



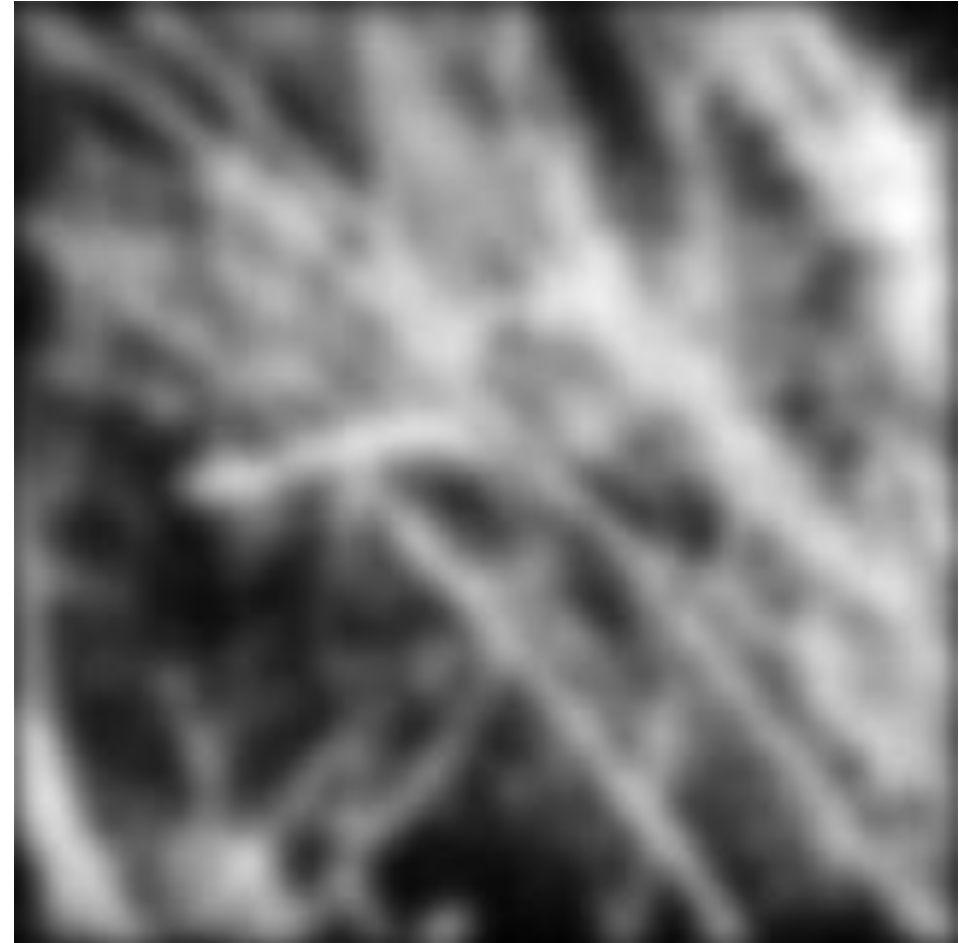
COMPUTER VISION

Le Thanh Ha, Ph.D

Assoc. Prof. at University of Engineering and Technology,
Vietnam National University

ltha@vnu.edu.vn; lthavnu@gmail.com; 0983 692 592

Linear 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?

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 a 3x3 moving average?

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

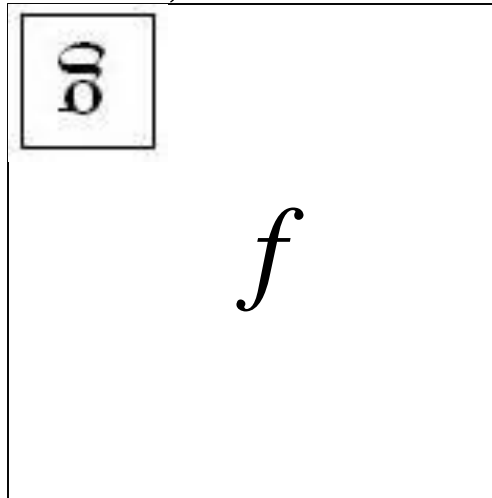
1	1	1
1	1	1
1	1	1

“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”
- MATLAB: conv2 vs. filter2 (also imfilter)

Key properties

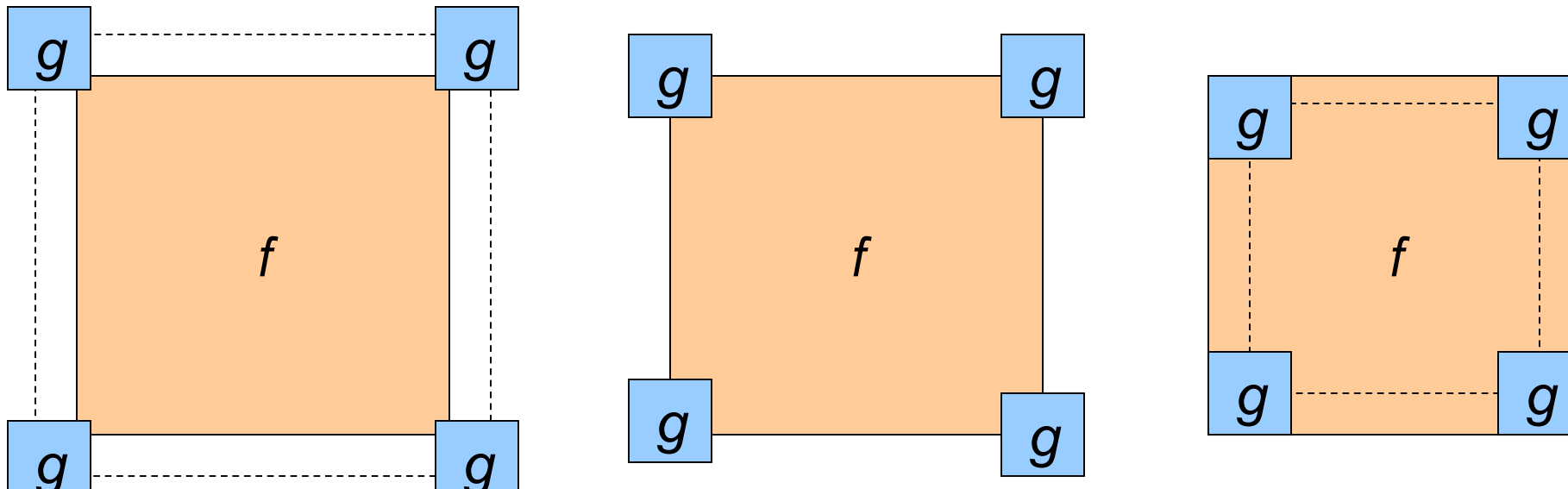
- **Linearity:** $filter(f_1 + f_2) = filter(f_1) + filter(f_2)$
- **Shift invariance:** same behavior regardless of pixel location:
 $filter(shift(f)) = shift(filter(f))$
- **Theoretical result:** any linear shift-invariant operator can be represented as a convolution

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

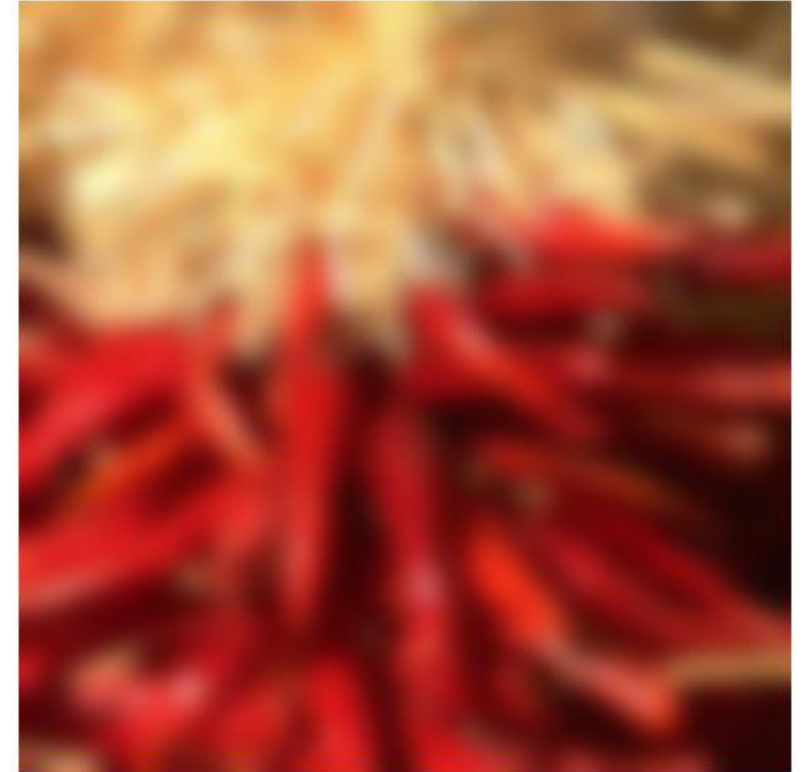
Annoying details

- What is the size of the output?
- MATLAB: `filter2(g, f, 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



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')`

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

?

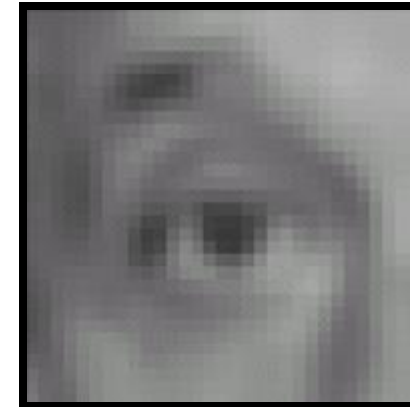
Practice with linear filters



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

—

$\frac{1}{9}$

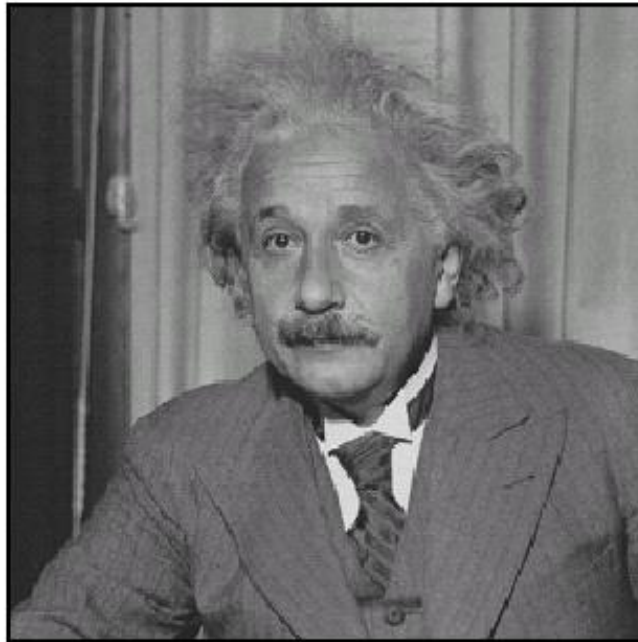
1	1	1
1	1	1
1	1	1



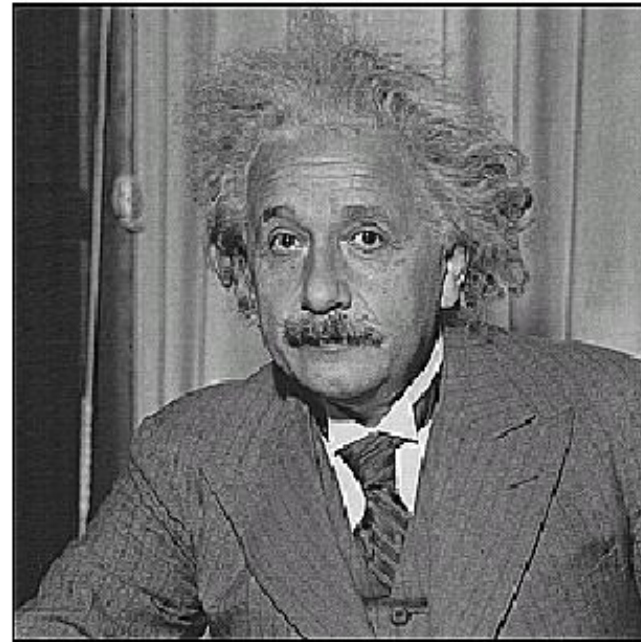
Sharpening filter

- Accentuates differences with local average

Sharpening



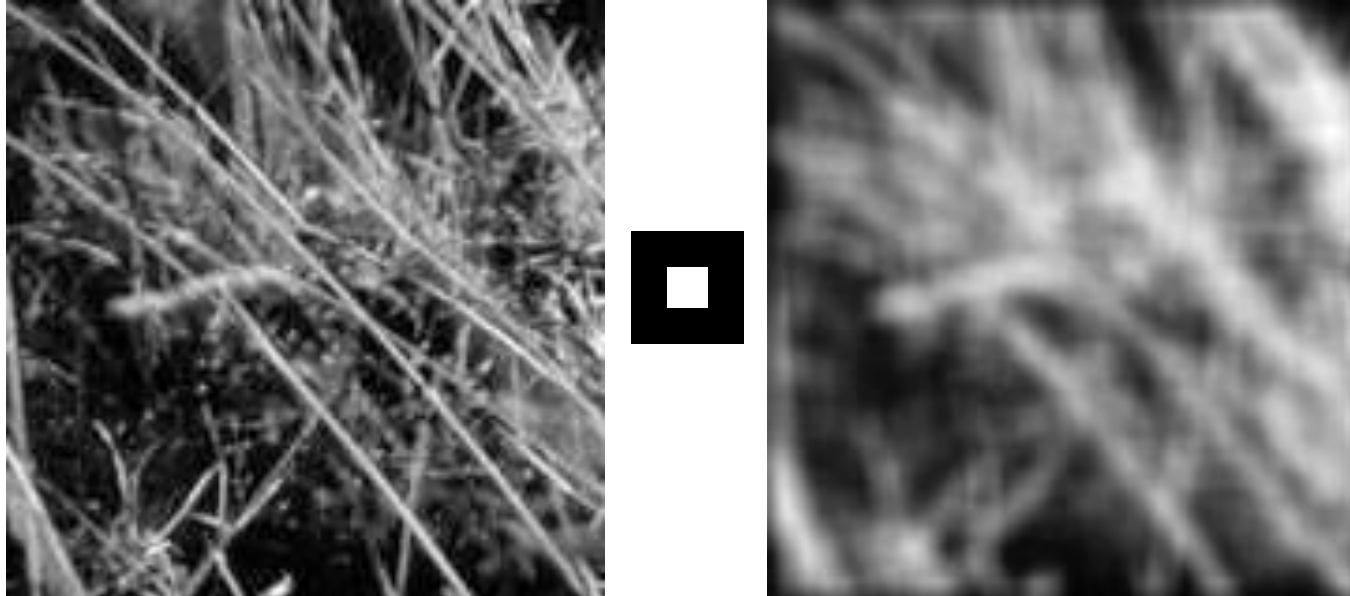
before



after

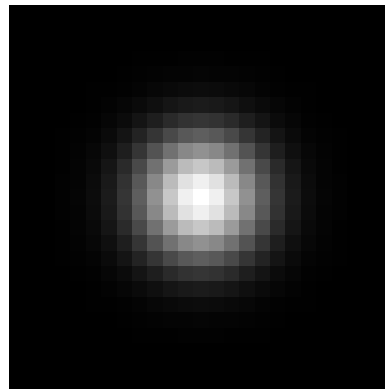
Smoothing with box filter revisited

- 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



Smoothing with box filter revisited

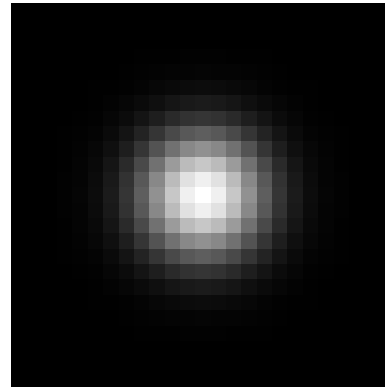
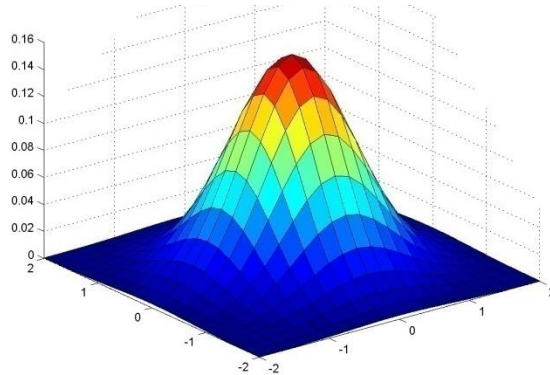
- Smoothing with an average actually doesn't compare at all well with a defocused lens
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square
- Better idea: to eliminate edge effects, weight contribution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



“fuzzy blob”

Gaussian Kernel

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



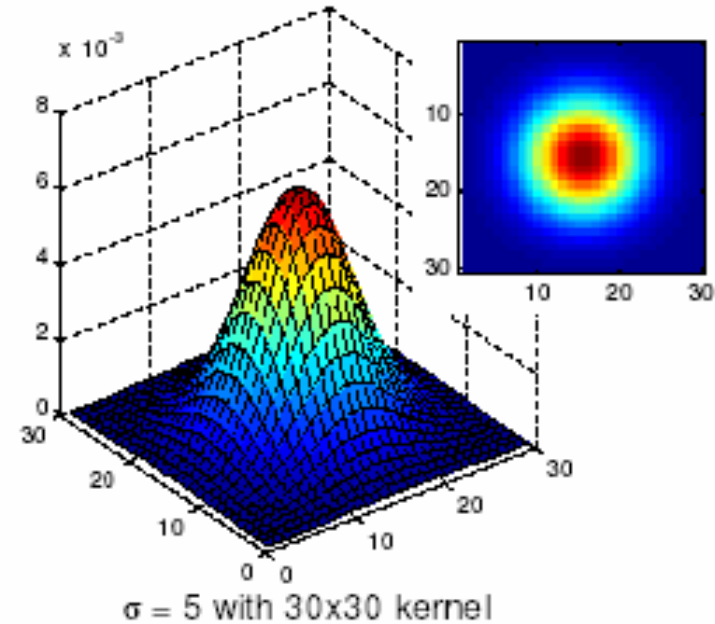
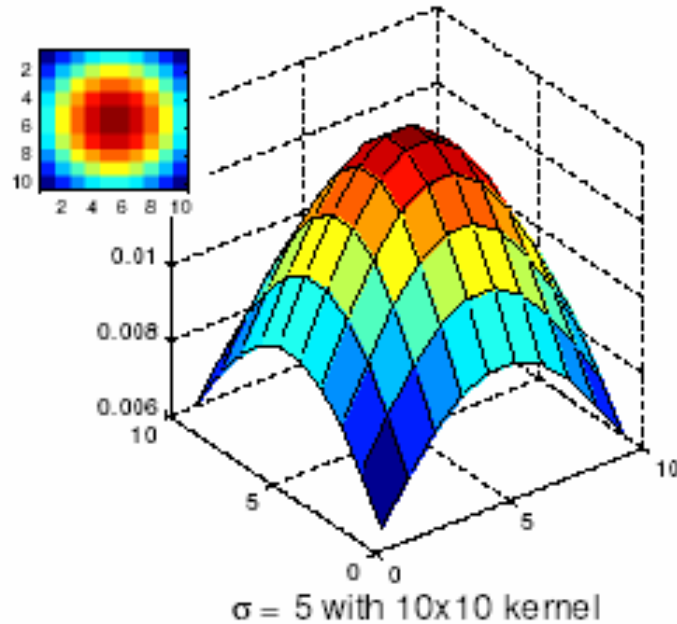
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

- 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)

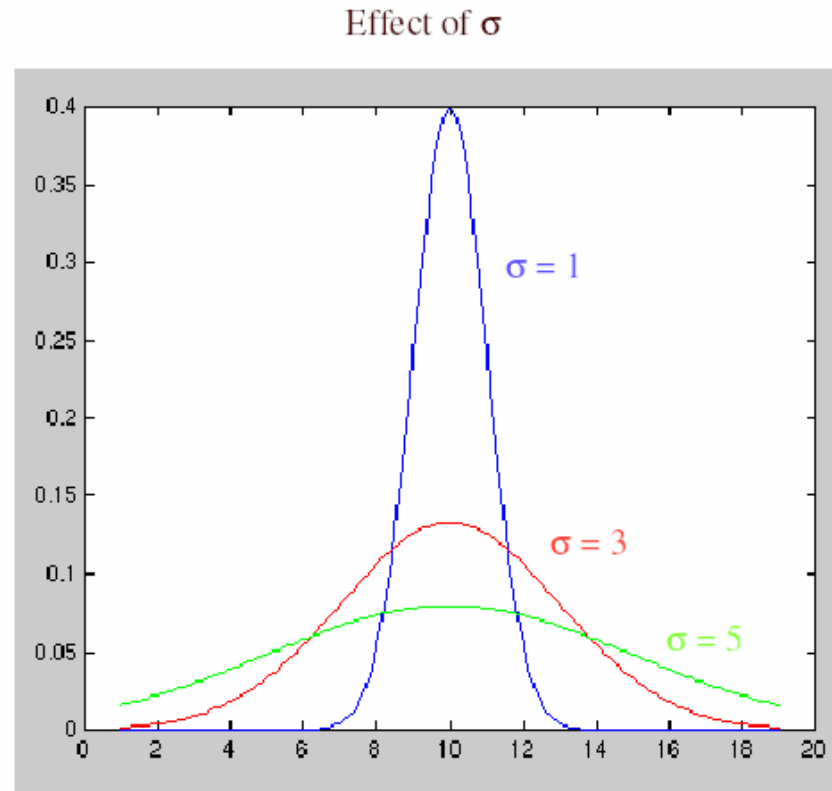
Choosing kernel width

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

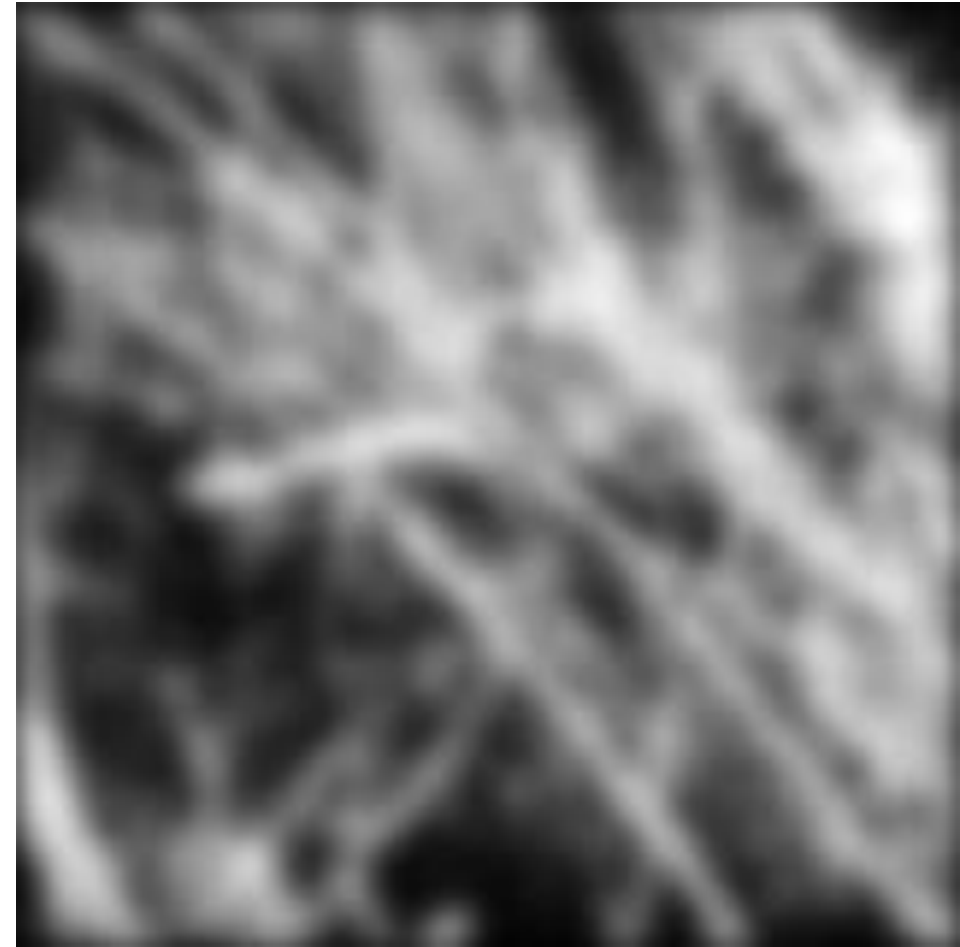
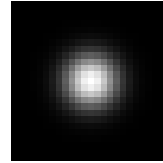


Choosing kernel width

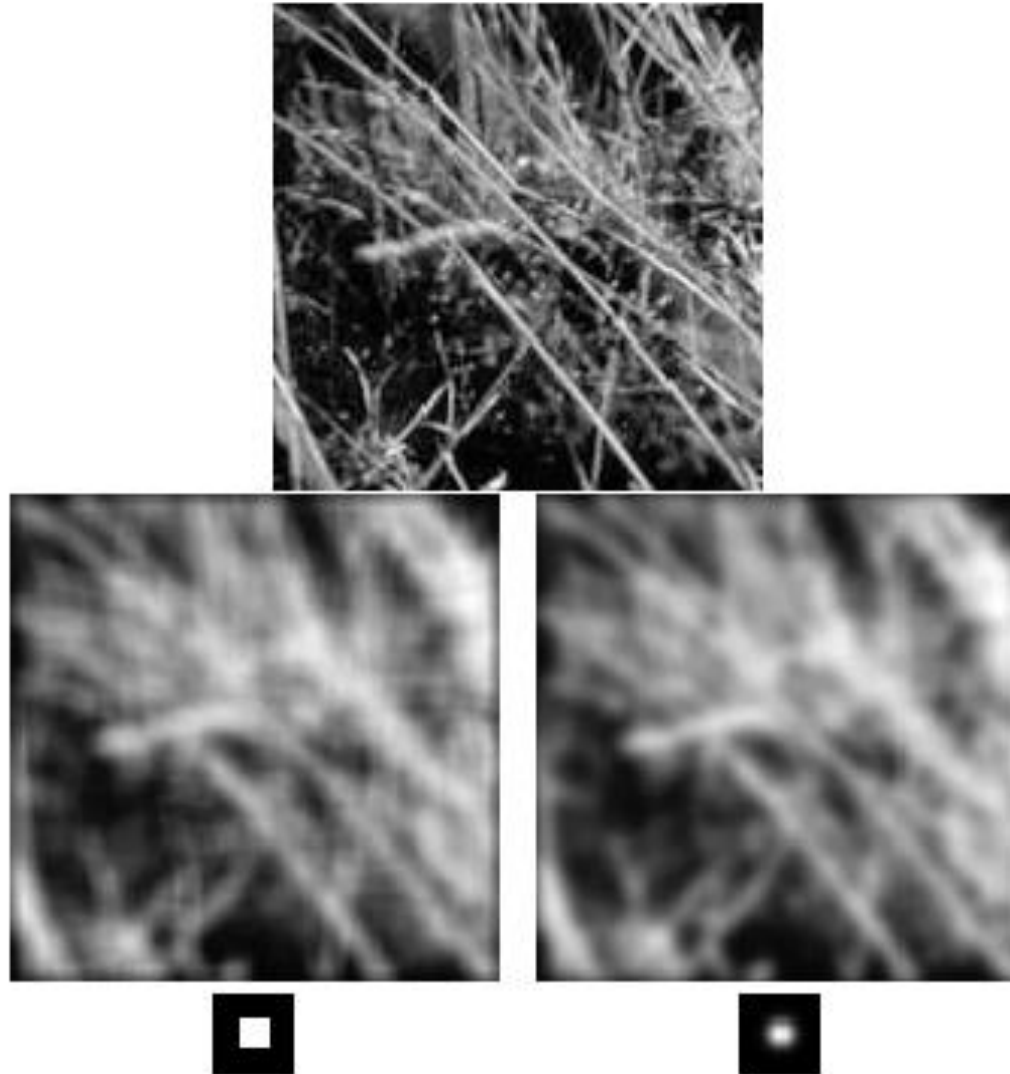
- Rule of thumb: set filter half-width to about 3σ



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
 - Convoluting two times with Gaussian kernel of width σ is same as convoluting once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
 - Factors into product of two 1D Gaussians

Separability of the Gaussian filter

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

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 example

2D convolution
(center location only)

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{l} = 2 + 6 + 3 = 11 \\ = 6 + 20 + 10 = 36 \\ = 4 + 8 + 6 = 18 \\ \hline 65 \end{array}$$

The filter factors
into a product of 1D
filters:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array}$$

Perform convolution
along rows:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array}$$

Followed by convolution
along the remaining column:

$$\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & & \\ \hline & 65 & \\ \hline & & \\ \hline \end{array}$$

Separability

- Why is separability useful in practice?

Noise



Original



Salt and pepper noise



Impulse noise

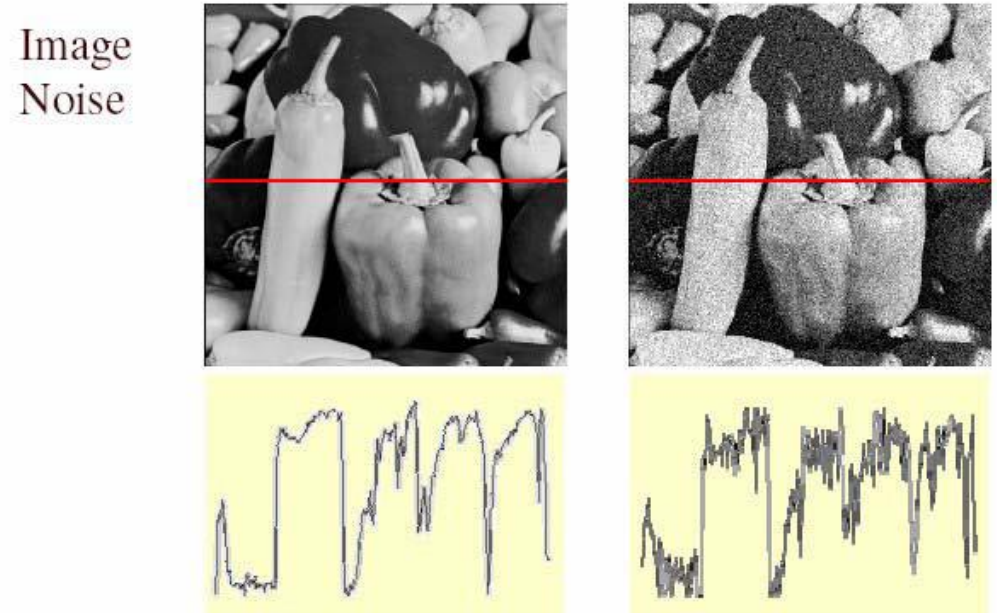


Gaussian noise

- **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

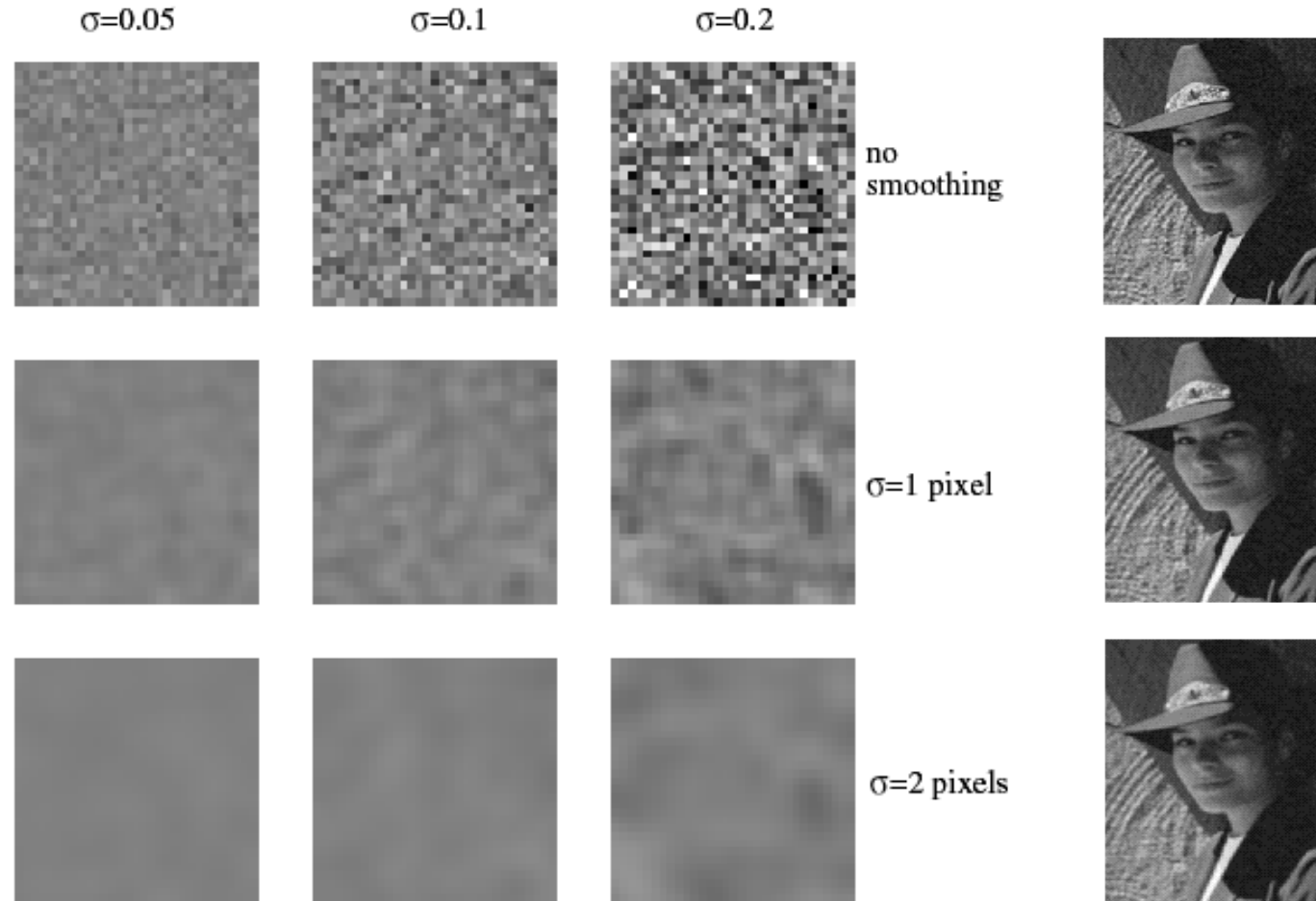
- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise



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

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



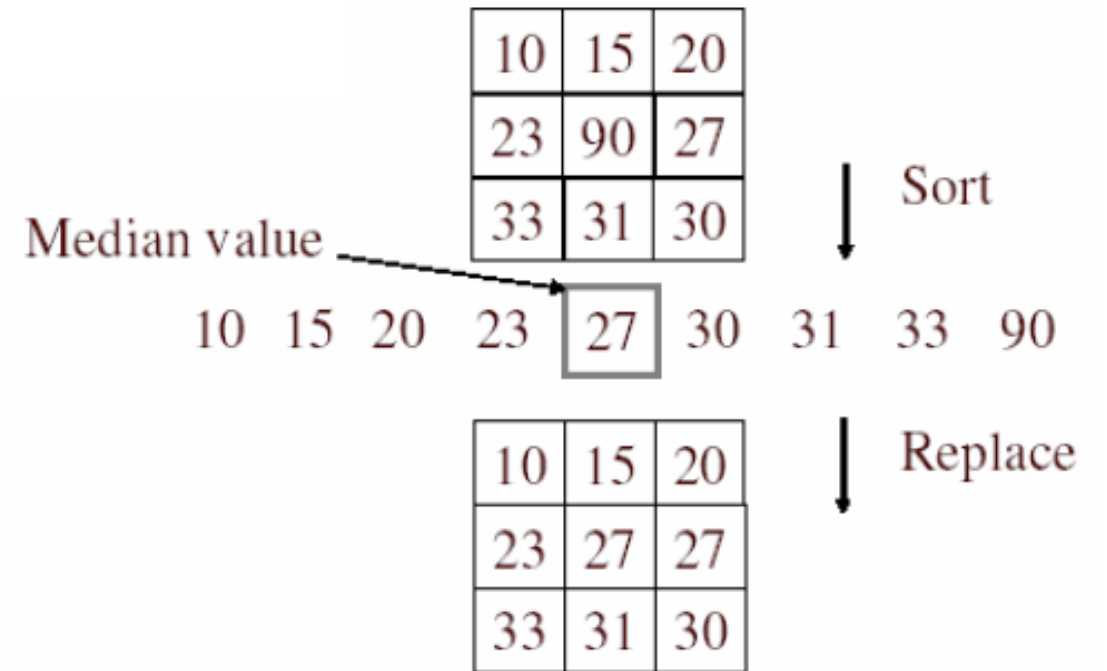
7x7



- 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

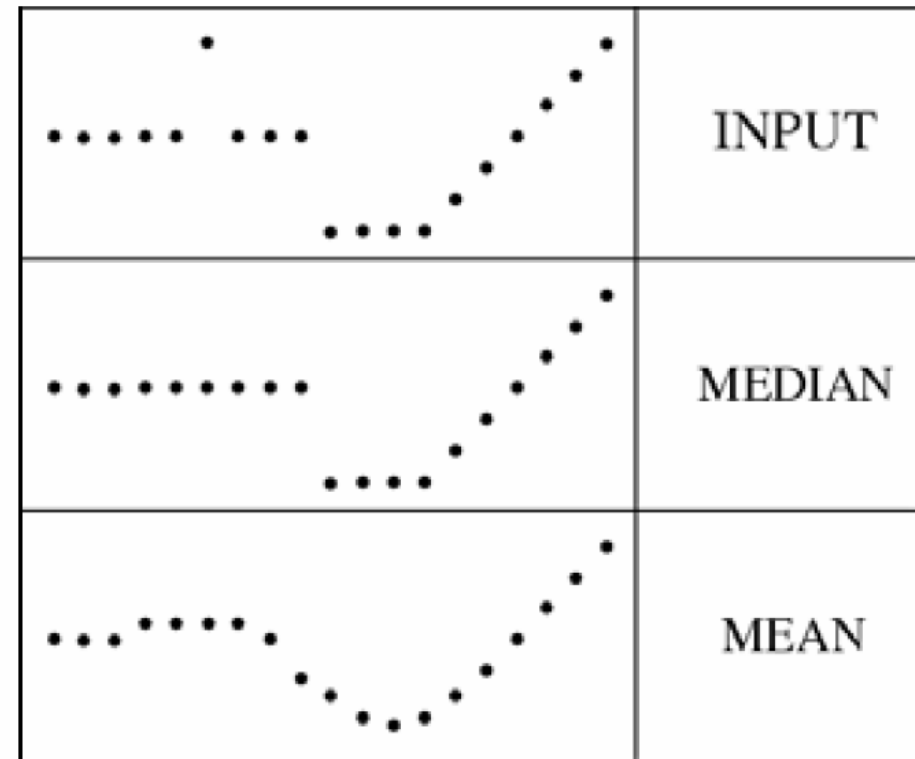


- Is median filtering linear?

Median filter

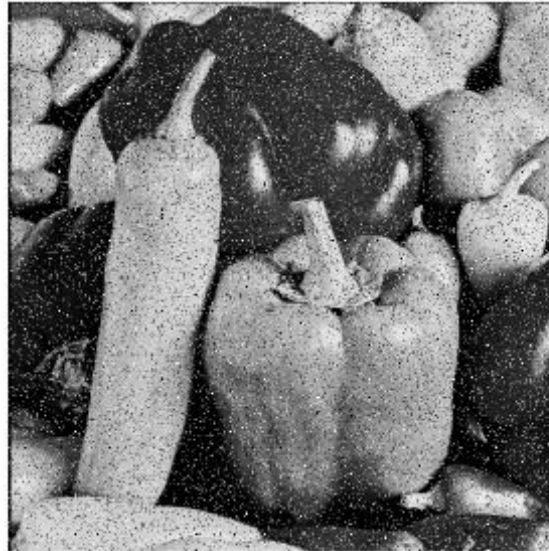
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :

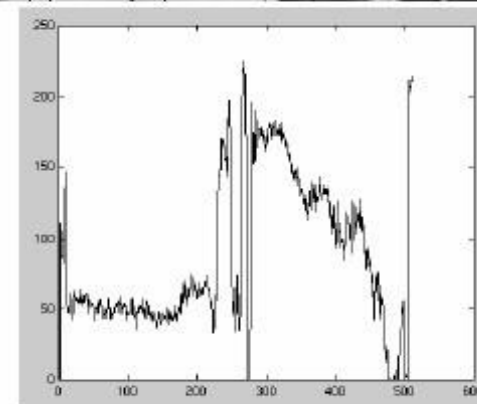
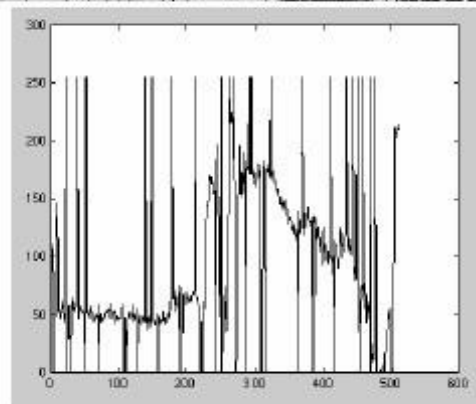


Median filter

Salt-and-pepper noise



Median filtered



Median vs. Gaussian filtering

3x3

5x5

7x7

Gaussian



Median

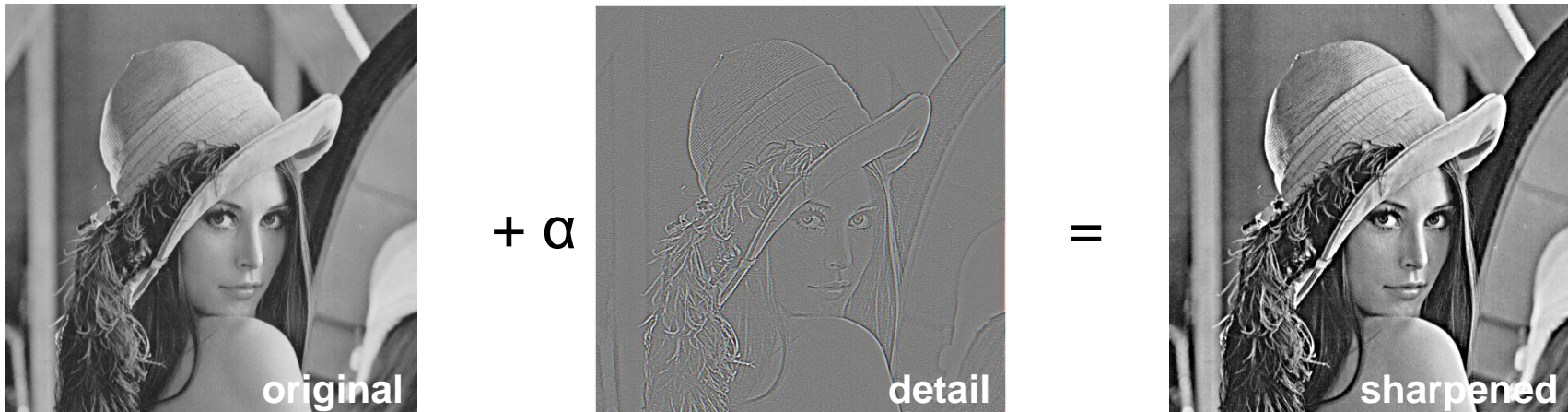


Sharpening revisited

- What does blurring take away?



Let's add it back:



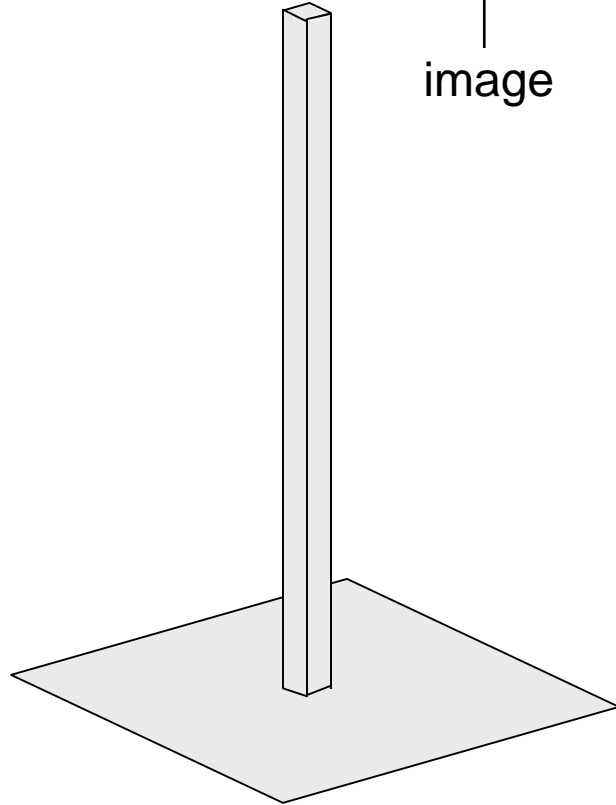
Unsharp mask filter

$$f + \alpha(f - f * g) = (1 + \alpha)f - \alpha f * g = f * ((1 + \alpha)e - g)$$

↑
image

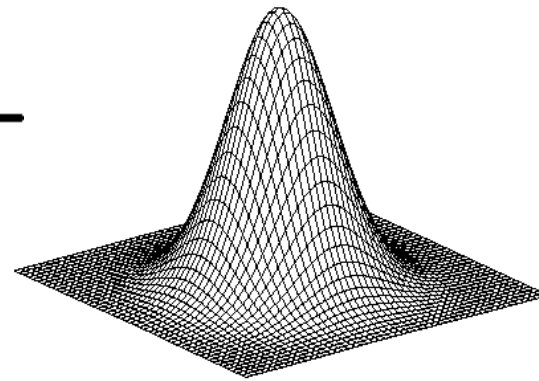
↑
blurred
image

↑
unit impulse
(identity)



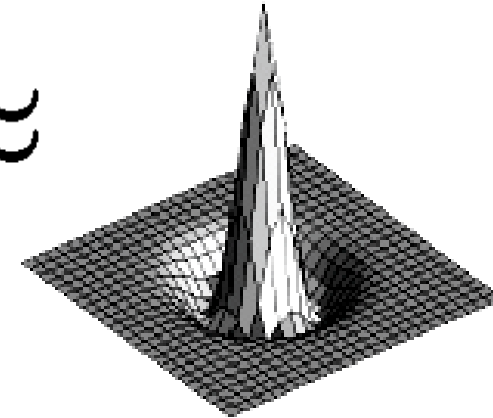
unit impulse

—



Gaussian

≈



Laplacian of Gaussian