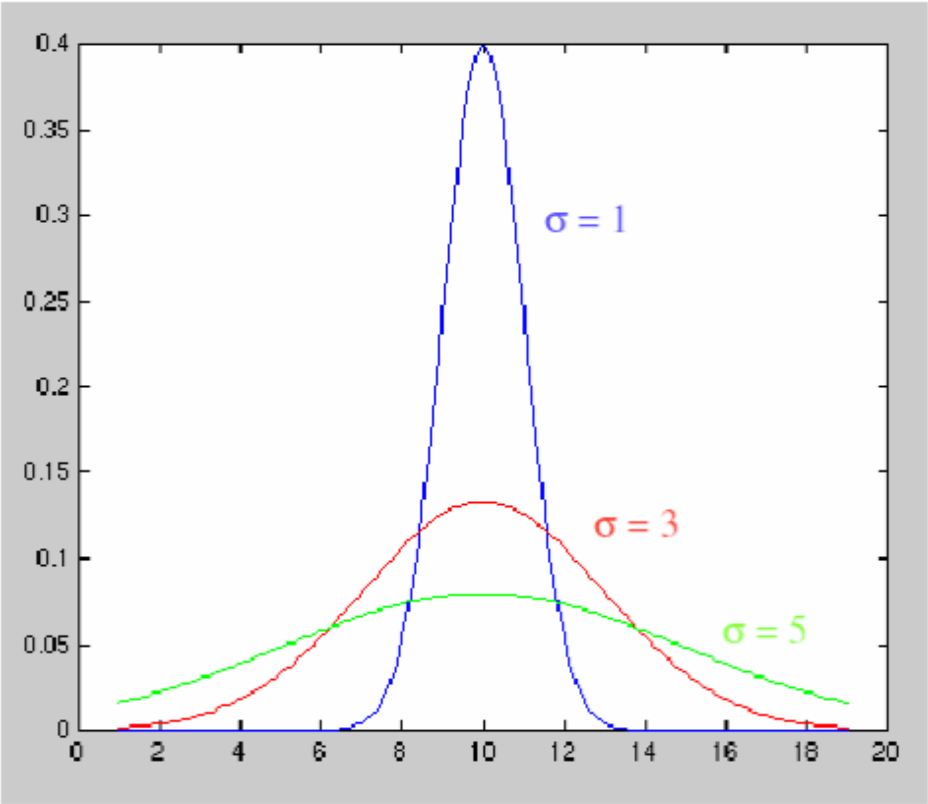
$\sigma = 5$ with 10x10 kernel

$\sigma = 5$ with 30x30 kernel

Source: K. Grauman

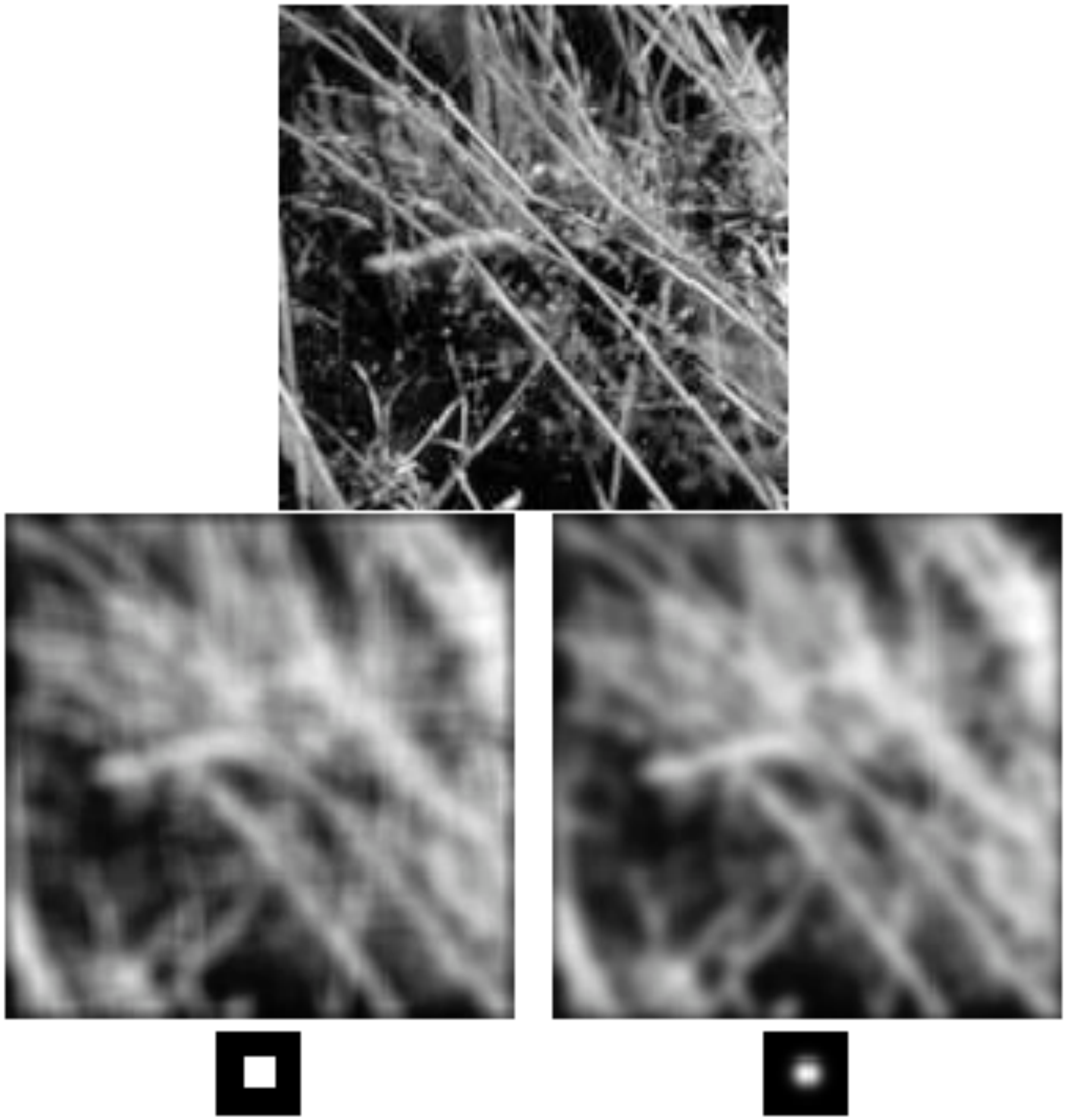
Choosing kernel width

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



Source: S. Lazebnik

Gaussian vs. box filtering



Source: S. Lazebnik

Gaussian filters properties

- 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?

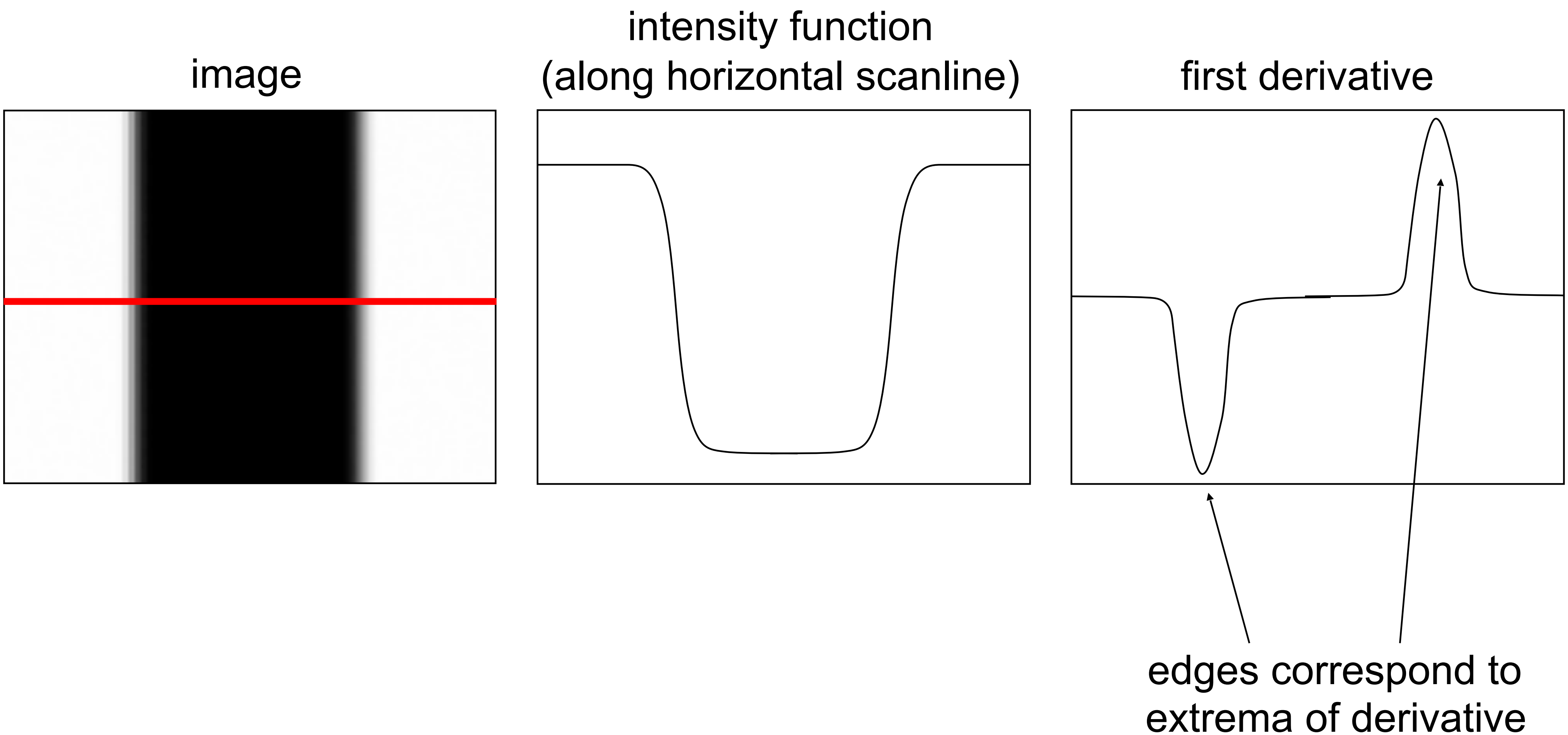
- Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Source: S. Lazebnik

Coming back to edge detection

Winter in Kraków photographed by Marcin Ryczek

Edge detection



Source: S. Lazebnik

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:

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

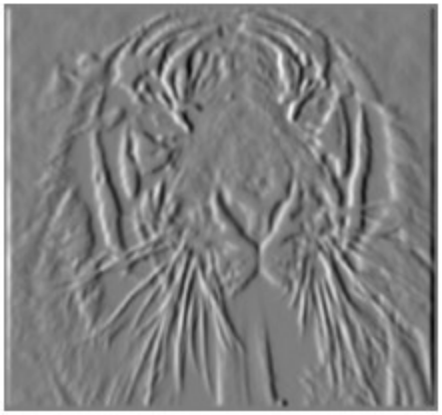
To implement the above as convolution, what would be the associated filter?

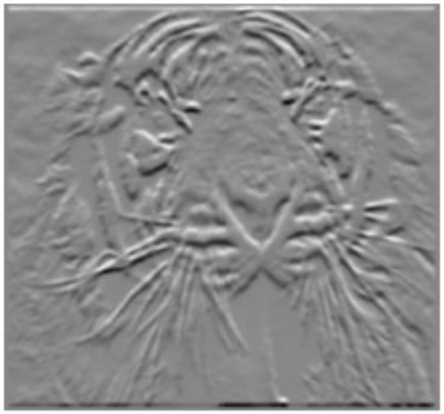
Source: K. Grauman

Partial derivatives of an image

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

-1	1





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

-1
1

or

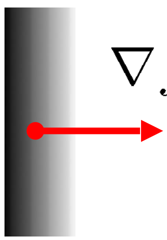
1
-1

Which shows changes with respect to x?

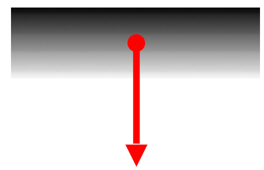
Source: S. Lazebnik

Image gradient

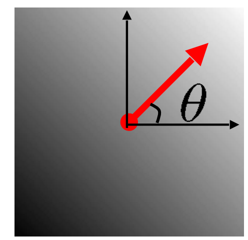
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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 $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

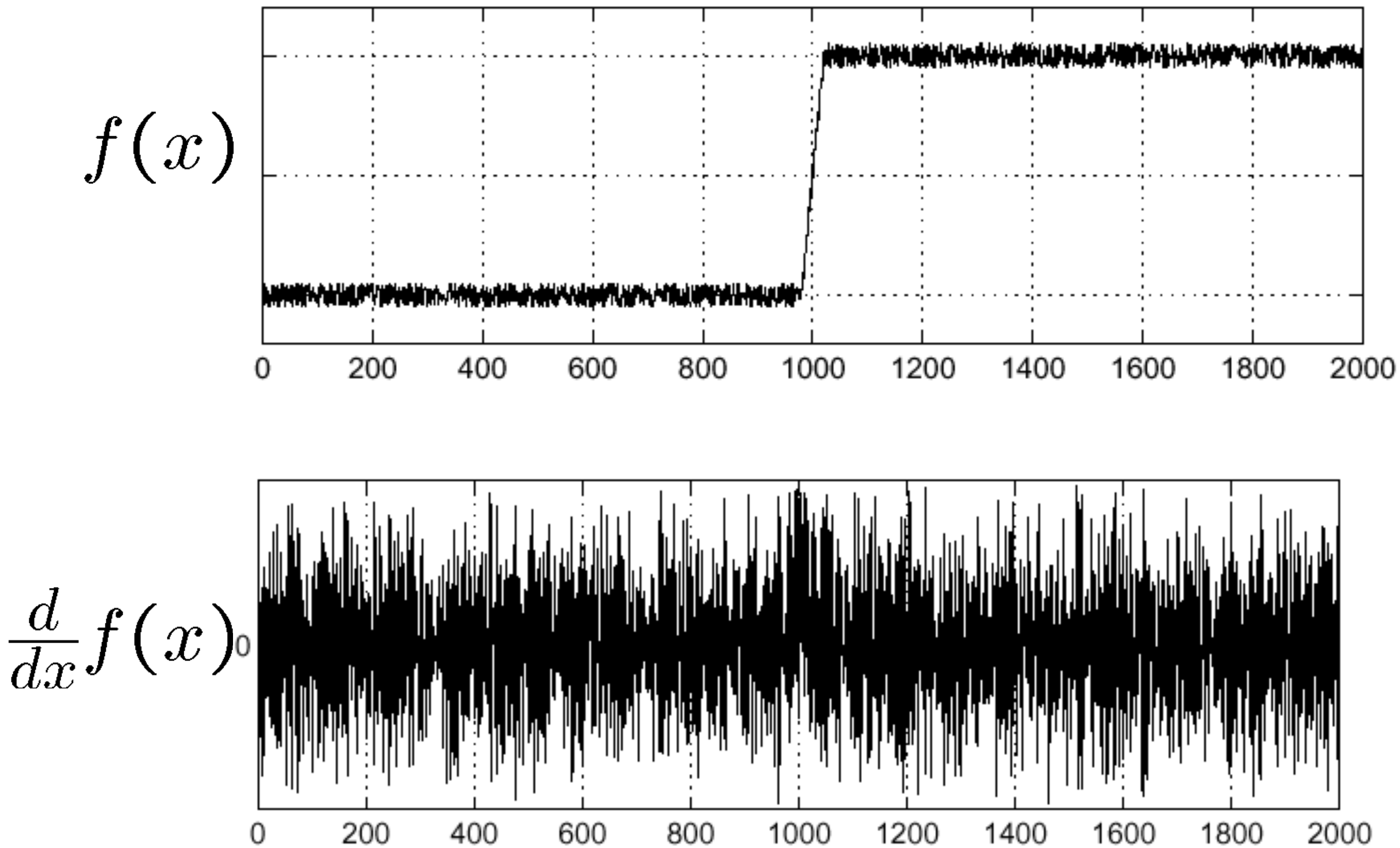
The edge strength is given by the gradient magnitude

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

Source: Steve Seitz

Effects of noise

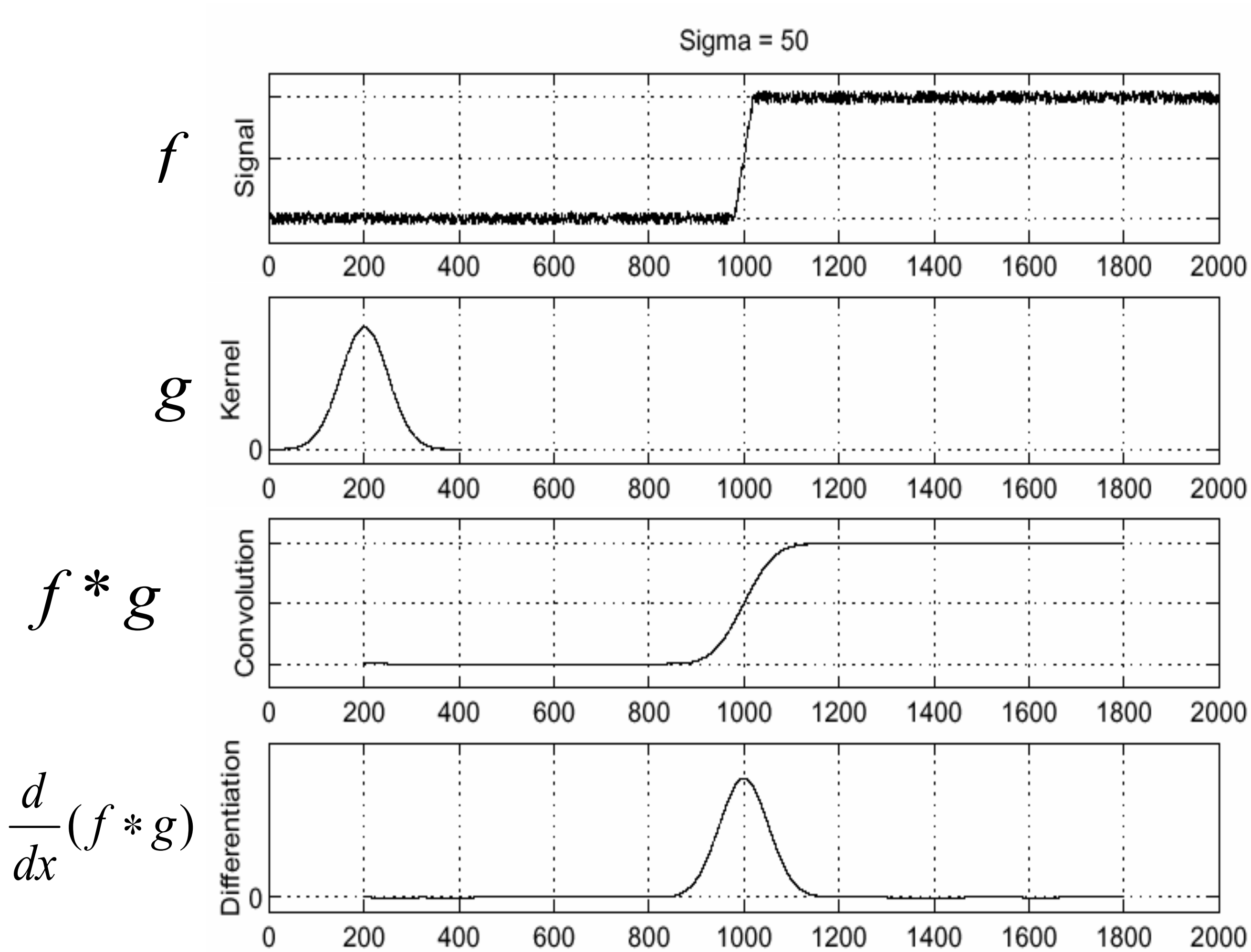
Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

Solution: smooth first



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

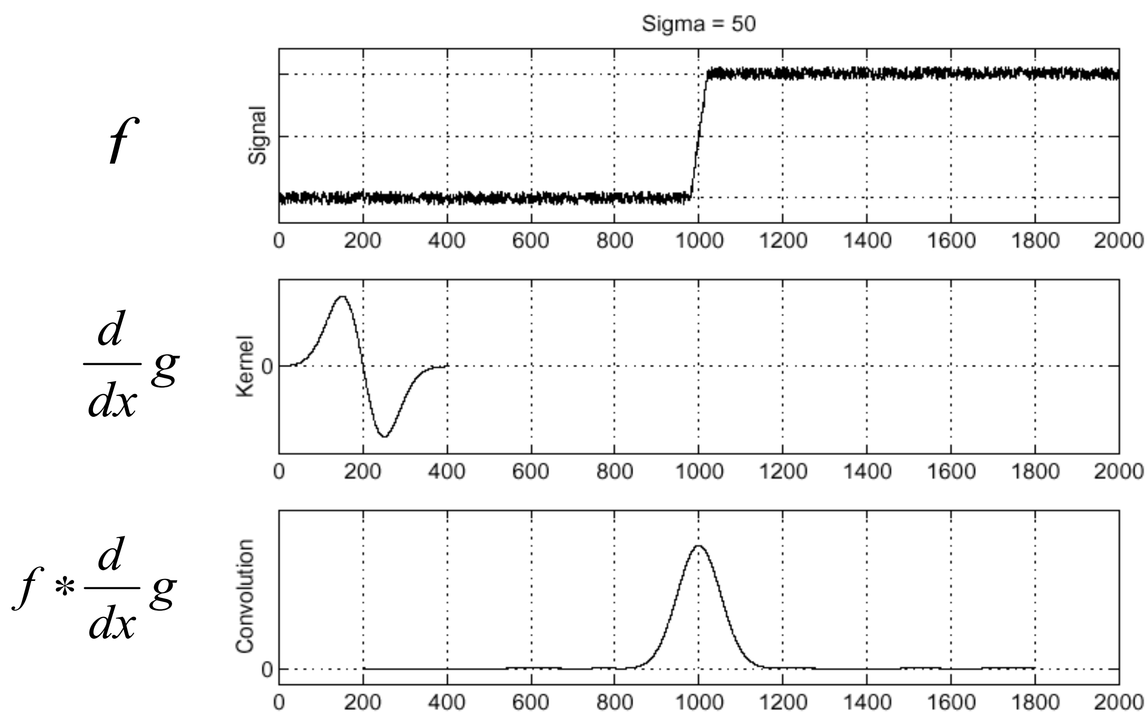
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

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

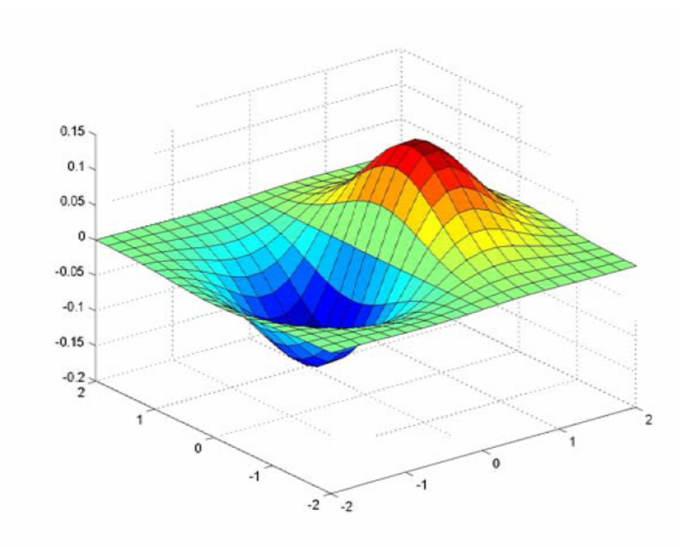
- This saves us one operation:



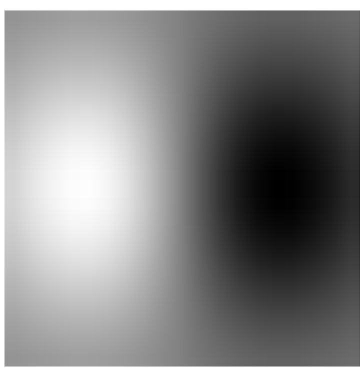
Source: S. Seitz

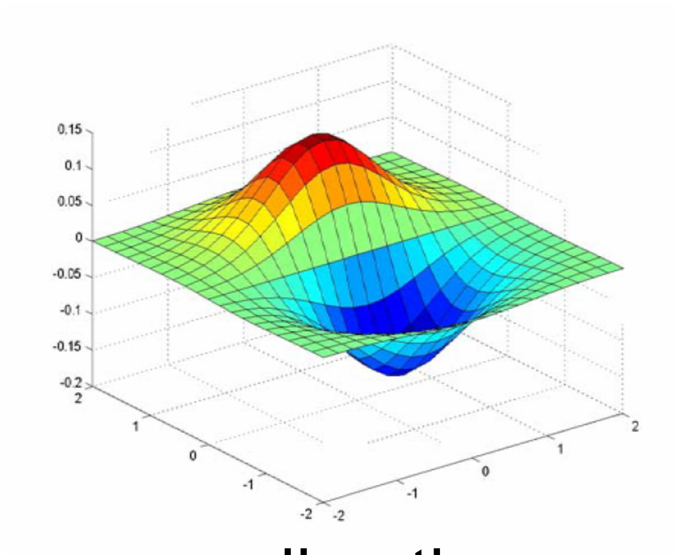
Derivative of Gaussian filters

Which one finds horizontal/vertical edges?

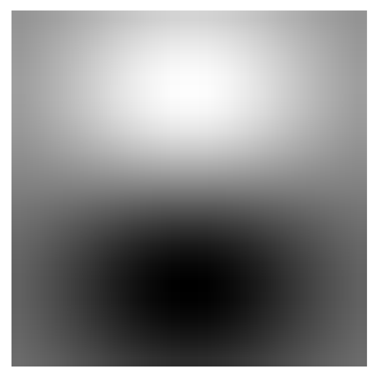


x-direction





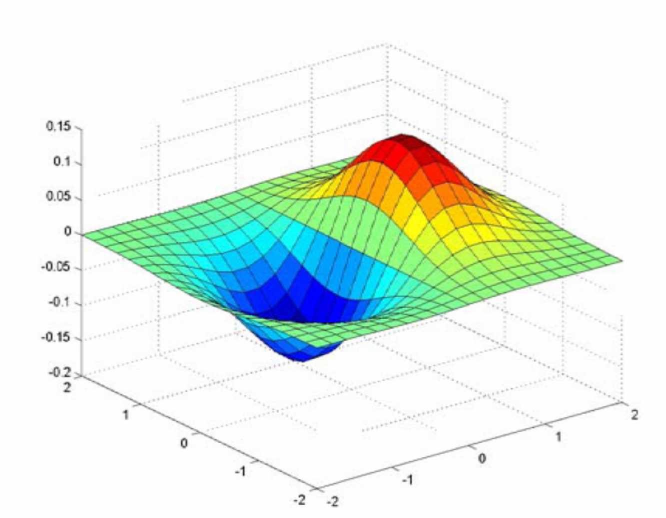
y-direction

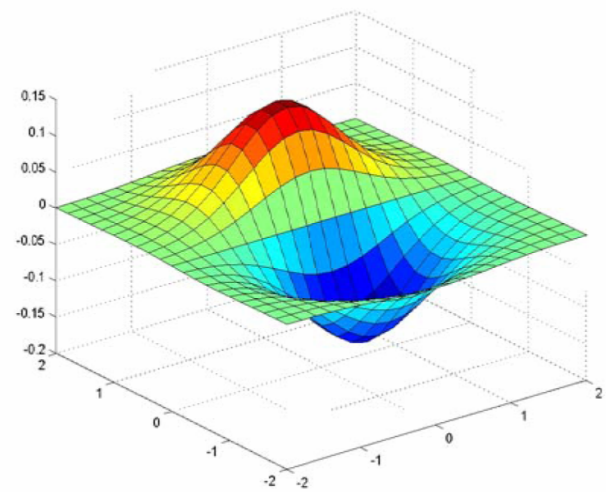


Source: S. Lazebnik

Derivative of Gaussian filters

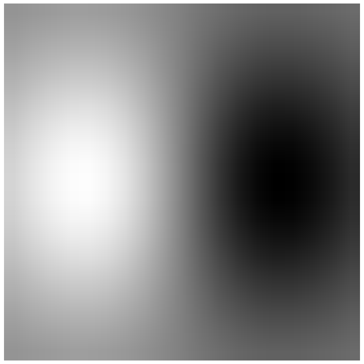
Are these filters separable?

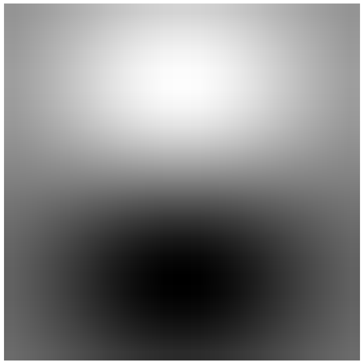




x-direction

y-direction





Source: S. Lazebnik

Recall: separability of the Gaussian filter

$$
\begin{aligned}
G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right)
\end{aligned}
$$

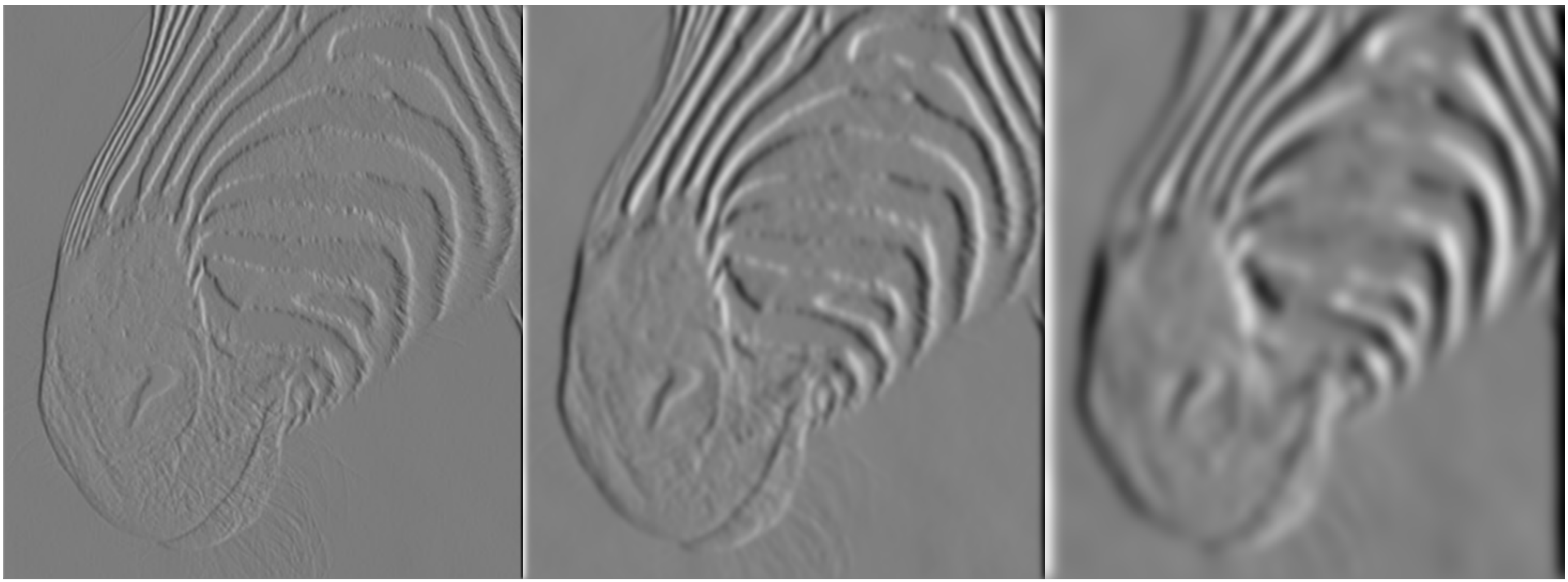
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

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

Source: D. Lowe

Scale of Gaussian derivative filter

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



1 pixel 3 pixels 7 pixels

Source: D. Forsyth

Review: Smoothing vs. derivative filters

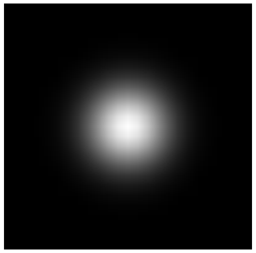
Smoothing filters

– 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

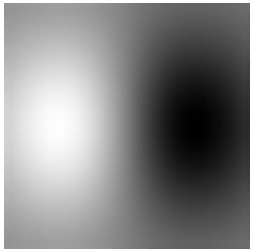
– Derivatives of Gaussian

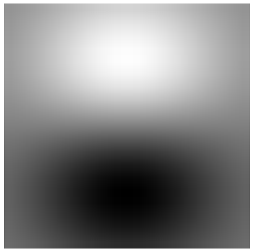
– Can the values of a derivative filter be negative?

– What should the values sum to?

  • **Zero:** no response in constant regions

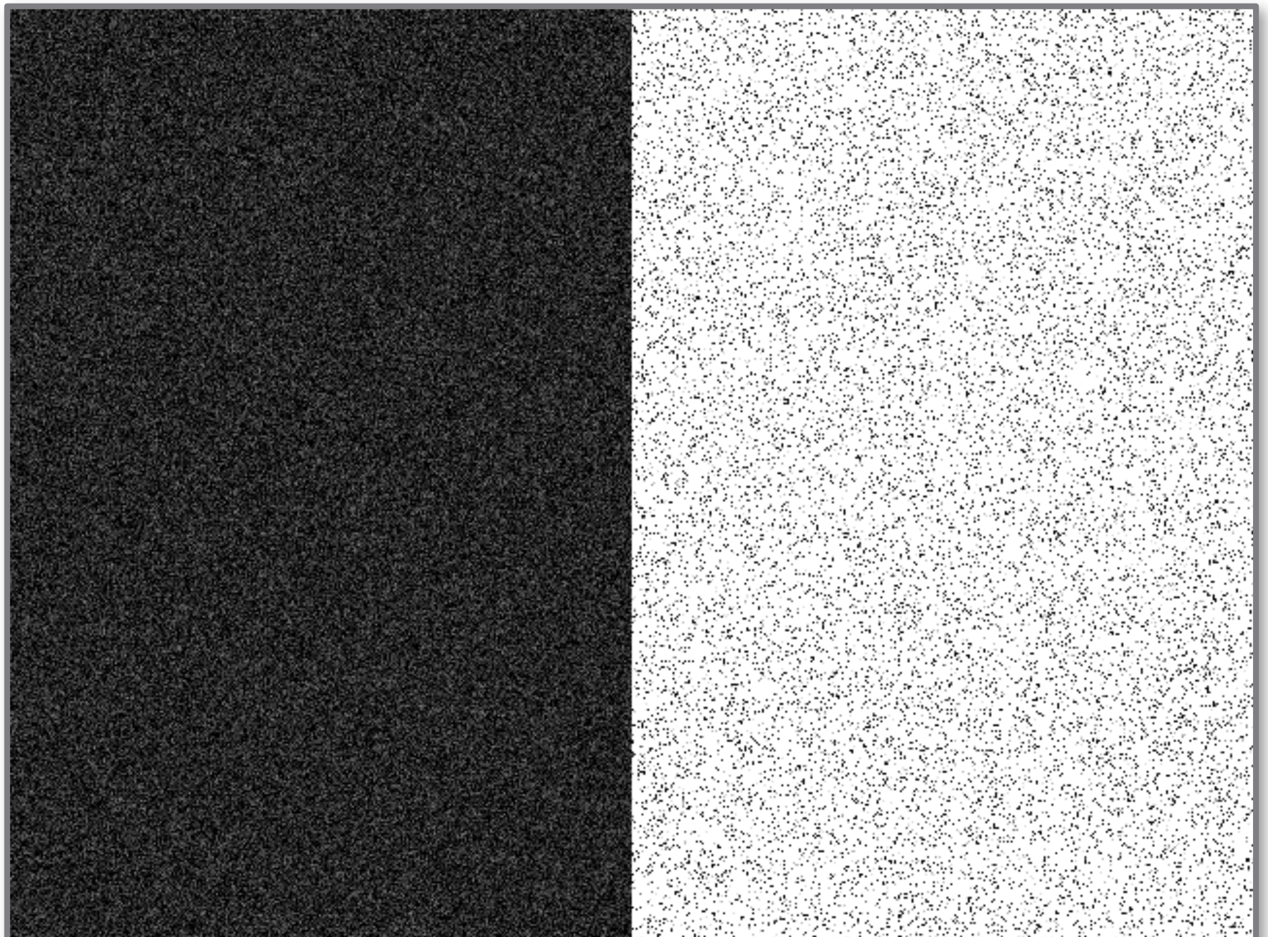
– High absolute value at points of high contrast





Edge detection algorithms

What is an Edge?



Noise: have to distinguish noise from actual edge

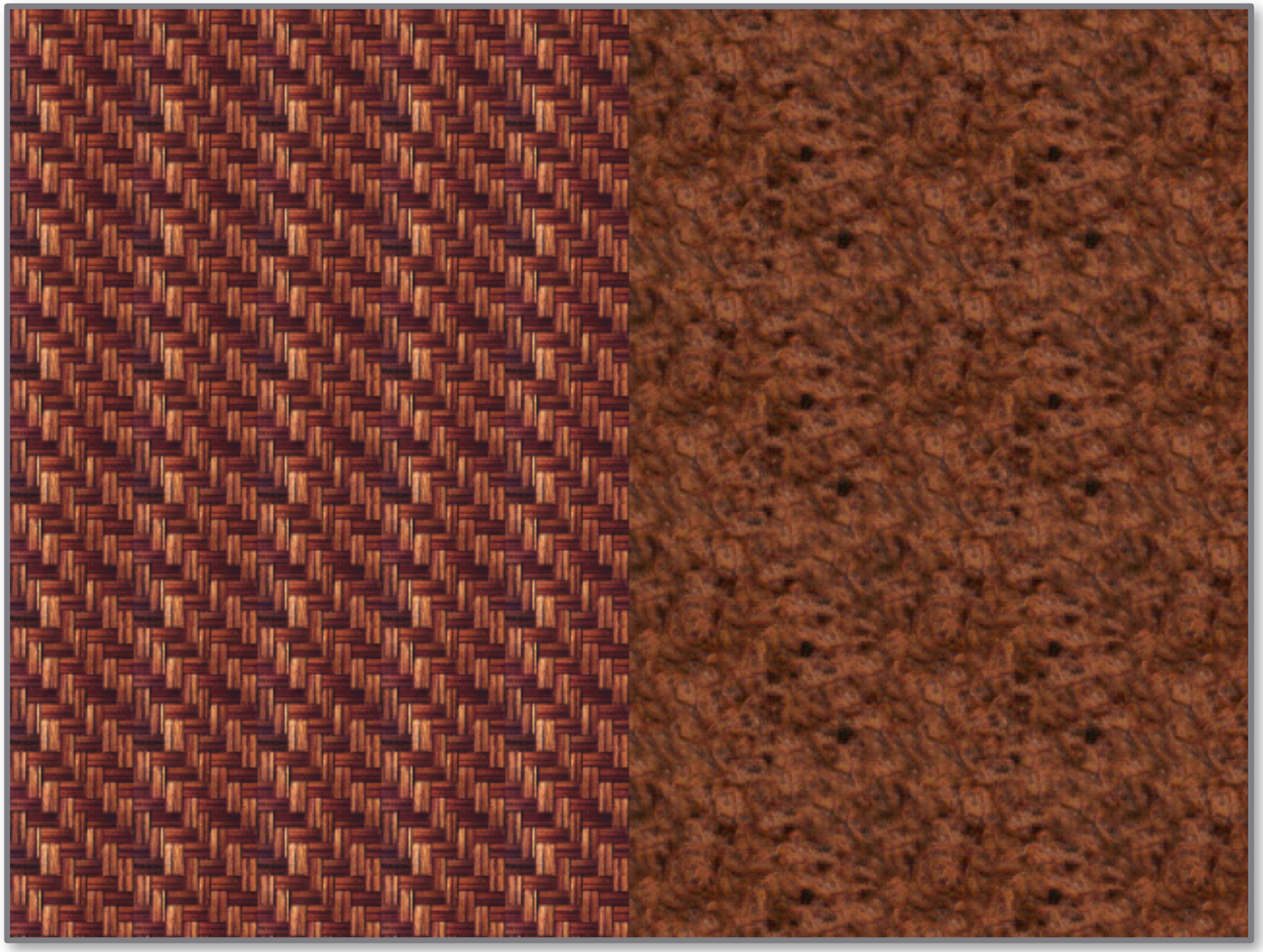
Source: S. Rusinkiewicz

What is an Edge?

Is this one edge or two?

Source: S. Rusinkiewicz

What is an Edge?



Texture discontinuity

Source: S. Rusinkiewicz

The Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. …
4. …

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

The Canny edge detector

magnitude of the gradient

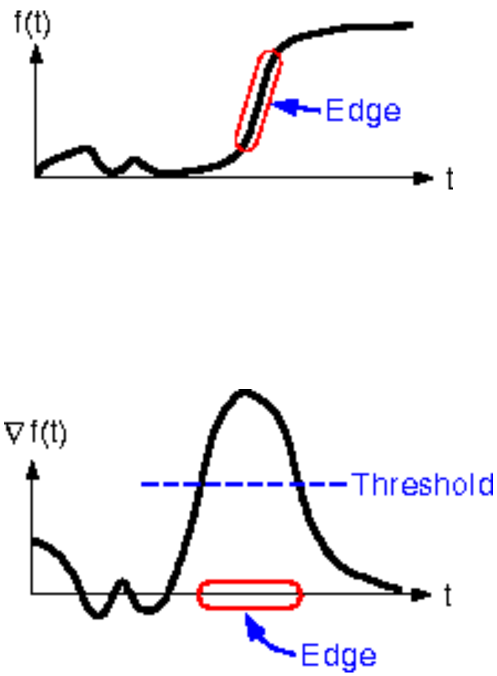
Slide credit: Steve Seitz

The Canny edge detector

thresholding

Slide credit: Steve Seitz

The Canny edge detector



thresholding

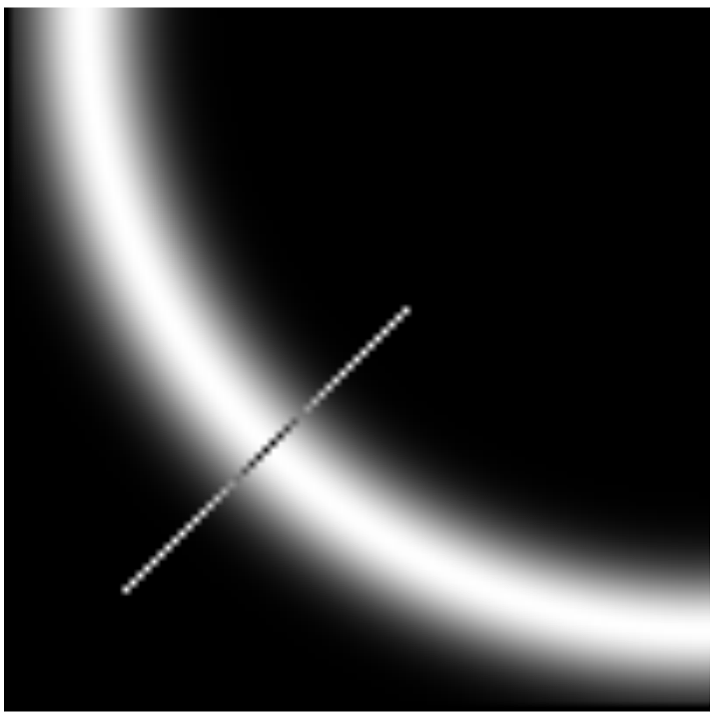
How to turn these thick regions of the gradient into curves?

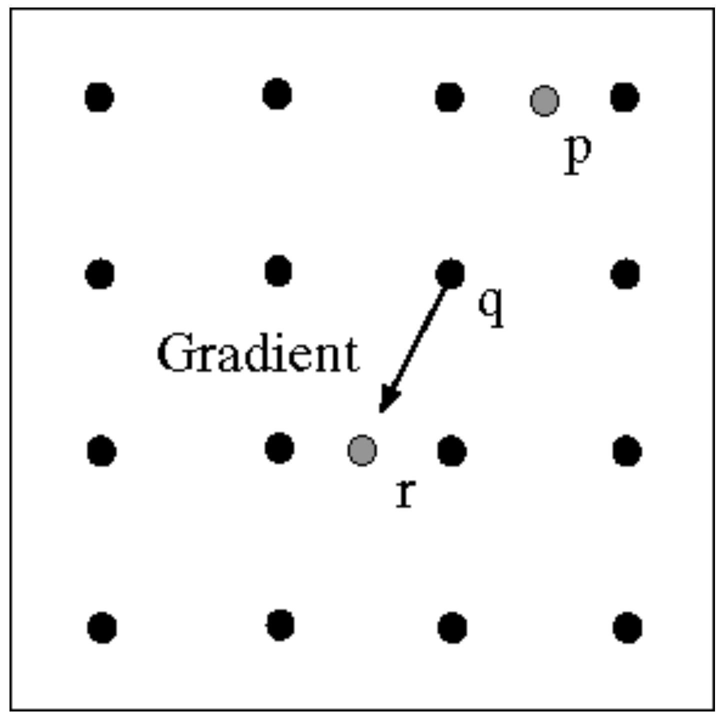
Slide credit: Steve Seitz

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





Slide credit: Steve Seitz

Non-maximum suppression

Slide credit: Steve Seitz

Non-maximum suppression

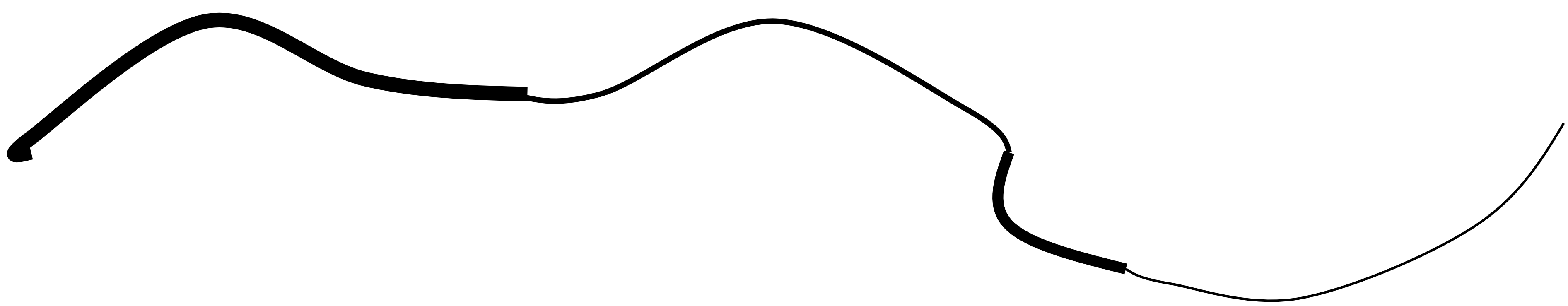
thinning
(non-maximum suppression)

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

Slide credit: Steve Seitz

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

Canny edge detector

1. Compute x and y gradient images
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   – Thin wide "ridges" down to single pixel width
4. 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

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

Slide credit: S. Lazebnik

Faster Edge Detectors

- Can build simpler, faster edge detector by omitting some steps:
  - No nonmaximum suppression
  - No hysteresis in thresholding
  - Simpler filters (approx. to gradient of Gaussian)
    - Sobel: $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$
    - Roberts: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

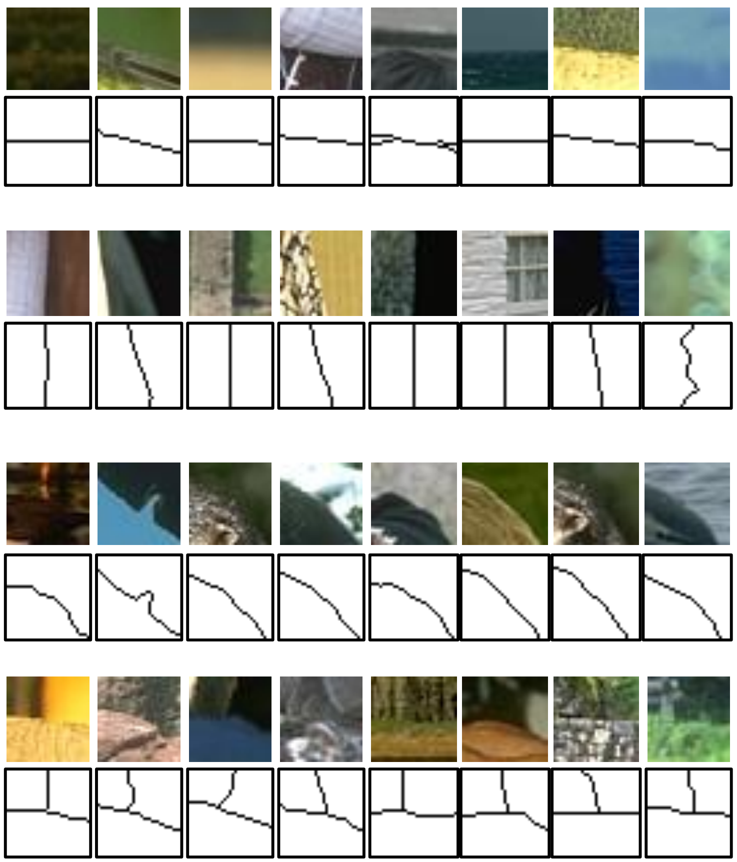
Source: S. Rusinkiewicz

Image gradients vs. meaningful contours

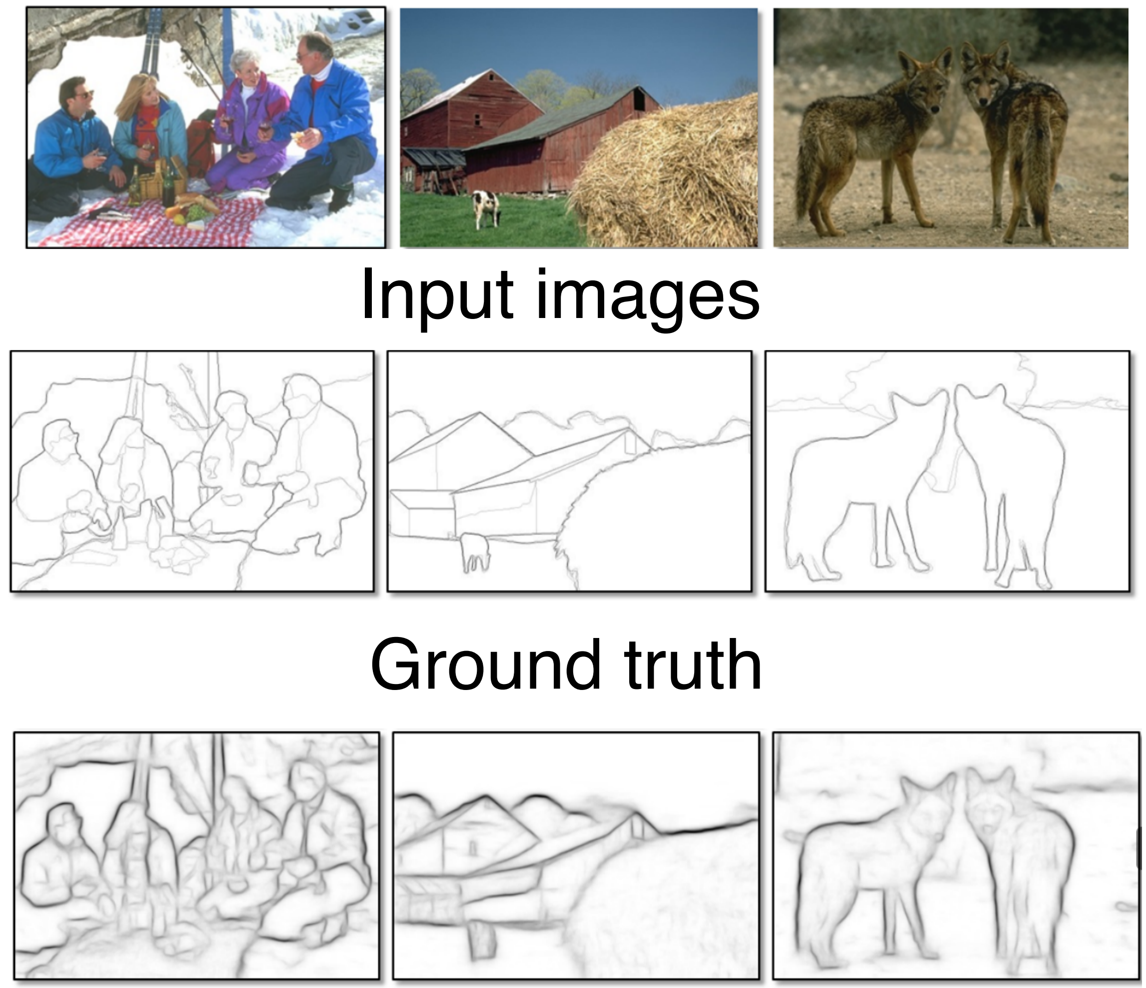
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

image | human segmentation | gradient magnitude

Data-Driven Edge Detection



Training data



Output

P. Dollar and L. Zitnick, [Structured forests for fast edge detection](), ICCV 2013

Next class: feature detectors and descriptors