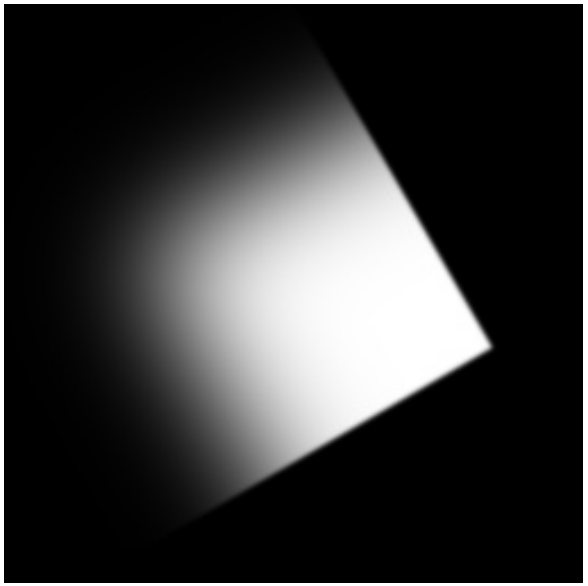
Today's lecture: filtering I

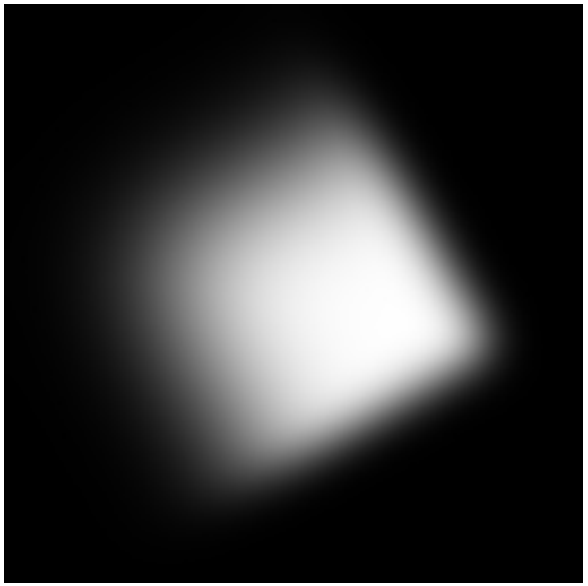
- Smoothing:
  - Low-pass filtering (linear)
  - Non-linear smoothing
- 1st order derivatives:
  - Linear filters
  - Enhance/detect edges
- 2nd order derivatives:
  - Linear filters
  - Enhance/detect lines
- Normalised convolution
- Next lecture: detection & analysis

Filtering properties

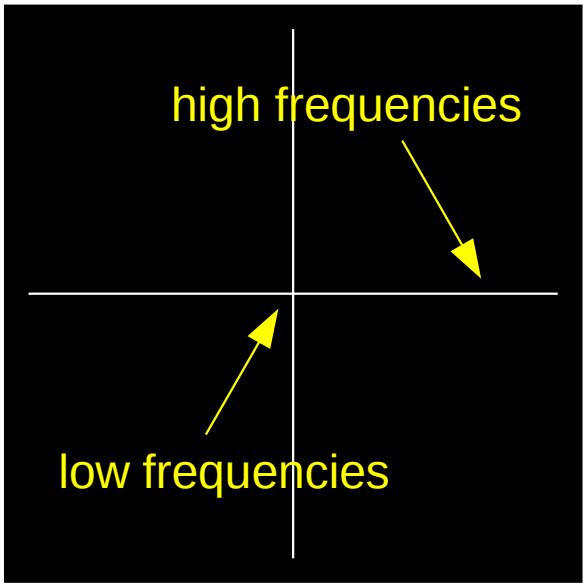
- Shift invariant:
  - The result of the filter is independent of location within the image
- Rotation invariant:
  - The result of the filter should be independent of the orientation of the image w.r.t. the axes
- These two rules make the result dependent only on the object being imaged, not on the exact positioning of the imaging system

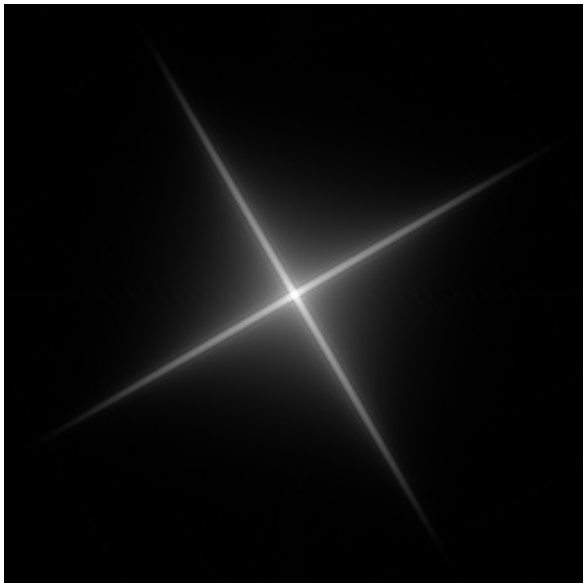
Low-pass filters

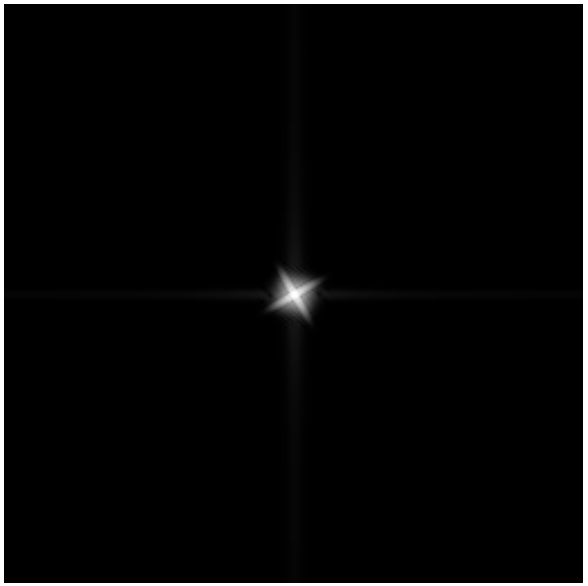




spatial domain





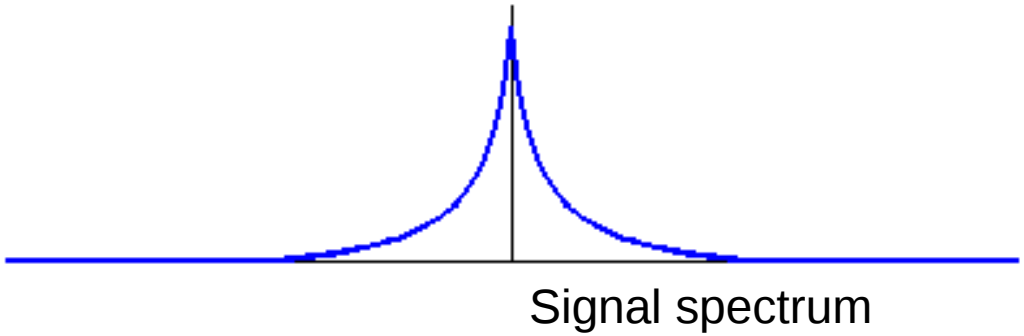


frequency domain

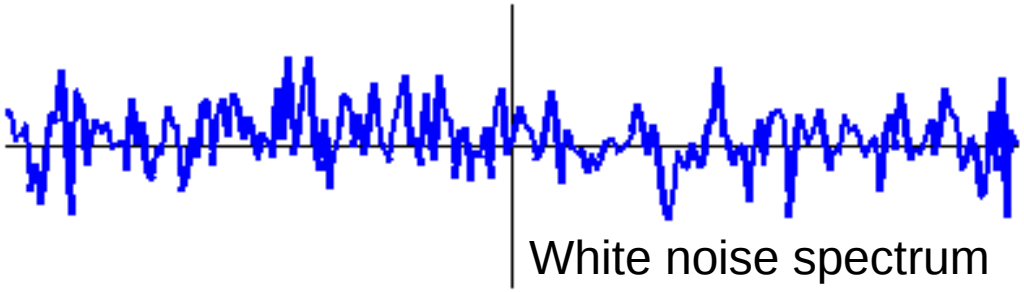
input

low-pass filtered

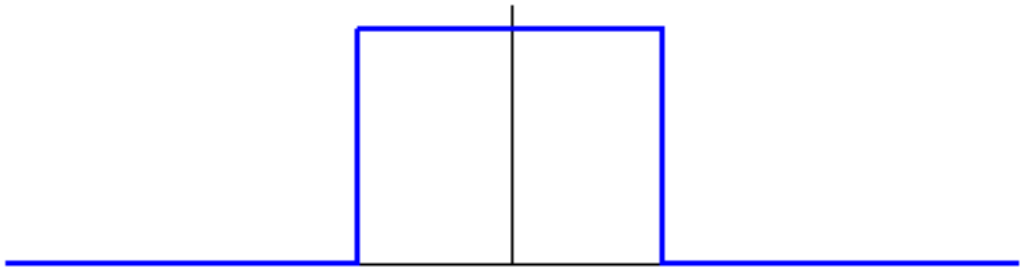
Application: noise reduction



Signal spectrum



White noise spectrum

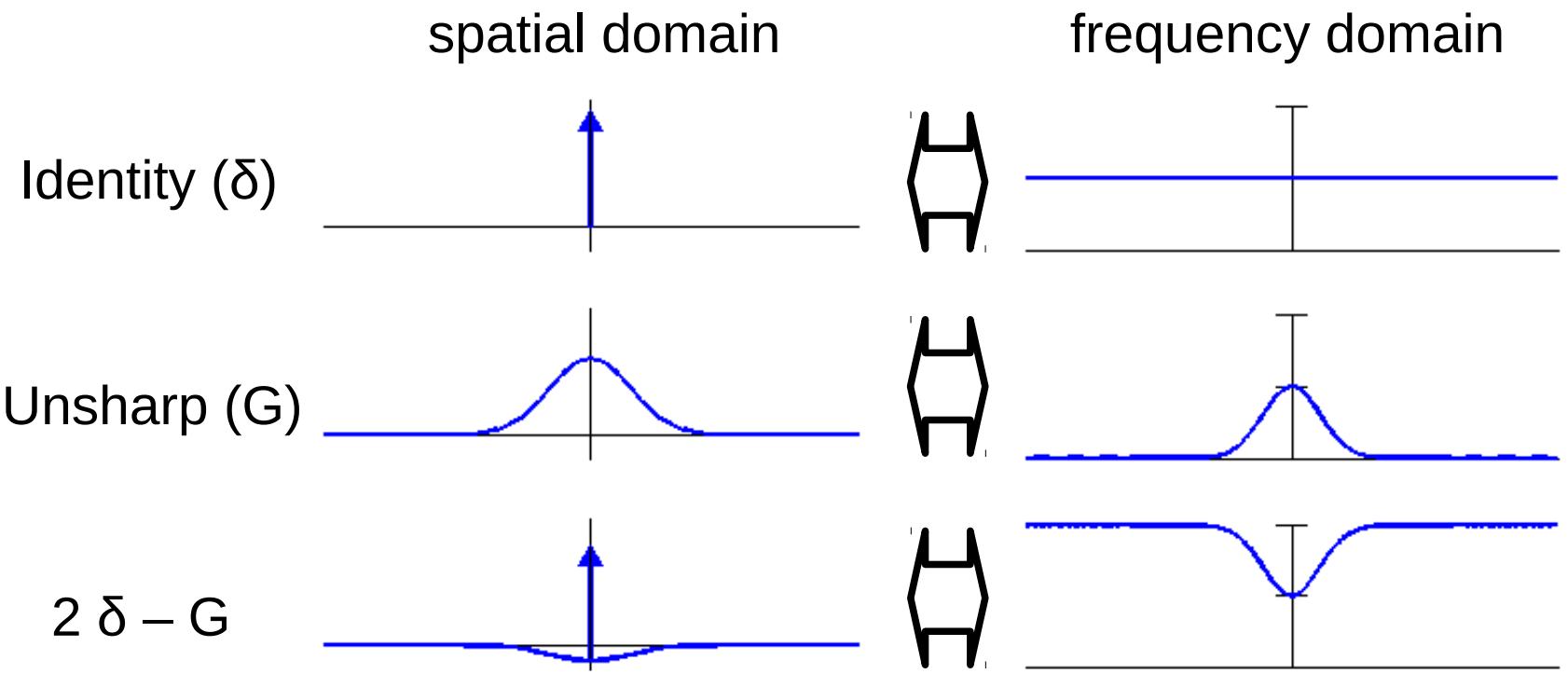


Low-pass filter

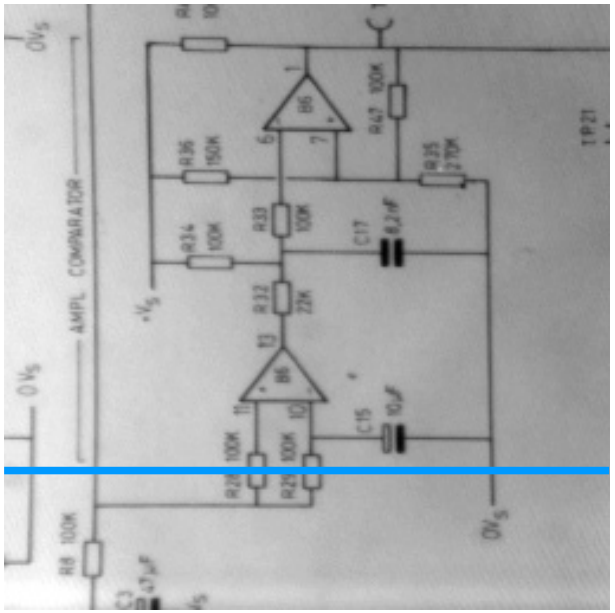
Application: unsharp masking

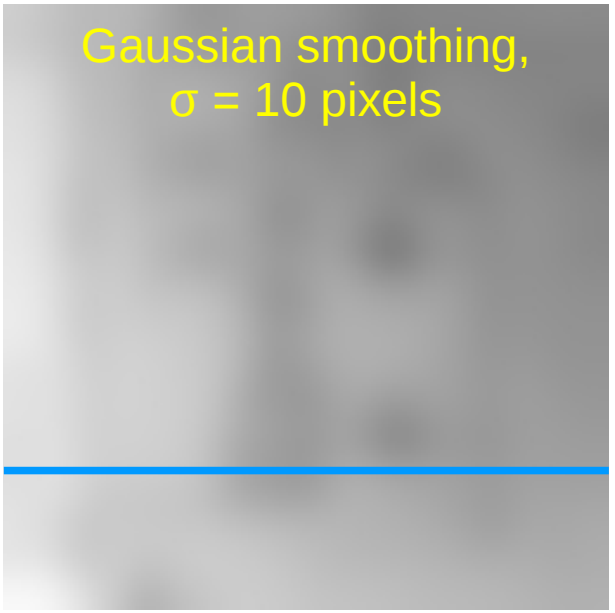
Application: unsharp masking

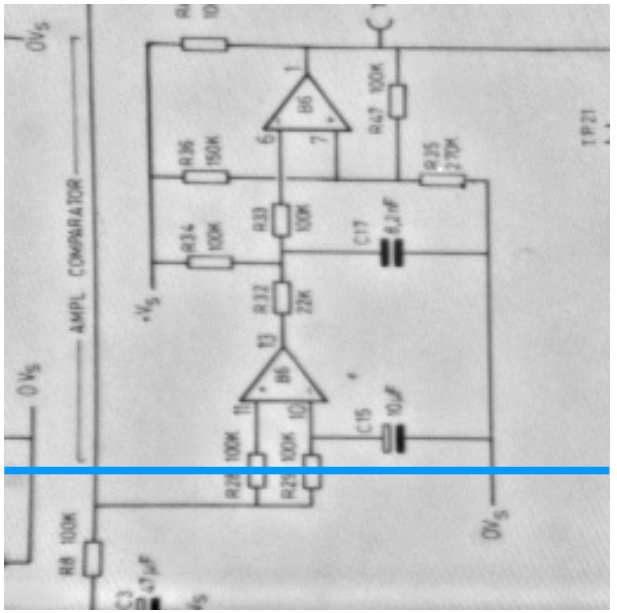
- Subtract smoothed image from original image
- Darkroom technique: implemented by projecting out-of-focus image onto a negative, then using the two negatives together

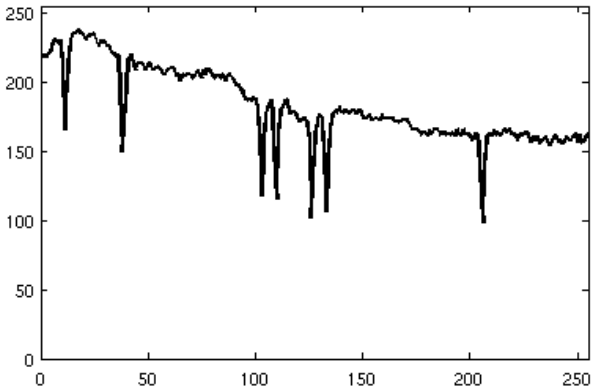


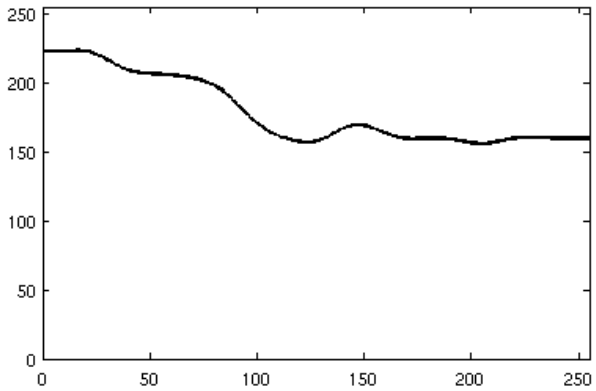
Application: shading correction

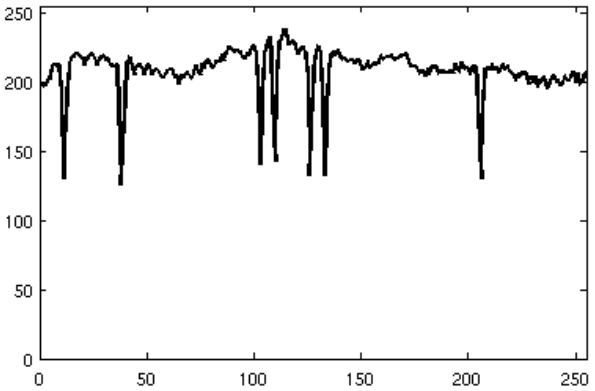








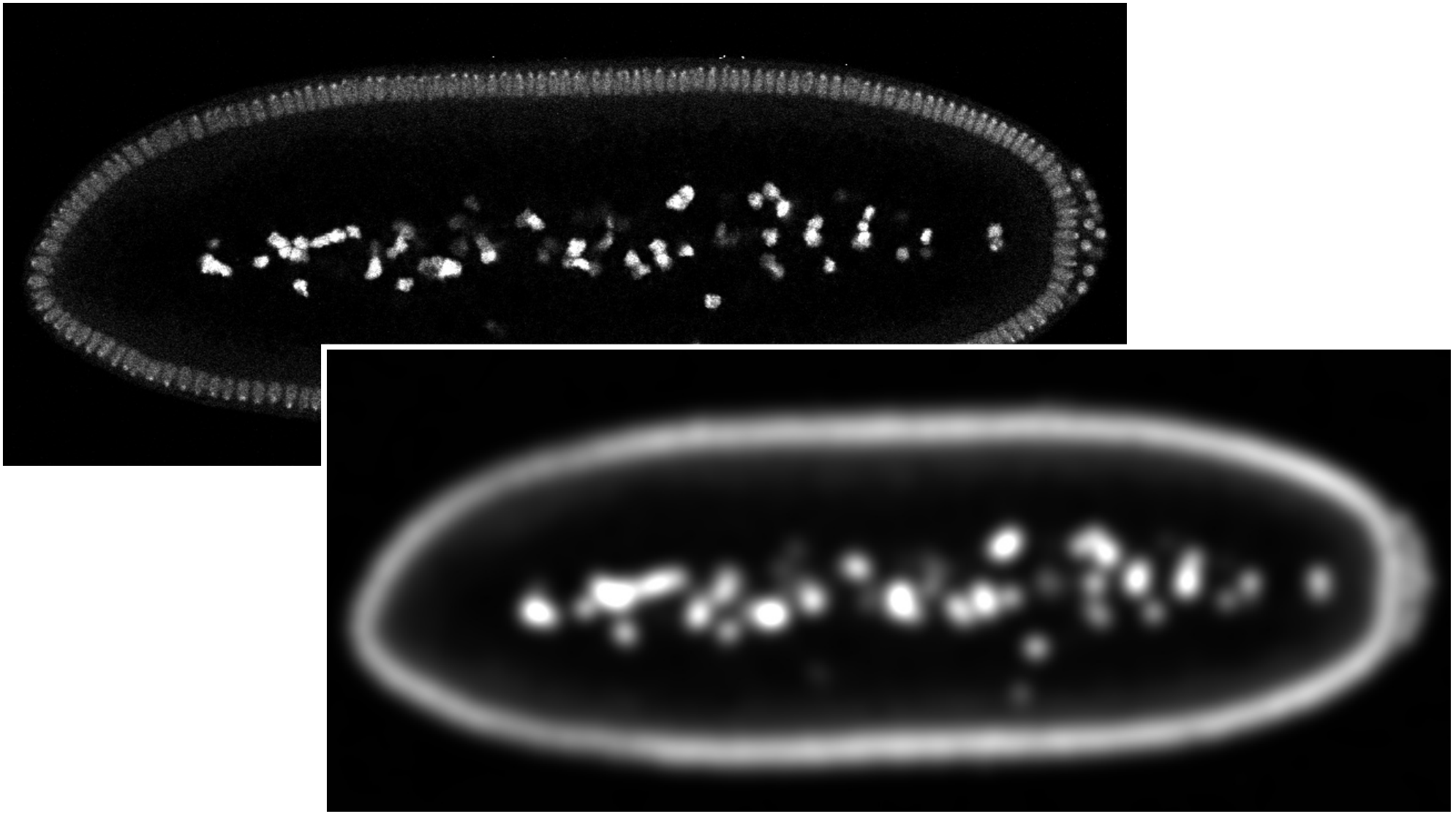




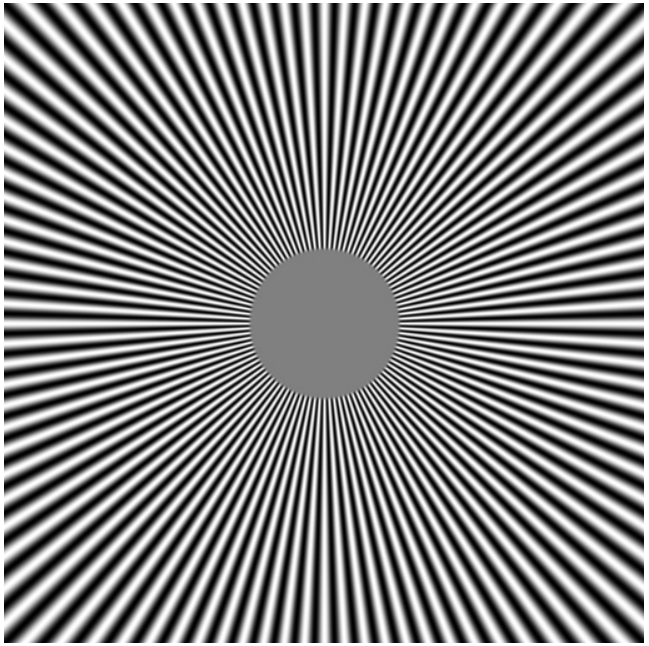
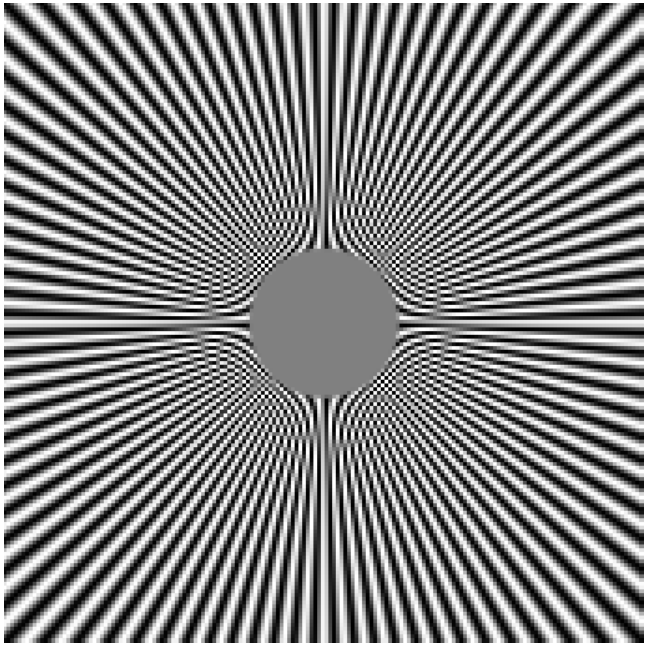
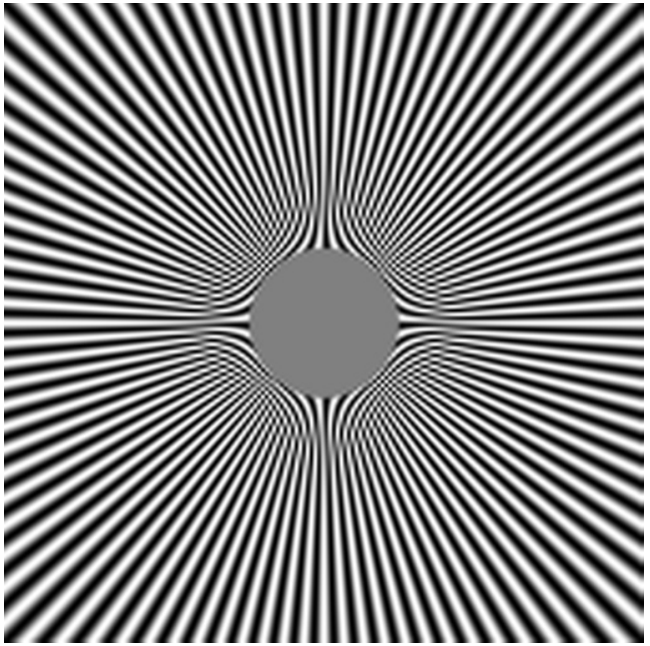
Subtract the background or divide by the background, depending on the imaging model.

Application: abstraction

- Sometimes you just don’t want all those details...

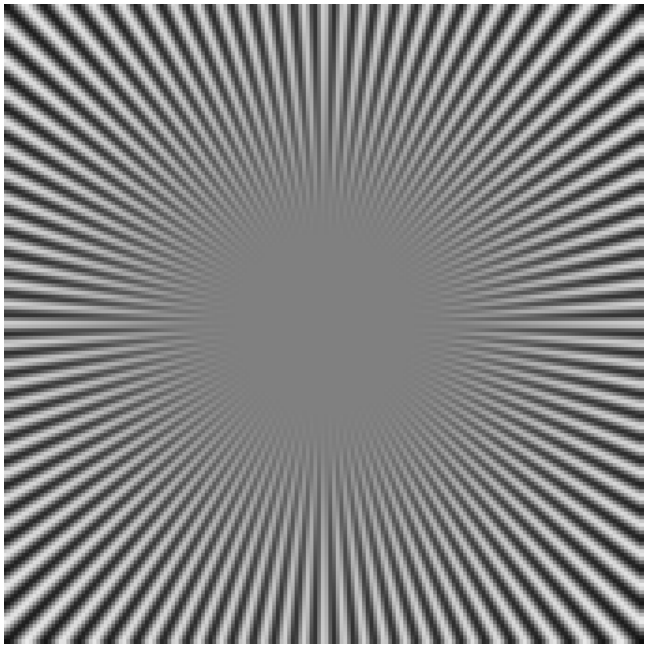
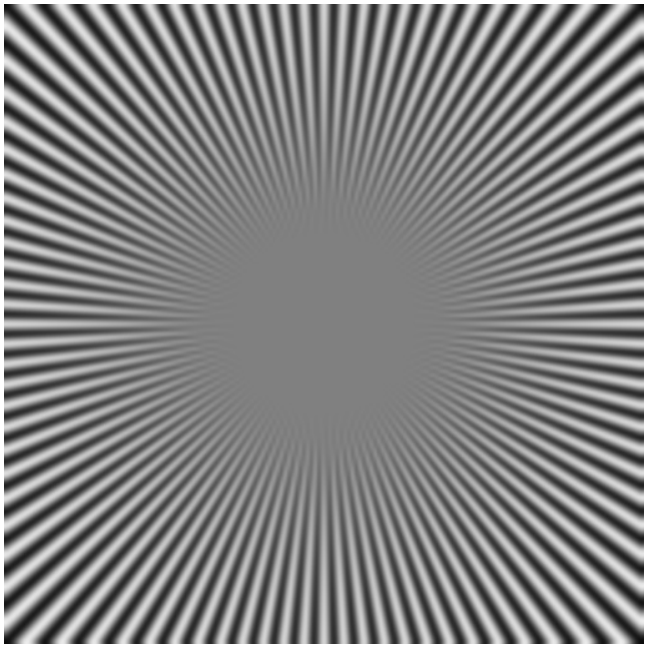


Application: down-sampling

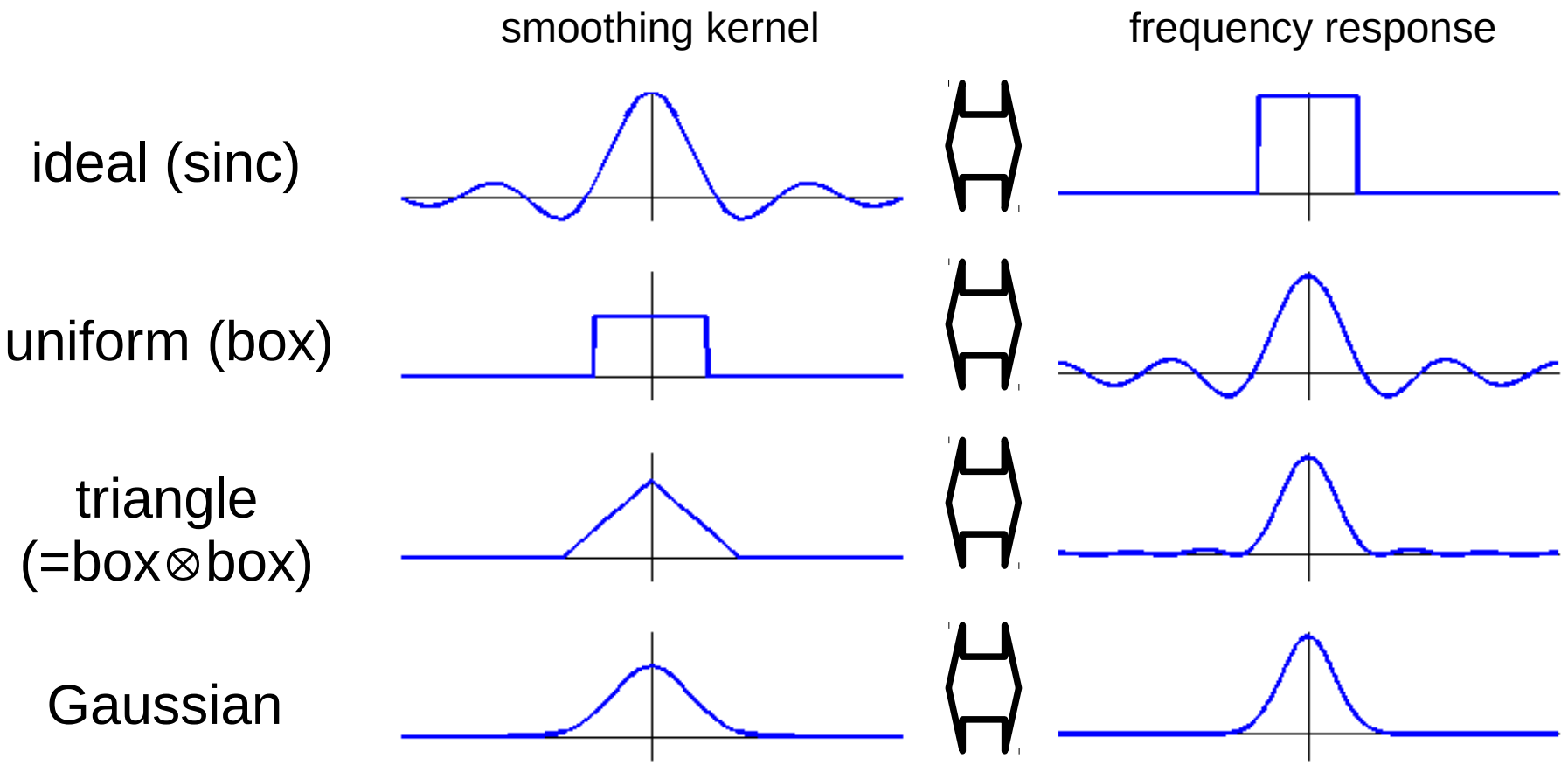
Low-pass filter before down-sampling to avoid aliasing:

Filtering removes high frequencies that cannot be represented by lower sampling rate

Linear smoothing in 1-D

- Low-pass filtering: removing high-frequency components

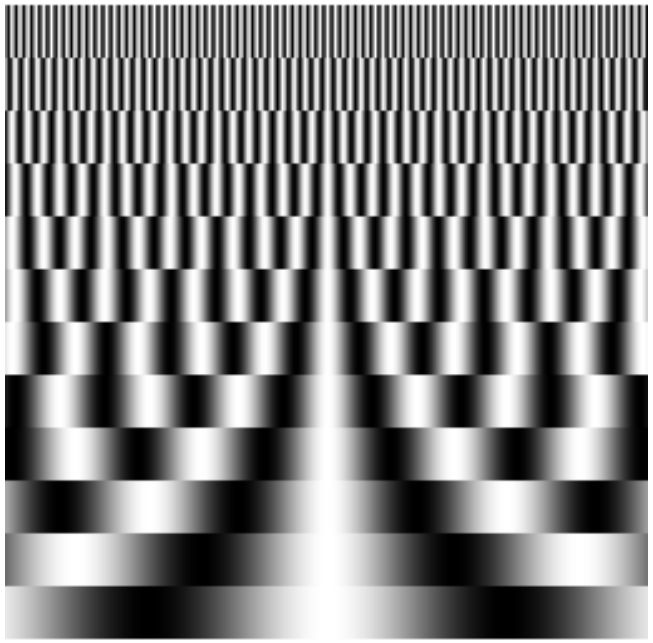


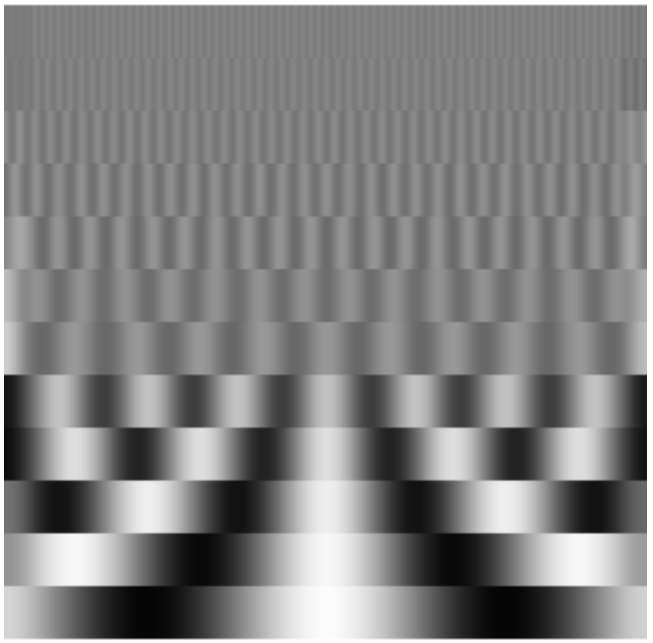
(note: book pg. 139, eq. 5.47 needs normalization: $1/2\pi\sigma^2$ in 2D)

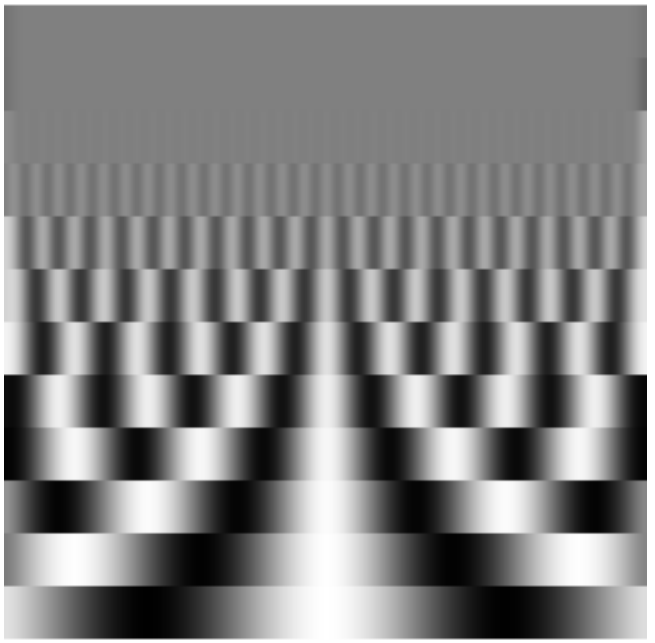
Phase reversal of uniform filter

uniform

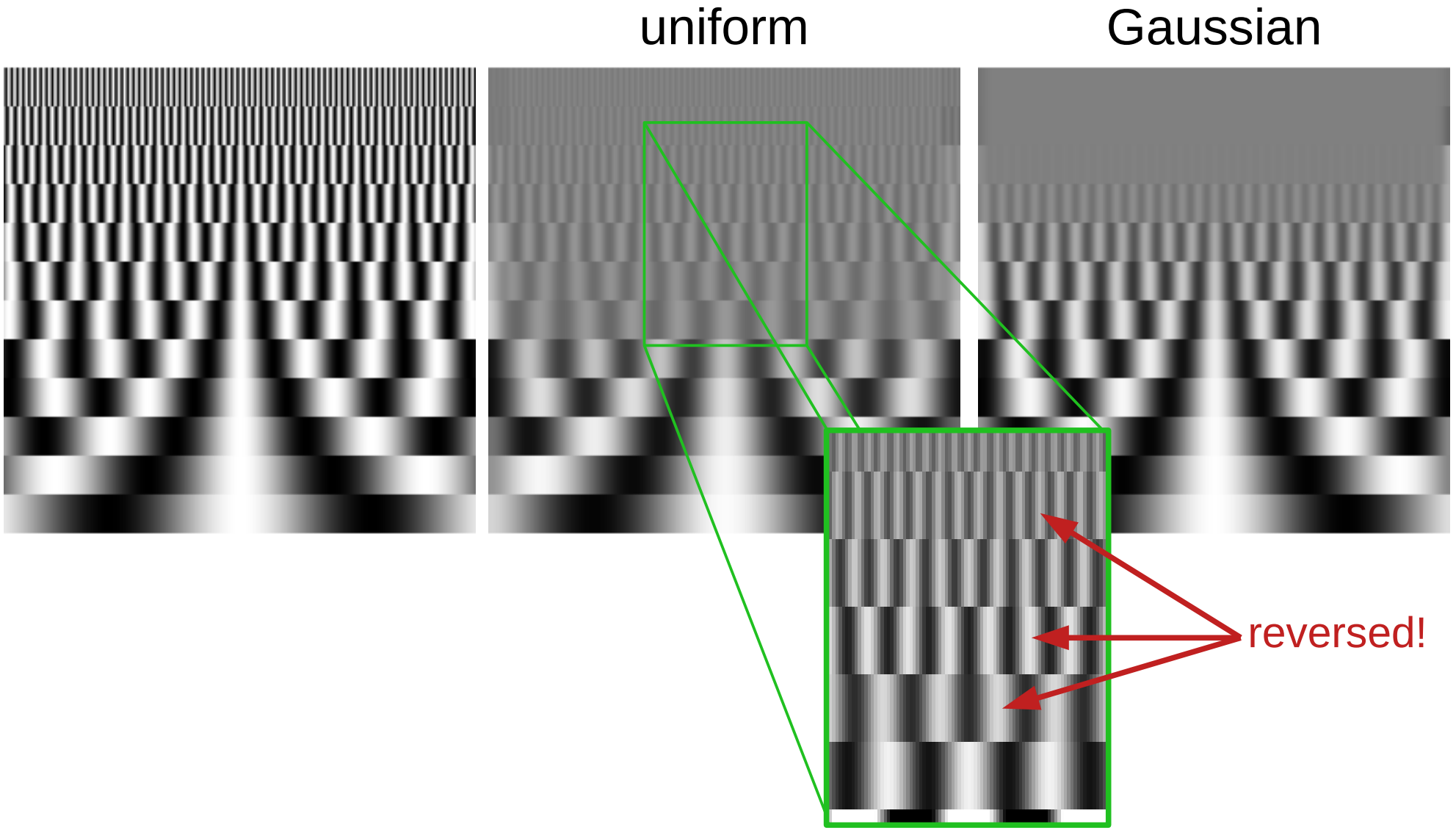
Gaussian







Phase reversal of uniform filter



Central limit theorem

box ⊗ box ⊗ box ⊗ box ⊗ box ⊗ ... = Gaussian

Gaussian ⊗ Gaussian = Gaussian

with increased sigma!

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$$

Other linear low-pass filters

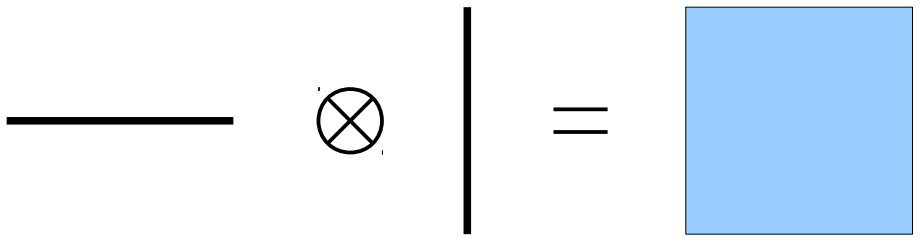
- Square neighbourhood, significance of centre pixel increased:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \frac{1}{10} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \frac{1}{16}$$

  - Book says: “it better approximates the properties of noise with a Gaussian probability distribution.” (pg. 125) **Wrong!**
  - You now now why these filters are used!
- The Butterworth Filter
  - Compare: Chebyshev, Elliptic, etc.
  - Designed for electric circuitry
  - But: electric circuitry has different constraints!
  - Has no purpose in digital signal/image processing

Linear smoothing in *n*-D

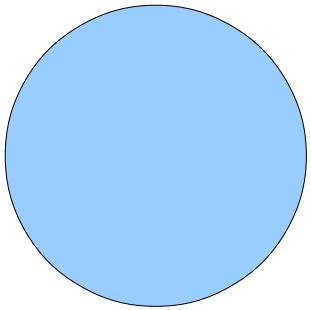
- Separable filter: $h = h_x \otimes h_y$
- Convolution is associative: $f \otimes \left\{ h_x \otimes h_y \right\} = \left\{ f \otimes h_x \right\} \otimes h_y$
- Separable filters are faster & easier to implement



the square uniform kernel is separable

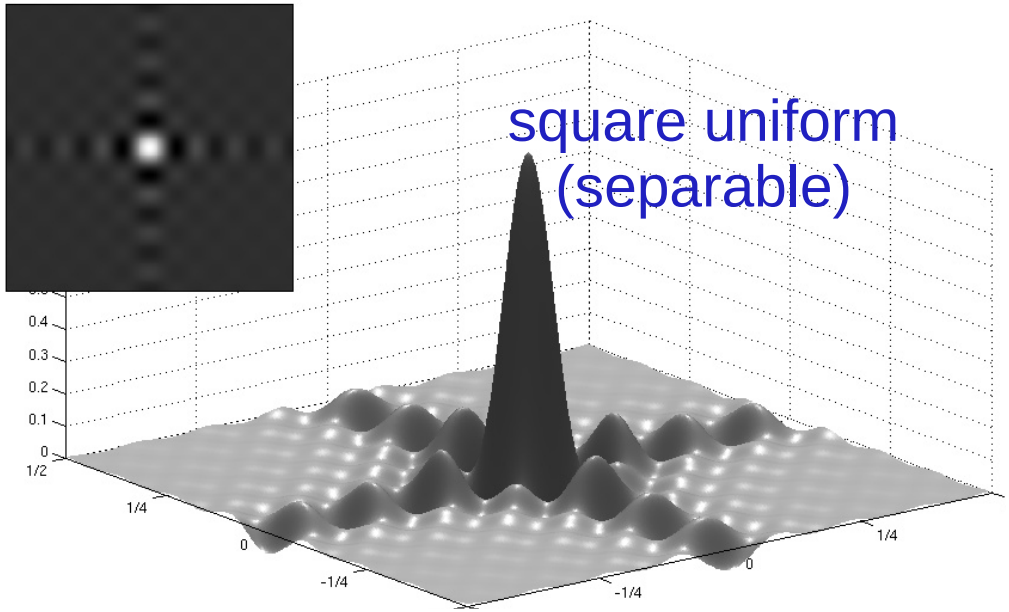
$$G(x) \otimes G(y) = G(x,y)$$

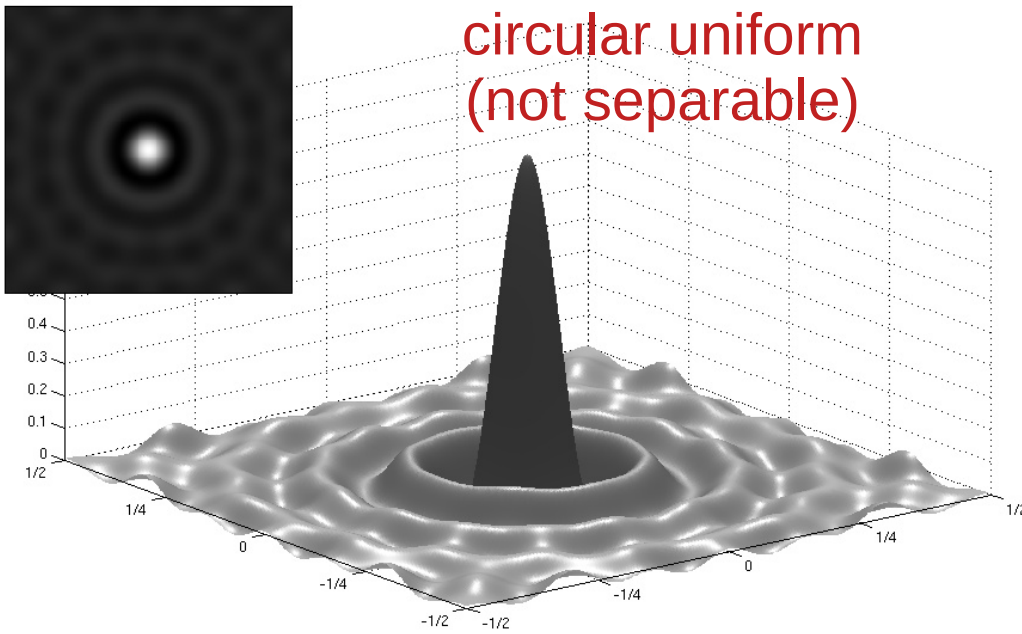
the Gaussian kernel is separable

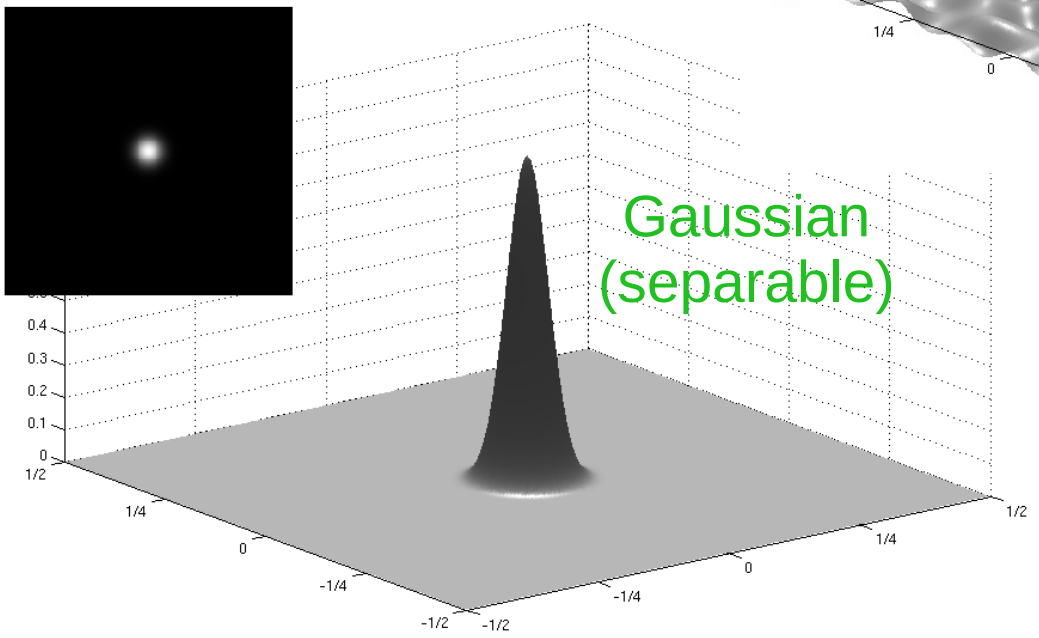


the circular uniform kernel is not separable

Linear smoothing in *n*-D

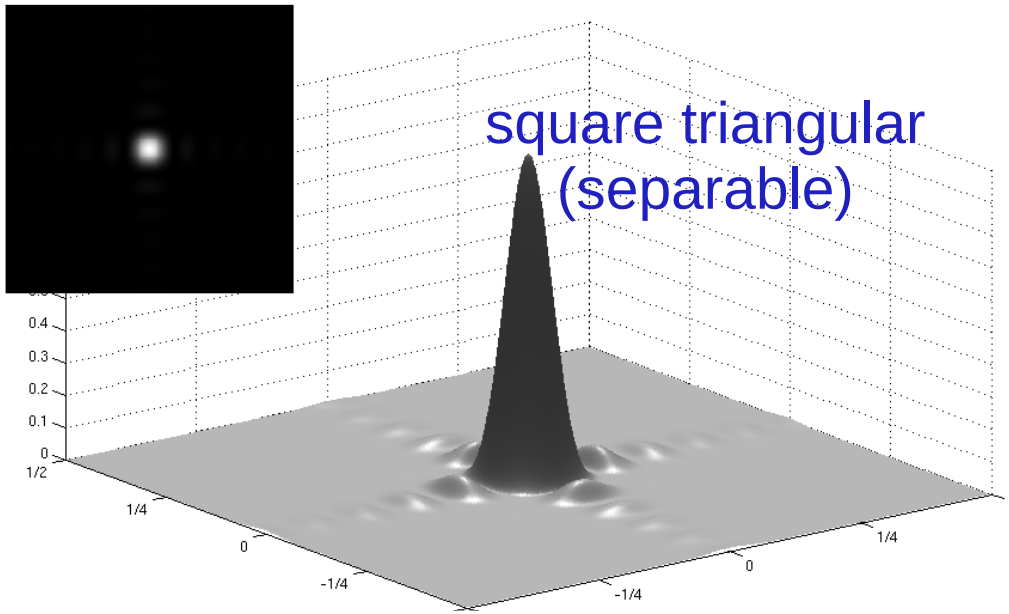


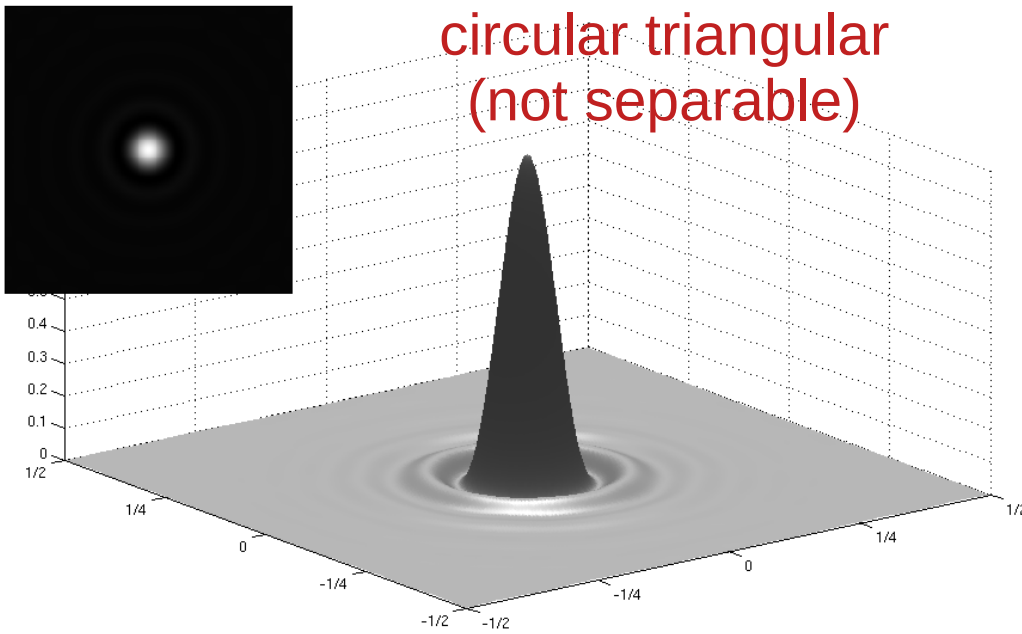


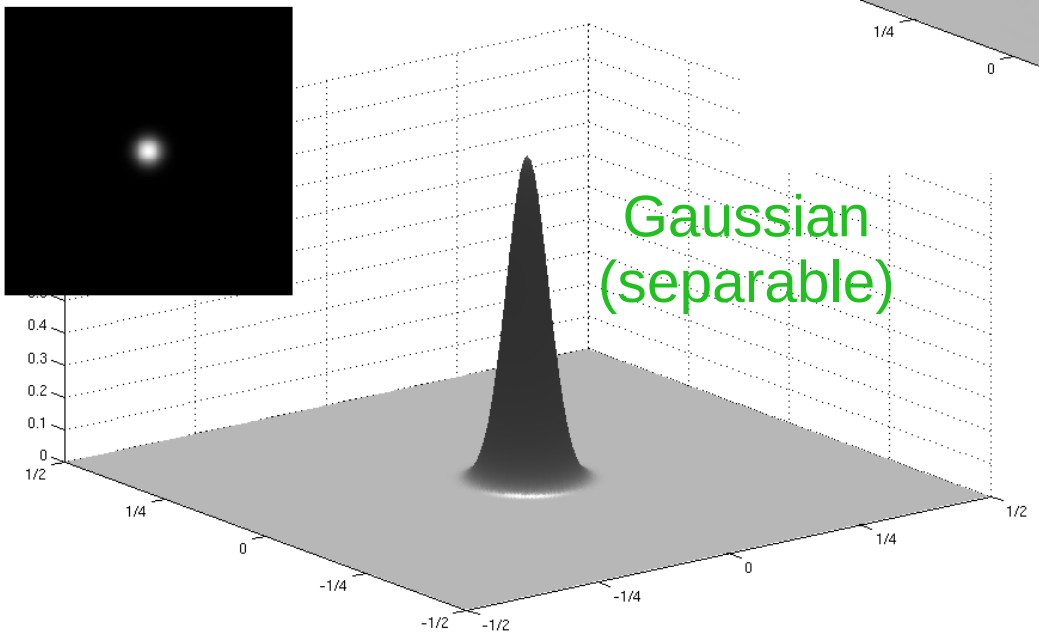


Frequency response

Linear smoothing in *n*-D

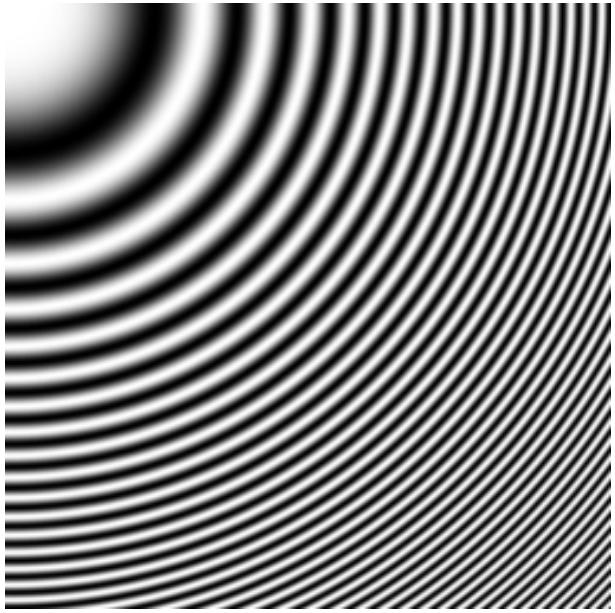




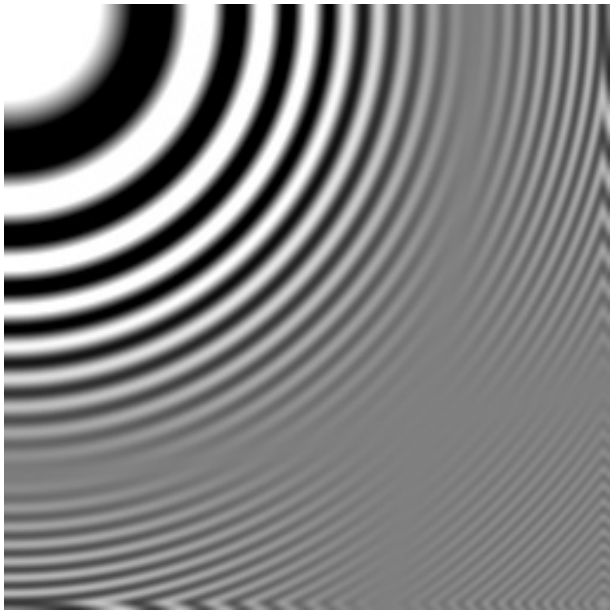


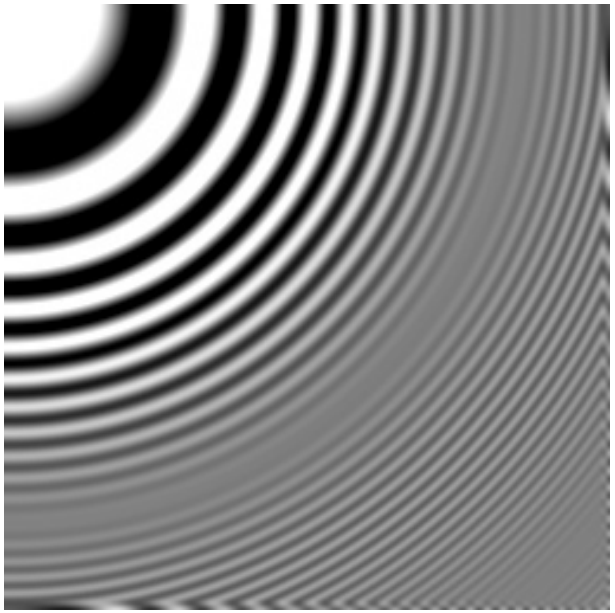
Frequency response

Linear smoothing in *n*-D



Gaussian filter
σ = 3



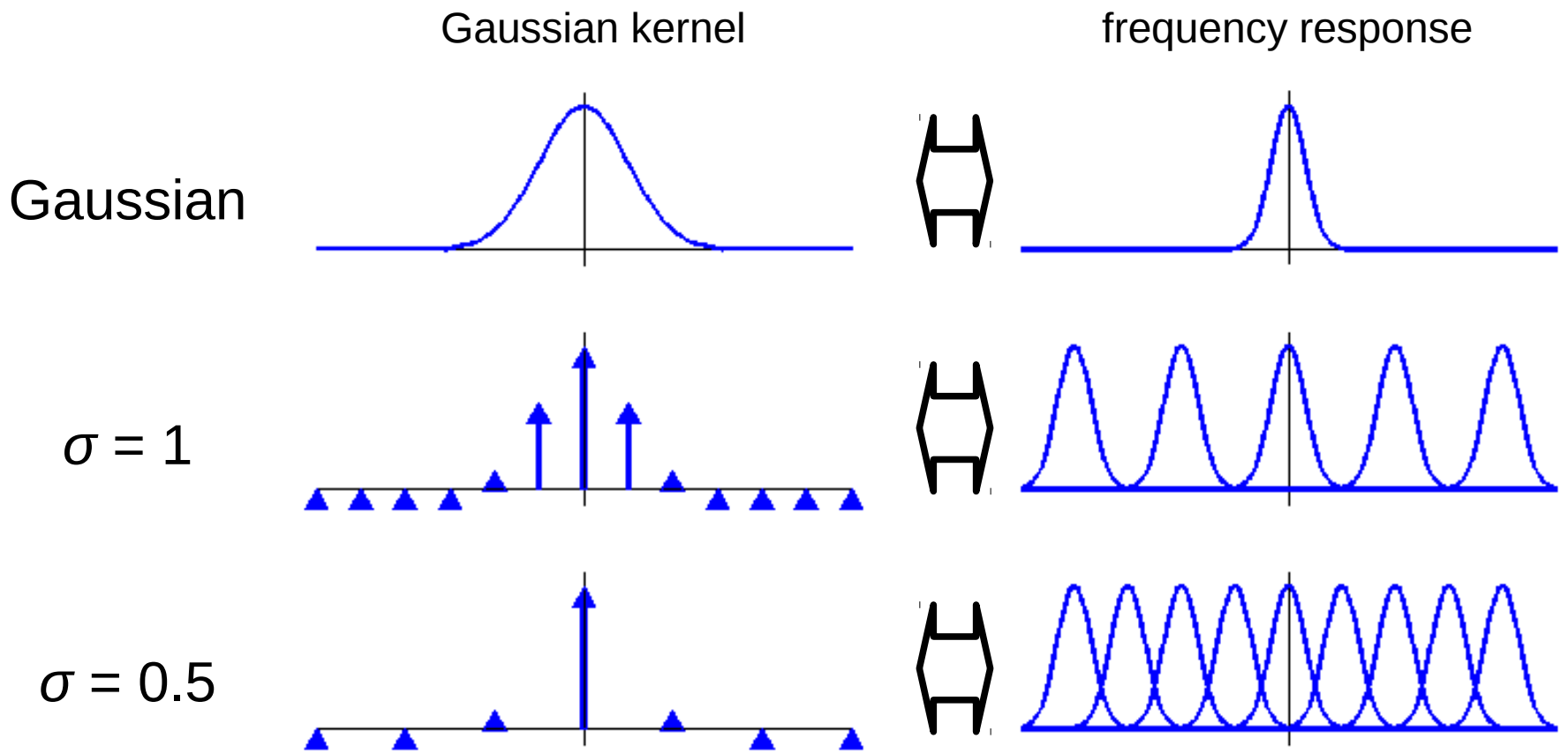


Uniform
circle d = 9

Uniform
square s = 9

Discretizing the Gaussian

- "Proper" sampling occurs for sampling period <= σ
- Thus: σ >= 1 for proper sampling of Gaussian kernel
  - Some people say σ >= 0.8



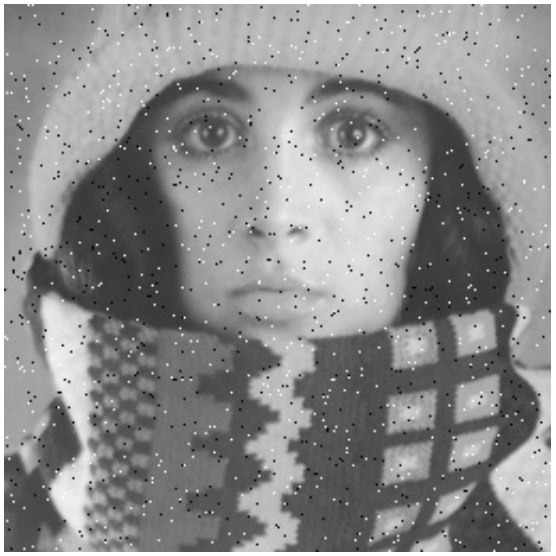
Median filter

- Takes the median of the values in some neighbourhood
- Not separable
- Median is an estimator for the mean
  - Better than mean (linear filters) for some noise models
  - Conserves edges slightly better than linear filters
- Generalizes to percentile filter
  - Median is 50%
  - 0% is min filter (= erosion)
  - 100% is max filter (= dilation)
  - Other useful filters: 5%, 95% (like min and max, but less sensitive to noise)

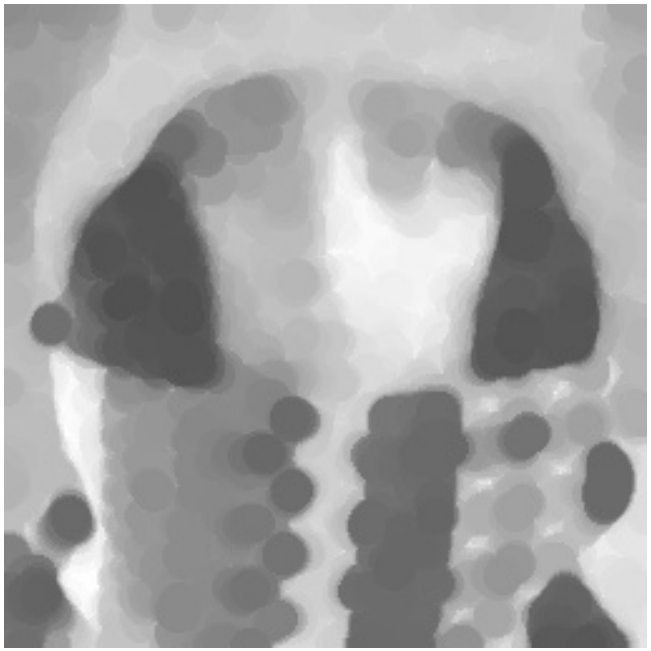
Median statistics

	one input image	mean of 5 images	median of 5 images
Normally distributed noise			
Salt & pepper noise			

Median statistics

	input image	3x3 uniform filter	3x3 median filter
Normally distributed noise			
Salt & pepper noise			

Max-min and min-max filter



max-min
(=closing)

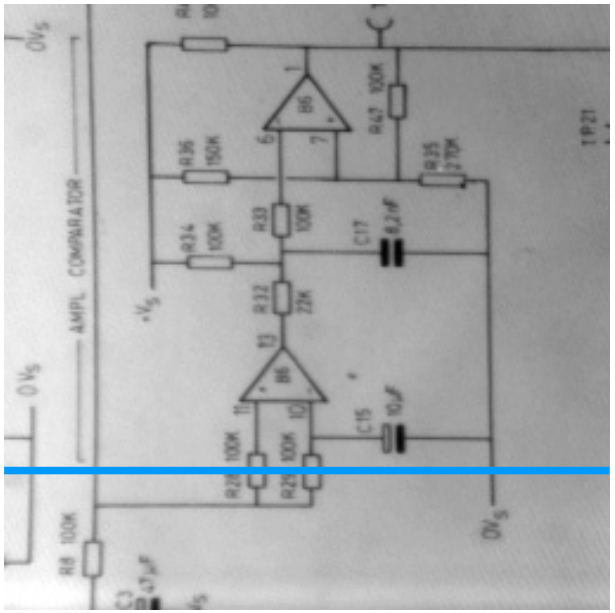
min-max
(=opening)

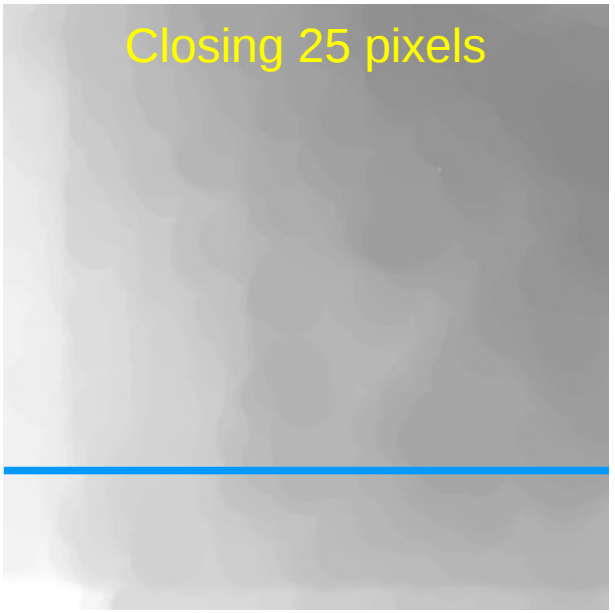
(More on this in Ida-Maria's lecture on Mathematical morphology)

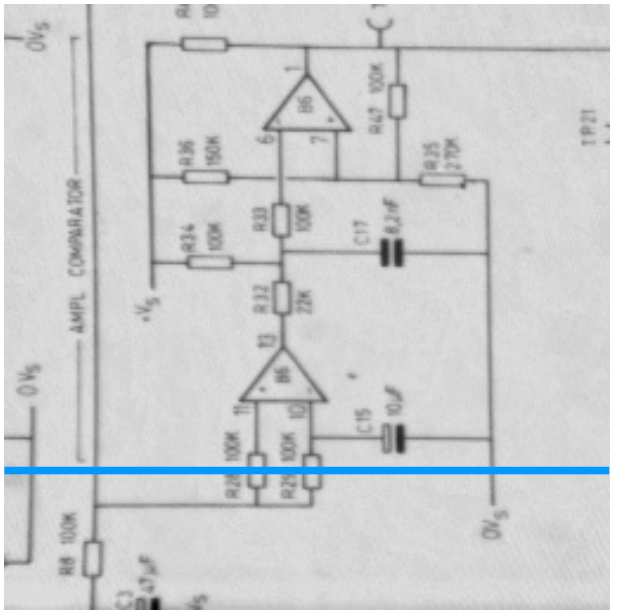
Sequence of max and min filters

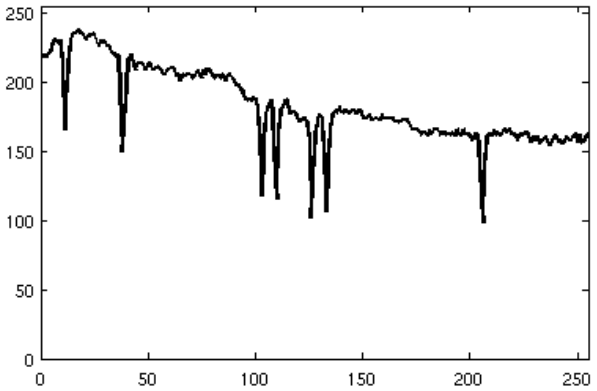
- max min min max f ≈ min max max min f
- Removes local maxima and minima
- Apply first with very small neighbourhood
- Apply repeatedly with increasing size

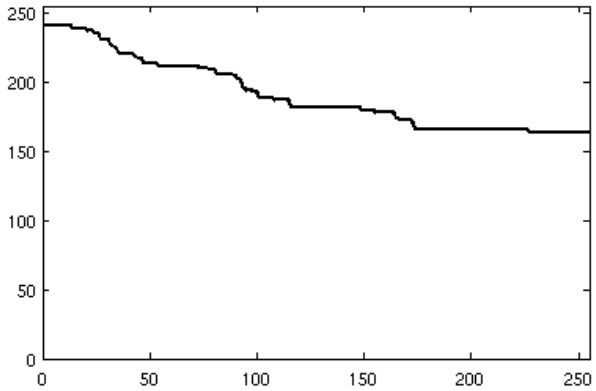
Application: shading correction

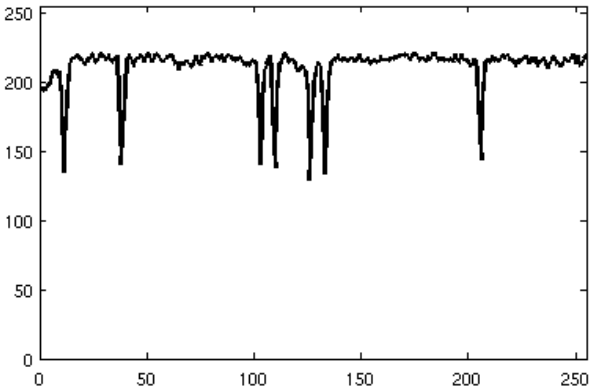






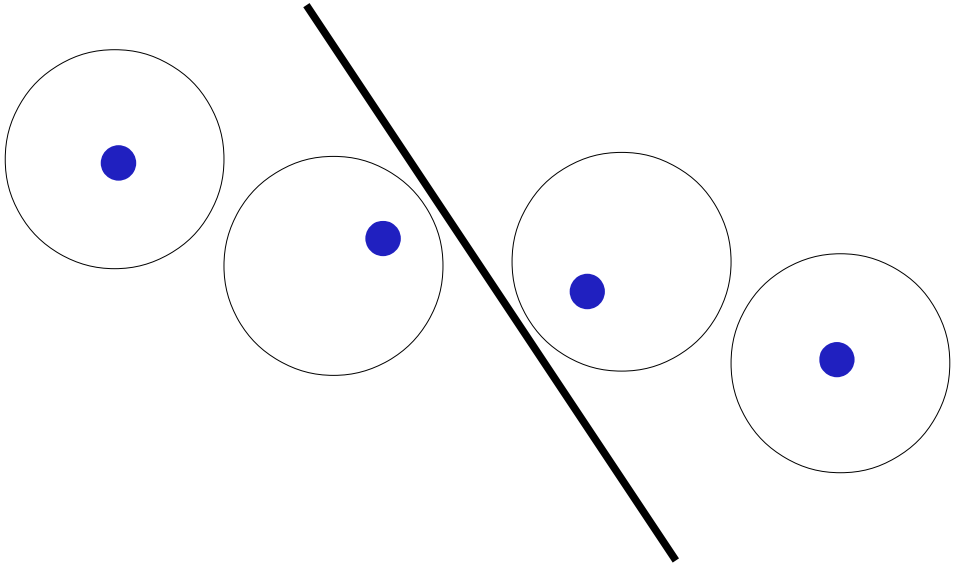






Kuwahara / Nagao filter

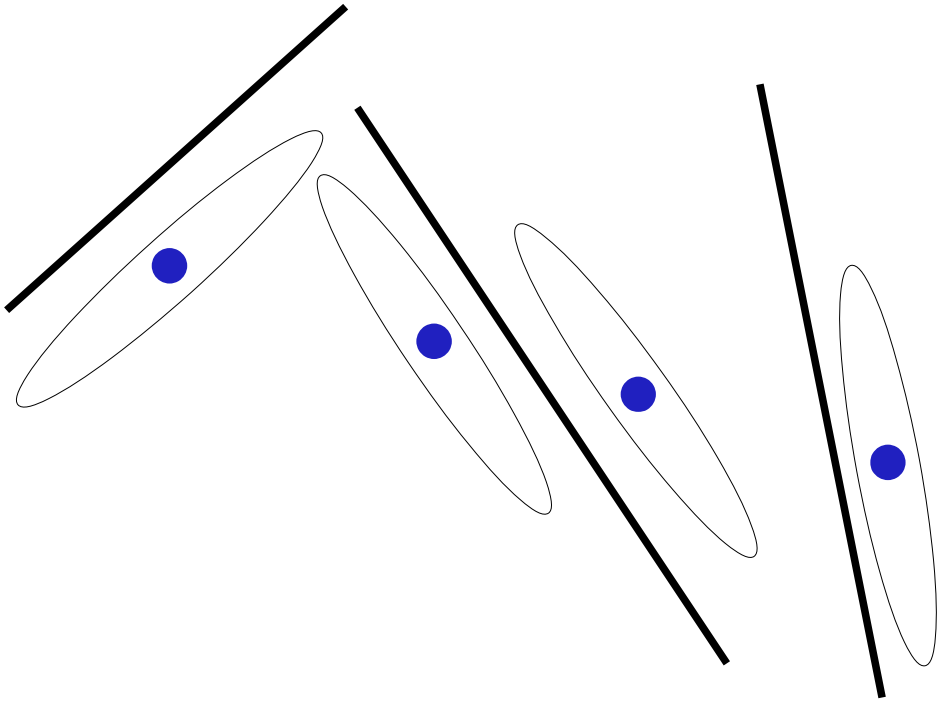
- Mean in neighbourhood
- The neighbourhood shifts for each pixel
  - Neighbourhood with minimum variance chosen
- Does not average across edges



In book: “averaging using rotating mask”

Adaptive filters

- Generalization of Kuwahara / Nagao
- We can turn or grow and shrink our neighbourhood
- Many, many ways of directing the filter



Gaussian (3x0.8) rotated to align to local contours

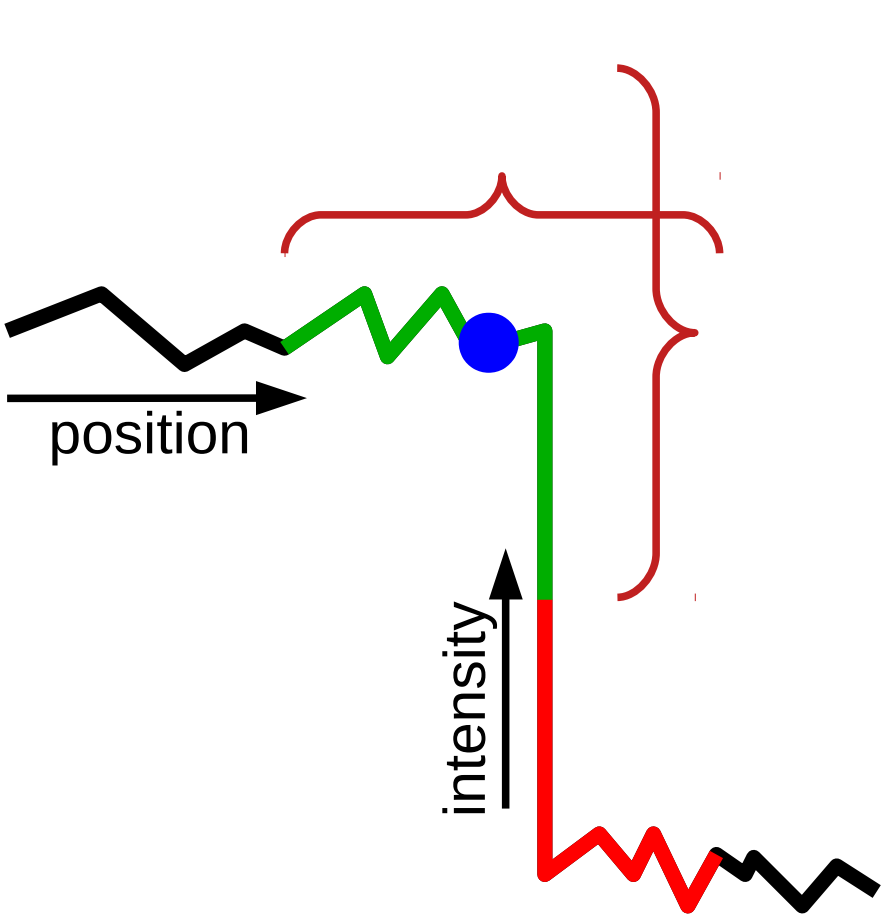
Sigma filter

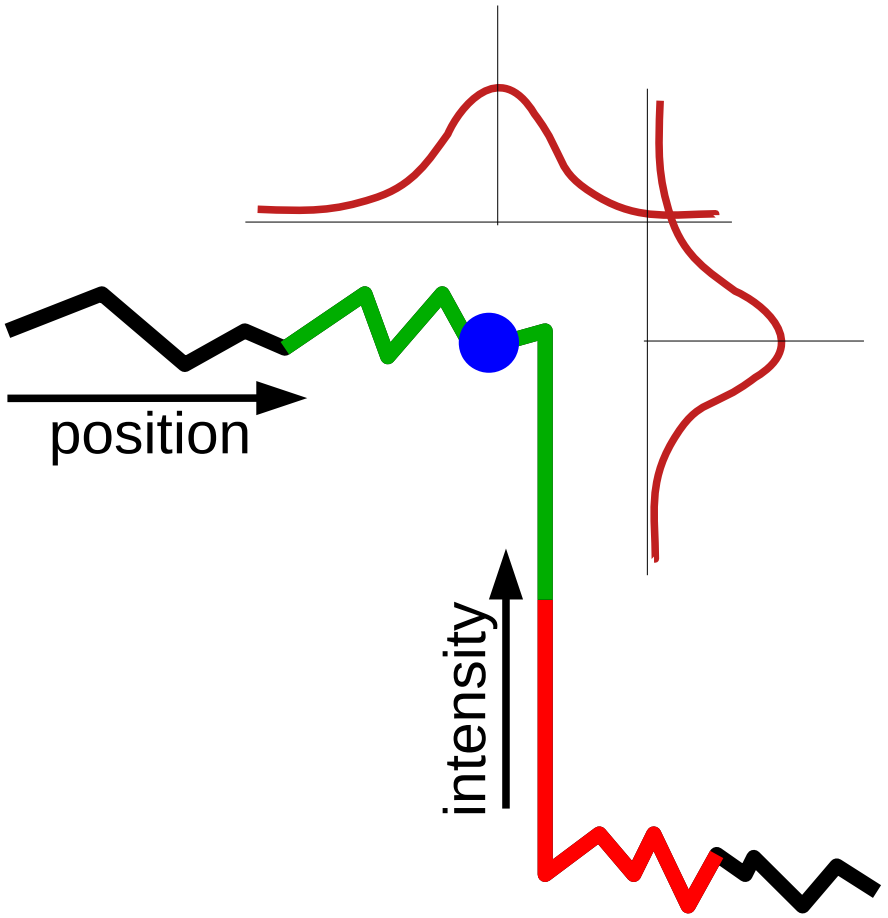
- Computes mean in neighbourhood, like uniform filter
- But: excludes pixels that differ more than *s* from the central value

7-pixel circular uniform filter

7-pixel circular uniform sigma filter, with *s*=20

Generalizing the sigma filter

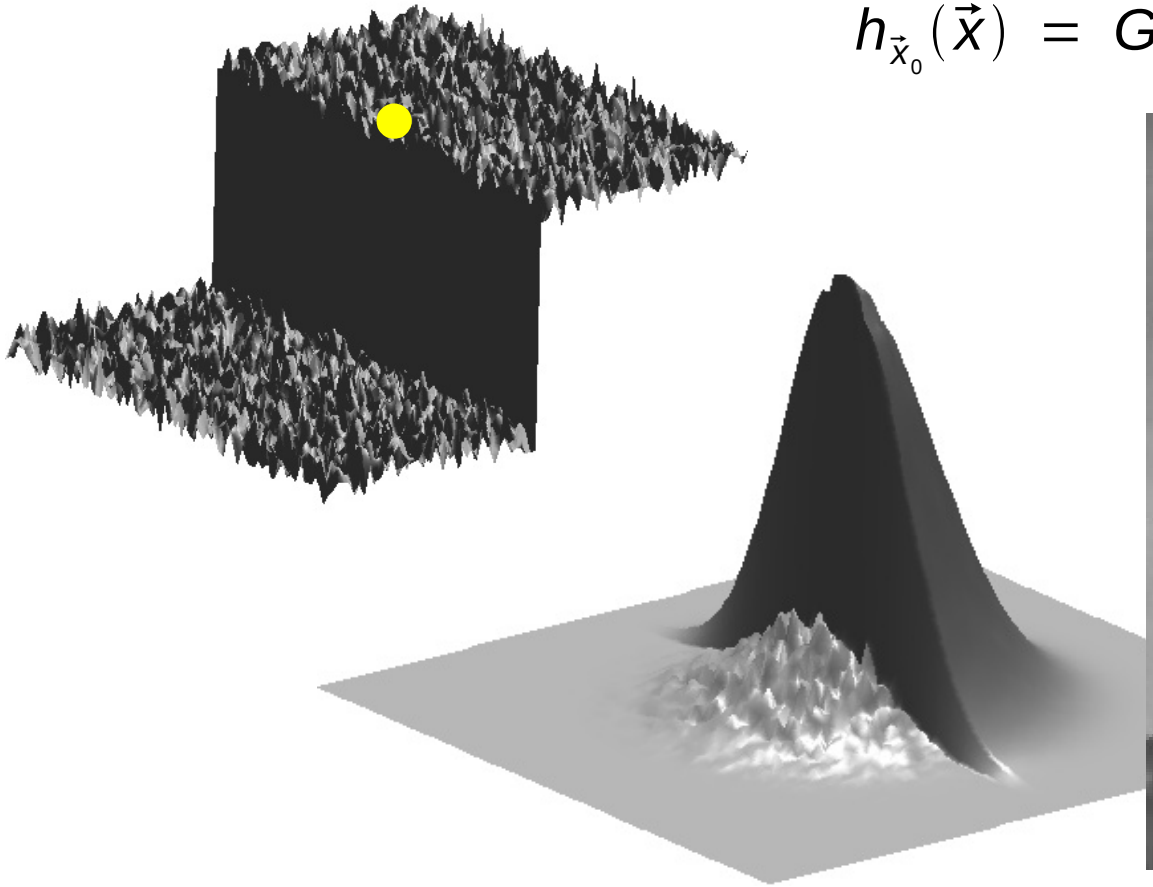




Bilateral filter

- Kernel is Gaussian in distance, like linear Gaussian
- Kernel is also Gaussian in intensity difference
  - Edges attenuate kernel significantly

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



Anisotropic diffusion

$$\frac{\partial f}{\partial t} = \nabla \cdot \left(D(f, \vec{x}) \nabla f \right) \quad , \quad f(\vec{x}, t)$$

- Linear diffusion = heat equation = Gaussian filter

$$\frac{\partial f}{\partial t} = D \nabla^2 f$$

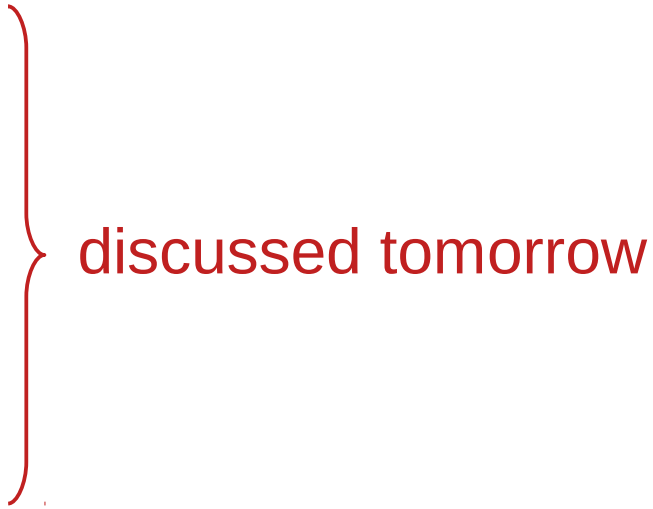
- Choose diffusion coefficient to be low at edges:

$$D(f, \vec{x}) = g(|\nabla f|)$$

$$\begin{cases} g(u) = \mathrm{e}^{-(u/K)^2} \\ g(u) = \dfrac{1}{1 + (u/K)^2} \end{cases}$$

Filtering for detection

- Edge detection
  - linear: gradient magnitude
  - non-linear
- Line detection:
  - linear: Laplace
  - non-linear
- Template matching
  - linear: correlation
  - non-linear:
    - mean square error
    - mean absolute error
    - etc.



First order derivatives

$$\frac{\partial}{\partial x} f(x) = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta}$$

$$\mathcal{F}\left\{\frac{\partial}{\partial x} f(x)\right\} = i\omega \mathcal{F}\left\{f(x)\right\}$$

- In a discrete grid, the smallest δ is 1
- Convolve with [1 -1] filter
  - Asymmetric
- Convolve with [1 0 -1] / 2 filter
  - Larger δ = worse approximation to derivative

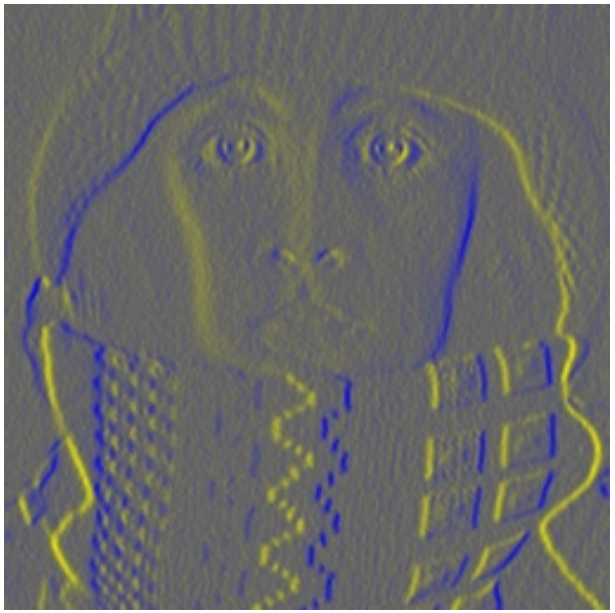
Gaussian derivatives

- Both filters have problems:
  - High response to noise
  - Poor approximation of gradient vector in *n*-D images
- Solution: use Gaussian derivatives

$$\frac{\partial}{\partial x}\left\{f(x)\otimes G(x)\right\} = f(x)\otimes\frac{\partial}{\partial x}G(x)$$

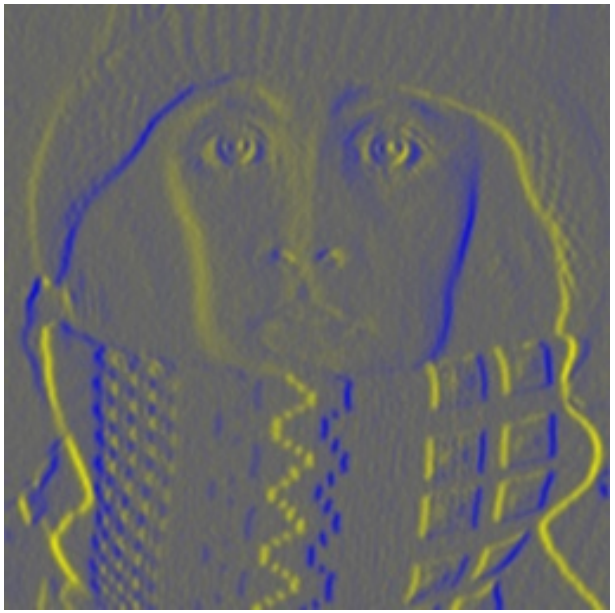
  - Reduced response to noise
  - Computes exact derivative of smoothed function
    (meaning gradient vector has correct direction)
  - “Band-limited”, so discretisable
  - Separable

Gradients



finite difference filter

[1 0 -1]/2



Gaussian derivative

Other derivatives

- In the first course you learned about some other derivative operators:
  - Prewitt

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} / 6 = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} / 2 \otimes \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} / 3$$

  - Sobel

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} / 8 = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} / 2 \otimes \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} / 4$$

  - Etc.
- You should understand now why Gaussian is better!

Uses of the gradient

- The gradient is a vector perpendicular to the edge
- Gradient magnitude is a measure for edge strength

$$|\nabla f| = \sqrt{\left(\frac{\partial}{\partial x} f\right)^2 + \left(\frac{\partial}{\partial y} f\right)^2 + \ldots}$$

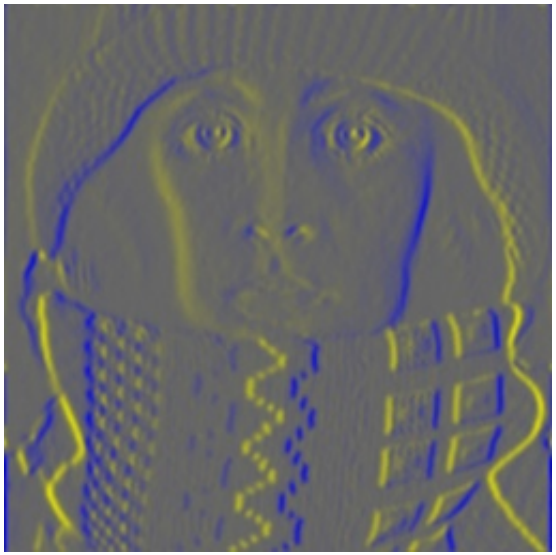
- Gradient direction is a measure for local orientation

$$\sphericalangle(\nabla f) = \operatorname{atan2}\left(\frac{\partial}{\partial y} f, \frac{\partial}{\partial x} f\right)$$

- Of course, you need a rotation invariant filter to accurately measure orientation. The Gaussian gradient operator is rotation invariant

Gradient magnitude

$$|\nabla f| = \sqrt{\left(\frac{\partial}{\partial x} f\right)^2 + \left(\frac{\partial}{\partial y} f\right)^2 + \ldots}$$



$\frac{\partial}{\partial x} f$

$\frac{\partial}{\partial y} f$

$|\nabla f|$

Detecting edges

Second order derivatives

$$\frac{\partial^2}{\partial x^2} f(x) = \lim_{\delta \to 0} \frac{f(x+\delta) - 2f(x) + f(x-\delta)}{\delta^2}$$

$$\mathcal{F}\left\{\frac{\partial^2}{\partial x^2} f(x)\right\} = -\omega^2 \mathcal{F}\{f(x)\}$$

- Finite difference approximation (δ=1)
  - Convolve with [1 -2 1]
  - Note that [1 -2 1] = [1 -1] ⊗ [1 -1]
- Gaussian approximation
  - Convolve with second derivative of Gaussian

Laplace operator

- The Laplace operator is everywhere in physics
  - e.g. remember heat equation: $\frac{\partial f}{\partial t} = D\nabla^2 f$

$$\nabla^2 f = \nabla \cdot \nabla f = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \ldots \right) f$$

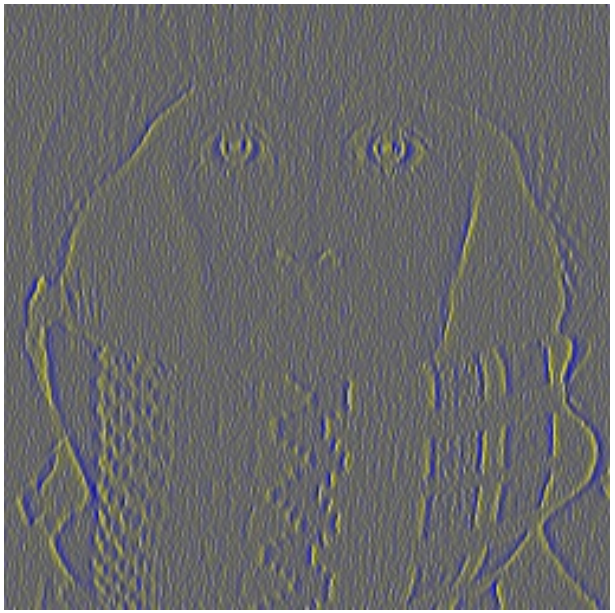
- The Laplace operator is:
  - isotropic
  - generalized 2nd derivative
- It detects lines, and responds strongly to edges
- It does not measure edge magnitude, as commonly claimed (and reported in book on pg. 133)

Laplace operator

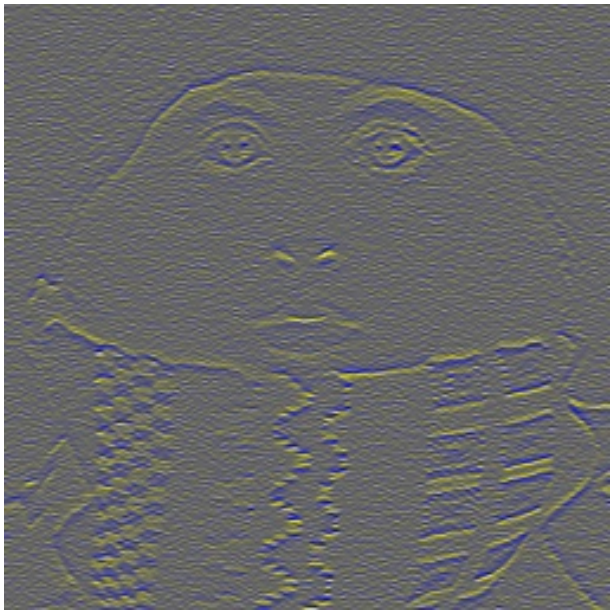
$$\nabla^2 f = \nabla \cdot \nabla f = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \ldots \right) f$$

- Finite difference approximation: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ or $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- Gaussian approximation:
  - Less sensitive to noise, more isotropic
  - But: not separable!
- Another approximation: difference of Gaussians (DoG)
  - Advantage: separable

Laplace operator



$$\frac{\partial^2}{\partial x^2} f$$



$$\frac{\partial^2}{\partial y^2} f$$

$$\nabla^2 f$$

finite difference
2nd derivative:
[1 -2 1]

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplace operator

$$\frac{\partial^2}{\partial x^2} f$$

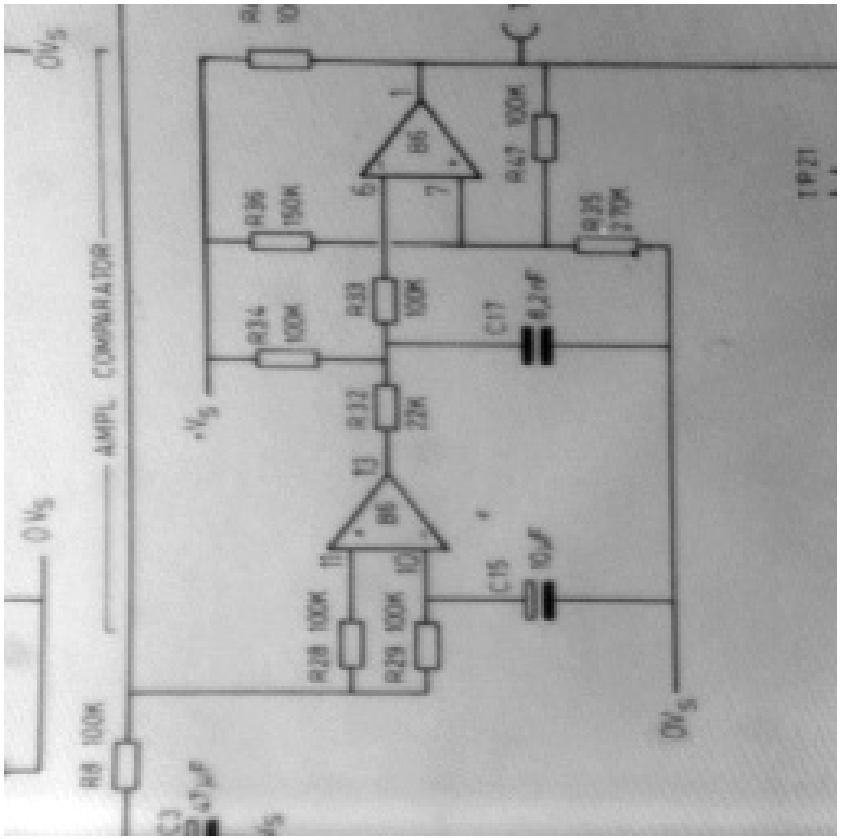
$$\frac{\partial^2}{\partial y^2} f$$

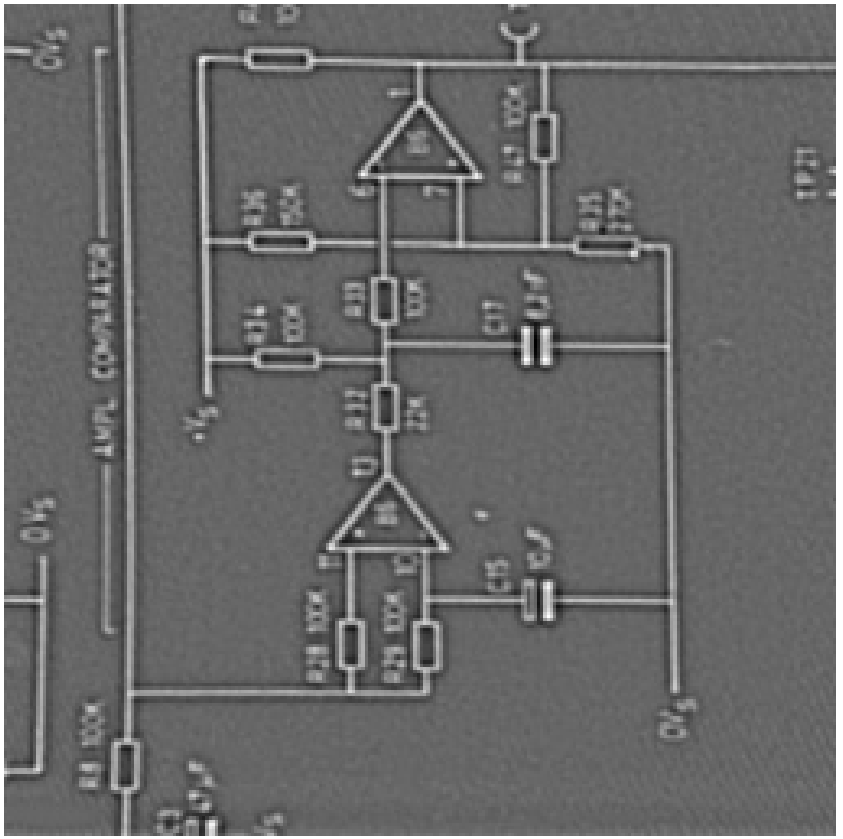
$$\nabla^2 f$$

Gaussian
2nd derivative

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Detecting lines



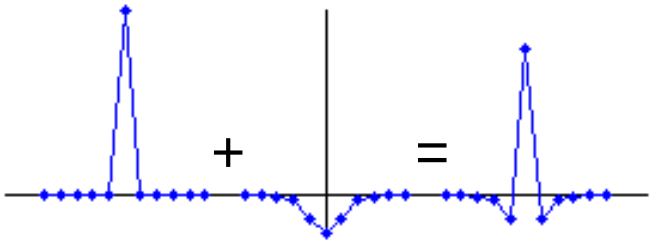


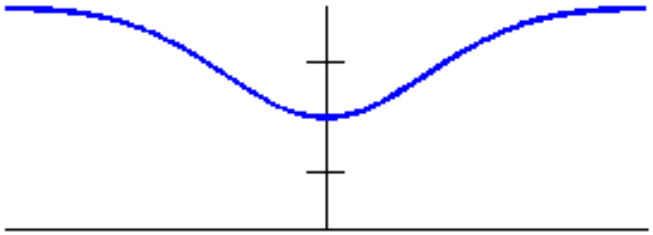
Unsharp masking revisited

spatial domain

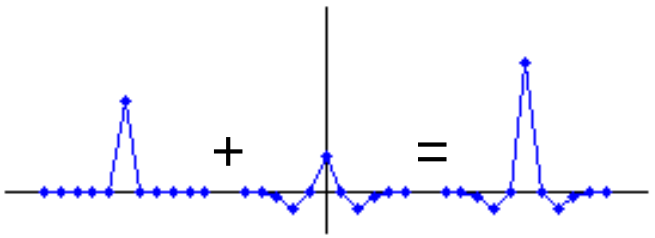
frequency domain

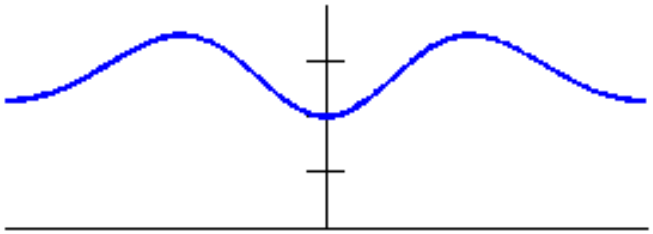
$2\delta - G_{\sigma=1}$



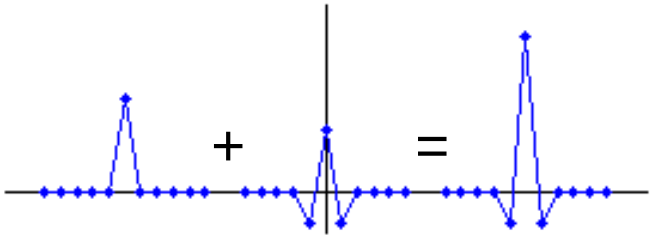


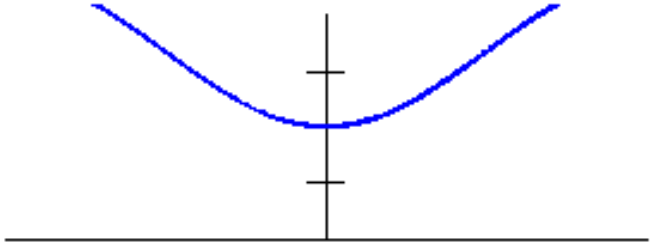
$\delta - \nabla^2 G_{\sigma=1}$



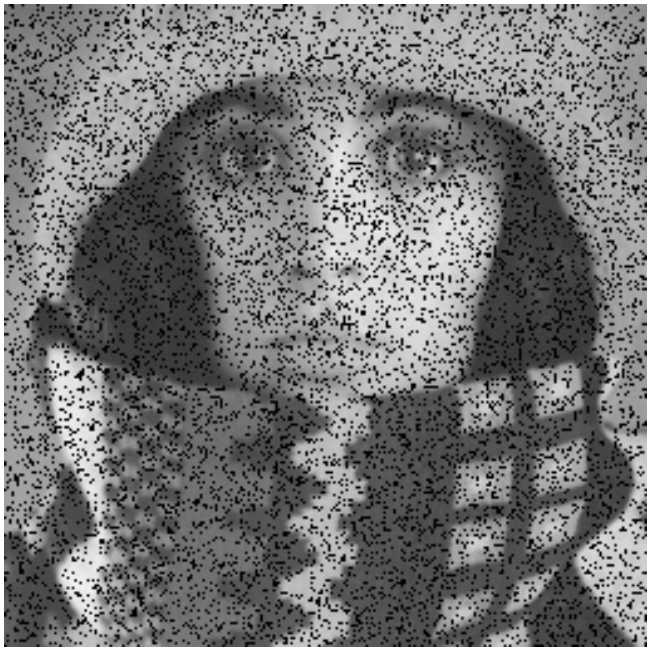


$\delta - \nabla^2 \delta$

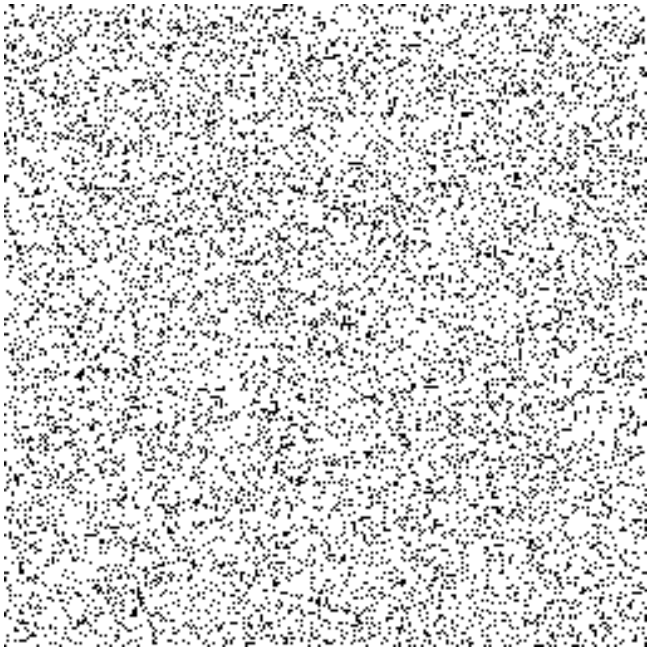




Normalised convolution



Input data, with missing samples

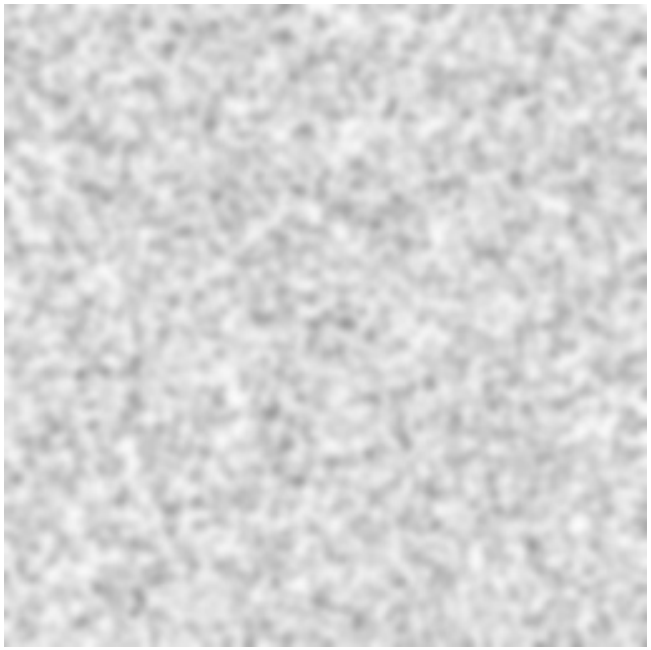


We also know which samples are missing

Simply filtering the input image with the missing samples produces a bad output

Normalised convolution

$\{f(x)m(x)\}\otimes h(x)$

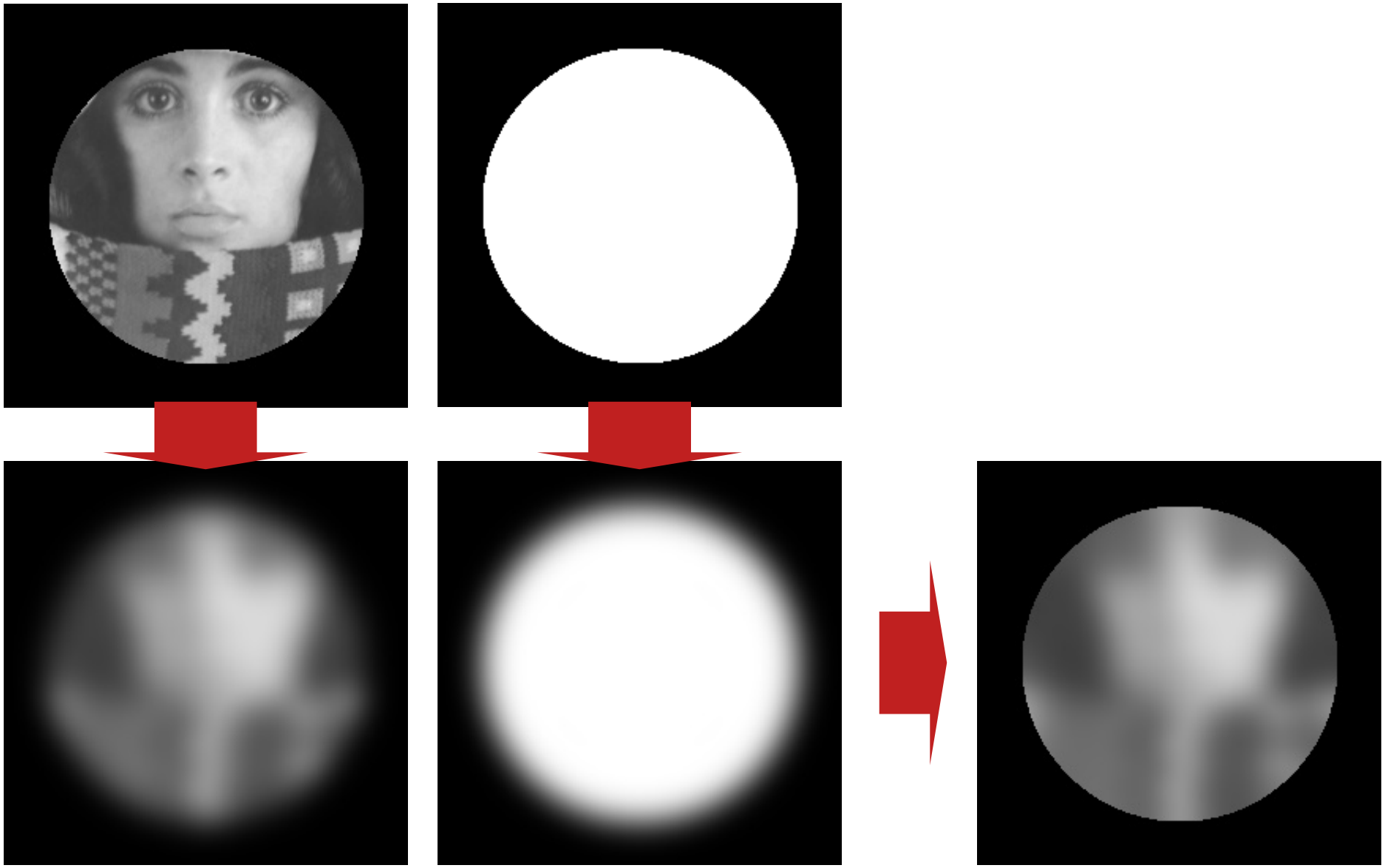


$m(x)\otimes h(x)$

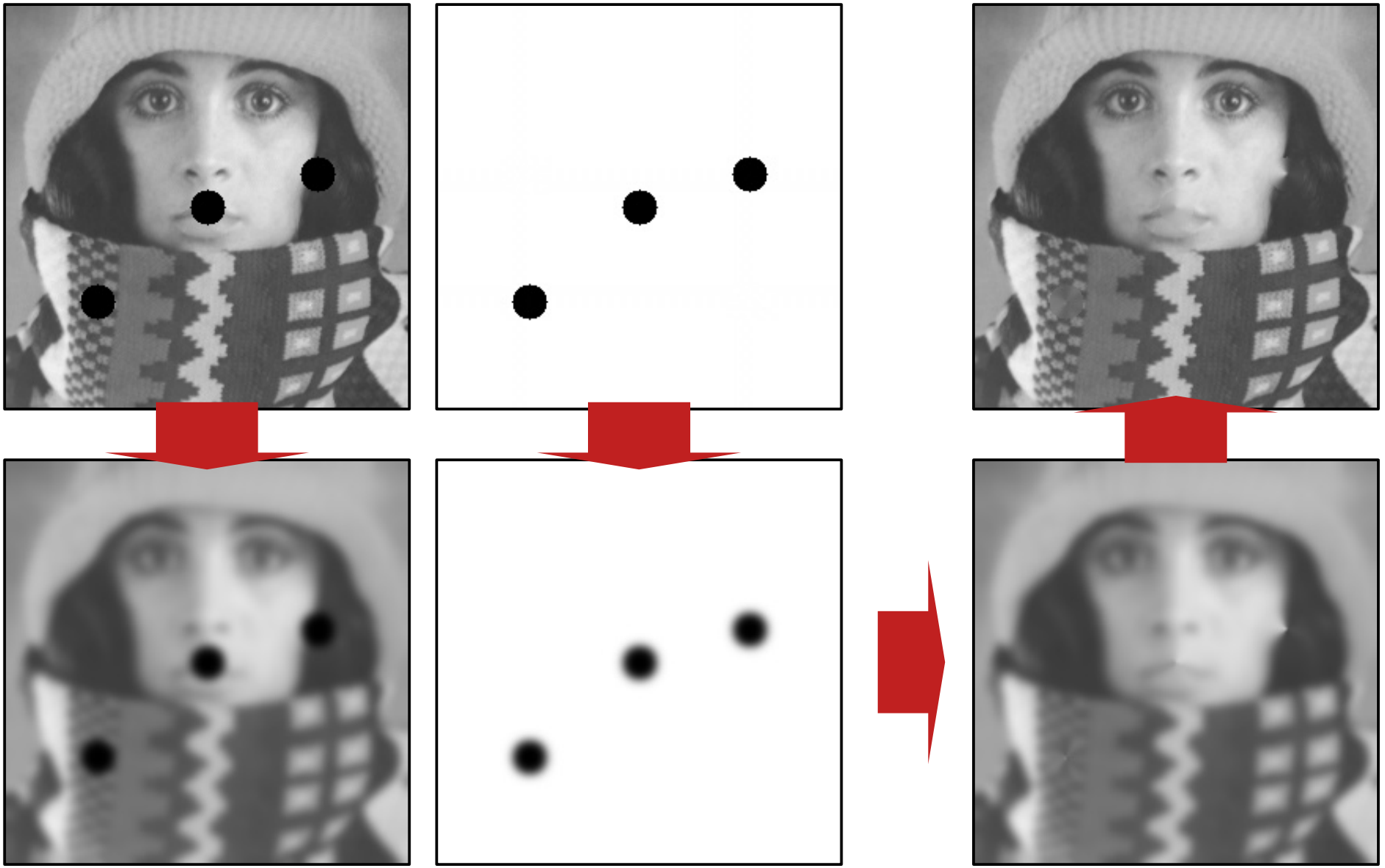
$$\frac{\{f(x)m(x)\}\otimes h(x)}{m(x)\otimes h(x)}$$

By normalising with the mask, the filter "skips" missing input samples

Normalised convolution



Normalised convolution



Summary of today's lecture

- Gaussian filters for:
  - Smoothing
  - Derivatives
  - Laplace operator
- Non-linear filters for edge-preserving smoothing
- Smoothing filters for:
  - Noise reduction
  - Image abstraction (simplification)
  - Shading correction
  - Edge sharpening
- First order derivatives used for edge detection
- Second order derivatives used for line detection