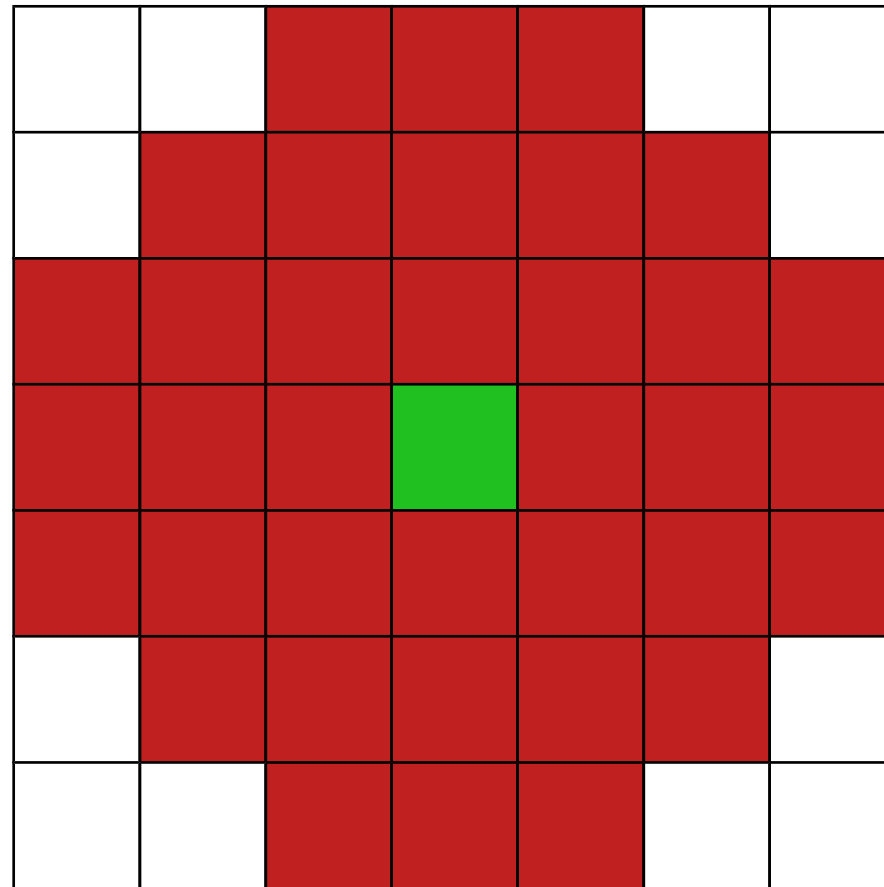
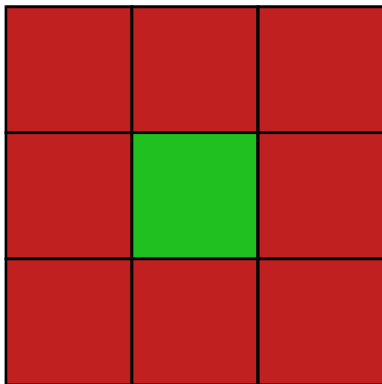
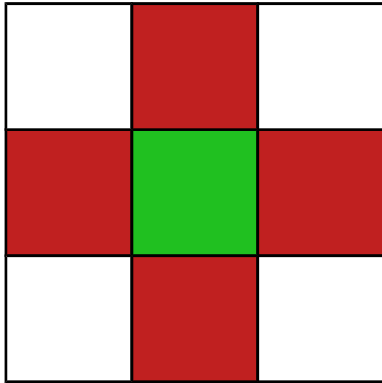


Today's lecture

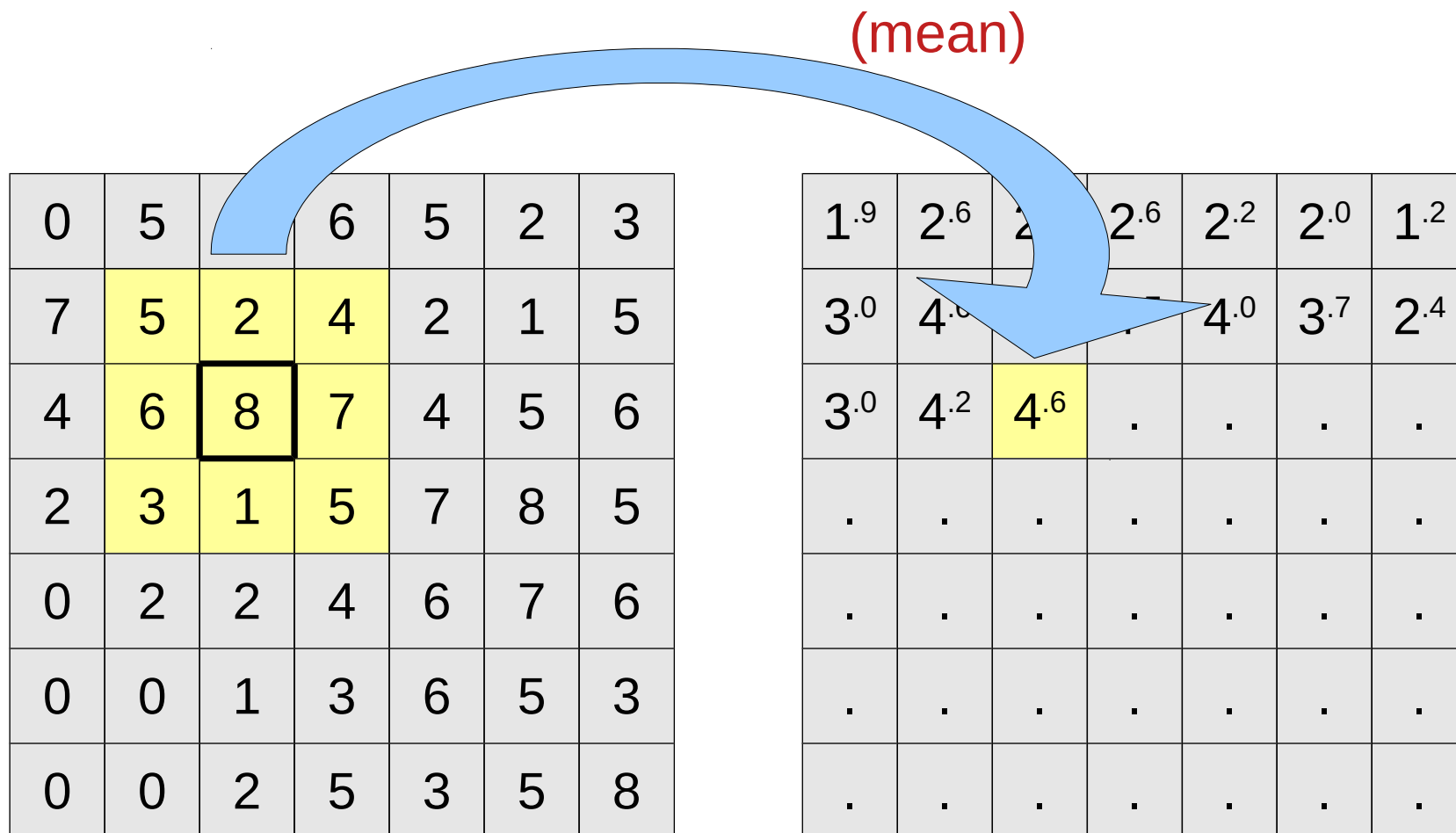
- Local neighbourhood processing
 - smoothing an image
 - sharpening an image
 - And more...
- Convolution
 - What is it?
 - What is it useful for?
 - How can I compute it?
- Removing uncorrelated noise from an image
- Mathematical morphology, filtering/modifying binary shapes in an image.

Neighbourhoods



Local neighbourhood operation

- For each pixel, examine its neighbourhood and compute an output value.



Local neighbourhood operation

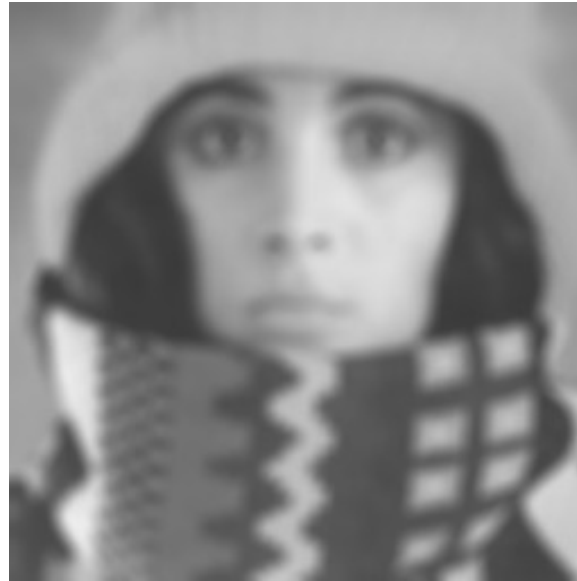
- Possible operations to do for each neighbourhood:
 - average (mean, median, etc.)
 - weighted average
 - other statistics (variance, maximum, etc.)
 - difference (to compute derivative)
 - ...
- Neighbourhood size and shape is very important
 - round neighbourhood gives rotation invariance
- Adaptive filtering:
 - changing size, shape and/or operation depending on local image properties

Smoothing an image

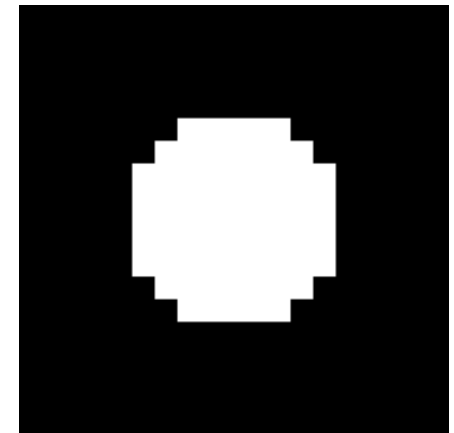
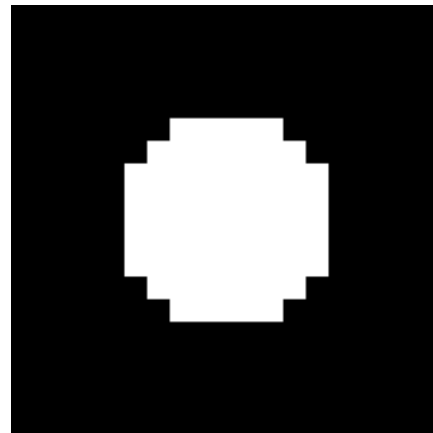
Input image



mean filter



median filter

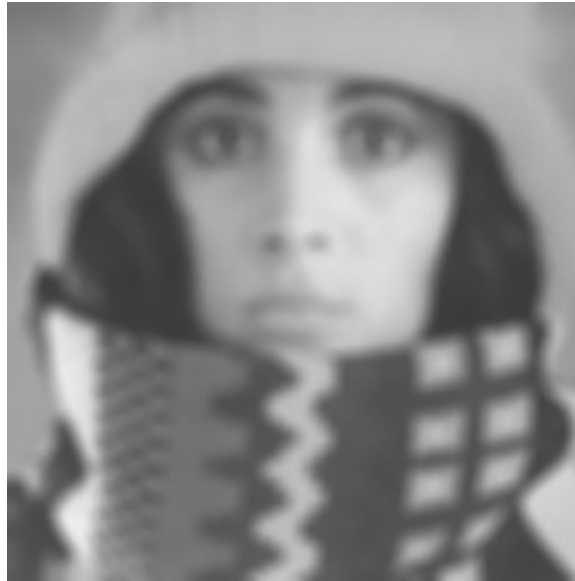


Smoothing an image

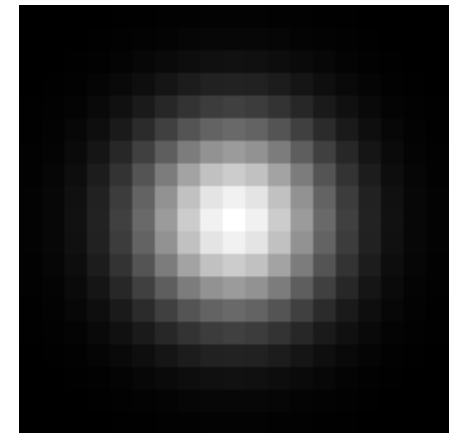
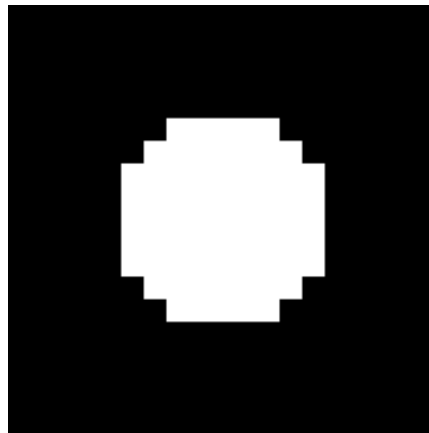
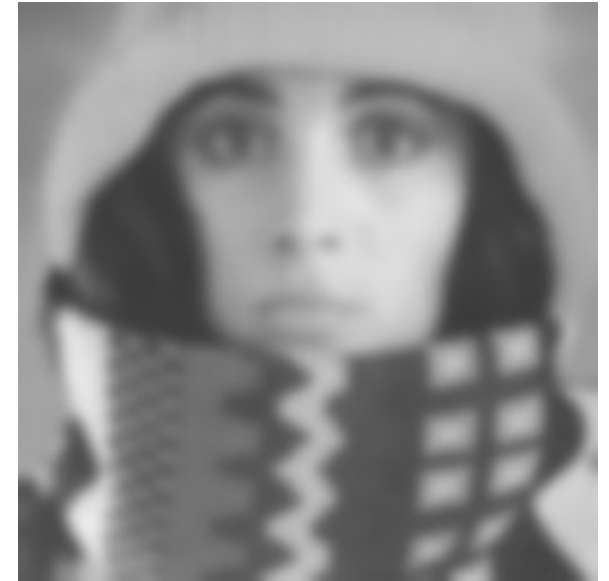
Input image



mean filter



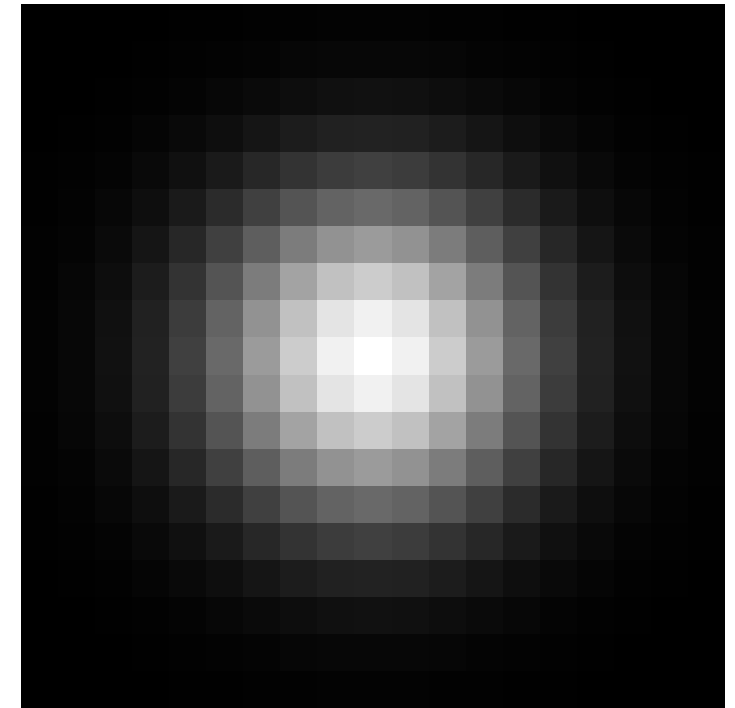
weighted mean filter



How to define Gaussian weights

- σ determines the amount of smoothing
- the neighbourhood size should be large enough to contain the whole Gaussian bell!
rule of thumb: $2 \text{ceil}(3\sigma) + 1$
- sum of all weights normalised to 1

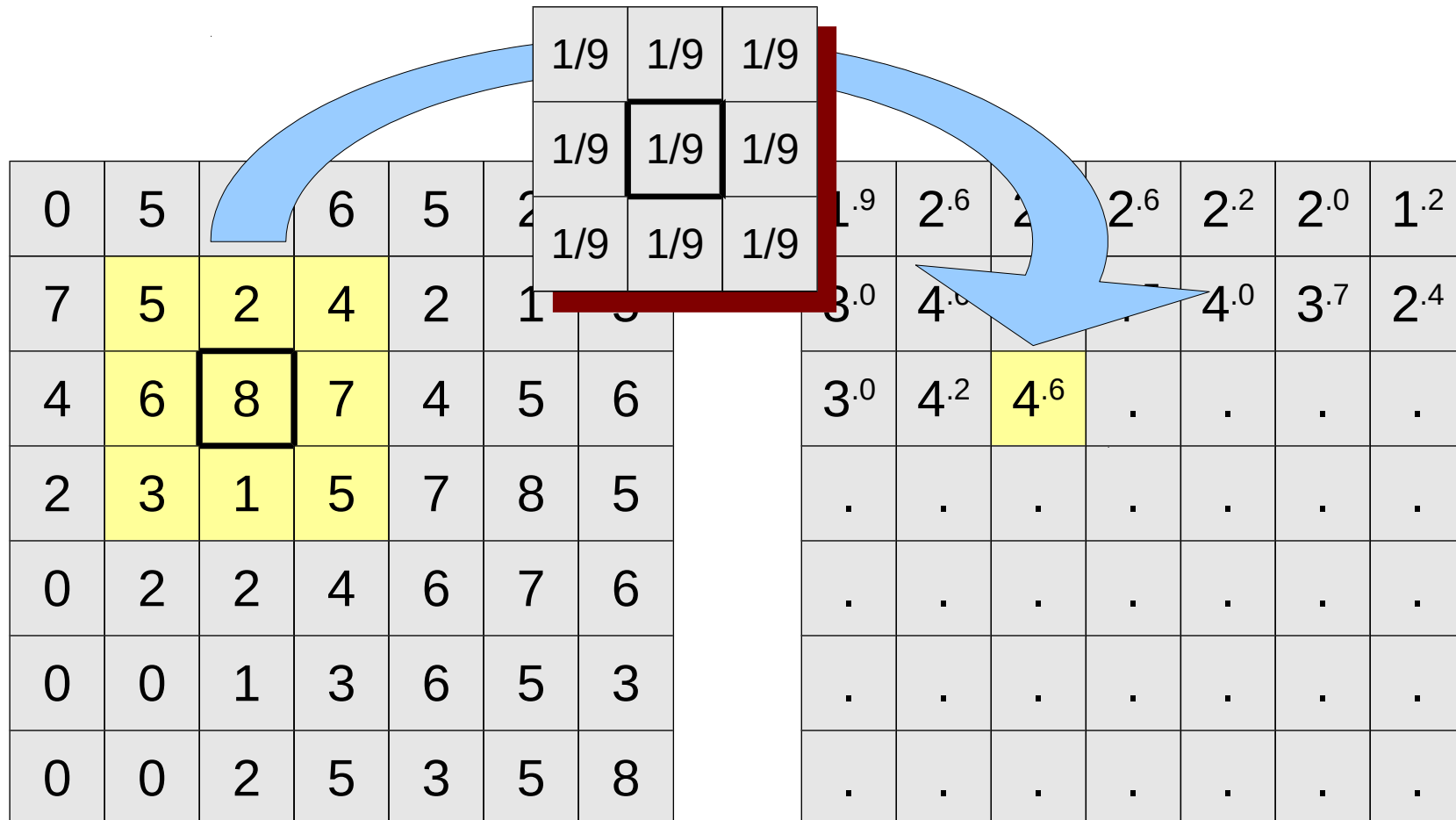
$$\frac{1}{2\pi\sigma^2} \exp\left(\frac{-x^2 + y^2}{2\sigma^2}\right)$$



$2 \text{ceil}(3\sigma) + 1$

Weighted mean filter

- For each pixel, multiply the values in its neighbourhood with the corresponding weights, then sum.



Applications?

Write down as many applications of a smoothing filter as you can come up with.

Application: noise reduction

input image

3x3 mean filter

3x3 median filter

Normally
distributed
noise

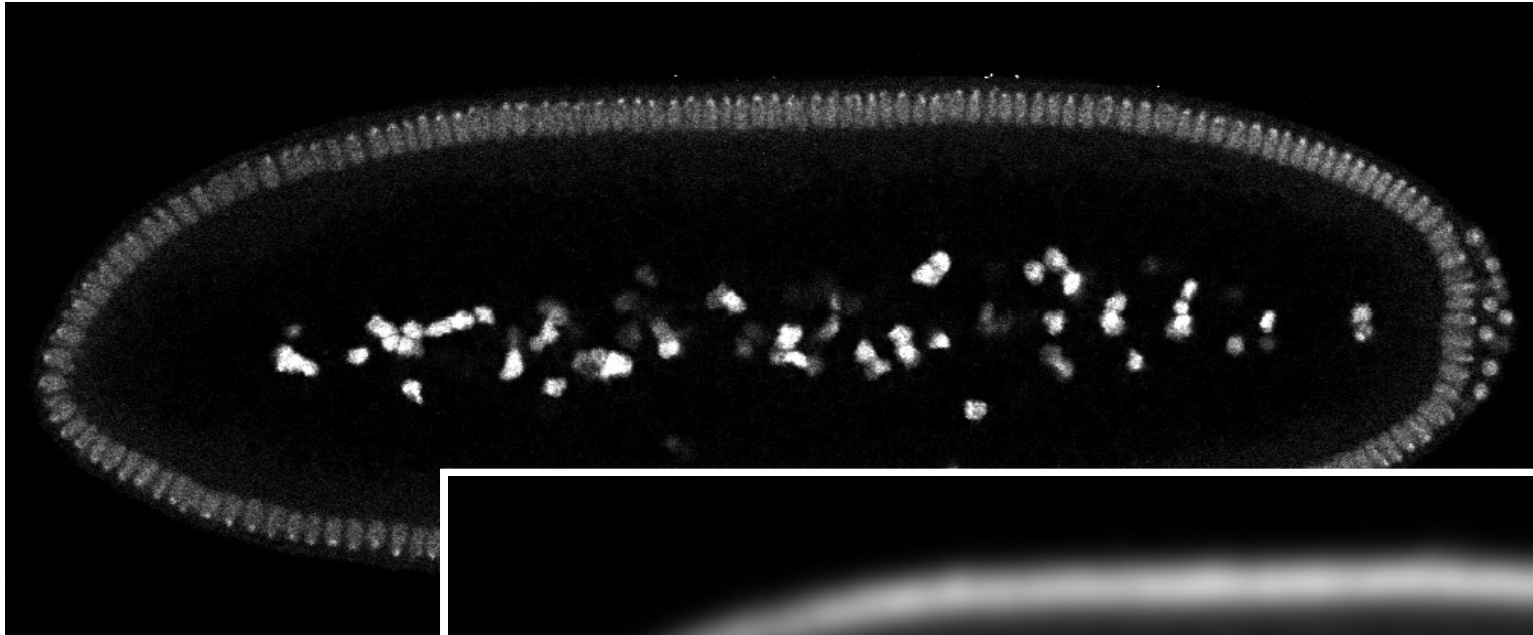


Salt &
pepper
noise

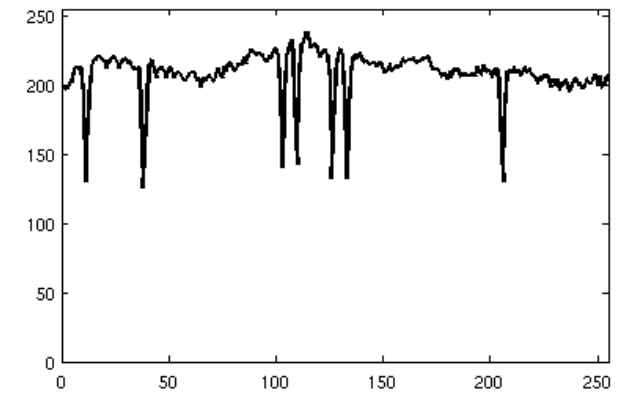
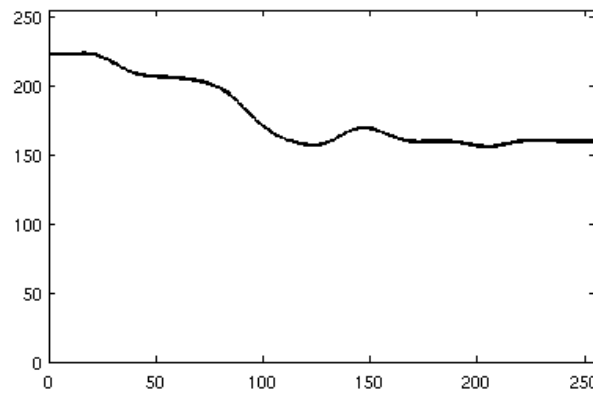
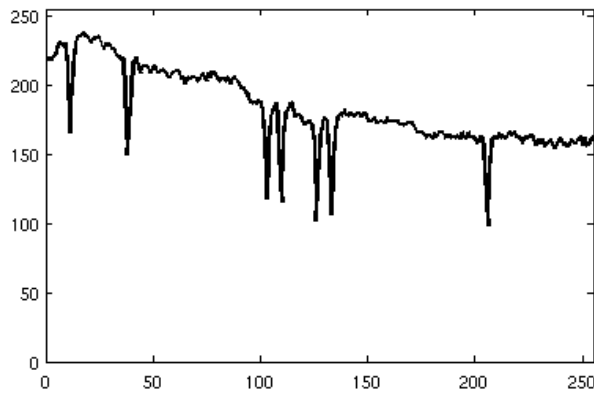
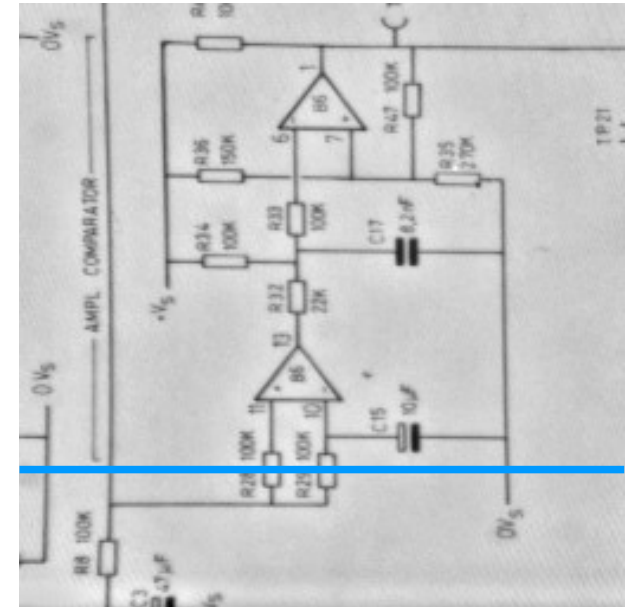
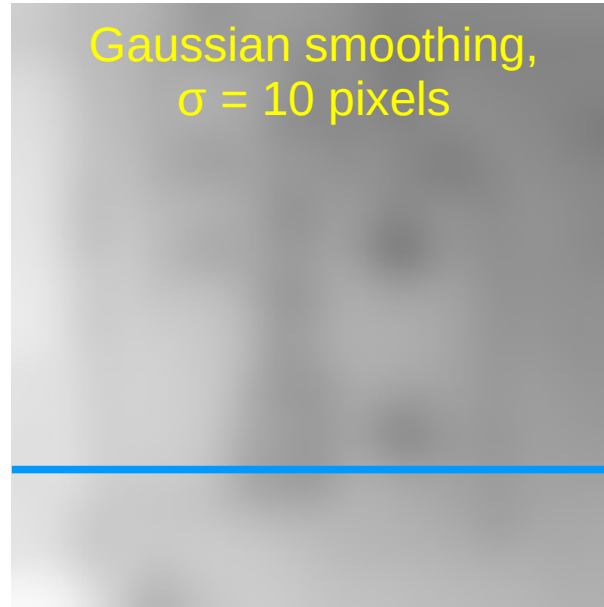
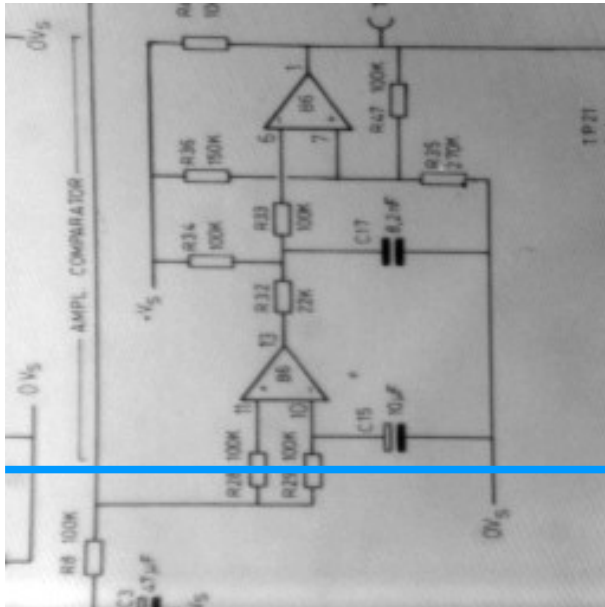


Application: abstraction

Sometimes you just don't want all those details...



Application: shading correction



Sharpening an image

“Unsharp masking”



original



smoothed
(3x3)



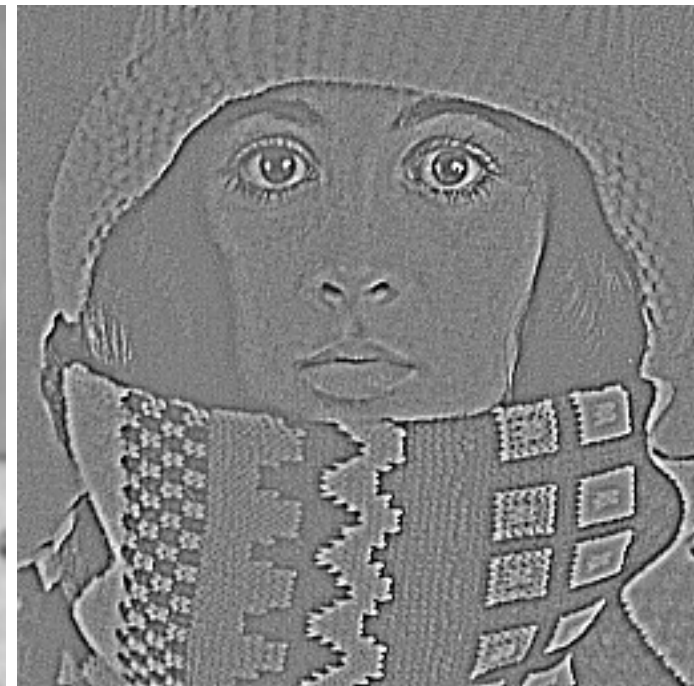
sharpened
($\alpha = 9$)

$$\text{sharpened} = (1+\alpha) \cdot \text{original} - \alpha \cdot \text{smoothed}$$

Sharpening an image

$$\text{sharpened} = (1+\alpha) \cdot \text{original} - \alpha \cdot \text{smoothed}$$

$$\text{sharpened} = \text{original} + \alpha \cdot (\text{original} - \text{smoothed})$$



0	0	0
0	1	0
0	0	0

original

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

smoothed

-1	-1	-1
-1	8	-1
-1	-1	-1

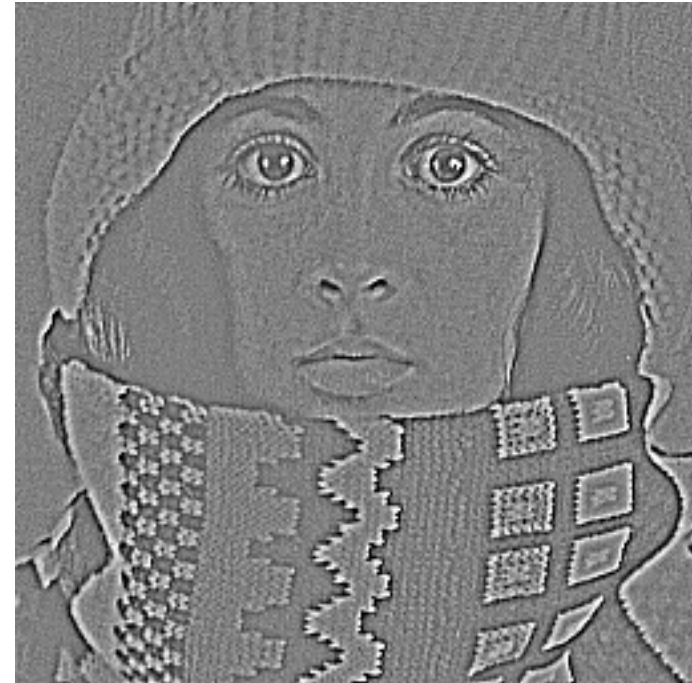
9 · diff.

Laplace filter

Laplace operator: $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

0	1	0
1	-4	1
0	1	0

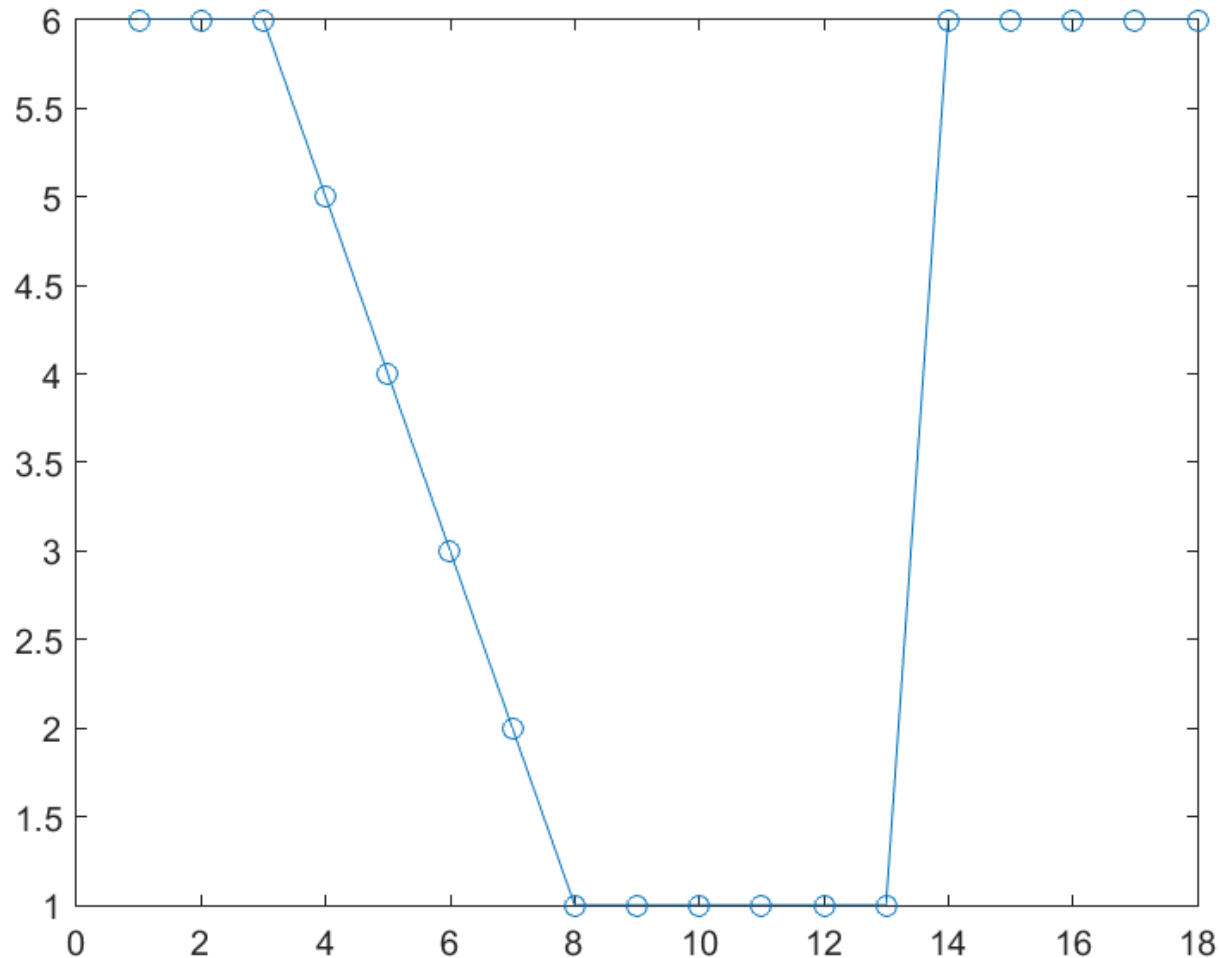
1	1	1
1	-8	1
1	1	1



sharpened = original + 9 · (original – smoothed)

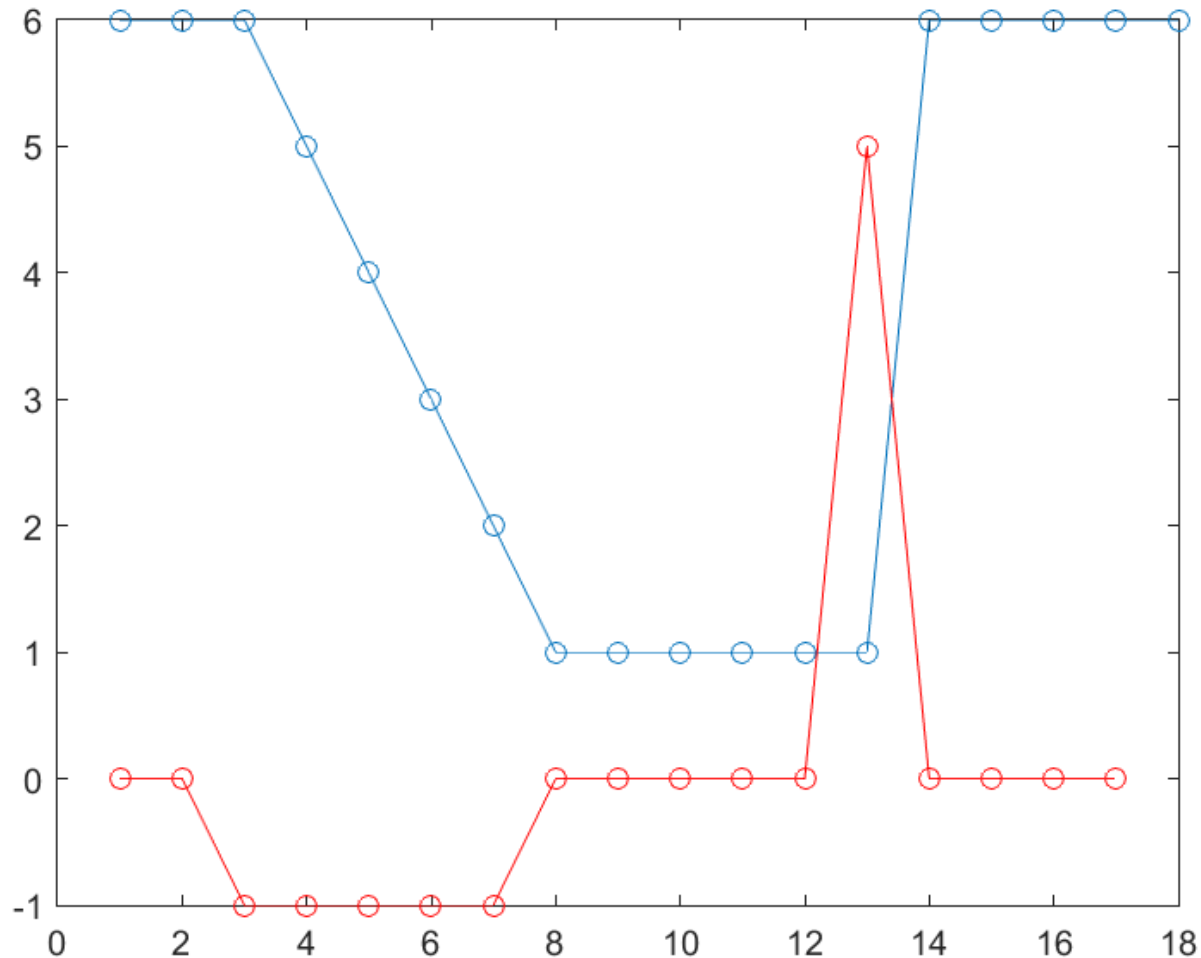
sharpened = original - Laplace

Approximating derivatives



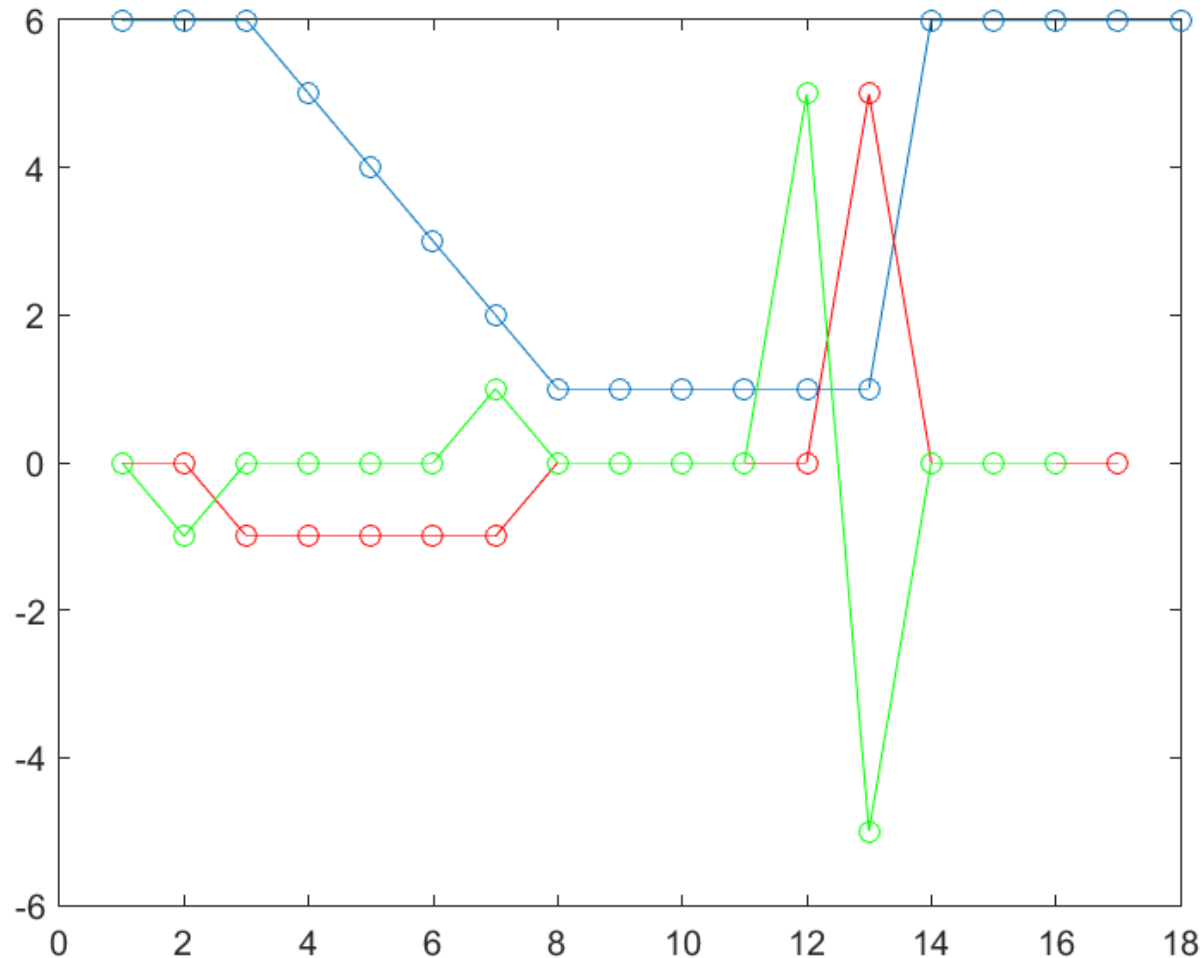
A discrete function, 1D

Approximating derivatives



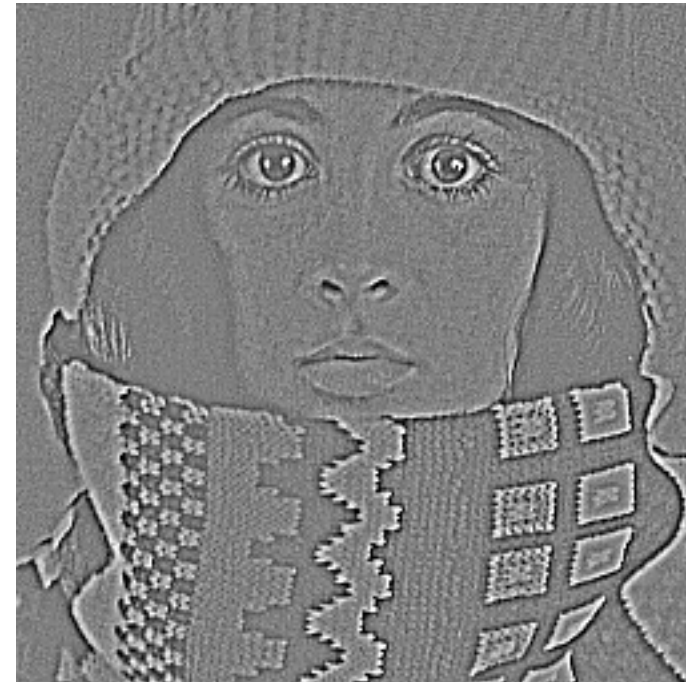
1st derivative by local differences

Approximating derivatives



2nd derivative by local differences

Laplace filter



Laplace operator: $\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

Sobel filter

Approximates the first derivatives: $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$



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

S_x

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

S_y

Detecting edges

Approximates the gradient magnitude: $\sqrt{\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2}$



$$\text{sqrt} (S_x^2 + S_y^2)$$

Adaptive filtering

- Many non-linear filters are meant to reduce noise without blurring the edges.
- One common technique is to “adapt” the kernel so that it does not extend across any edges.
- The bilateral filter is the most common one.

input image



median filter



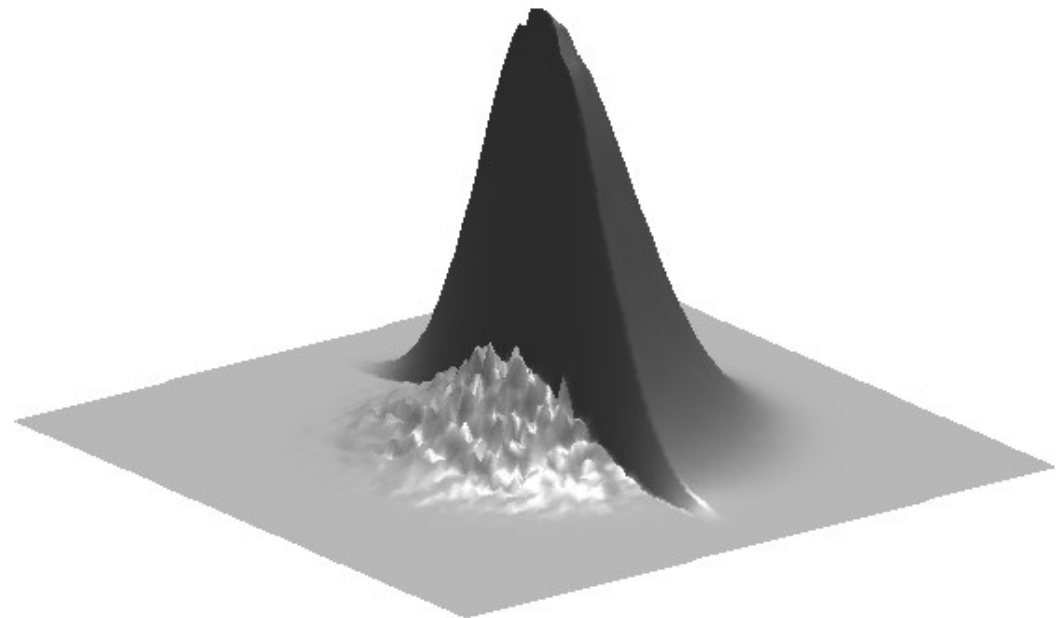
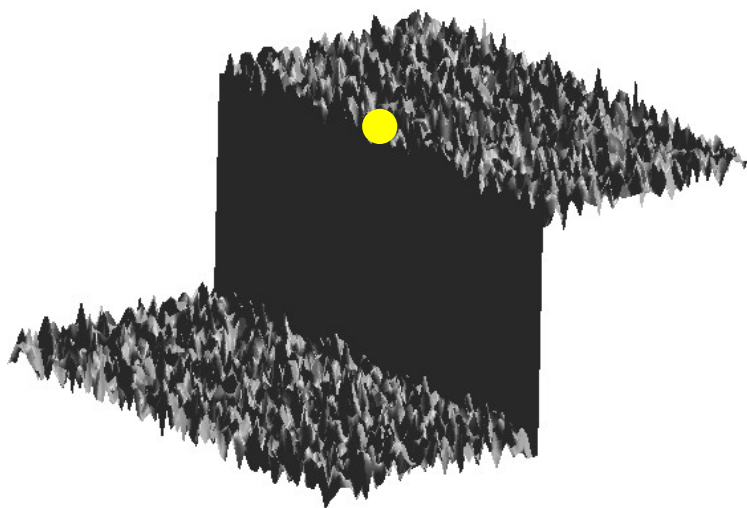
bilateral filter



Bilateral filter

- A new kernel is designed for each output pixel.
- Kernel weights are reduced if the corresponding pixel in the input image has a large difference in intensity with the central pixel.

$$h_{\vec{x}_0}(\vec{x}) = G_{\sigma_x}(\vec{x} - \vec{x}_0) G_{\sigma_f}(f(\vec{x}) - f(\vec{x}_0))$$



What happens at the image edge?



What happens at the image edge?

Write down as many different ways of extending the edge as you can think of.

What happens at the image edge?



Mean padding
 $f[\text{end}+x] = \text{mean}(f)$



Zero order hold
 $f[\text{end}+x] = f[\text{end}]$

What happens at the image edge?



Periodic boundary condition

$$f[\text{end}+x] = f[x]$$



Symmetric boundary condition

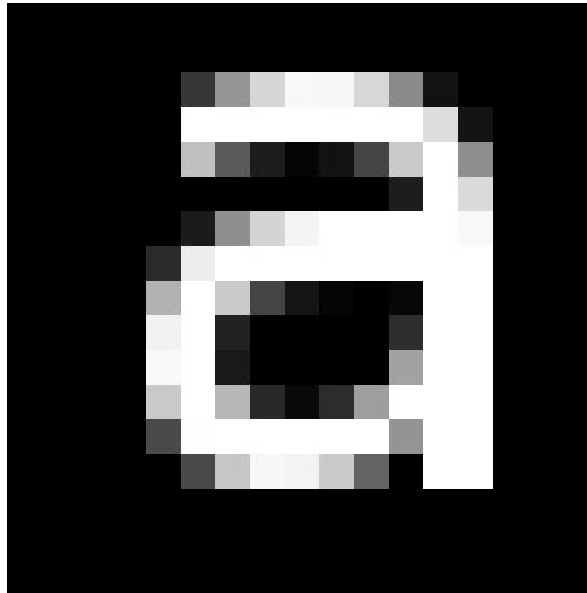
$$f[\text{end}+x] = f[\text{end}-x]$$

Beyond smoothing and sharpening

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

An image of a piece of text.

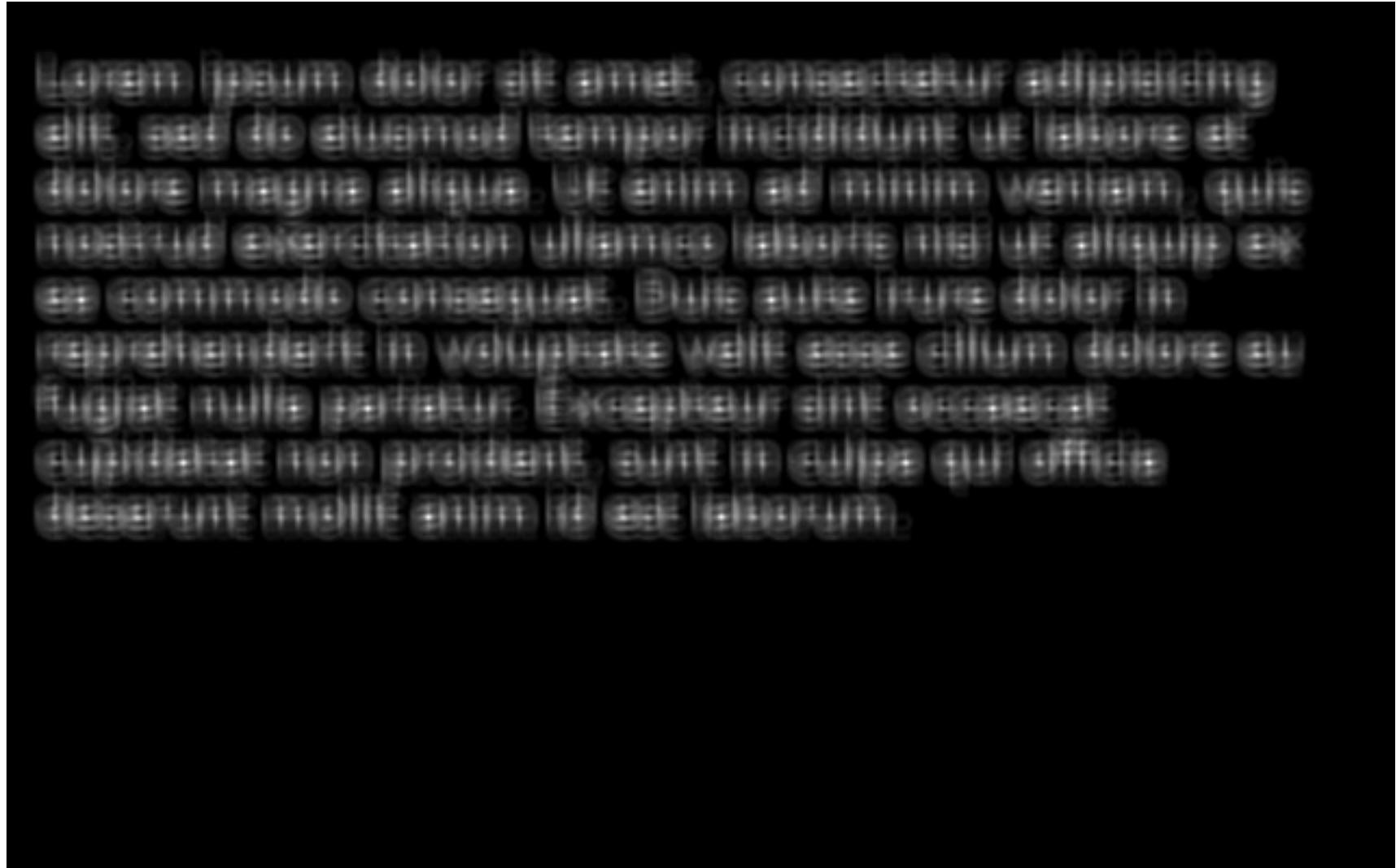
Beyond smoothing and sharpening



Filter kernel, image of letter “a”.

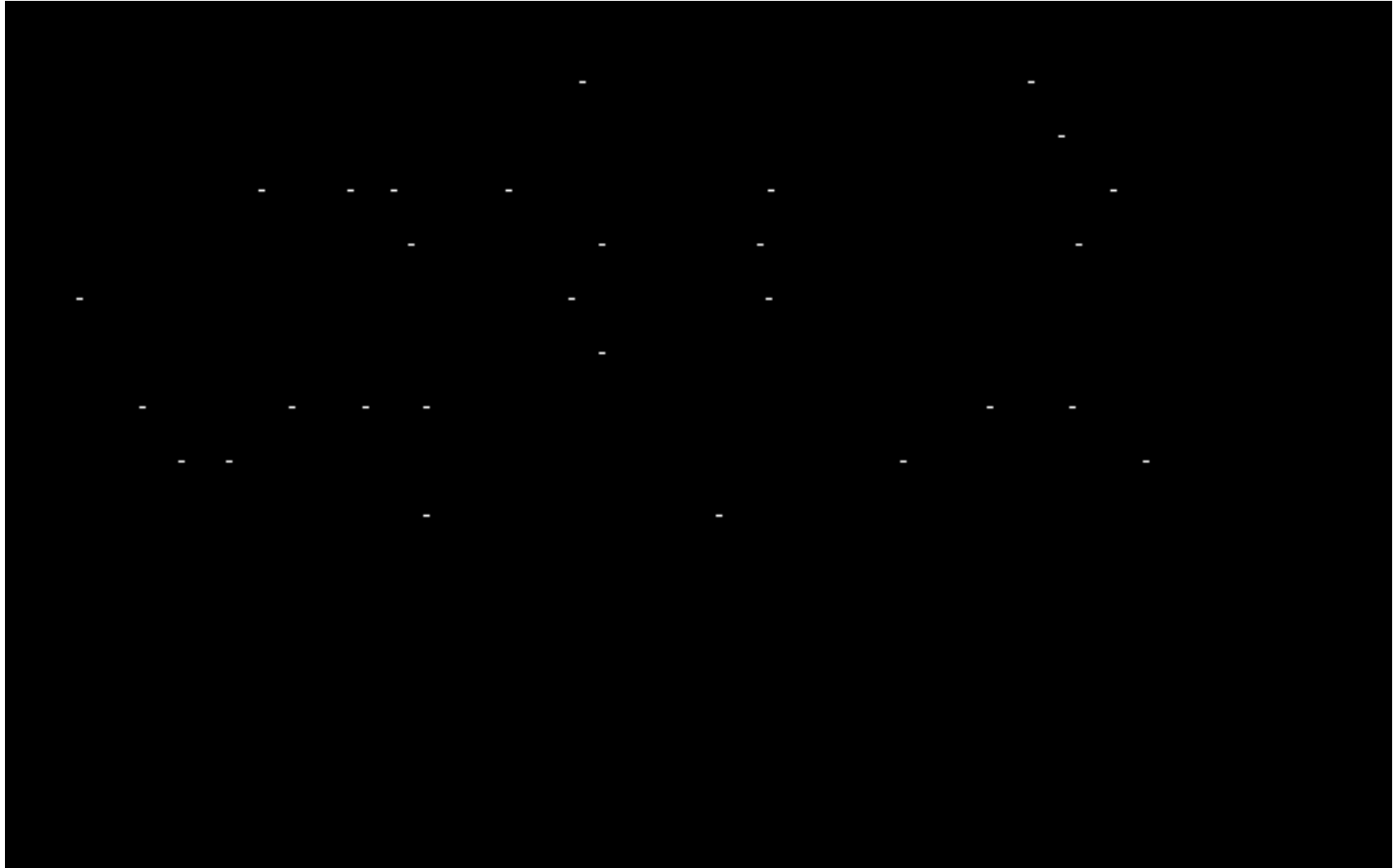
*What happens when we apply this kernel as a linear filter?
When is the output of this filter maximum/minimum?*

Beyond smoothing and sharpening



After linear filtering

Beyond smoothing and sharpening



Finding all pixels brighter than a manually selected threshold value

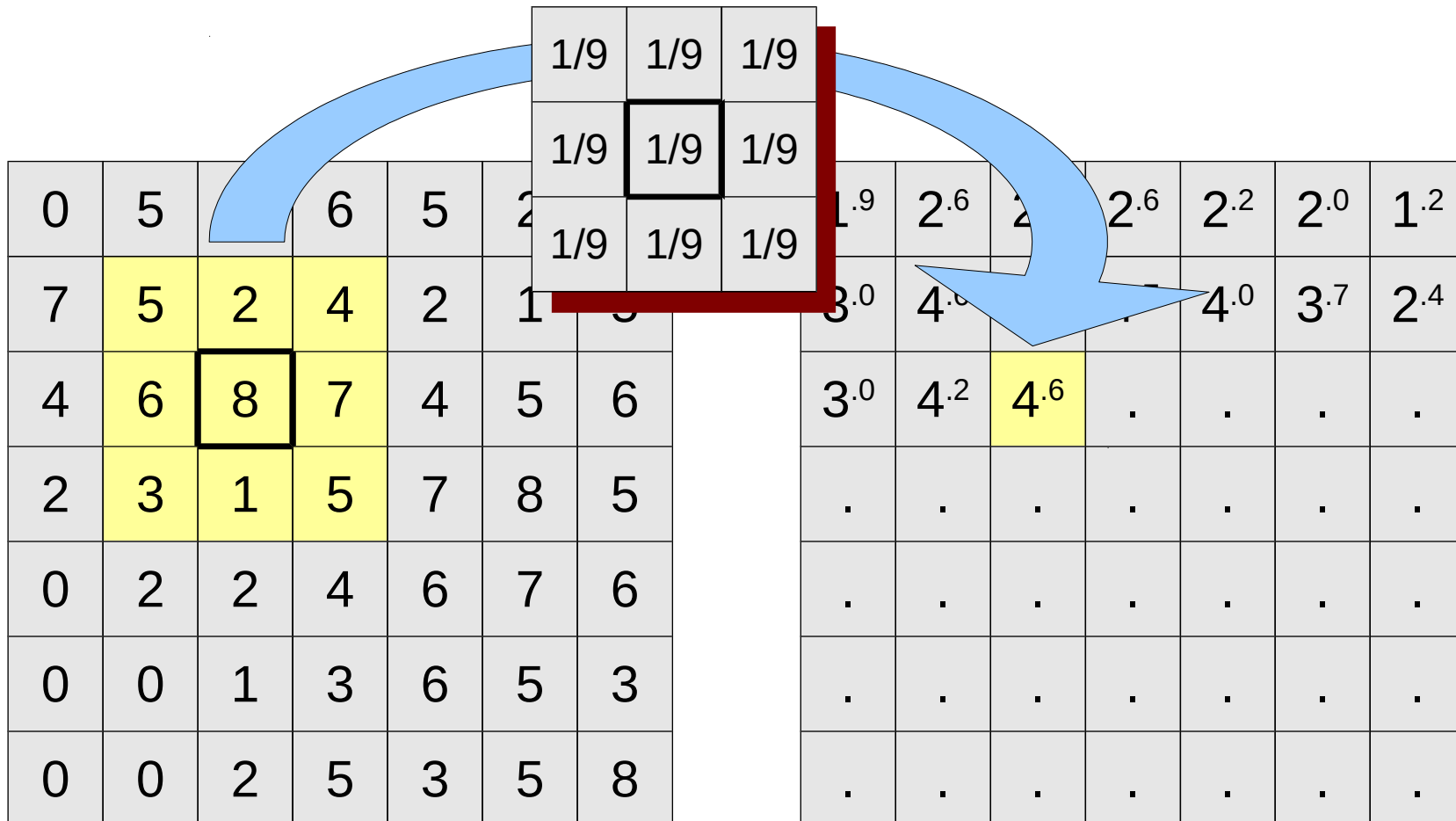
Beyond smoothing and sharpening

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Detected instances of letter “a”.

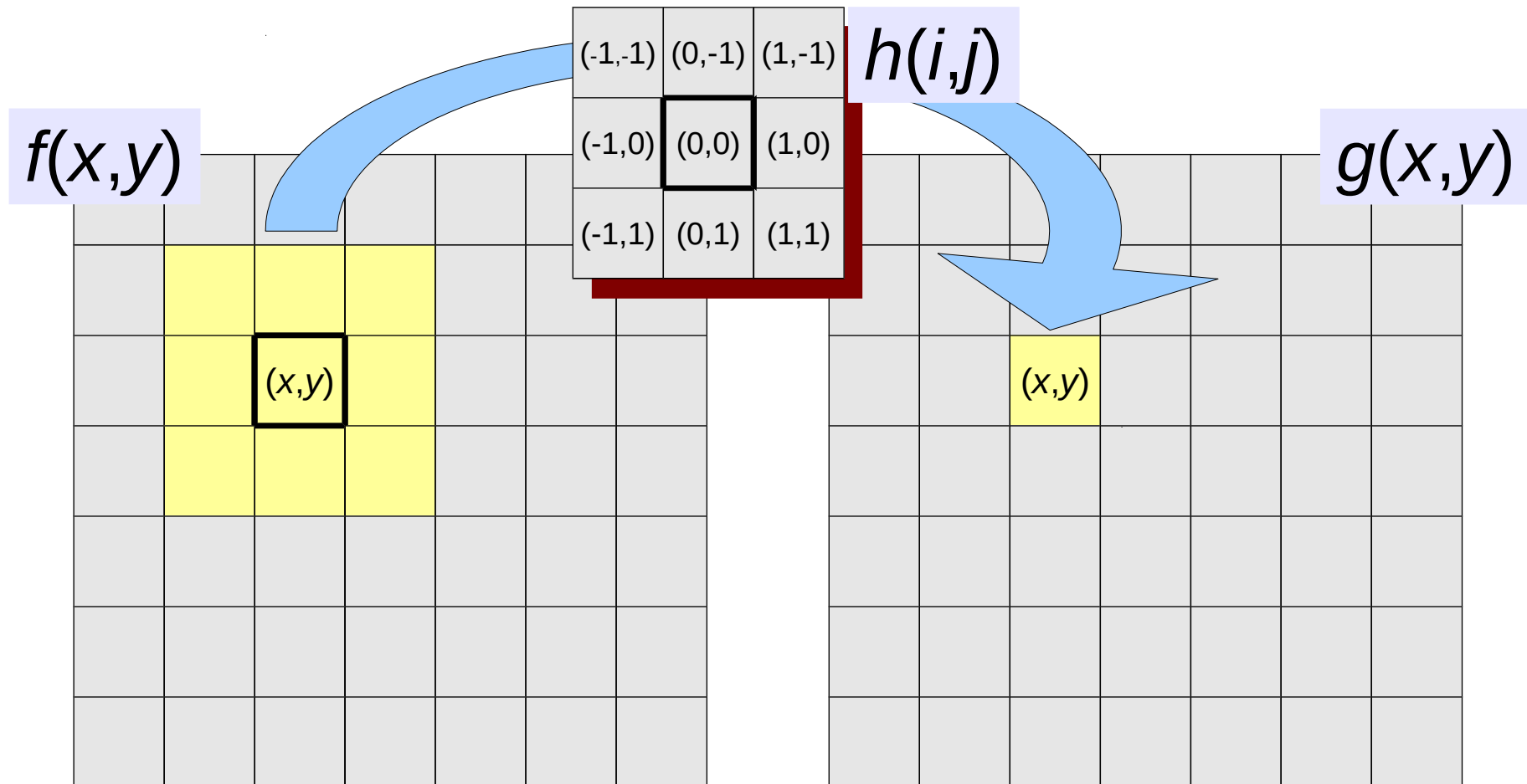
Linear neighbourhood operation

- For each pixel, multiply the values in its neighbourhood with the corresponding weights, then sum.



Linear neighbourhood operation

- For each pixel, multiply the values in its neighbourhood with the corresponding weights, then sum.



Correlation and convolution

- Two fundamental linear filtering operations.
- *Correlation*: move a filter mask over the image, and compute the sum of products at each location (exactly what we have done so far).
- *Convolution*: Same as correlation, but first rotate filter by 180 degrees (or *mirror* it in both x and y directions).

Correlation and convolution

Consider a 1D signal and small filter:

Signal:

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

Filter:

3 2 1



This signal is a discrete *impulse*.

What happens when we apply the filter as a *correlation*?

Correlation and convolution

Consider a 1D signal and small filter:

Signal:

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

Filter:

3 2 1

What happens when we apply the filter as a *correlation*?

Result:

0 0 0 0 0 0 1 2 3 0 0 0 0 0 0

We get a “mirrored” copy of the filter at the location of the impulse! (Verify this)

Correlation and convolution

Consider a 1D signal and small filter:

Signal:

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

Filter:

3 2 1

Mirrored filter:

1 2 3

What happens when we instead apply the filter as a convolution?

Result:

0 0 0 0 0 0 3 2 1 0 0 0 0 0 0

We get a copy of the filter at the location of the impulse! (Verify this)

Convolution

$$g(t) = f(t) \otimes h(t)$$

h is:

- impulse response function
- point-spread function
- convolution kernel

$$g(t) = \int_{-\infty}^{\infty} f(t-\tau) h(\tau) d\tau$$

$$g[n] = \sum_{k=a}^b f[n-k] h[k]$$

$[a,b]$ is the interval where h is defined, e.g. $[-1,1]$

Convolution properties

- Linear:

- Scaling invariant:

$$(C f) \otimes h = C (f \otimes h)$$

- Distributive:

$$(f + g) \otimes h = f \otimes h + g \otimes h$$

- Time Invariant:

(= *shift invariant*)

$$\text{shift}(f) \otimes h = \text{shift}(f \otimes h)$$

- Commutative:

$$f \otimes h = h \otimes f$$

- Associative:

$$f \otimes (h_1 \otimes h_2) = (f \otimes h_1) \otimes h_2$$

Associativity of convolution

$$f \otimes (h_1 \otimes h_2) = (f \otimes h_1) \otimes h_2$$

if $h = h_1 \otimes h_2$

then $f \otimes h = (f \otimes h_1) \otimes h_2$

thus: you can decompose h to speed up the operation!

E.g. the Gaussian can be decomposed into two one-dimensional filters:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \otimes \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}}$$

Kernel decomposition

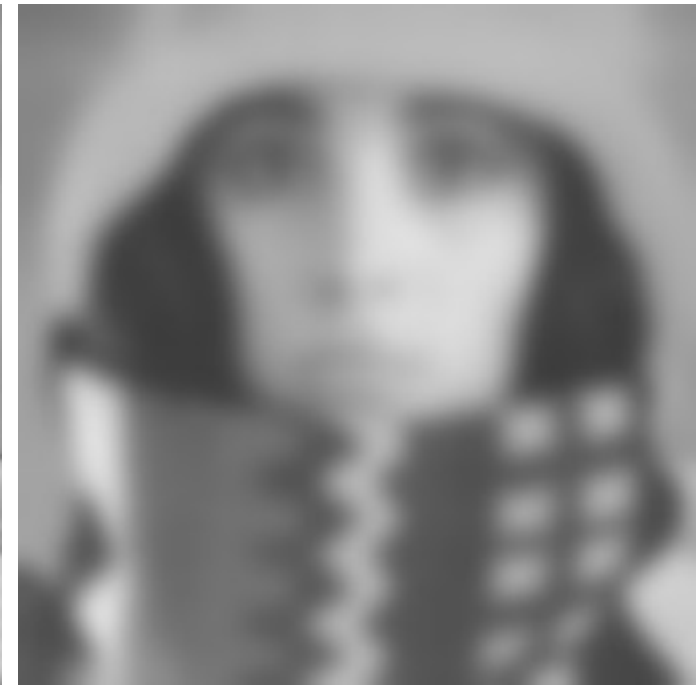
$$G = G_x \otimes G_y$$



original



convolved with G_x



convolved with G_y

G_x and G_y are both a kernel with 31x1 values

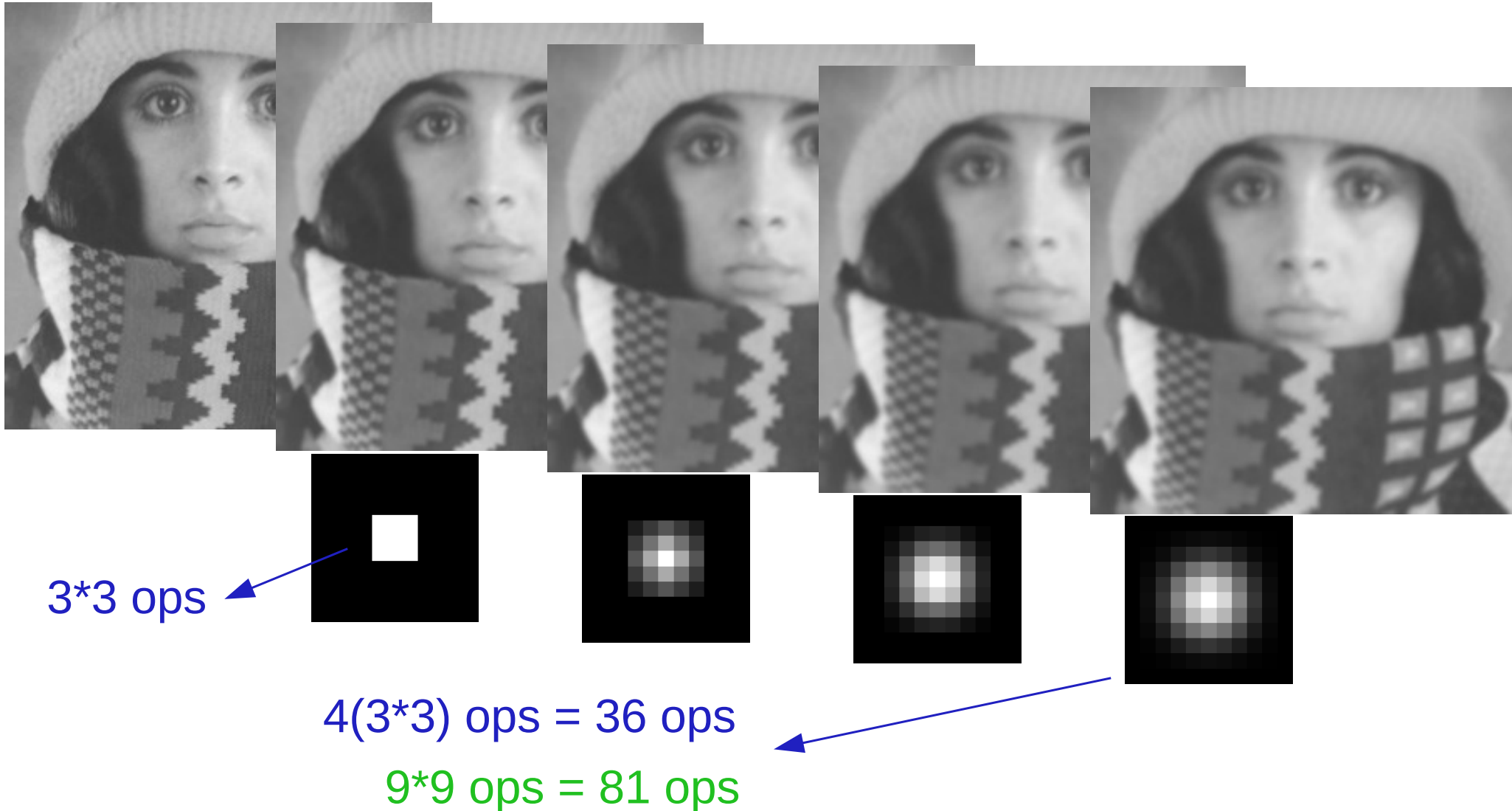
G is a kernel with 31x31 values

31+31 = 62 ops

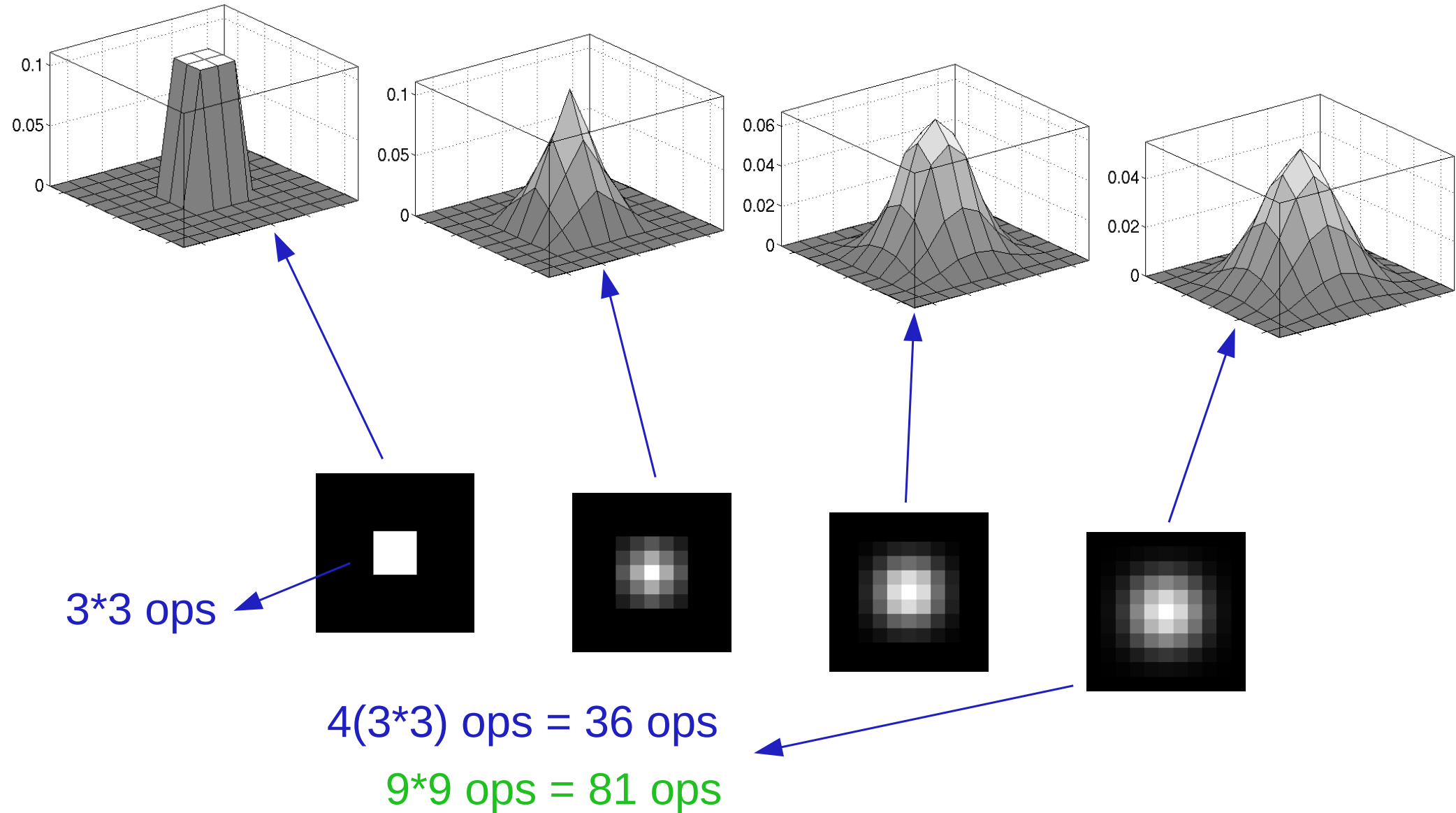
31x31 = 961 ops

Sequence of filters

$$f \otimes (h_1 \otimes h_2 \otimes h_3 \otimes \dots) = (((f \otimes h_1) \otimes h_2) \otimes h_3) \otimes \dots$$



Sequence of filters



Max/min filters

- Keep/enhance bright or dark details
- Rank filter or order-statistic filter
- Max filter sets the output pixelvalue to the maximum pixel intensity under the filtermask
 - => makes image brighter
- Min filter sets the output pixelvalue to the minimum intensity value under the filtermask
 - => makes image darker

Max/min filters

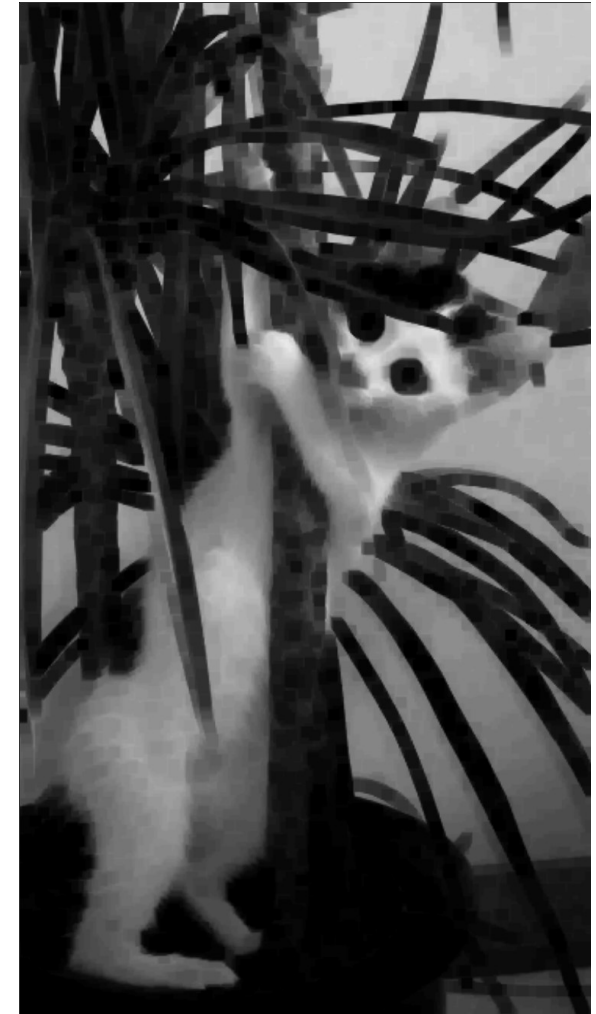
7 x 7 max



original



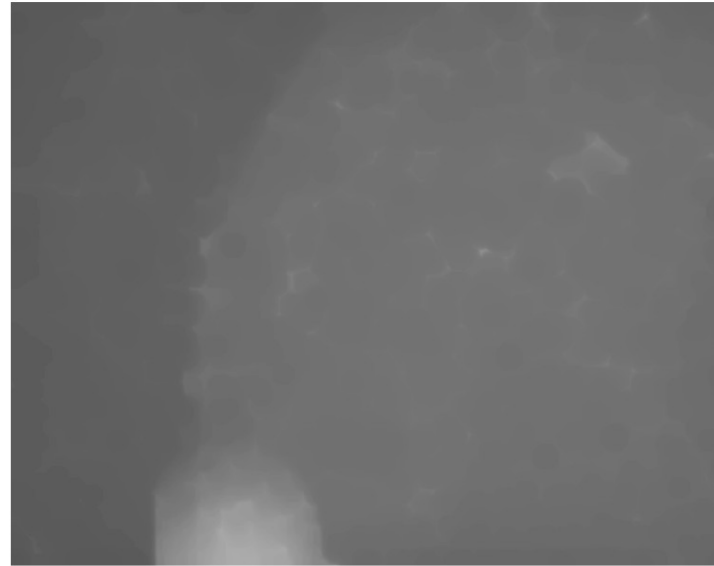
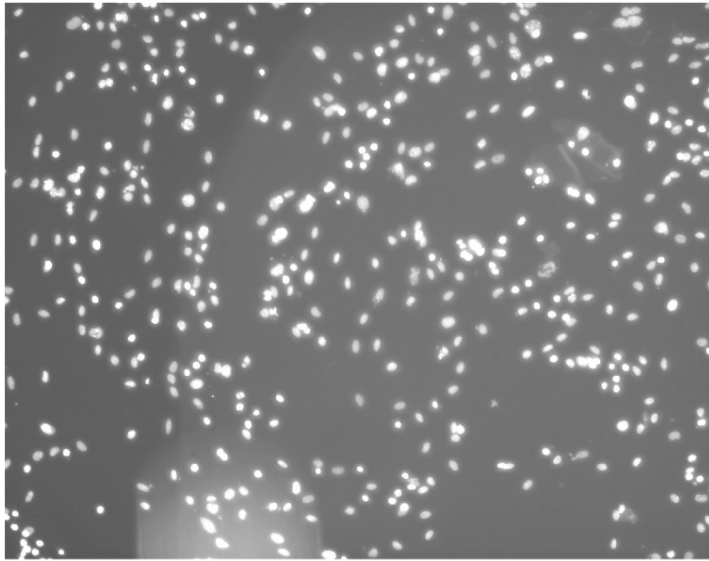
7 x 7 min



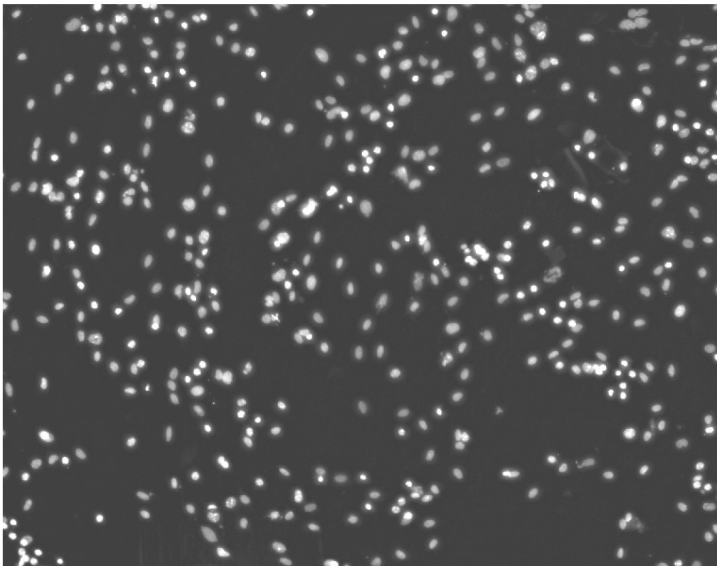
Background correction: TopHat filter

- Based combination of max/min filtering
 - 1)Need to know the approximate size of your objects of interest
 - 1) Estimate bg by a minfiltering followed by a maxfiltering. The filtersize should be larger than your objects of interest
 - 2) Subtract the bg image from the original

Background correction: TopHat filter

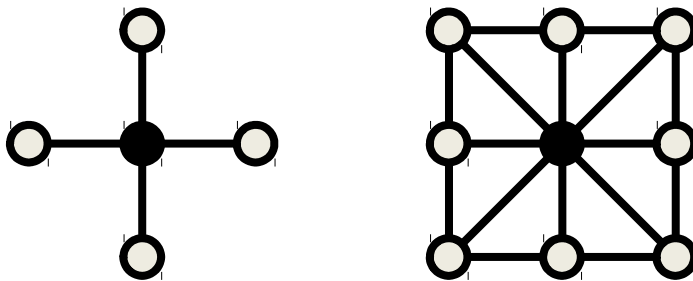


Circular
filter with
 $r=20$ pixels



Mathematical morphology

- Manipulation, or “filtering”, of objects in images, represented as binary masks. (0=background, 1=object)
- Structuring Element (SE): small set or structuring element SE to probe the image under study
- For each SE, define origo
- Shape and size must be adapted to geometric properties for the objects



Mathematical morphology

Four basic operations:

Erosion 

Dilation 

Opening 

Closing 



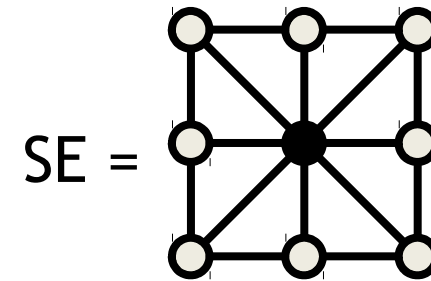
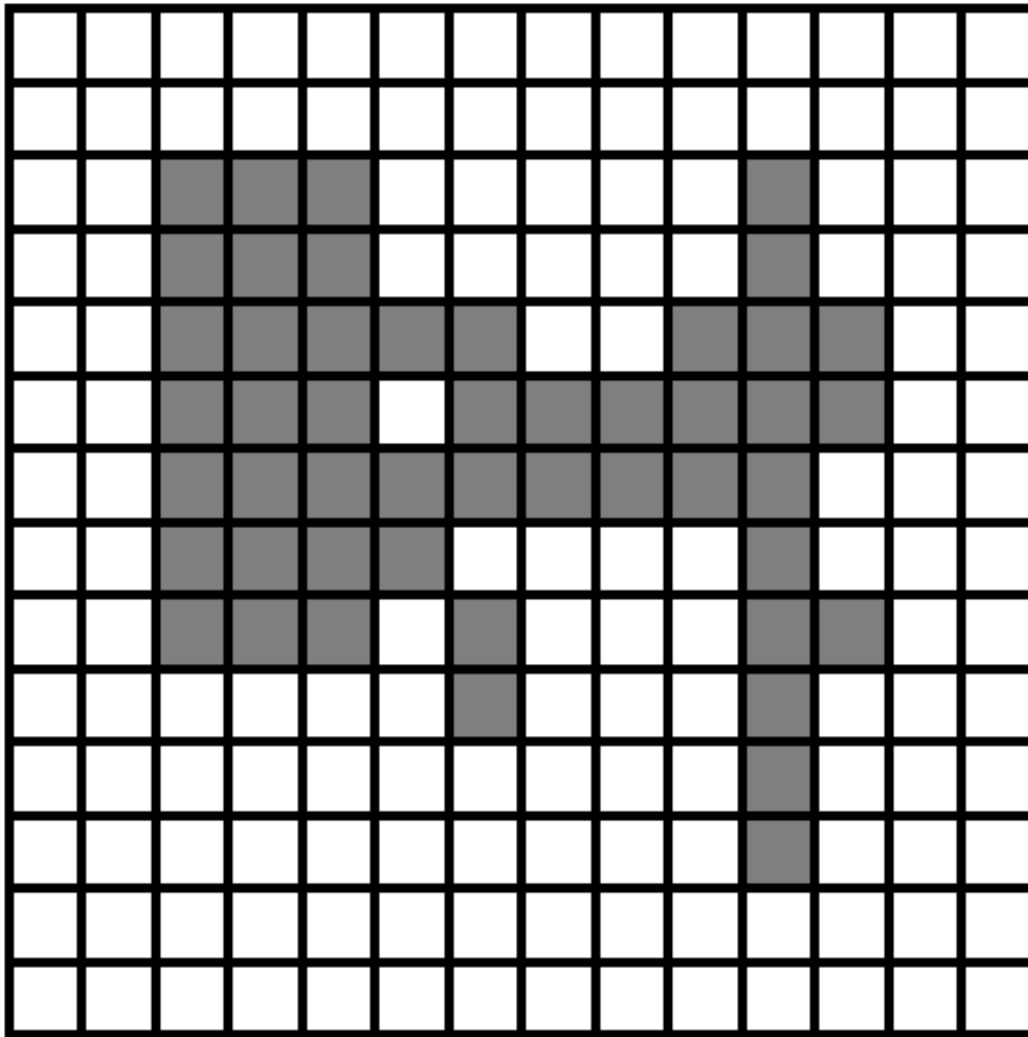
Erosion (shrinking)

- Does the structuring element fit inside the object?
- Keep only the object pixels corresponding to the origo of the SE when the SE fits entirely inside the object.

*(min filter with binary inputs,
with mirrored SE)*



Example, erosion





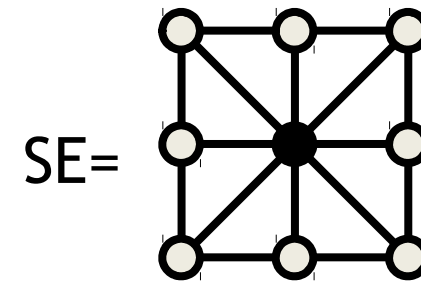
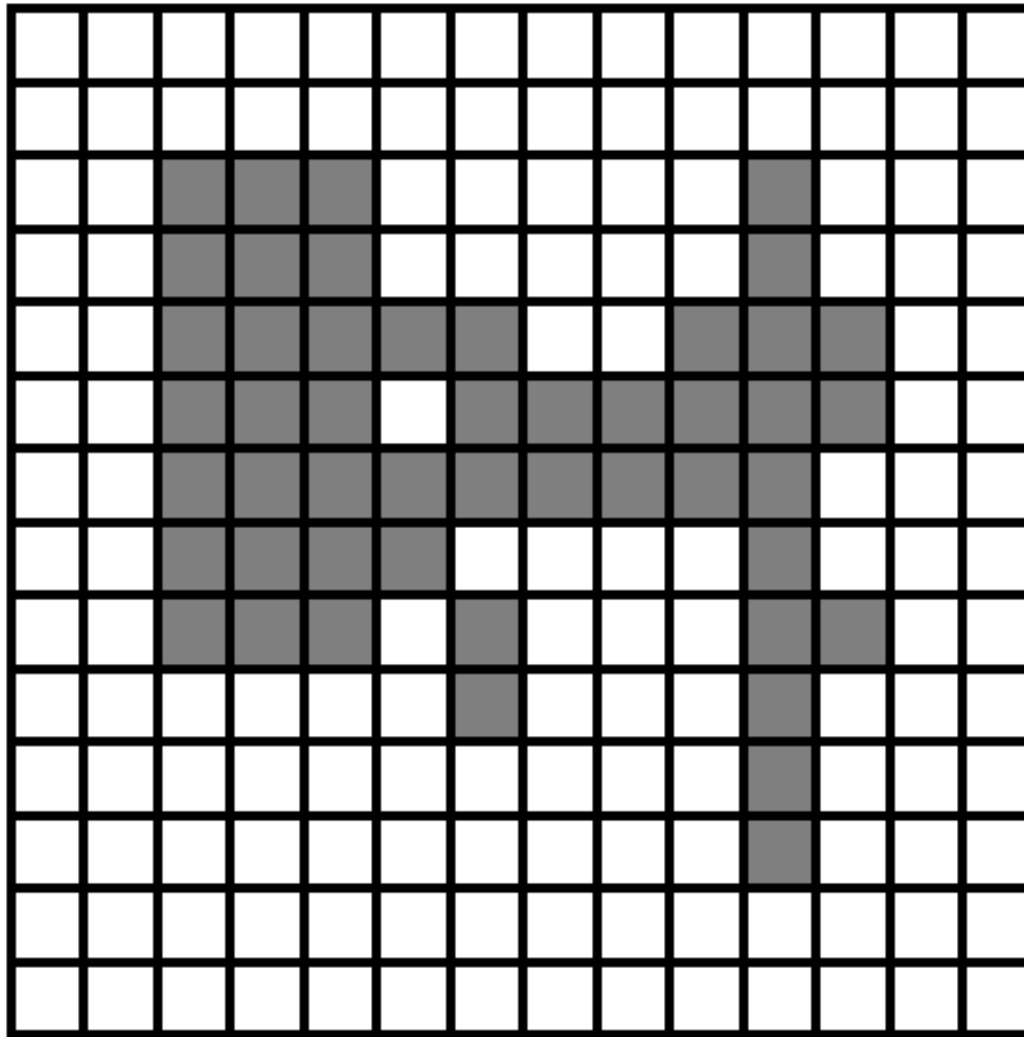
Dilation (growing)

- Grow the object with the SE
- Expand your object with all pixels in the SE, when the origo of the SE hits the object

(max filter with binary inputs, with mirrored SE)



Example: Dilation

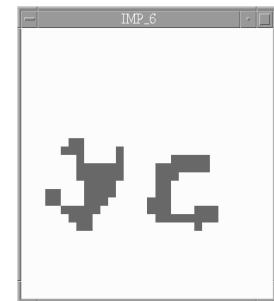
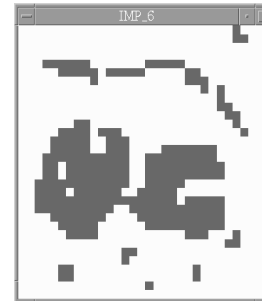


Effects of erosion and dilation

- erosion

- removal of structures of certain shape and size, given by SE (structure element)

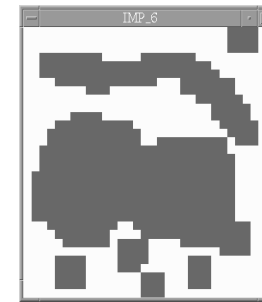
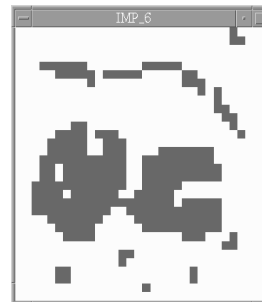
Example
3x3 SE



- dilation

- filling of holes of certain shape and size, given by SE

Example
3x3 SE



Combining erosion and dilation

WANTED:

remove structures / fill holes without affecting remaining parts
(overall size of objects)

SOLUTION:

combine erosion and dilation (using same SE)

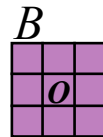
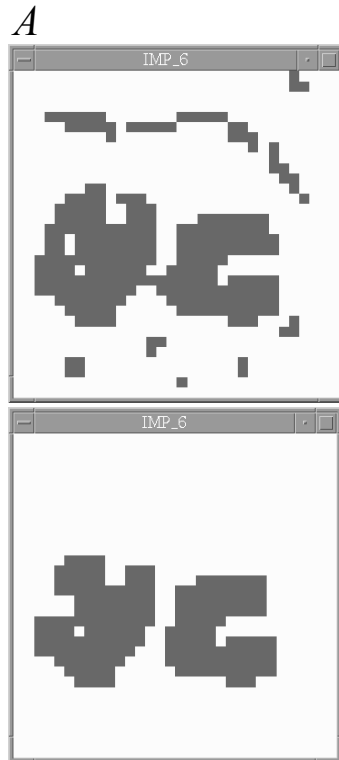
Opening 

Closing 

Opening

erosion followed by dilation, denoted \circ

$$A \circ B = (A \ominus B) \oplus B$$



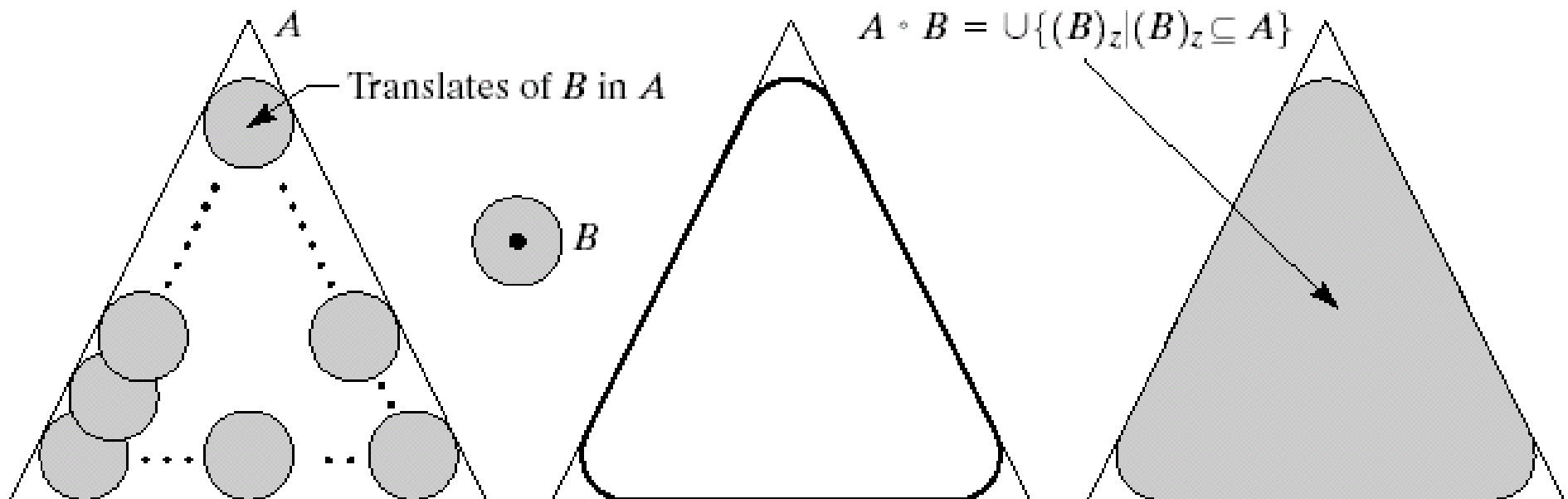
- eliminates protrusions
- breaks necks
- smooths contour

Rolling ball analogy

opening: roll ball(=SE) inside object

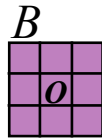
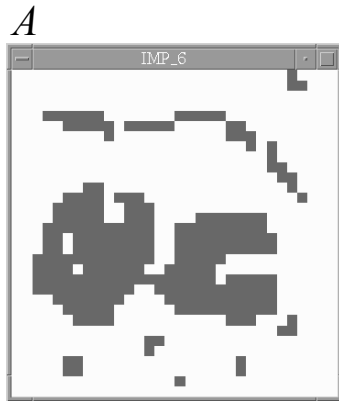
see SE as a "rolling ball"

boundary of $A \circ B$ = points in B that reaches closest to A boundary when B is rolled *inside* A

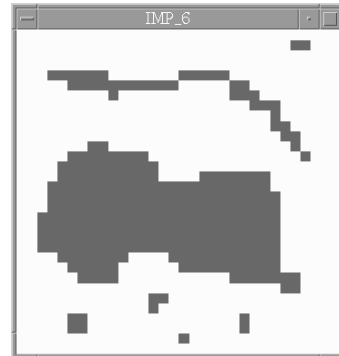


Closing

dilation followed by erosion, denoted •



$$A \bullet B = (A \oplus B) \ominus B$$

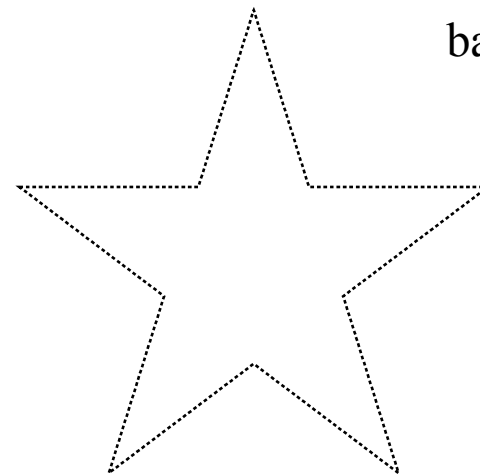
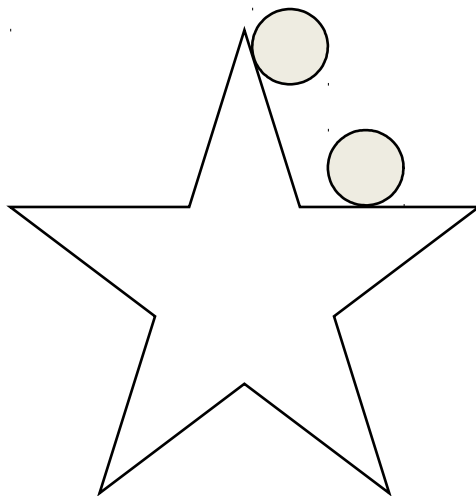


- smooth contour
- fuse narrow breaks and long thin gulfs
- eliminate small holes
- fill gaps in the contour

Rolling ball analogy

Closing: roll ball(=SE) outside object

boundary of $A \circ B$ = points in B that reaches
closest to A boundary when B is rolled
outside A



Fill in true border
after closing with
ball as SE

Summary of today's lecture

- Virtually all filtering is a local neighbourhood operation
- Convolution = linear and shift-invariant filters
 - e.g. mean filter, Gaussian weighted filter
 - kernel can sometimes be decomposed
- Many non-linear filters exist also
 - e.g. median filter, bilateral filter, max/min filters, tophat filter
- Mathematical morphology, “filtering” of shapes.
 - Useful for post-processing of segmentation masks