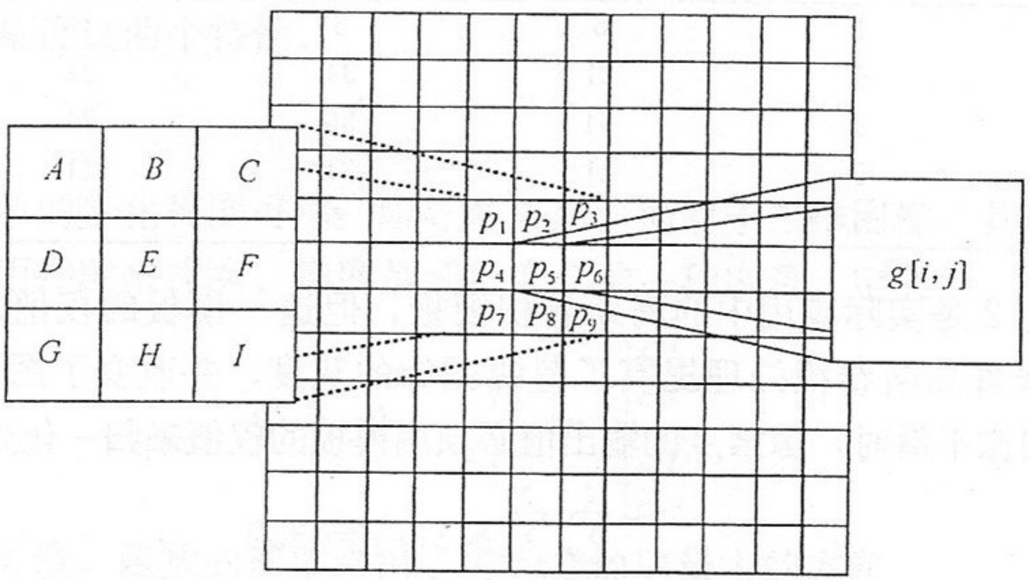
Mask Operation

- Example for 3x3 mask operator

$$g[i,j] = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9 \quad (3.3)$$



Convolution operations

Mask Operations on Images

0	1	3	4	5	7	8	9
9	3	4	5	3	2	6	7
5	6	5	7	6	4	5	8
5	8	0	9	5	5	5	3
2	2	9	6	4	6	9	2
4	5	3	8	3	0	9	6
7	7	7	7	2	2	4	7

1/2	1/3	¼
1/9	1/9	1/9
1/5	1/6	1/7

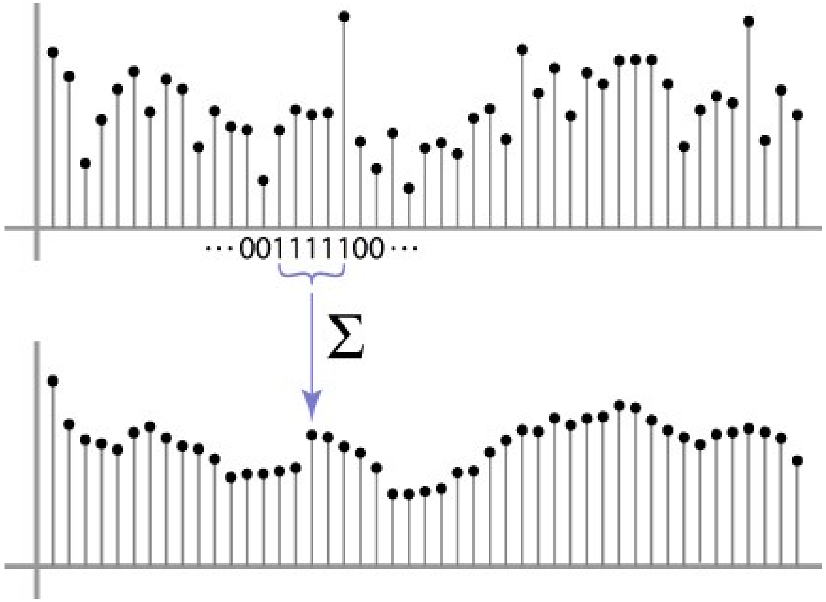
	x						
	y						

$$I(x) = 0\times\frac{1}{2} + 1\times\frac{1}{3} + 3\times\frac{1}{4} + 9\times\frac{1}{9} + 3\times\frac{1}{9} + 4\times\frac{1}{9} + 5\times\frac{1}{5} + 6\times\frac{1}{6} + 5\times\frac{1}{7}$$

$$I(y) = 2\times\frac{1}{2} + 2\times\frac{1}{3} + 9\times\frac{1}{4} + 4\times\frac{1}{9} + 5\times\frac{1}{9} + 3\times\frac{1}{9} + 7\times\frac{1}{5} + 7\times\frac{1}{6} + 7\times\frac{1}{7}$$

Weighted Moving Average

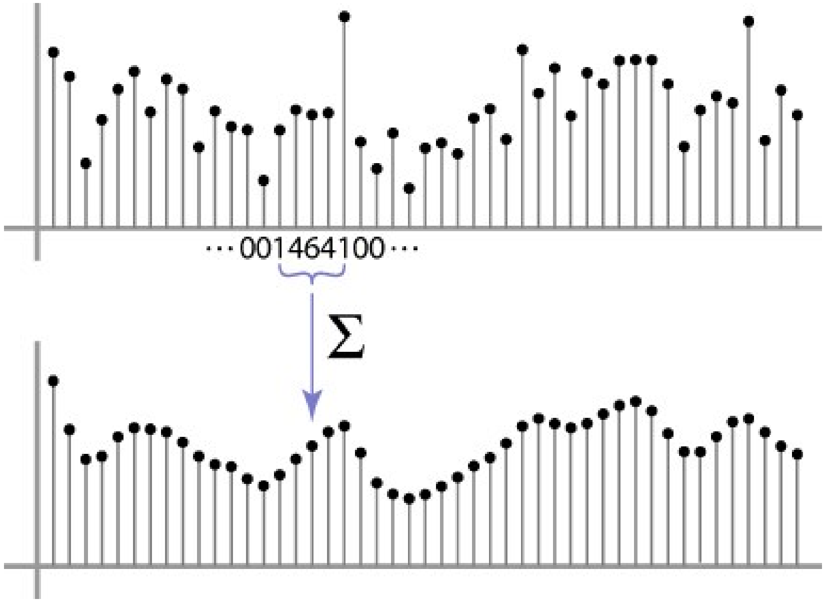
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0								

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

	0	10							

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	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	

Source: S. Seitz

Generalization of 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 a 3x3 moving average?

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

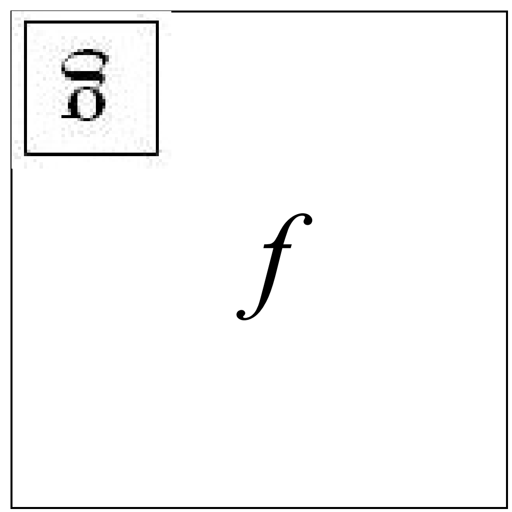
"box filter"

Source: D. Lowe

Defining convolution

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

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



- Convention: kernel is “flipped”
- MATLAB: conv2 vs. filter2 (also imfilter)

Source: F. Durand

Operations on Pictures

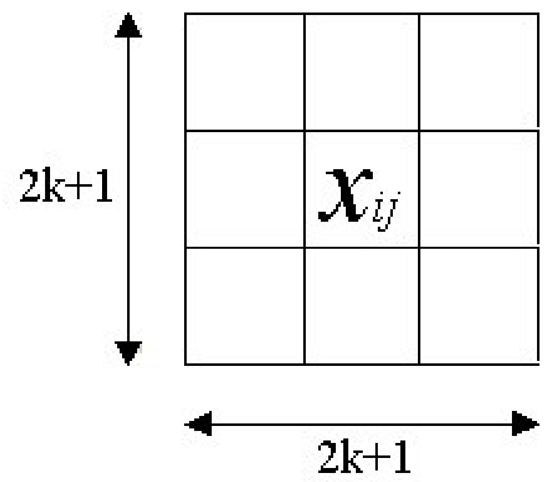
Mask Operation = Convolution Operation

- Convolution
  - A linear filtering process using the filter *m*

$$
\begin{aligned}
g(x,y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(a,b)m(x-a,y-b)\,dadb \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} m(a,b)f(x-a,y-b)\,dadb \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a,y-b)m(a,b)\,dadb \\
&= (f*m)(x,y) \\
&= (m*f)(x,y)
\end{aligned}
$$

I. Noise Reduction

1a) Averaging

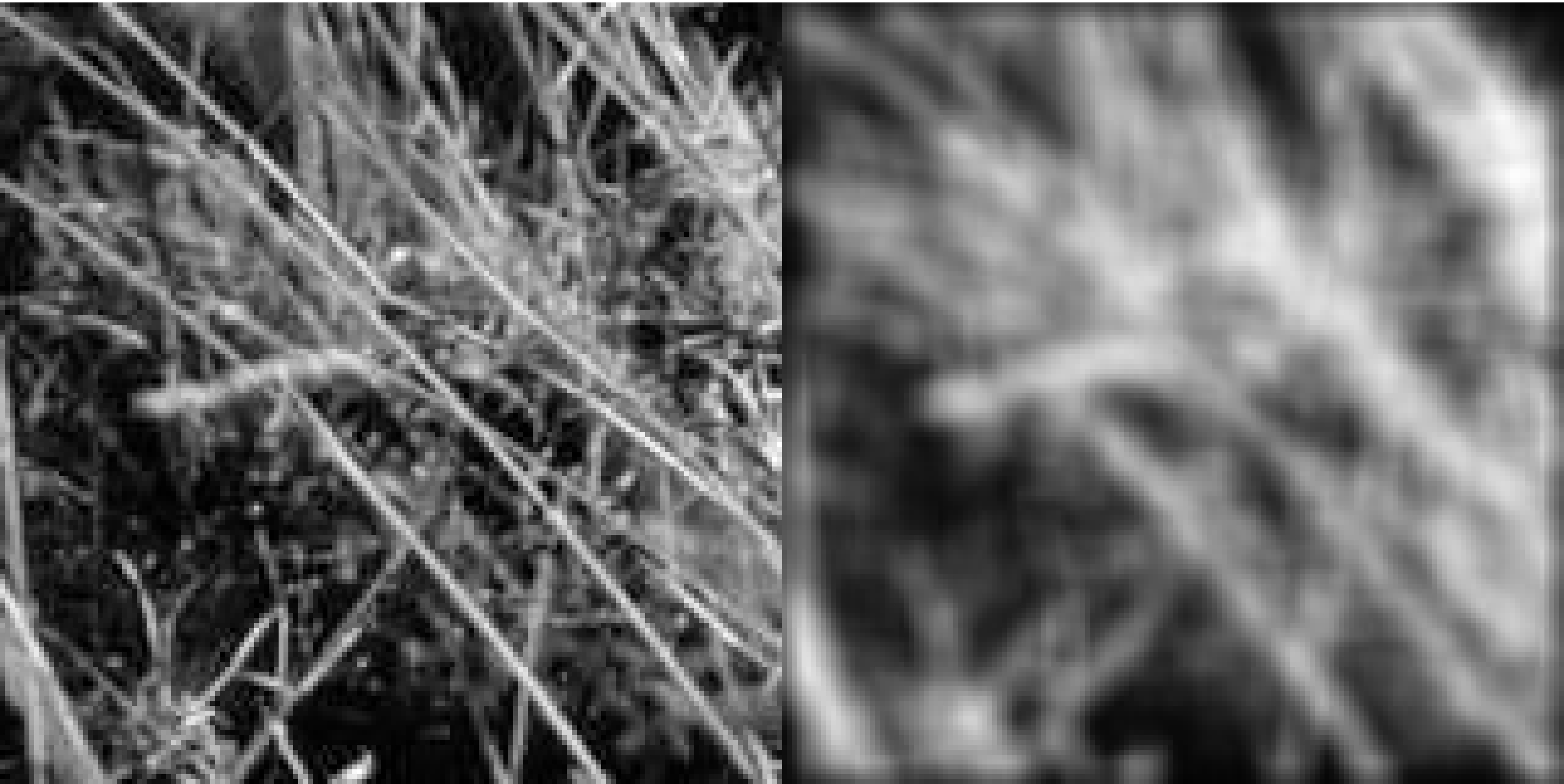


$$y_{ij} = \sum_{m=-k,n=-k}^{k} a_{mn} x_{i+m,j+n}$$

$$\text{window size} = (2k+1)\times(2k+1)$$

$$\sum_{\mathrm{m,n}} a_{mn} = 1$$

Example: Smoothing by Averaging



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

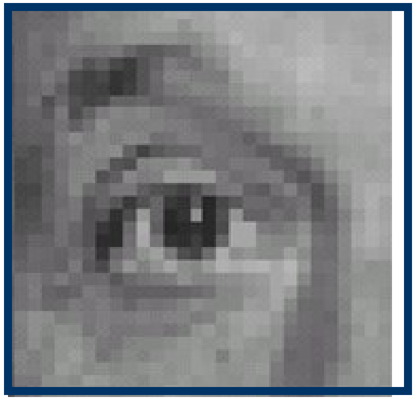
?

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

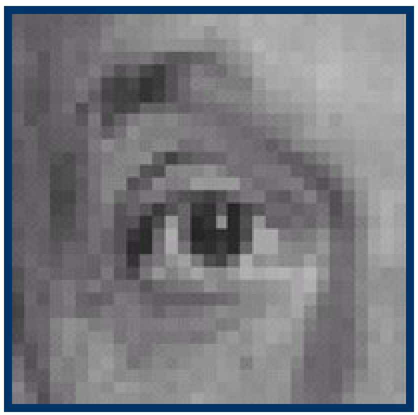
Original

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

?

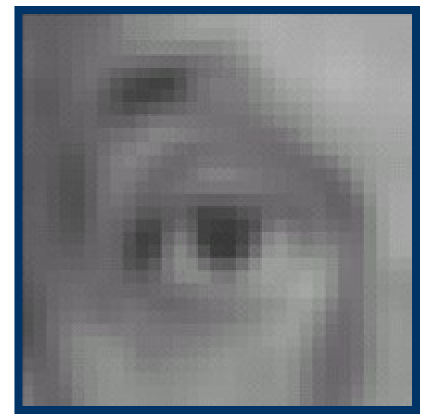
Source: D. Lowe

Practice 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

Practice with linear filters

Original

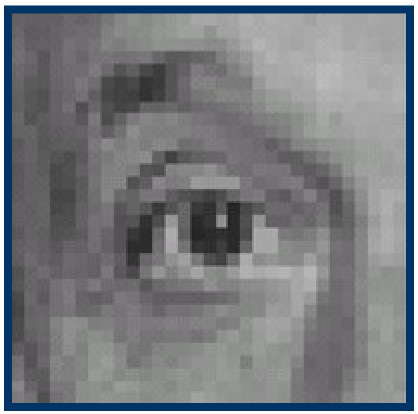
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters



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

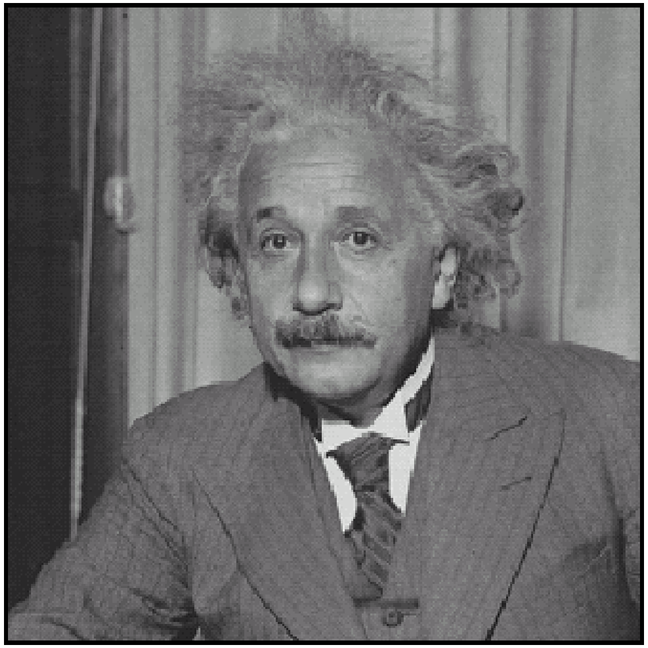
Original

Sharpening filter

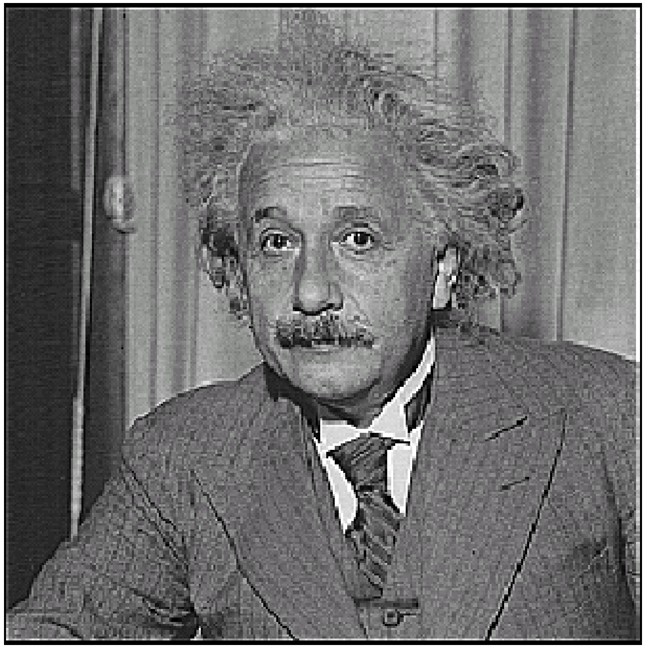
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before



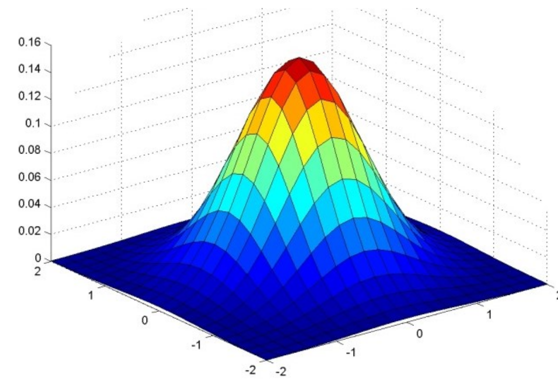
after

Source: D. Lowe

Mask for Gaussian Function

$$h[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

- σ值越大，平滑程度越好，但同時也造成影像特徵模糊，一般取σ=1~10。
- When σ=1, the mask becomes



5×5

h[i,j]	-2	-1	0	1	2
-2	0.018	0.082	0.135	0.082	0.018
-1	0.082	0.368	0.607	0.368	0.082
0	0.135	0.607	1.000	0.607	0.135
1	0.082	0.368	0.607	0.368	0.082
2	0.018	0.082	0.135	0.082	0.018

Gaussian Mask

Mask for Gaussian Function

- Integer mask will be better for computation
- Choose the minimum of h[i,j] to normalize

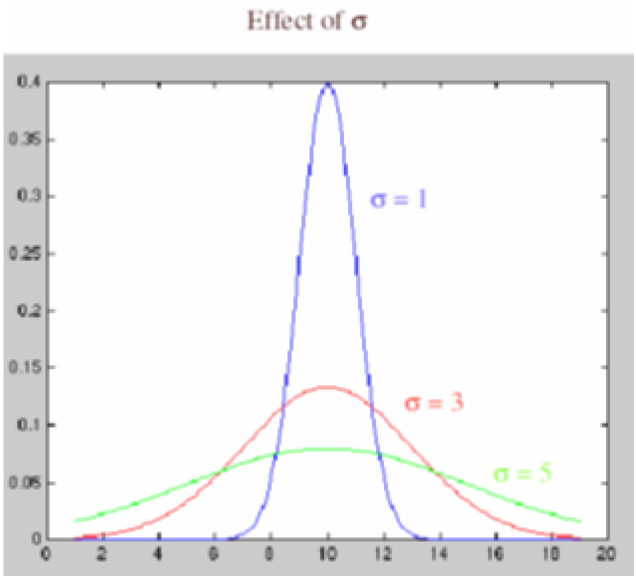
$$c = \frac{h[-2,-2]}{0.018} = \frac{1}{0.018} = 56$$

[i,j]	-2	-1	0	1	2
-2	1	5	8	5	1
-1	5	21	34	21	5
0	8	34	56	34	8
1	5	21	34	21	5
2	1	5	8	5	1

Integer mask for Gaussian function

Mask for Gaussian Function

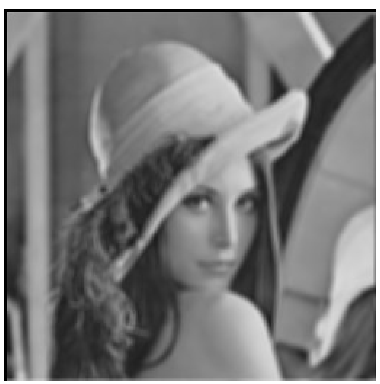
- Sum of the weights should be 1
- Normalization by $\sum_{i=-2}^{2}\sum_{j=-2}^{2} h[i,j] = 352$
- EX：



$$g[i,j] = \frac{1}{352}\left(f[i,j] \otimes h[i,j]\right)$$

(a) Image

(b) σ=1



(c) σ=5

Edge detection

Winter in Kraków photographed by Marcin Ryczek

Edge detection

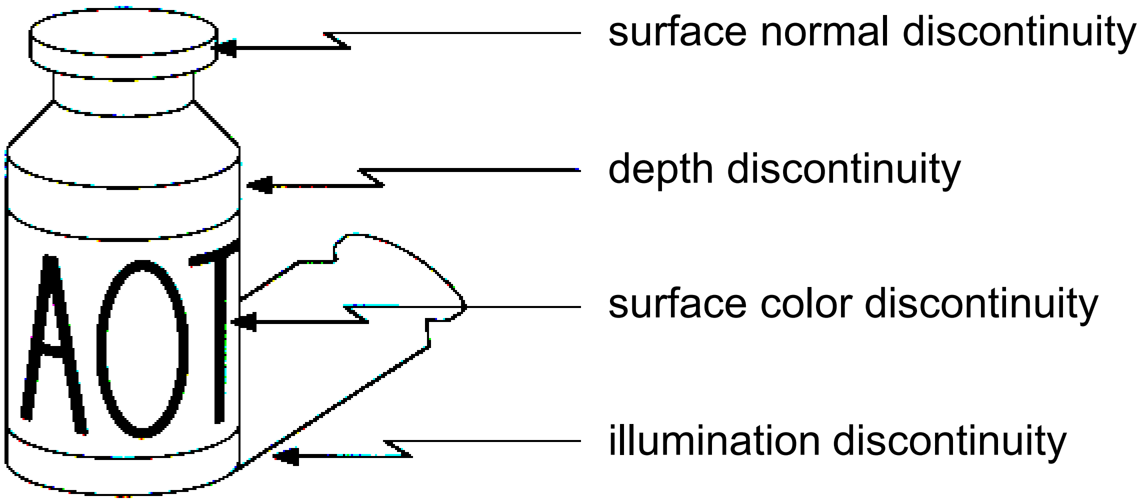
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

Origin of edges

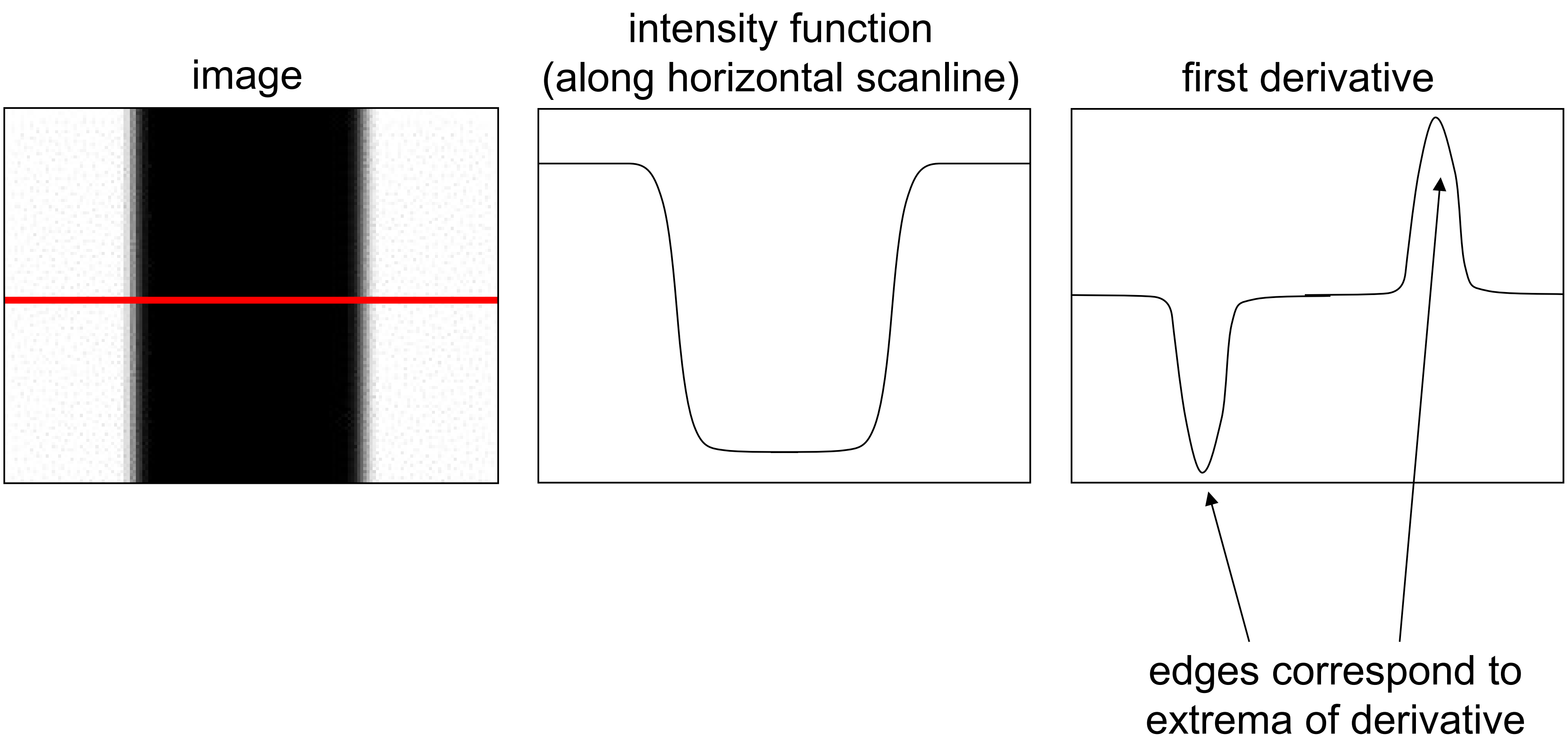
Edges are caused by a variety of factors:



Source: Steve Seitz

Characterizing edges

- 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 above as convolution, what would be the associated filter?

Source: K. Grauman

Differentiation and convolution

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

$$I(x) \approx \frac{f(x) - f(x-1)}{1} \approx \frac{f(x+1) - f(x)}{1} \approx \frac{f(x+1) - f(x-1)}{2}$$

$$I_x(x,y) \approx \frac{f(x+1, y) - f(x-1, y)}{2}$$

$$I_y(x,y) \approx \frac{f(x, y+1) - f(x, y-1)}{2}$$

Differentiation and convolution

$$I_x(x,y) \approx \frac{f(x+1,y) - f(x-1,y)}{2}$$

$$= 0 \times f(x-1,y-1) + 0 \times f(x,y-1) + 0 \times f(x+1,y-1) +$$

$$\frac{-1}{2} \times f(x-1,y) \quad + 0 \times f(x,y) \qquad + \frac{1}{2} \times f(x+1,y) +$$

$$0 \times f(x-1,y+1) + 0 \times f(x,y+1) \quad + 0 \times f(x+1,y+1)$$

	f(x-1,y-1)	f(x,y-1)	f(x+11,y-1)				
	f(x-1,y)	f(x,y)	f(x+1,y)				
	f(x-1,y+1)	f(x,y+1)	f(x+11,y+1)				

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

Differentiation and convolution

$$I_y(x,y) \approx \frac{f(x,y+1) - f(x,y-1)}{2}$$

$$= 0 \times f(x-1,y-1) + \frac{-1}{2} \times f(x,y-1) + 0 \times f(x+1,y-1) +$$

$$0 \times f(x-1,y) \quad + 0 \times f(x,y) \qquad + 0 \times f(x+1,y) +$$

$$0 \times f(x-1,y+1) + \frac{1}{2} \times f(x,y+1) \;\; + 0 \times f(x+1,y+1)$$

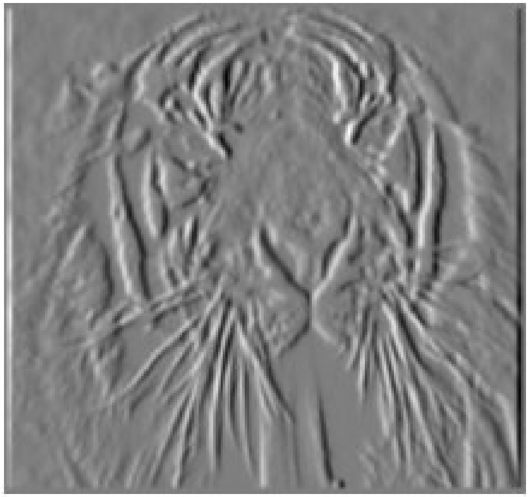
	f(x-1,y-1)	f(x,y-1)	f(x+11,y-1)				
	f(x-1,y)	f(x,y)	f(x+1,y)				
	f(x-1,y+1)	f(x,y+1)	f(x+11,y+1)				

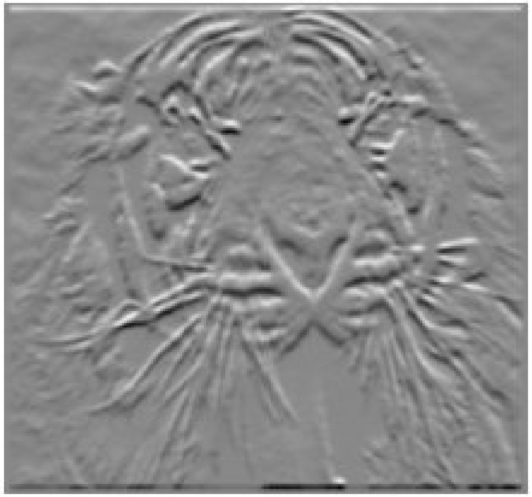
0	-1/2	0
0	0	0
0	1/2	0

Partial derivatives of an image

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

-1	1





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

$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

Which shows changes with respect to x?

Finite differences

Edge detection

Gradients operations:

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

0	-1	-1
1	0	-1
1	1	0

-1	-1	0
-1	0	1
0	1	1

Sobel operator

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Edge的法向量角度:

$$\theta = \arctan(\frac{I_y}{I_x})$$

Finite difference filters

Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

Properties of Gradient

The gradients of $I(x,y)$ are $I_x(x,y)$ and $I_y(x,y)$

For a new position $(x+\Delta\cos\theta, y+\Delta\sin\theta)$,

by Tayler's expansion, we have

$$I(x+\Delta\cos\theta, y+\Delta\sin\theta) \cong I(x,y) + \Delta\cos\theta I_x(x,y) + \Delta\sin\theta I_y(x,y)$$

Set $v \equiv (\Delta\cos\theta, \Delta\sin\theta)$ and $\nabla \mathrm{I}=(I_x, I_y)$.

$$\Rightarrow I(x+\Delta\cos\theta, y+\Delta\sin\theta) - I(x,y) \cong < v, \nabla \mathrm{I} > = |v||\nabla \mathrm{I}|\cos\alpha$$

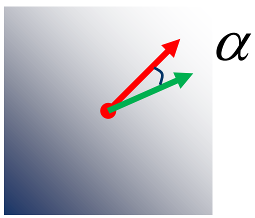
α : the angle between v and $\nabla \mathrm{I}$

$\alpha=0 \Rightarrow < v, \nabla \mathrm{I} >$ is maximized

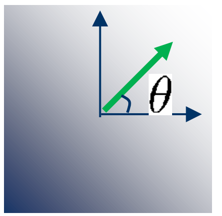
$\Rightarrow v$ and $\nabla \mathrm{I}$ are the same orientation

$< v, \nabla \mathrm{I} >$ is maximized

$\Rightarrow \nabla \mathrm{I}$ is *vertical* to edge orientation



$v \equiv (\Delta\cos\theta, \Delta\sin\theta)$



$$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}) = (I_x, I_y)$$

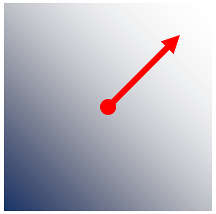
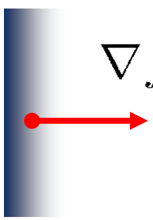


Image gradient

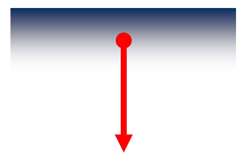
- The gradient of an image:

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

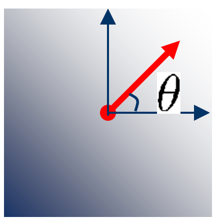
- The gradient points in the direction of most rapid change in intensity



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



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



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

- The gradient direction is given by:

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

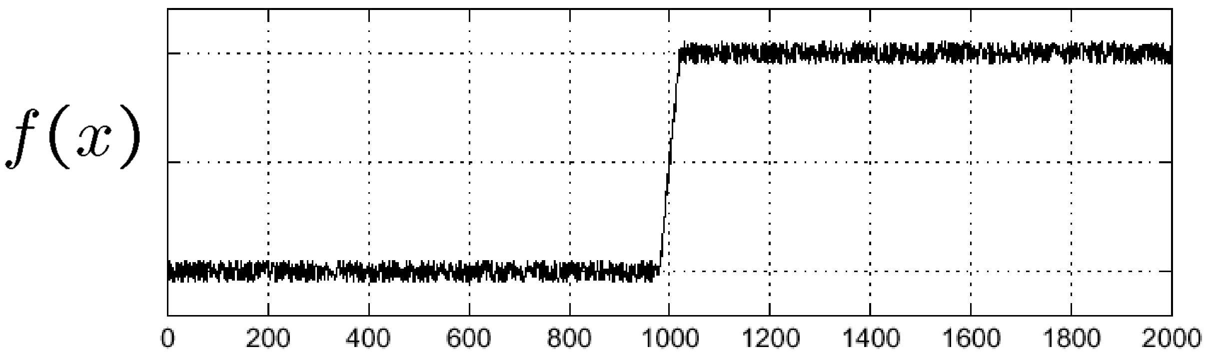
- The *edge strength* is given by the gradient magnitude

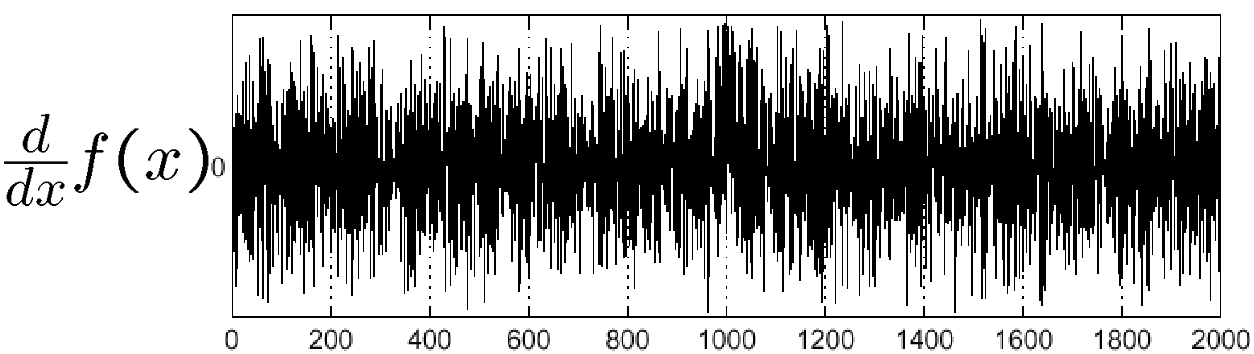
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

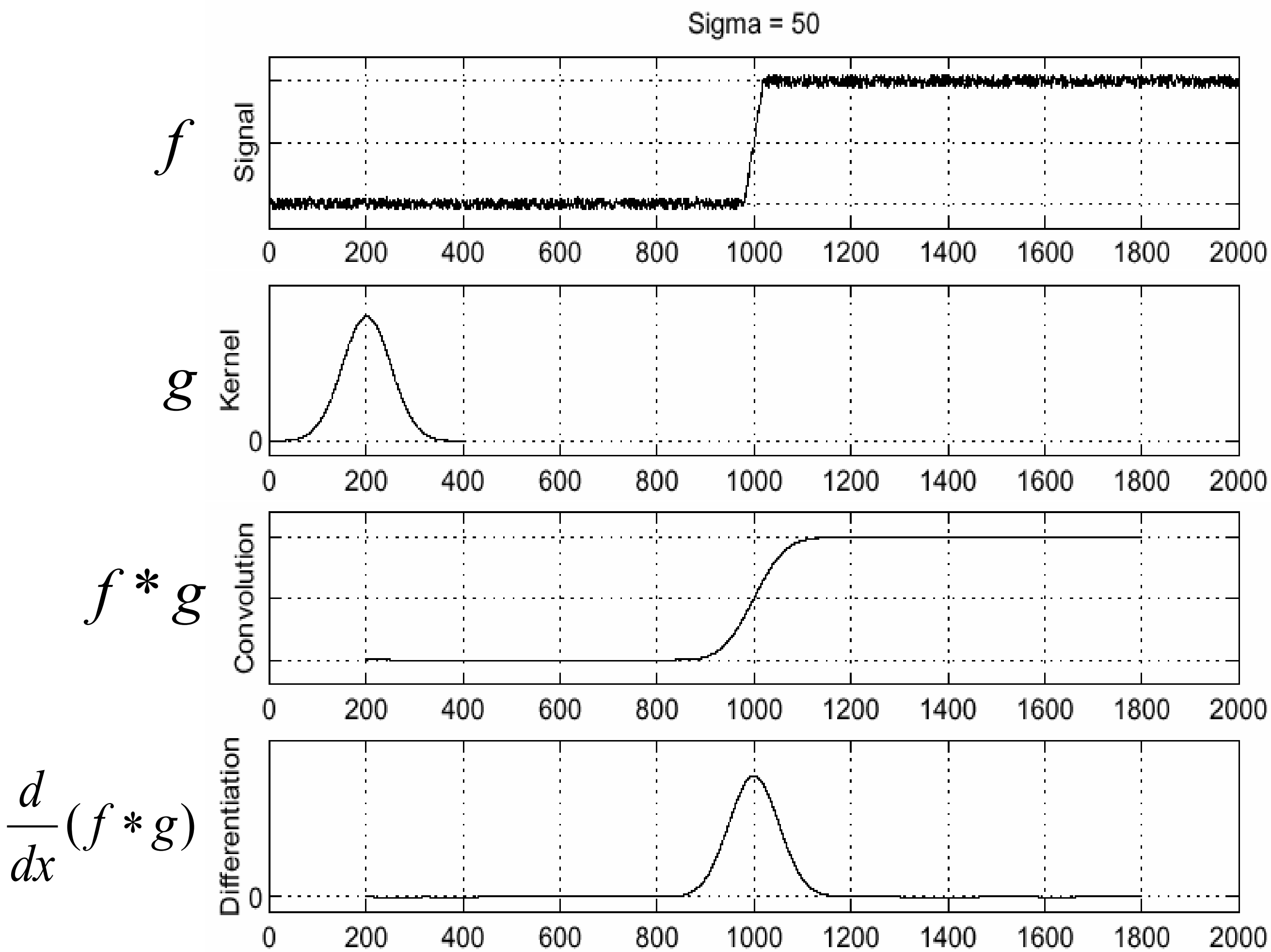




Where is the edge?

Source: S. Seitz

Solution: smooth first

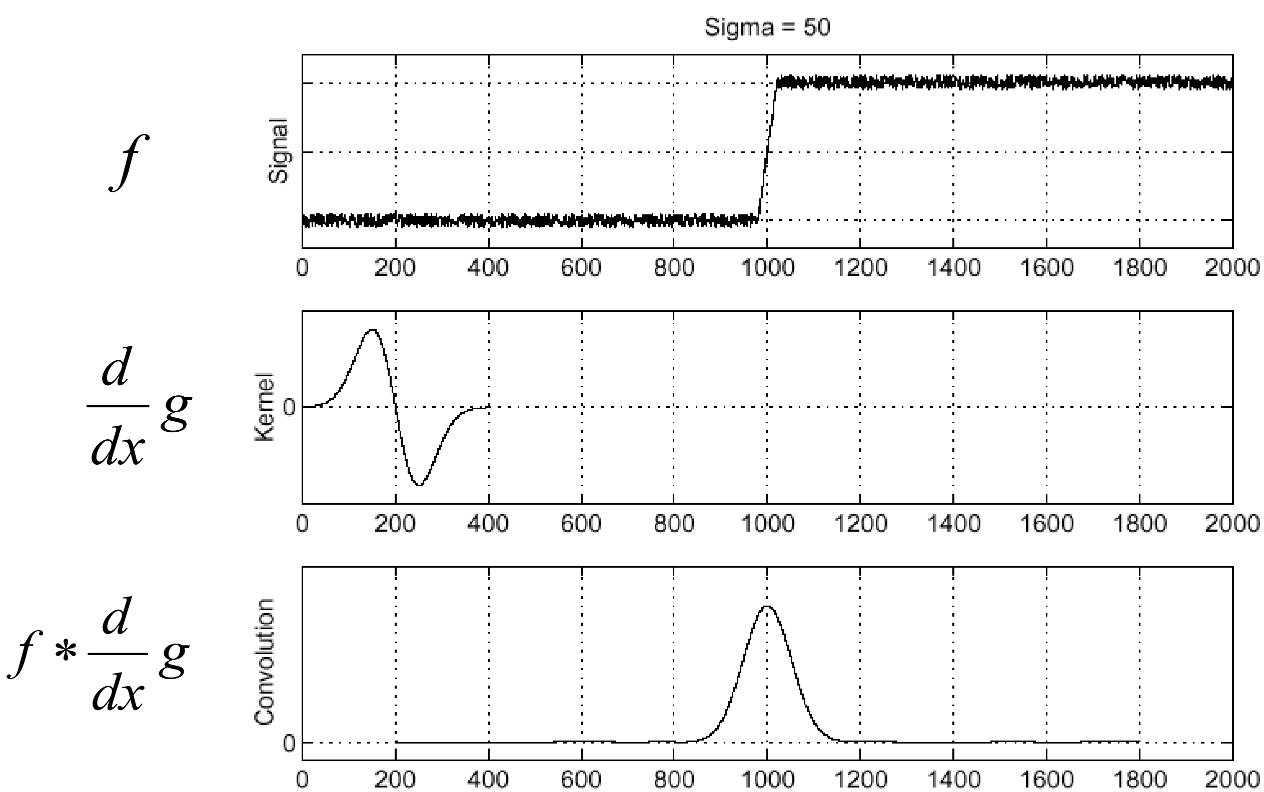


- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

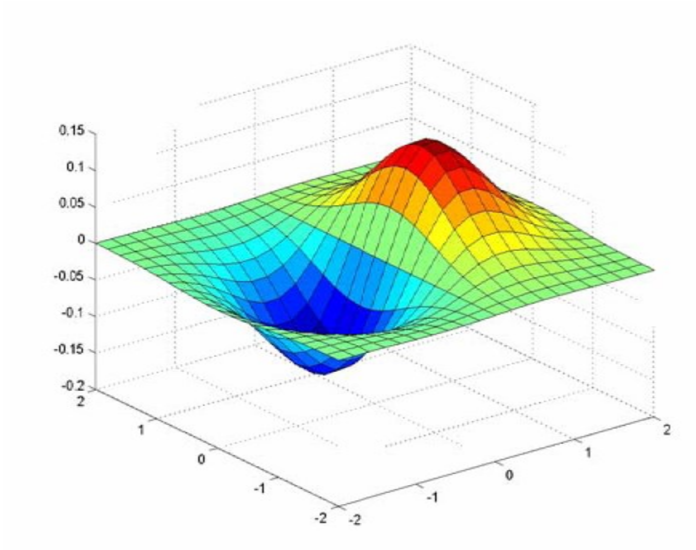
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

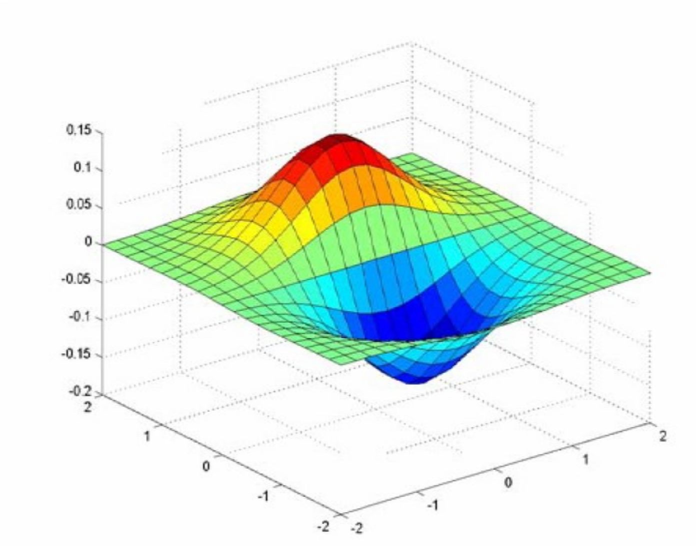


Source: S. Seitz

Derivative of Gaussian filter



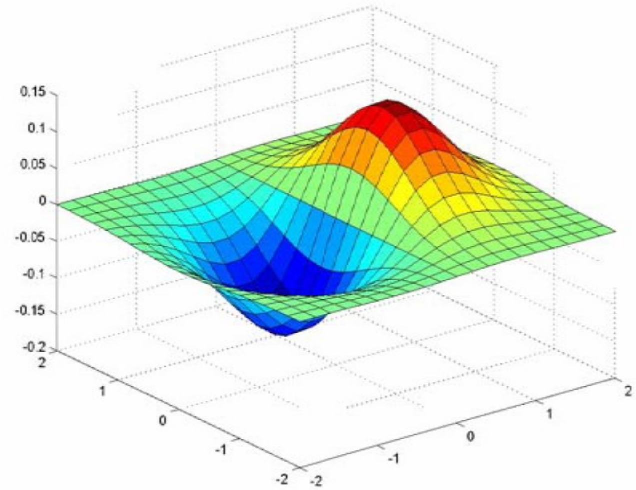
x-direction



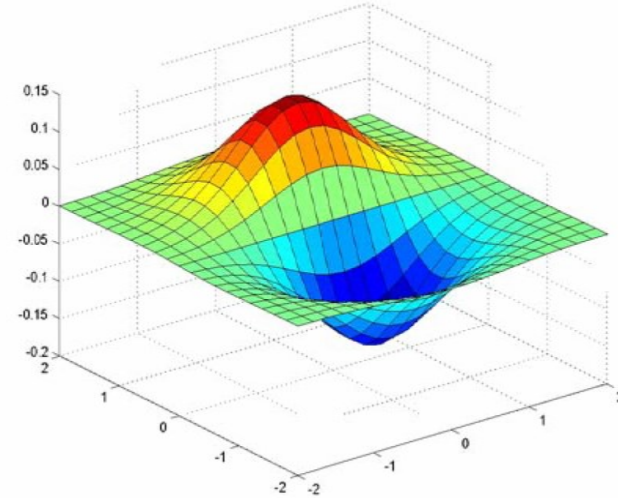
y-direction

Are these filters separable?

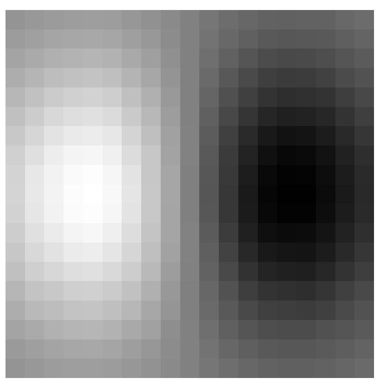
Derivative of Gaussian filter

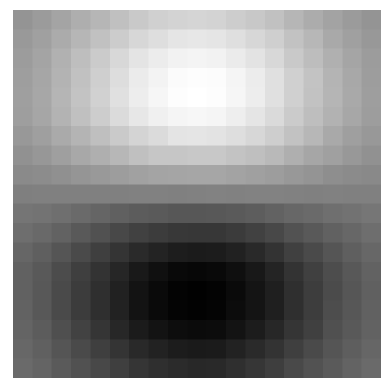


x-direction



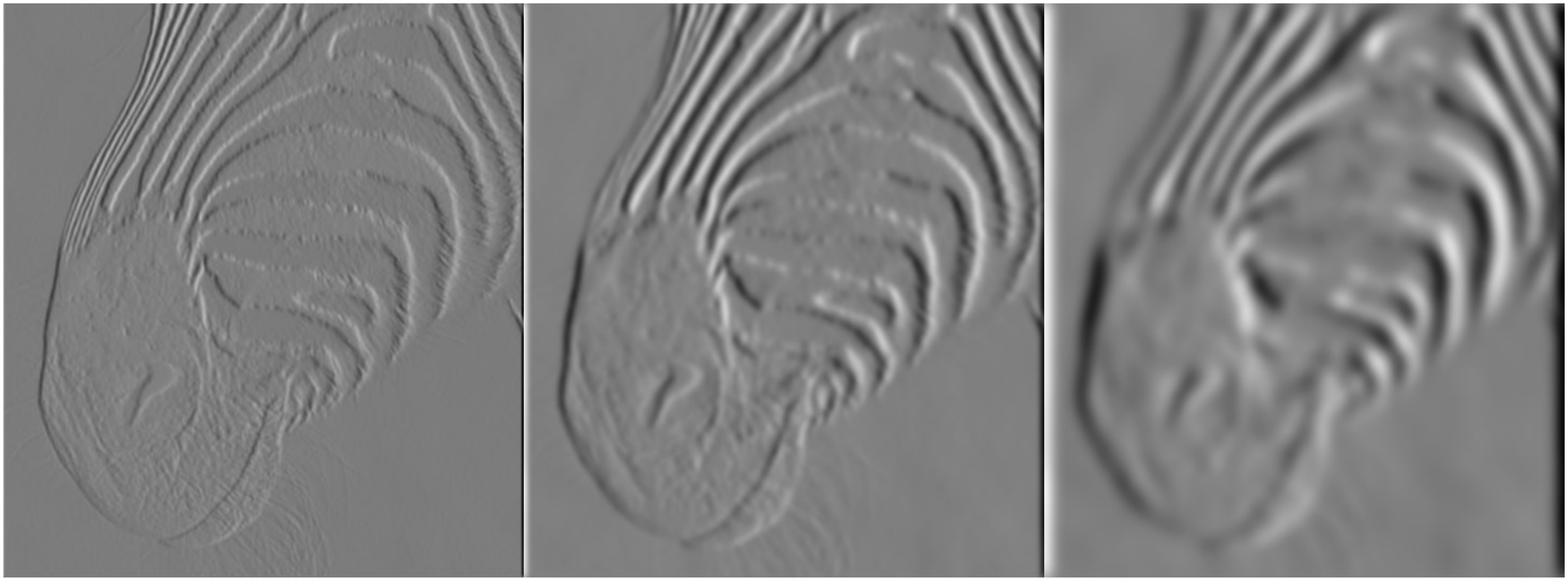
y-direction





Which one finds horizontal/vertical edges?

Scale of Gaussian derivative filter



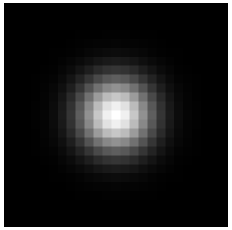
1 pixel 3 pixels 7 pixels

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

Source: D. Forsyth

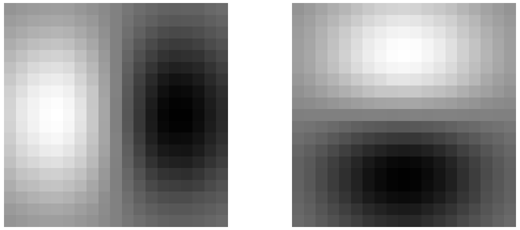
Review: Smoothing vs. derivative filters

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

The Canny edge detector

original image

Slide credit: Steve Seitz

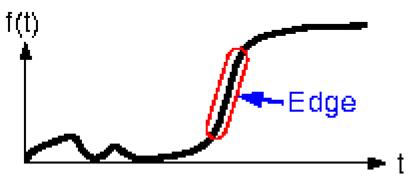
The Canny edge detector

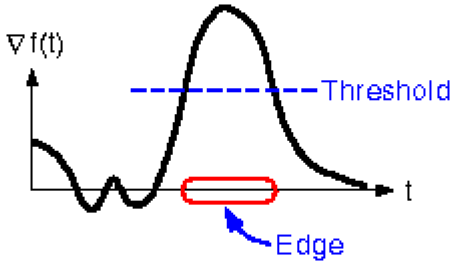
norm of the gradient

The Canny edge detector

thresholding

The Canny edge detector

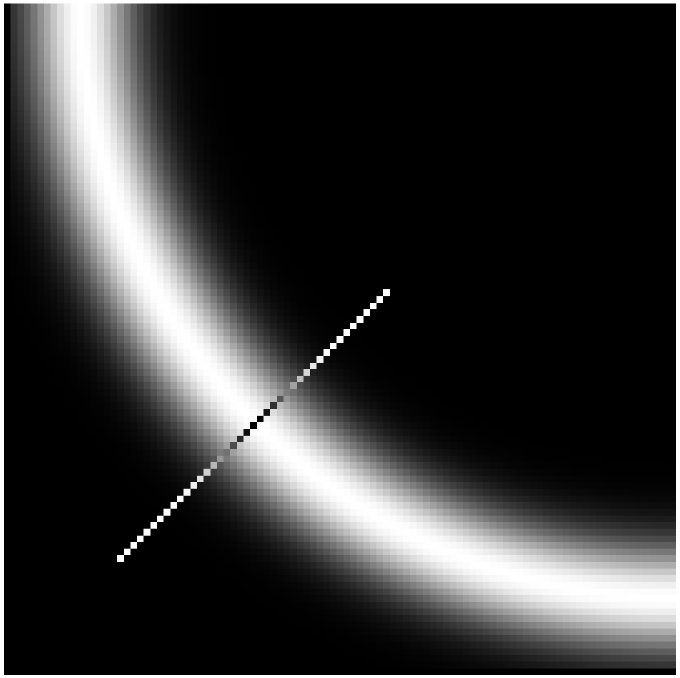


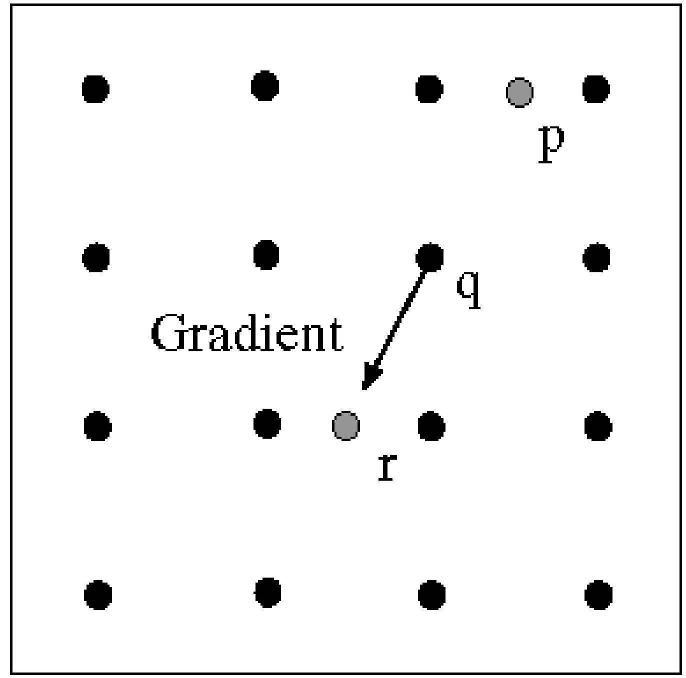


thresholding

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

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

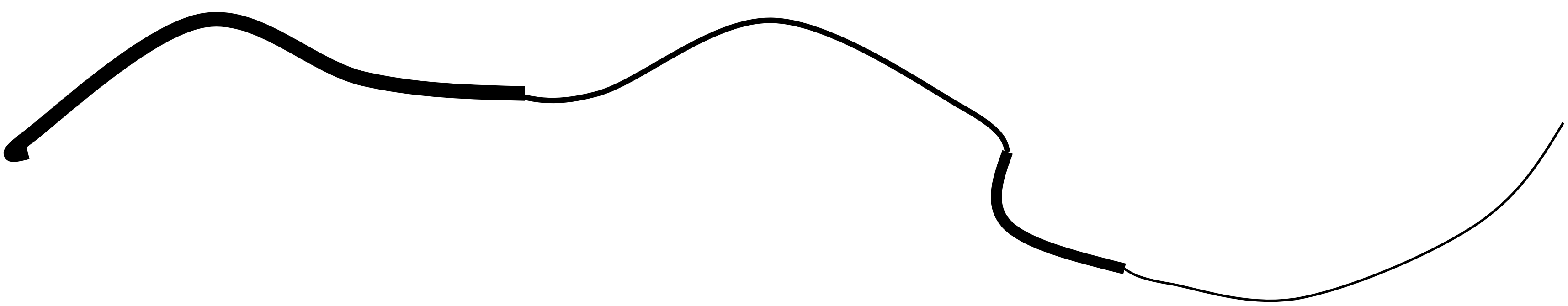
The Canny edge detector

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

thinning
(non-maximum suppression)

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

Recap: Canny edge detector

1. Filter image with derivative of Gaussian
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

MATLAB: `edge(image, 'canny');`

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

Edge detection is just the beginning…

image | human segmentation | gradient magnitude

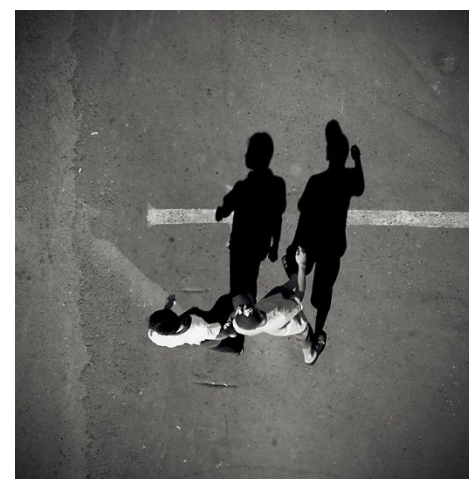
Berkeley segmentation database:

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Low-level edges vs. perceived contours

Background

Texture



Shadows