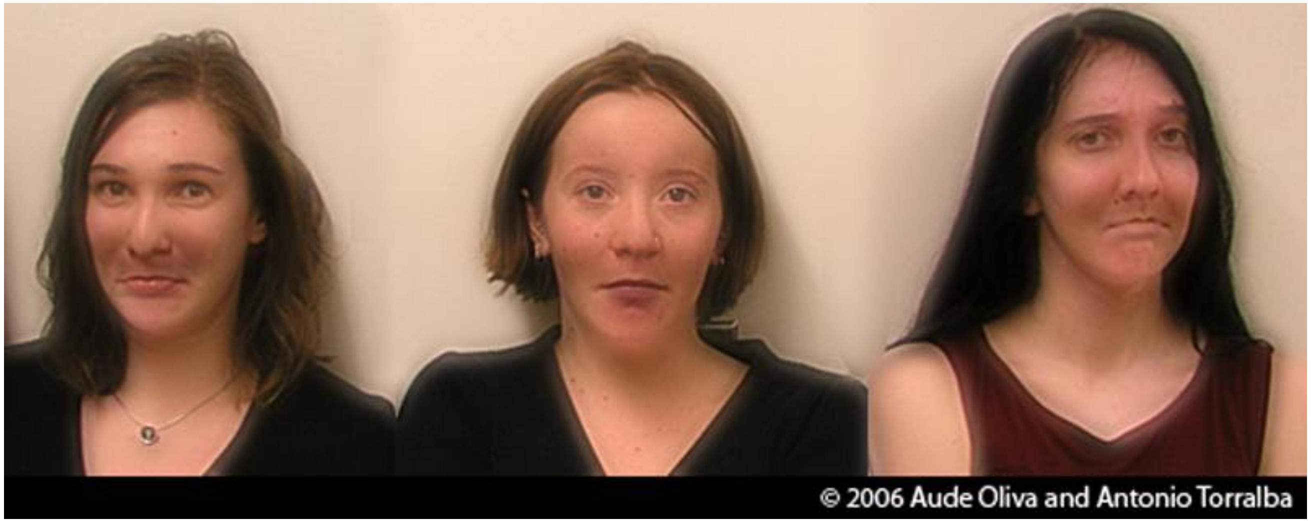
CS6670: Computer Vision

Noah Snavely

Lecture 2: Image filtering

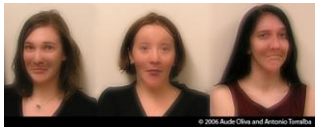


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS6670: Computer Vision

Noah Snavely

Lecture 2: Image filtering

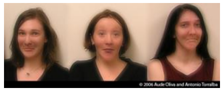


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS6670: Computer Vision

Noah Snavely

Lecture 2: Image filtering

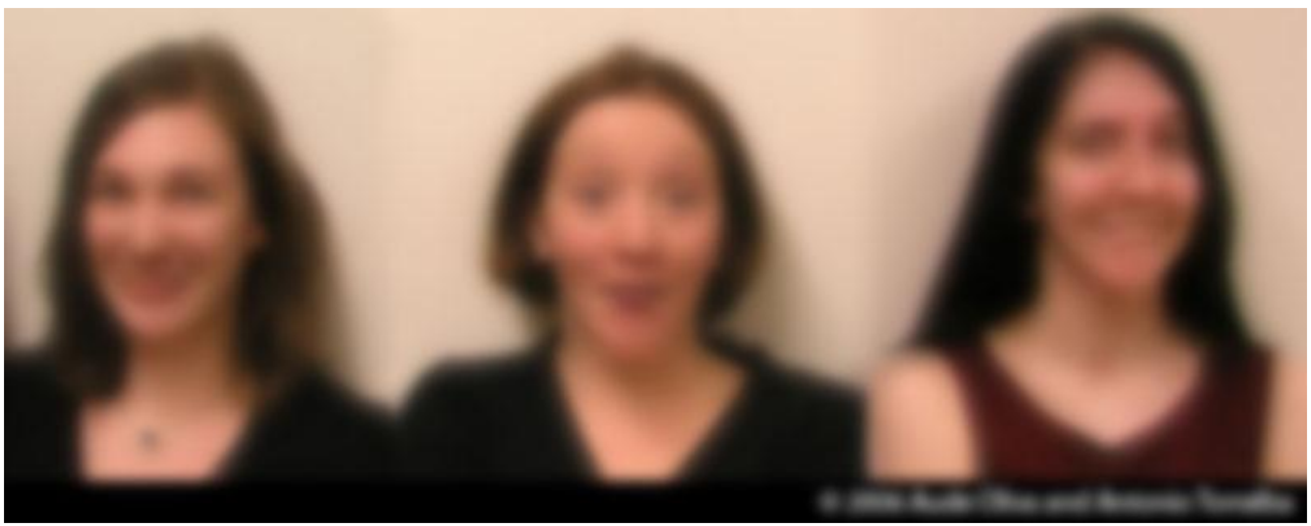


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS6670: Computer Vision

Noah Snavely

Lecture 2: Image filtering



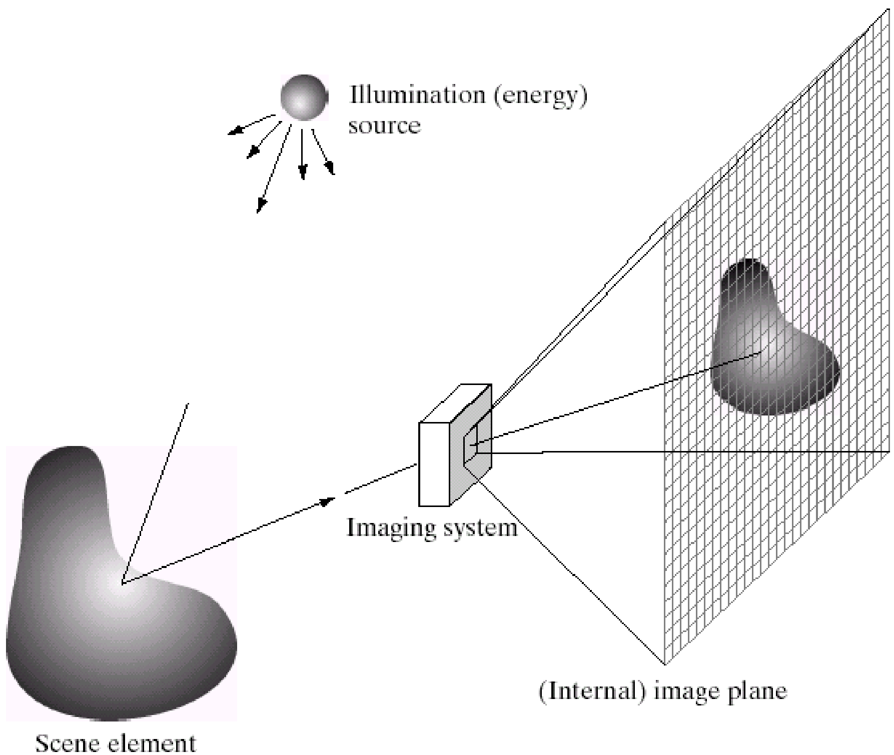
Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

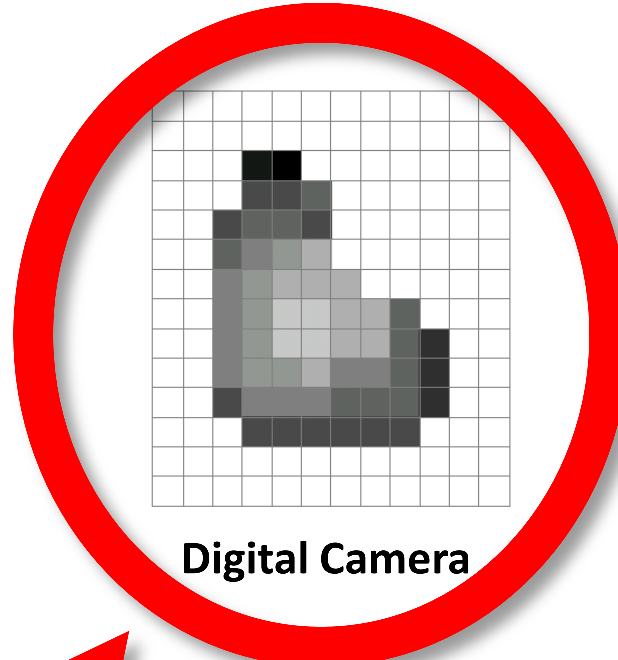
Reading

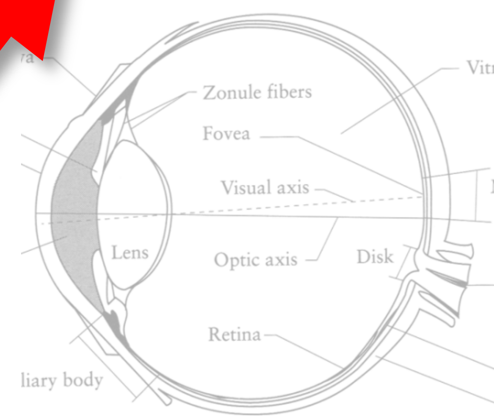
- Szeliski, Chapter 3.1-3.2

What is an image?

What is an image?







The Eye

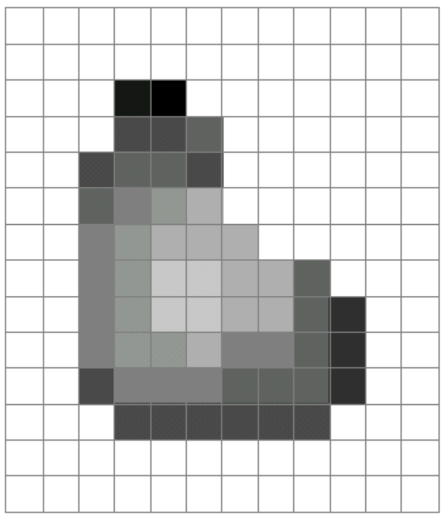
We'll focus on these in this class

(More on this process later)

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



=

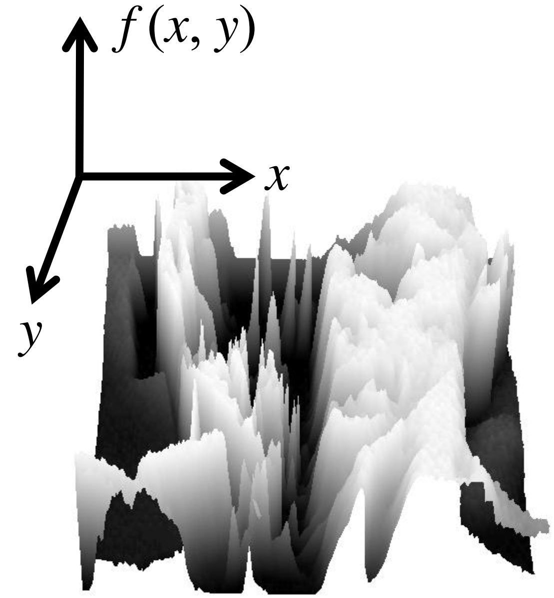
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

What is an image?

- We can think of a (grayscale) image as a **function**, f, from R^2 to R (or a 2D *signal*):
  - $f(x,y)$ gives the **intensity** at position (x,y)

snoop

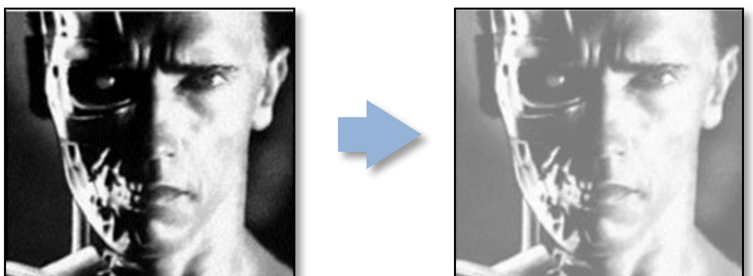


3D view

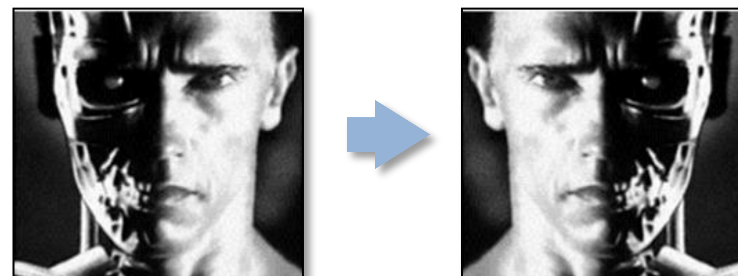
  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

- We’ll talk about a special kind of operator, *convolution* (linear filtering)

Question: Noise reduction

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

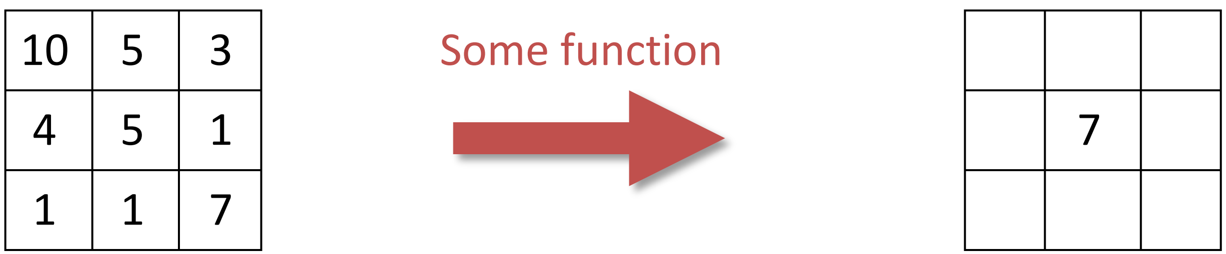
Take lots of images and average them!

What's the next best thing?

Source: S. Seitz

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel



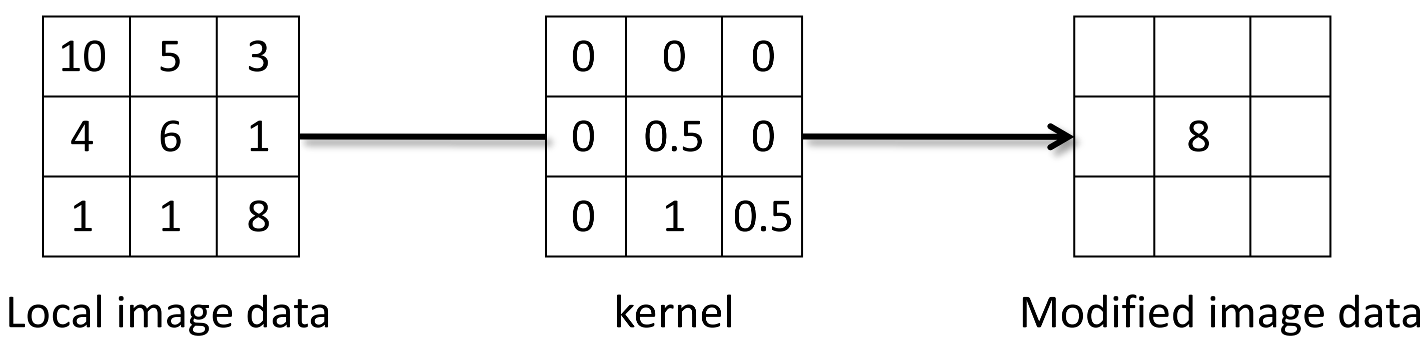
Local image data

Modified image data

Source: L. Zhang

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)



Source: L. Zhang

Cross-correlation

Let F be the image, H be the kernel (of size 2k+1 x 2k+1), and G be the output image

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

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

$$G = H \otimes F$$

Convolution

- Same as cross-correlation, except that the kernel is “flipped” (horizontally and vertically)

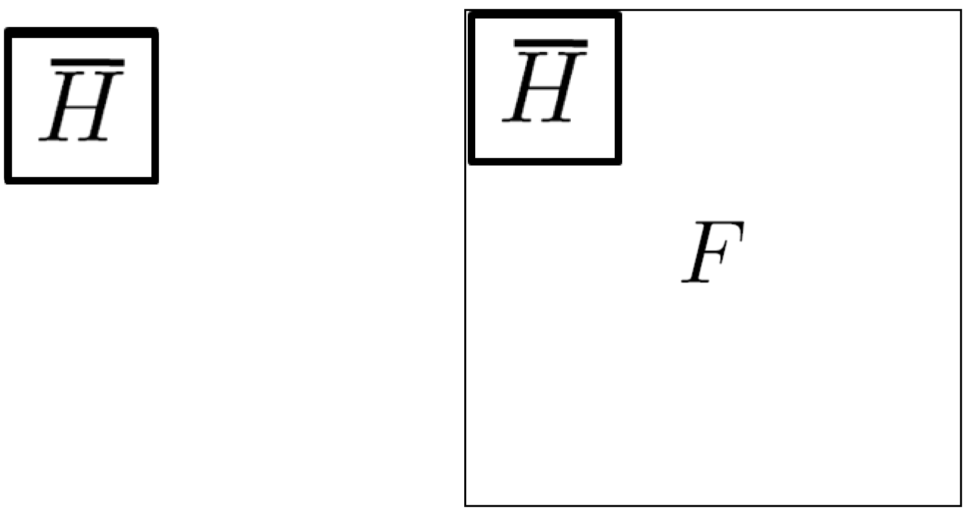
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Convolution



Adapted from F. Durand

Mean filtering

H

$*$

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

F

$=$

	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	

G

Linear filters: examples

Original

$*$

0	0	0
0	1	0
0	0	0

$=$

Identical image

Source: D. Lowe

Linear filters: examples

Original

*

0	0	0
1	0	0
0	0	0

=

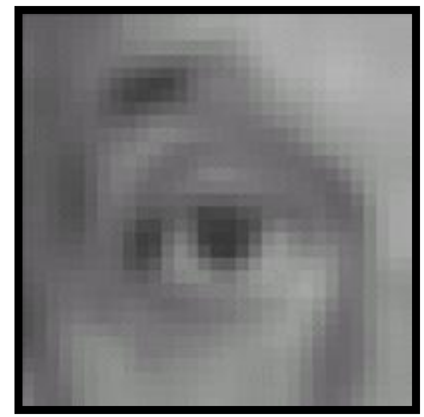
Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples

Original

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



Blur (with a mean filter)

Source: D. Lowe

Linear filters: examples

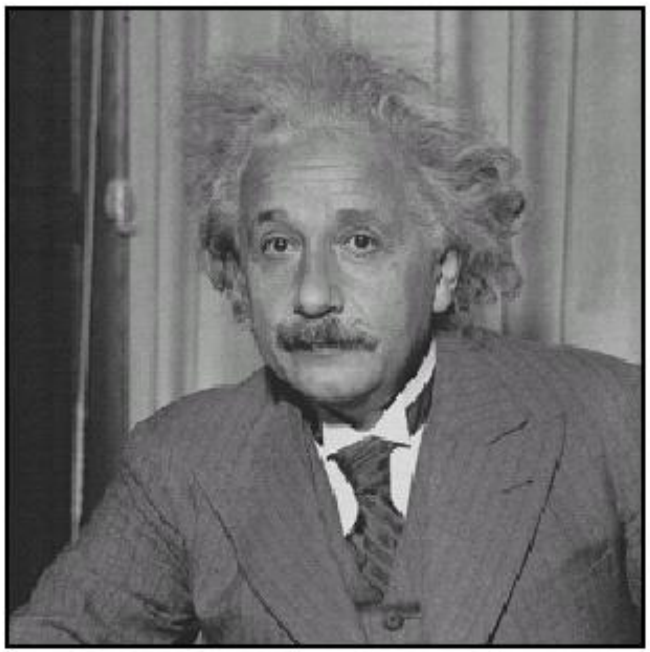
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

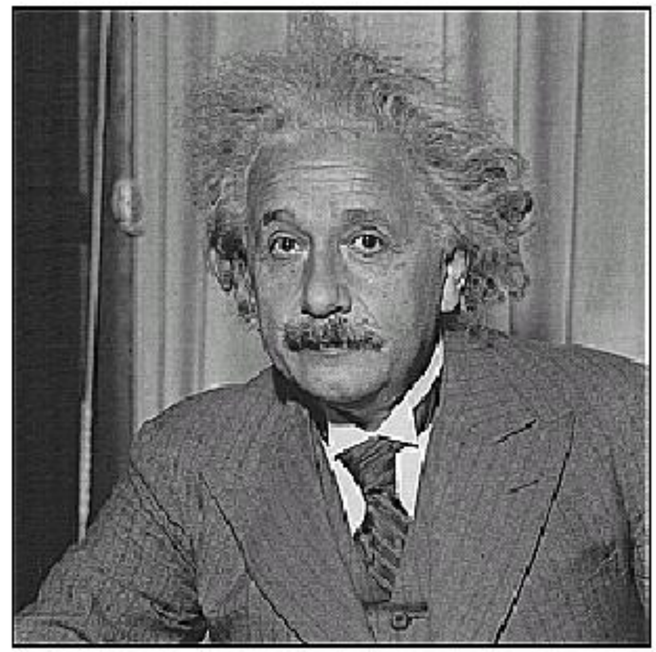
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



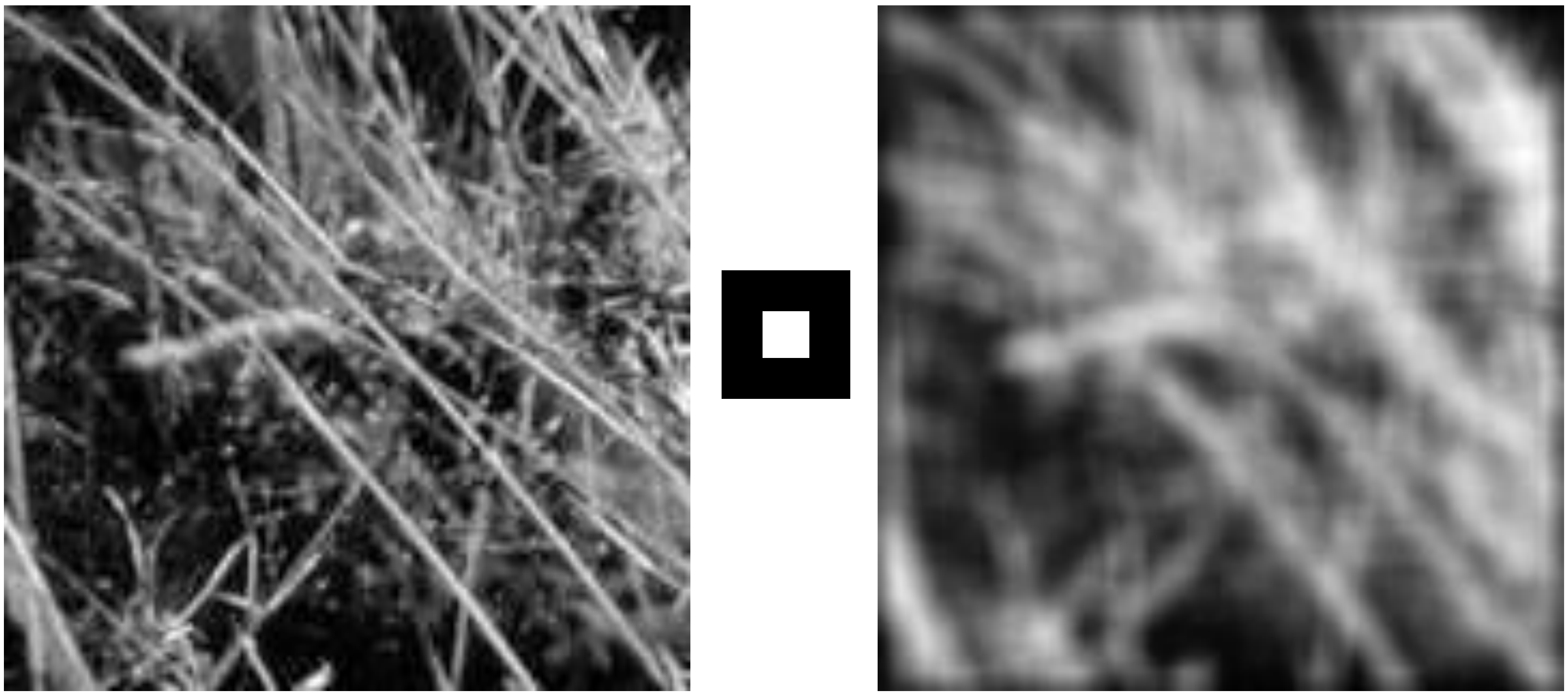
before



after

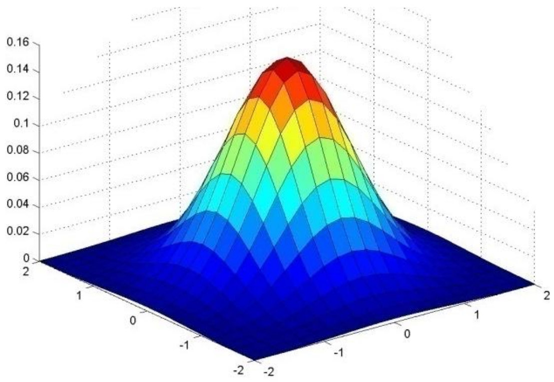
Source: D. Lowe

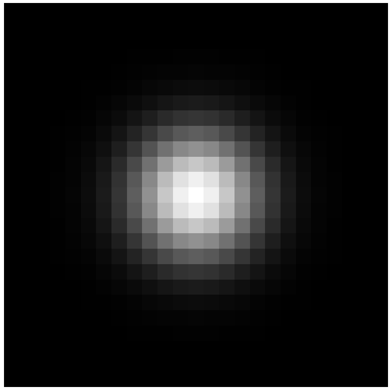
Smoothing with box filter revisited



Source: D. Forsyth

Gaussian Kernel

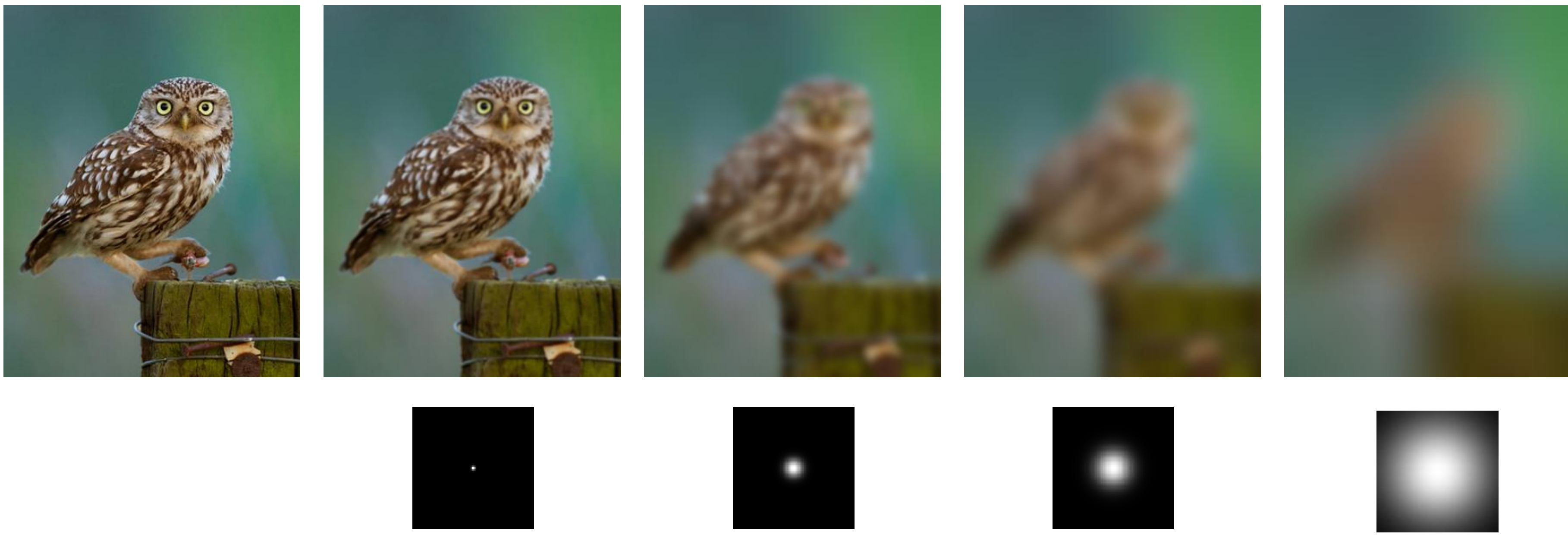




$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

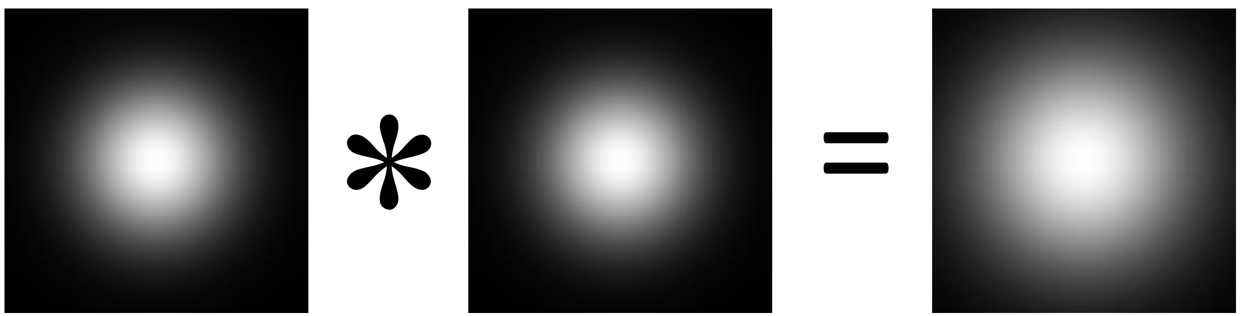
Gaussian filters



σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Gaussian filter

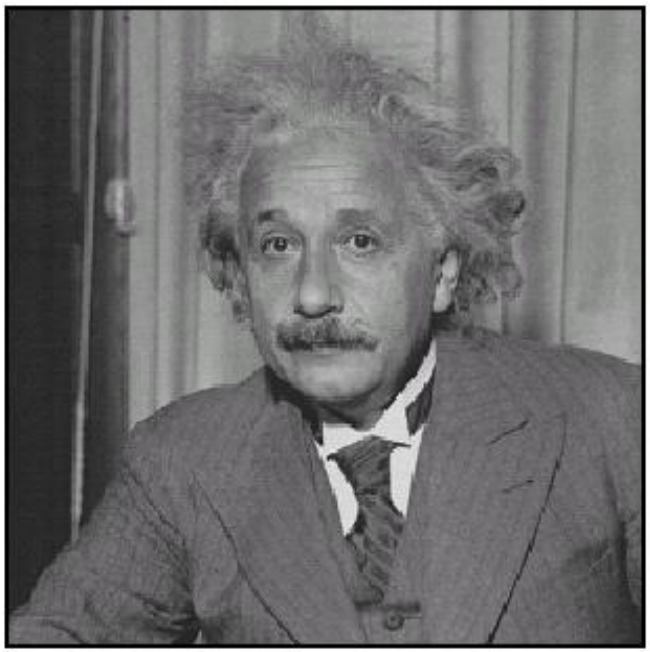
- Removes "high-frequency" components from the image (low-pass filter)
- Convolution with self is another Gaussian



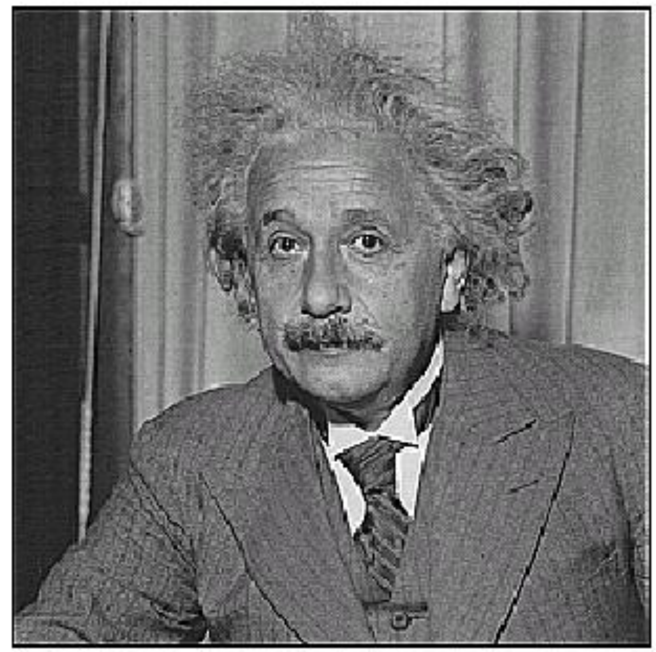
  - Convolving two times with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

Sharpening



before



after

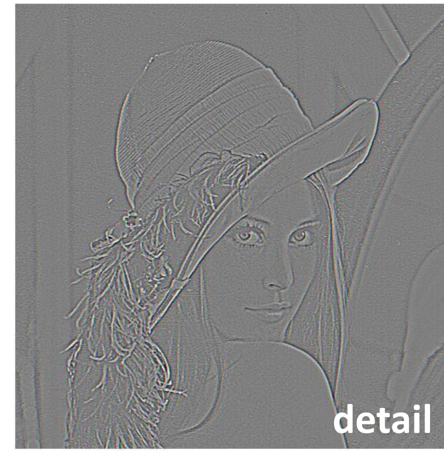
Source: D. Lowe

Sharpening revisited

- What does blurring take away?

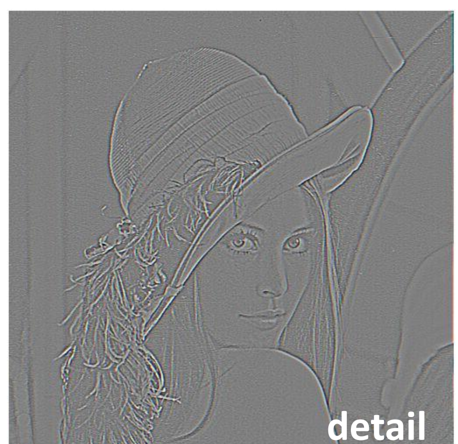
−

=

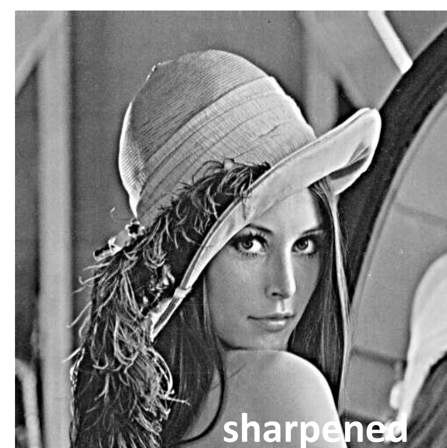


Let's add it back:

+ α



=



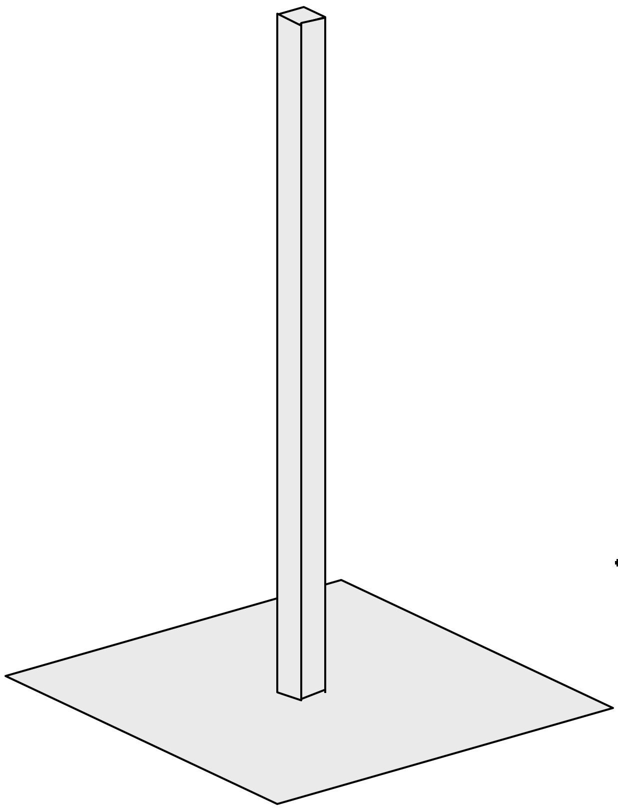
Source: S. Lazebnik

Sharpen filter

$$\underset{\uparrow \text{ image}}{F} + \alpha\left(F - \underbrace{F * H}_{\text{blurred image}}\right)$$

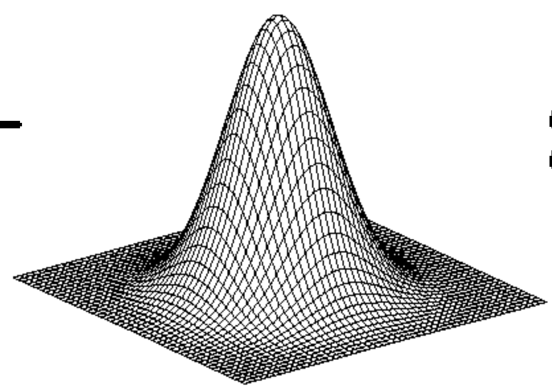
$\uparrow$

unit impulse (identity)



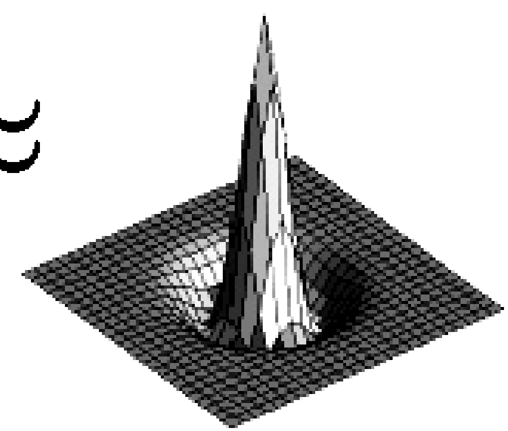
scaled impulse

$-$



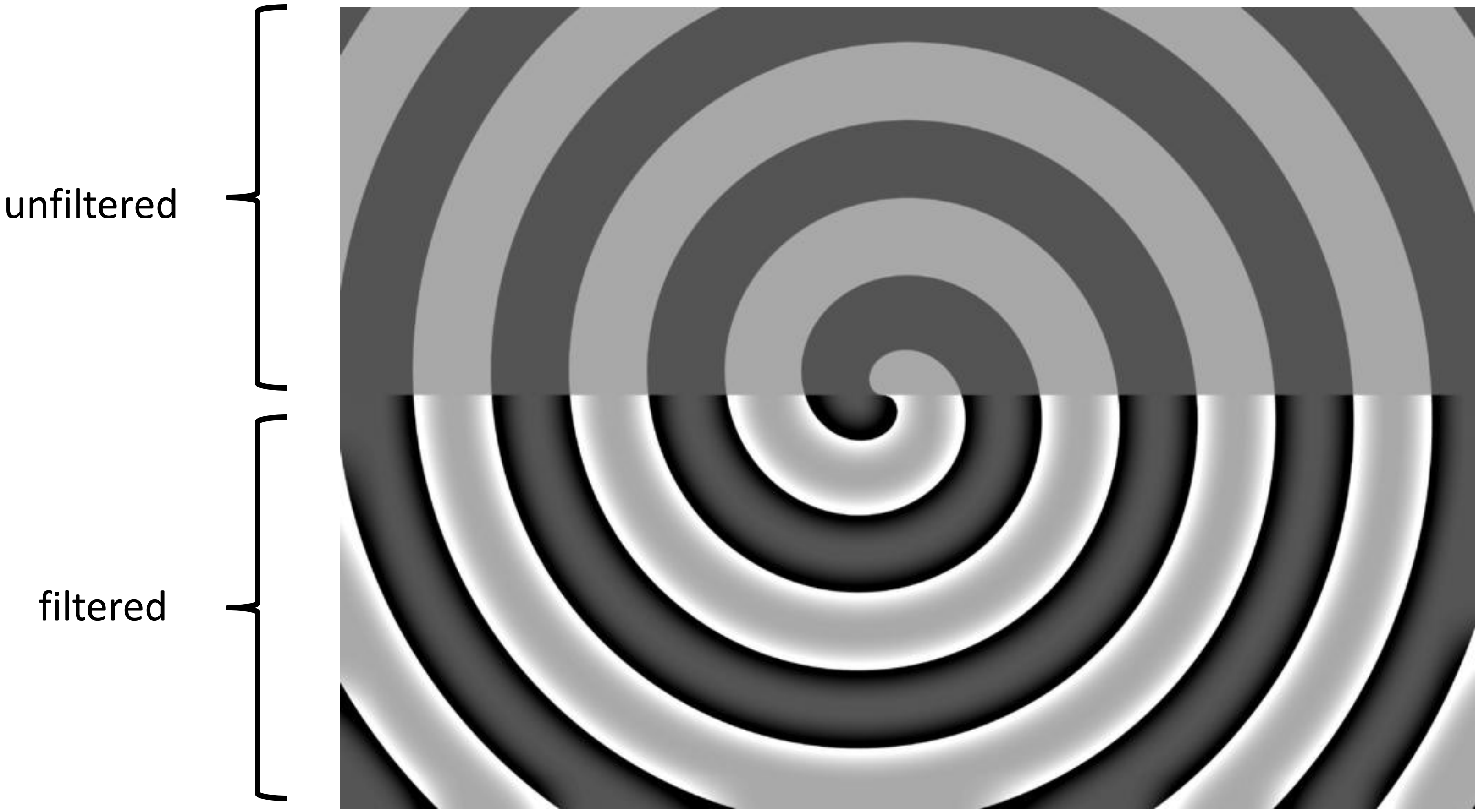
Gaussian

$\approx$



Laplacian of Gaussian

Sharpen filter



Convolution in the real world

Camera shake

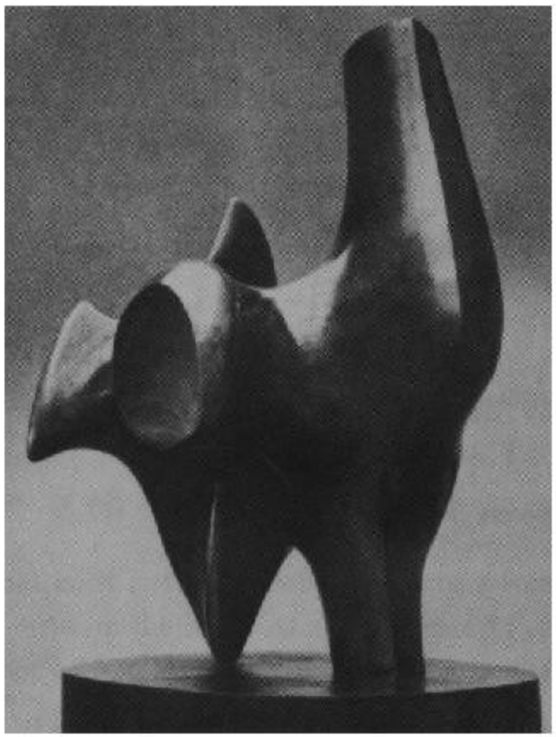
Source: Fergus, *et al.* "Removing Camera Shake from a Single Photograph", SIGGRAPH 2006

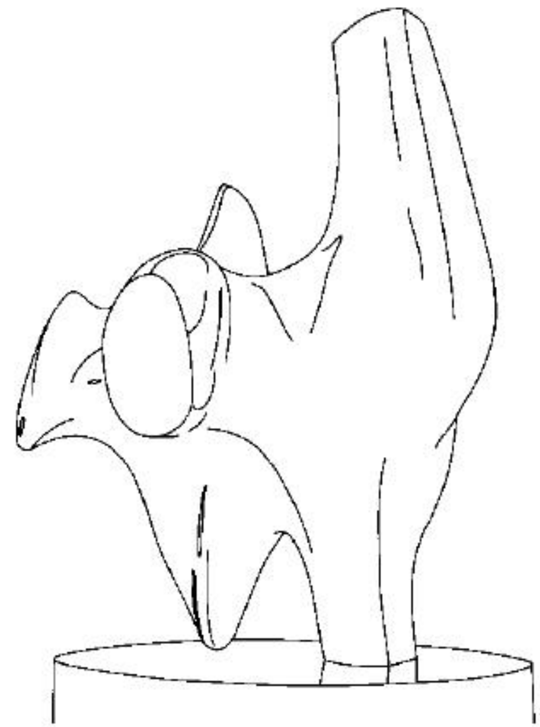
Bokeh: Blur in out-of-focus regions of an image.

Source: http://lullaby.homepage.dk/diy-camera/bokeh.html

Questions?

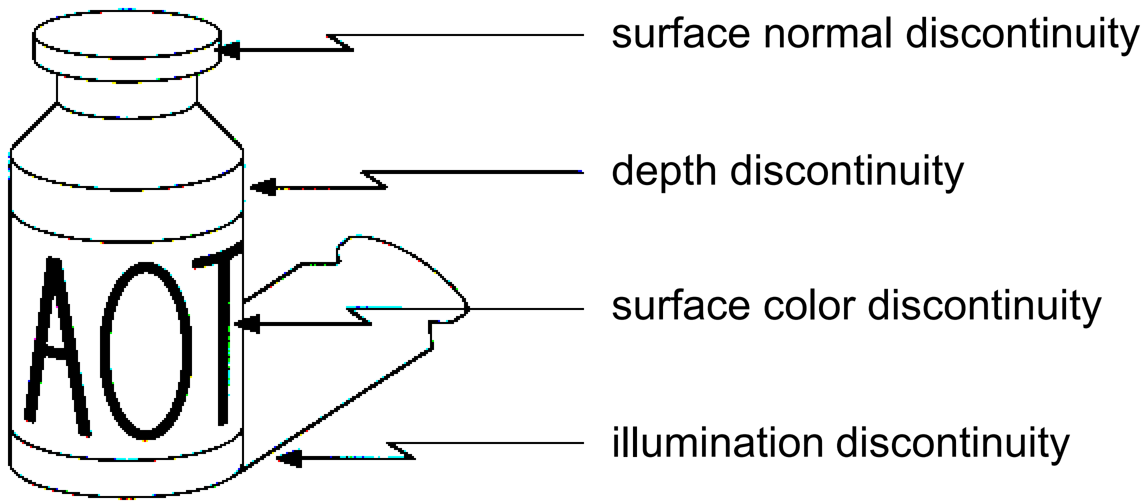
Edge detection





- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of Edges



- Edges are caused by a variety of factors

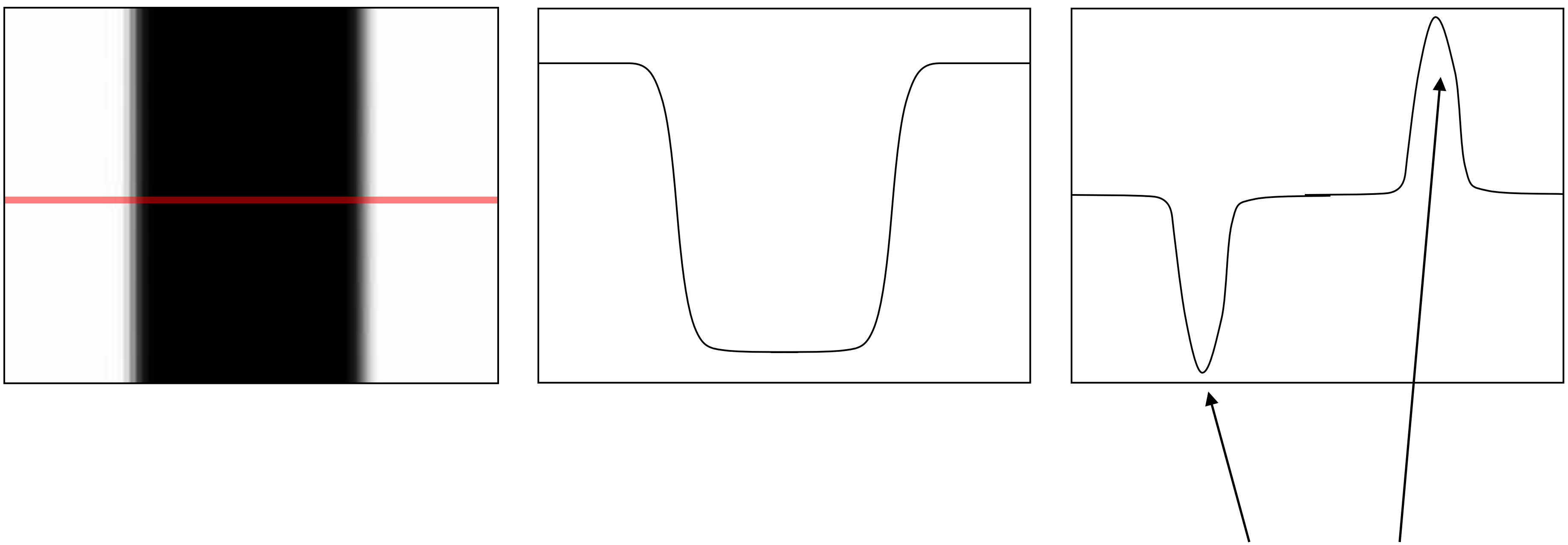
Characterizing edges

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

image

intensity function (along horizontal scanline)

first derivative



edges correspond to extrema of derivative

Source: L. Lazebnik

Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f,* then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

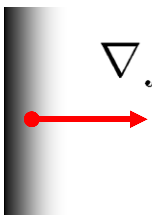
H_y

Source: S. Seitz

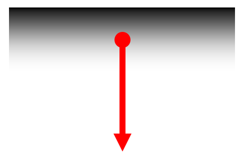
Image gradient

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

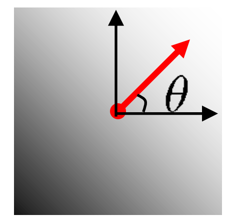
The gradient points in the direction of most rapid increase in intensity



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



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



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

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

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

The gradient direction is given by:

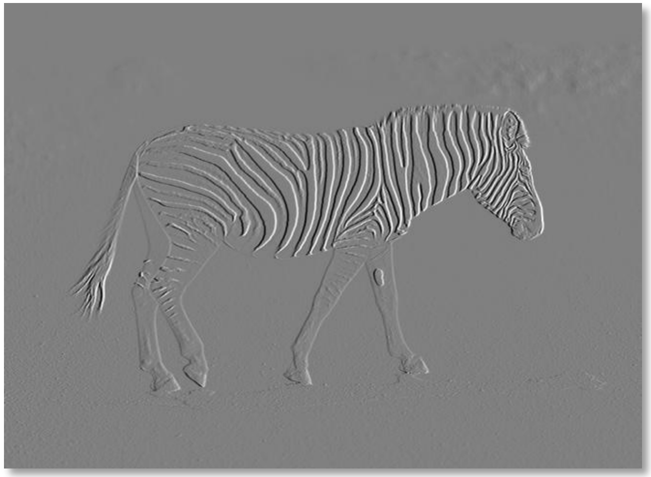
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

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

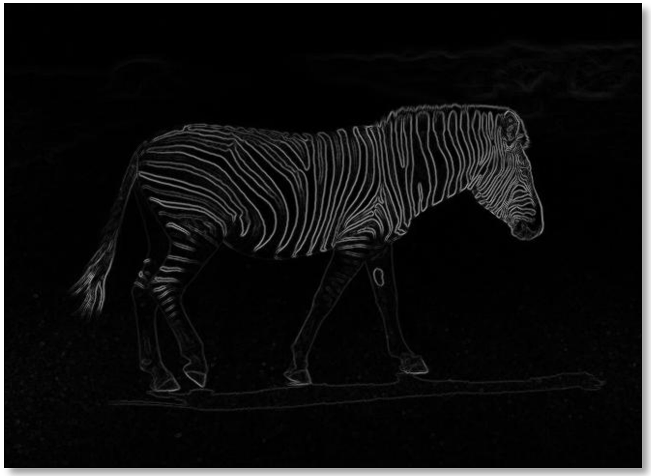
Source: Steve Seitz

Image gradient

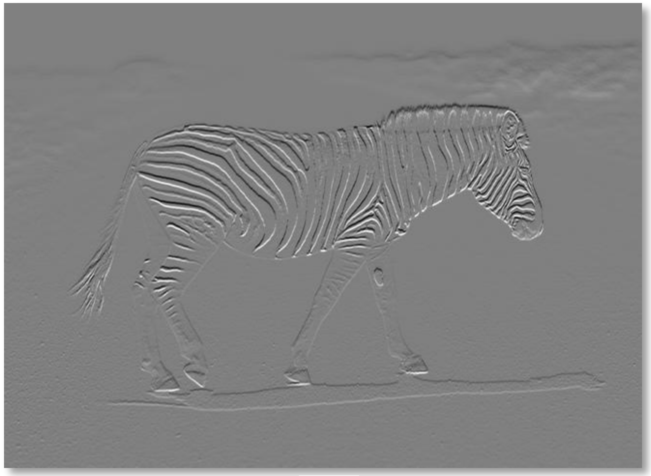
$$f$$



$$\frac{\partial f}{\partial x}$$



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



$$\frac{\partial f}{\partial y}$$

The Sobel operator

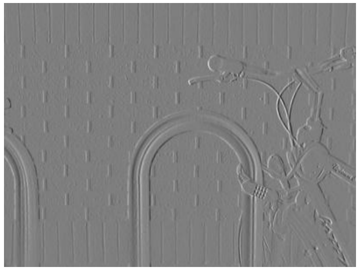
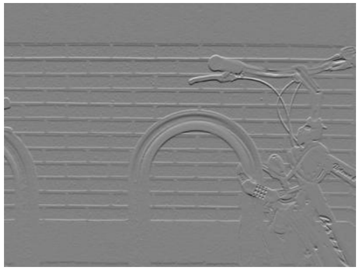
- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value

Sobel operator: example


Source: Wikipedia