Image Filtering

Overview of Filtering

- Convolution
- Gaussian filtering
- Median filtering

Overview of Filtering

- Convolution
- Gaussian filtering
- Median filtering

Motivation: Noise reduction

- Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!

What's the next best thing?

Source: S. Seitz

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?

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

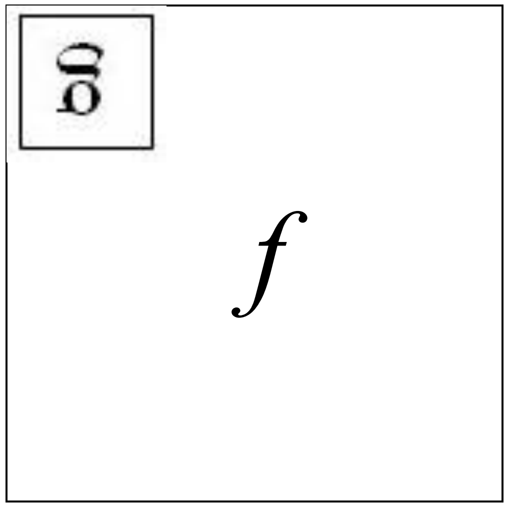
"box filter"

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 (also imfilter)

Source: F. Durand

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))

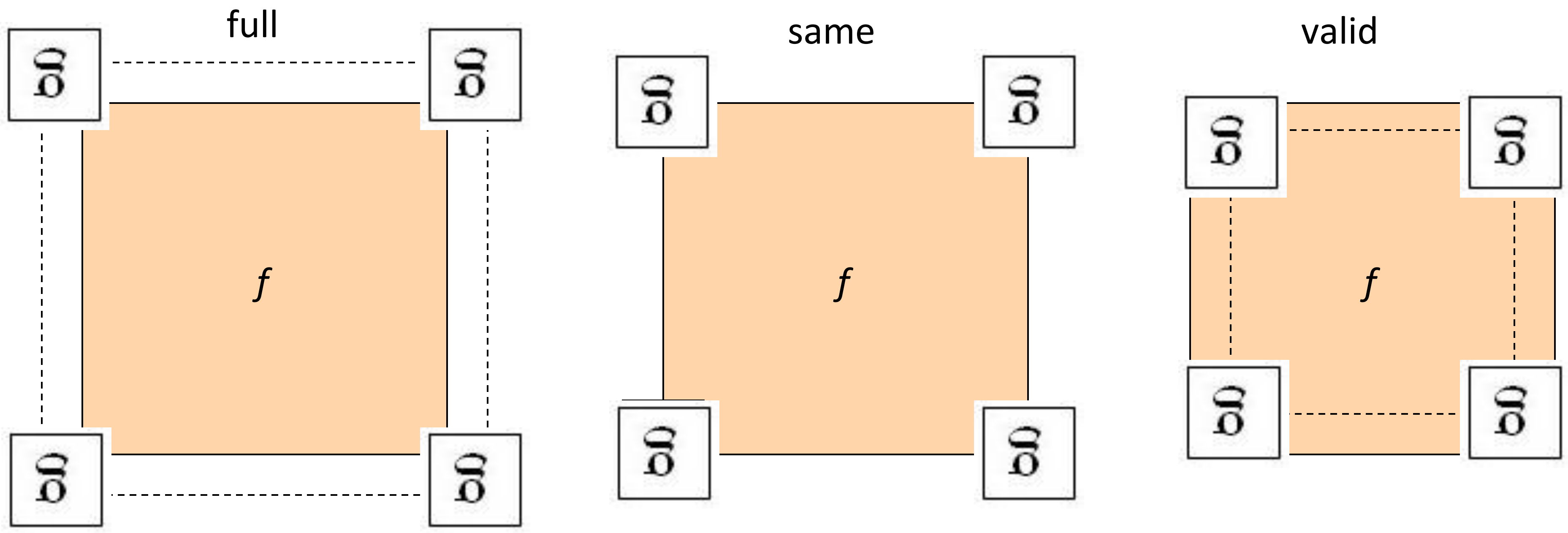
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

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: conv2(f, g,*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

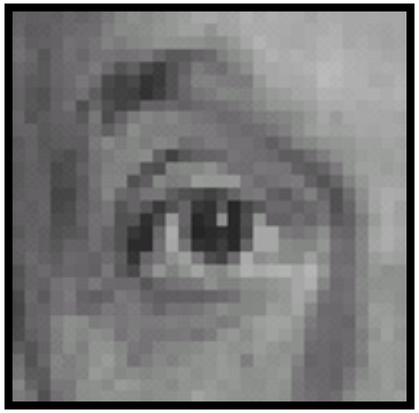
Source: S. Marschner

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

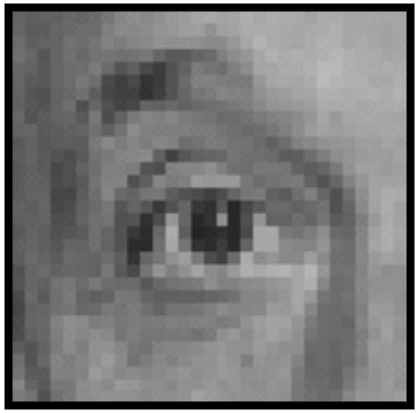

Original

0	0	0
0	1	0
0	0	0

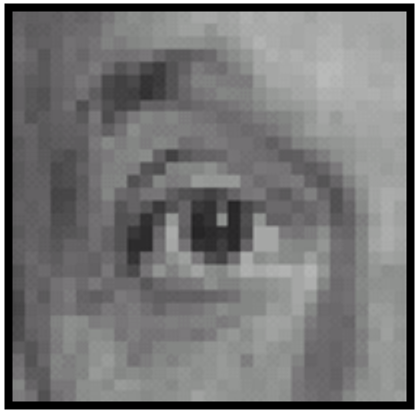
?

Source: D. Lowe

Practice with linear filters

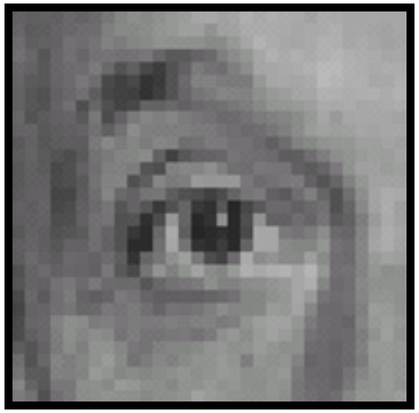

Original

0	0	0
0	1	0
0	0	0


Filtered
(no change)

Source: D. Lowe

Practice with linear filters



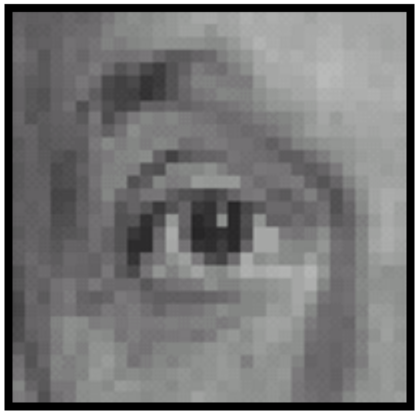
Original

0	0	0
0	0	1
0	0	0

?

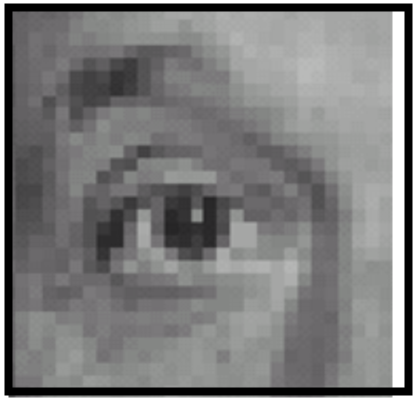
Source: D. Lowe

Practice with linear filters



Original

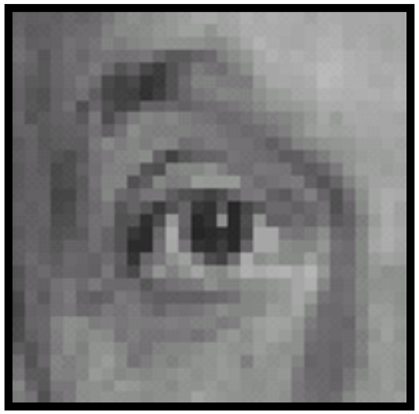
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters



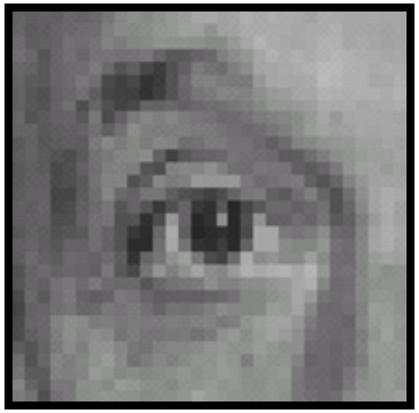
Original

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

?

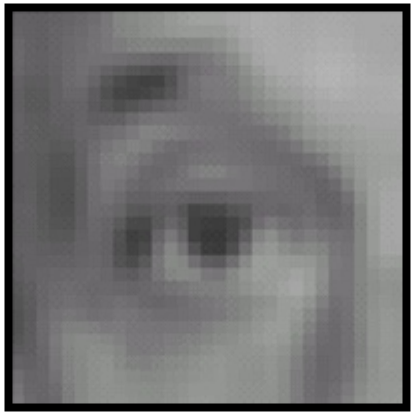
Source: D. Lowe

Practice with linear filters



Original

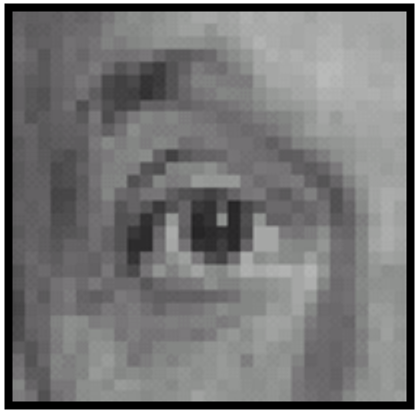
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

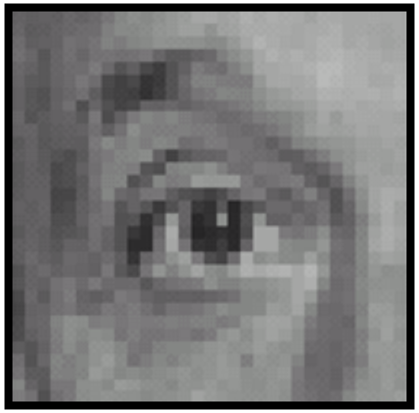

Original

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

(Note that filter sums to 1)

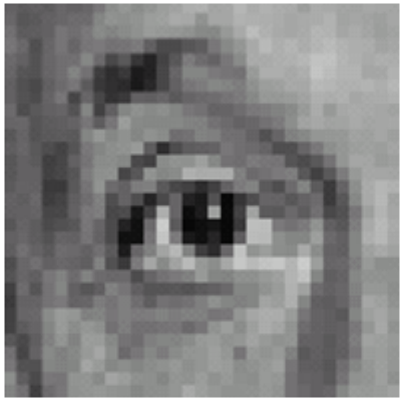
Source: D. Lowe

Practice with linear filters



Original

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

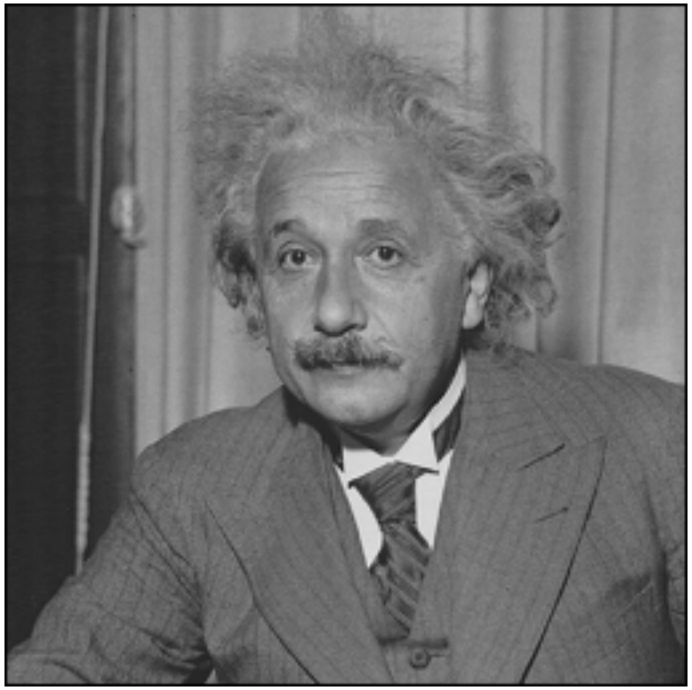


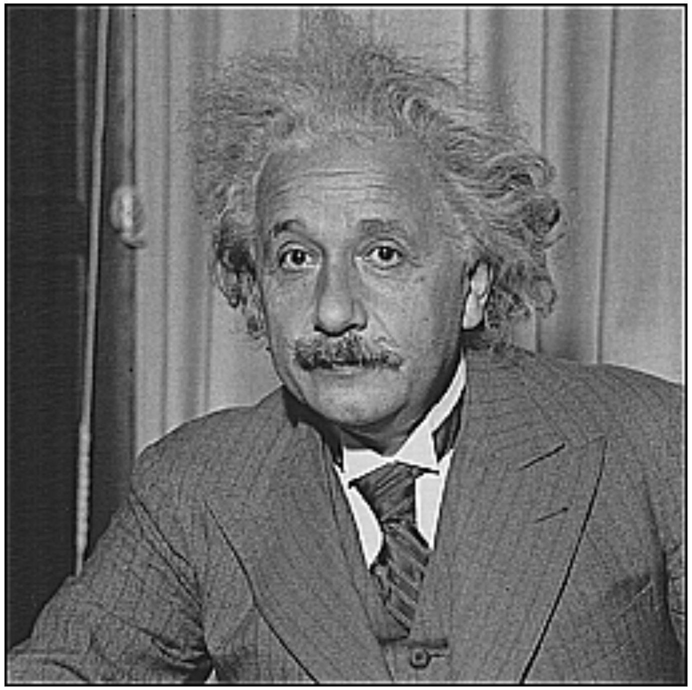
Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

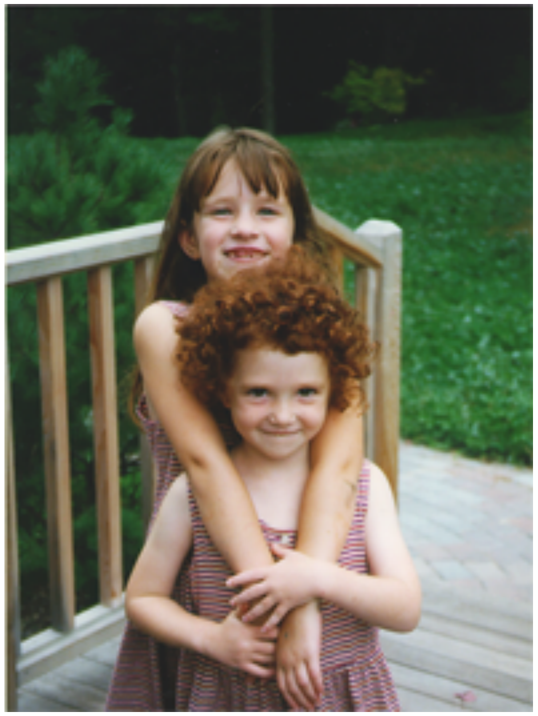
Sharpening

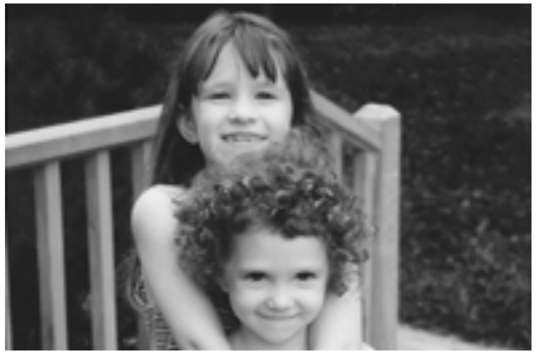

before

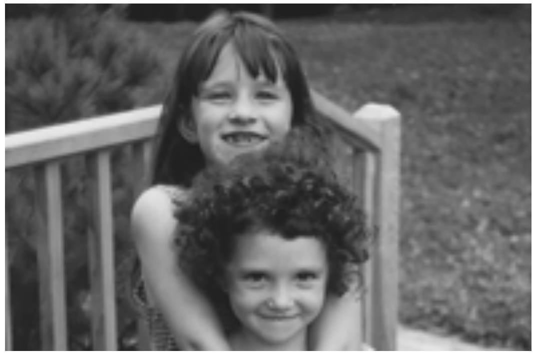

after

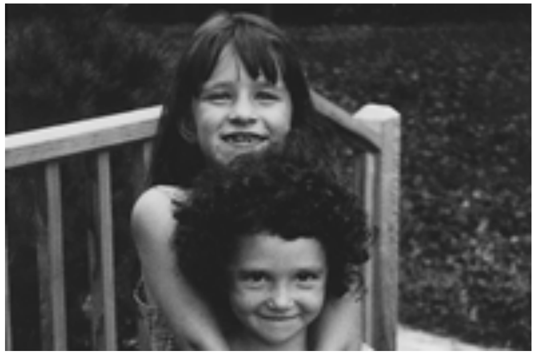
Slide credit: Bill Freeman

Spatial resolution and color


original

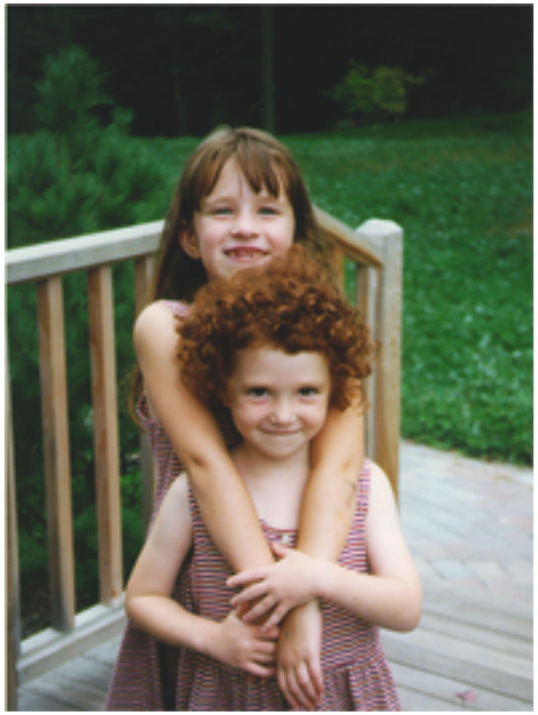

R

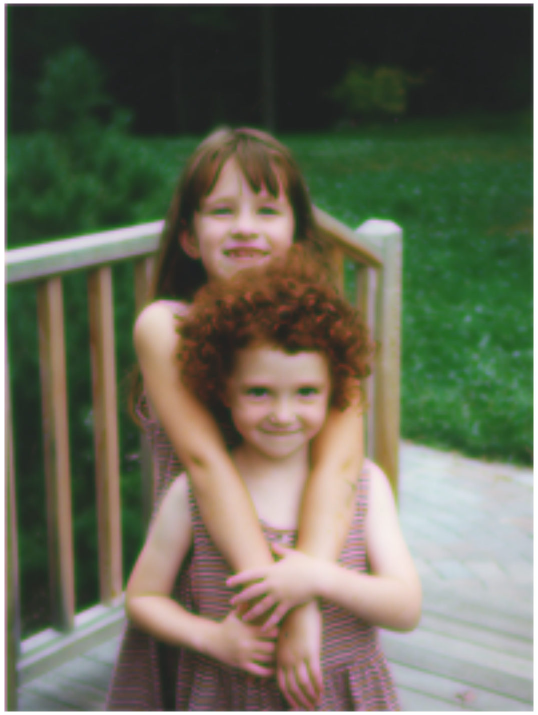

G

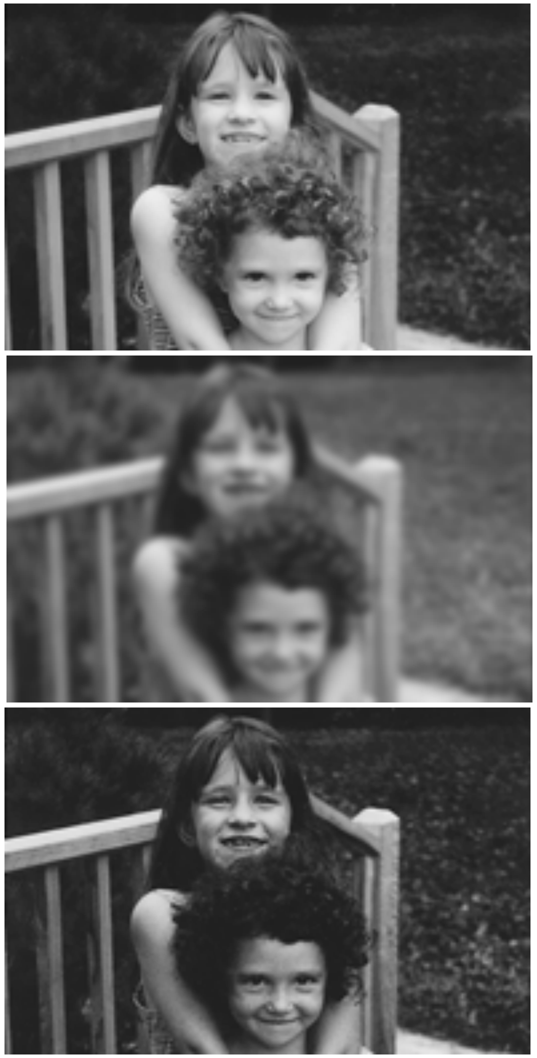

B

Slide credit: Bill Freeman

Blurring the G component

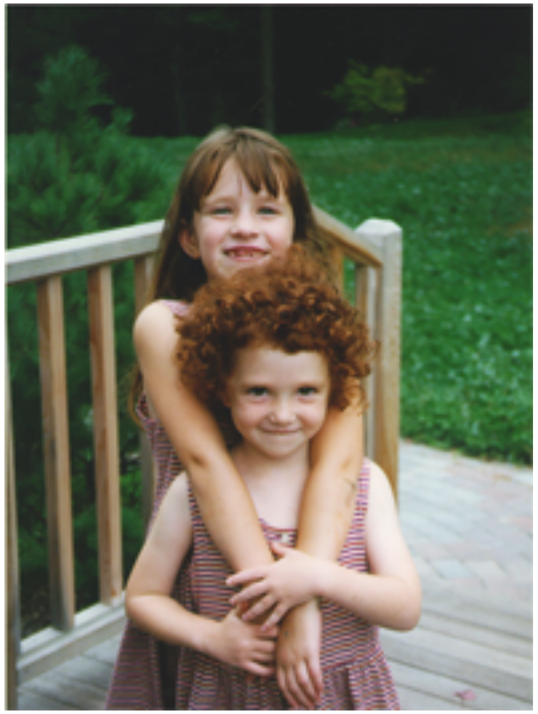

original

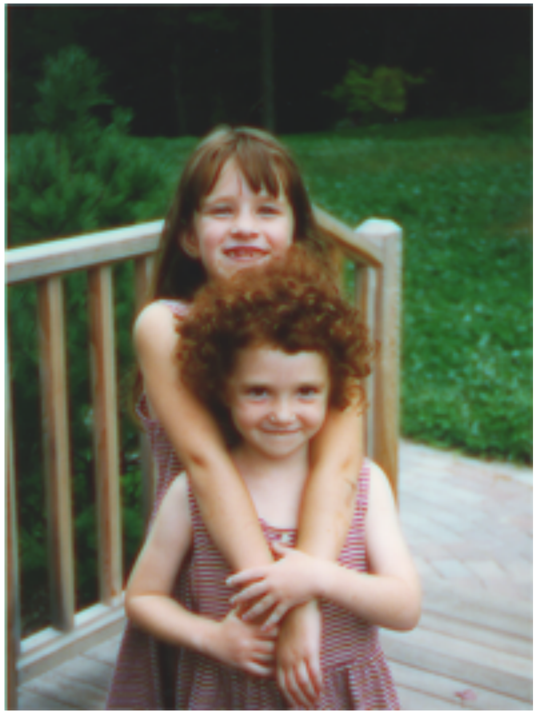

processed

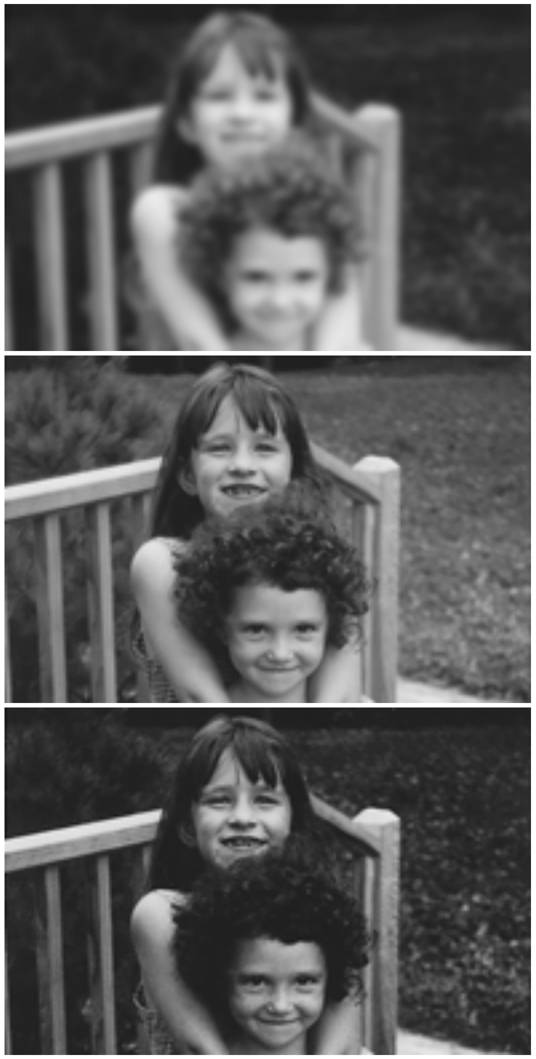


Slide credit: Bill Freeman

Blurring the R component

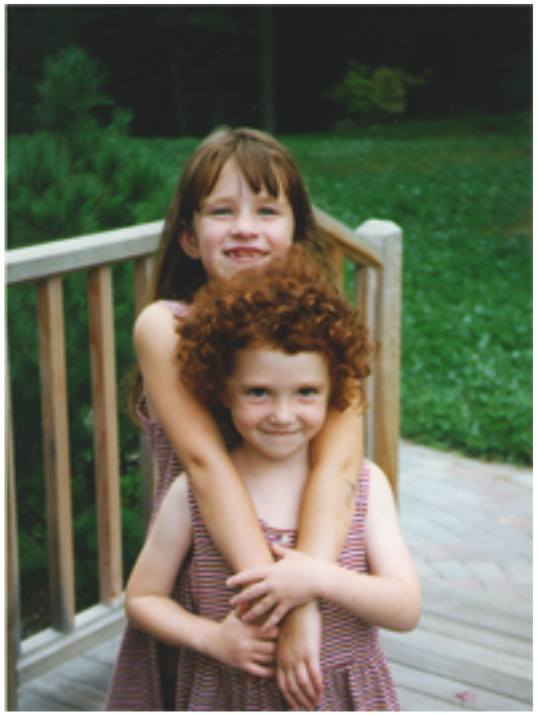

original

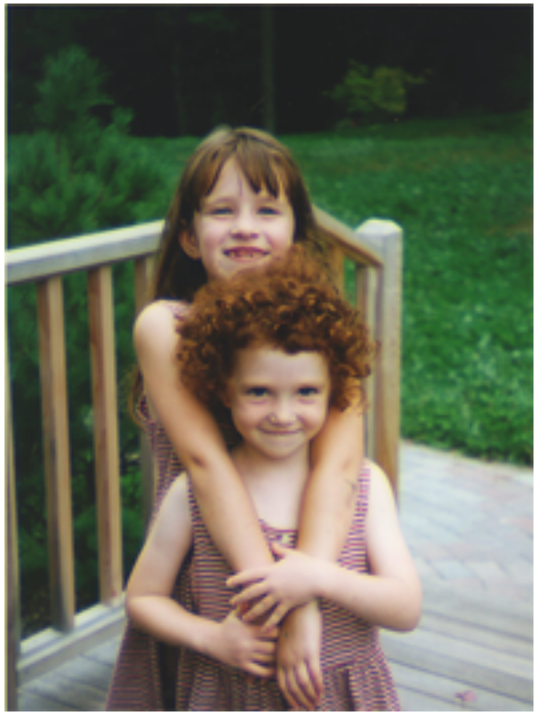

processed

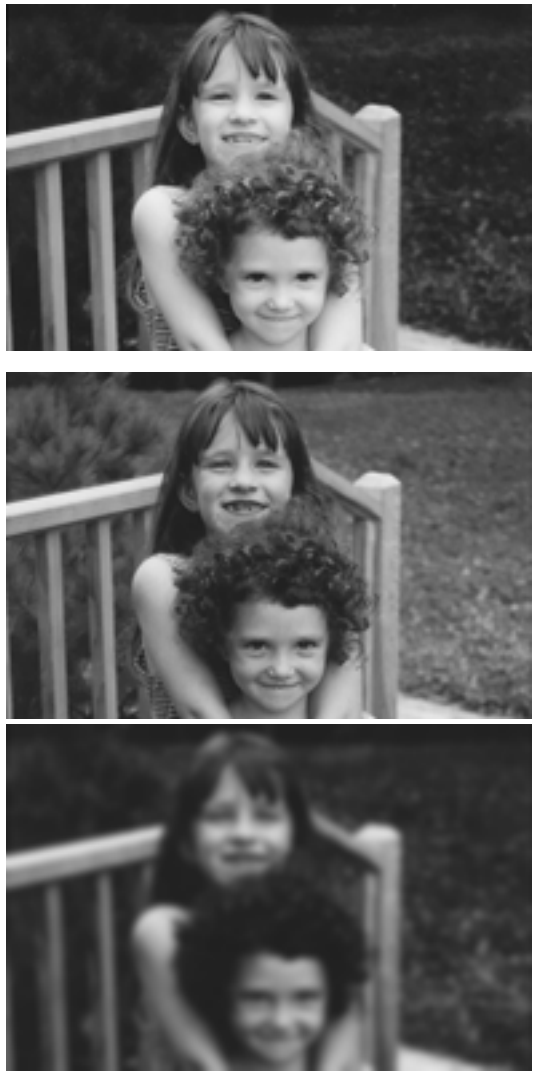


Slide credit: Bill Freeman

Blurring the B component


original


processed



Slide credit: Bill Freeman

From W. E. Glenn, in Digital Images and Human Vision, MIT Press, edited by Watson, 1993

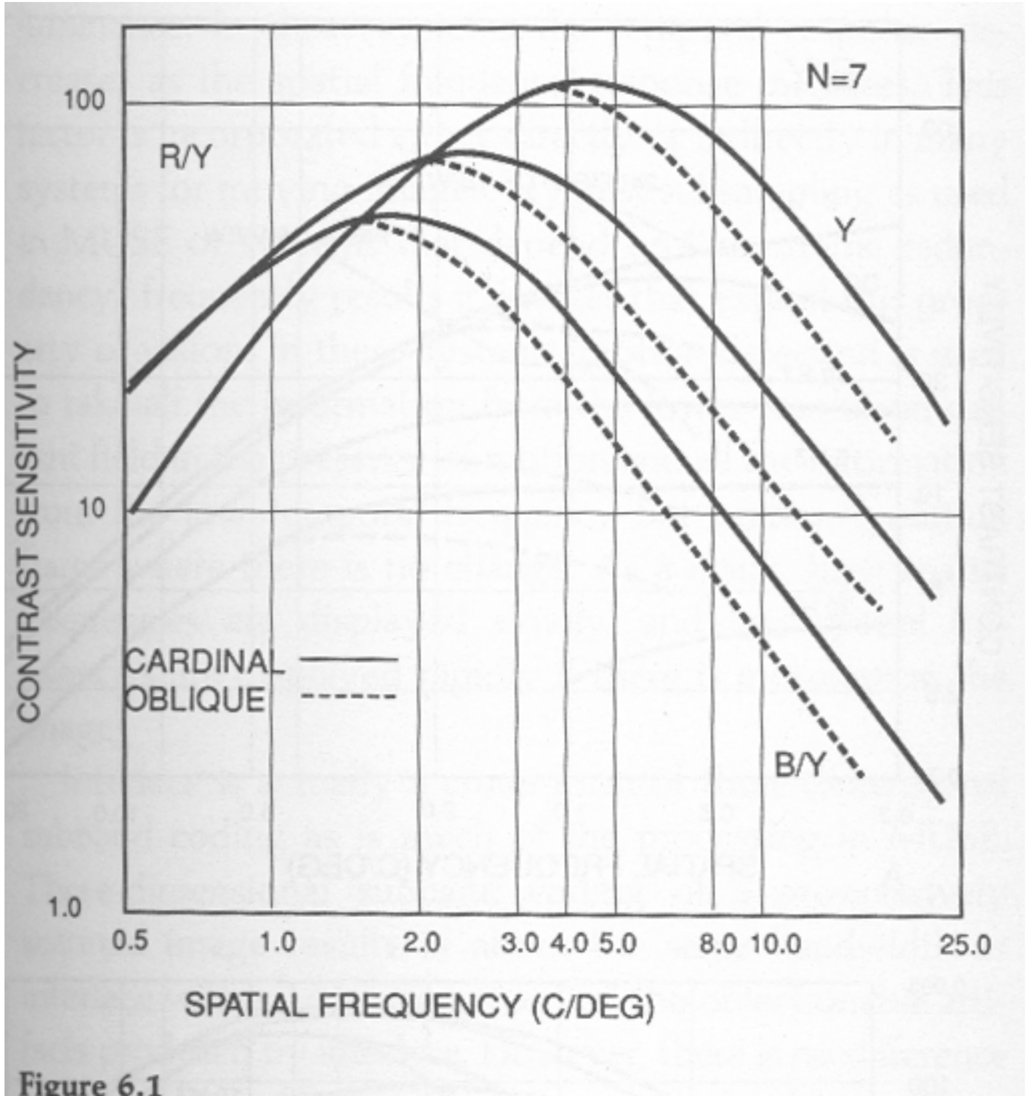
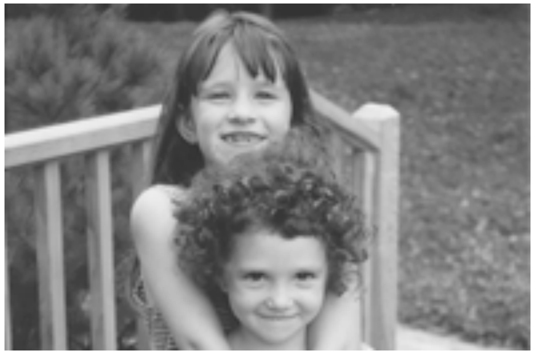


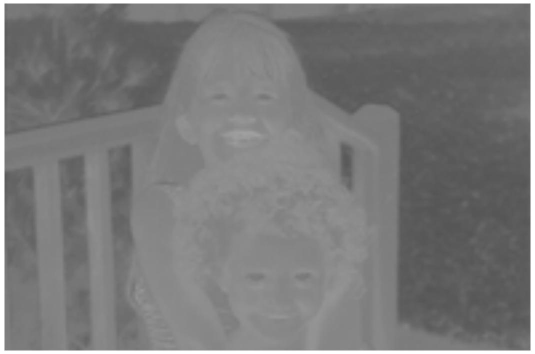
Figure 6.1
Contrast sensitivity threshold functions for static luminance gratings (Y) and isoluminance chromaticity gratings (R/Y,B/Y) averaged over seven observers.

Slide credit: Bill Freeman

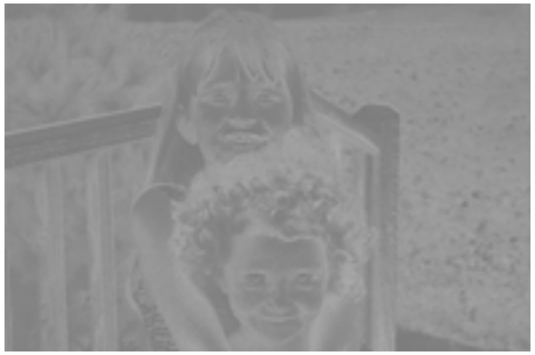
Lab color components



L



a

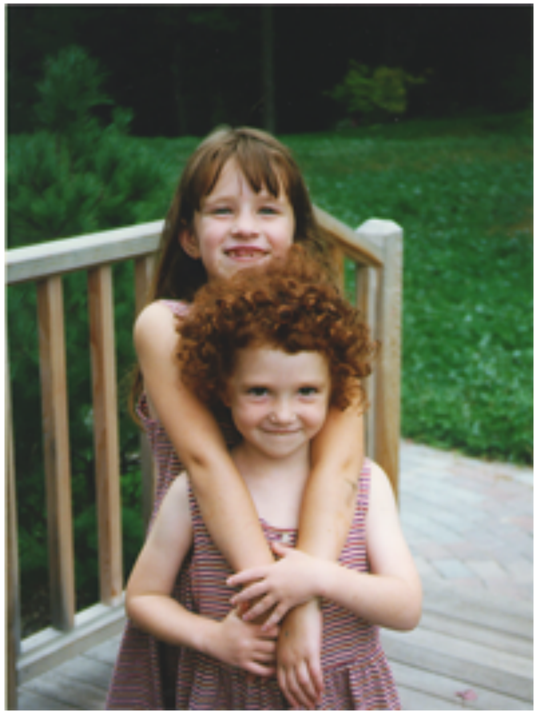


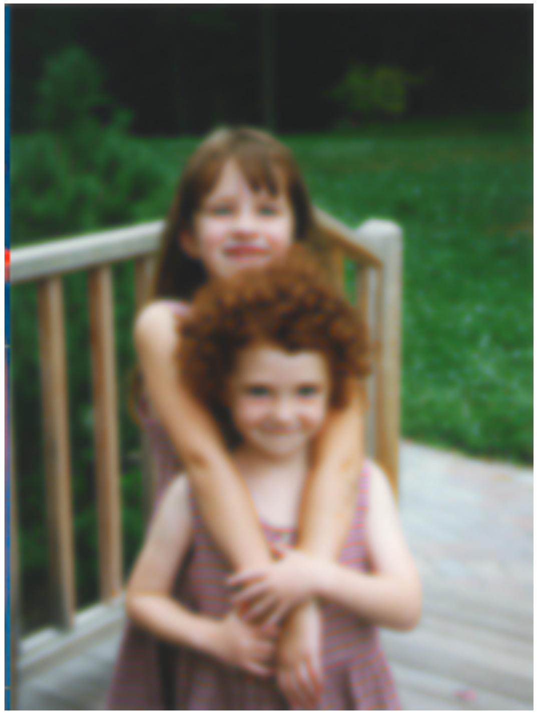
b

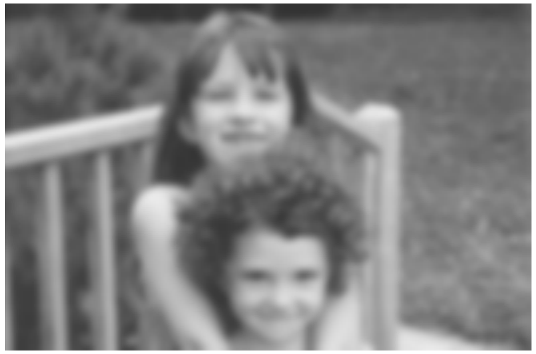
A rotation of the color coordinates into directions that are more perceptually meaningful:
L: luminance,
a: red-green,
b: blue-yellow

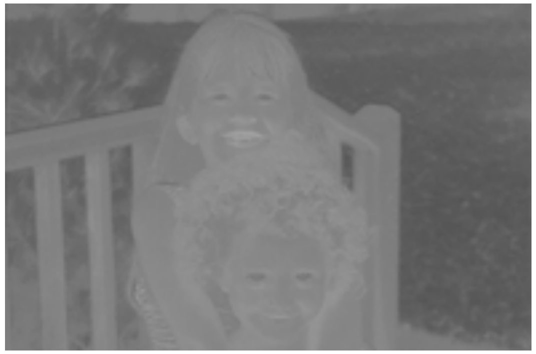
Slide credit: Bill Freeman

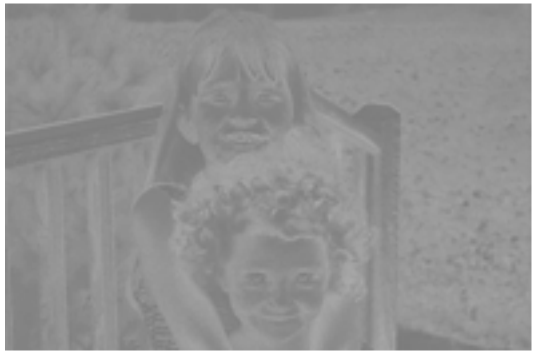
Blurring the L Lab component


original


processed


L

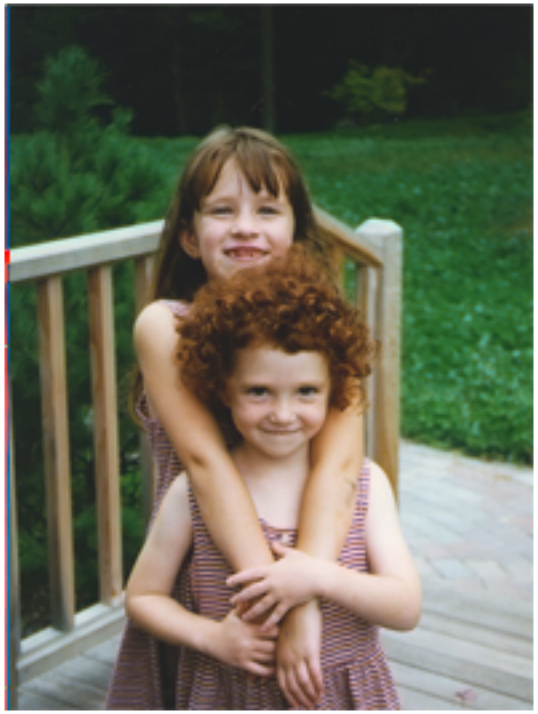

a

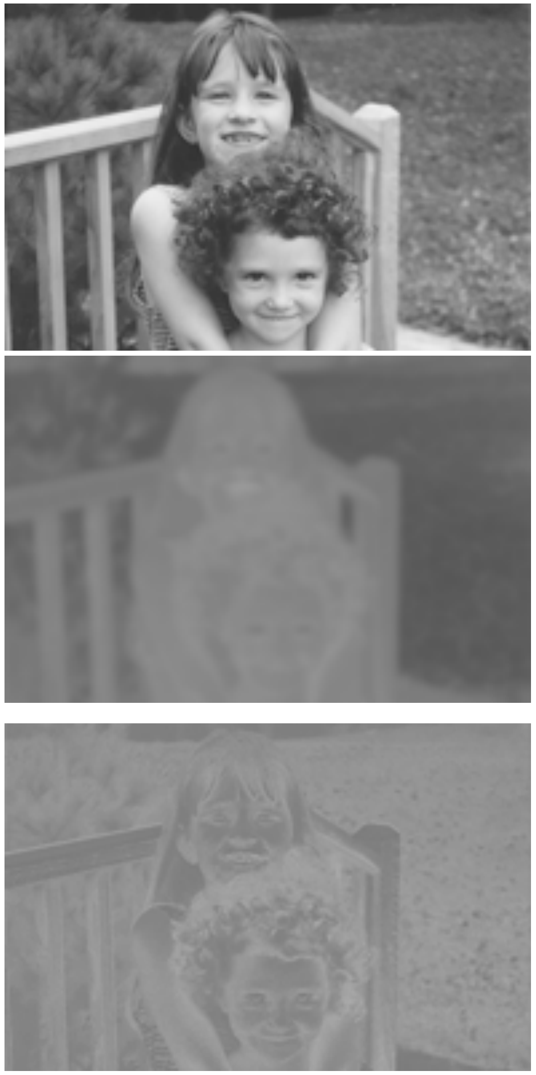

b

Slide credit: Bill Freeman

Blurring the a Lab component

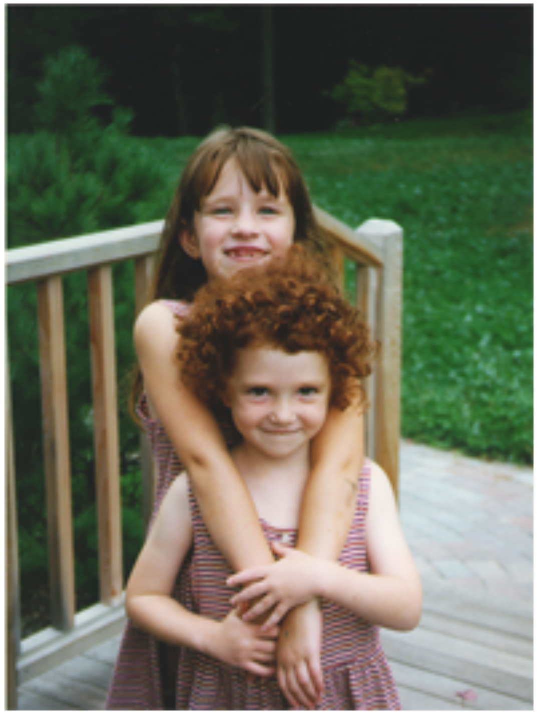
original

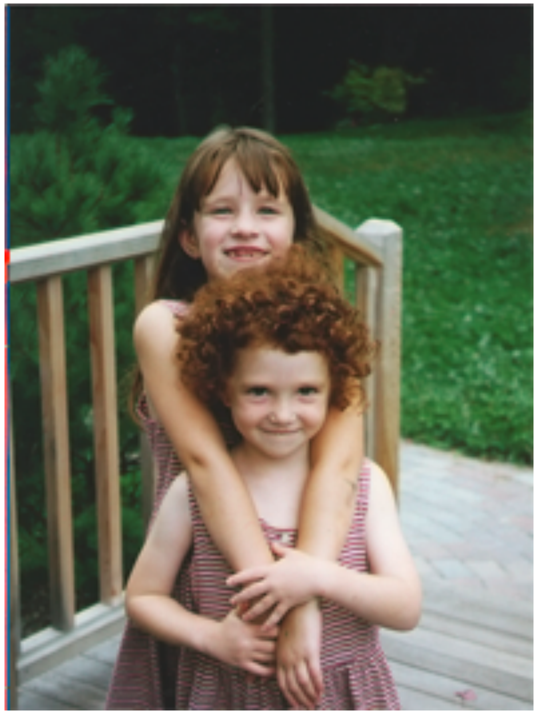

processed

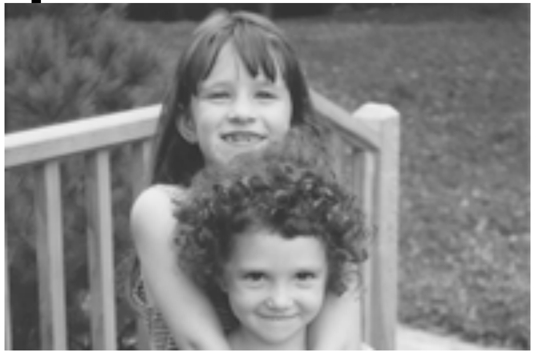


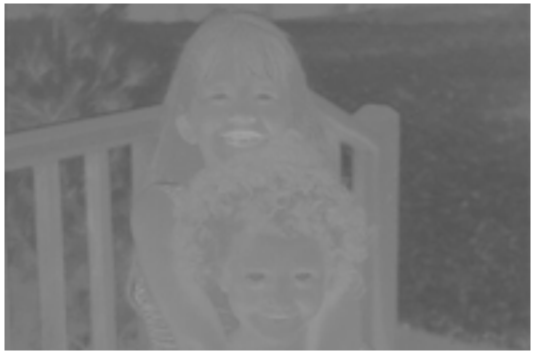
Slide credit: Bill Freeman

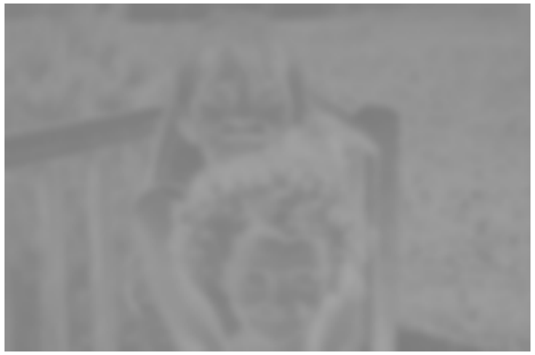
Blurring the b Lab component


original


processed


L


a


b

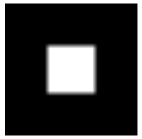
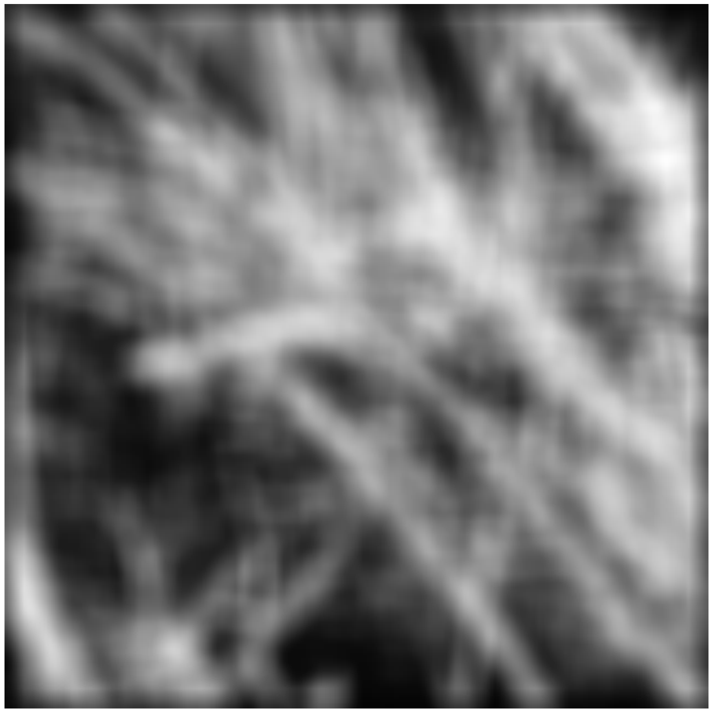
Slide credit: Bill Freeman

Overview of Filtering

- Convolution
- Gaussian filtering
- Median filtering

Smoothing with box filter revisited

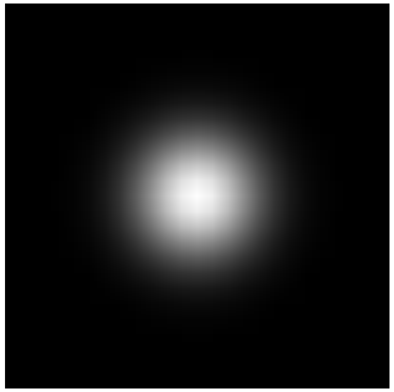
- Smoothing with an average actually doesn't compare at all well with a defocused lens
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square

Source: D. Forsyth

Smoothing with box filter revisited

- Smoothing with an average actually doesn't compare at all well with a defocused lens
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square
- Better idea: to eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center, like so:

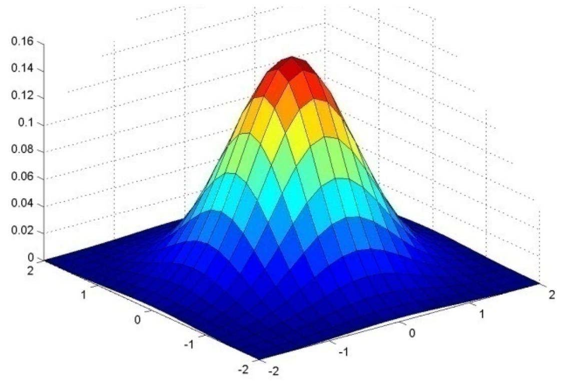


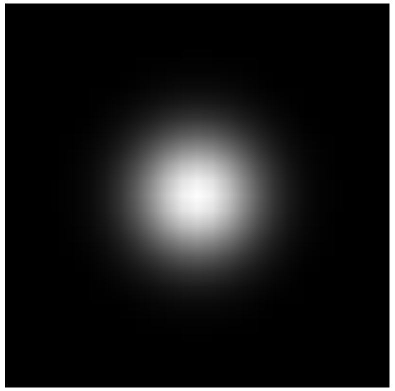
"fuzzy blob"

Source: D. Forsyth

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

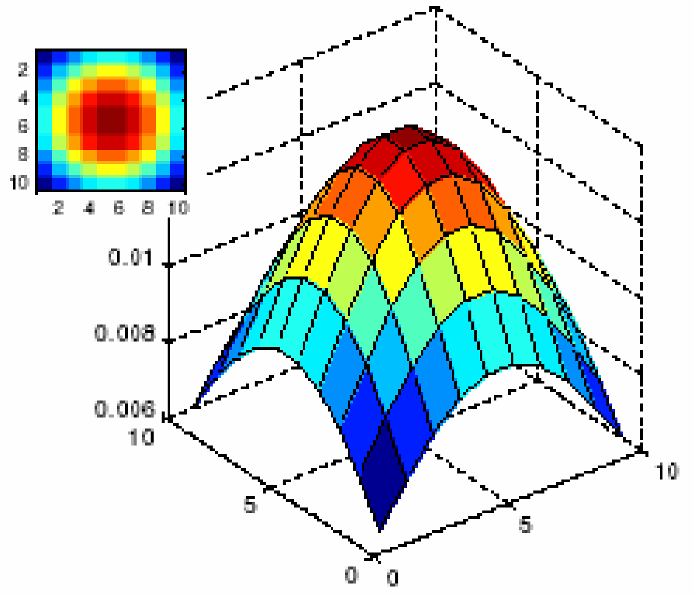
5 x 5, σ = 1

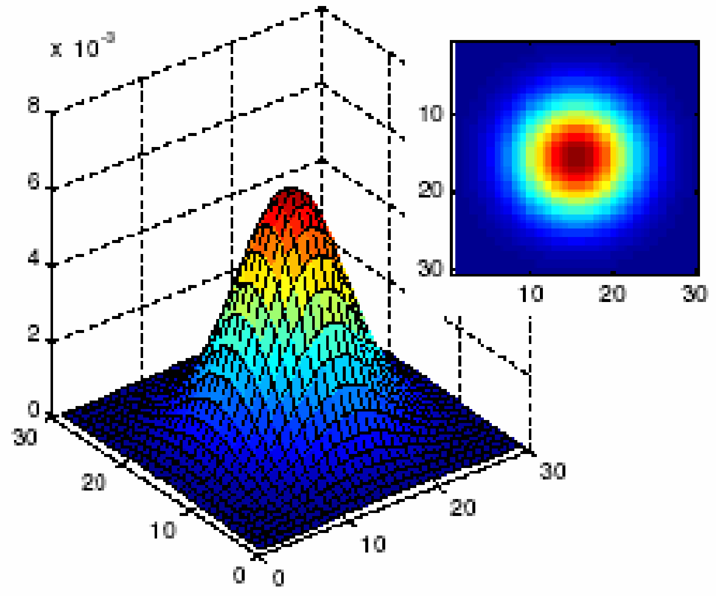
- Constant factor at front makes volume sum to 1 (can be ignored, as we should re-normalize weights to sum to 1 in any case)

Source: C. Rasmussen

Choosing kernel width

- Gaussian filters have infinite support, but discrete filters use finite kernels

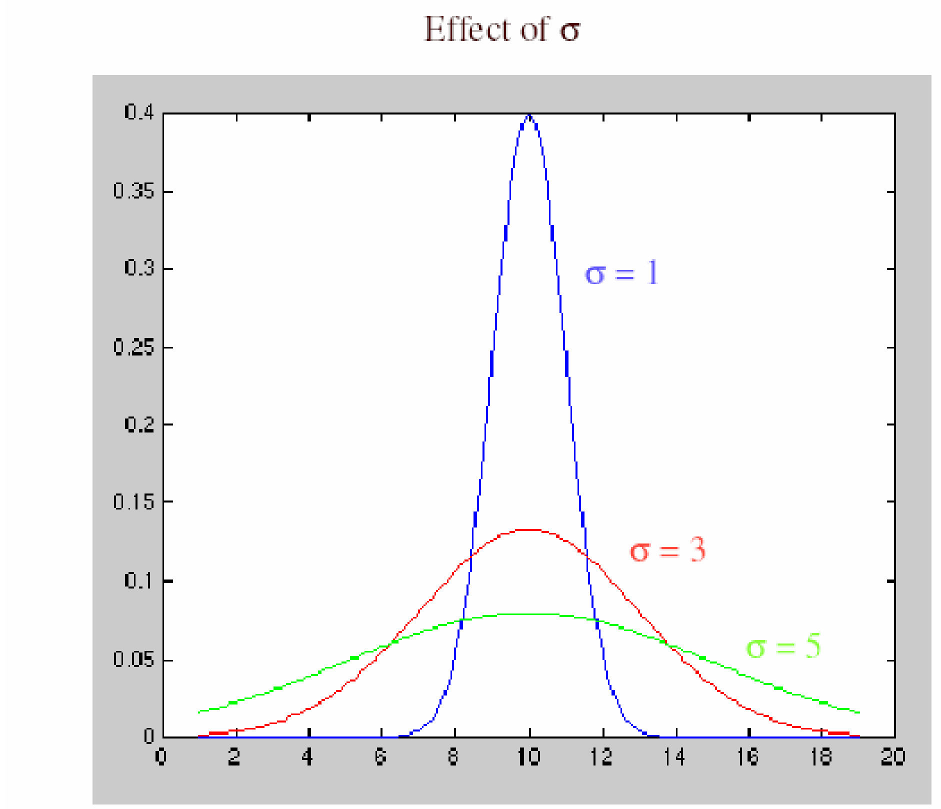

σ = 5 with 10x10 kernel


σ = 5 with 30x30 kernel

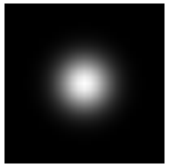
Source: K. Grauman

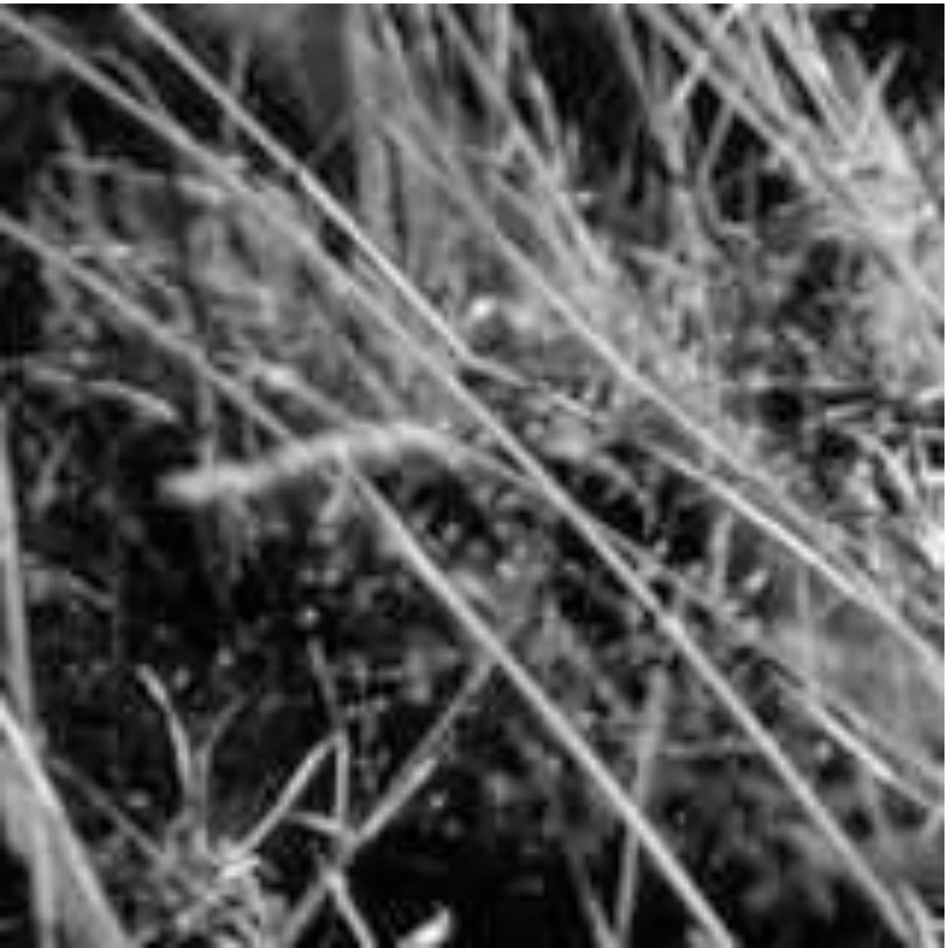
Choosing kernel width

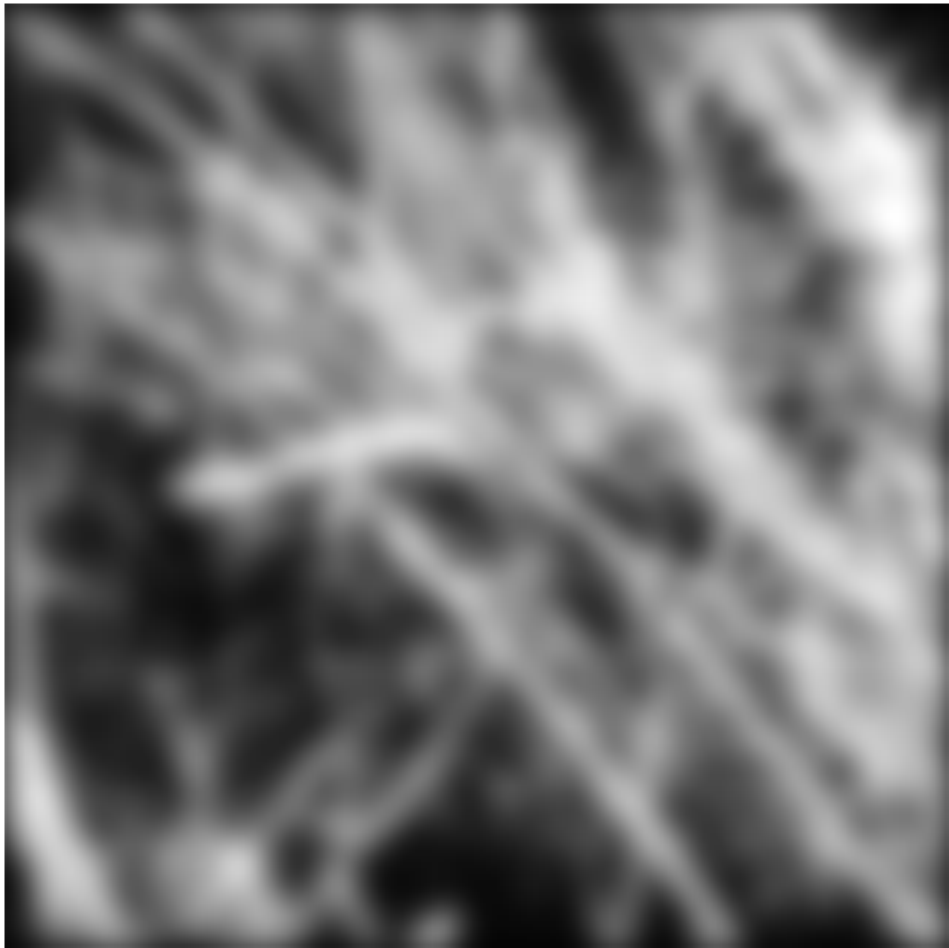
- Rule of thumb: set filter half-width to about 3 σ



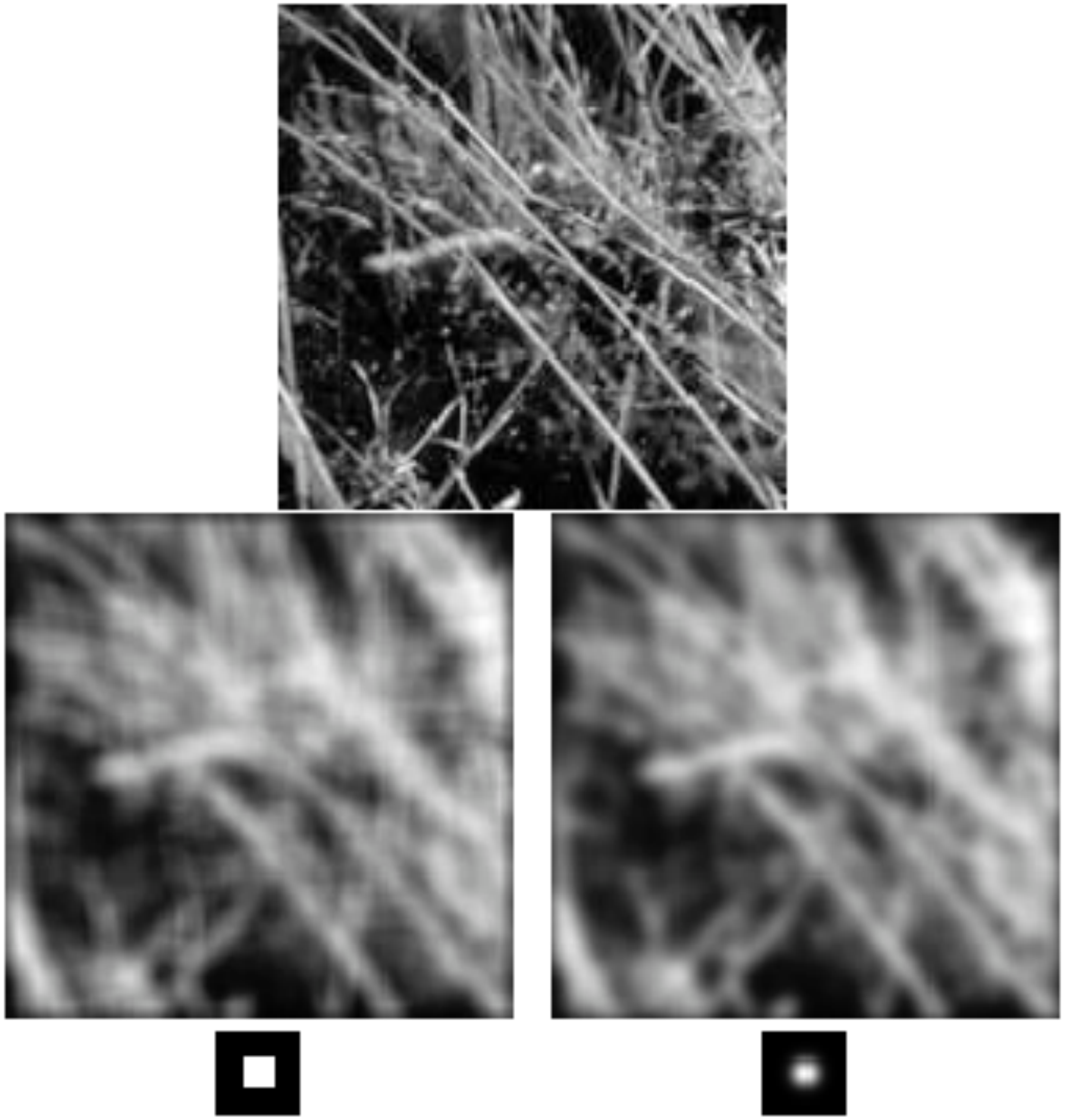
Example: Smoothing with a Gaussian







Mean vs. Gaussian filtering



Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
- 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 $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

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

Separability example

2D convolution (center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors into a product of 1D filters:

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

Perform convolution along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution along the remaining column:

For MN image, PQ filter: 2D takes MNPQ add/times, while 1D takes MN(P + Q)

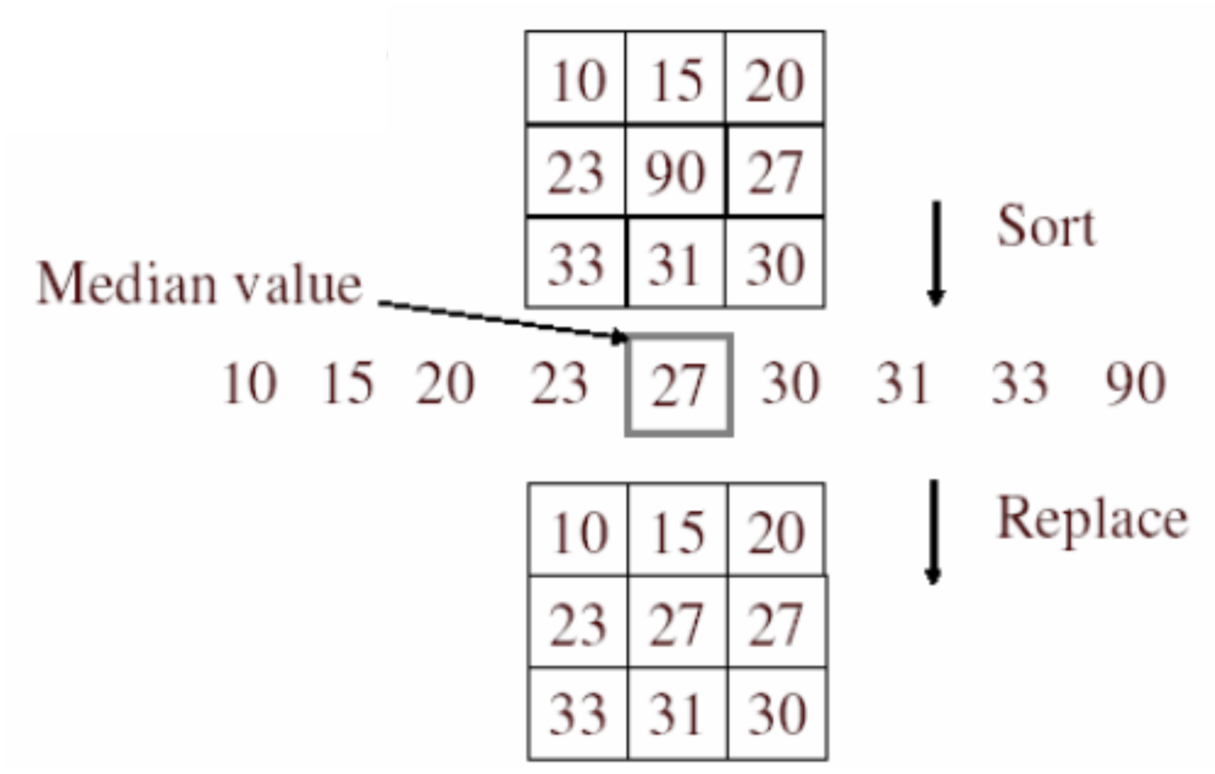
Source: K. Grauman

Overview of Filtering

- Convolution
- Gaussian filtering
- Median filtering

Alternative idea: Median filtering

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



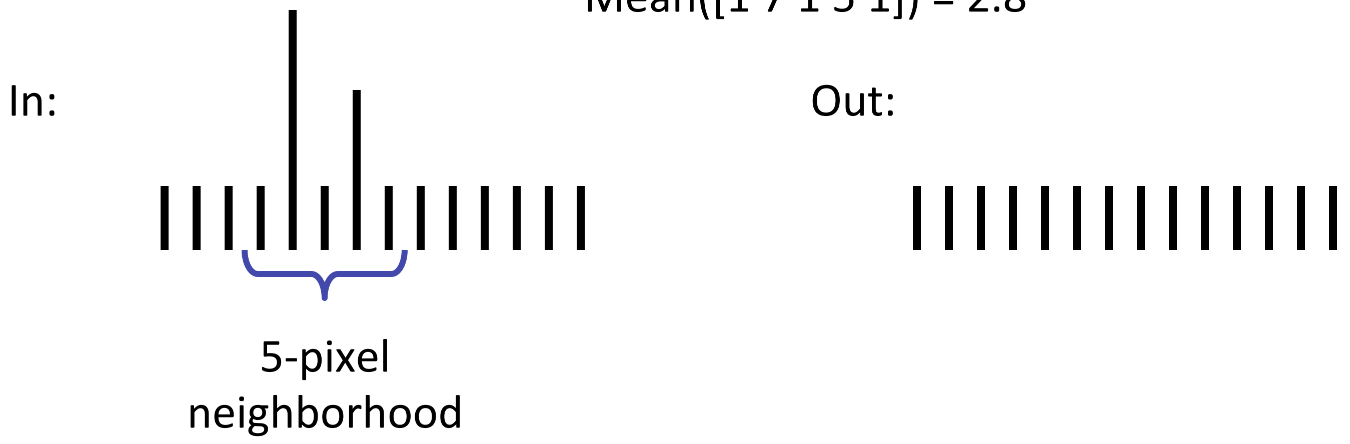
- Is median filtering linear?

Source: K. Grauman

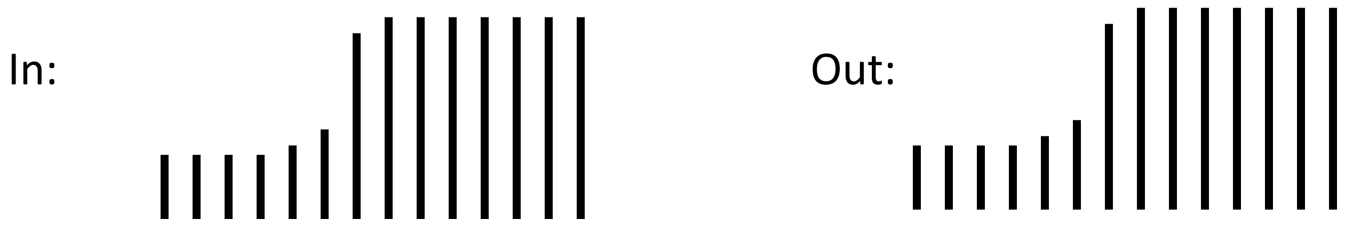
Median filter

Replace each pixel by the median over N pixels (5 pixels, for these examples). Generalizes to "rank order" filters.

Median([1 7 1 5 1]) = 1
Mean([1 7 1 5 1]) = 2.8



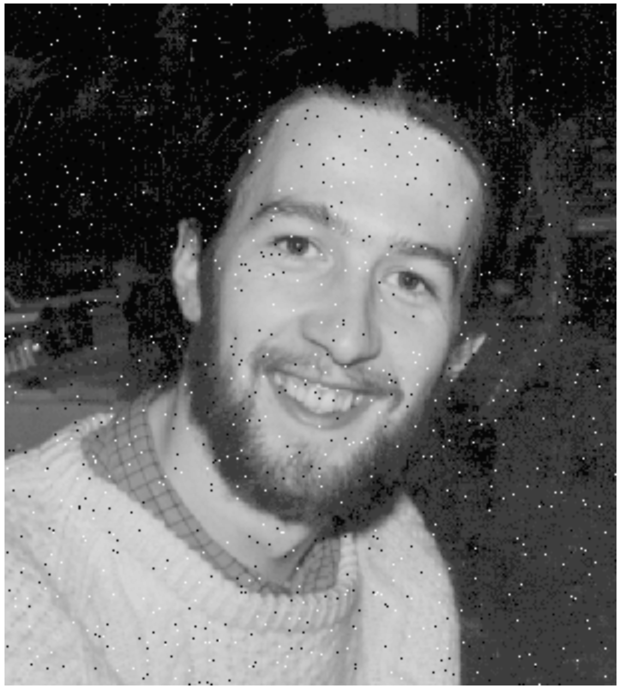
Spike noise is removed

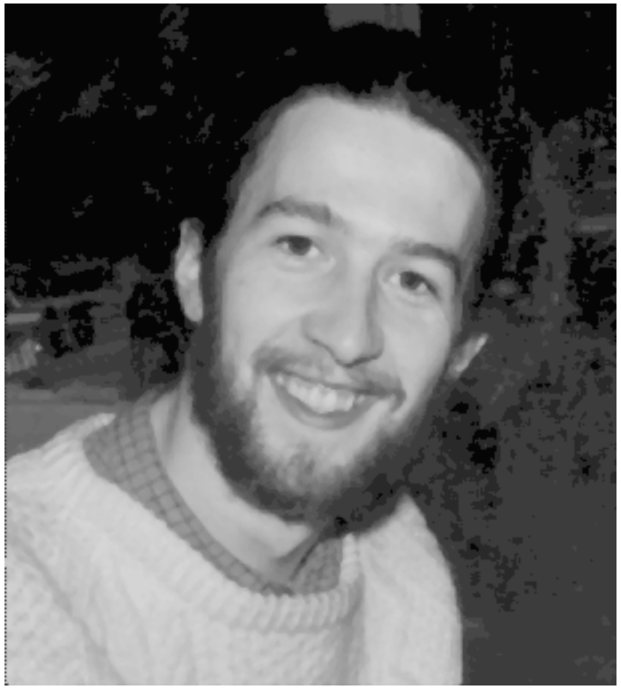


Monotonic edges remain unchanged

Median filtering results

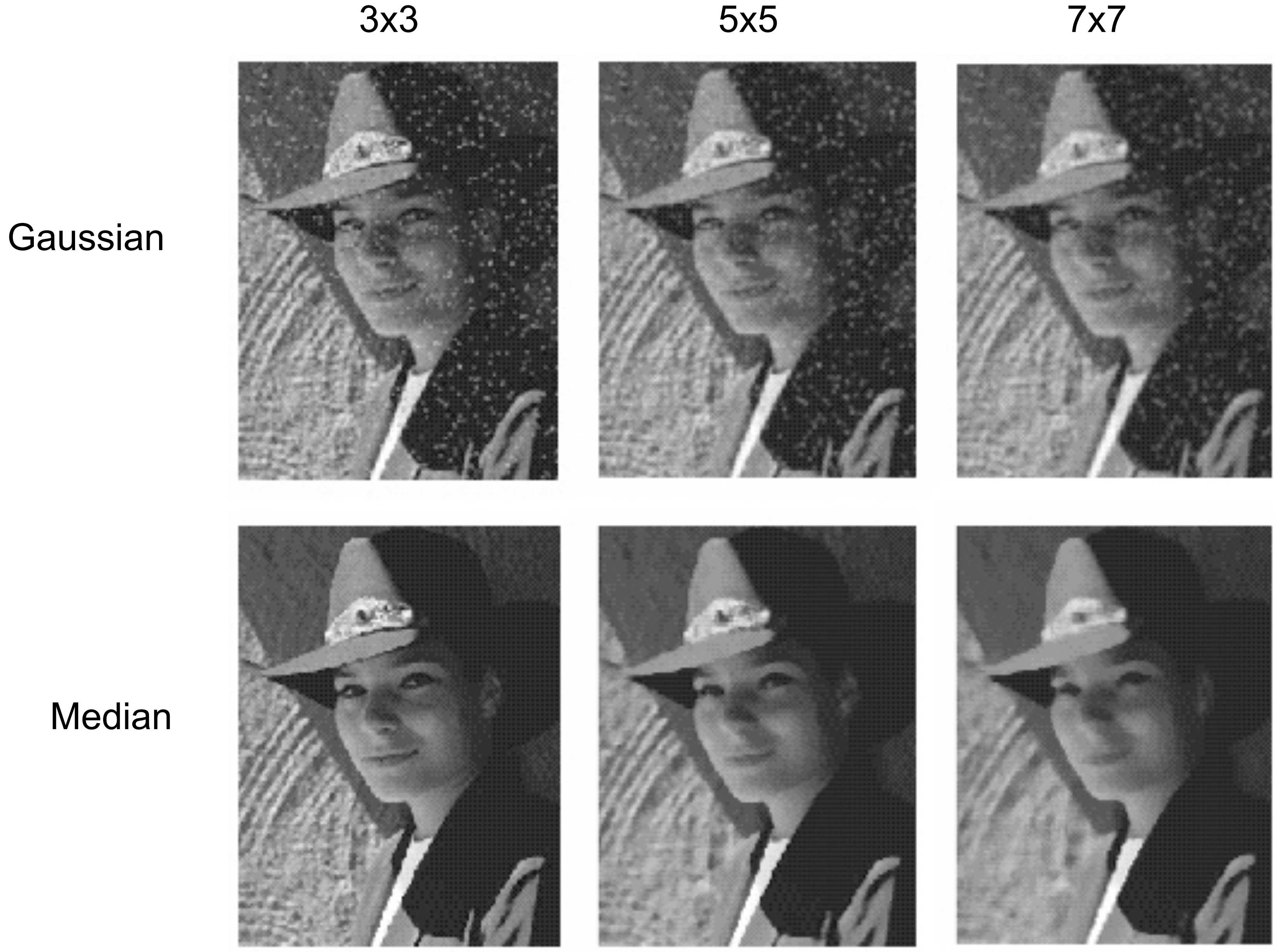
Best for salt and pepper noise



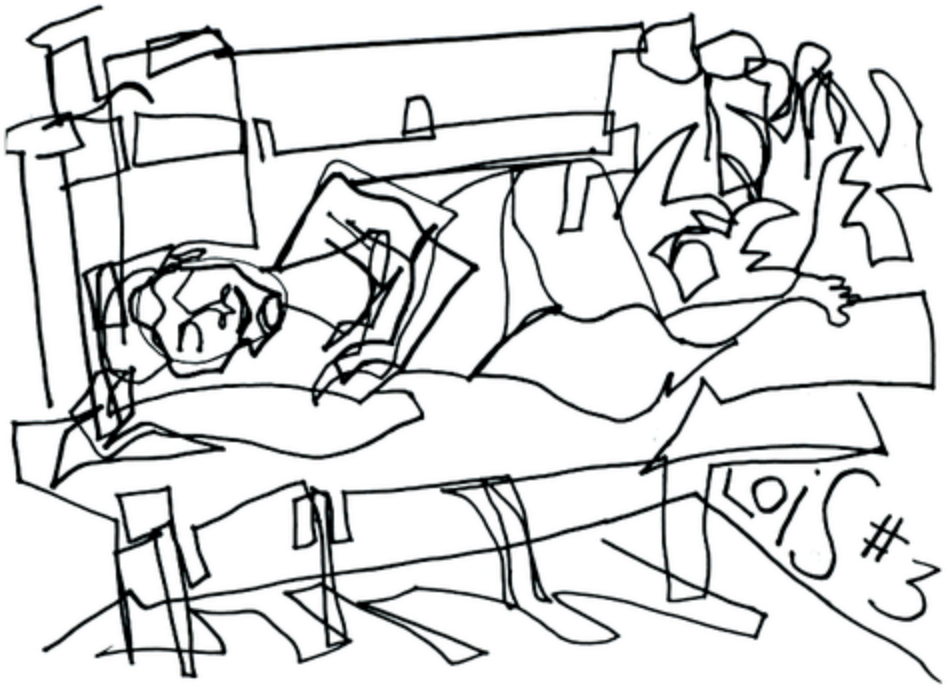


http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm#guidelines

Median vs. Gaussian filtering



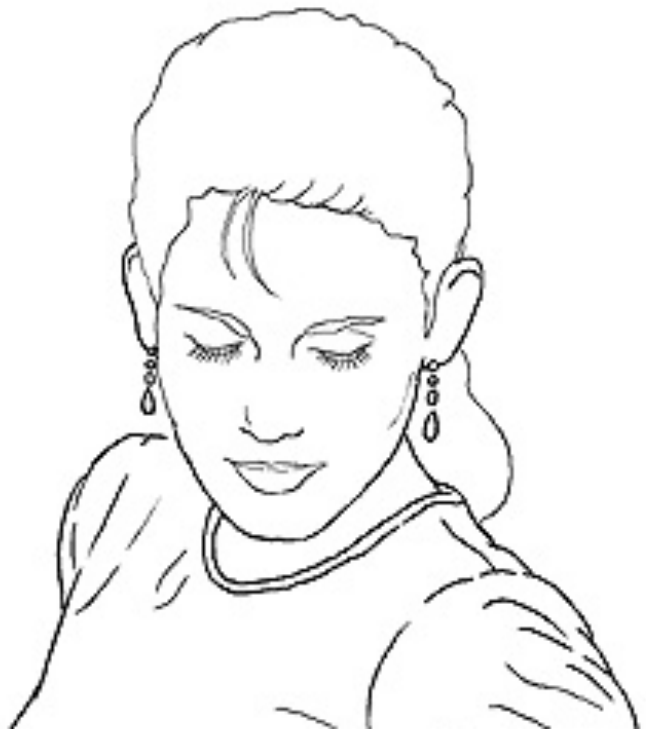
Edges



http://todayinart.com/files/2009/12/500x388xblind-contour-line-drawing.png.pagespeed.ic.DOli66Ckz1.png

Edge detection

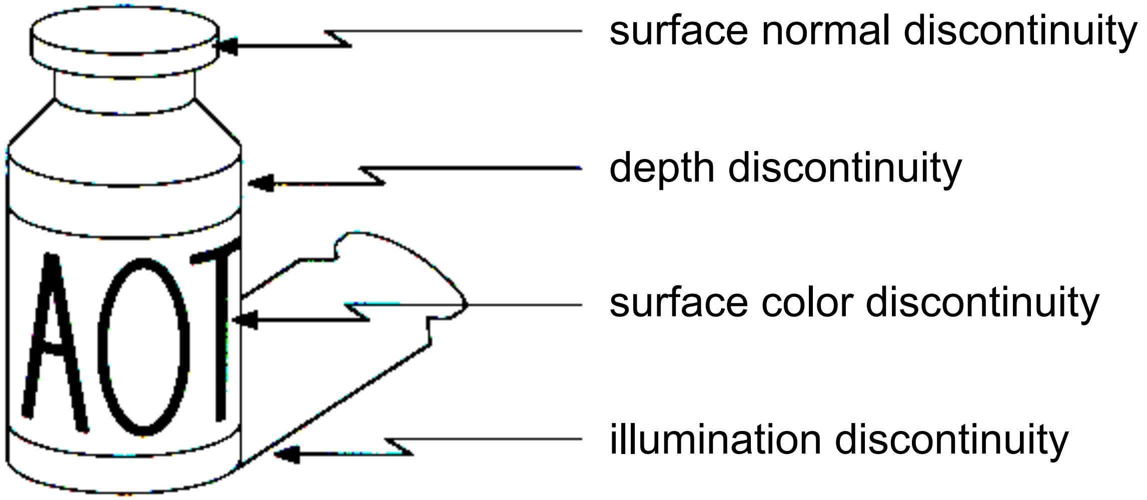
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



Source: D. Lowe

Origin of edges

Edges are caused by a variety of factors:

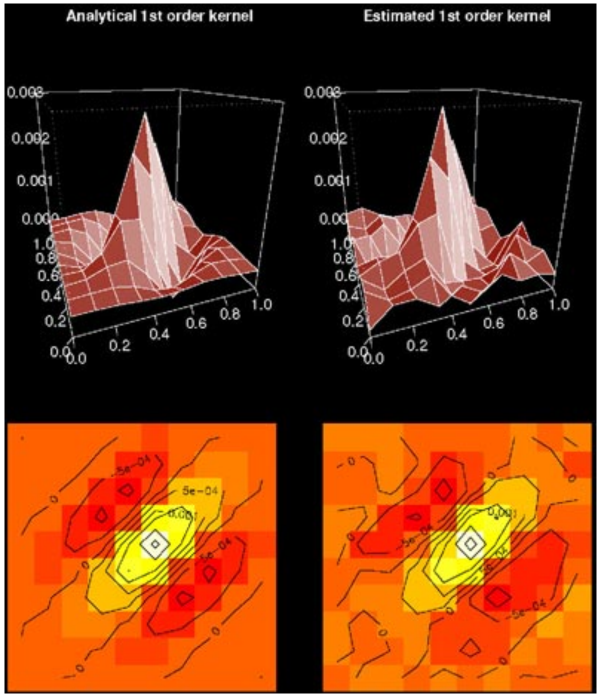

AOT
surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

Source: Steve Seitz

Edges in the Visual Cortex

Extract compact, generic, representation of image that carries sufficient information for higher-level processing tasks

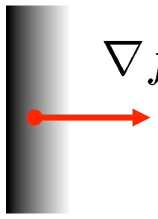
Essentially what area V1 does in our visual cortex.



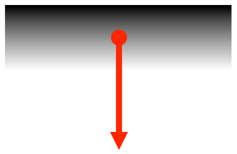
http://www.usc.edu/programs/vpl/private/photos/research/retinal_circuits/figure_2.jpg

Image gradient

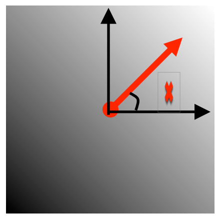
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

Differentiation and convolution

Recall, for 2D function, f(x,y):

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

This is linear and shift invariant, so must be the result of a convolution.

We could approximate this as

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

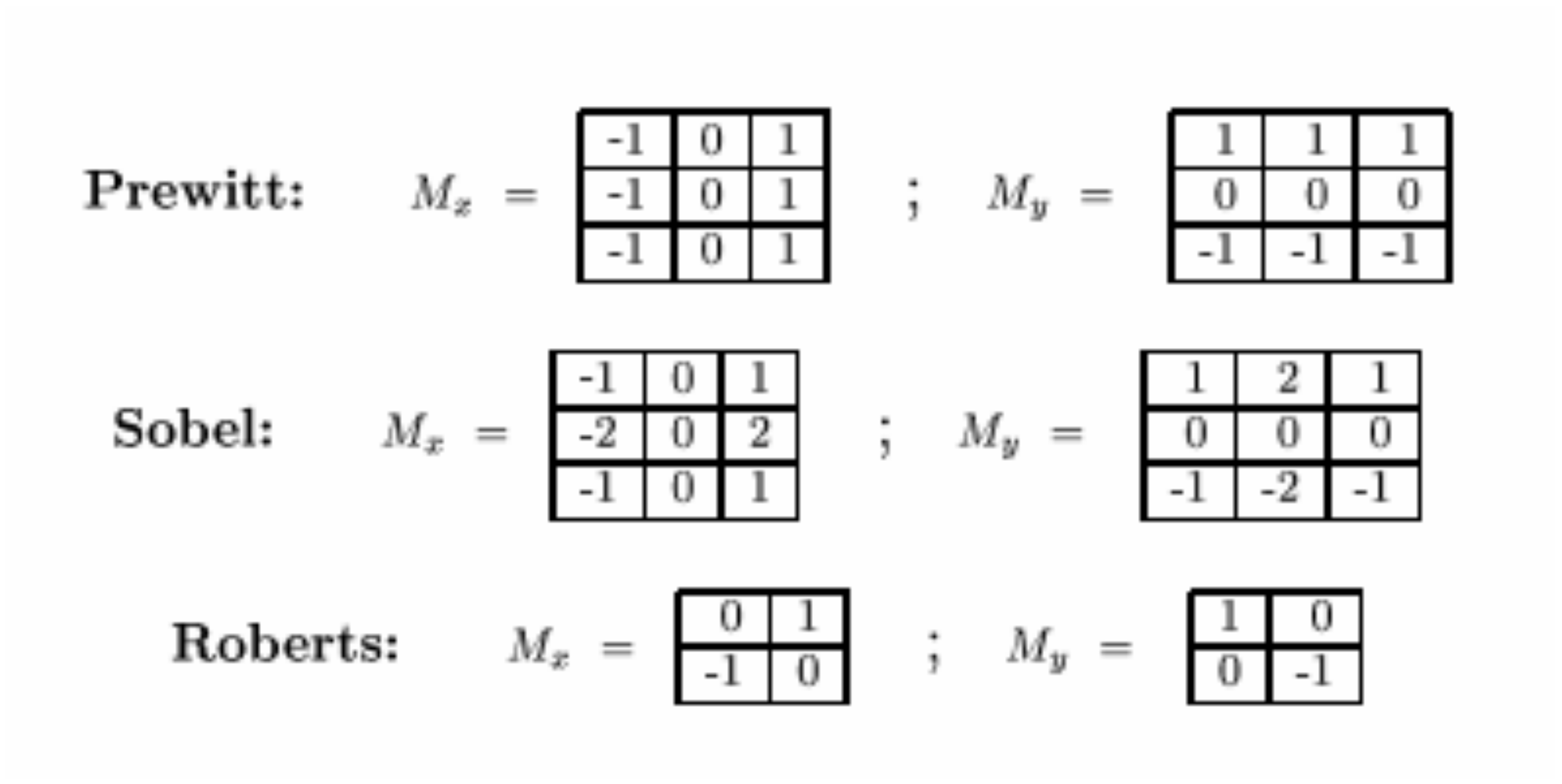
(which is obviously a convolution)

-1	**1**

Source: D. Forsyth, D. Lowe

Finite difference filters

Other approximations of derivative filters exist:



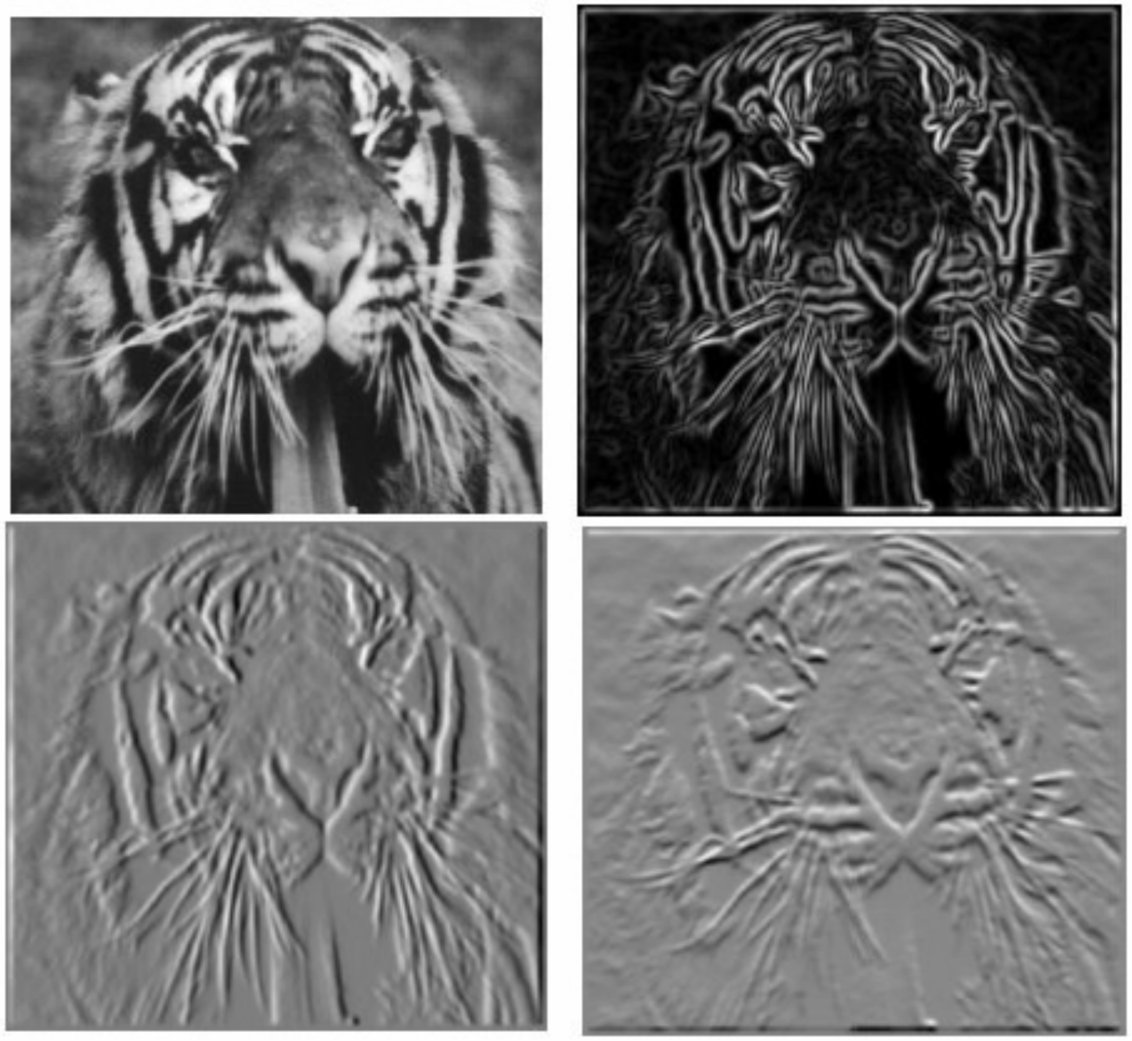
Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

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

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

Source: K. Grauman

Finite differences: example



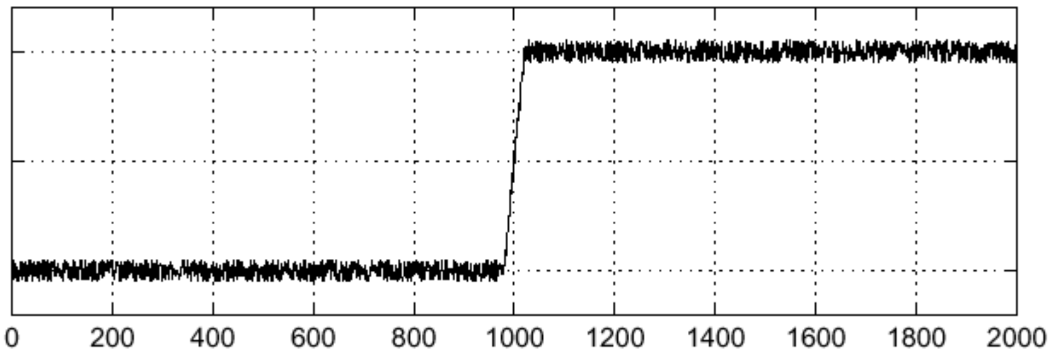
Which one is the gradient in the x-direction (resp. y-direction)?

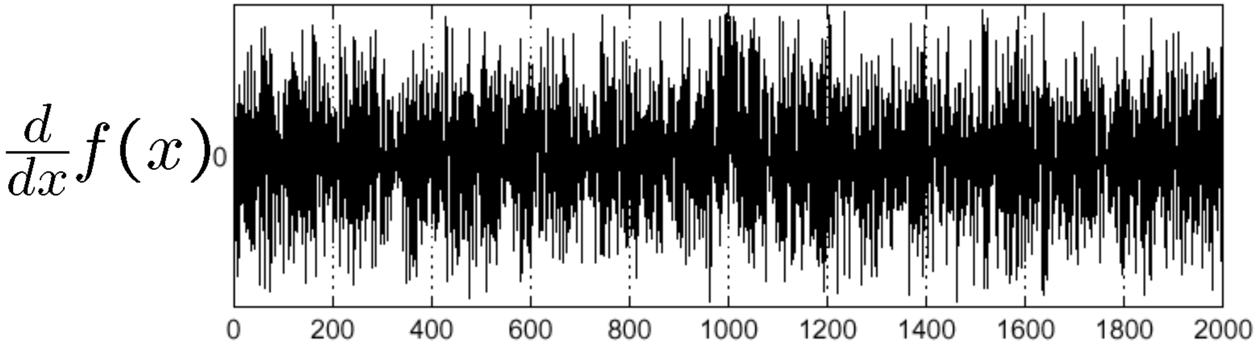
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

The image cannot be displayed. Your computer may





Where is the edge?

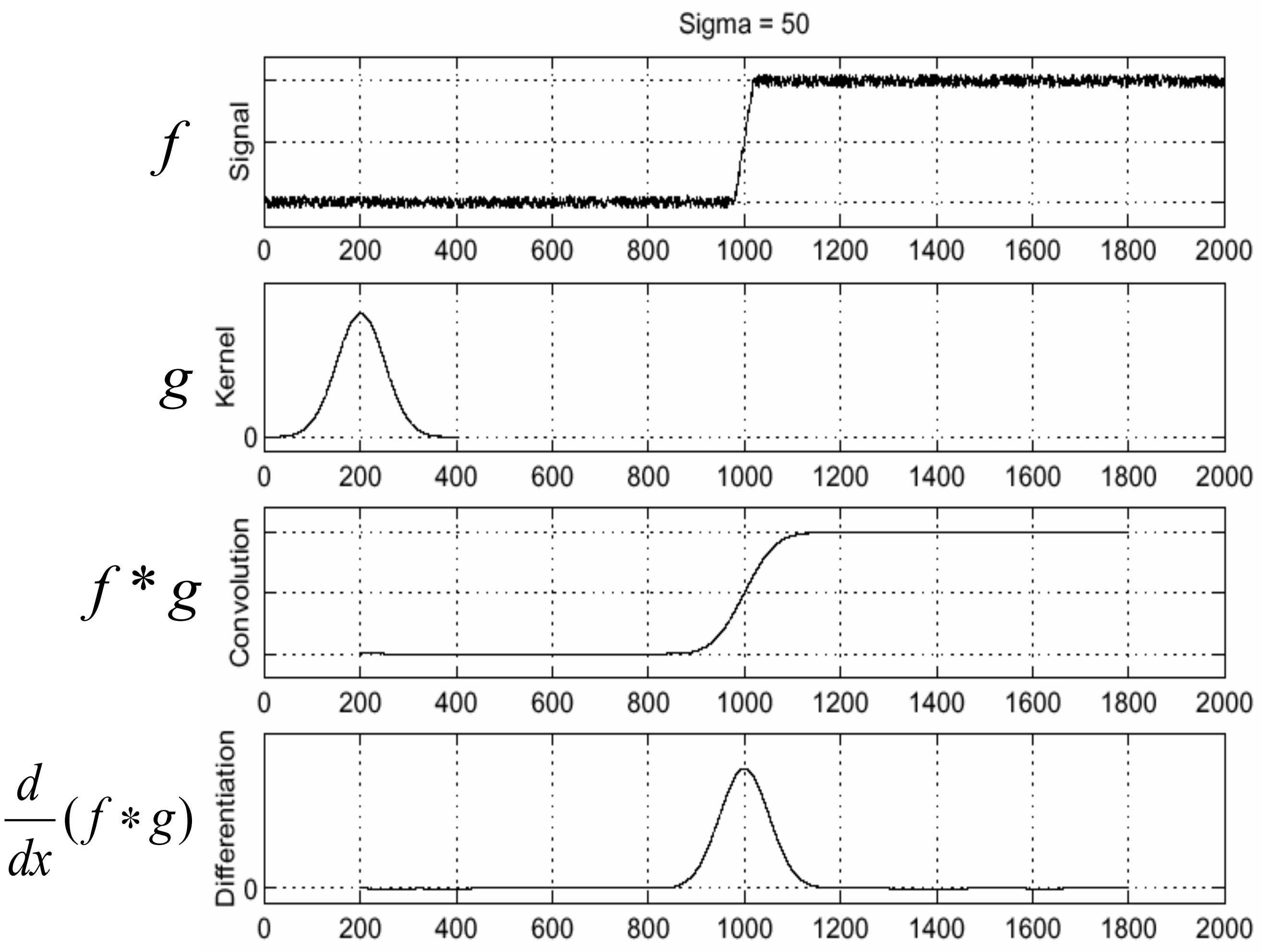
Source: S. Seitz

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different from their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Solution: smooth first



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

Source: S. Seitz

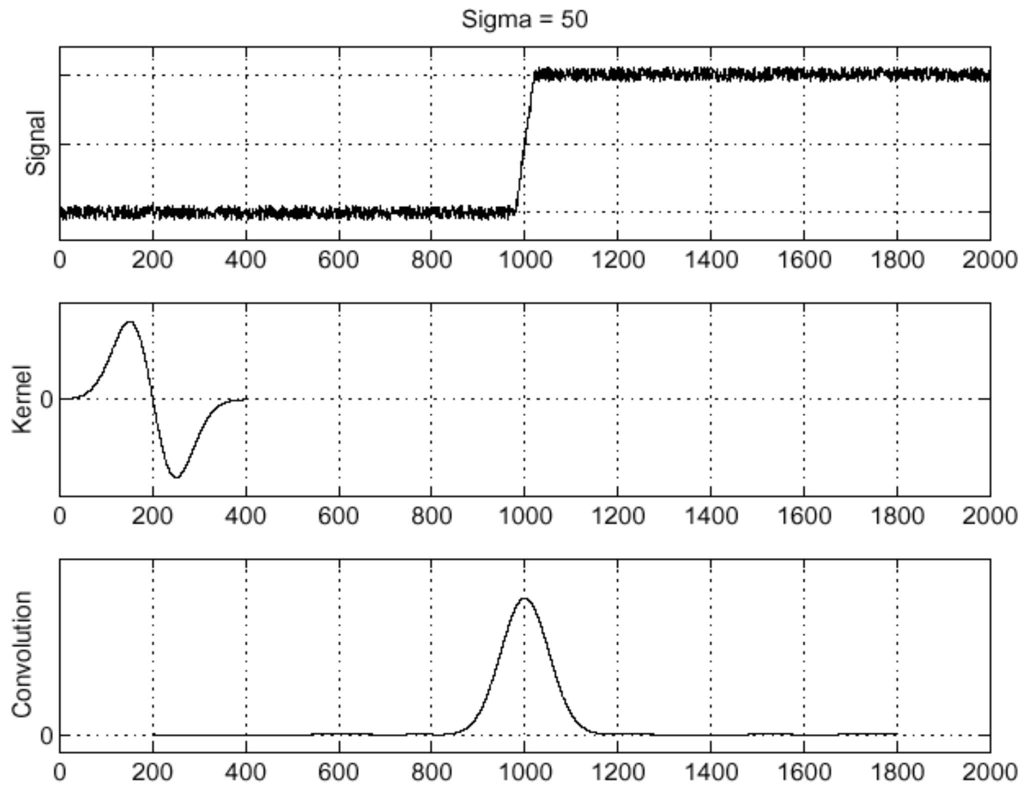
Derivative theorem of convolution

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

f

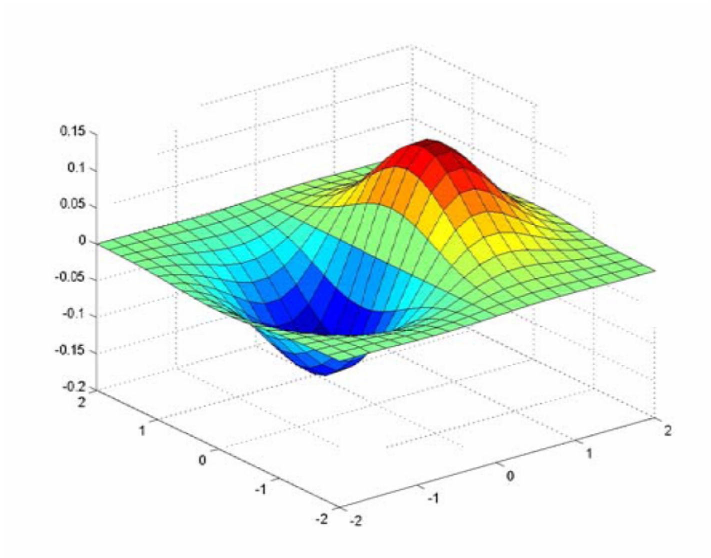
$\frac{d}{dx} g$

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

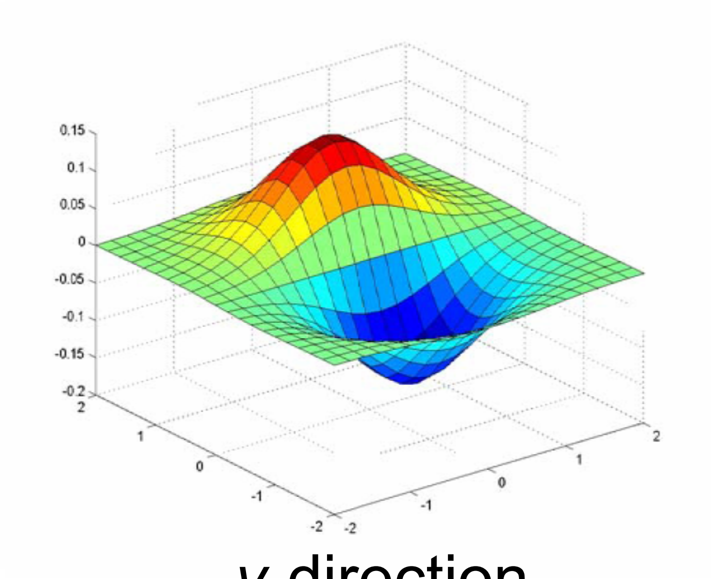


Source: S. Seitz

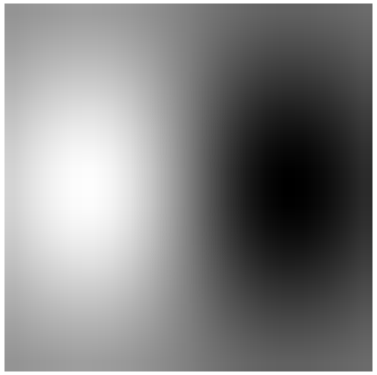
Derivative of Gaussian filter

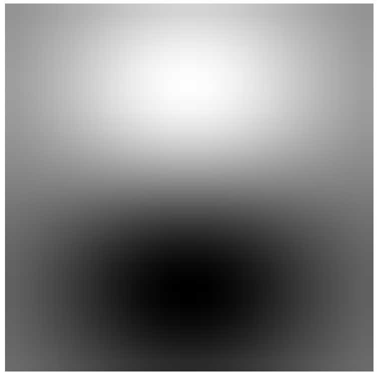


x-direction



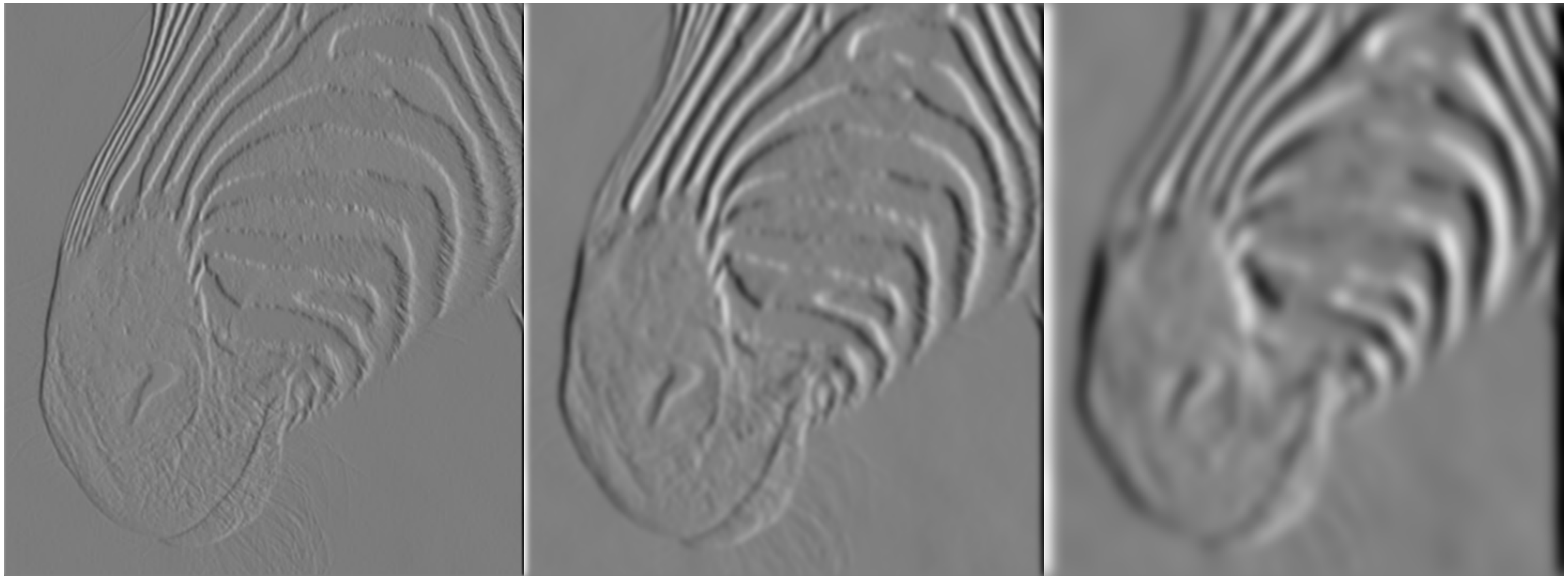
y-direction





Which one finds horizontal/vertical edges?

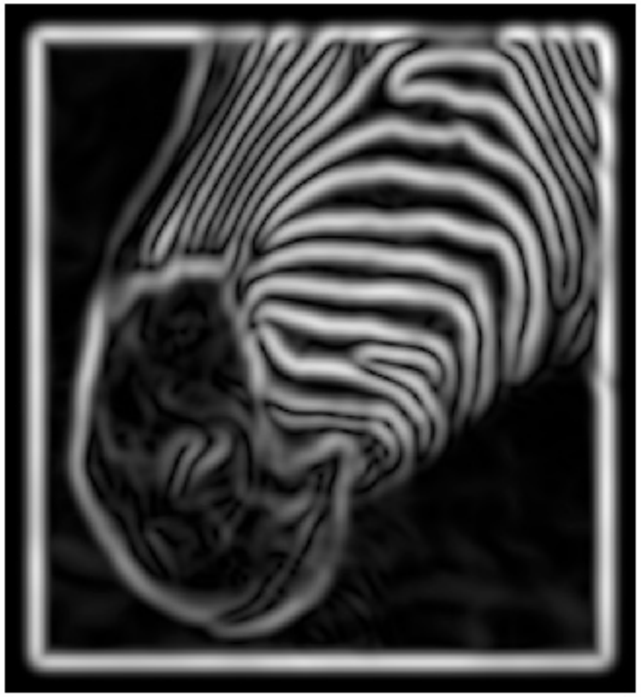
Scale of Gaussian derivative filter



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

Source: D. Forsyth

Implementation issues

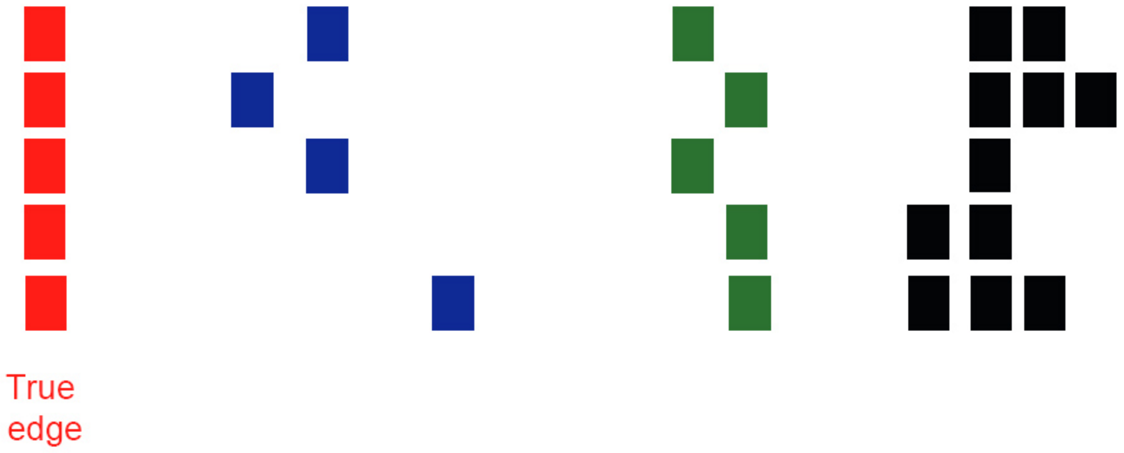


- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Designing an edge detector

- Criteria for an "optimal" edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Source: L. Fei-Fei

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization
- MATLAB: edge(image, 'canny')

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

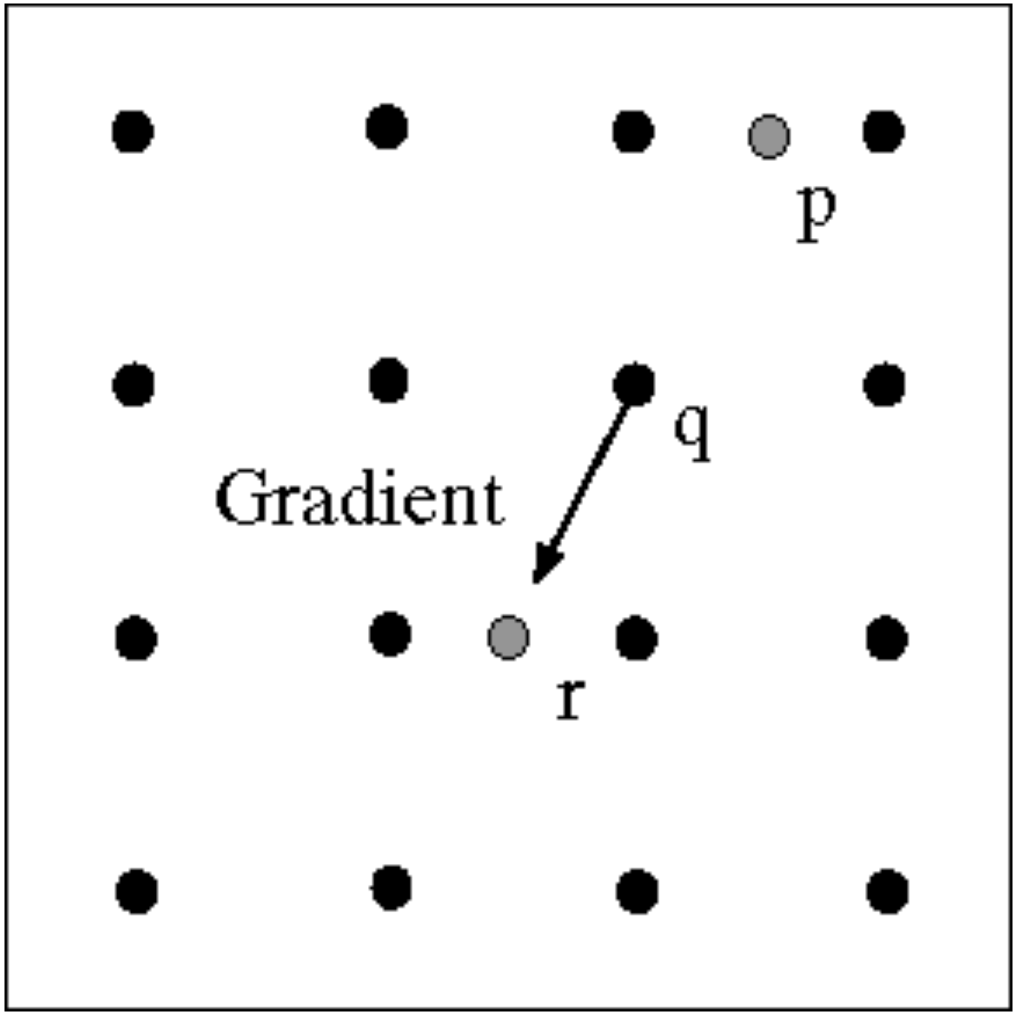
Source: L. Fei-Fei

Canny edge detector

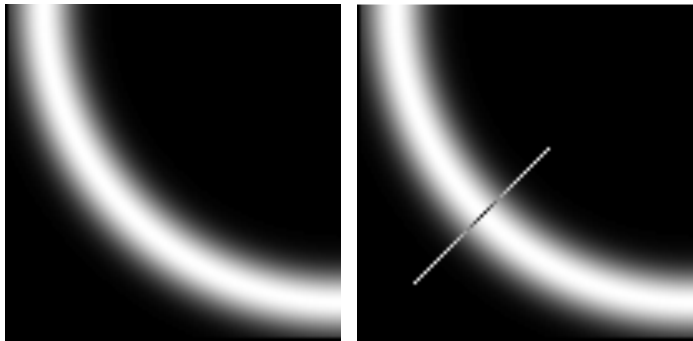
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
    - Thin multi-pixel wide “ridges” down to single pixel width

Source: D. Lowe, L. Fei-Fei

Non-maximum suppression



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

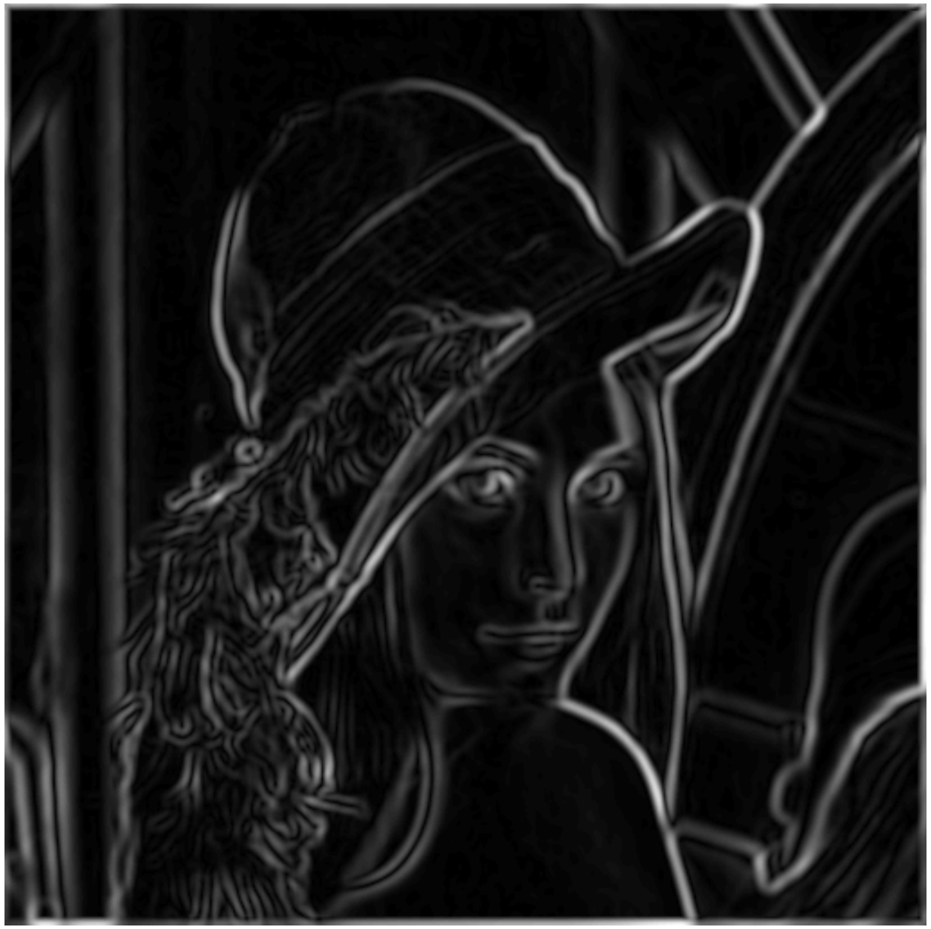


Source: D. Forsyth

Example

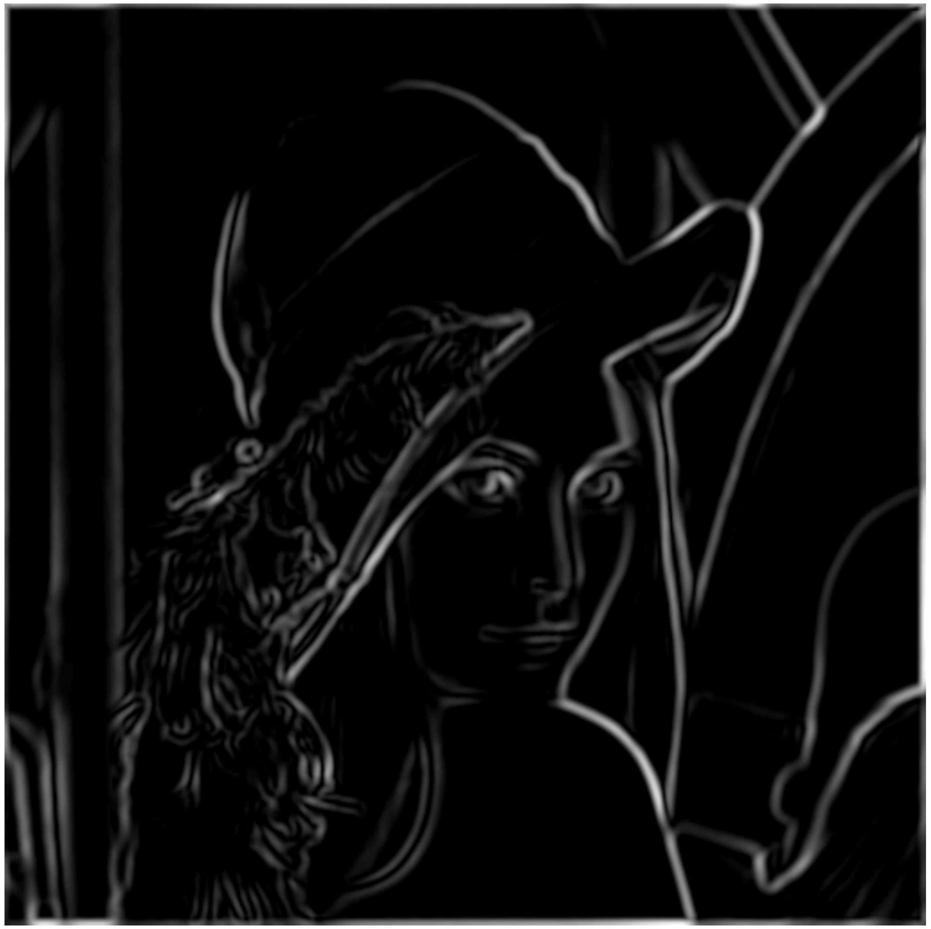
original image (Lena)

Example

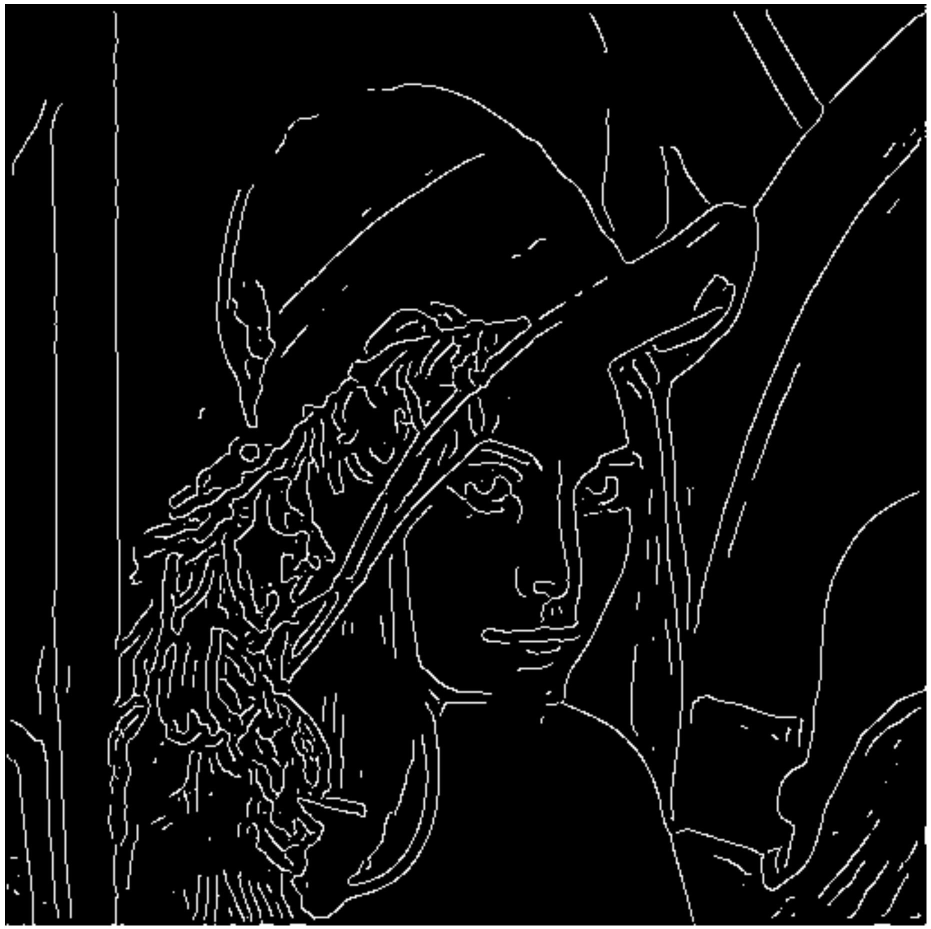


norm of the gradient

Example


thresholding

Example



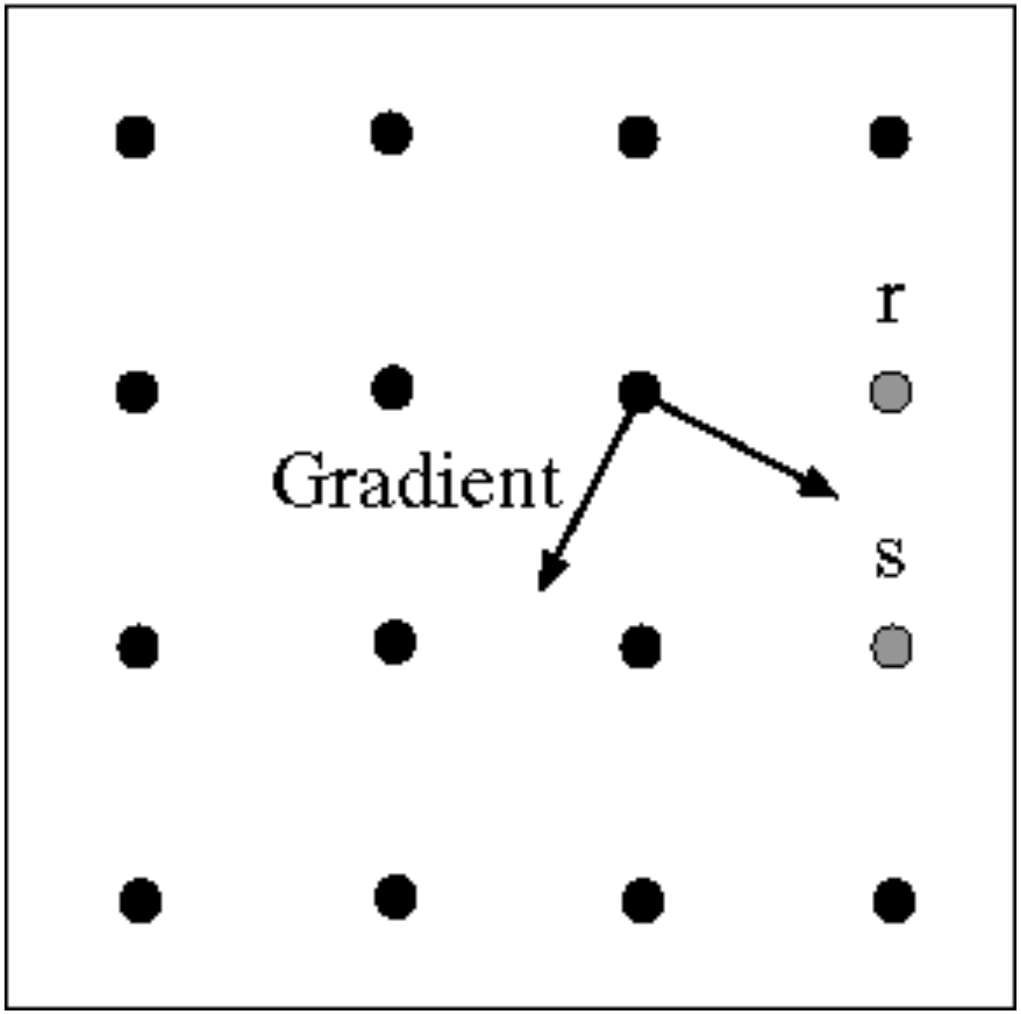
Non-maximum suppression

Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
    - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking of edge points

Source: D. Lowe, L. Fei-Fei

Edge linking



Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

Source: D. Forsyth

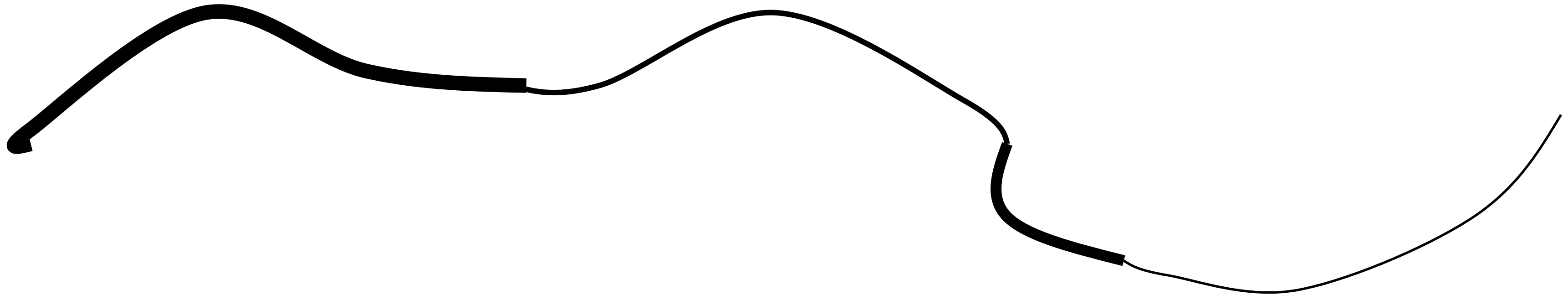
Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking of edge points
   - Hysteresis thresholding: use a higher threshold to start edge curves and a lower threshold to continue them

Source: D. Lowe, L. Fei-Fei

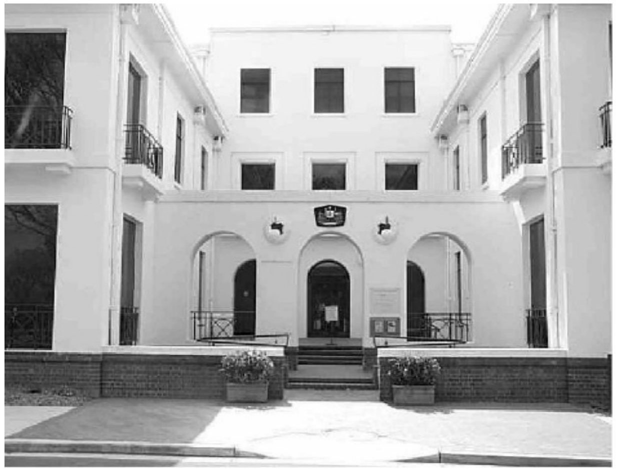
Hysteresis thresholding

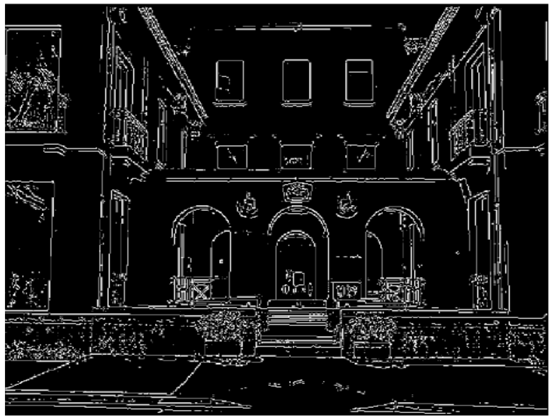
- Use a high threshold to start edge curves and a low threshold to continue them
  - Reduces *drop-outs*

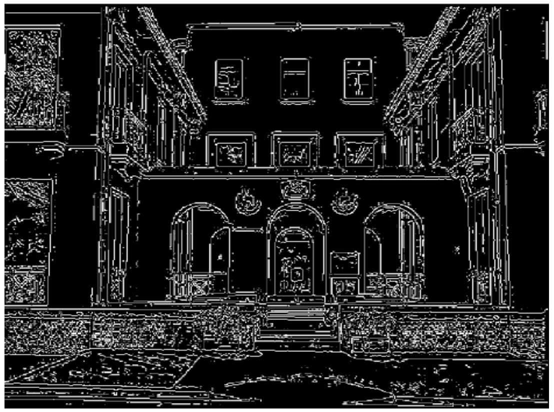


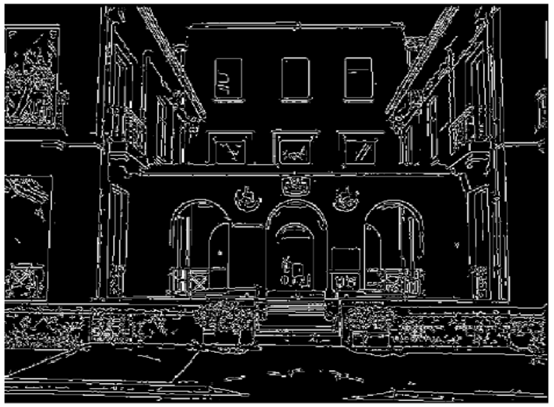
Source: S. Seitz

Hysteresis thresholding


original image

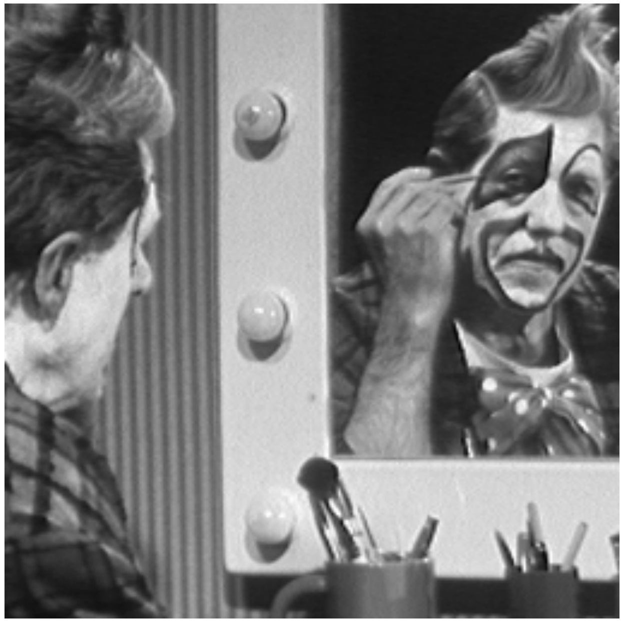

high threshold
(strong edges)


low threshold
(weak edges)

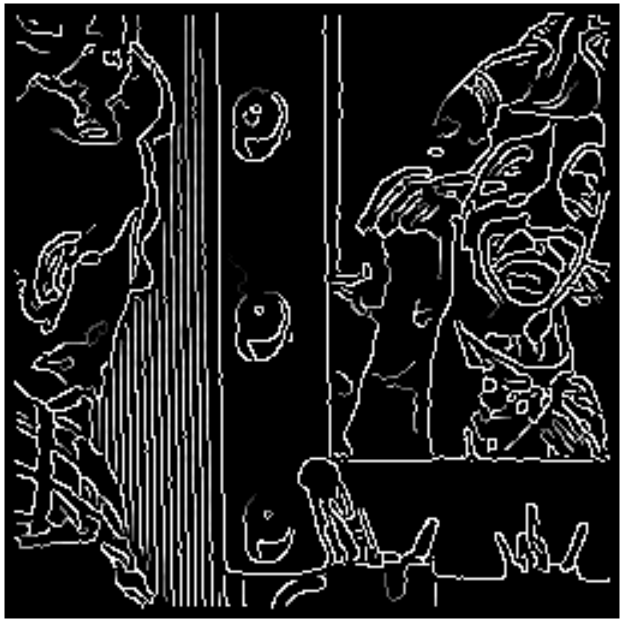

hysteresis threshold

Source: L. Fei-Fei

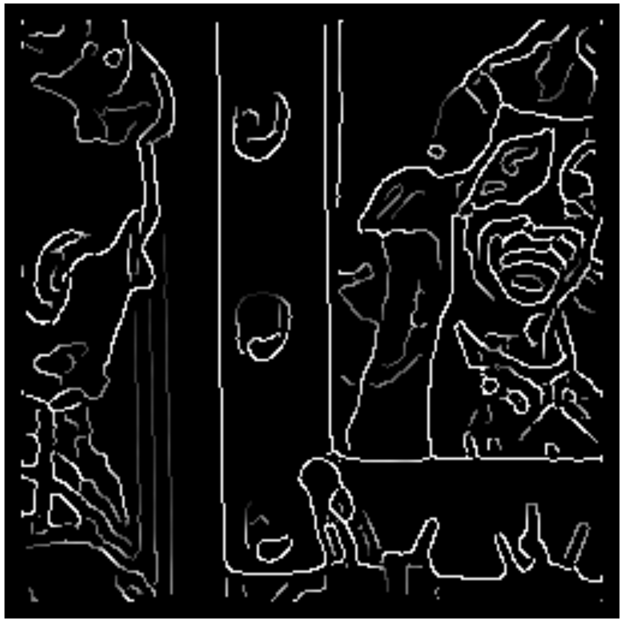
Effect of σ (Gaussian kernel spread/size)



original



Canny with $\sigma = 1$



Canny with The image cannot be displayed. Your computer may not have enough

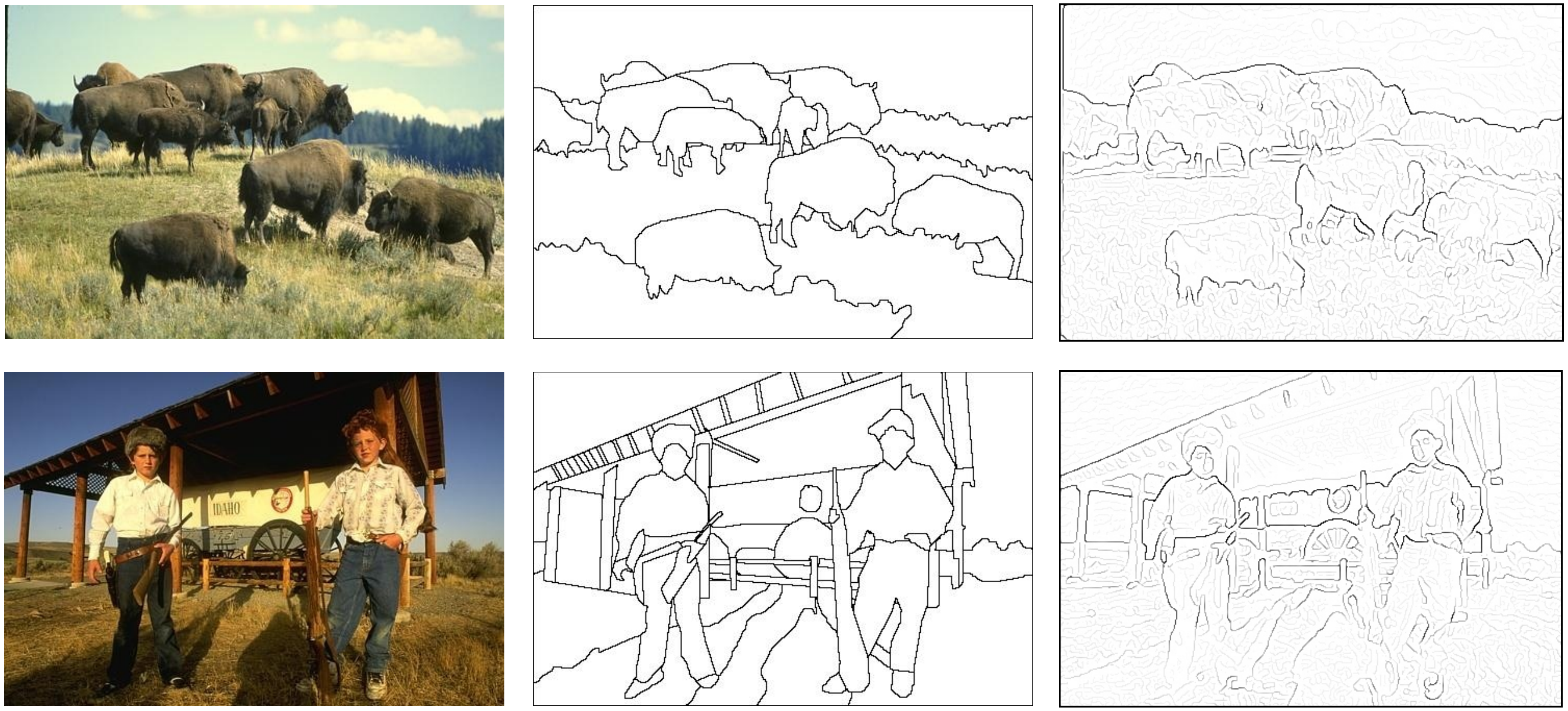
The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

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/