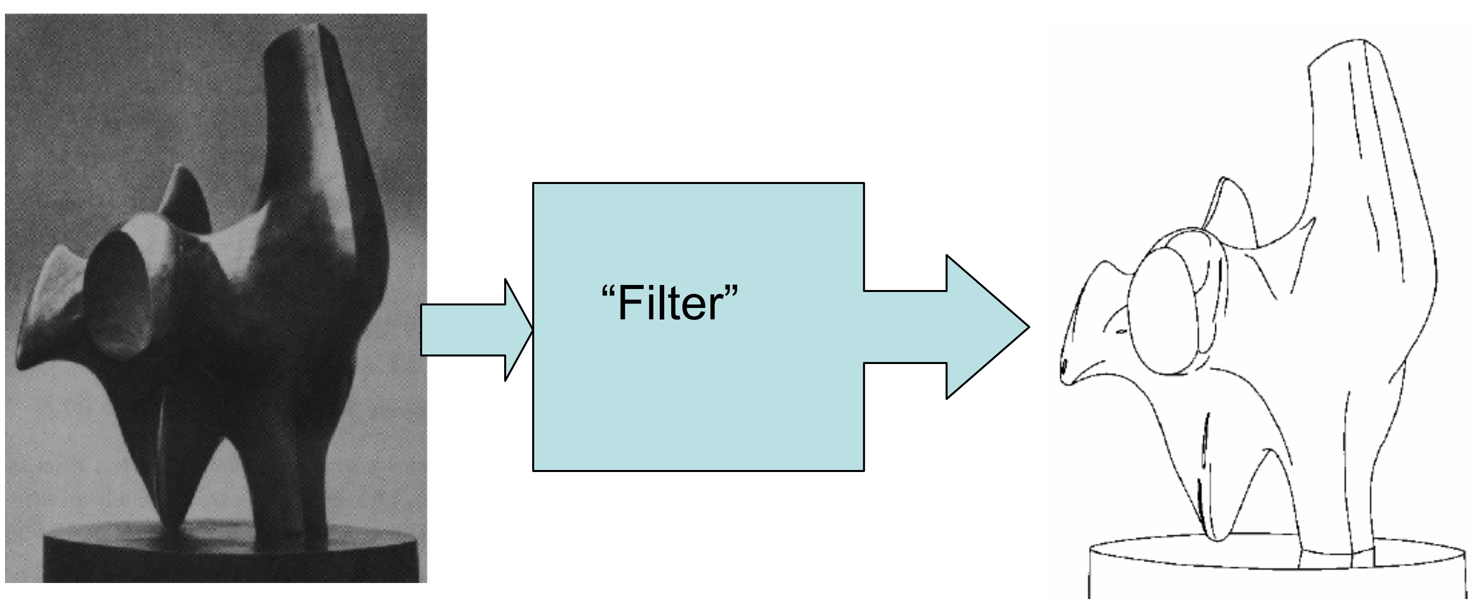
Lecture 5: Feature Extraction
Linear Filters

Features

- Reduce the size of the data on which we want to build our inferences
- Reduction should retain most informative pieces
  - Edges and Corners
  - Texture
- Mathematically/computationally



Mathematical view

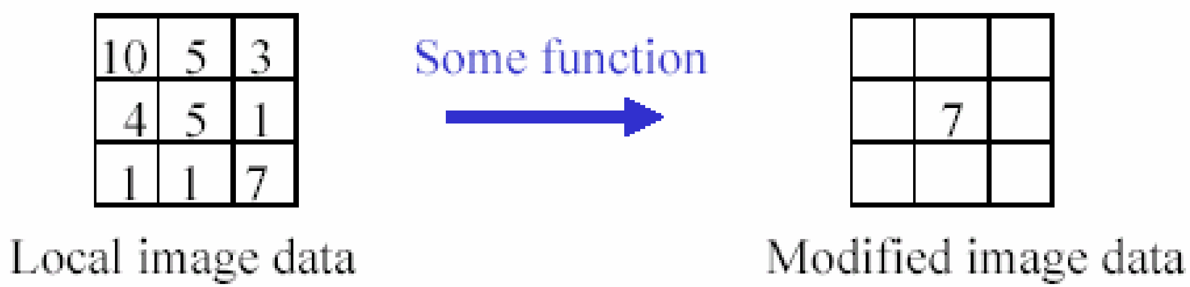
- Represent image in some way
  - Array of numbers in last class
- Operate on this representation to find values that are significant
- Filter out significant values
- Linear edge detectors: (discussed in last class)
- Today: discuss the idea of linear filters generally
- Consider: Apply a filtering function locally

Linear Filtering

- About modifying pixels based on neighborhood.
- Linear means linear combination of neighbors.
  - Linear methods simplest
  - Can combine linear methods in any order to achieve same result
  - Maybe easier to undo
  - “Filter Banks”
- Useful to:
  - Integrate information over constant regions.
  - Scale.
  - Detect changes (edge detection)
- Fourier analysis. (next class)
  - General linear filtering
  - General image representation
  - Image Coding

What is image filtering?

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



(Freeman)

Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	7	

Modified image data

(Freeman)

Convolution

- Convolution kernel *g*, represented as matrix.
  - it's associative

- Result is:

$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l] g[k,l]$$

Edge Filters

- Sobel

$\frac{1}{8}$

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

$\frac{1}{8}$

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

- Prewitt

$\frac{1}{6}$

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

$\frac{1}{6}$

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

- Roberts

1	0
0	-1

0	1
-1	0

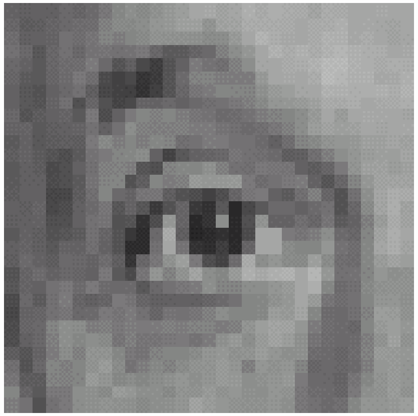
- Laplacian

0	1	0
1	-4	1
0	1	0

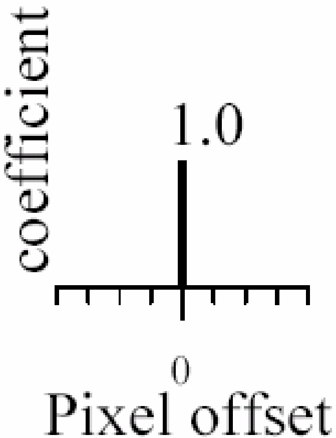
$\frac{1}{3}$

1	1	1
1	-8	1
1	1	1

Linear filtering (warm-up slide)

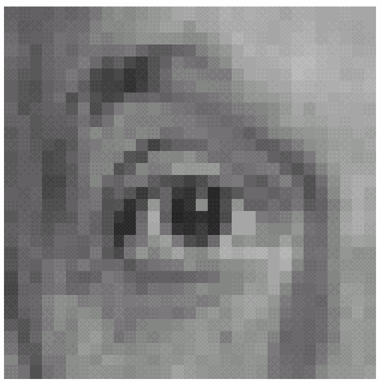


original

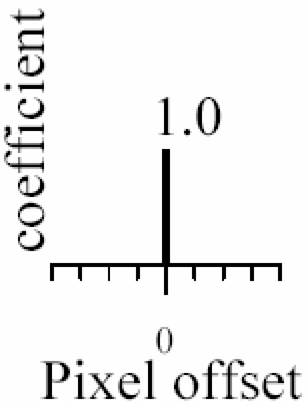


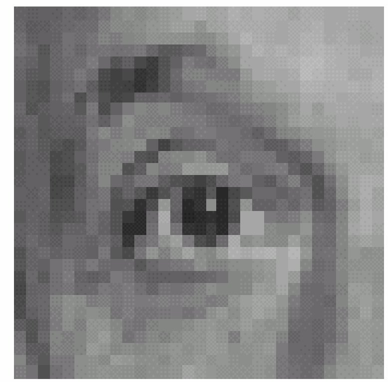
?

Linear filtering (warm-up slide)



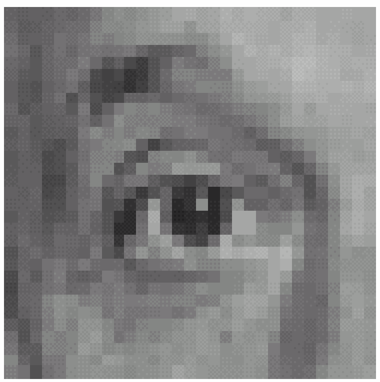
original



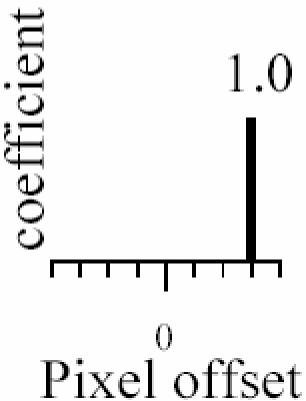


Filtered
(no change)

Linear filtering

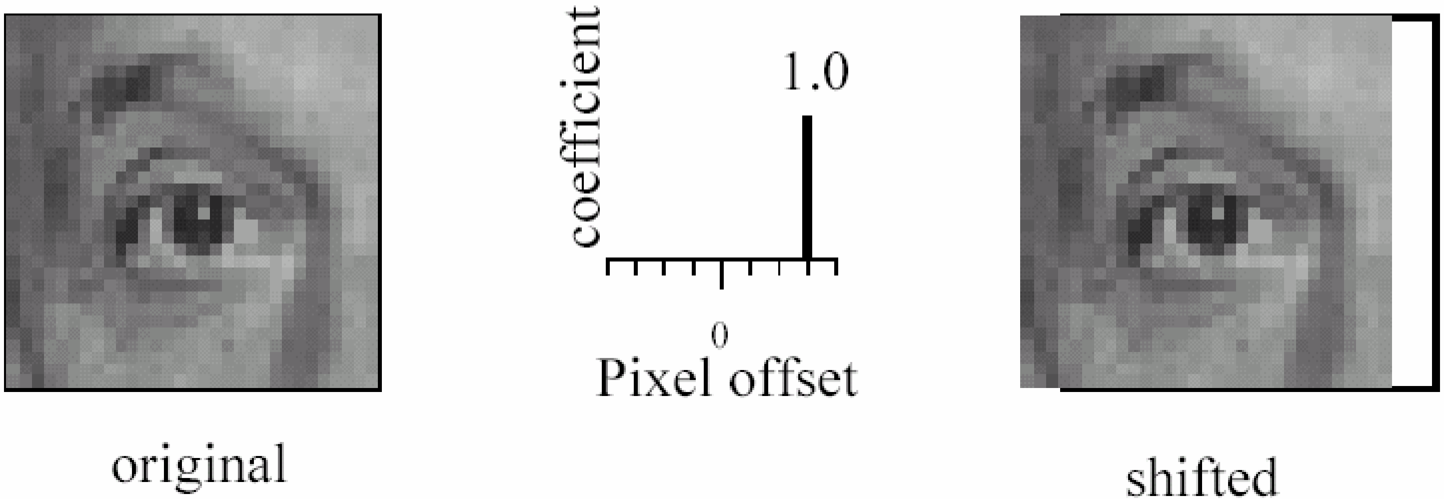


original

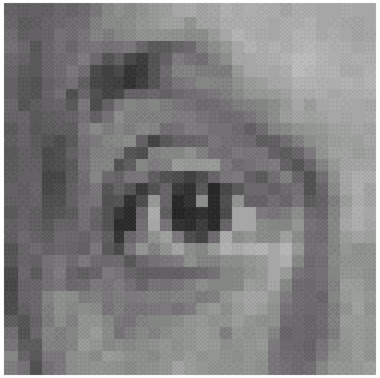


?

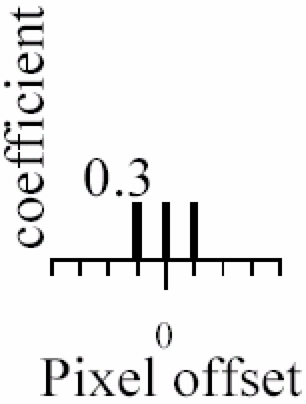
shift



Linear filtering

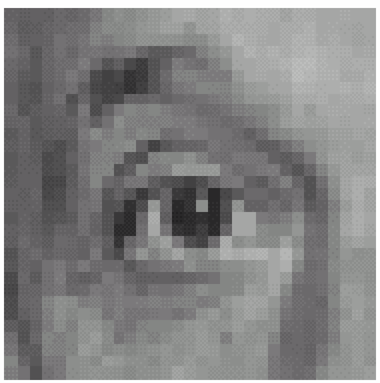


original

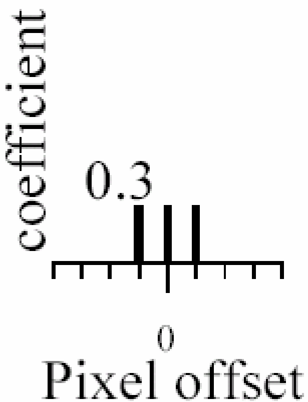


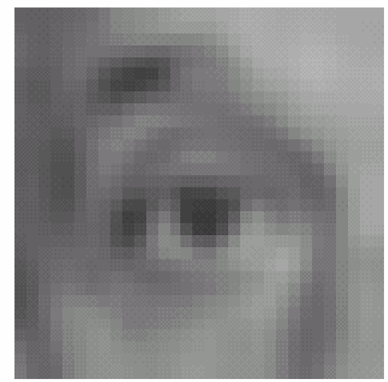
?

Blurring



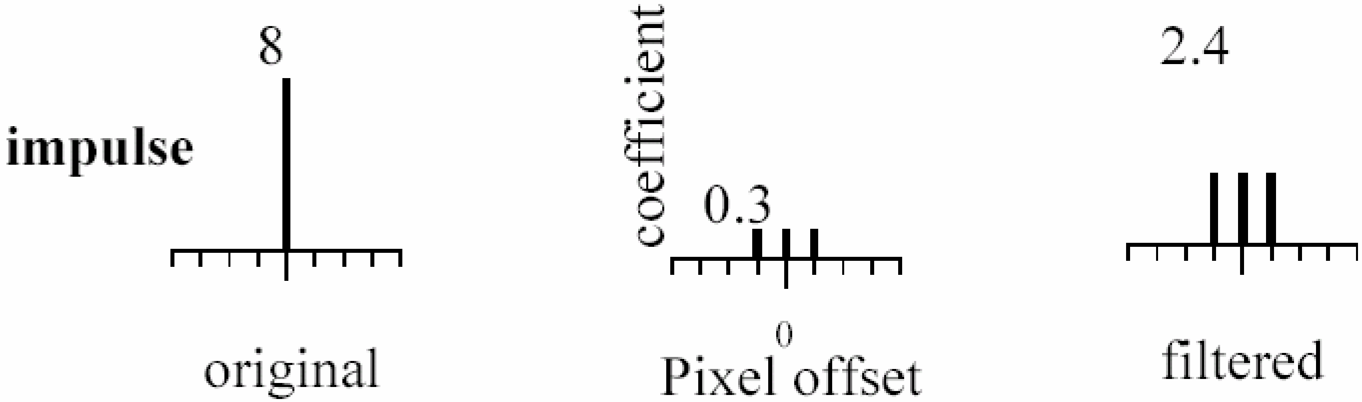
original



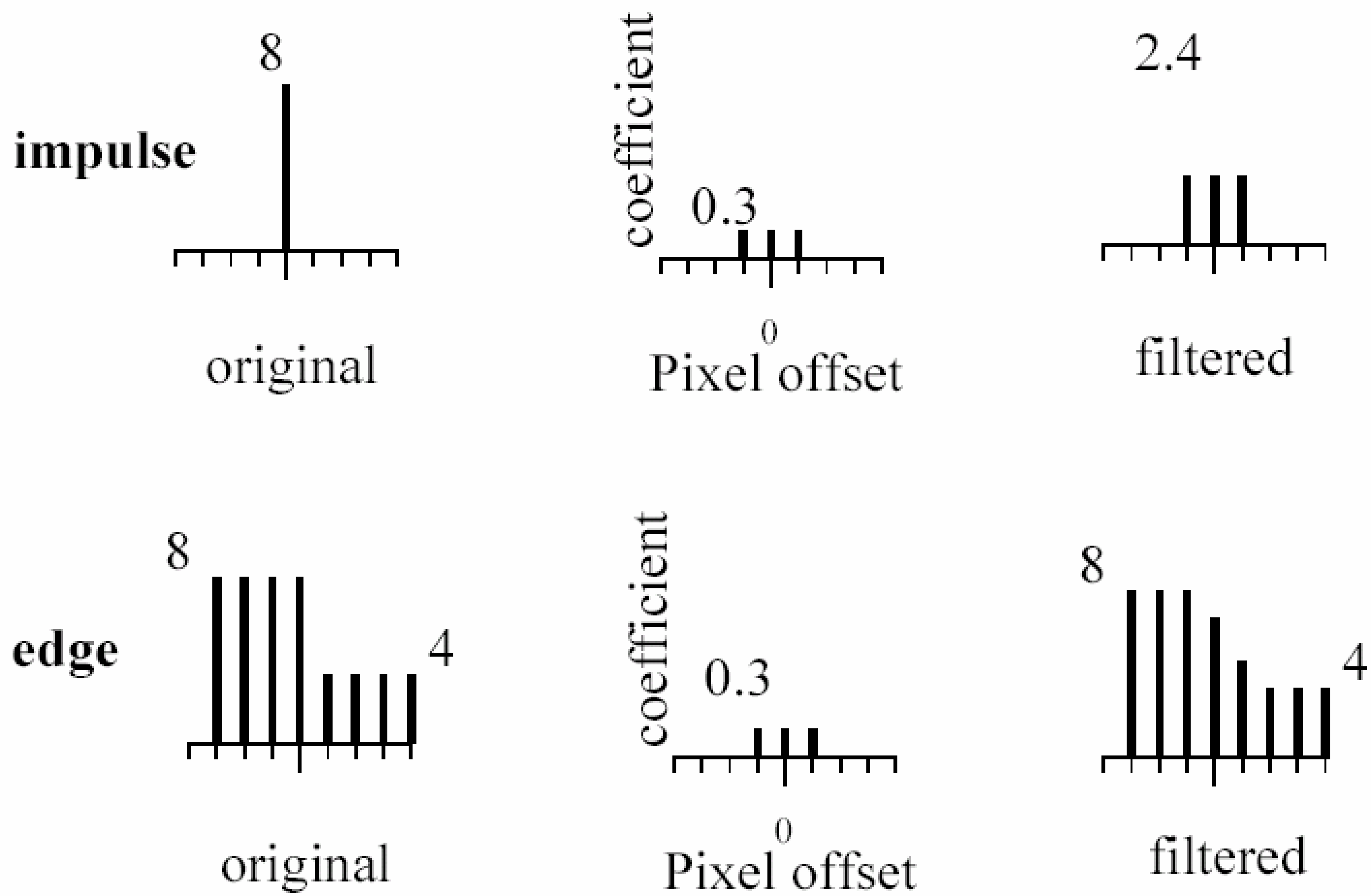


Blurred (filter applied in both dimensions).

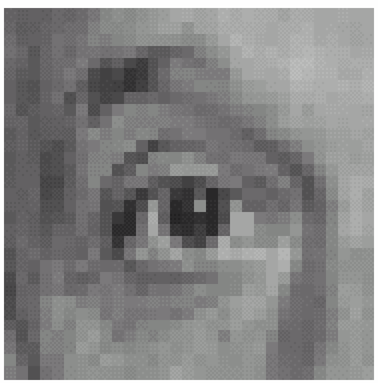
Blur examples



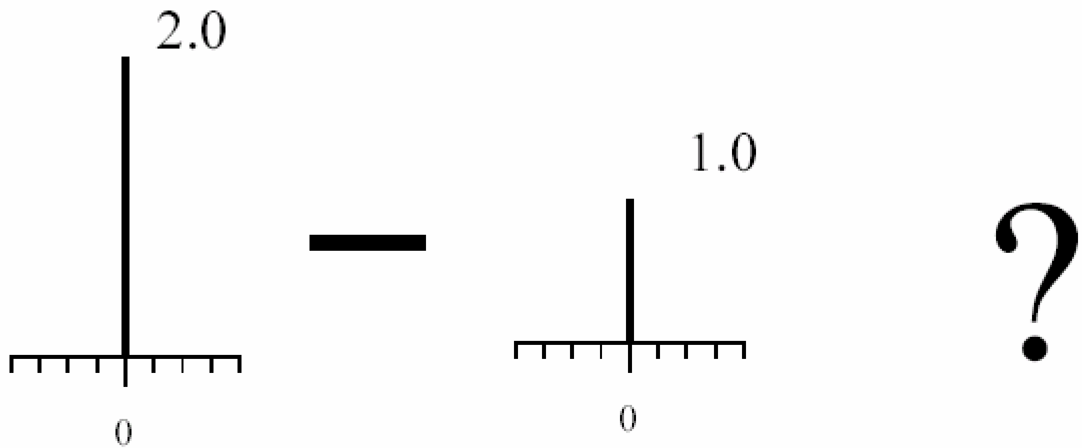
Blur examples



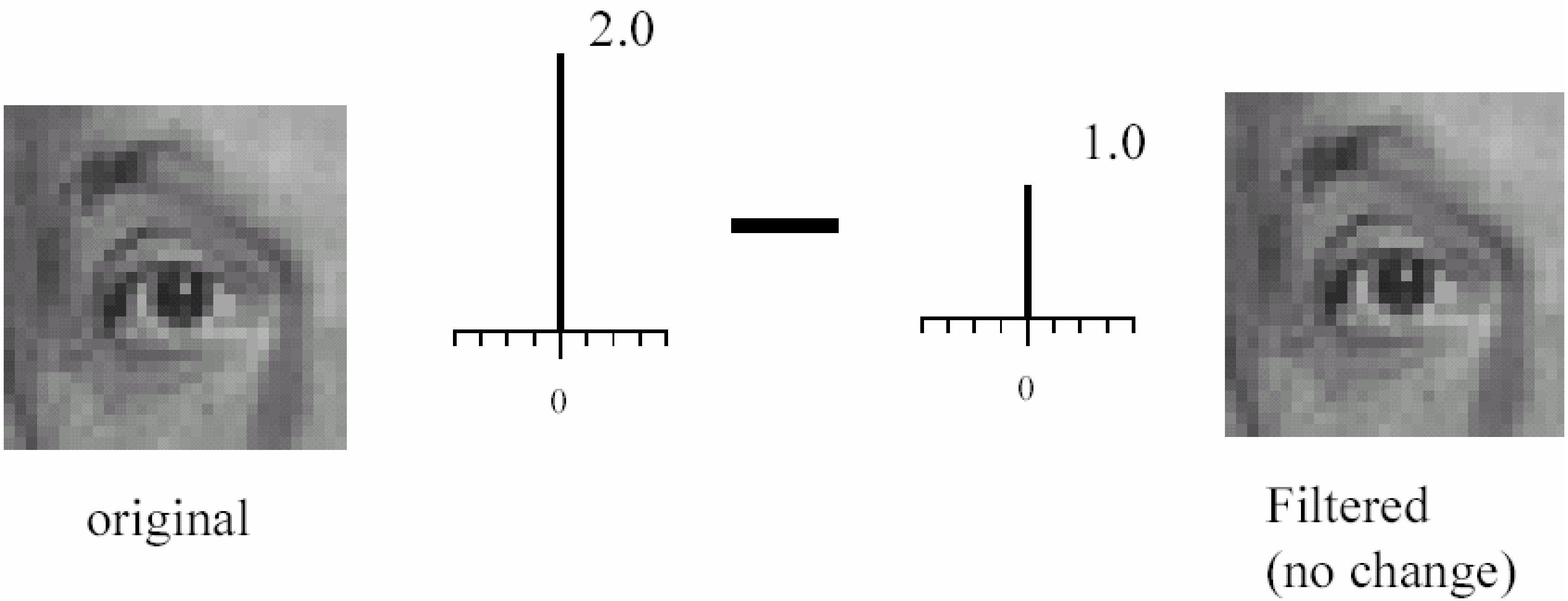
Linear filtering (warm-up slide)



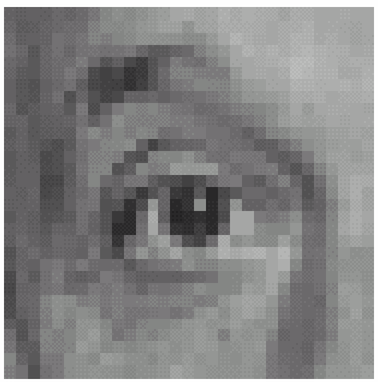
original



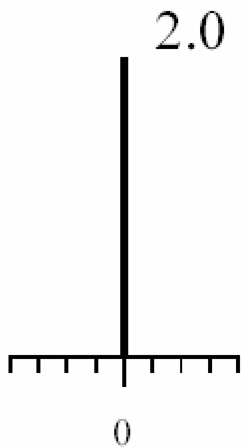
Linear filtering (no change)



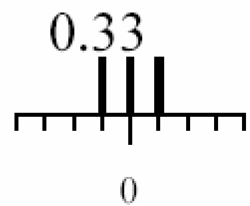
Linear filtering



original

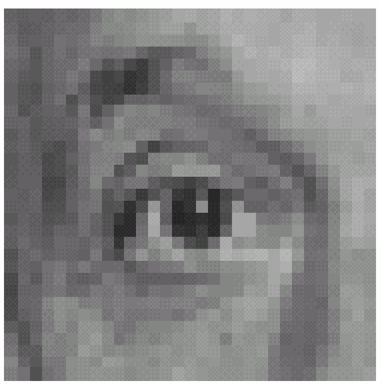


−

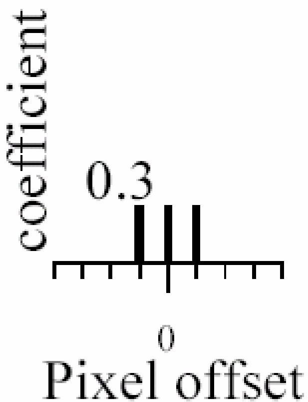


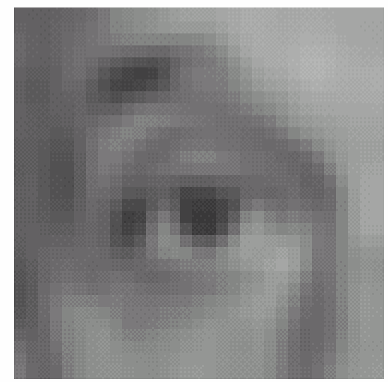
?

(remember blurring)



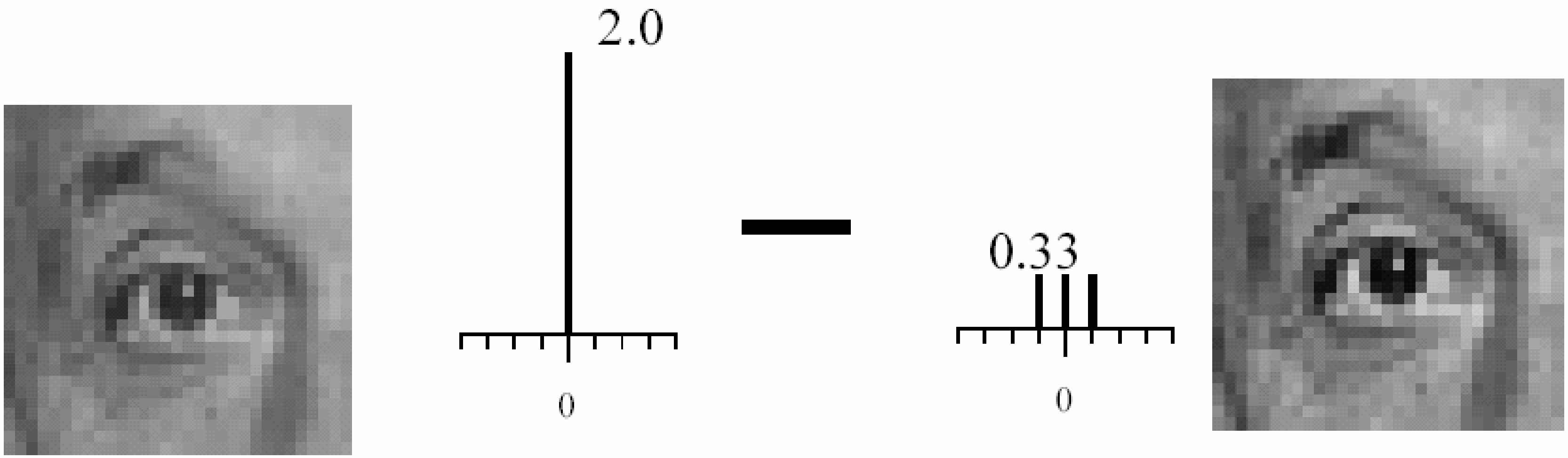
original





Blurred (filter applied in both dimensions).

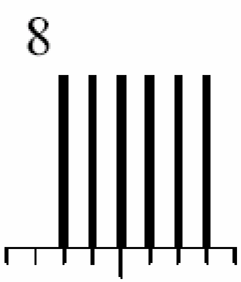
Sharpening



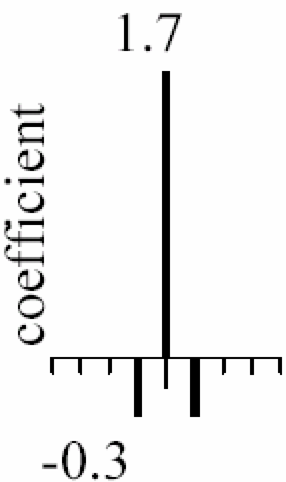
original

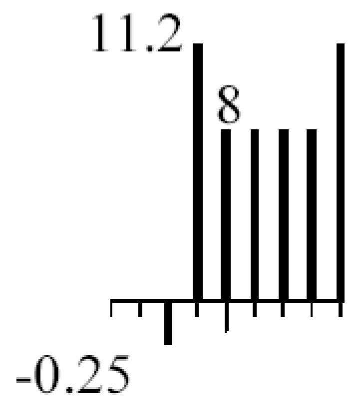
Sharpened original

Sharpening example



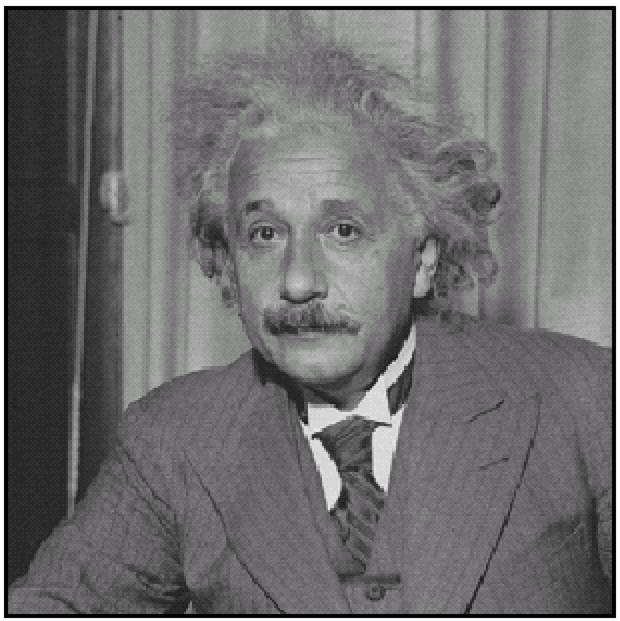
original



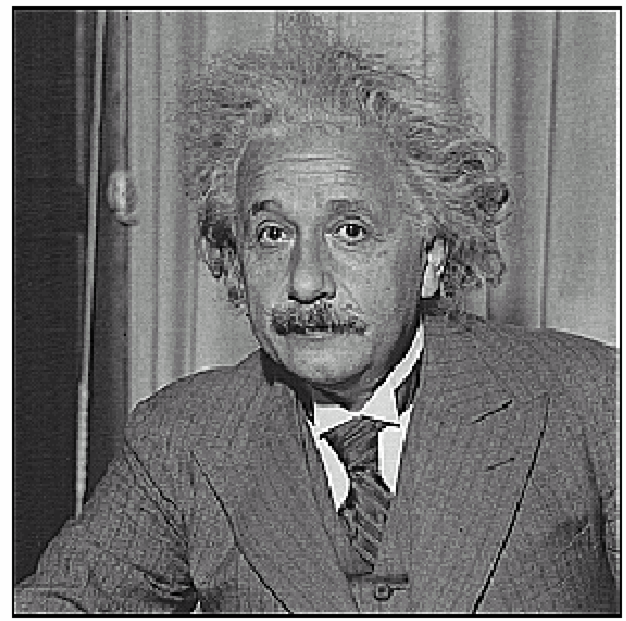


Sharpened
(differences are accentuated; constant areas are left untouched).

Sharpening



before



after

Filtering to reduce noise

- Noise is what we're not interested in.
  - We'll discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

- I = S + N. Noise doesn't depend on signal.
- We'll consider:

$$I_i = s_i + n_i \text{ with } E(n_i) = 0$$

$$s_i \text{ deterministic.}$$

$$n_i, n_j \text{ independent for } n_i \neq n_j$$

$$n_i, n_j \text{ identically distributed}$$

Average Filter

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

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

(Camps)

Does it reduce noise?

- Intuitively, takes out small variations.

$$I(i,j) = \hat{I}(i,j) + N(i,j) \text{ with } \mathrm{N(i, j)} \sim \mathrm{N}(0, \sigma)$$

$$\mathrm{O(i, j)} = \frac{1}{\mathrm{m}^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + N(i-h, j-k) =$$

$$= \frac{1}{\mathrm{m}^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} \hat{I}(i-h, j-k) + \underbrace{\frac{1}{m^2} \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} N(i-h, j-k)}_{\hat{N}(i,j)}$$

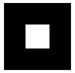
$$E(\hat{N}(i,j)) = 0$$

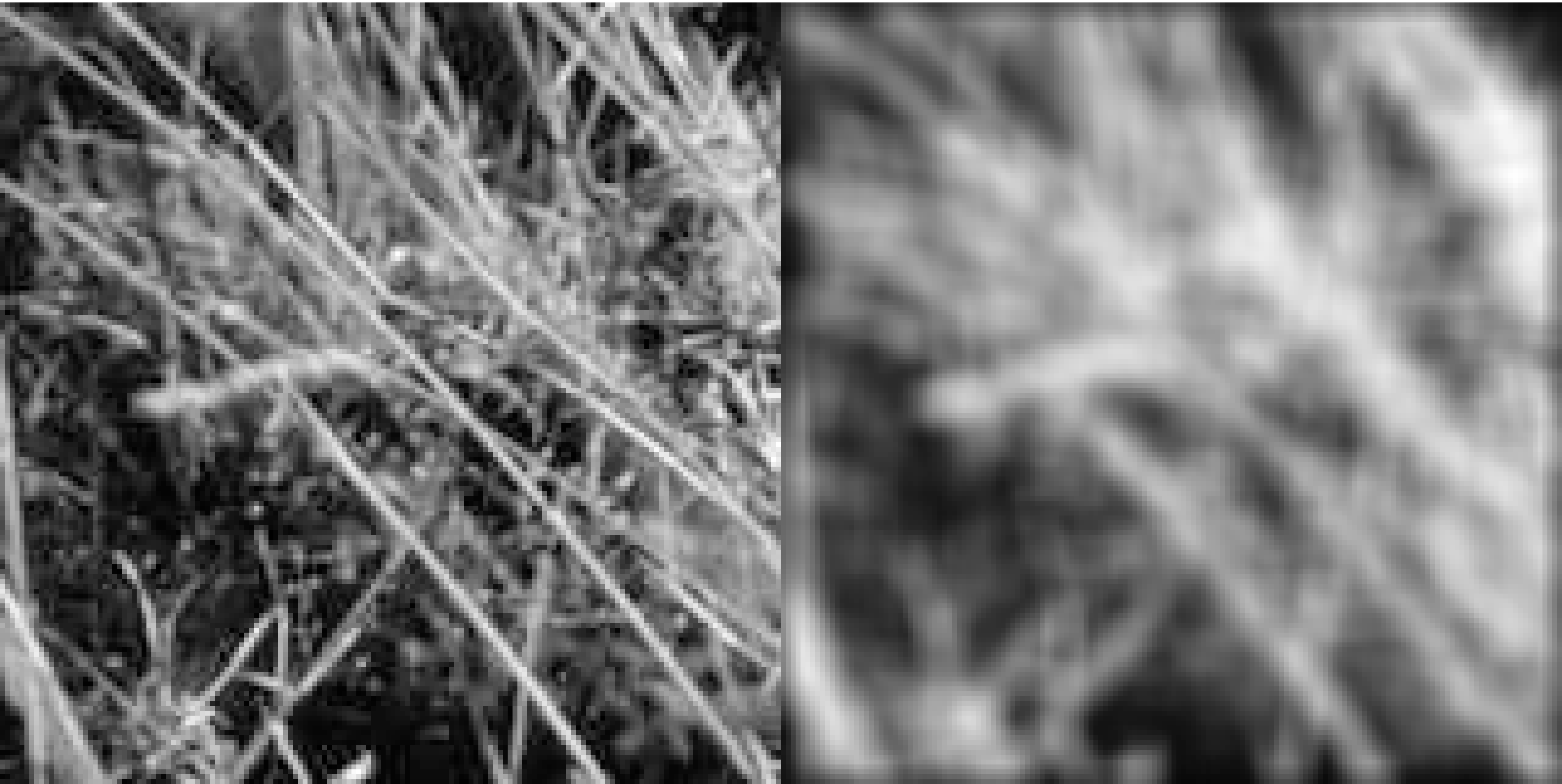
$$E(\hat{N}^2(i,j)) = \frac{1}{m^2} m \sigma^2 = \frac{\sigma^2}{m} \Rightarrow \hat{N}(i,j) \sim N(0, \frac{\sigma}{\sqrt{m}})$$

(Camps)

Matlab Demo of Averaging

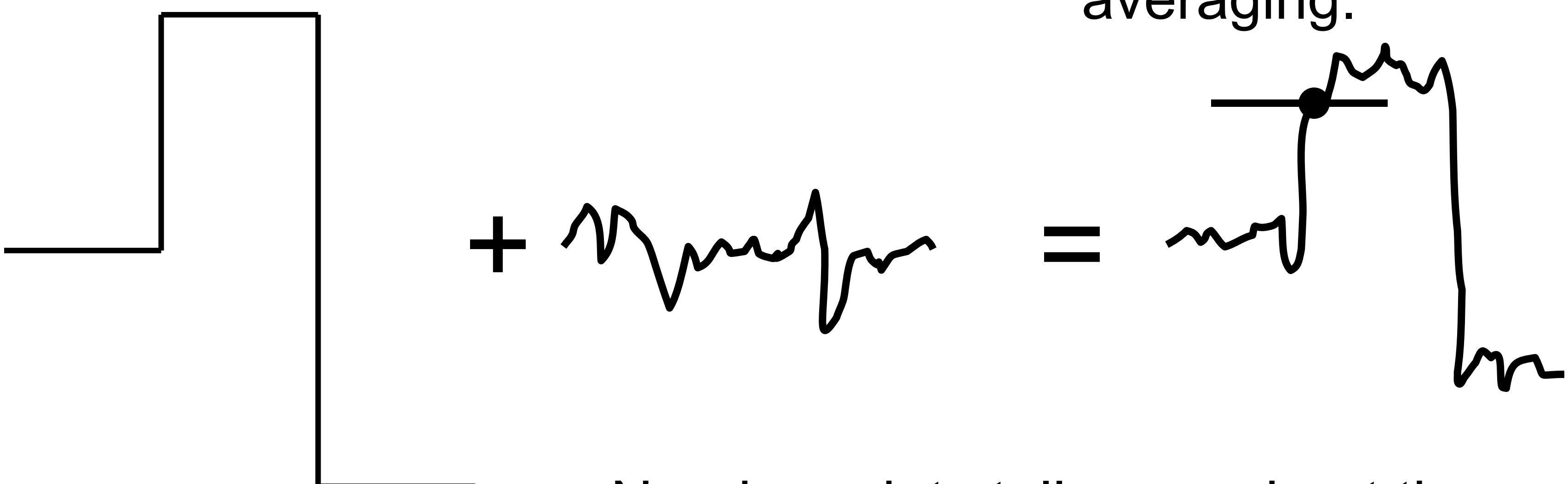
Example: Smoothing by Averaging





Smoothing as Inference About the Signal

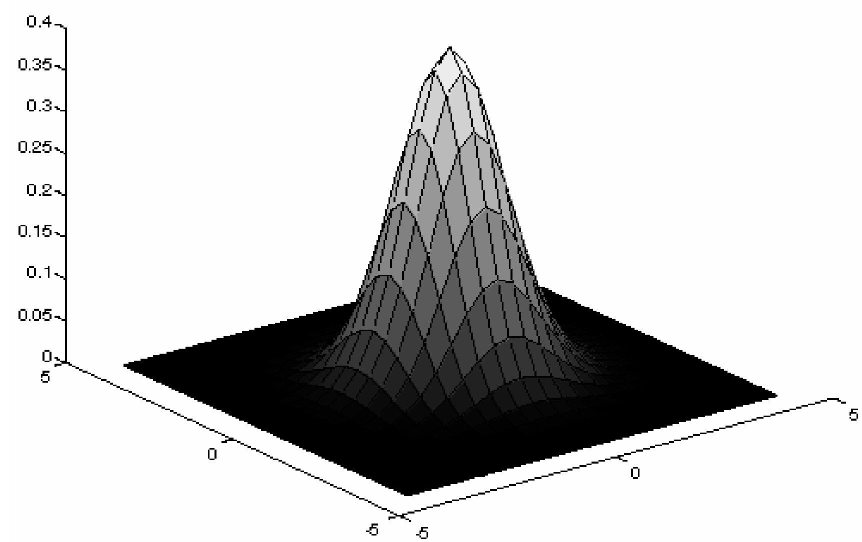
Neighborhood for averaging.



Nearby points tell more about the signal than distant ones.

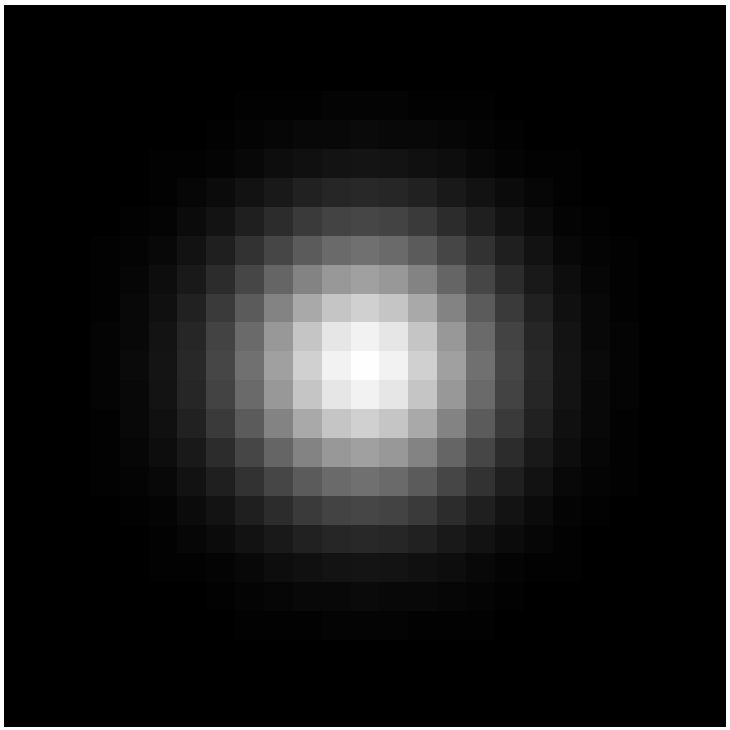
Gaussian Averaging

- Rotationally symmetric.
- Weights nearby pixels more than distant ones.
  - This makes sense as probabalistic inference.



- A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian

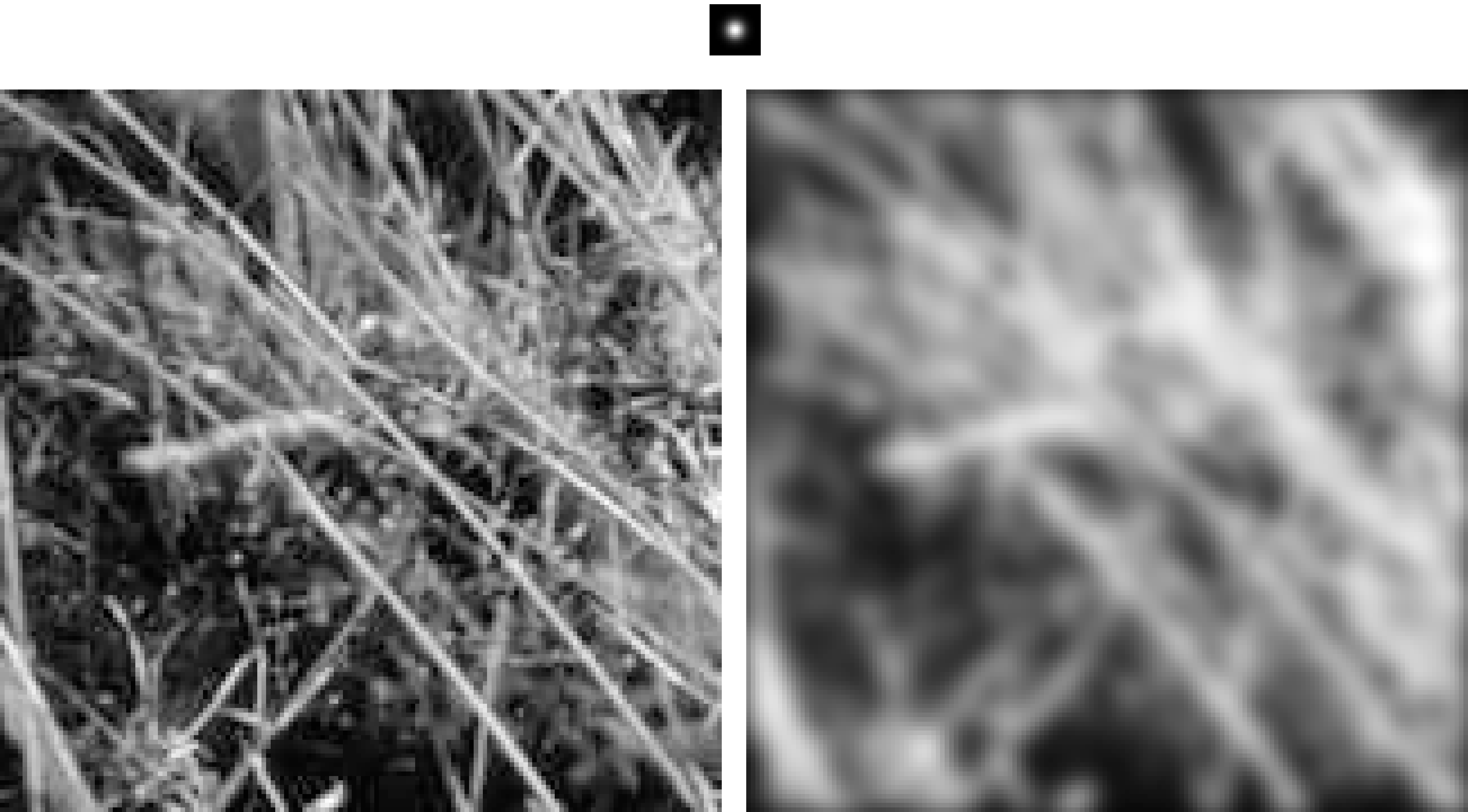


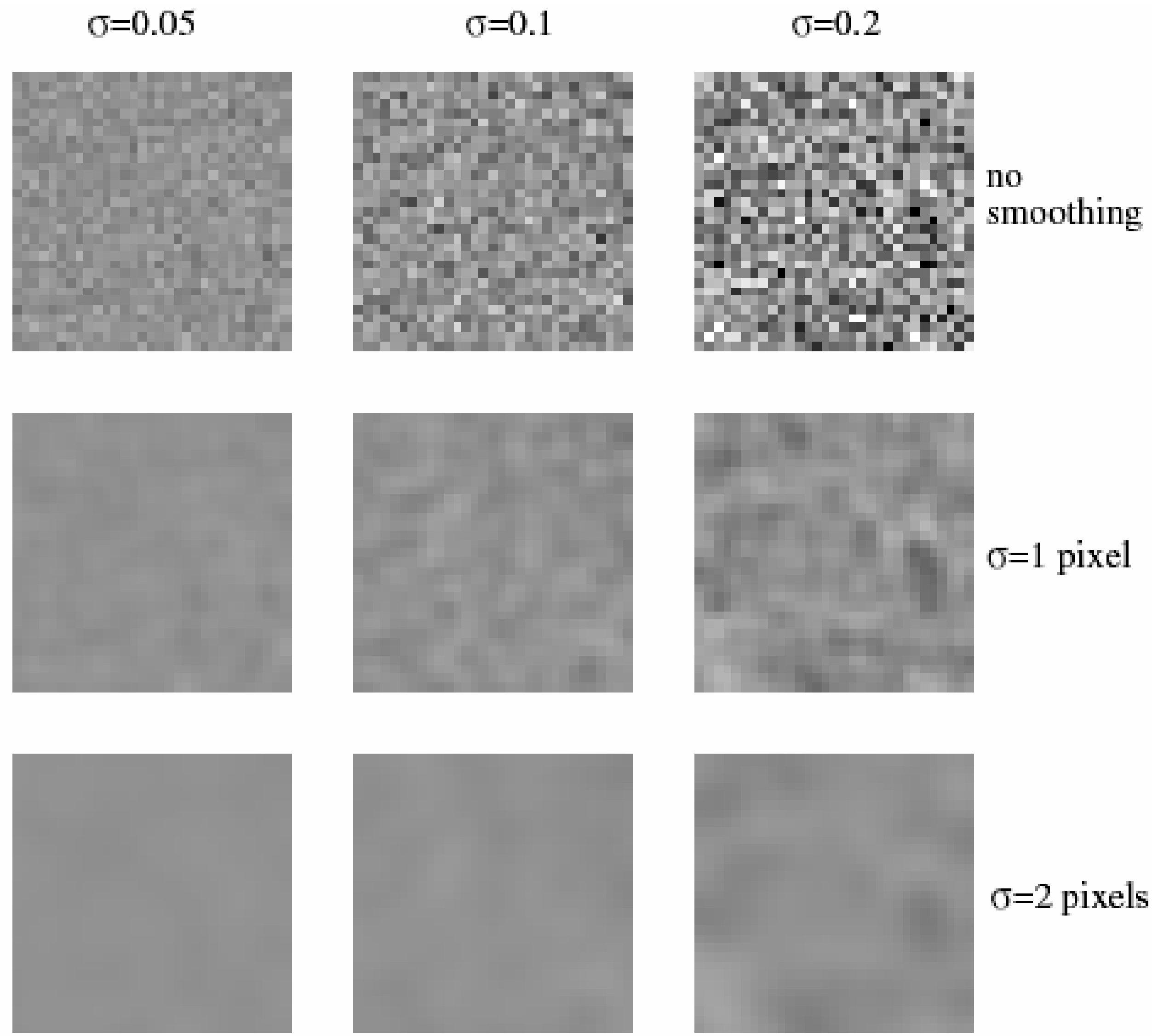
- The picture shows a smoothing kernel proportional to

$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian



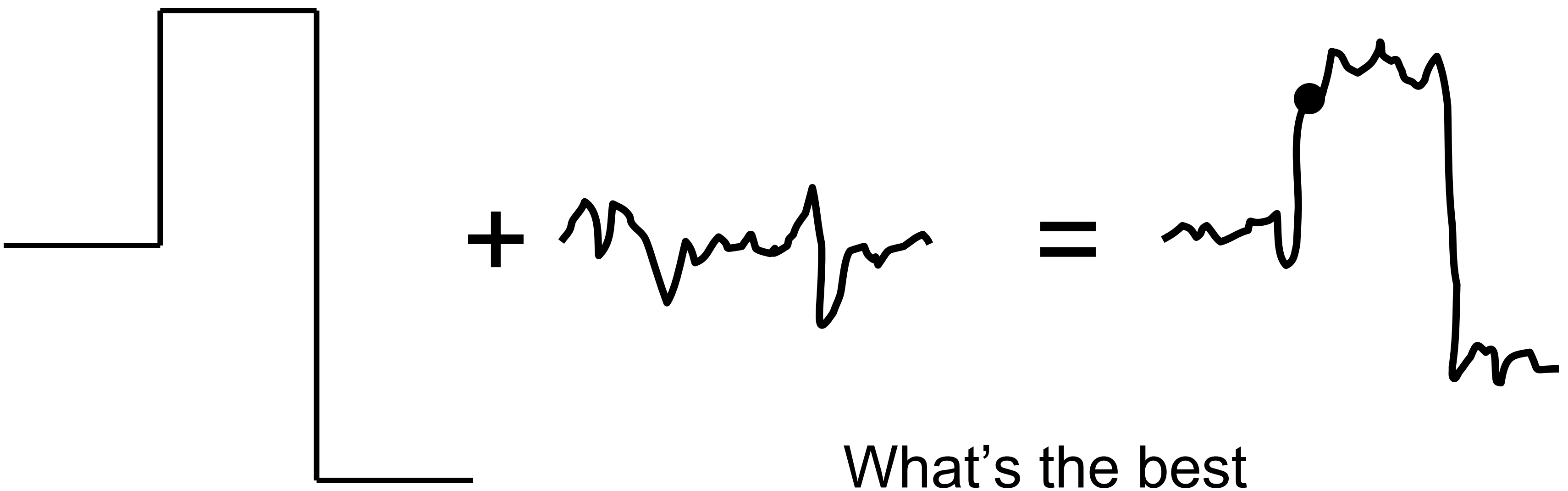


The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

Smoothing as Inference About the Signal: Non-linear Filters.



What's the best neighborhood for inference?

Filtering to reduce noise: Lessons

- Noise reduction is probabilistic inference.
- Depends on knowledge of signal and noise.
- In practice, simplicity and efficiency important.