CSCI 497/597P: Computer Vision

Scott Wehrwein

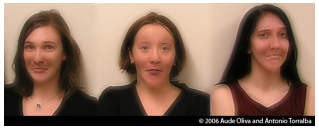
Images and image filtering

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

CSCI 497/597P: Computer Vision

Scott Wehrwein

Images and image filtering



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

CSCI 497/597P: Computer Vision

Scott Wehrwein

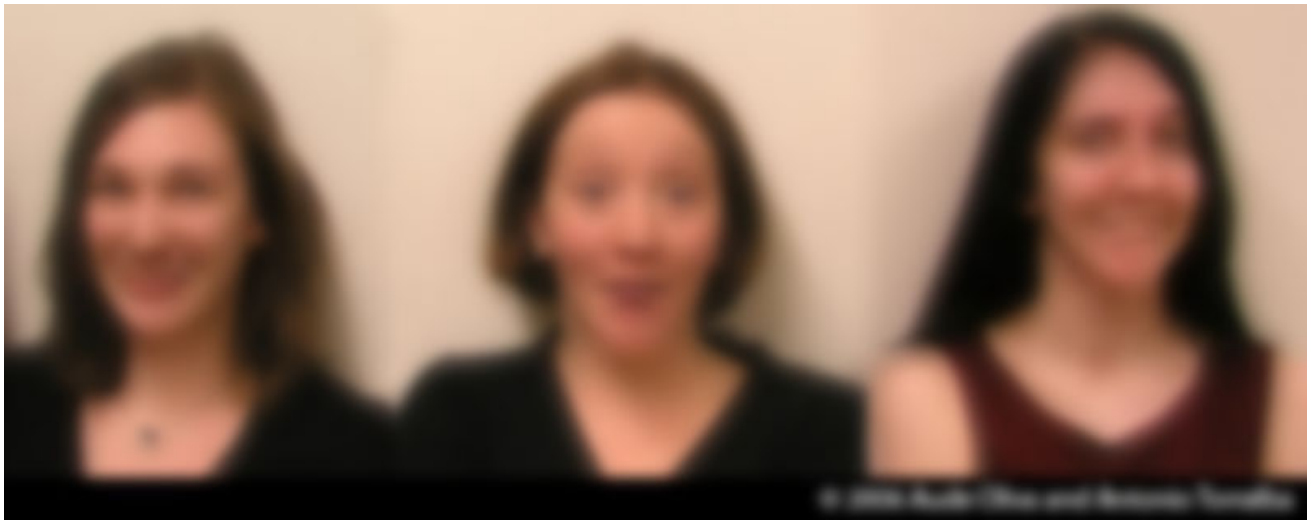
Images and image filtering

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

CSCI 497/597P: Computer Vision

Scott Wehrwein

Images and image filtering



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

Reading

- Szeliski, Chapter 3.1-3.2

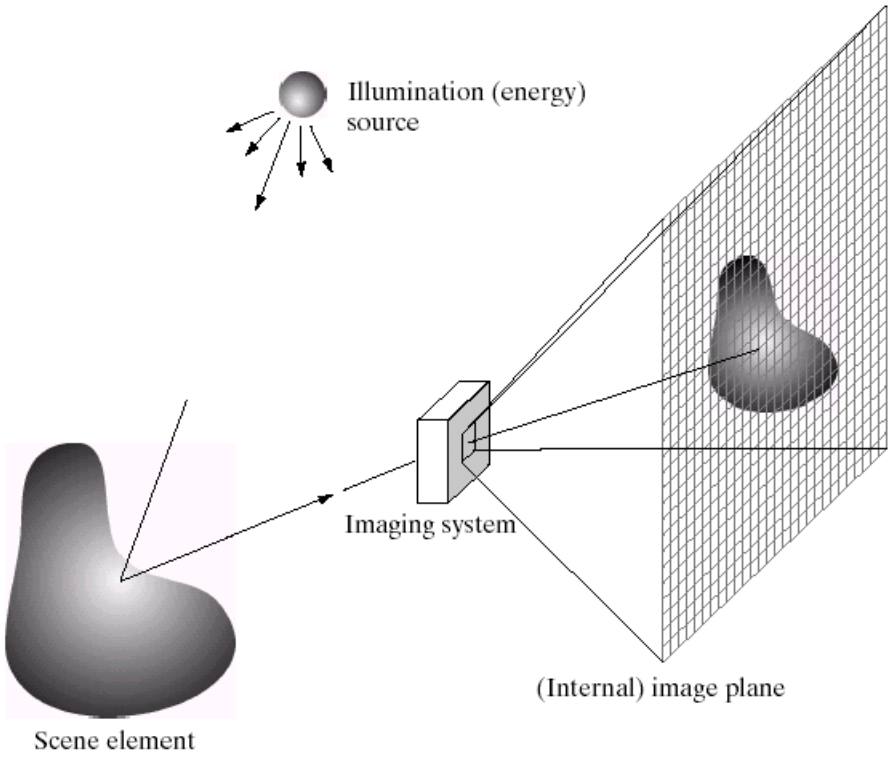
Announcements

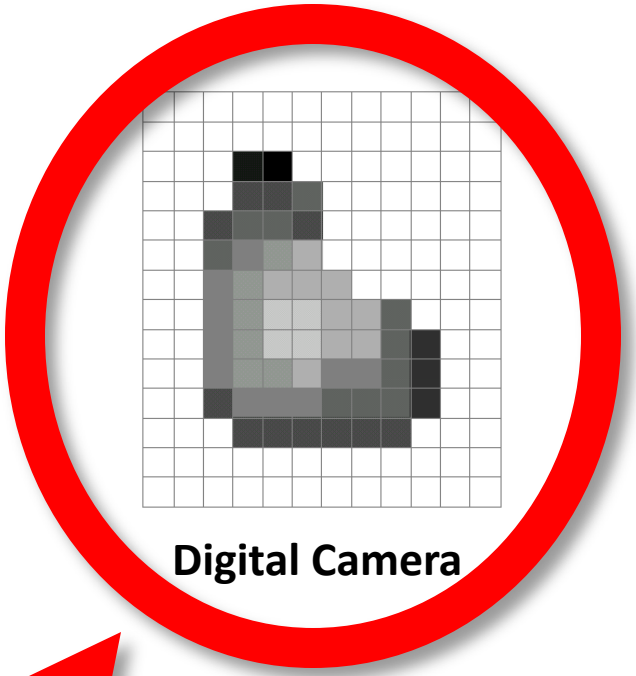
- You should have gotten a Piazza invite link
- Let me know if you still need to be added

- Project 1 (Hybrid Images) out soon (by the end of the weekend)
  - This project will be done solo
  - You'll need a github account and familiarity with git

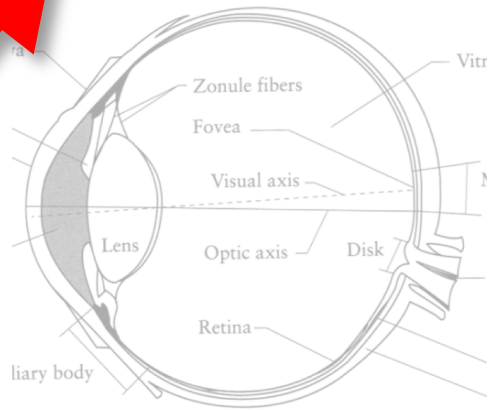
What is an image?

What is an image?





We'll focus on these in this class

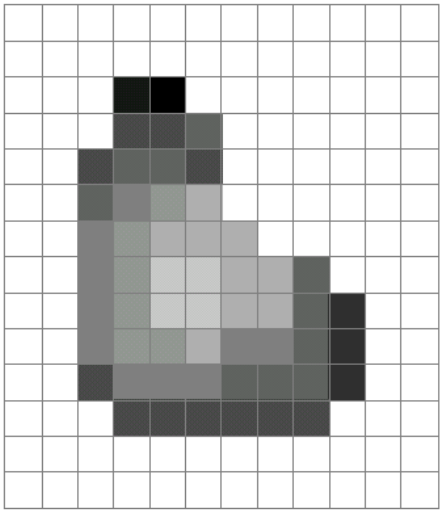


(More on this process later)

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



=

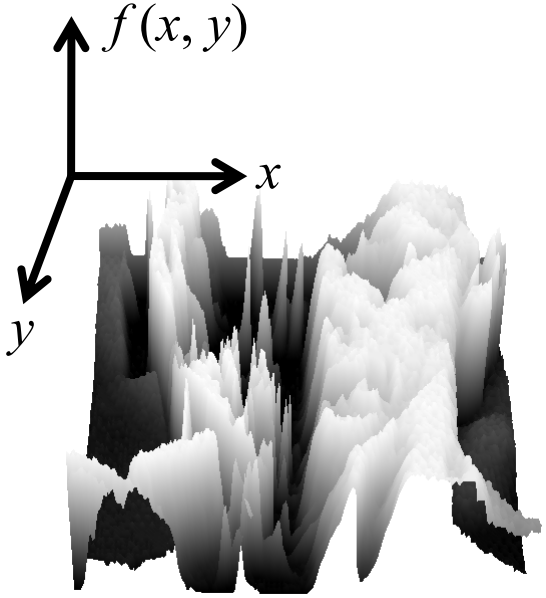
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	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:
  - $f(x,y)$ gives the **intensity** at position (x,y)

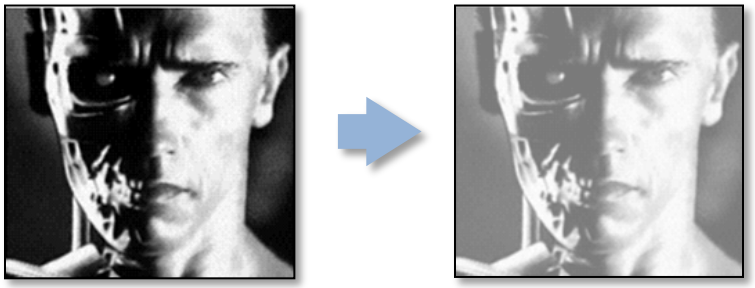
image



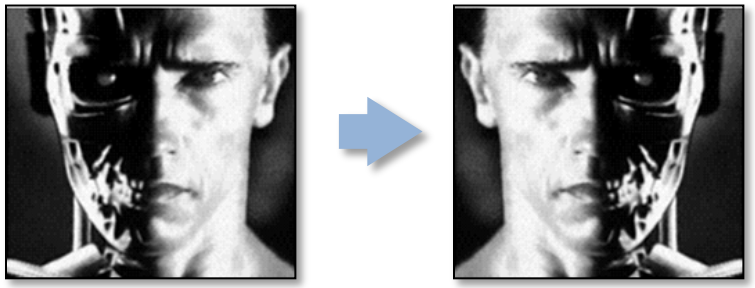
  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

Image transformations

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



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



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

- Today we'll talk about a special kind of operator, *convolution* (linear filtering)

Filters

- Filtering
  - Form a new image whose pixels are a combination of the original pixels
- Why?
  - To get useful information from images
    - E.g., extract edges or contours (to understand shape)
  - To enhance the image
    - E.g., to remove noise
    - E.g., to sharpen and “enhance image” a la CSI

Canonical Image Processing problems

- Image Restoration
  - denoising
  - deblurring
- Image Compression
  - JPEG, JPEG2000, MPEG..
- Computing Field Properties
  - optical flow
  - disparity
- Locating Structural Features
  - corners
  - edges

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

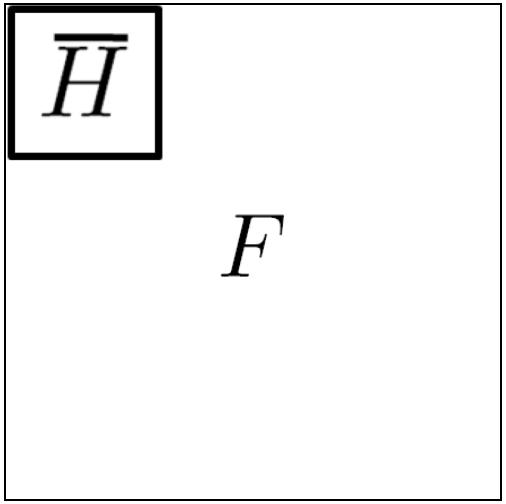
- Modify the pixels in an image based on some function of a local neighborhood of each pixel

Local image data

Modified image data

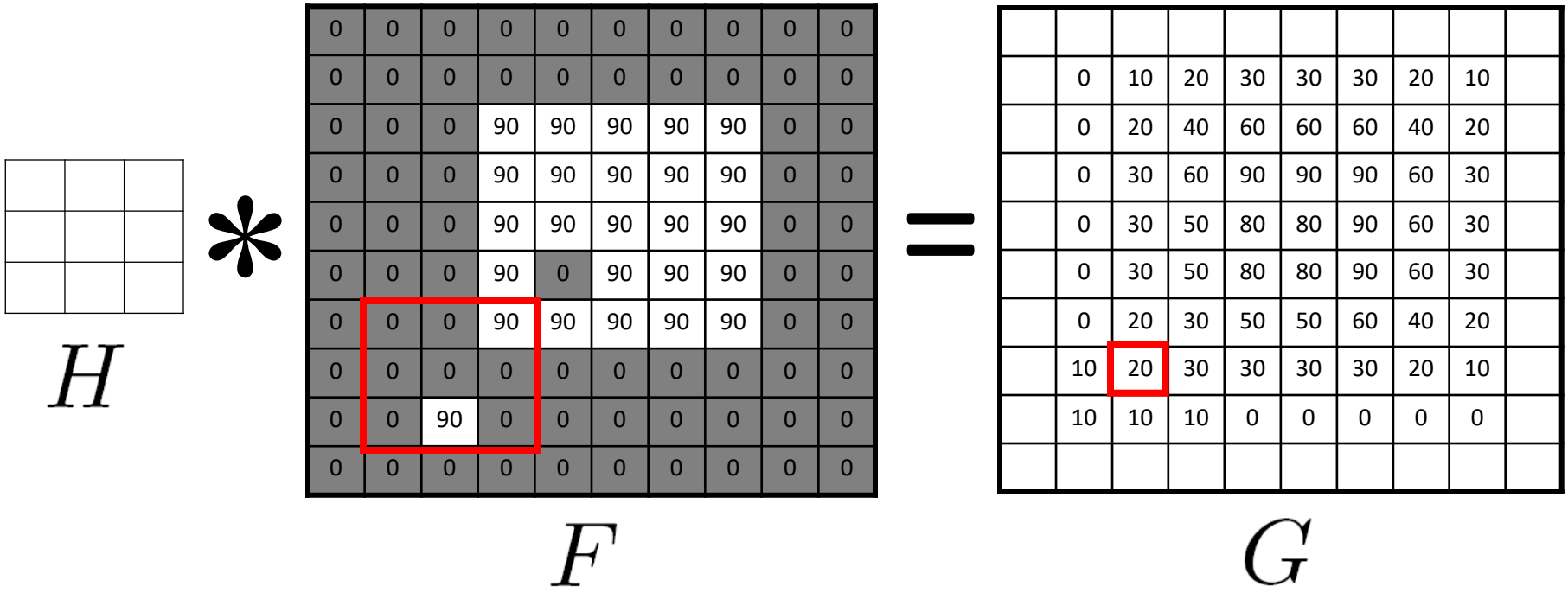
Source: L. Zhang

Convolution

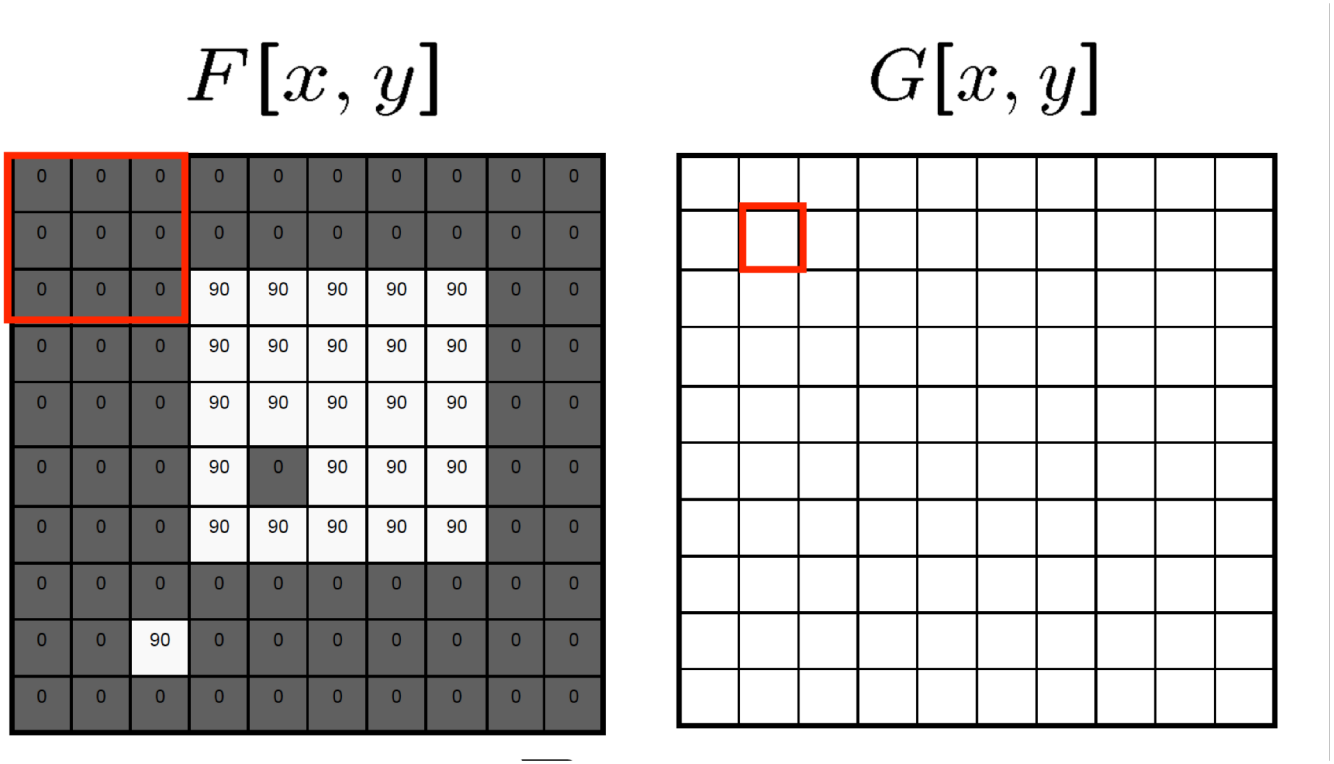


Adapted from F. Durand

Mean filtering

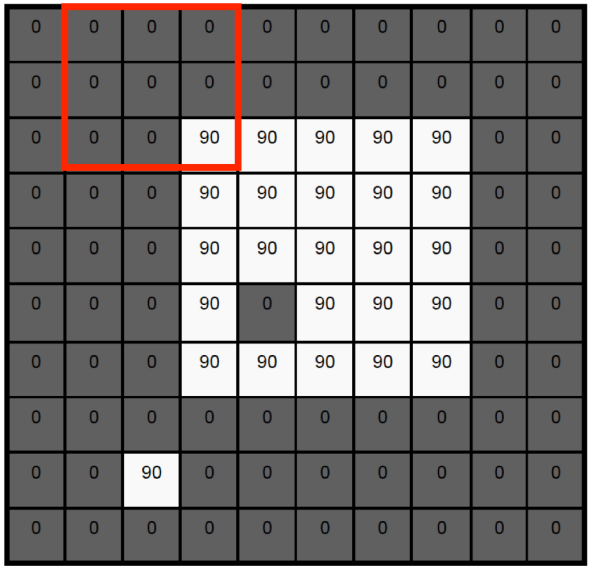


Mean filtering/Moving average

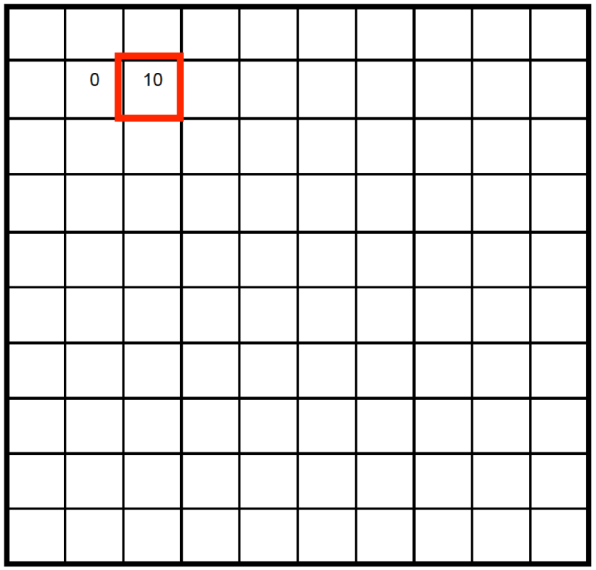


Mean filtering/Moving average

$F[x, y]$

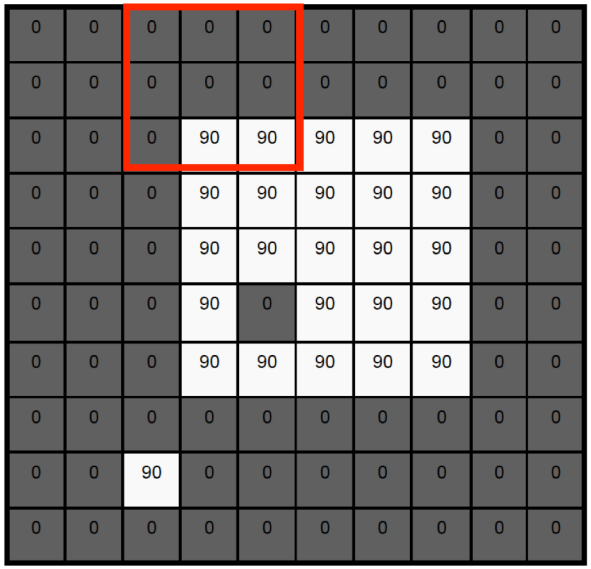


$G[x, y]$

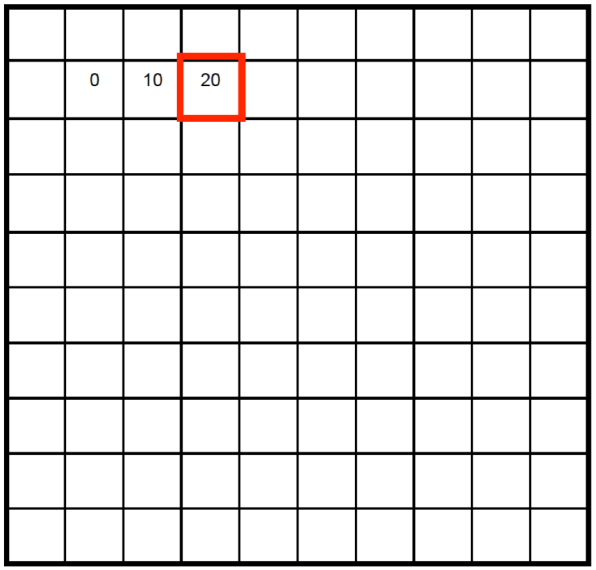


Mean filtering/Moving average

$F[x, y]$

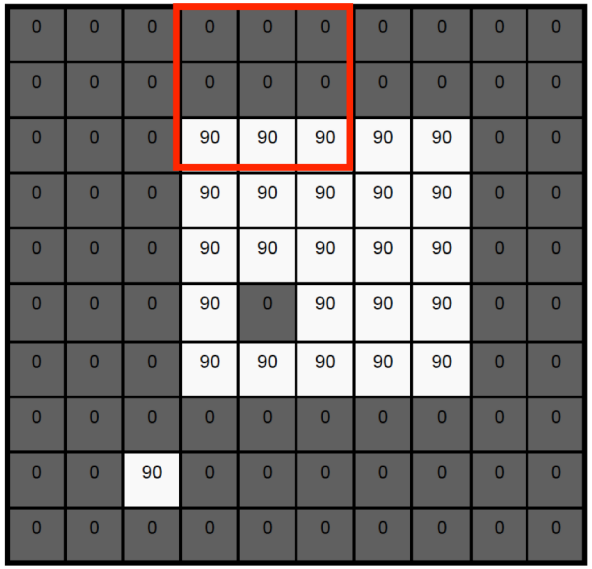


$G[x, y]$

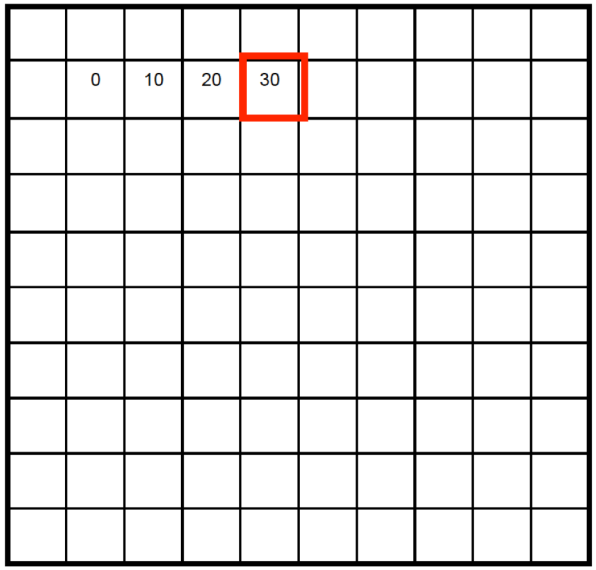


Mean filtering/Moving average

$F[x, y]$

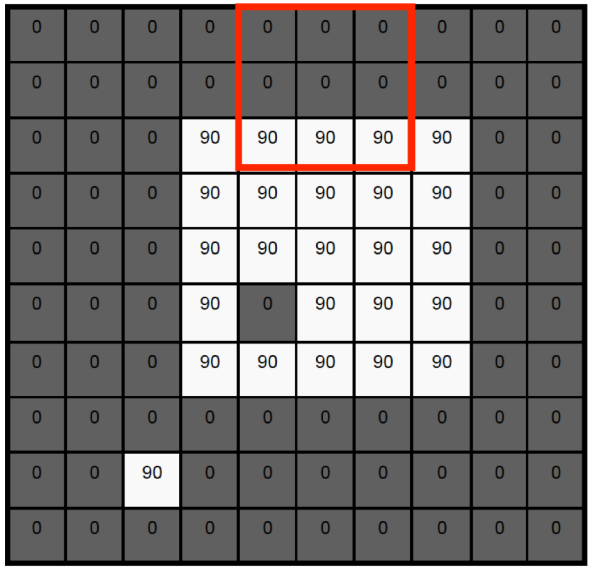


$G[x, y]$

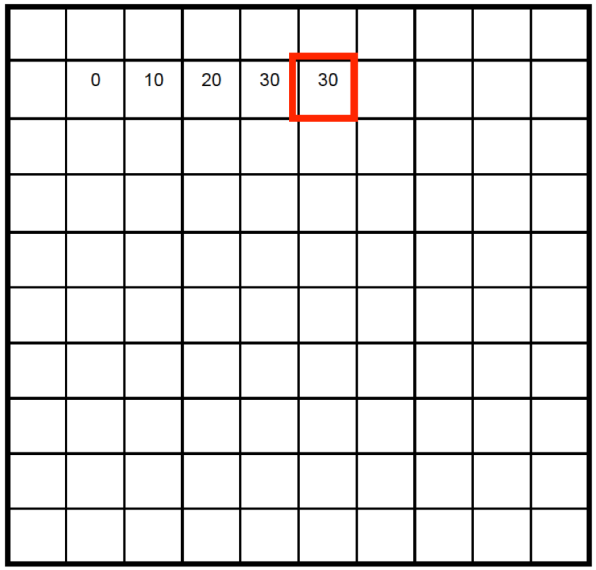


Mean filtering/Moving average

$F[x, y]$



$G[x, y]$



Mean filtering/Moving average

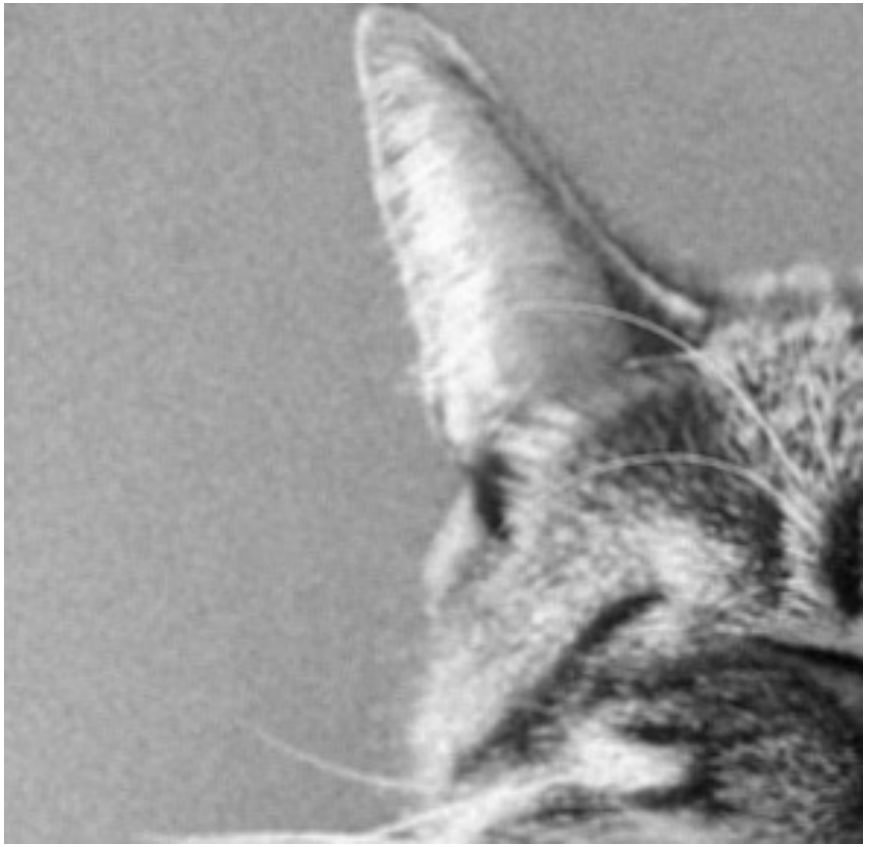
$F[x, y]$

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	

Noise reduction using mean filtering



Mean filtering

- Replace pixel by mean of neighborhood

10	5	3
4	5	1
1	1	7

Local image data

f

	7	

Modified image data

$S[f]$

$$S[f](m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(m+i, n+j)/9$$

A more general version

10	5	3
4	5	1
1	1	7

Local image data

	7	

Kernel / filter

$$S[f](m,n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} w(i,j) f(m+i, n+j)$$

A more general version

0	10	5	7	0
5	11	6	8	3
9	22	4	5	1
2	9	14	6	7
3	10	15	12	9

Local image data

		7		

Kernel size = 2k+1

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

Convolution and cross-correlation

- Cross correlation

$$S[f] = w \otimes f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m+i, n+j)$$

- Convolution

$$S[f] = w * f$$

$$S[f](m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i,j) f(m-i, n-j)$$

Convolution and Cross-Correlation: Properties

- Both are linear:

$$w' = aw + bv$$

$$w' \otimes f = a(w \otimes f) + b(v \otimes f)$$

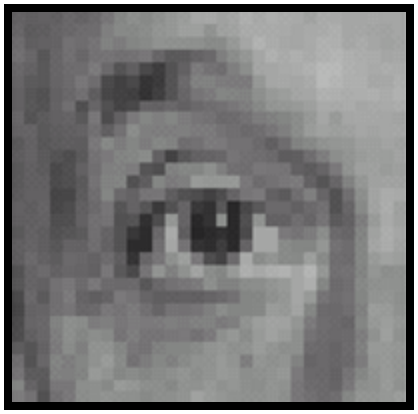
- Convolution is **commutative**:

$$w * f = f * w$$

- Convolution is **associative:**

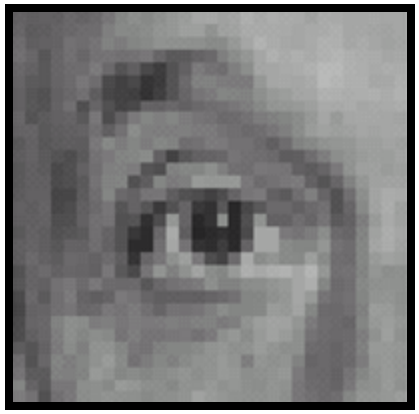
$$h * (w * f) = (h * w) * f$$

Linear filters: examples



Original

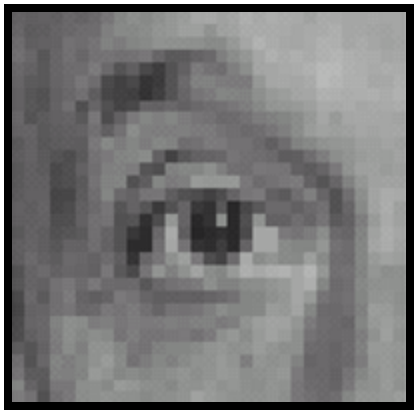
$$*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$



Identical image

Source: D. Lowe

Linear filters: examples

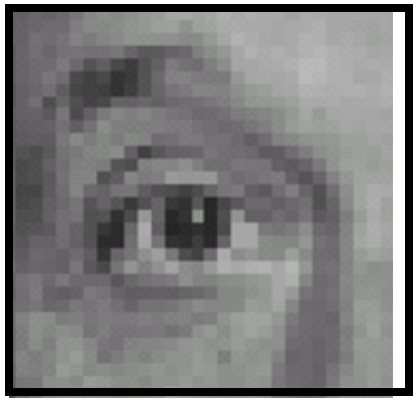


Original

$*$

0	0	0
1	0	0
0	0	0

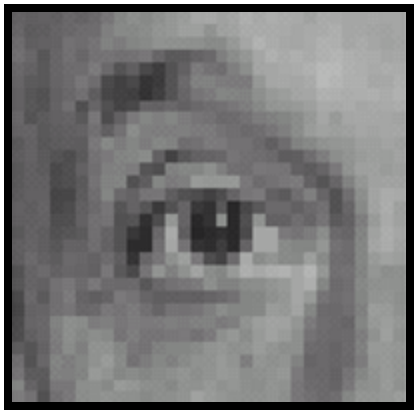
$=$



Shifted left
By 1 pixel

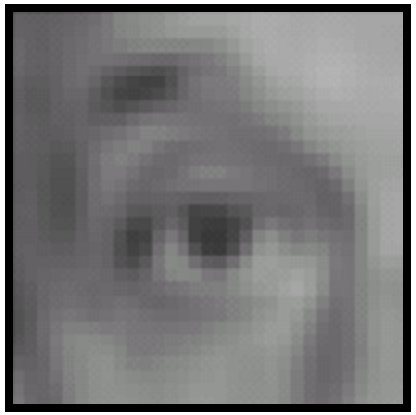
Source: D. Lowe

Linear filters: examples



Original

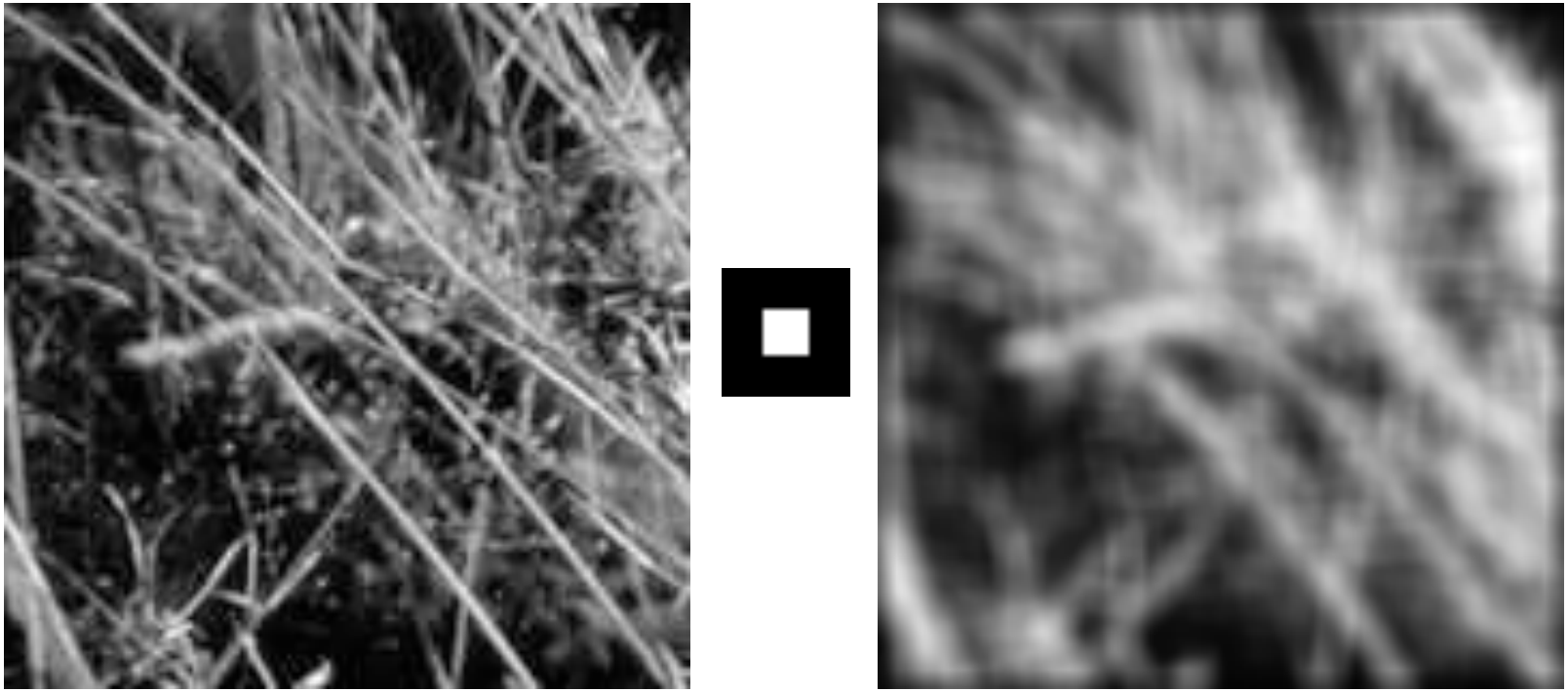
$$*\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$



Blur (with a mean filter)

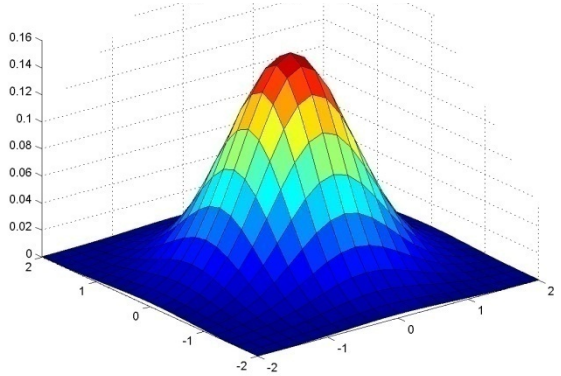
Source: D. Lowe

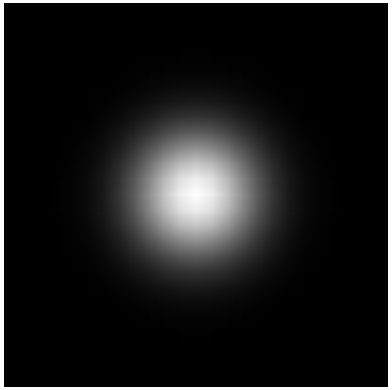
Smoothing with box filter revisited



Source: D. Forsyth

Gaussian Kernel

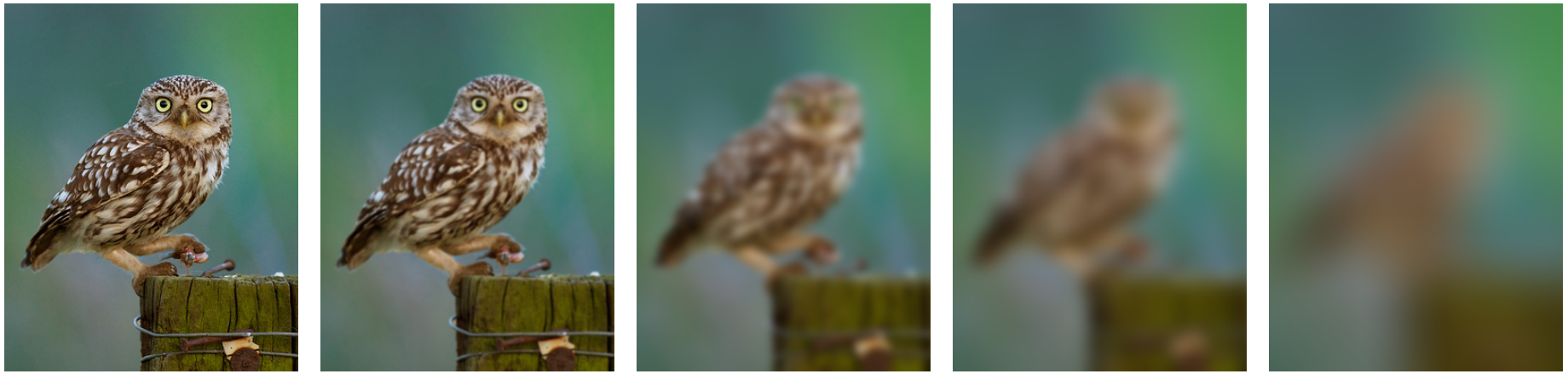




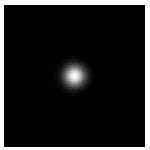
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

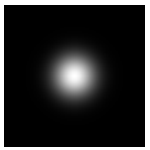
Gaussian filters



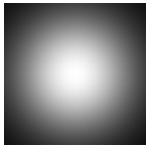
σ = 1 pixel



σ = 5 pixels

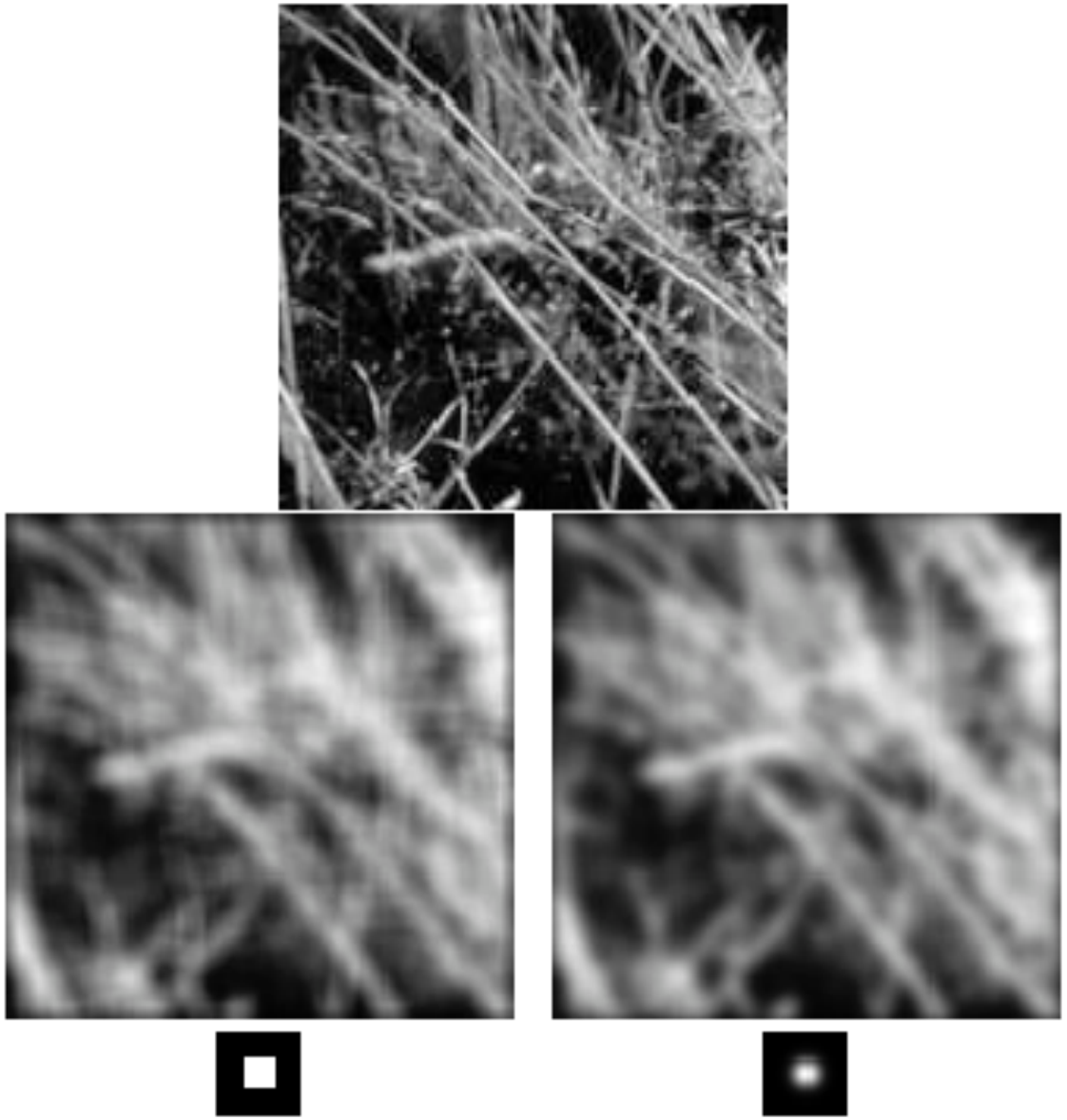


σ = 10 pixels



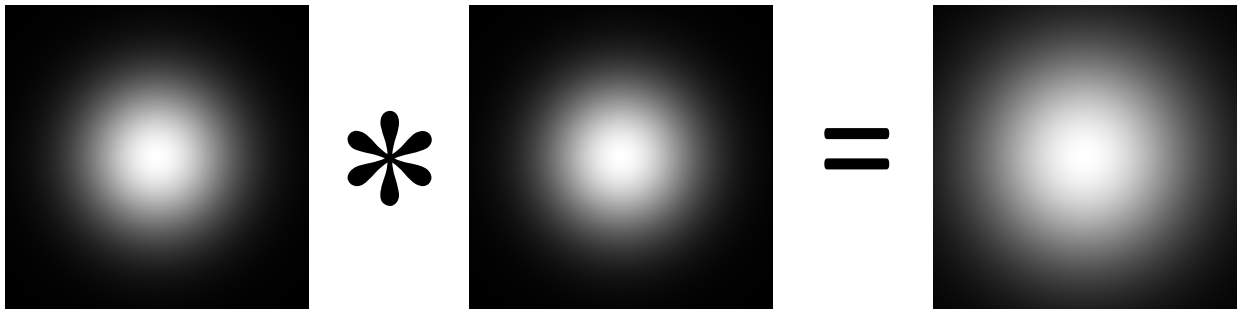
σ = 30 pixels

Mean vs. Gaussian filtering



Gaussian filter

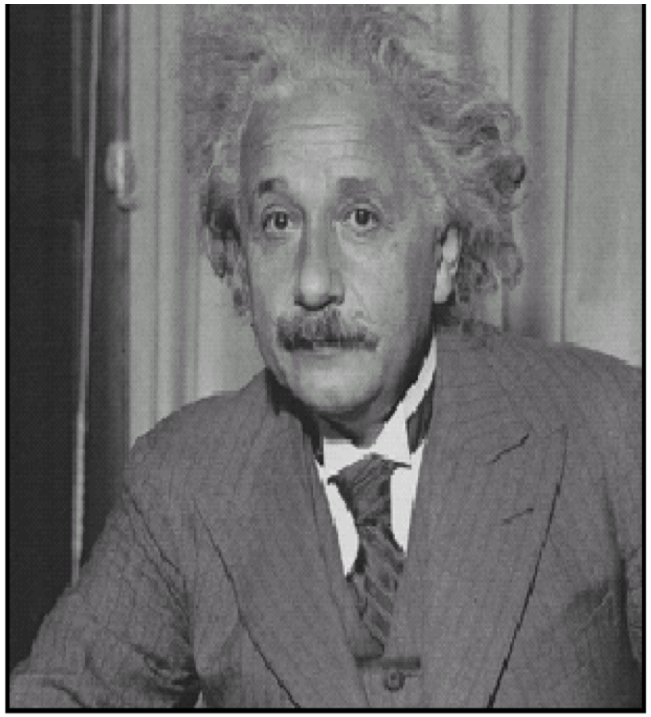
- Removes “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian



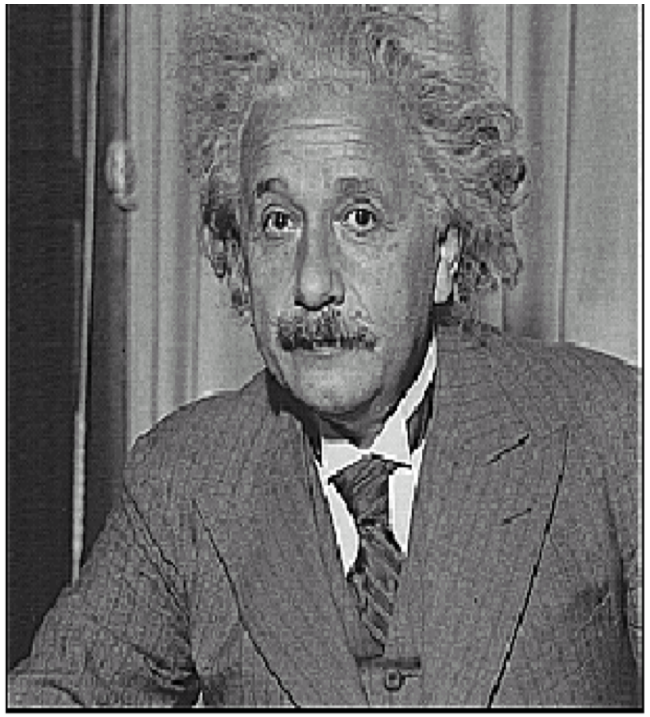
  - Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

Sharpening



before

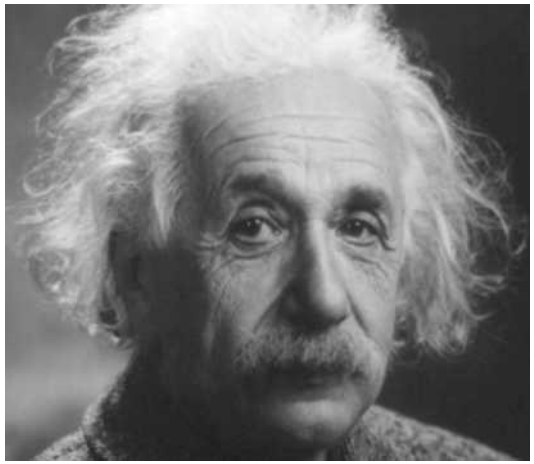
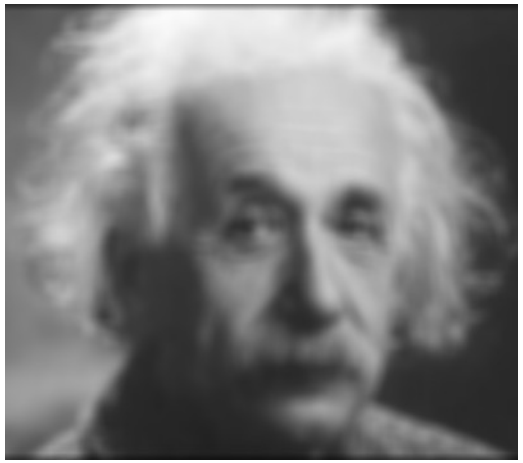
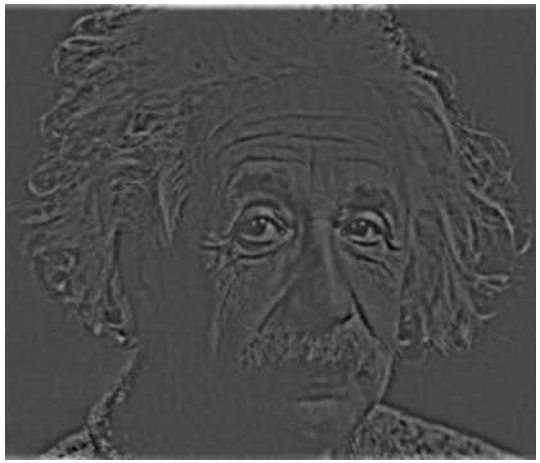


after

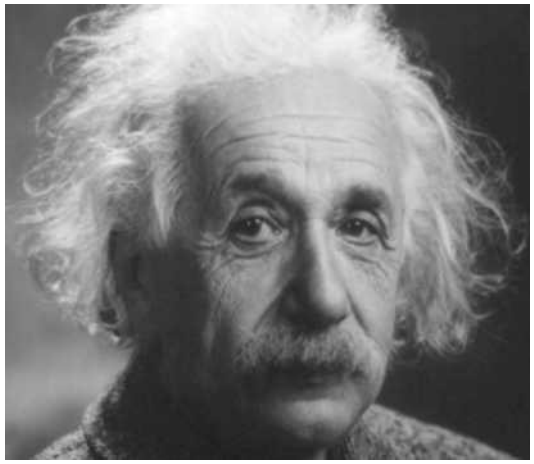
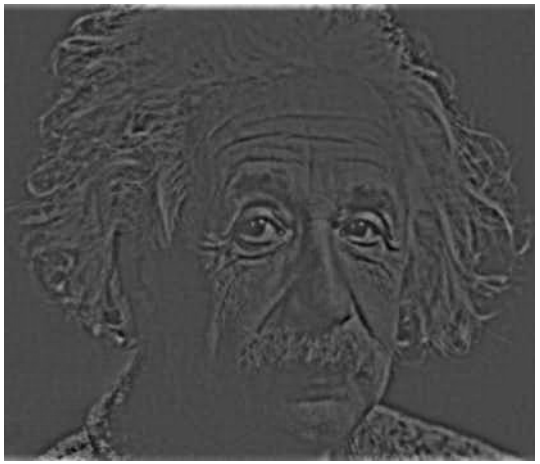
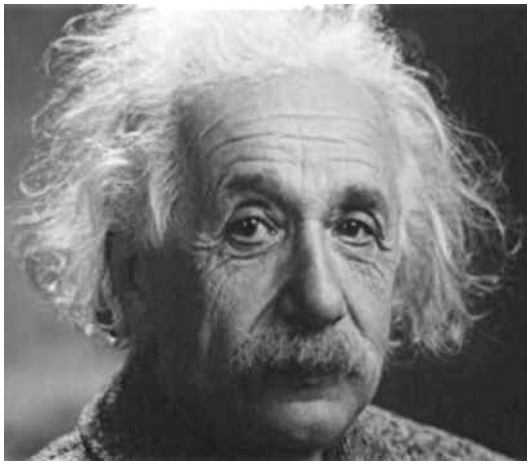
Source: D. Lowe

Sharpening

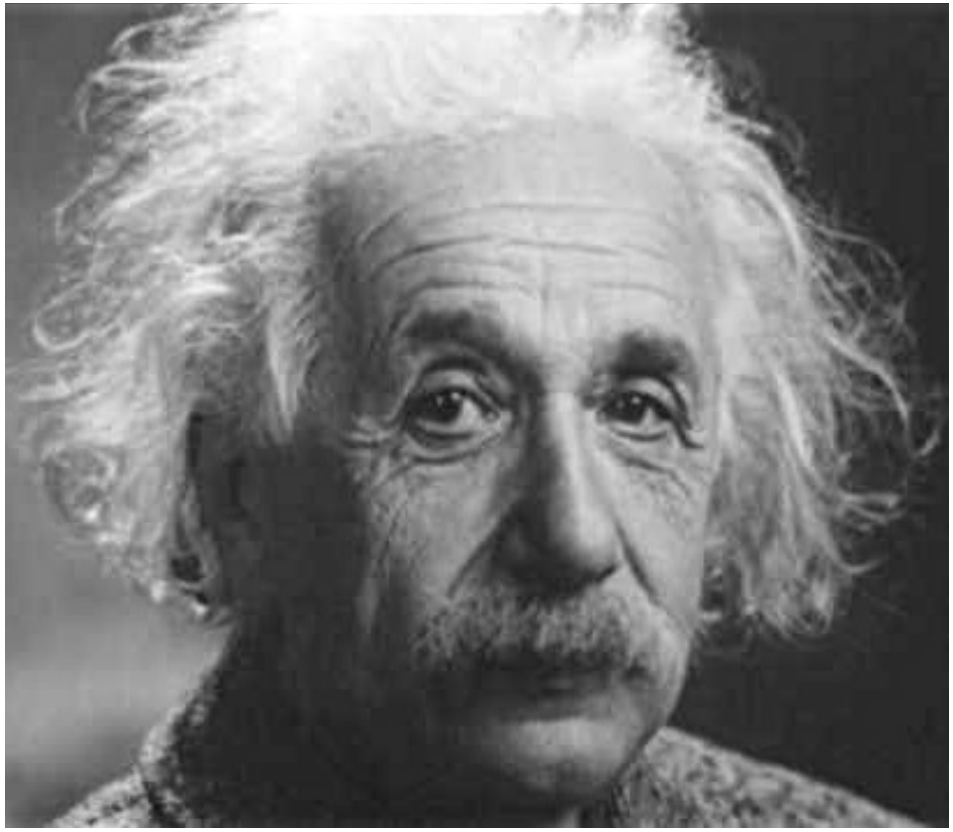
- What does blurring take away?

Let's add it back:

Sharpening

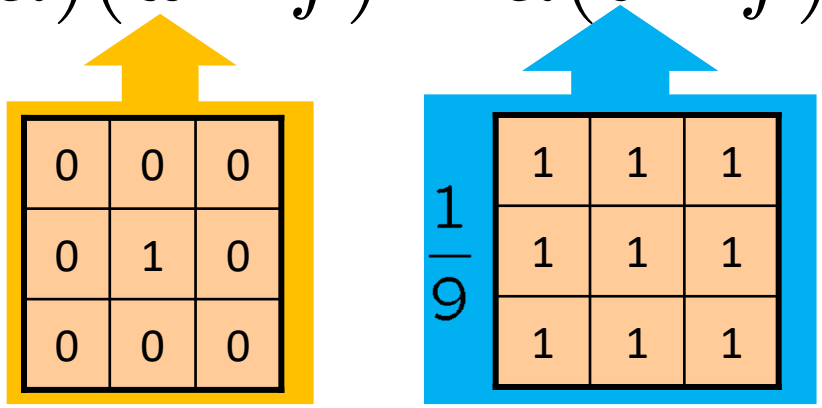


Sharpening

$$f_{sharp} = f + \alpha(f - f_{blur})$$

$$= (1+\alpha)f - \alpha f_{blur}$$

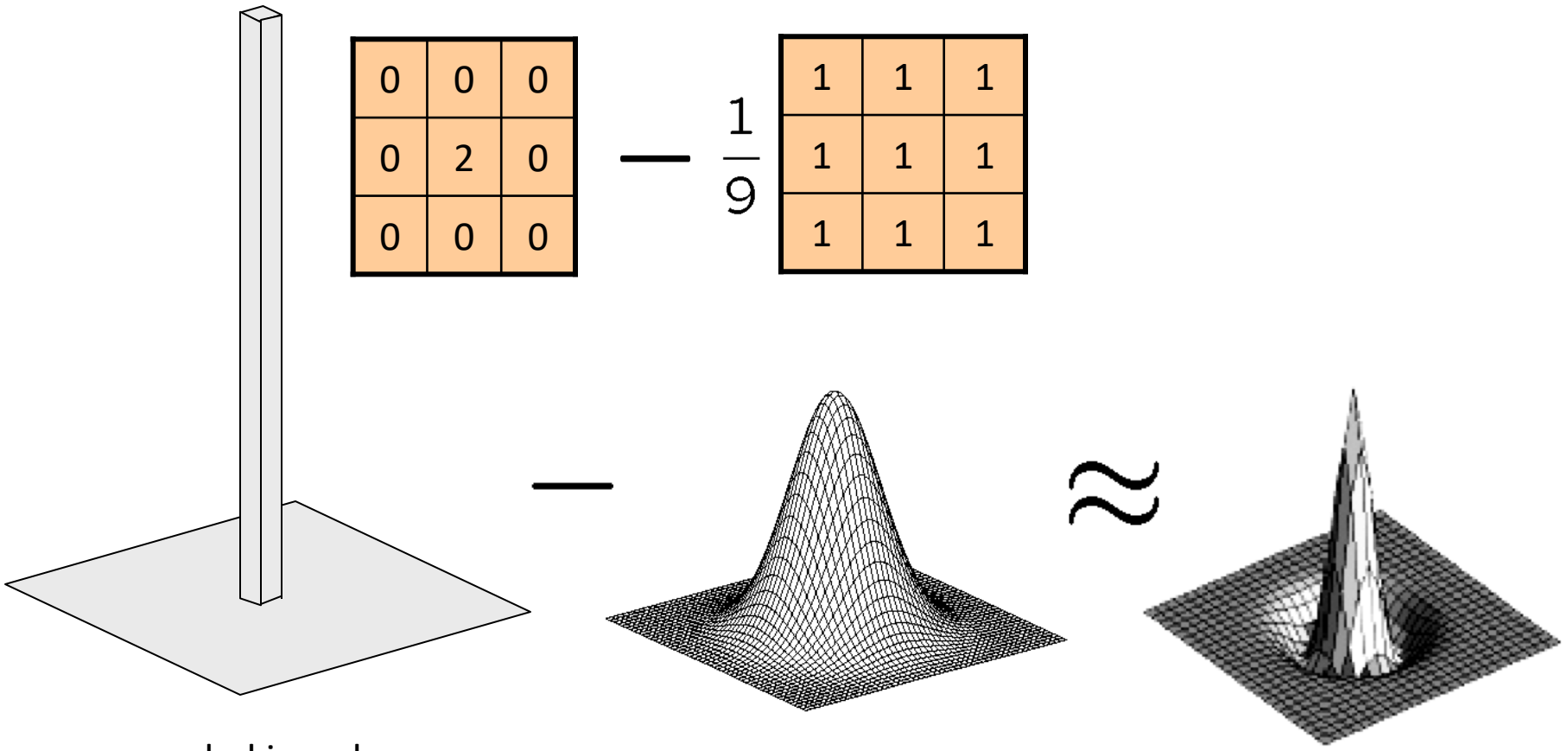
$$= (1+\alpha)(w * f) - \alpha(v * f)$$



$$= ((1+\alpha)w - \alpha v) * f$$

Sharpening filter

$$((1+\alpha)w - \alpha v)$$



Sharpen filter

“Optical” Convolution

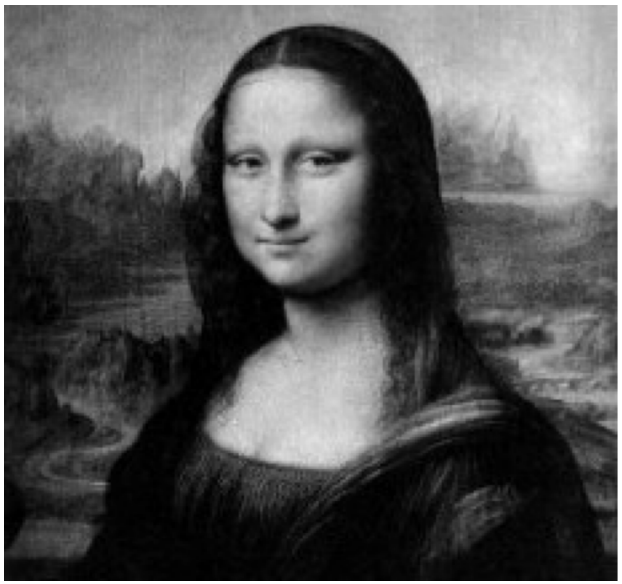
Camera shake

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

Filters: Thresholding



$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

Linear filters

- Is thresholding a linear filter?

- How about a maximum, minimum, or median filter?

Questions?