

Computer Vision I:

Jingya Wang

Email: wangjingya@shanghaitech.edu.cn



Linear filtering

Types of Images

Binary



Gray Scale

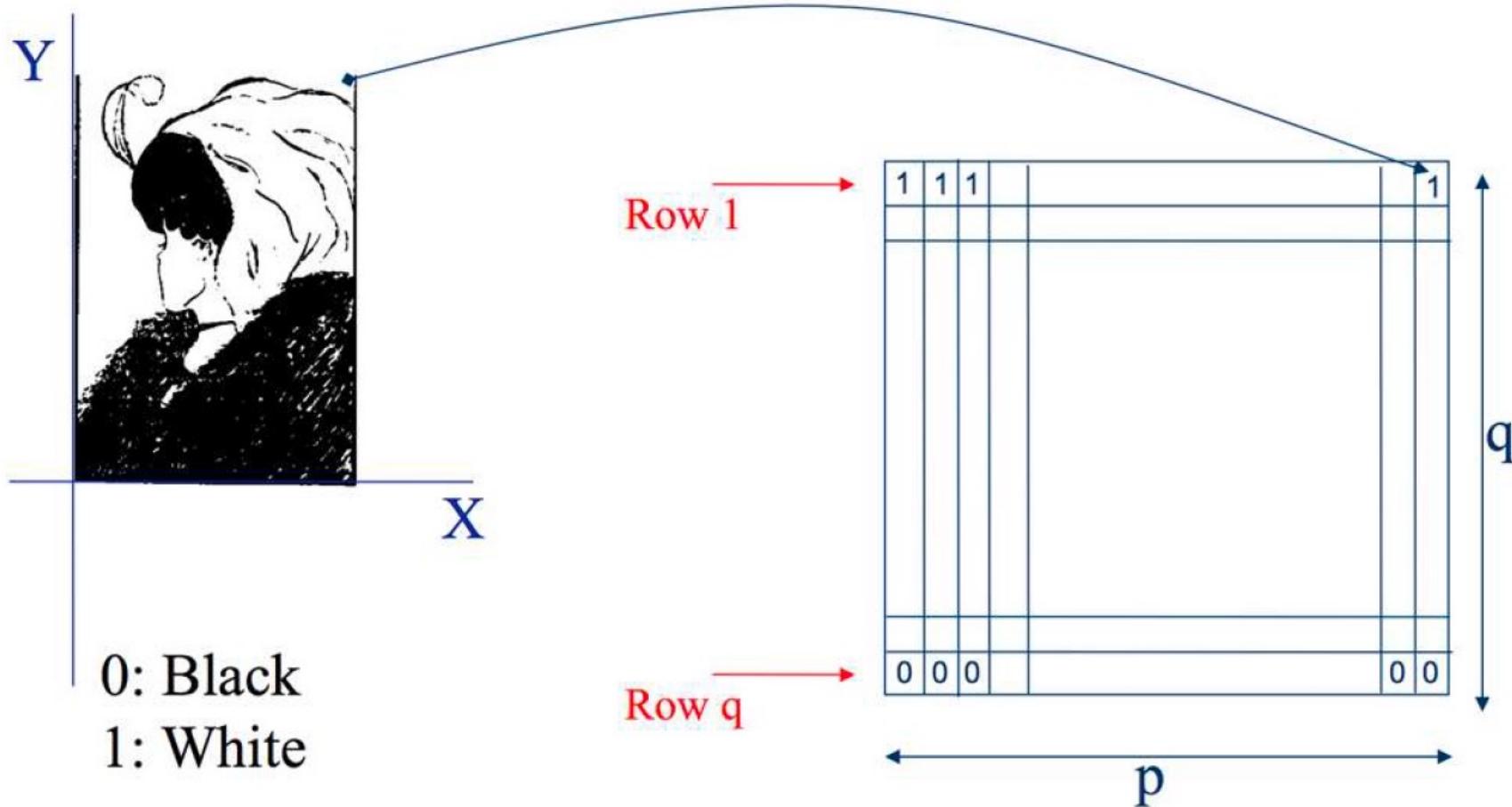


Color



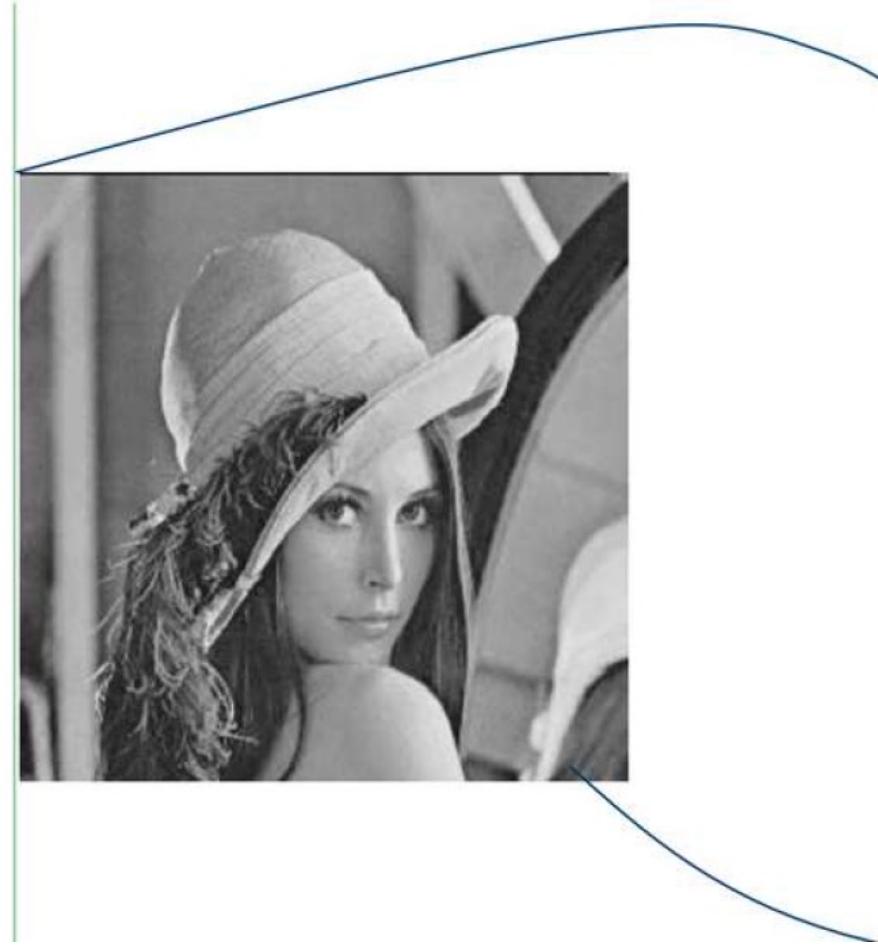
Source: Ulas Bagci

Binary image representation



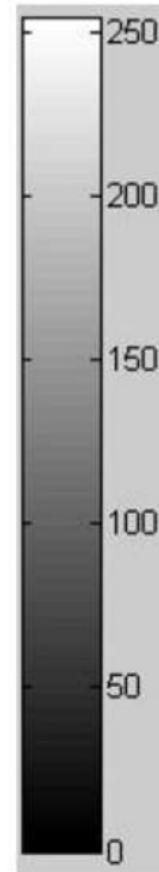
Source: Ulas Bagci

Grayscale image representation



10	5	9							

Slide credit:
Ulas Bagci



Source: Ulas Bagci

Color Image - one channel



Source: Ulas Bagci

Color image representation



Phil Noble / AP



Source: Ulas Bagci

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?

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

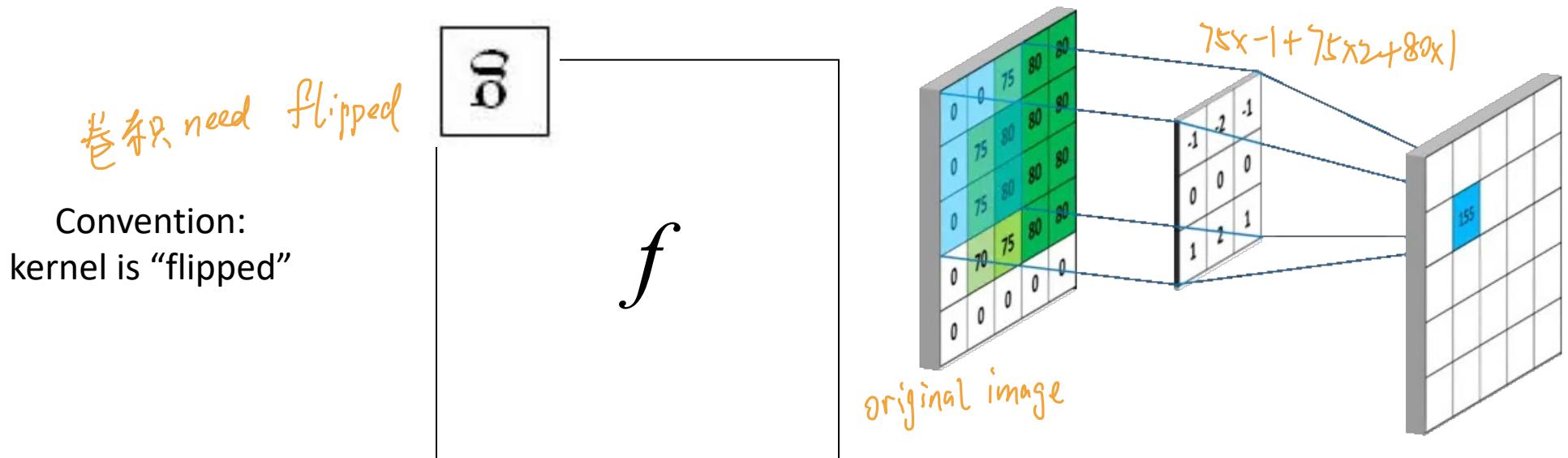
“box filter”

Defining convolution

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

k 奇数 $\Rightarrow k=1 \Rightarrow$ kernel 3×3
 $k=2 \Rightarrow$ kernel 5×5

$$(f * g)(x, y) = \sum_{i=-k}^k \sum_{j=-k}^k f(x - i, y - j) \cdot g(i, j)$$



About the kernel

- 1. The kernel size should be odd, then it has a kernel center and a kernel radius
- 2. The sum of all the elements in the kernel should be 1, otherwise,
 - >1 , the image becomes brighter
 - <1 , the image becomes darker
- 3. After convolution, some values may be greater than 255 or less than 0, let the values be $\min(\text{value}, 255)$ or $\max(\text{value}, 0)$

Key properties

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

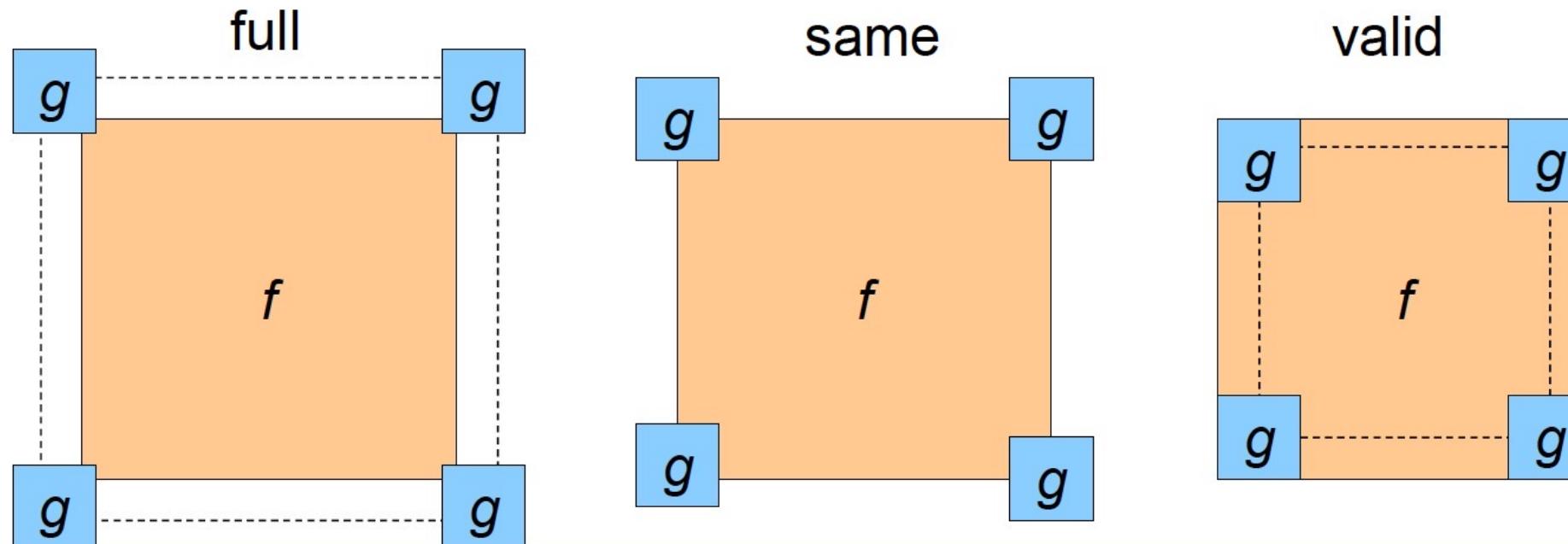
Properties in more detail

- 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 $e = [..., 0, 0, 1, 0, 0, ...]$,
 $a * e = a$
恒等, 单位脉冲

Annoying details

What is the size of the output?

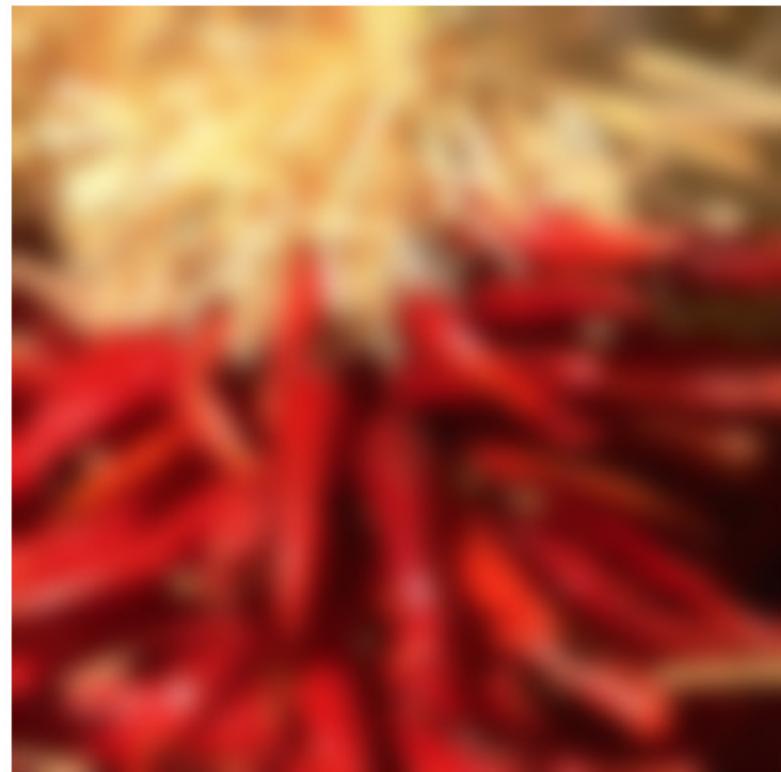
- MATLAB: `filter2(g, f, shape)`
 - `shape = 'full'`: output size is sum of sizes of f and g
 - `shape = 'same'`: output size is same as f
 - `shape = 'valid'`: output size is difference of sizes of f and g



Source: S. Lazebnik

Dealing with edges

- What about missing pixel values?
 - 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

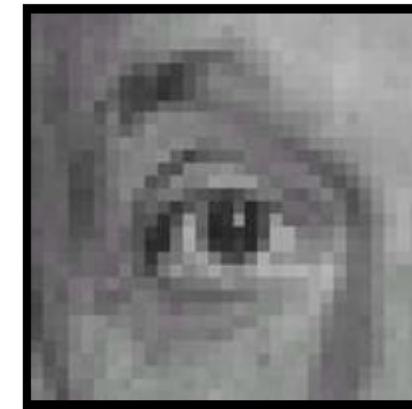
?

Practice with linear filters



Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters



0	0	0
0	0	1
0	0	0

?

Original

Practice with linear filters



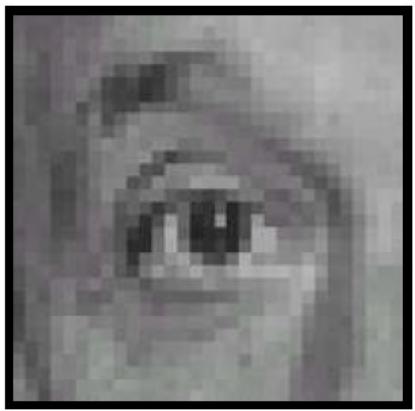
Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Practice with linear filters



$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

?

Original

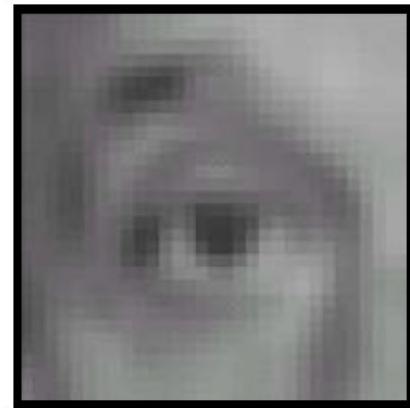
Practice with linear filters



Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

变 smooth



Blur (with a
box filter)

Practice with linear filters



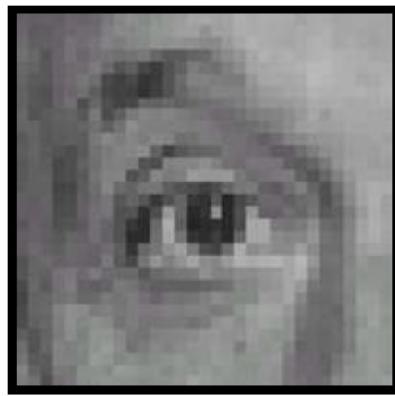
色差加重

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

?

(Note that filter sums to 1)

Practice with linear filters



Original

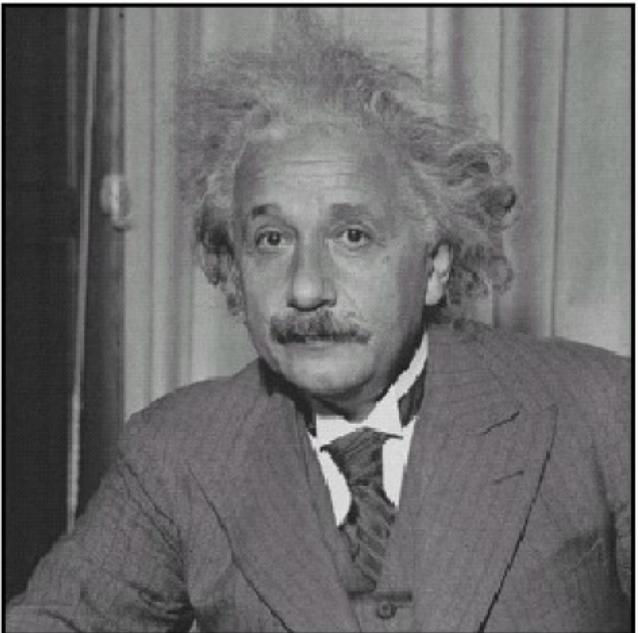
$$\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix}$$

$$- \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

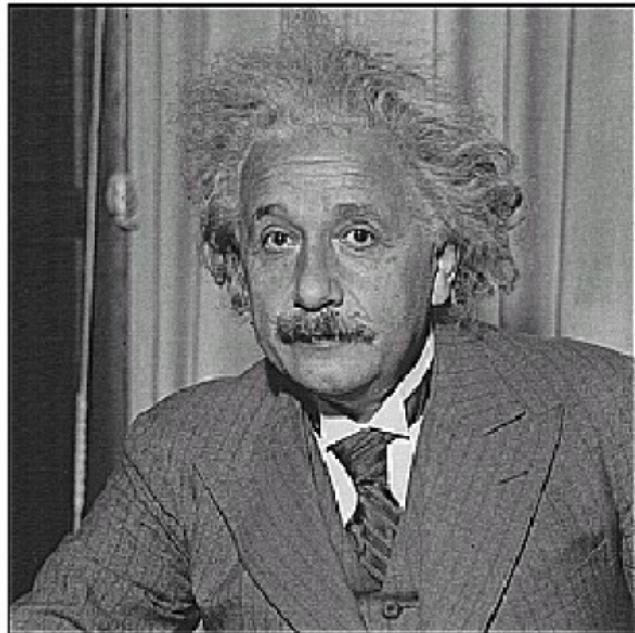


Sharpening filter 
- Accentuates differences
with local average

Sharpening



before



after

Source: D. Lowe

Sharpening

- What does blurring take away?

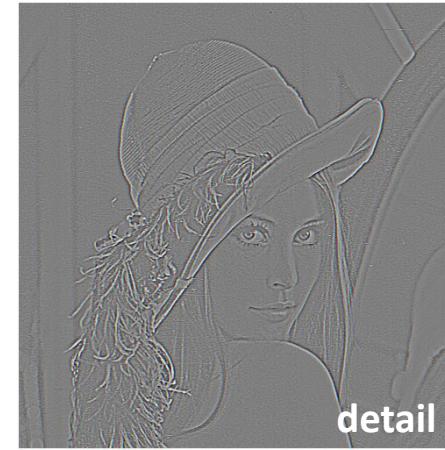


-



smoothed (5x5)

=

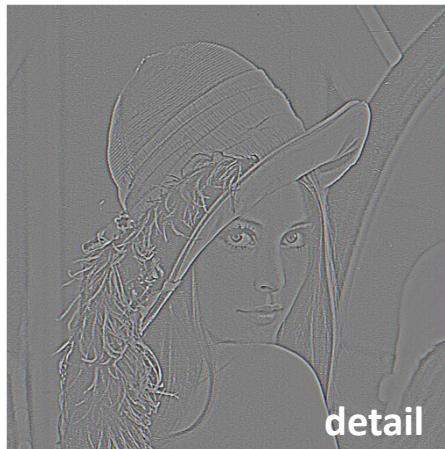


detail

Let's add it back:



+



detail

=



sharpened

Smoothing with box filter revisited

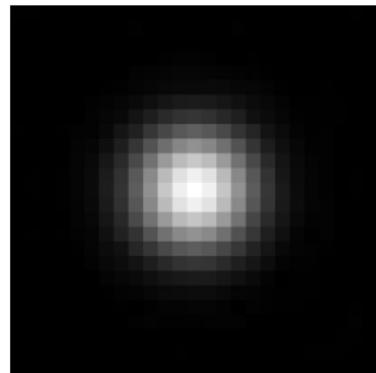
- What's wrong with this picture?
- What's the solution?



Smoothing with box filter revisited

温和且保留边界特征

- 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



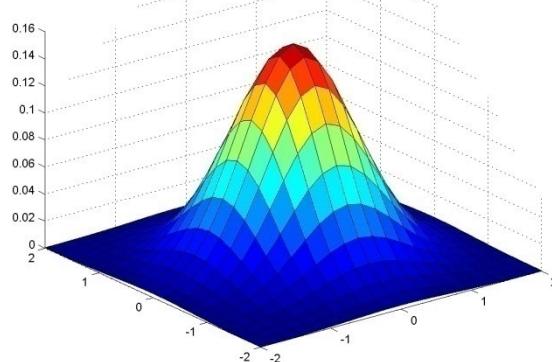
“fuzzy blob”

Source: S. Lazebnik

Gaussian Kernel

$r \Rightarrow$ 滤波器宽度

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



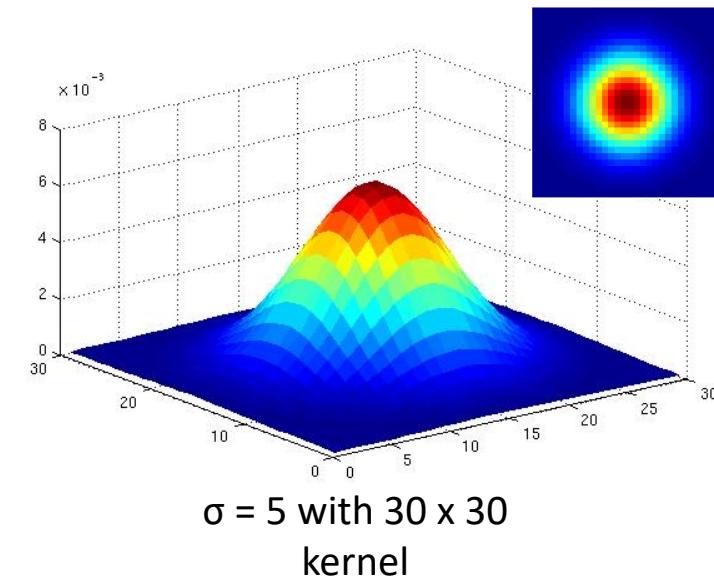
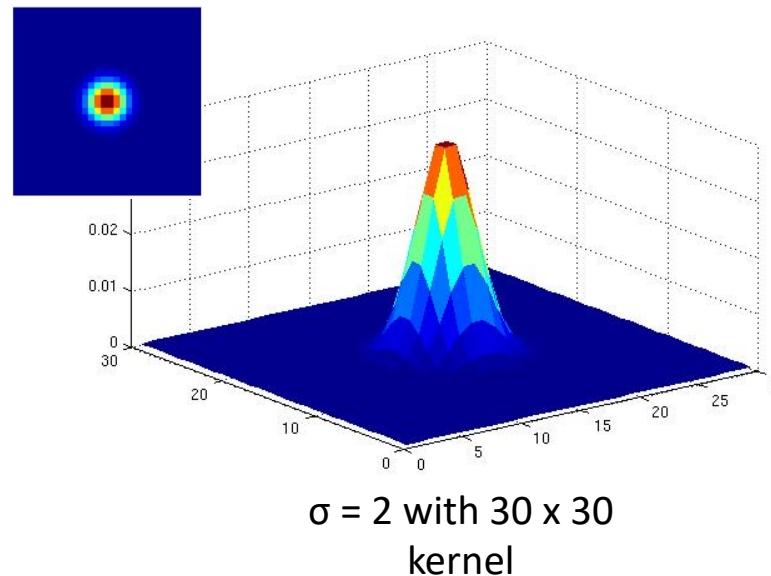
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$5 \times 5, \sigma = 1$

- 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

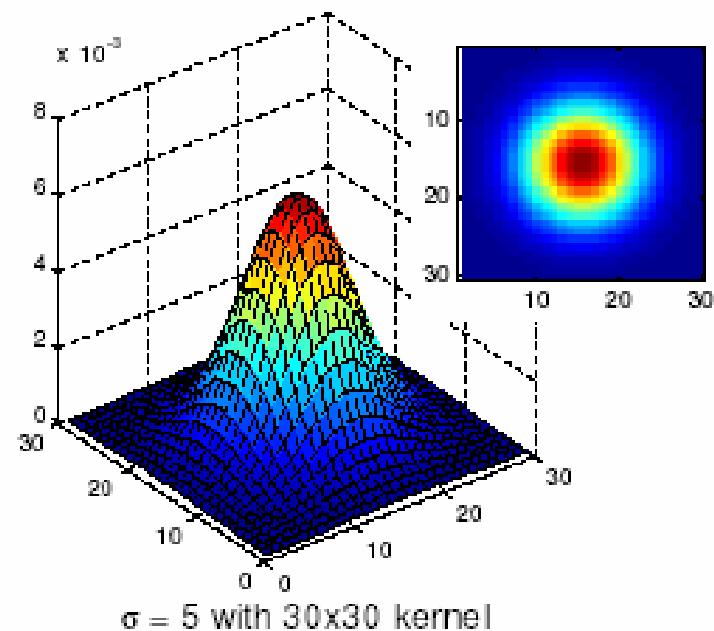
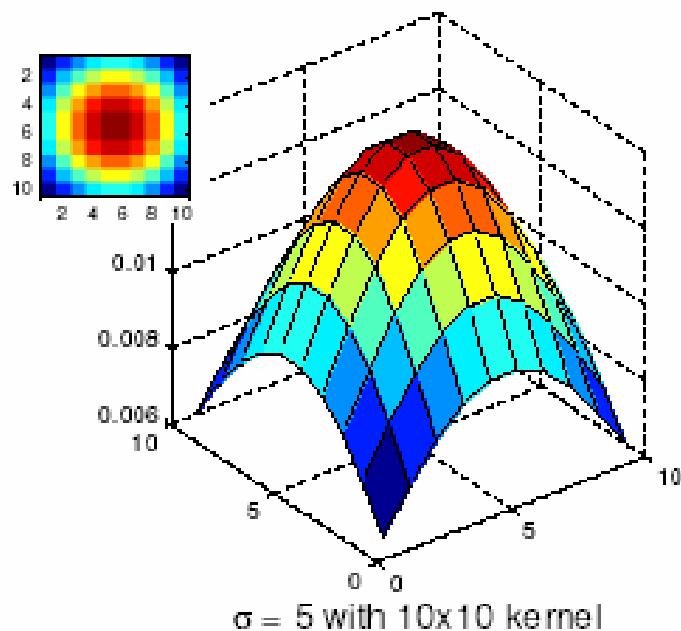
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



- Standard deviation σ : determines extent of smoothing

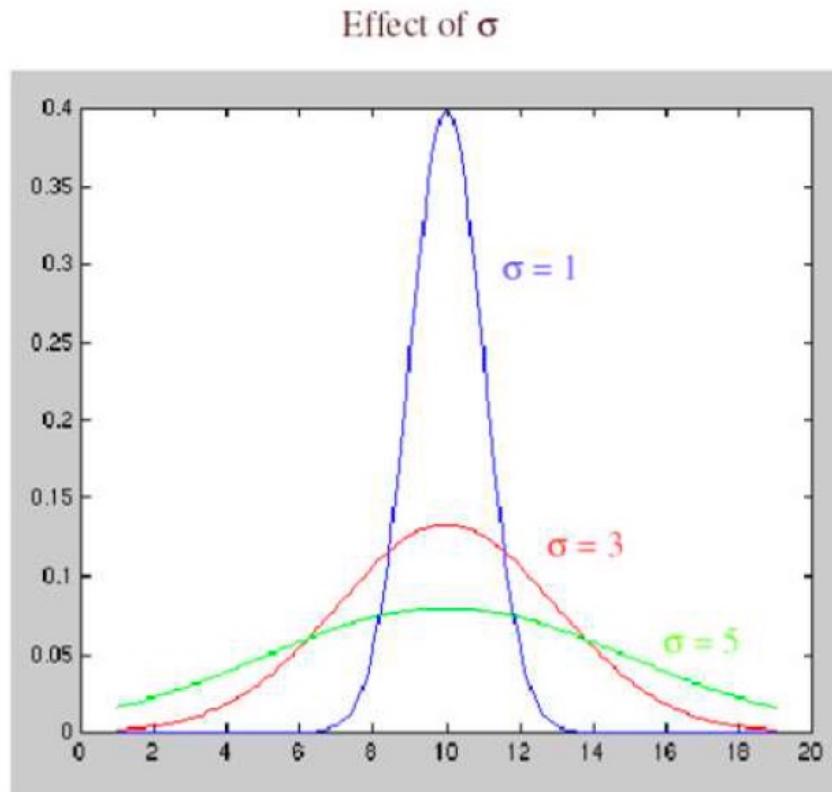
Choosing kernel width

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

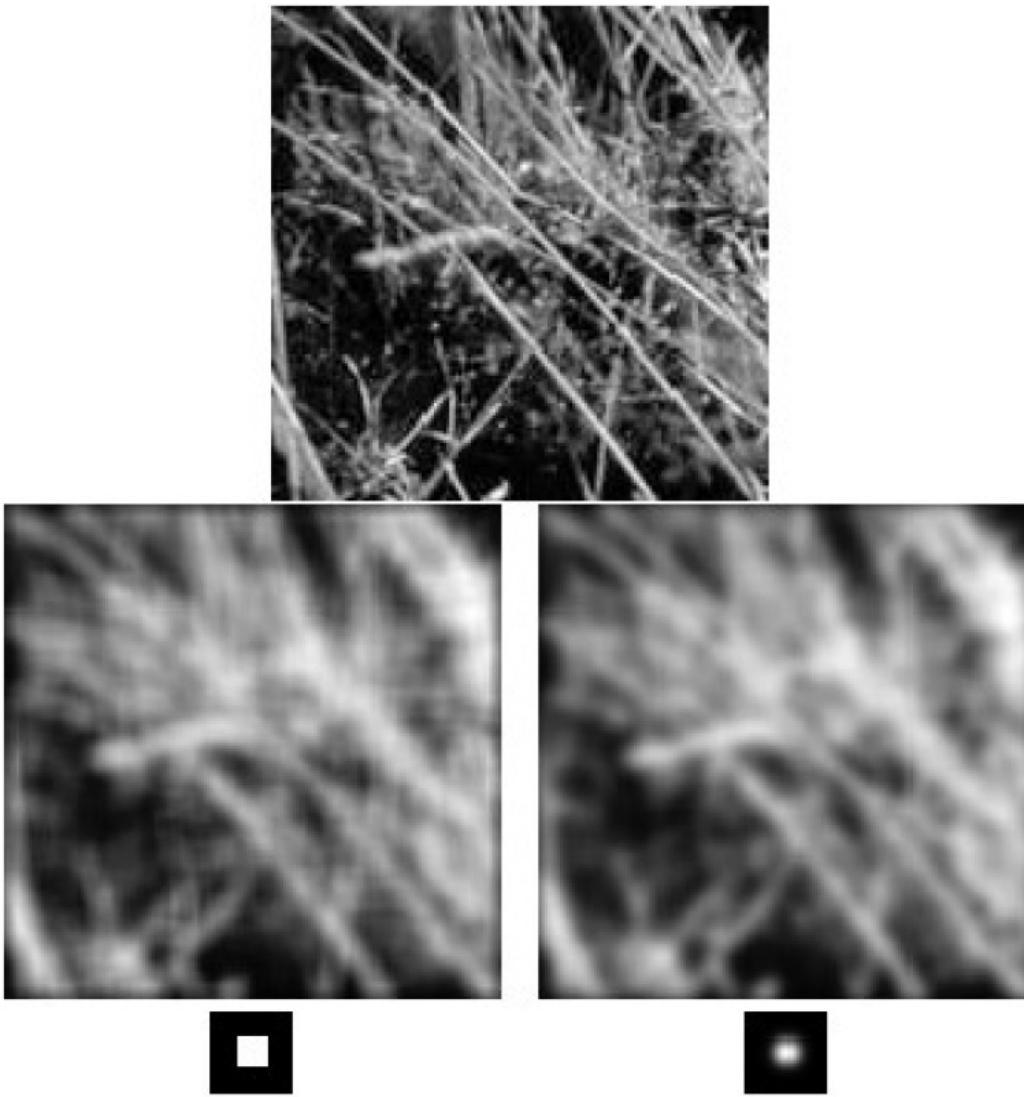


Choosing kernel width

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



Gaussian vs. box filtering



Source: S. Lazebnik

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
 - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
 - Factors into product of two 1D Gaussians

Separability example

2D convolution
(center location only)

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} = \begin{matrix} 1 \\ 2 \\ 1 \end{matrix} \times \begin{matrix} 1 & 2 & 1 \end{matrix}$$

Perform convolution
along rows:

$$\begin{matrix} 1 & 2 & 1 \end{matrix} * \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix} =$$

Followed by convolution
along the remaining column:

Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among 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



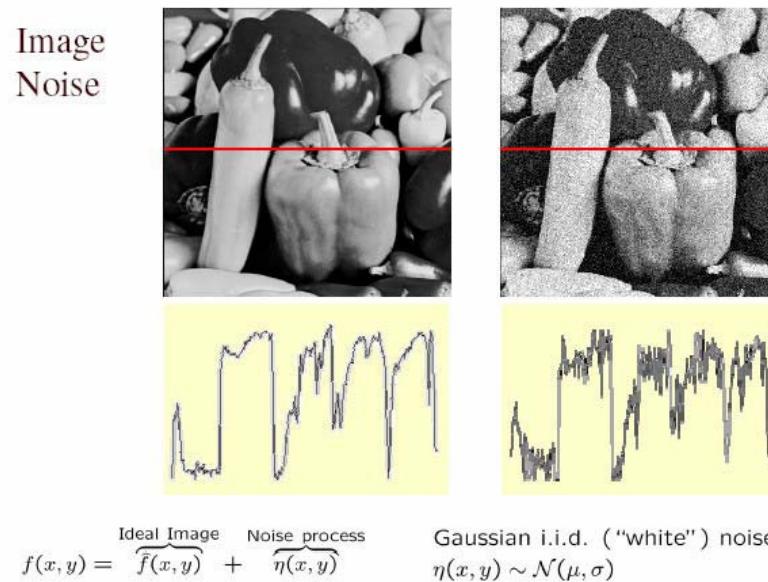
Gaussian noise

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

Gaussian noise

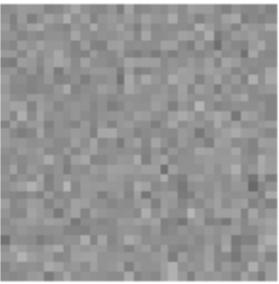
original input + noise

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

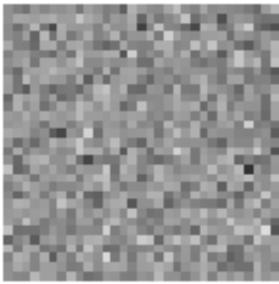


Reducing Gaussian noise

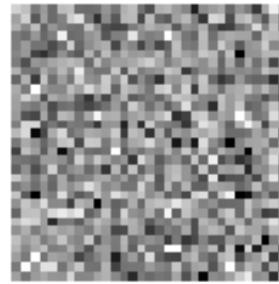
$\sigma=0.05$



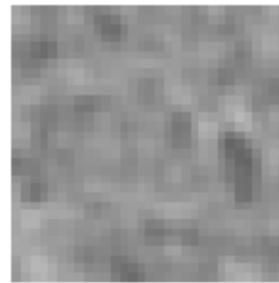
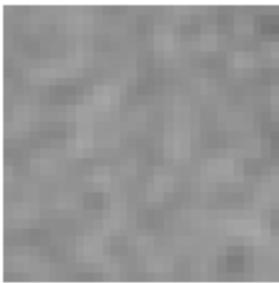
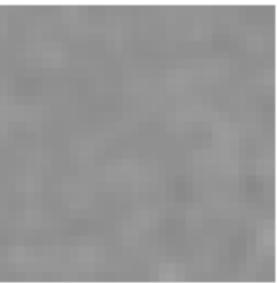
$\sigma=0.1$



$\sigma=0.2$



no
smoothing



$\sigma=1$ pixel



$\sigma=2$ pixels



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?

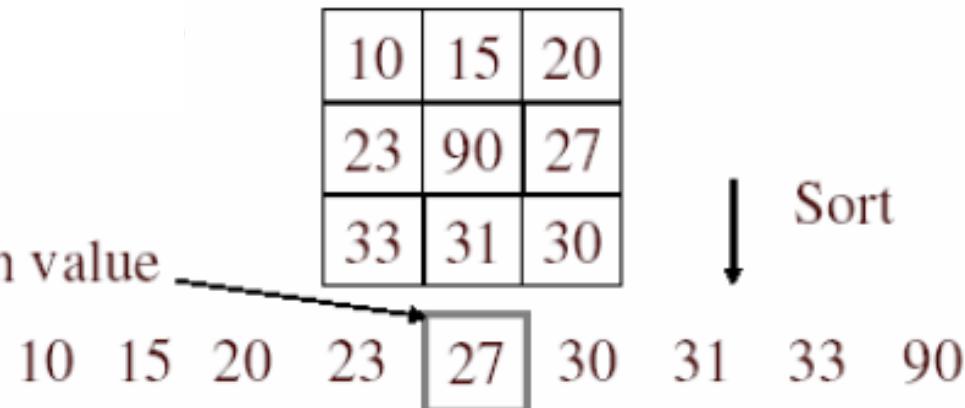
Alternative idea: Median filtering

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

中值滤波器

用周边 neighbour 中值 替代

Median value

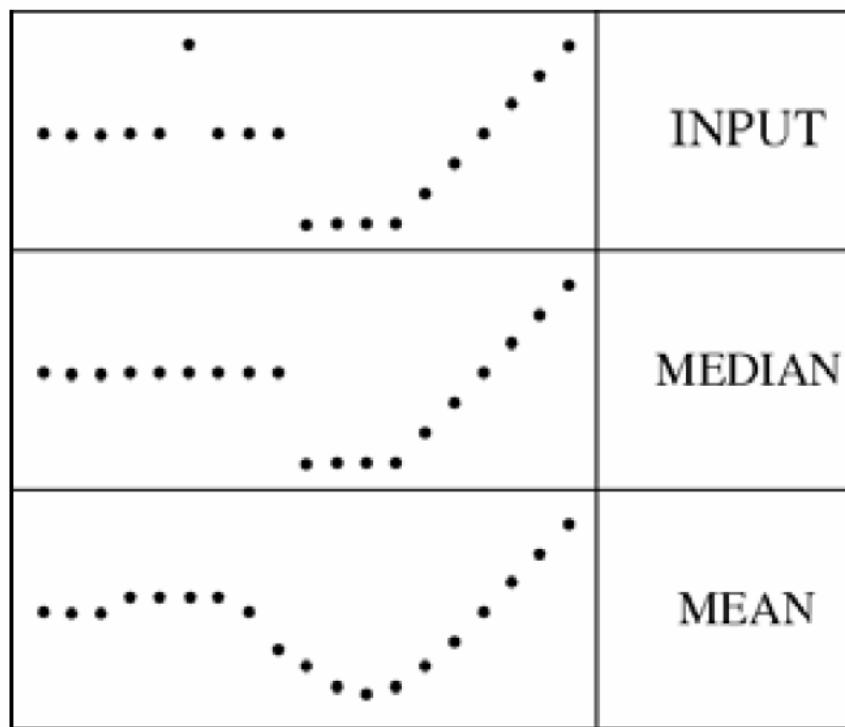


- Is median filtering linear?

Median filter

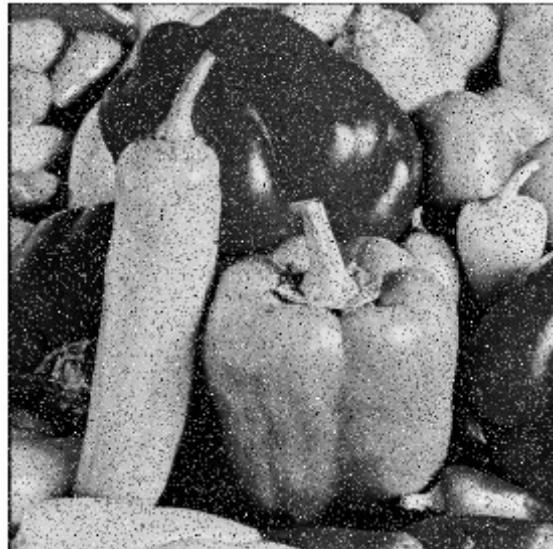
- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :

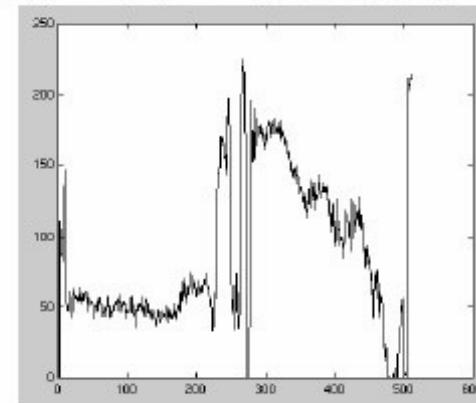
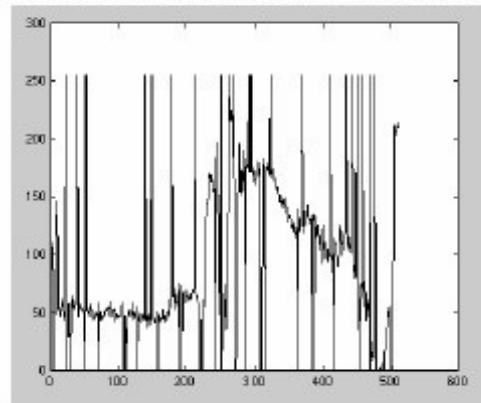


Median filter

Salt-and-pepper noise



Median filtered



- MATLAB: `medfilt2(image, [h w])`

Gaussian vs. median filtering

Gaussian



5x5



7x7



Median



Review: Image filtering

- Convolution
- Image smoothing
- Gaussian filter
- Nonlinear filtering

Application: Hybrid Images



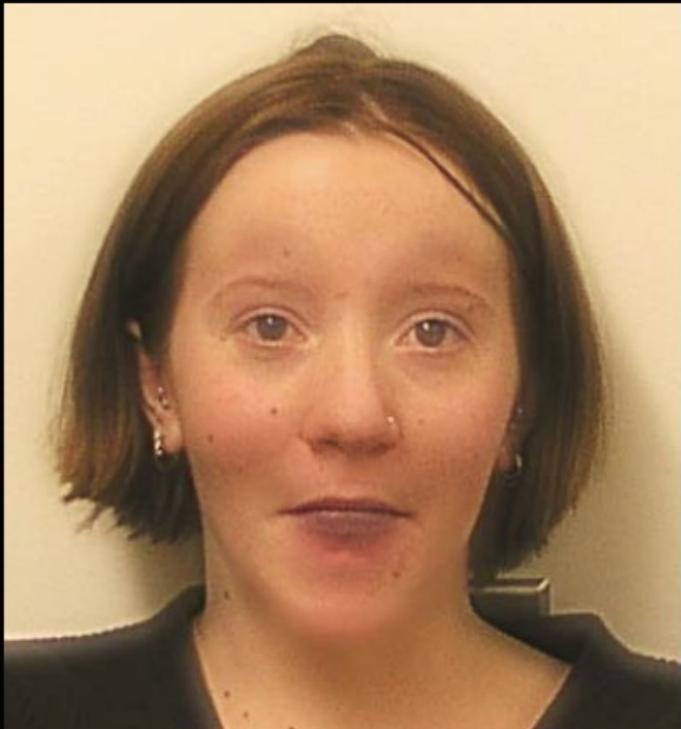
- A. Oliva, A. Torralba, P.G. Schyns,
[“Hybrid Images,”](#) SIGGRAPH 2006

Changing expression



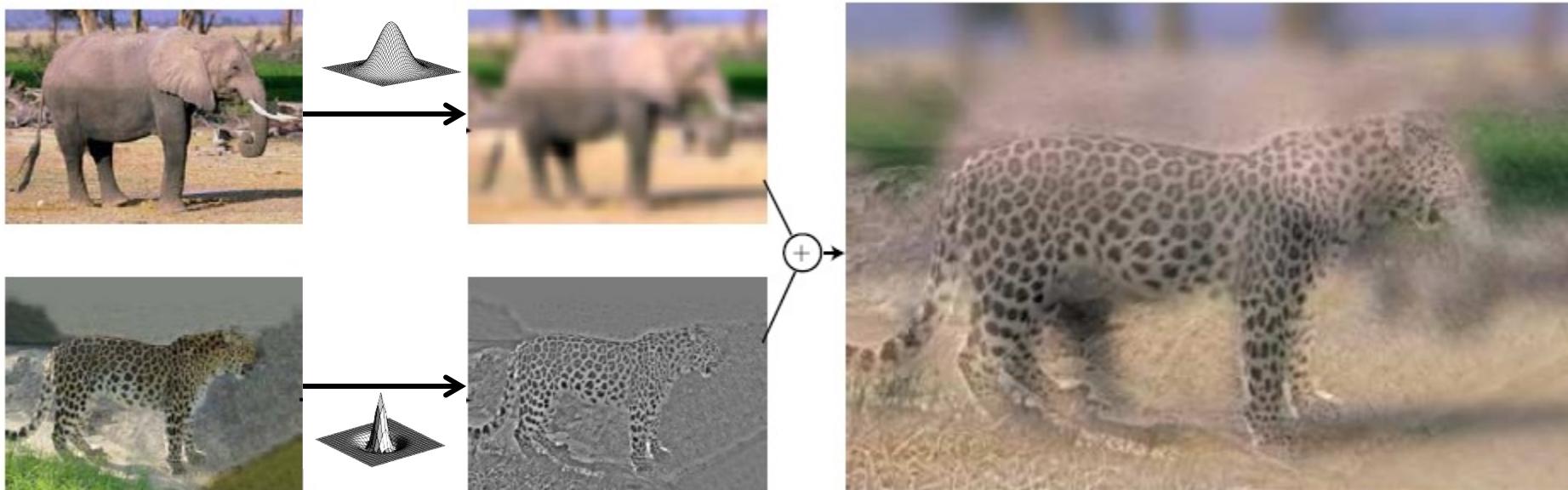
Sad

Surprised



Application: Hybrid Images

Gaussian Filter



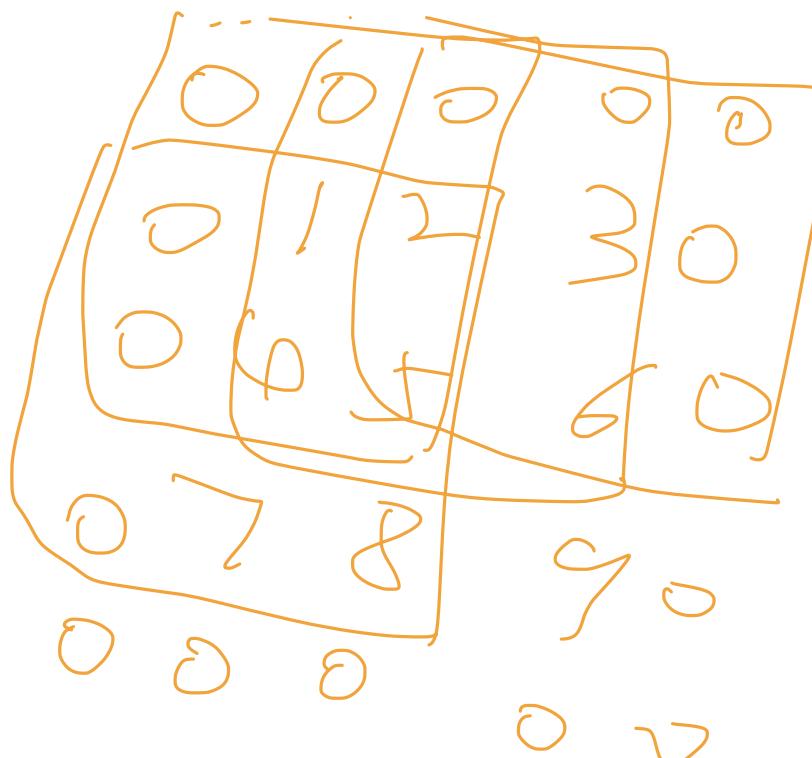
Laplacian Filter

- A. Oliva, A. Torralba, P.G. Schyns,
"Hybrid Images," SIGGRAPH 2006

Exercise

- $A = [1 \ 2 \ 3$
- $\quad \quad 4 \ 5 \ 6$
- $\quad \quad 7 \ 8 \ 9]$

- $B = [1 \quad 2 \quad 1$
- $\quad \quad 0 \quad 0 \quad 0$
- $\quad \quad -1 \quad -2 \quad -1]$
- $A^*B = ?$



- Thank you!