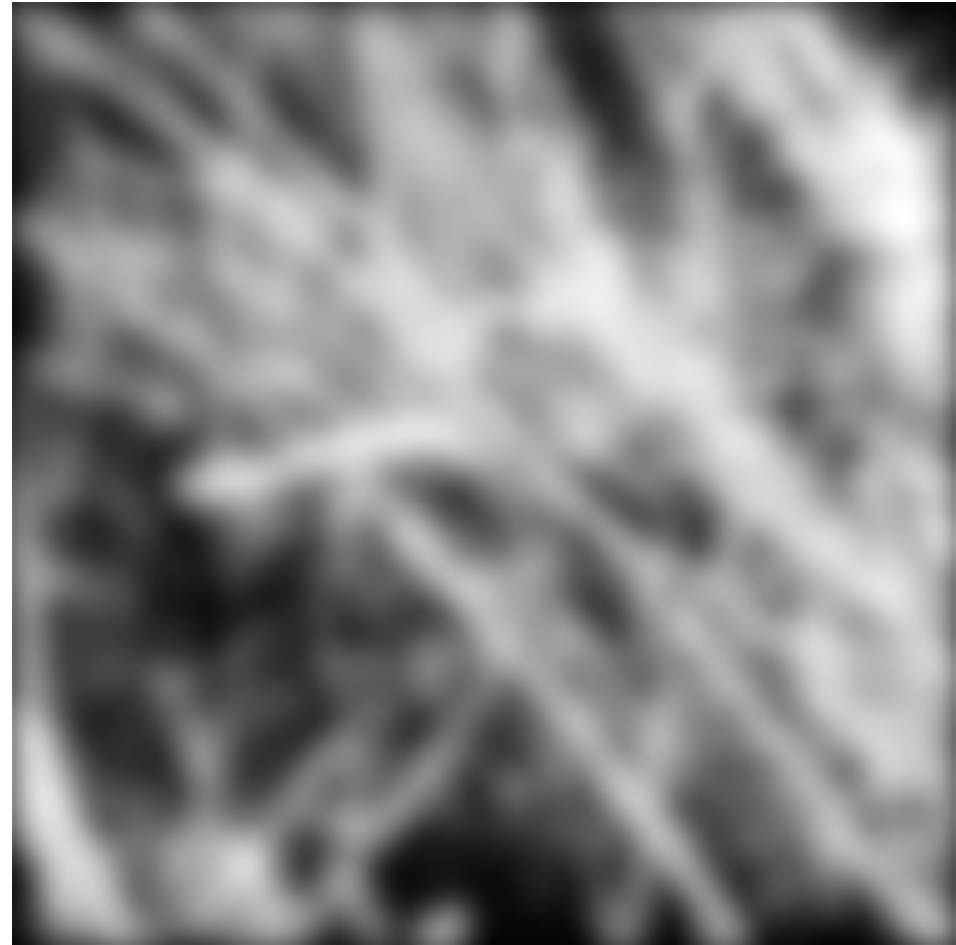


Linear filtering



Motivation: Image denoising

- How can we reduce noise in a photograph?



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?

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

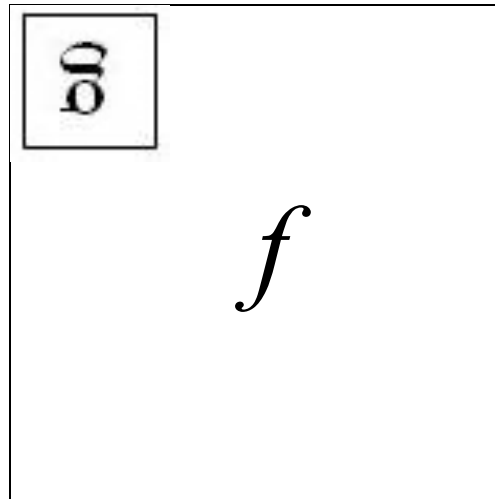
“box filter”

Defining convolution

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

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

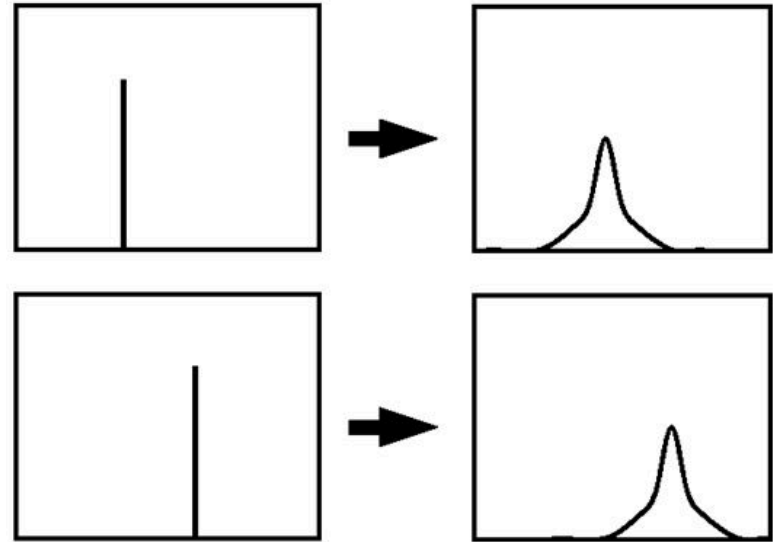
Convention:
kernel is “flipped”



Key properties

- **Shift invariance:** same behavior regardless of pixel location:

$$\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$$



- **Linearity:**

$$\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$$

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

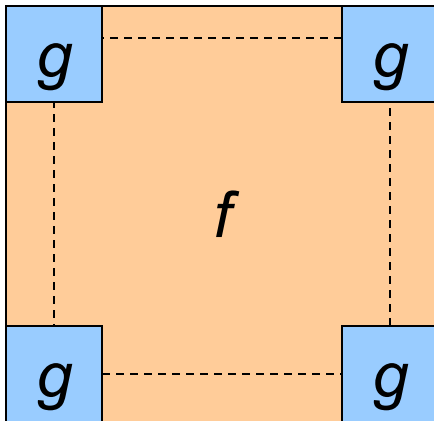
Properties in more detail

- 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 $e = [\dots, 0, 0, 1, 0, 0, \dots]$,
 $a * e = a$

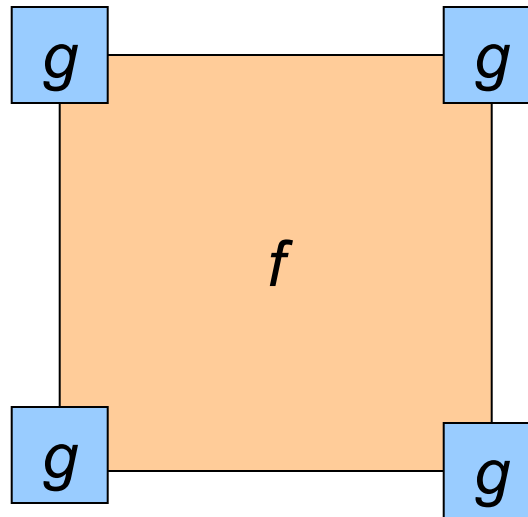
Dealing with edges

- If we convolve image f with filter g , what is the size of the output?

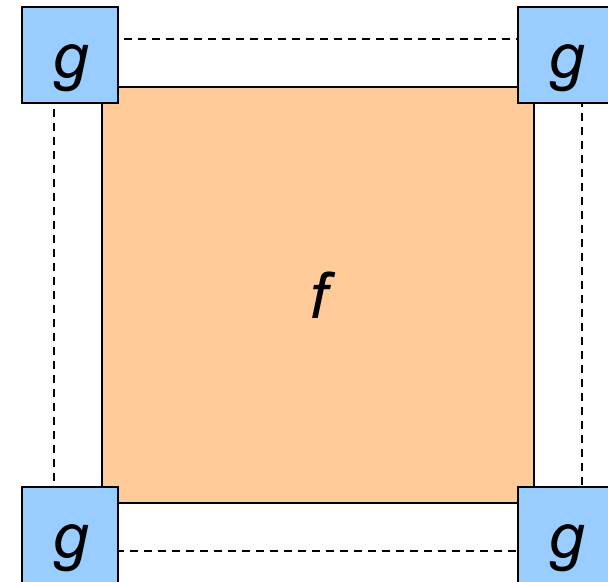
Output is smaller than input



Output is same size as input



Output is larger than input

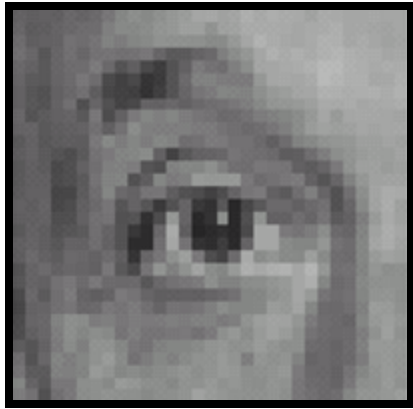


Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
 - Zero pad (or clip filter)
 - Wrap around
 - Copy edge
 - Reflect across edge



Practice with linear filters

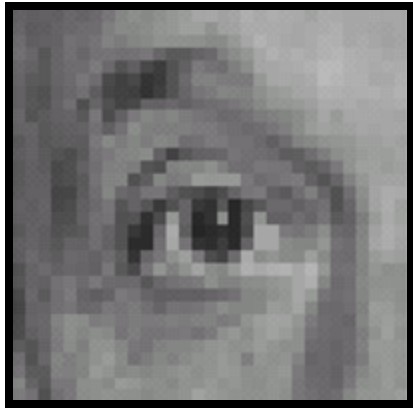


Original

0	0	0
0	1	0
0	0	0

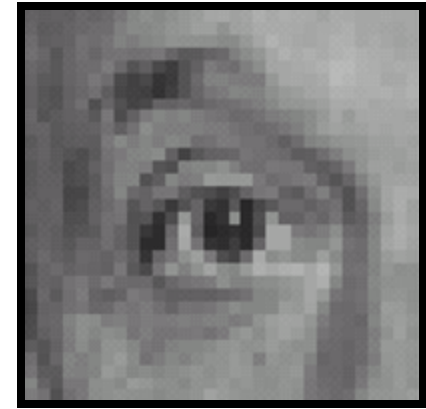
?

Practice with linear filters



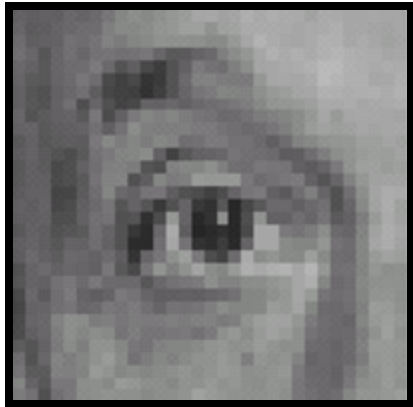
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters

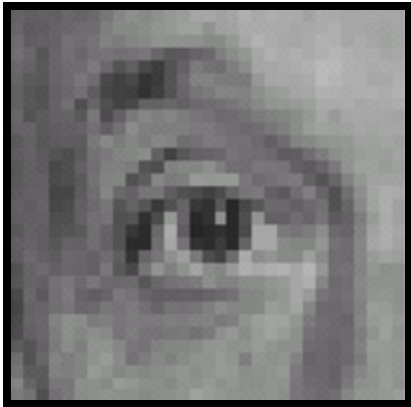


Original

0	0	0
0	0	1
0	0	0

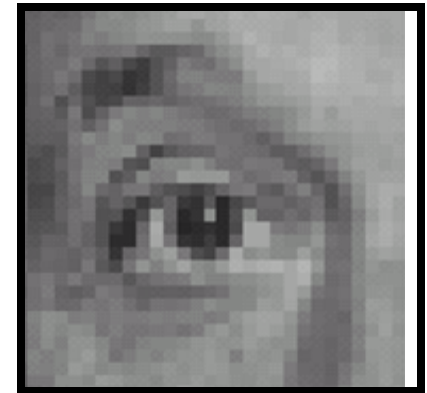
?

Practice with linear filters



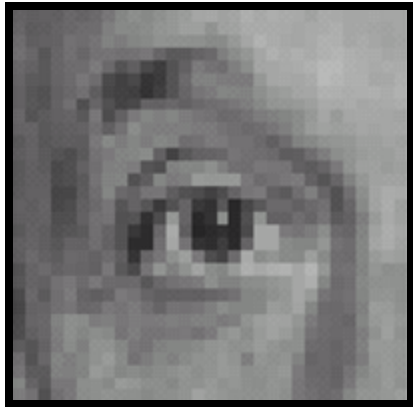
Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Practice with linear filters



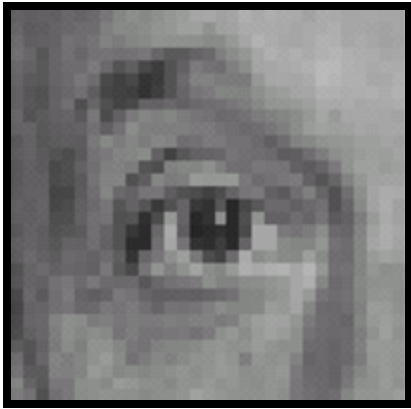
Original

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

?

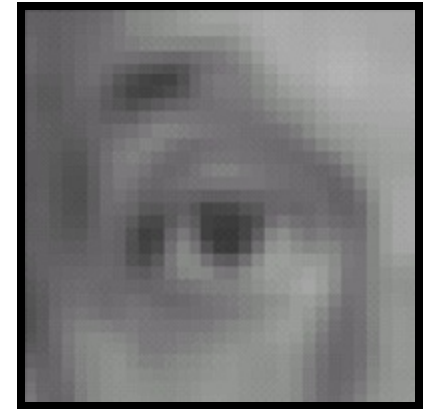
Practice with linear filters



Original

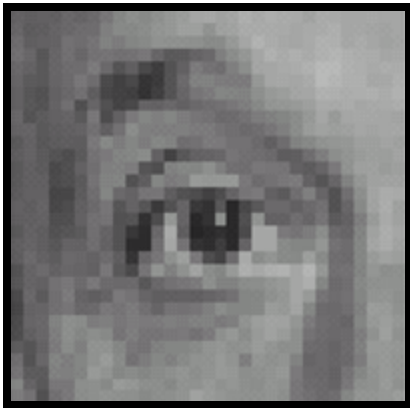
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1



Blur (with a
box filter)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

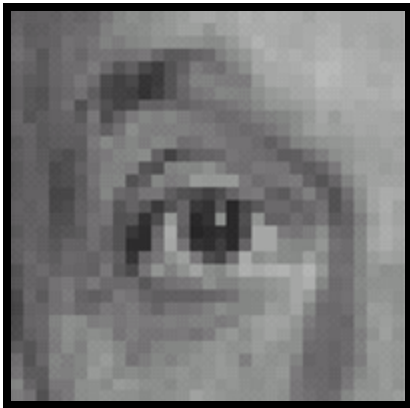
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

?

(Note that filter sums to 1)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$

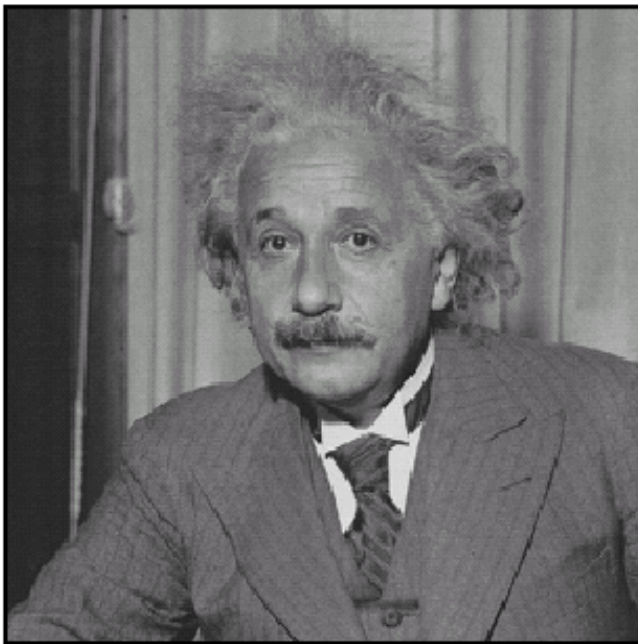
1	1	1
1	1	1
1	1	1



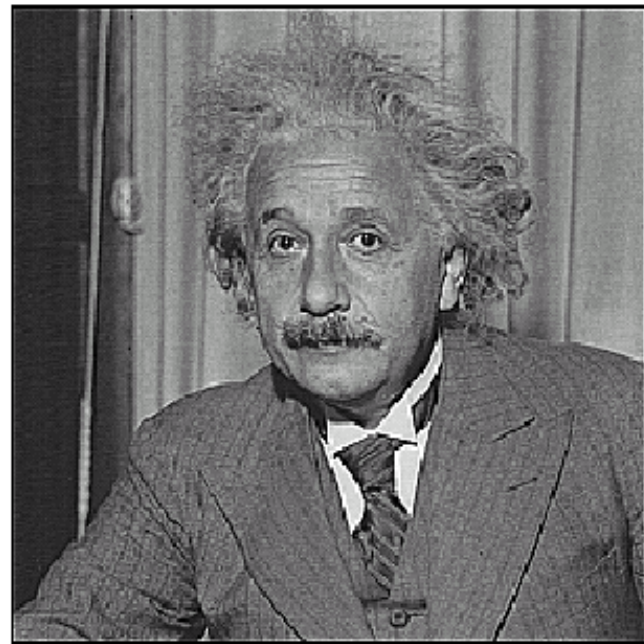
Sharpening filter

- Accentuates differences with local average

Sharpening



before



after

Sharpening

What does blurring take away?



-



=



Let's add it back in.



+ α

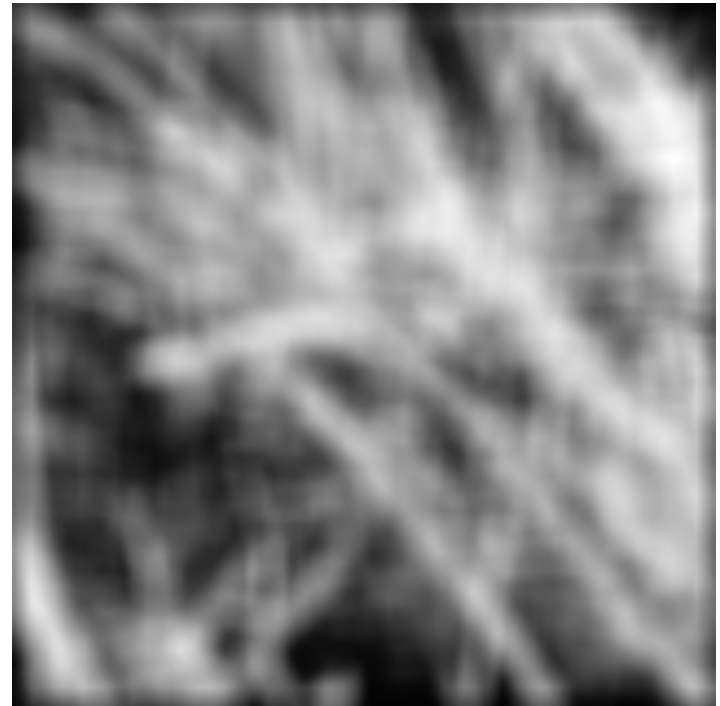
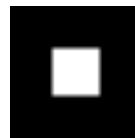


=



Smoothing with box filter revisited

- What's wrong with this picture?
- What's the solution?



Smoothing with box filter revisited

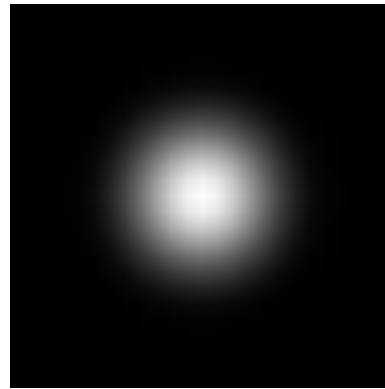
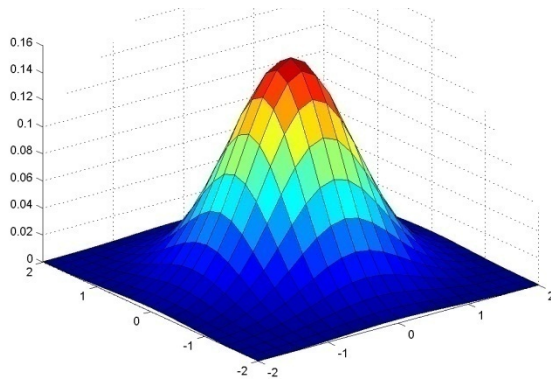
- What's wrong with this picture?
- What's the solution?
 - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center



“fuzzy blob”

Gaussian Kernel

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



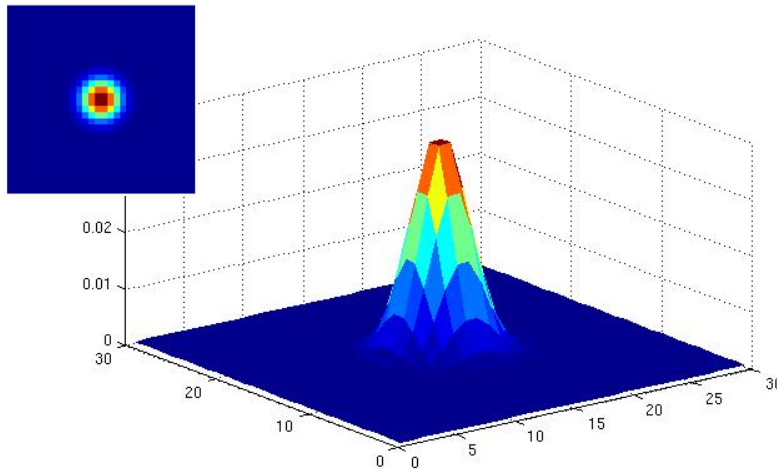
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

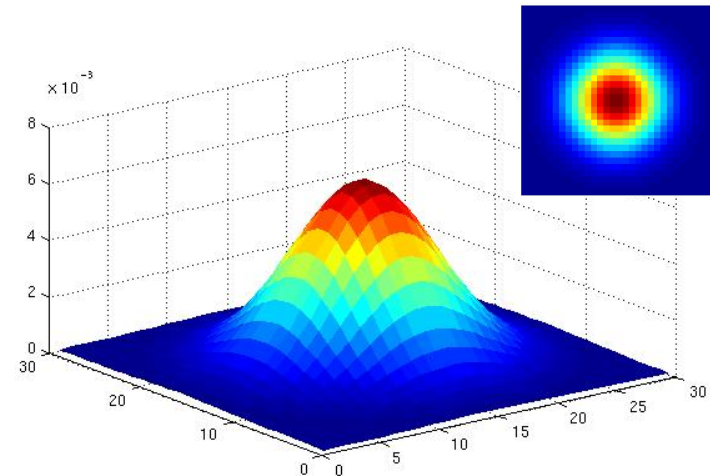
- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

Gaussian Kernel

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



$\sigma = 2$ with 30 x 30
kernel

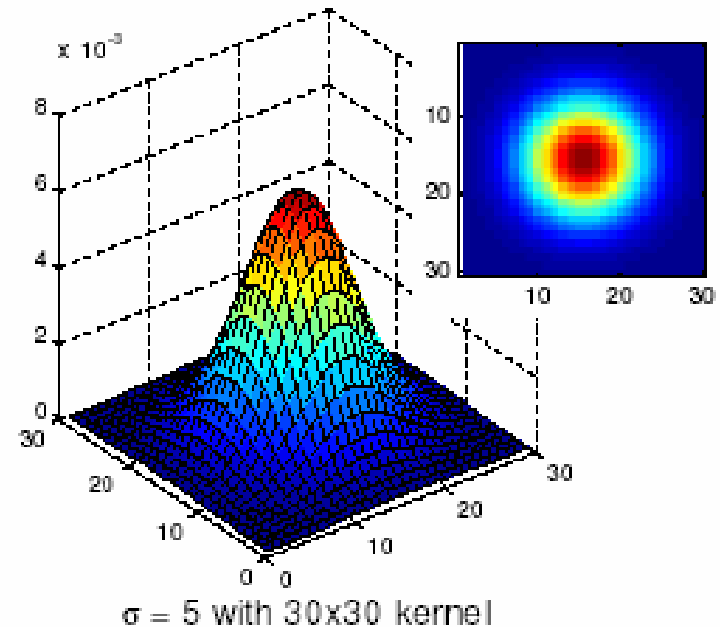
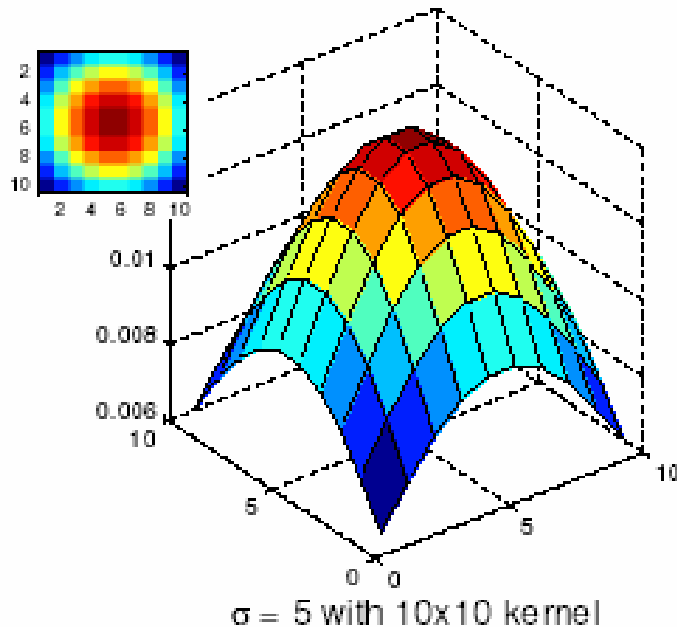


$\sigma = 5$ with 30 x 30
kernel

- Standard deviation σ : determines extent of smoothing

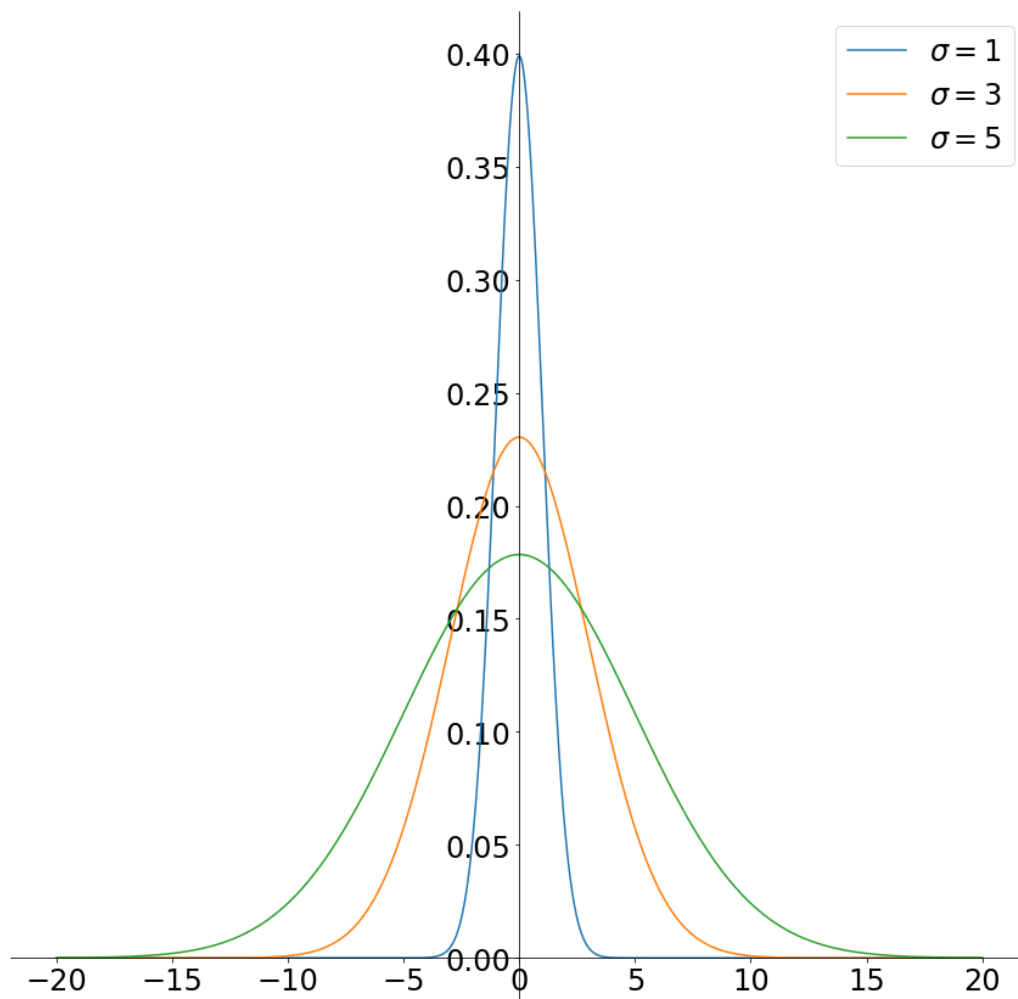
Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels

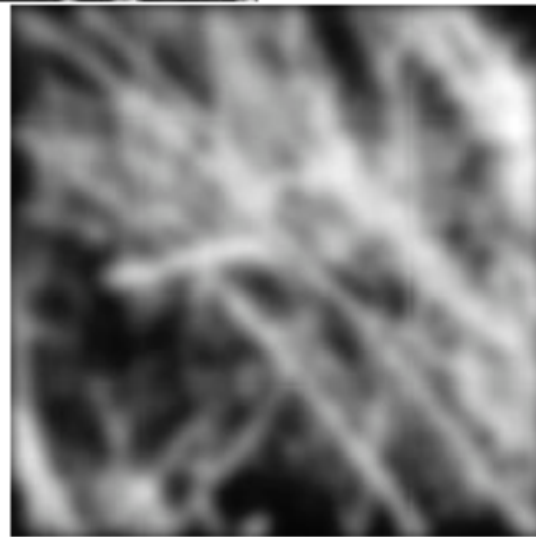


Choosing kernel width

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



Gaussian vs. box filtering



Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
 - So can smooth with small- σ kernel, repeat, and get same result as larger- σ kernel would have
 - Convoluting two times with Gaussian kernel with std. dev. σ is same as convoluting once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
 - Factors into product of two 1D Gaussians
 - Discrete example:

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

Separability of the Gaussian filter

$$\begin{aligned} G_{\sigma}(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

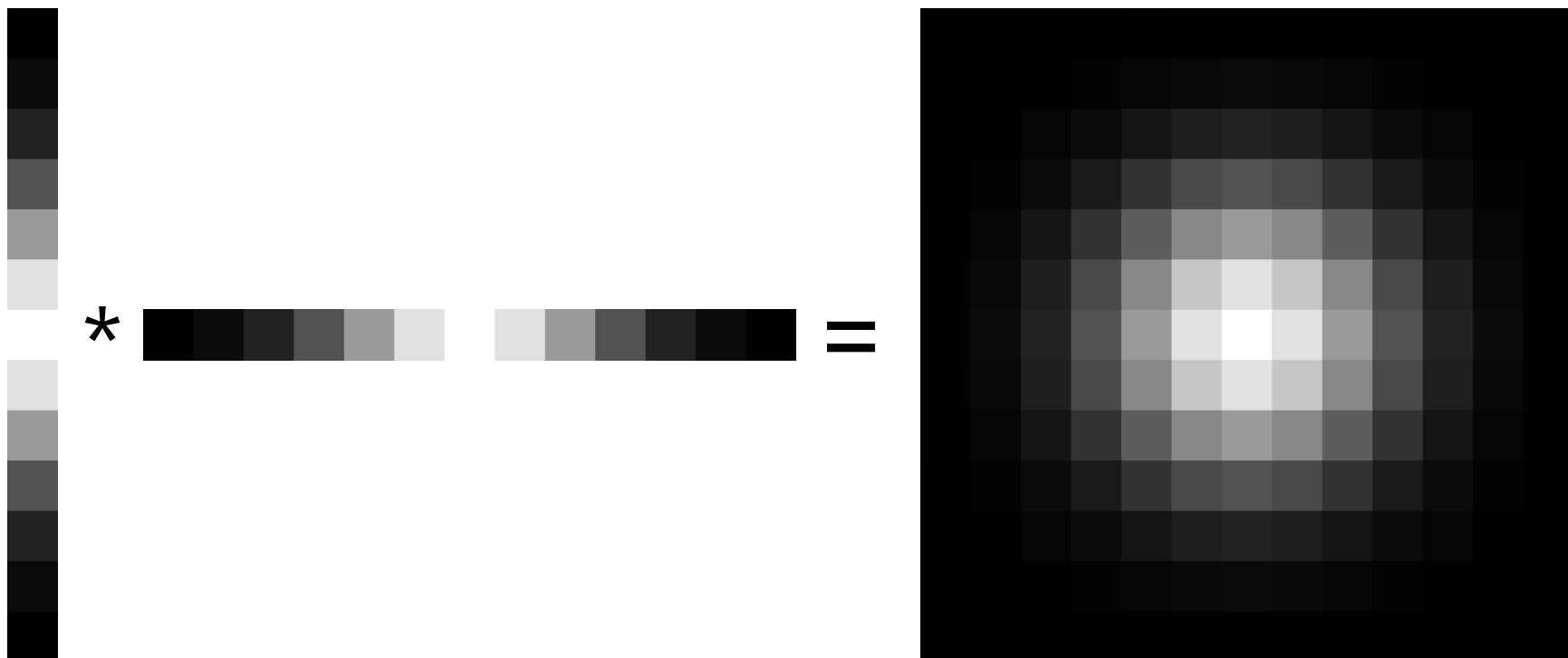
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y .

In this case the two functions are the (identical) 1D Gaussian.

Separability

1D Gaussian * 1D Gaussian = 2D Gaussian

Image * 2D Gauss = Image * (1D Gauss * 1D Gauss)
= (Image * 1D Gauss) * 1D Gauss



Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one along rows and one along columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
 - $O(n^2 m^2)$
- What if the kernel is separable?
 - $O(n^2 m)$

Noise



Original



Salt and pepper noise



Impulse noise



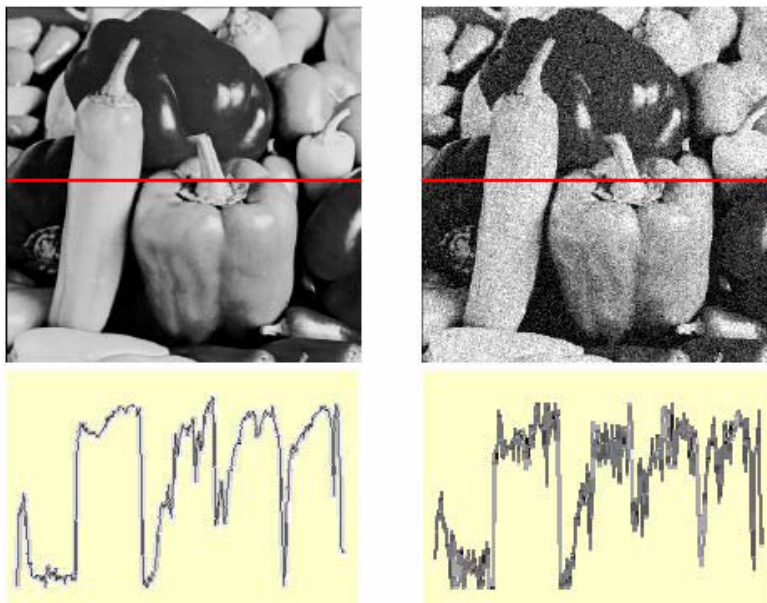
Gaussian noise

- **Salt and pepper noise:** contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

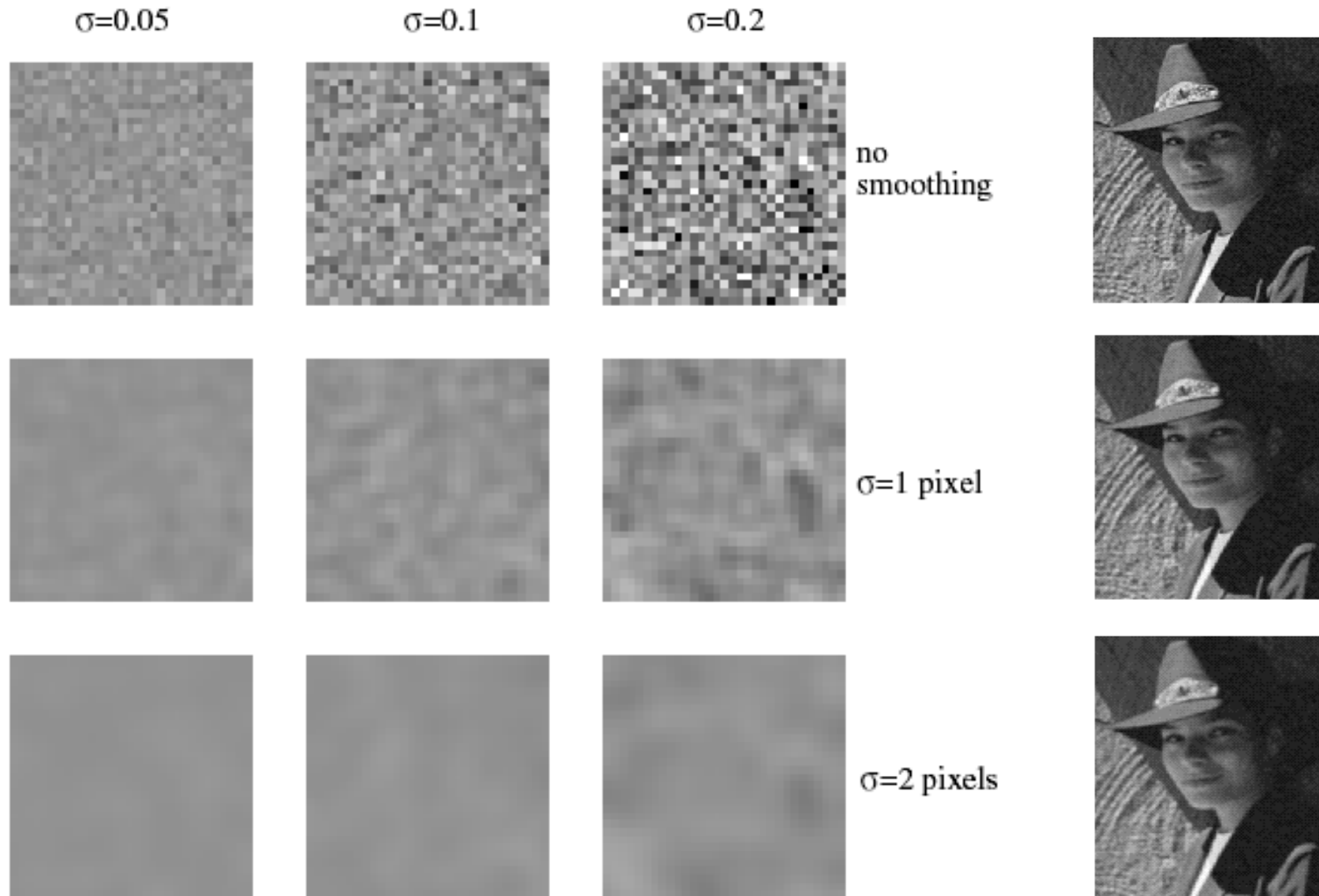
Image
Noise



$$f(x, y) = \overbrace{\hat{f}(x, y)}^{\text{Ideal Image}} + \overbrace{\eta(x, y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
 $\eta(x, y) \sim \mathcal{N}(\mu, \sigma)$

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise

3x3



5x5



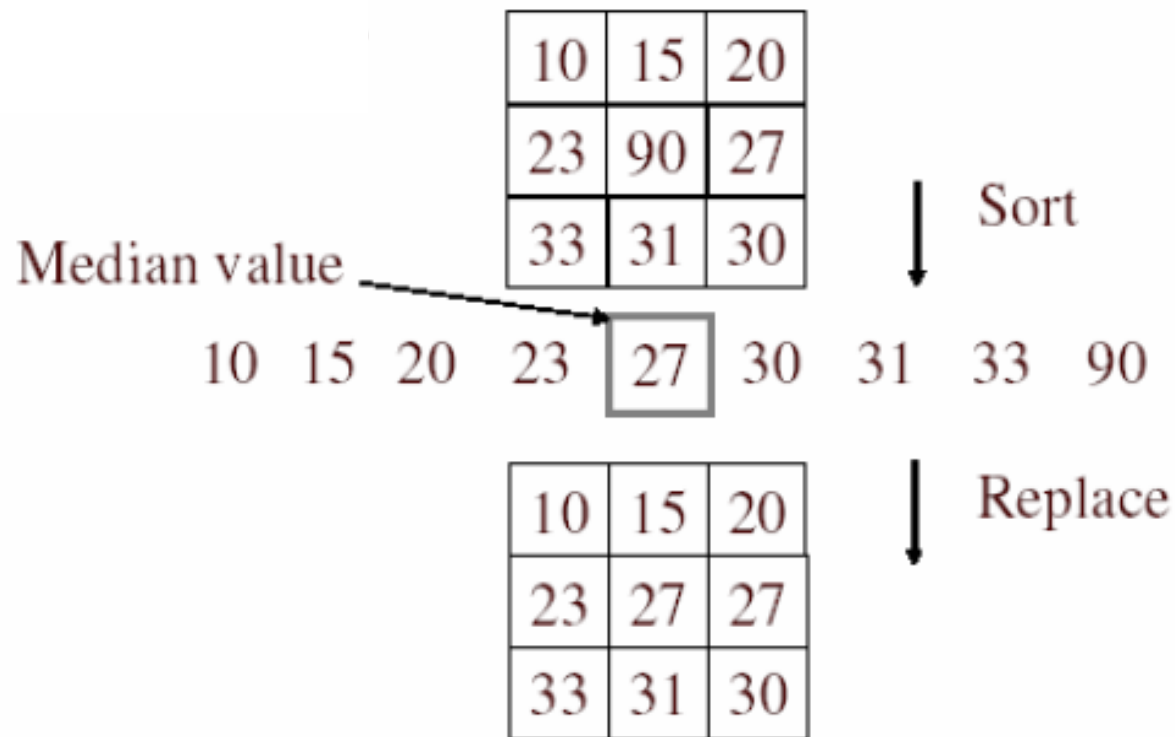
7x7



What's wrong with the results?

Alternative idea: Median filtering

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



- Is median filtering linear?

Median filter

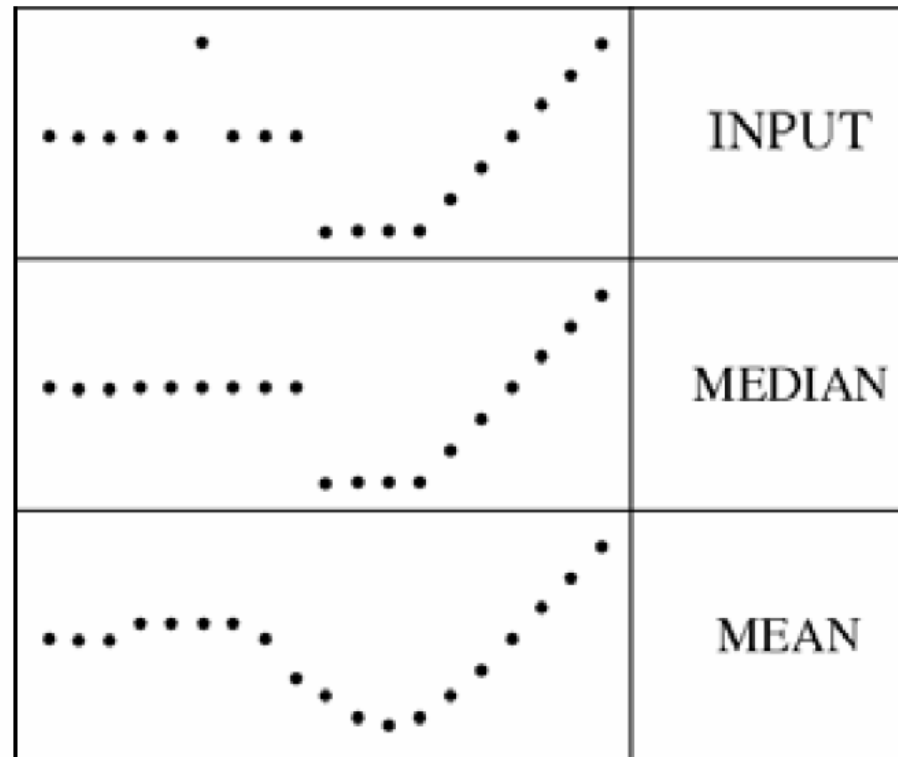
- Is median filtering linear?
- Let's try filtering

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

Median filter

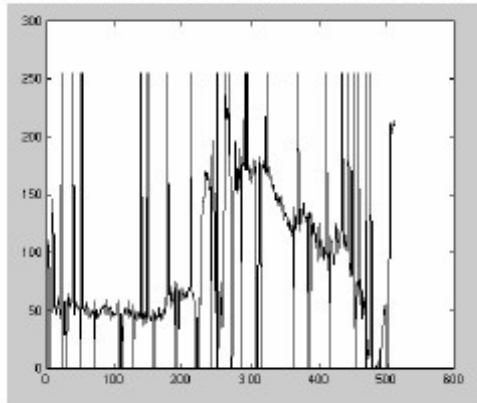
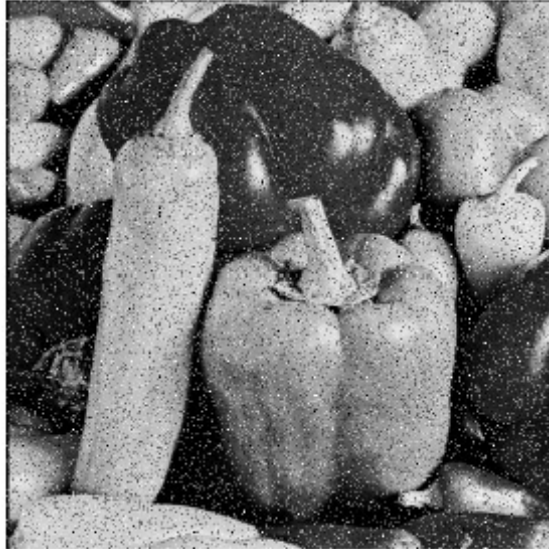
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :

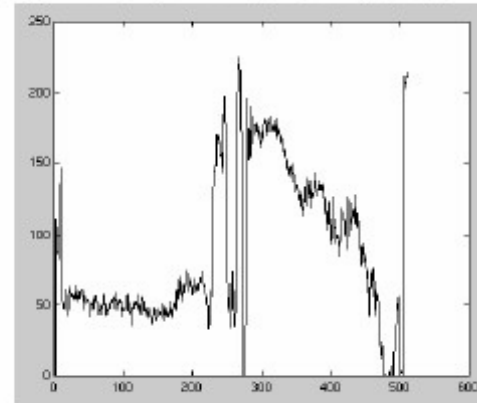


Median filter

Salt-and-pepper noise



Median filtered



Gaussian vs. median filtering

3x3

5x5

7x7

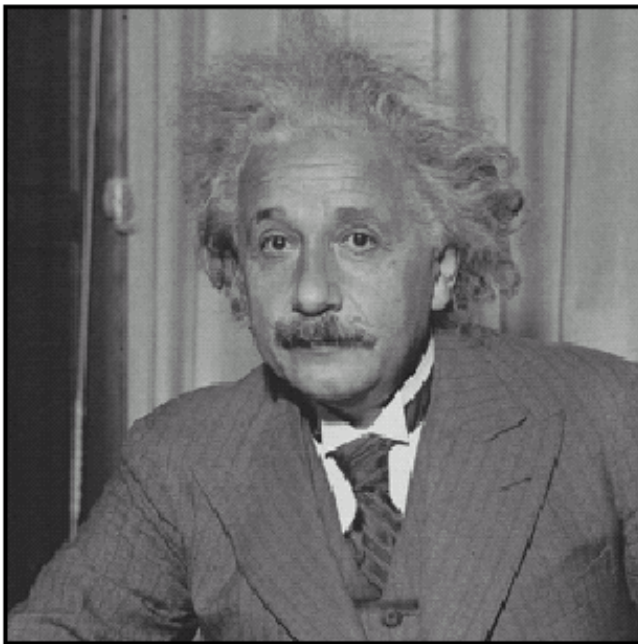
Gaussian



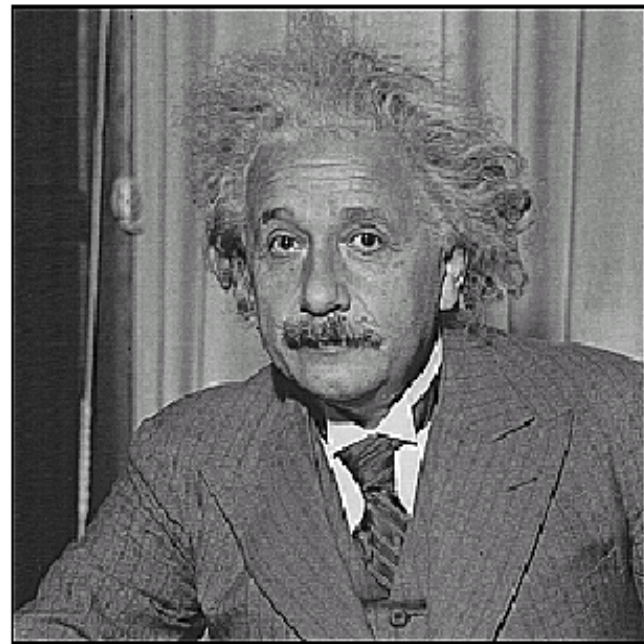
Median



Sharpening revisited



before



after

Sharpening

What does blurring take away?



-



=



Let's add it back in.



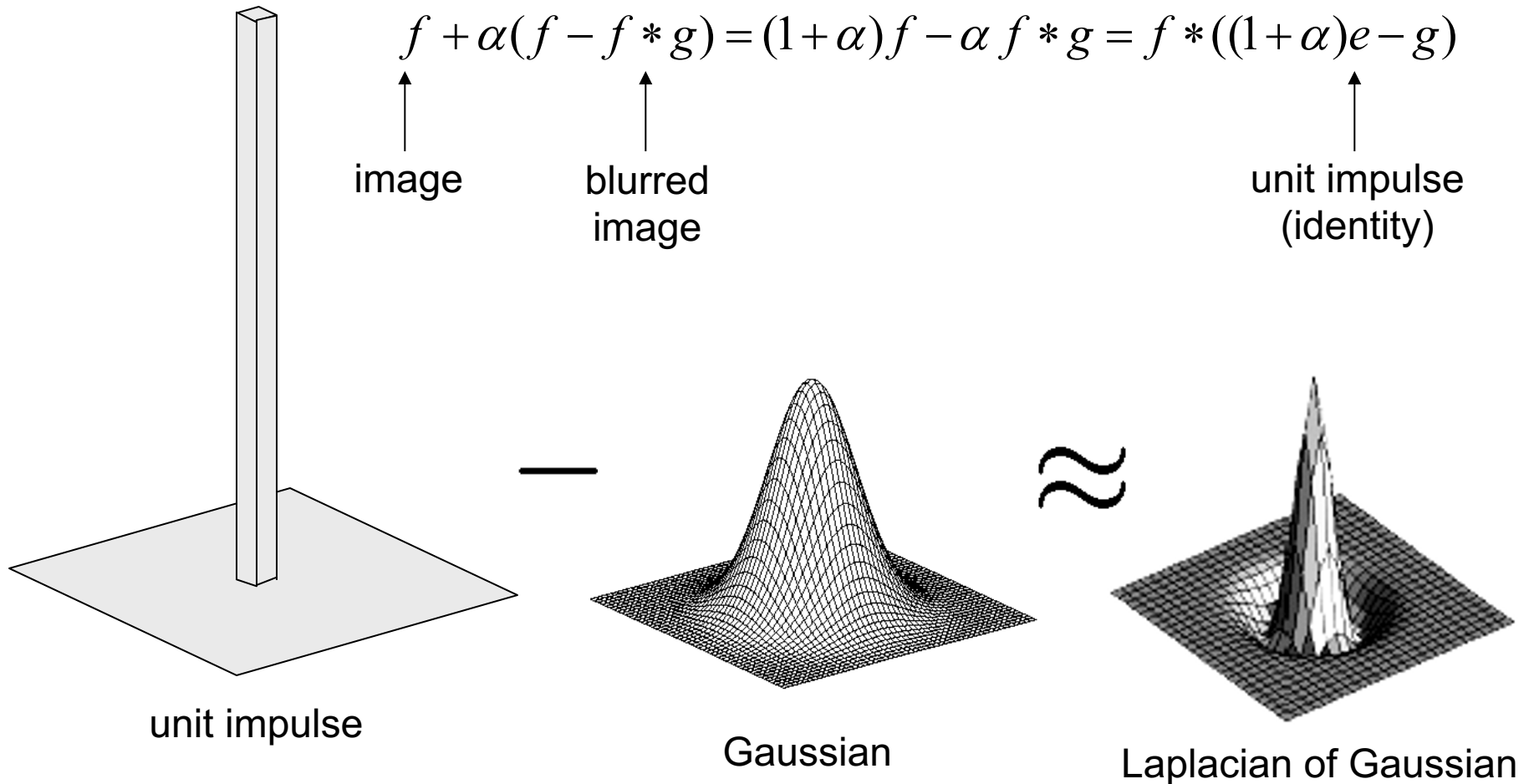
+ α



=



Unsharp mask filter



Next Class: Frequency view of filtering

Review: Image filtering

- Convolution
- Box vs. Gaussian filter
- Separability
- Median filter