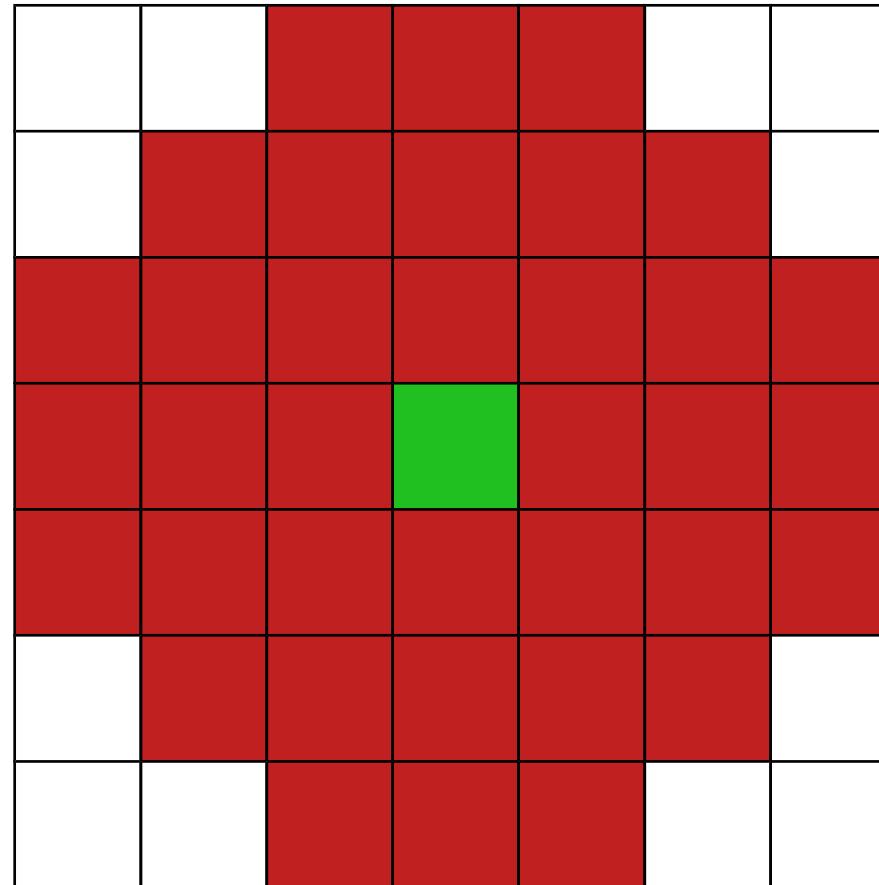
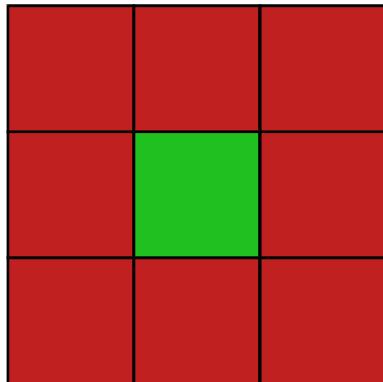
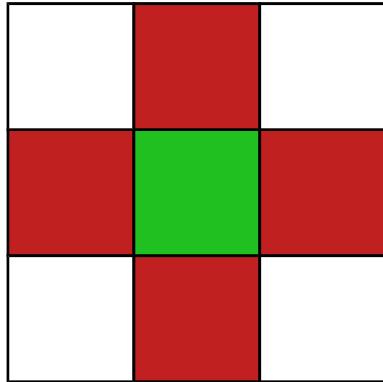


Today's lecture

- Local neighbourhood processing
 - smoothing an image
 - sharpening an image
- The convolution
 - What is it?
 - What is it useful for?
 - How can I compute it?
- Removing uncorrelated noise from an image
- The Fourier transform
 - What is it?
 - What is it useful for?

Neighbourhoods



Local neighbourhood operation

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

0	5		6	5	2	3
7	5	2	4	2	1	5
4	6	8	7	4	5	6
2	3	1	5	7	8	5
0	2	2	4	6	7	6
0	0	1	3	6	5	3
0	0	2	5	3	5	8

(mean)

1.9	2.6	2	2.6	2.2	2.0	1.2
3.0	4.0	4.0	4.0	4.0	3.7	2.4
3.0	4.2	4.6
.
.
.
.

Local neighbourhood operation

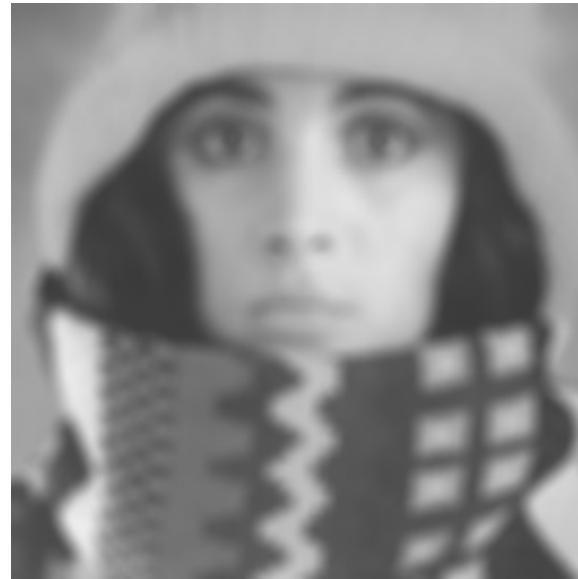
- Possible operations to do for each neighbourhood:
 - average (mean, median, etc.)
 - weighted average
 - other statistics (variance, maximum, etc.)
 - difference (to compute derivative)
 - ...
- Neighbourhood size and shape is very important
 - round neighbourhood gives rotation invariance
- Adaptive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

Smoothing an image

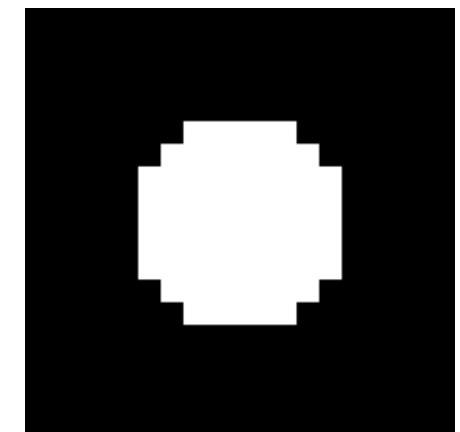
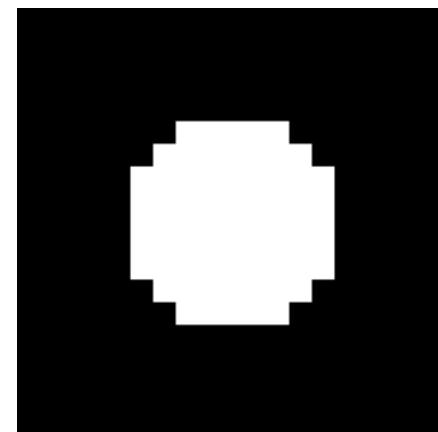
Input image



mean filter



median filter

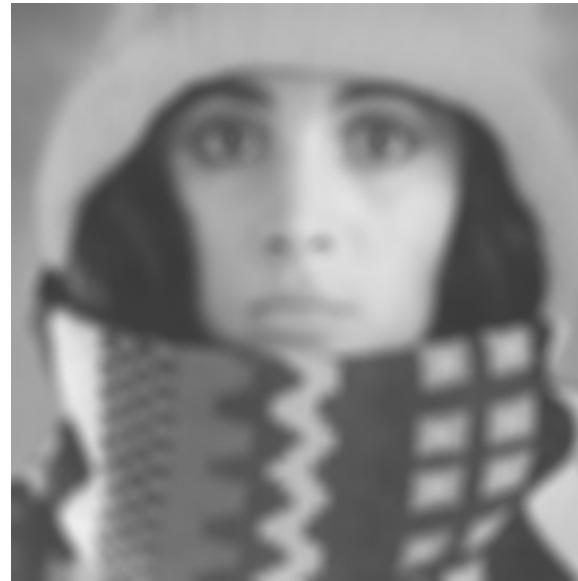


Smoothing an image

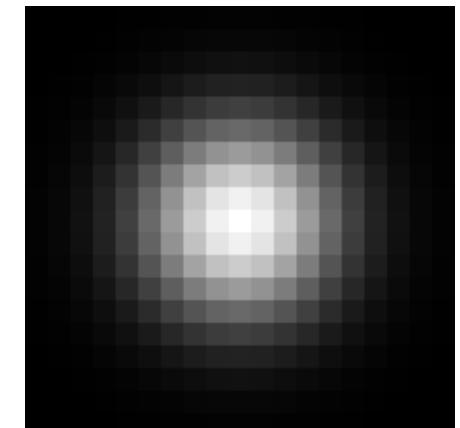
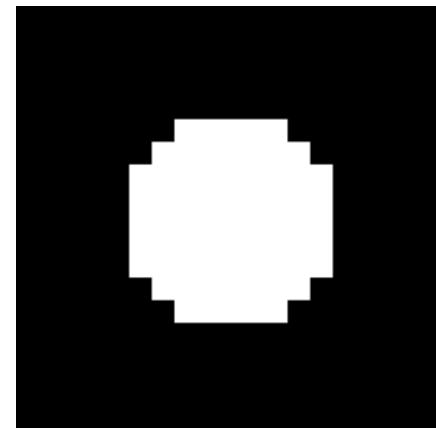
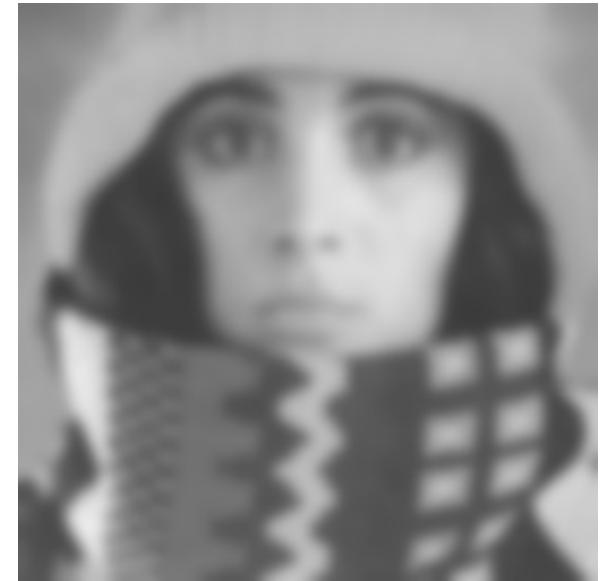
Input image



mean filter



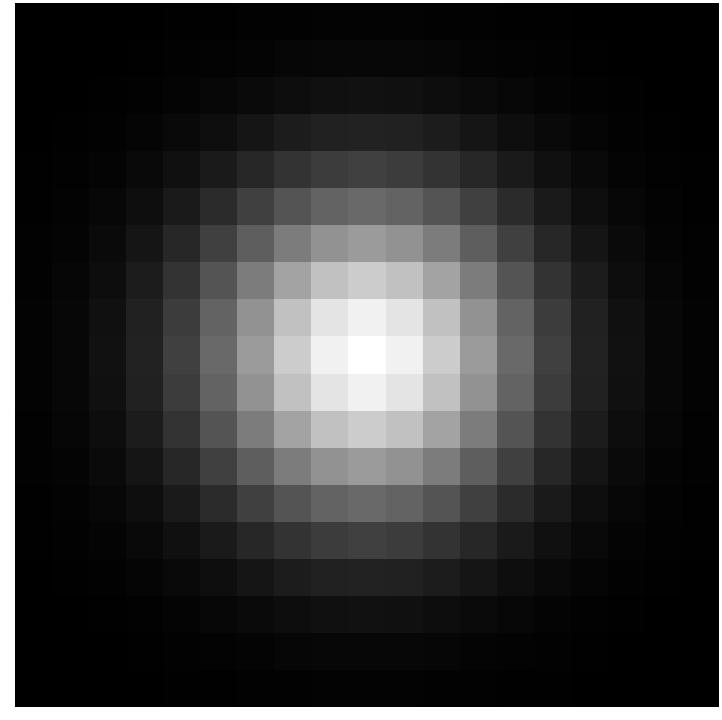
weighted mean filter



How to define Gaussian weights

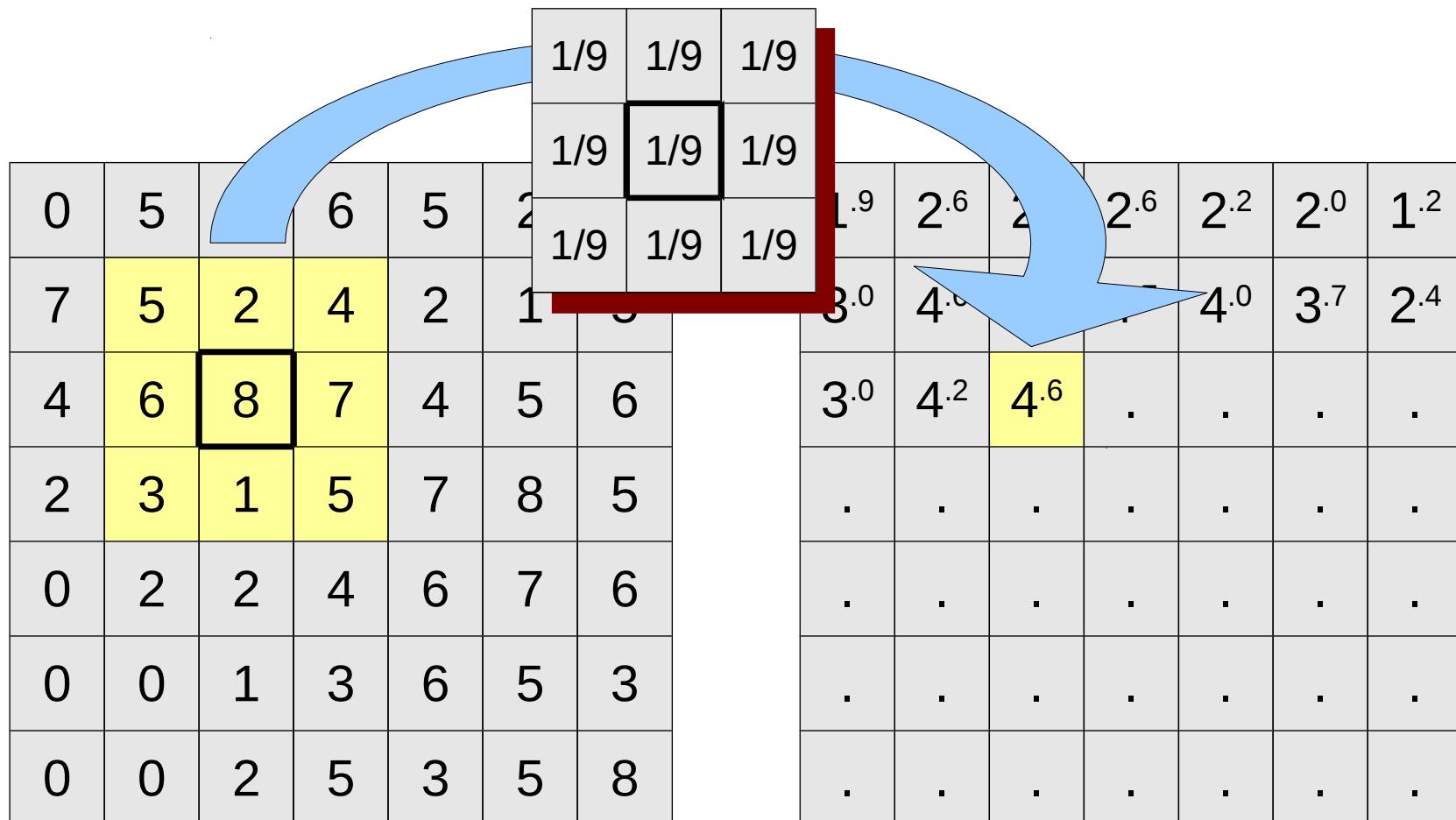
- σ determines the amount of smoothing
- the neighbourhood size should be large enough to contain the whole Gaussian bell!
rule of thumb: $\text{ceil}(6\sigma+1)$
- sum of all weights normalised to 1

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



Weighted mean filter

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



Application: noise reduction

input image



Normally
distributed
noise

3x3 mean filter



3x3 median filter

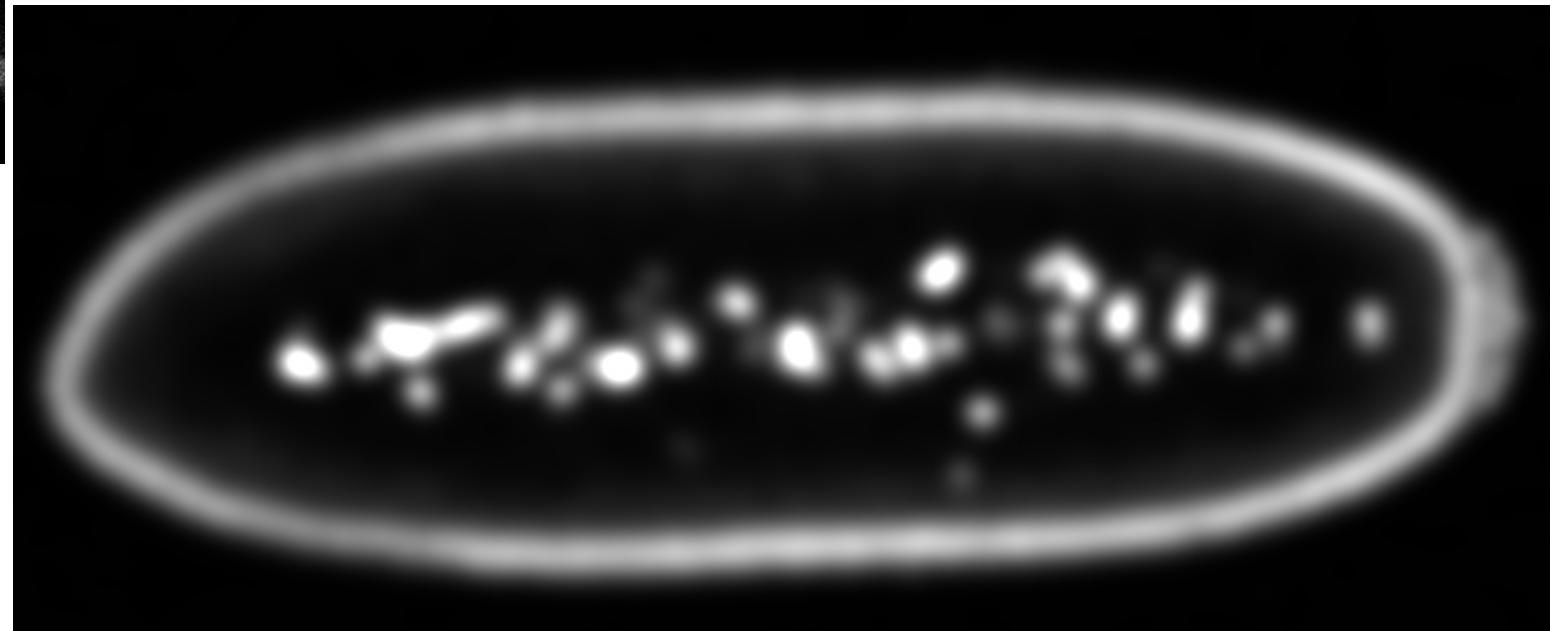
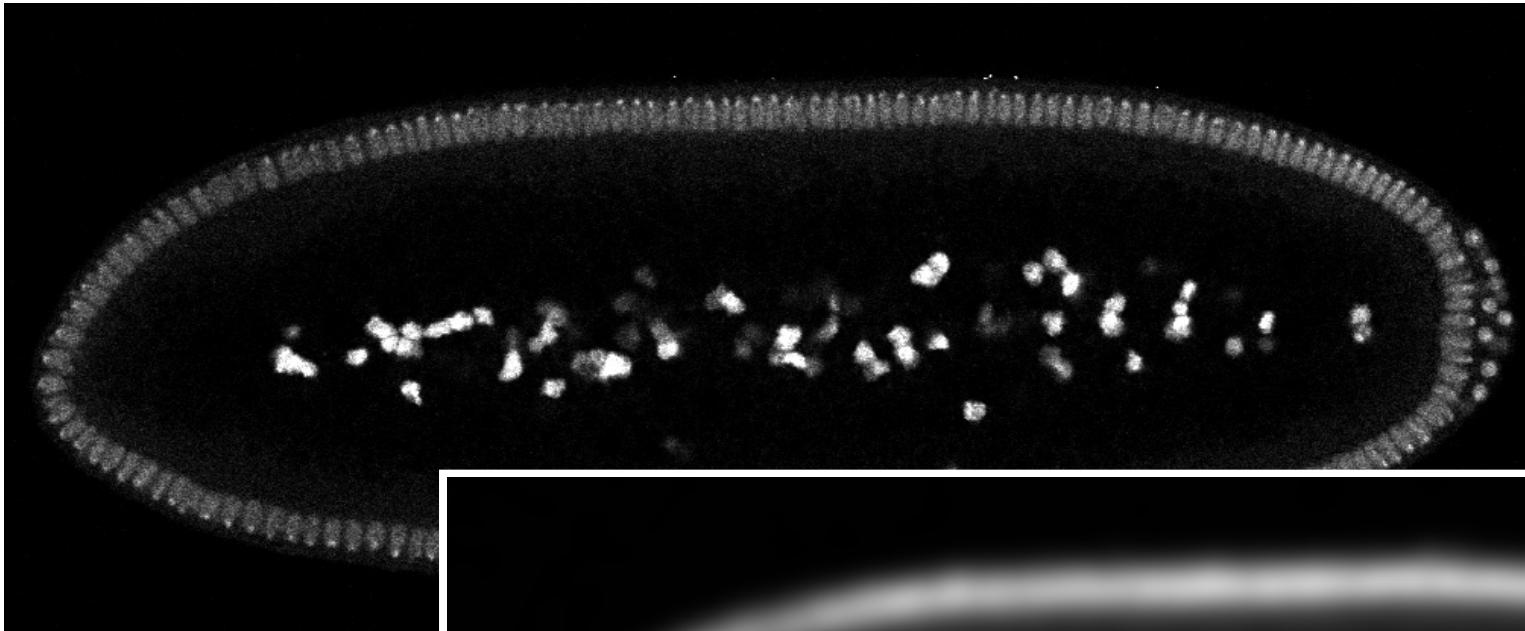


Salt &
pepper
noise

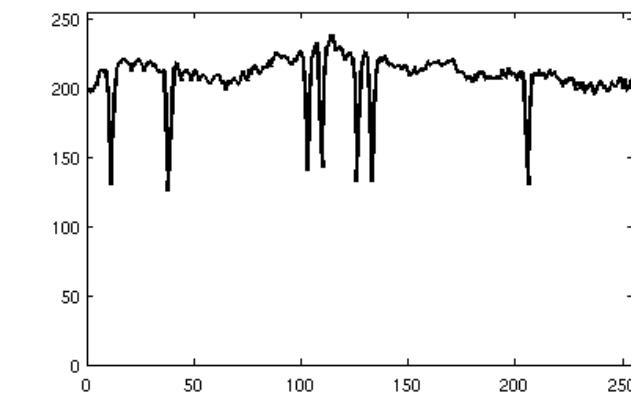
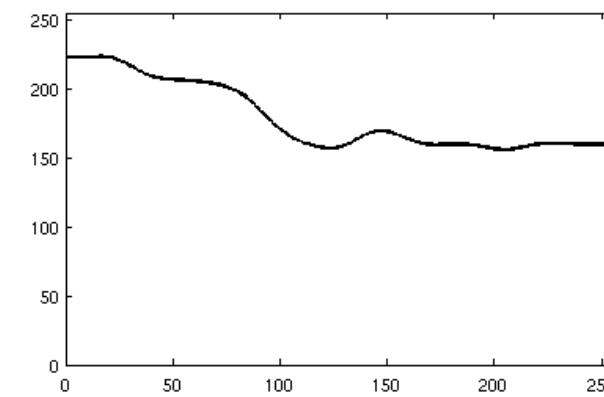
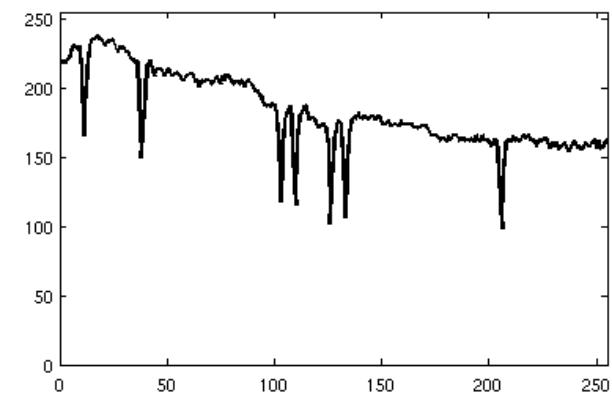
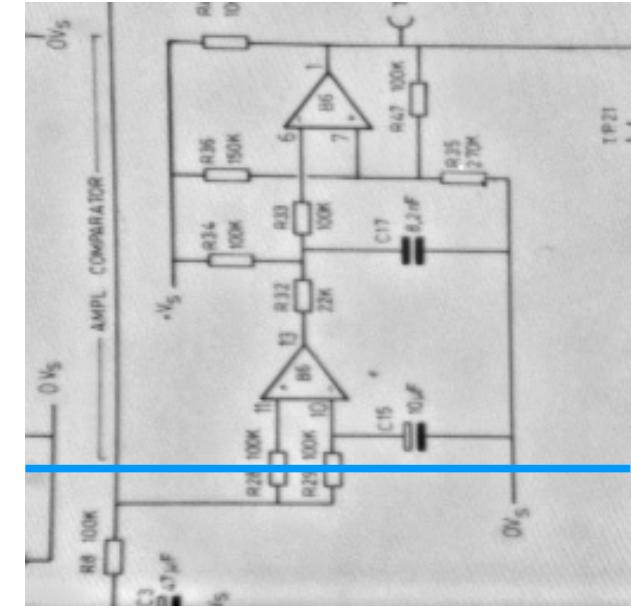
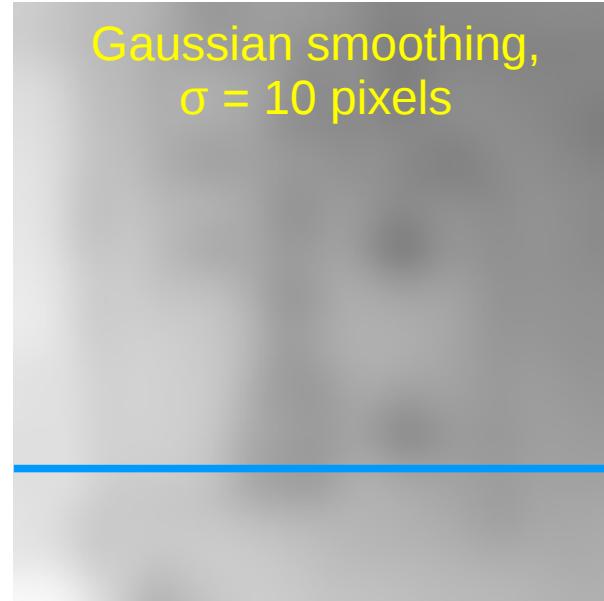
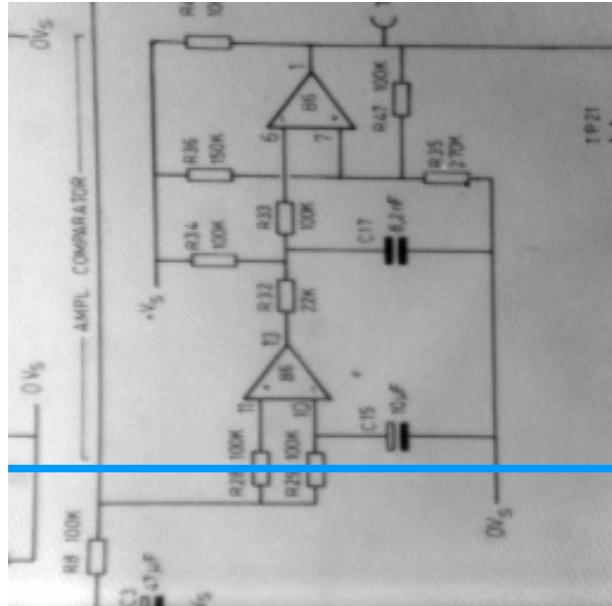


Application: abstraction

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



Application: shading correction



Sharpening an image

“Unsharp masking”



original



smoothed
(3x3)



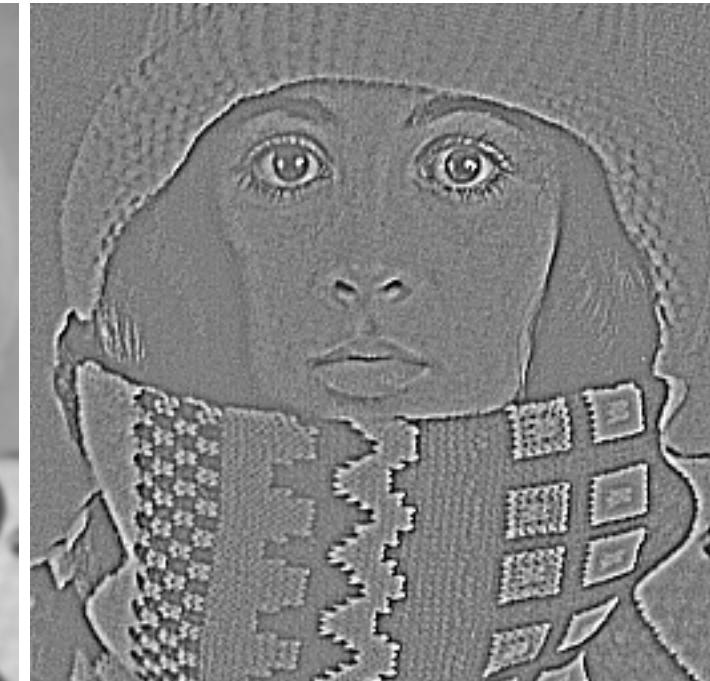
sharpened
($\alpha = 9$)

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

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

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

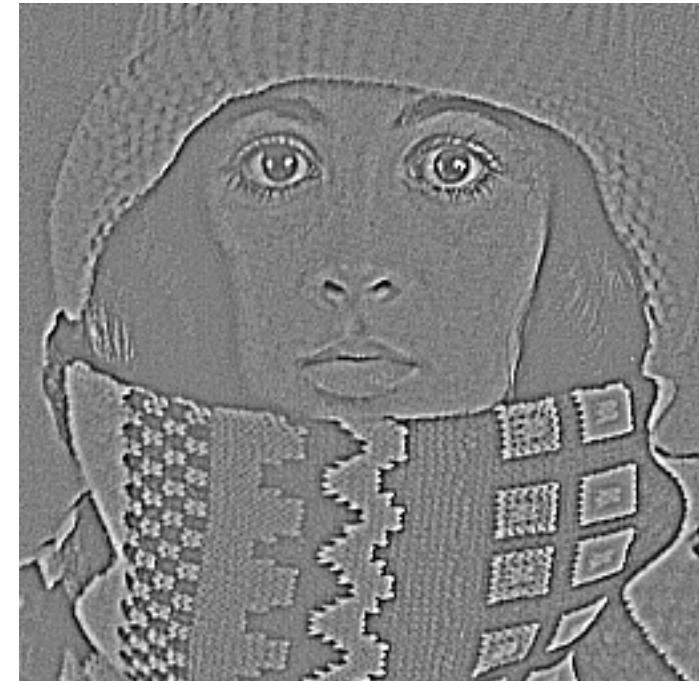
0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

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

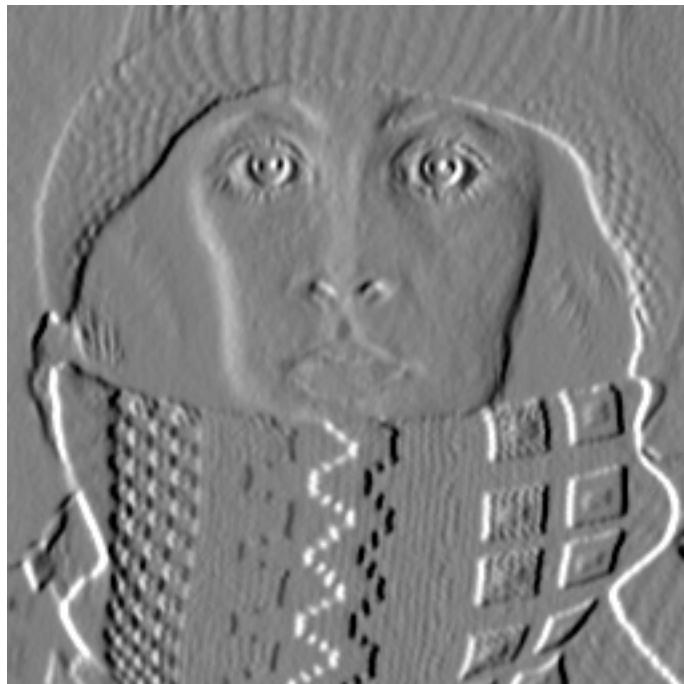
sharpened = original + 9 · (original – smoothed)

sharpened = original - Laplace



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



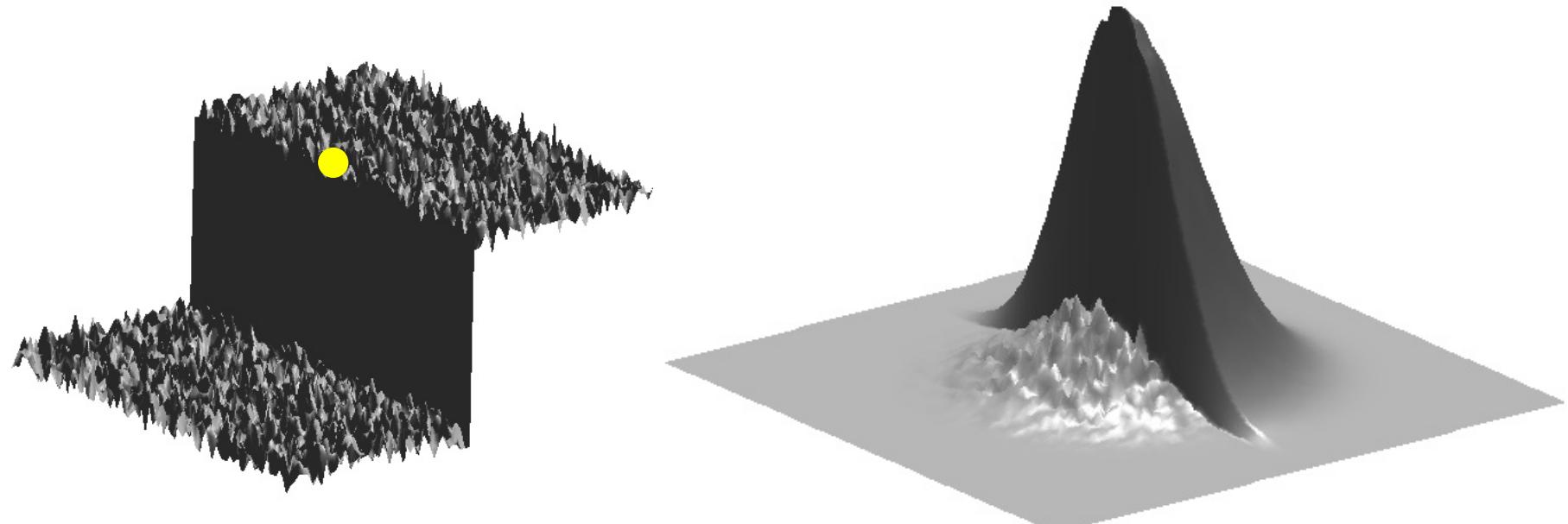
bilateral filter



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?



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



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

What happens at the image edge?



Periodic boundary condition

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

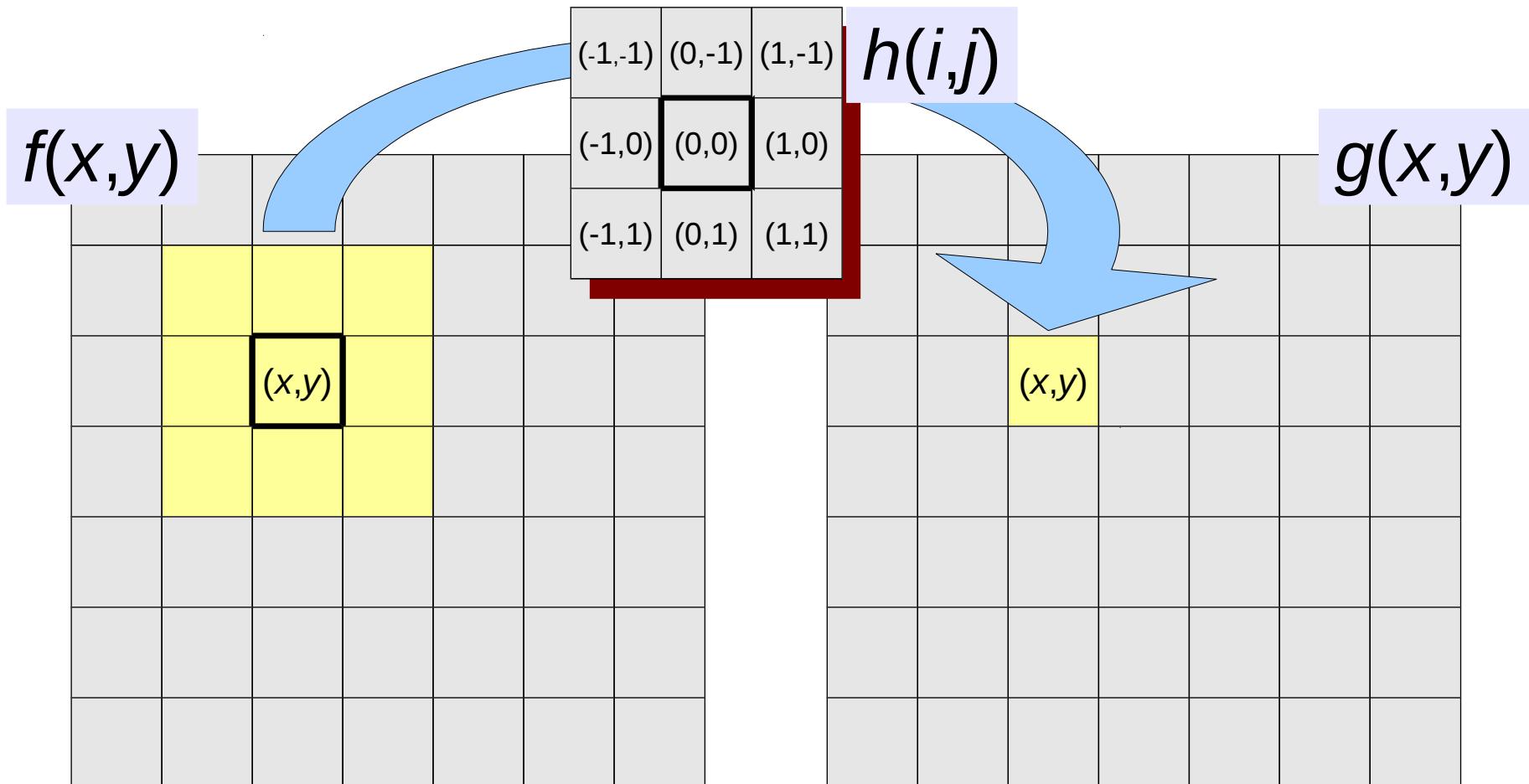


Symmetric boundary condition

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

Linear neighbourhood operation

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



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:
 $(= shift\ invariant)$ $shift(f) \otimes h = shift(f \otimes h)$
- 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

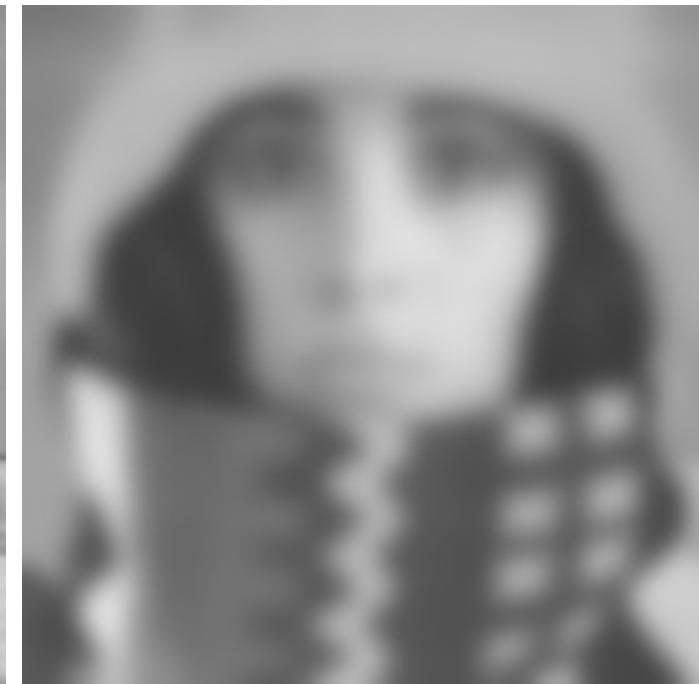
$$G = G_x \otimes G_y$$



original



convolved with G_x



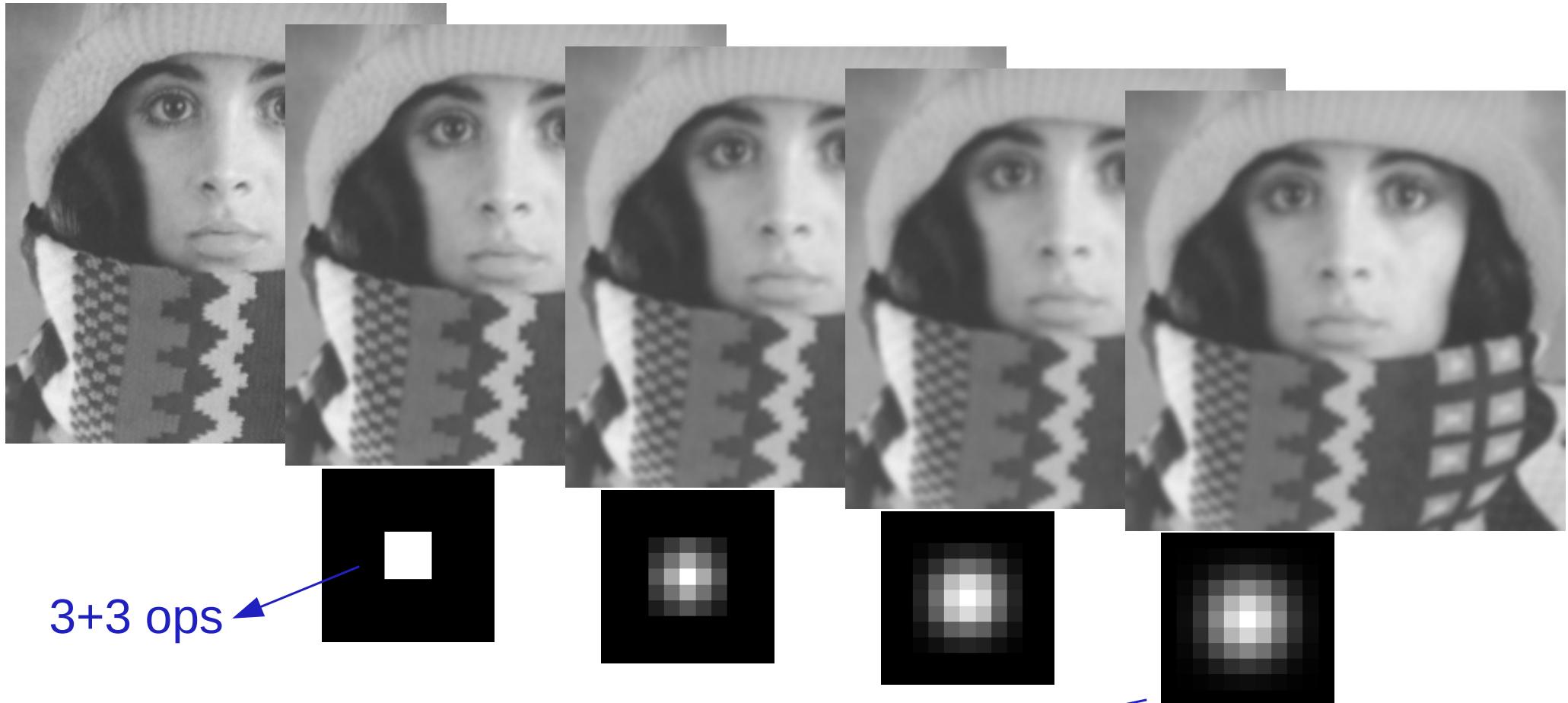
convolved with G_y

G_x and G_y are both a kernel with 31×1 values
 G is a kernel with 31×31 values

$31+31 = 62$ MADs
 $31 \times 31 = 961$ MADs

Sequence of filters

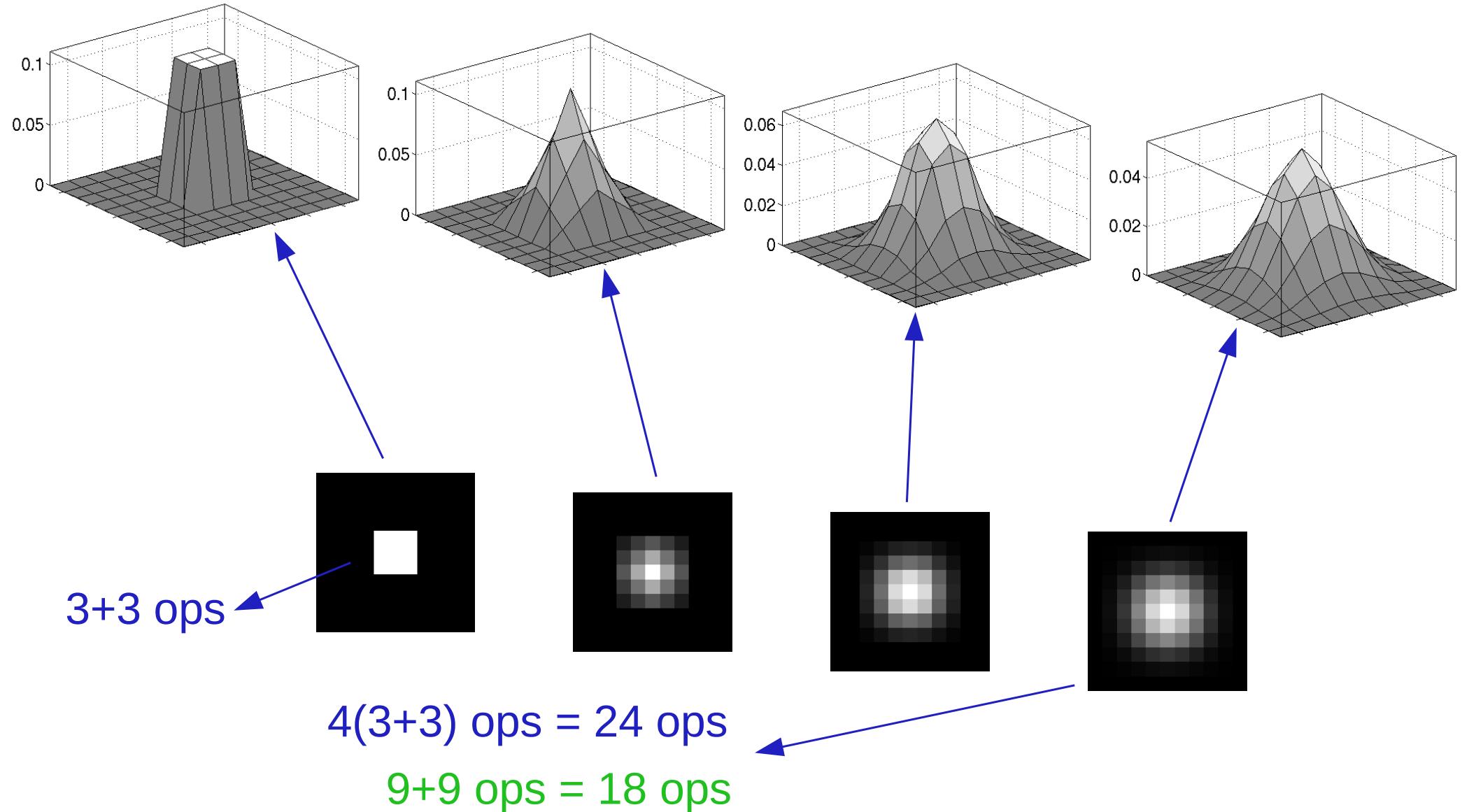
$$f \otimes (h_1 \otimes h_2 \otimes h_3 \otimes \dots) = (((f \otimes h_1) \otimes h_2) \otimes h_3) \otimes \dots$$



3+3 ops

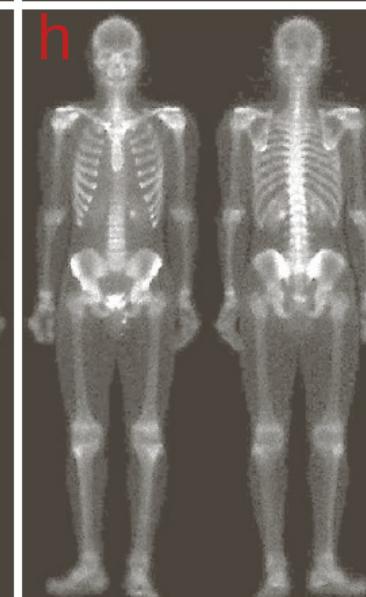
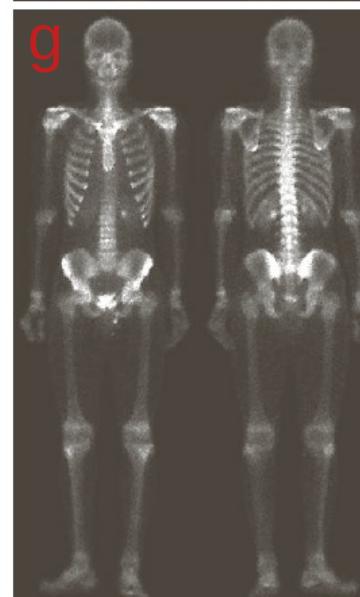
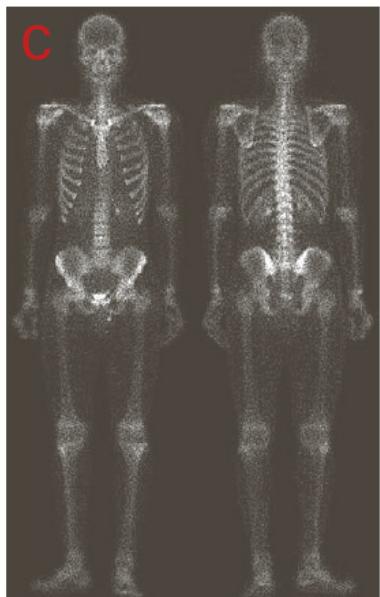
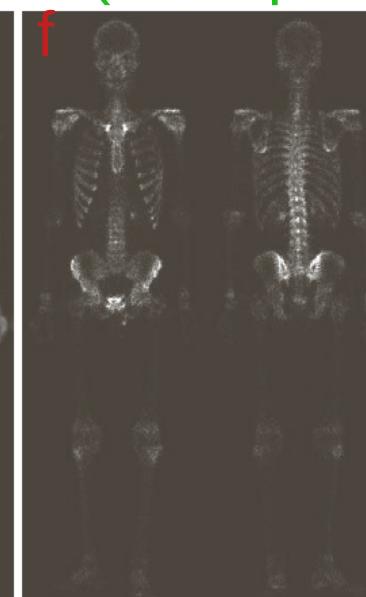
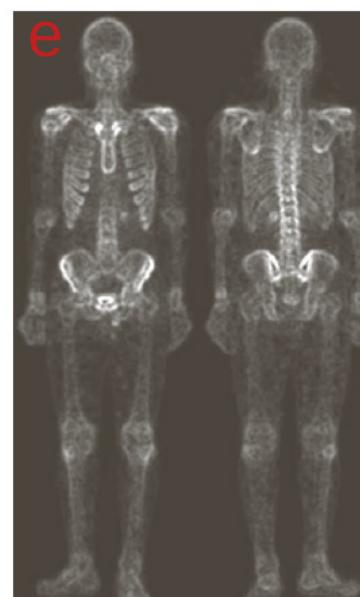
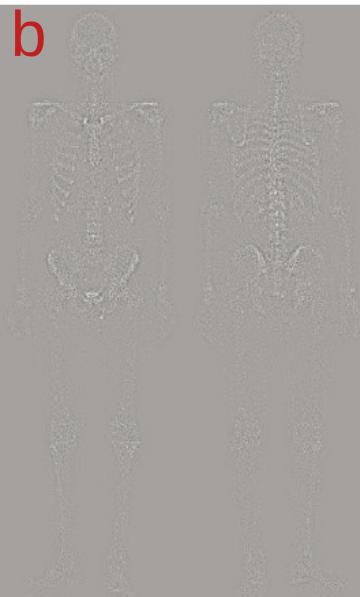
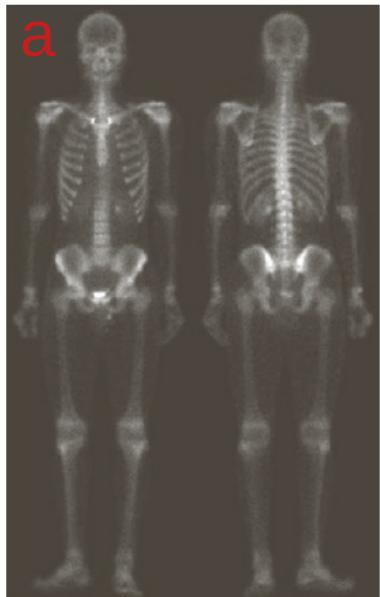
4(3+3) ops = 24 ops
9+9 ops = 18 ops

Sequence of filters



Sequence of filters

(example from Section 3.7)



b = Laplace(a)

c = a - b

d = Sobel(a)

e = smooth(d)

f = c * e

g = a + f

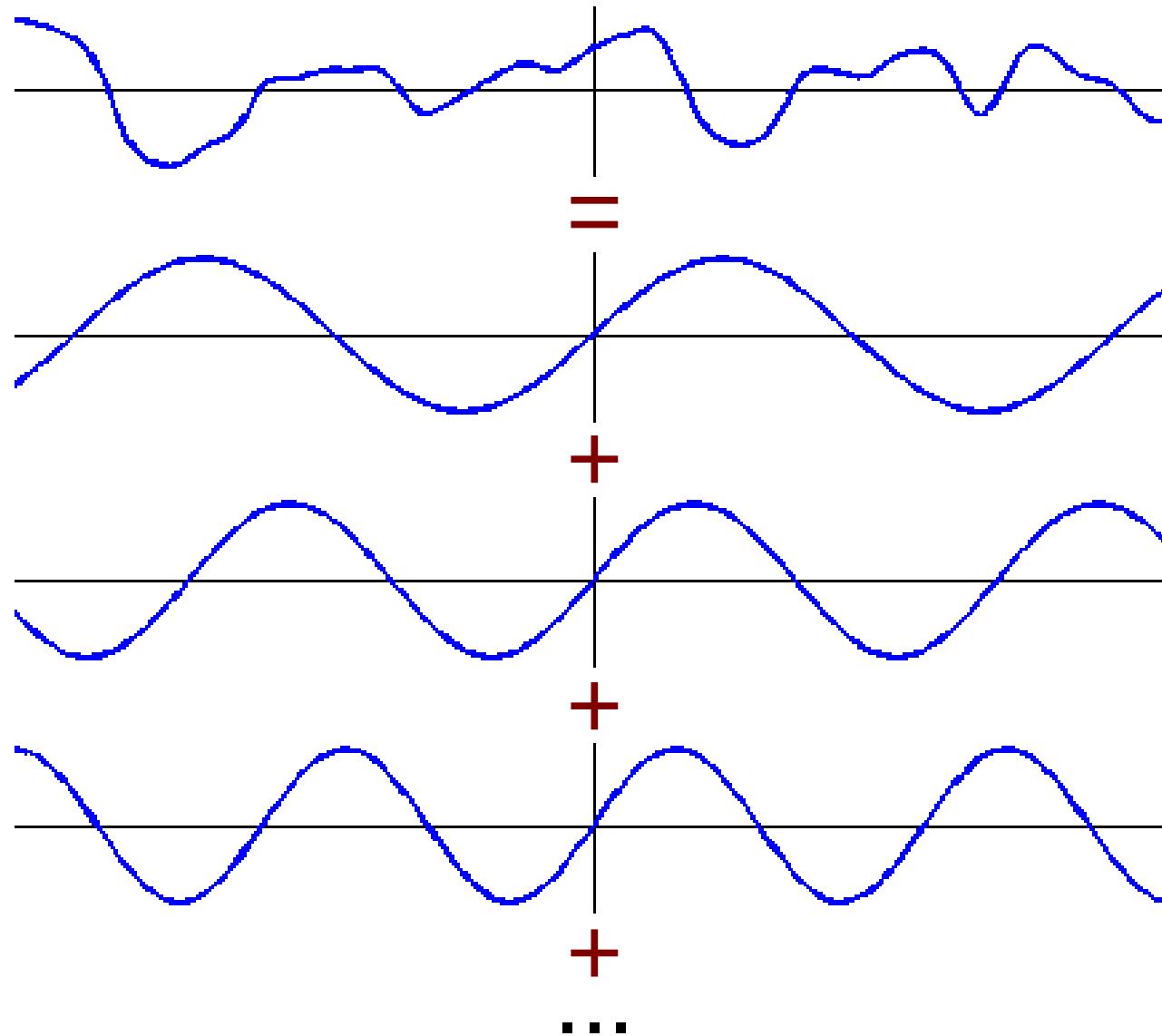
h = gamma(g)

Jean Baptiste Joseph Fourier

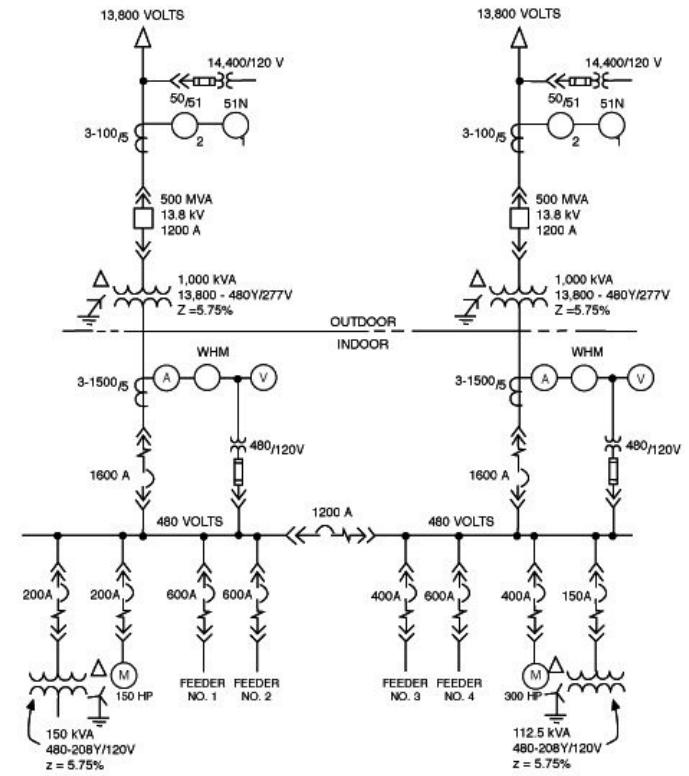
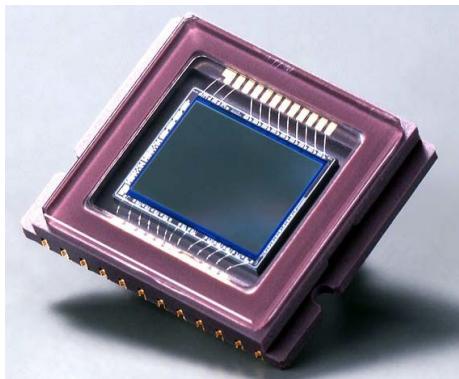
- Born 21 March 1768, Auxerre (Bourgogne region).
- Died 16 May 1830, Paris.
- Same age as Napoleon Bonaparte.
- Permanent Secretary of the French Academy of Sciences (1822-1830).
- Foreign member of the Royal Swedish Academy of Sciences (1830).



The Fourier transform



Fourier analysis



TYPICAL ONE-LINE DIAGRAM

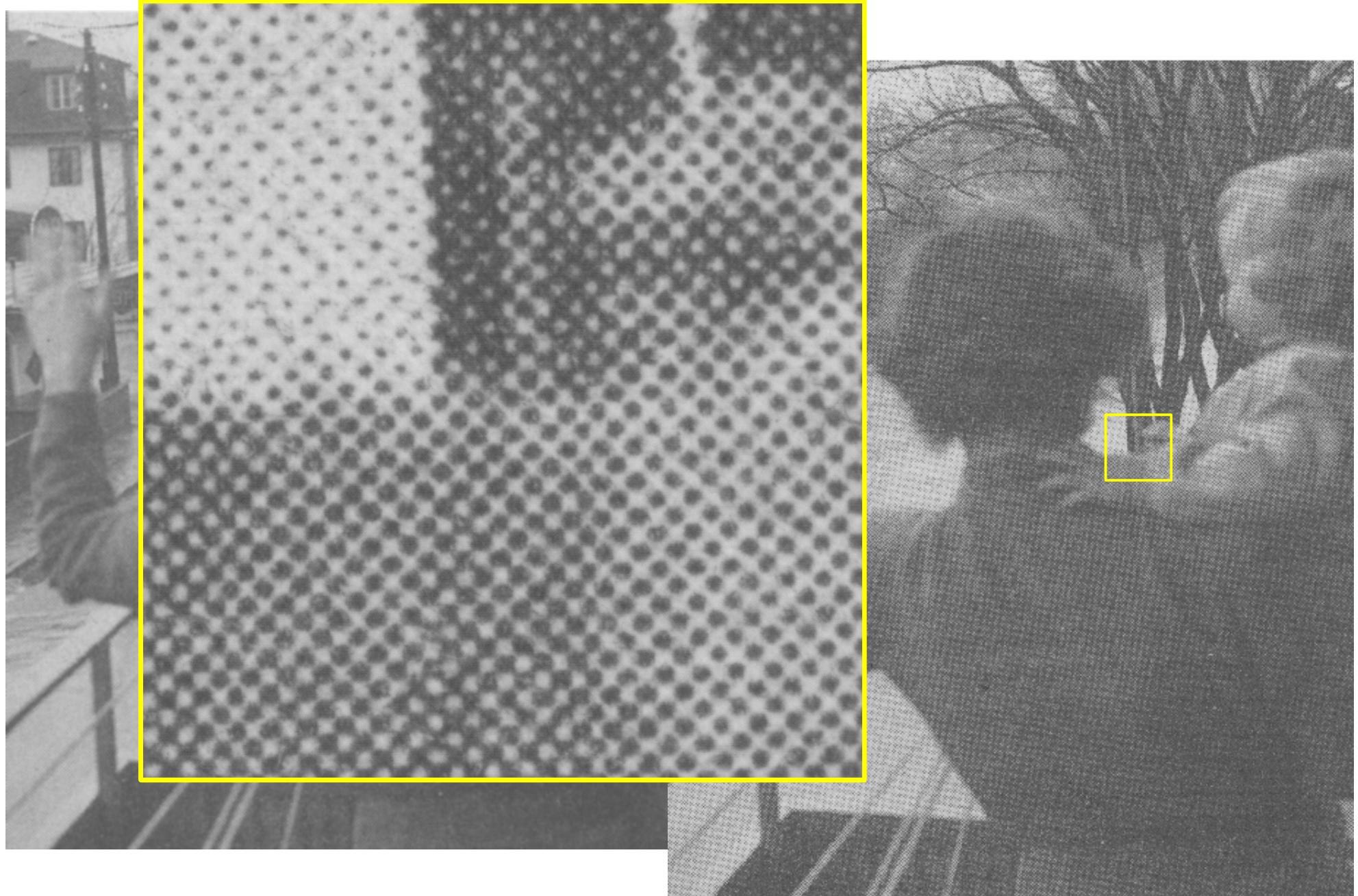
Fourier example



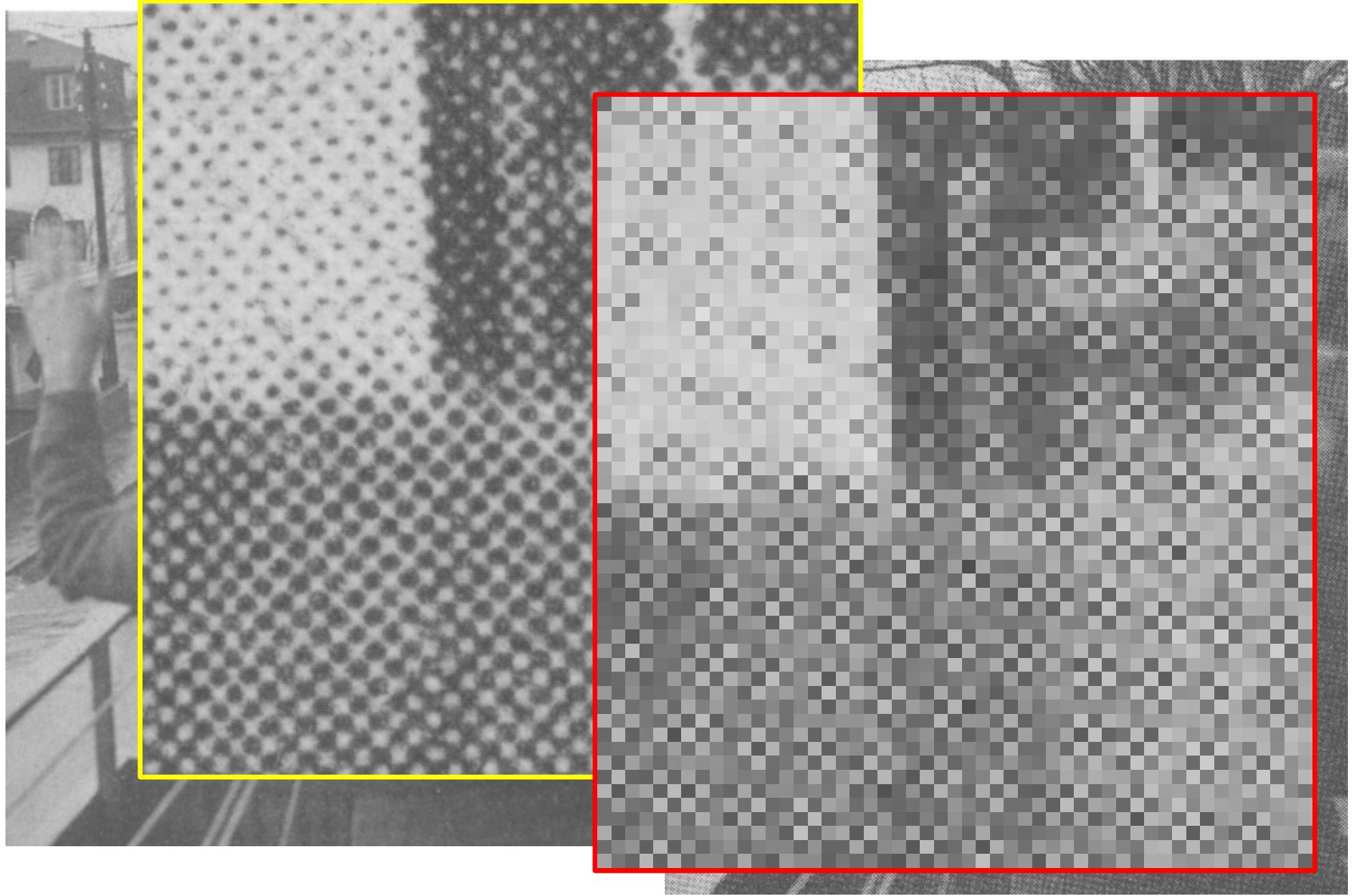
Fourier example



Fourier example



Fourier example



Complex numbers

$$i = \sqrt{-1} \quad \Rightarrow \quad i \cdot i = -1$$

$$x = a + i b$$

$$x^* = a - i b \quad (\text{complex conjugate})$$

$$x \cdot x^* = a^2 + b^2 = \|x\|^2$$

$$\arg x = \arctan\left(\frac{b}{a}\right)$$

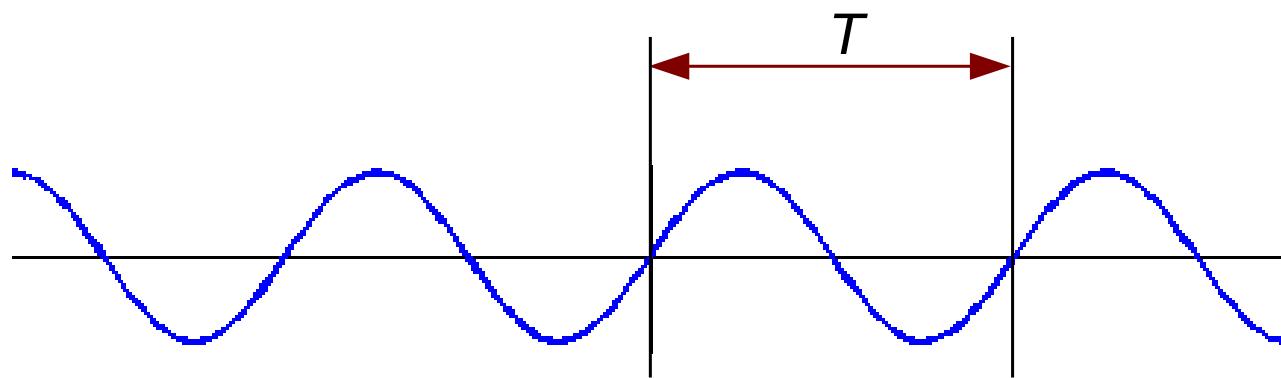
$$e^{i\phi} = \cos \phi + i \sin \phi \quad (\text{Euler's formula})$$

$$x = A e^{i\phi}$$

Fourier basis function

$$e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$$

$$\omega = 2\pi f = \frac{2\pi}{T}$$



Fourier basis function

$$e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$$

$A e^{i\omega x} + A^* e^{-i\omega x}$ is a real-valued function

This is why we need negative frequencies

At frequency ω we have weight A ,
at frequency $-\omega$ we have weight A^* .

Fourier basis function

$A e^{i \omega x} + A^* e^{-i \omega x}$ is a real-valued function

$$e^{i \omega x} = \cos(\omega x) + i \sin(\omega x)$$

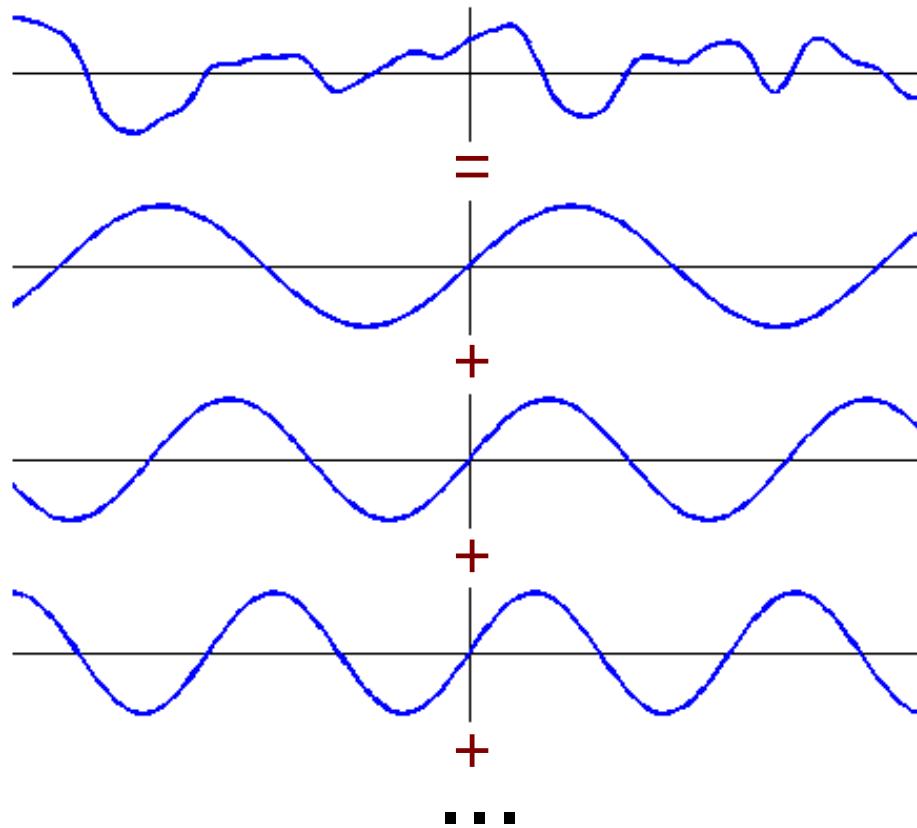
$$A = \|A\| \{ \cos \phi + i \sin \phi \}$$

$$A e^{i \omega x} + A^* e^{-i \omega x} = 2 \|A\| \cos(\omega x + \phi)$$

Fourier transform

$$\mathcal{F}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

$$\begin{aligned} f(x) &= \\ &+ \\ &+ \\ &+ \\ &+ \\ &\dots \end{aligned}$$
$$\mathcal{F}(\omega_1)e^{i\omega_1 x}$$
$$\mathcal{F}(\omega_2)e^{i\omega_2 x}$$
$$\mathcal{F}(\omega_3)e^{i\omega_3 x}$$



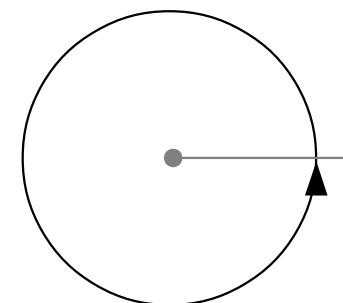
Inverse Fourier transform

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} d\omega$$

$$\omega = 2\pi\mu \Rightarrow f(x) = \int_{-\infty}^{\infty} F(\mu) e^{i2\pi\mu x} d\mu$$

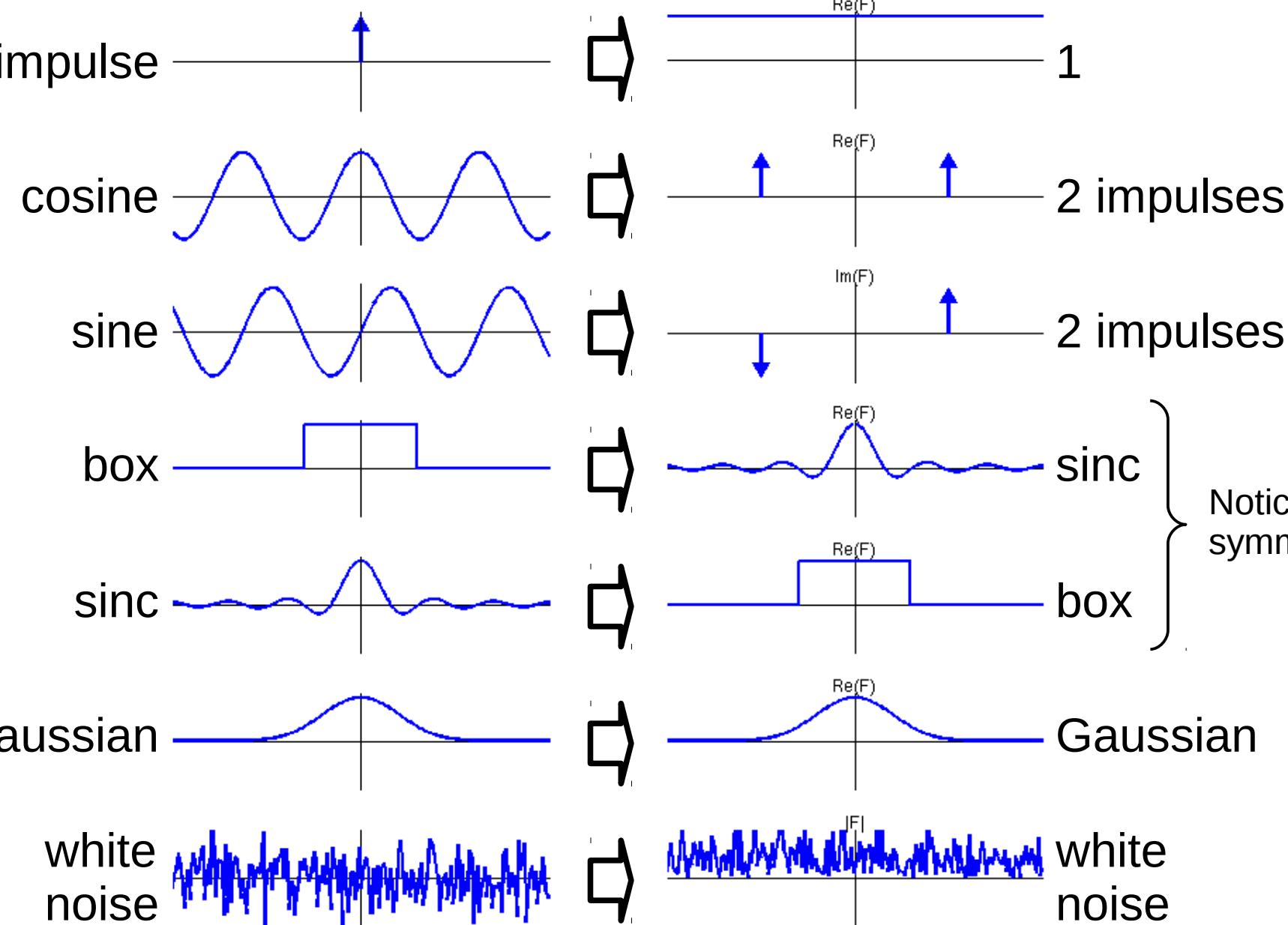
radial frequency
(radian/second)

frequency
(periods/second)



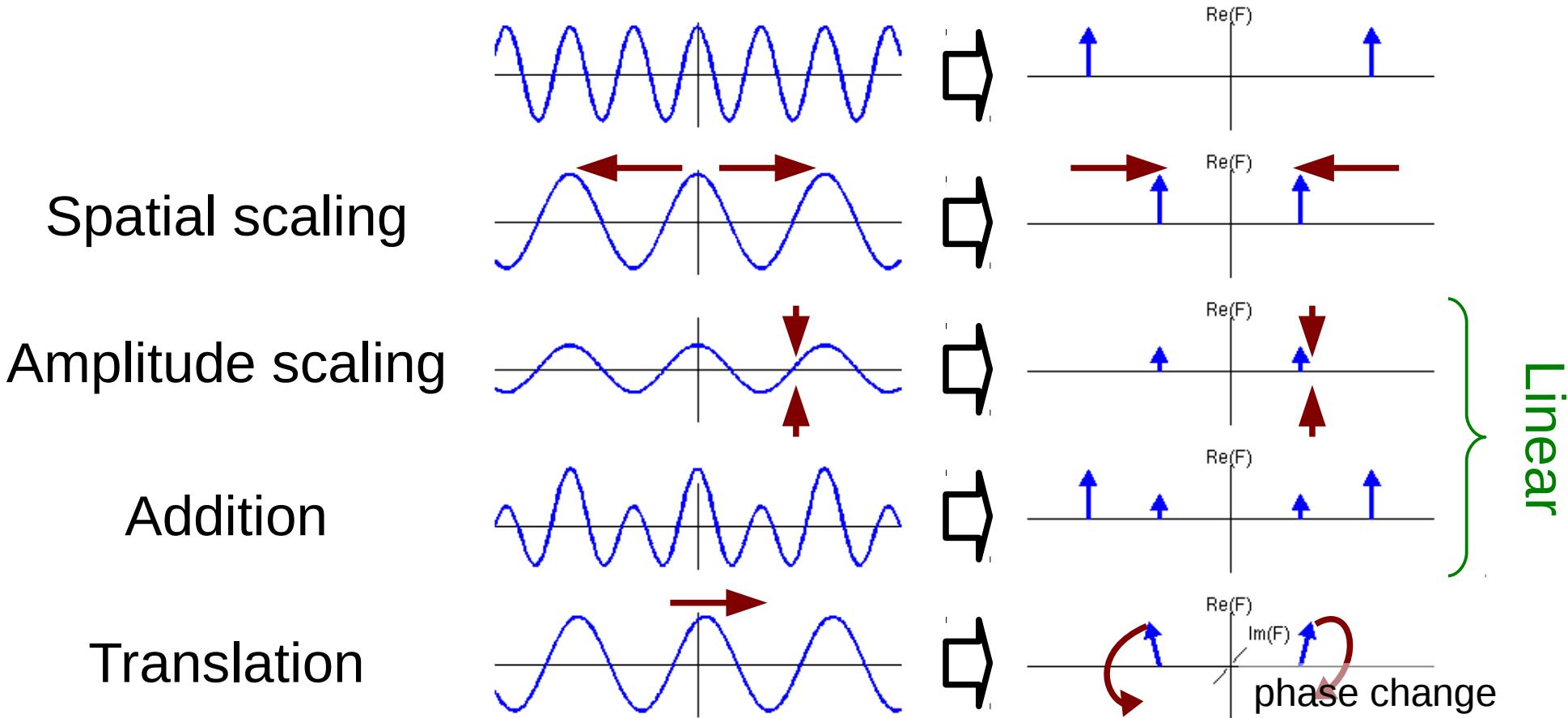
Fourier transform pairs

Spatial



Frequency

Properties of the Fourier transform



$$\mathcal{F}\{f \otimes h\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{h\}$$

$$\mathcal{F}\{f\} \cdot \mathcal{F}\{h\} = \mathcal{F}\{f \otimes h\}$$

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
- The Fourier transform decomposes a function (image) into trigonometric basis functions (sines & cosines).
- The Fourier transform is used to analyse frequency components of an image.

Reading assignment

- Filtering
 - Sections 3.4, 3.5, 3.6, 3.7, 4.2.5, 5.3
- The Fourier transform
 - Section 4.2
- Exercises:
 - 3.15, 3.16, 3.19, 3.23, 3.27, 3.28, 4.3

