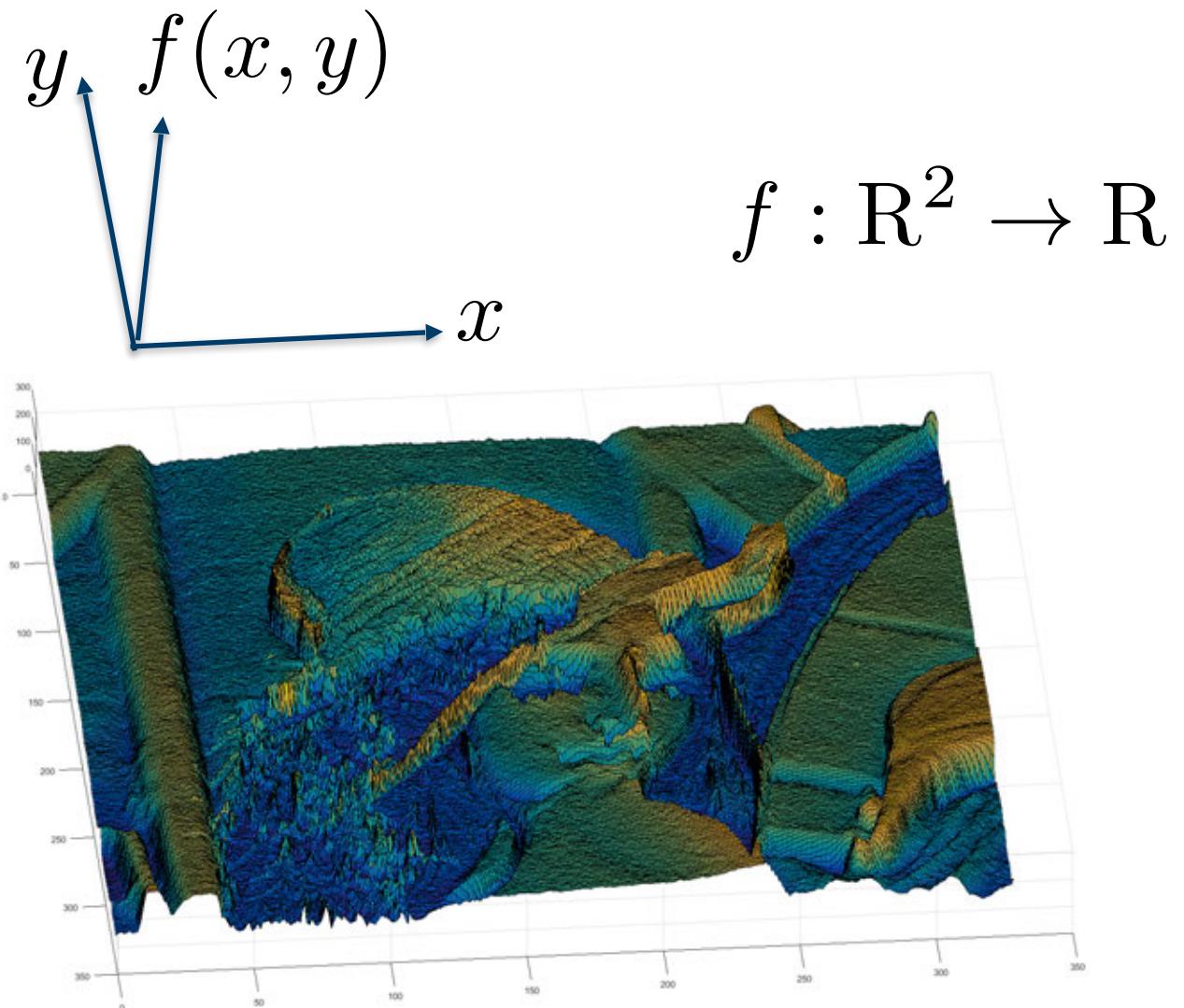


Image Processing

Lecture 2.1 - Image filtering

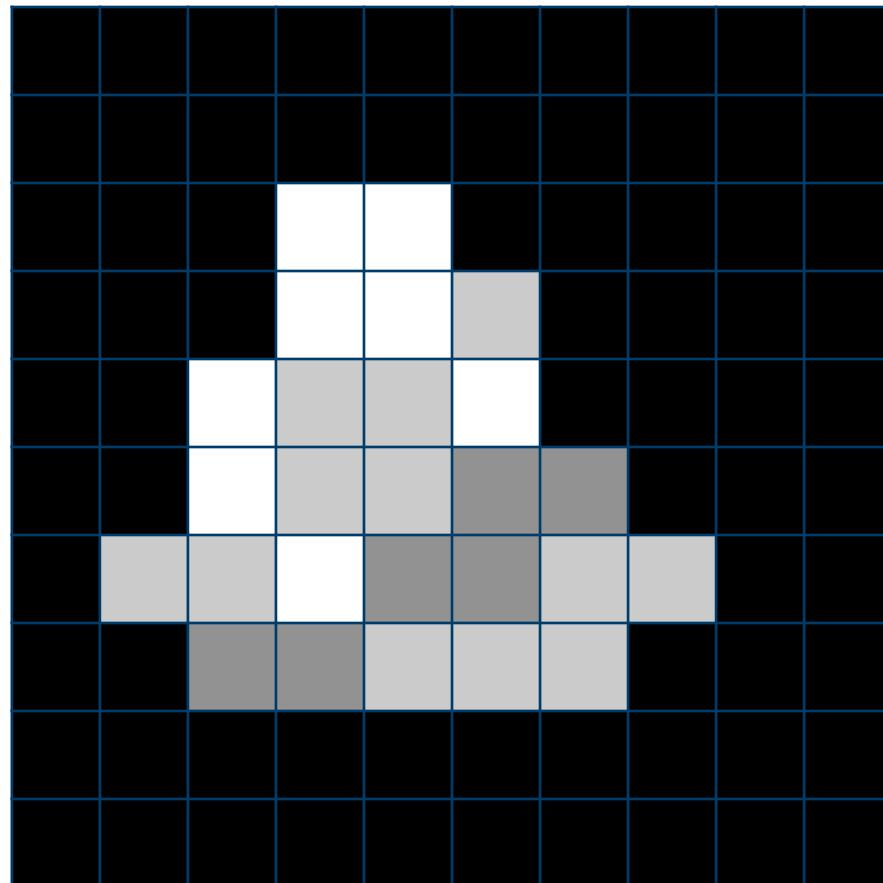
Idar Dyrdal

Image function



2D signal where $f(x, y)$ gives the **intensity** at position (x, y)

Digital Image



=

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0	0	0
0	0	0	255	255	170	0	0	0	0
0	0	255	170	170	255	0	0	0	0
0	0	255	170	170	85	85	0	0	0
0	170	170	255	85	85	170	170	0	0
0	0	85	85	170	170	170	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Discrete (**sampled** and **quantized**) version of the (continuous) image function $f(x, y)$

Image Processing

- Point operators
- Image filtering in spatial domain
 - Linear filters
 - Non-linear filters
- Image filtering in frequency domain
 - Fourier transform



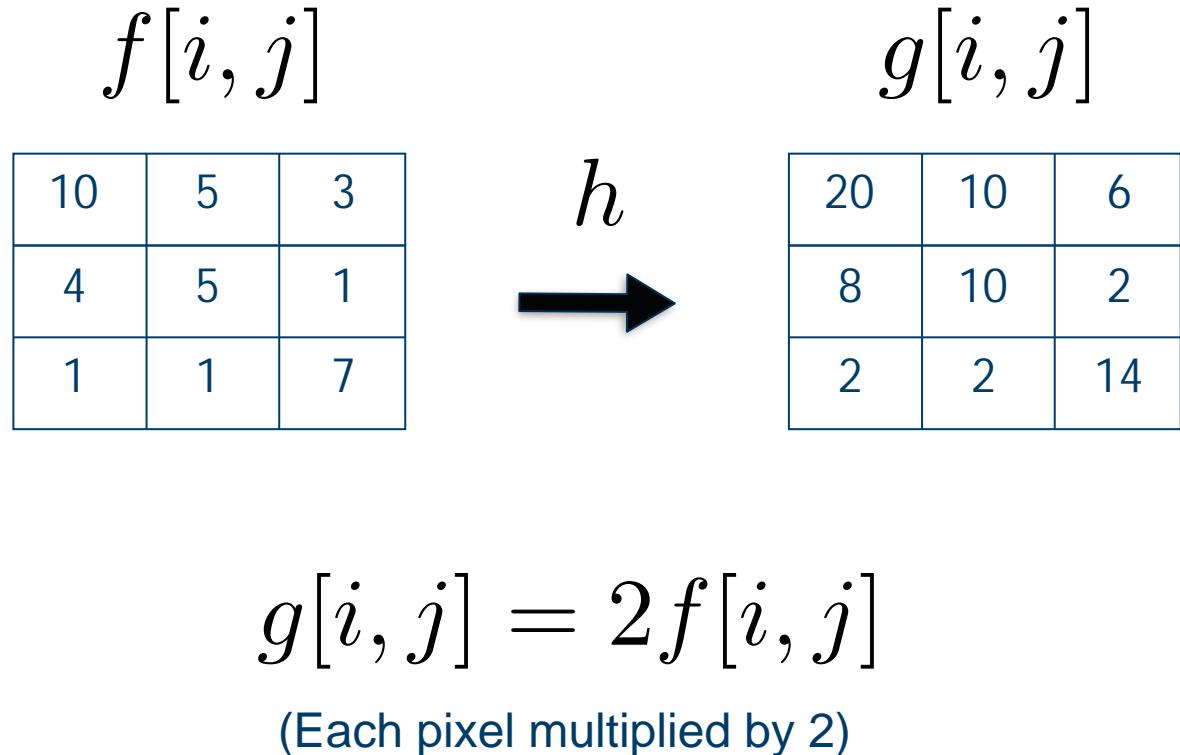
$$f[i, j] \rightarrow g[i, j]$$

Point Operators

- Pixel transforms
 - Brightness adjustment
 - Contrast adjustment
 - ...
- Colour transforms
- Histogram equalization
- ...

$$g[i, j] = h(f[i, j])$$

(Pixel-by-pixel transformation)



Pixel transforms - example

$$f[i, j]$$



Original image

$$g[i, j] = \sqrt{f[i, j]}$$



Processed image

Histogram equalization

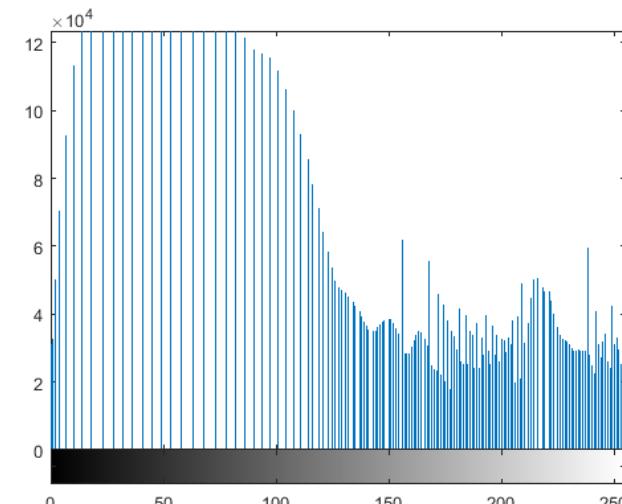
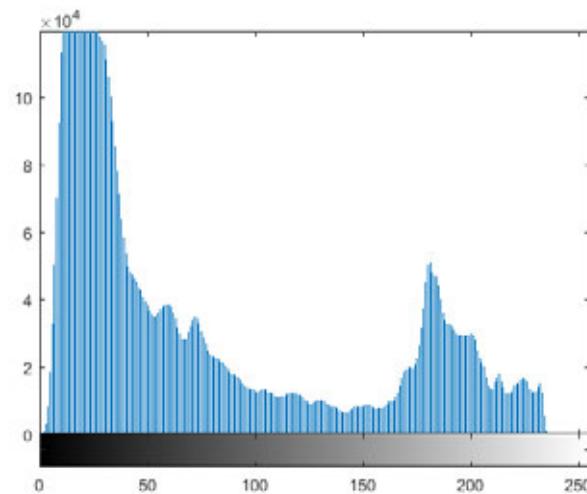


Image filtering

- **Image filters in the spatial domain:**
 - Filtering is a mathematical operation on a grid of numbers
 - Smoothing, sharpening (enhancing the image)
 - Feature extraction (measuring texture, finding edges, distinctive points and patterns).
- **Image filters in the frequency domain:**
 - Filtering is a way to modify the (spatial) frequencies of images
 - Noise removal, (re)sampling, image compression.

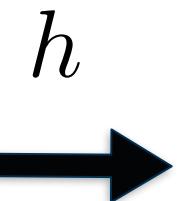
Image filtering in spatial domain

Modify the pixels in an image based on some function of a local neighborhood of each pixel:

$$f[i, j]$$

10	5	3
4	5	1
1	1	7

Local image data



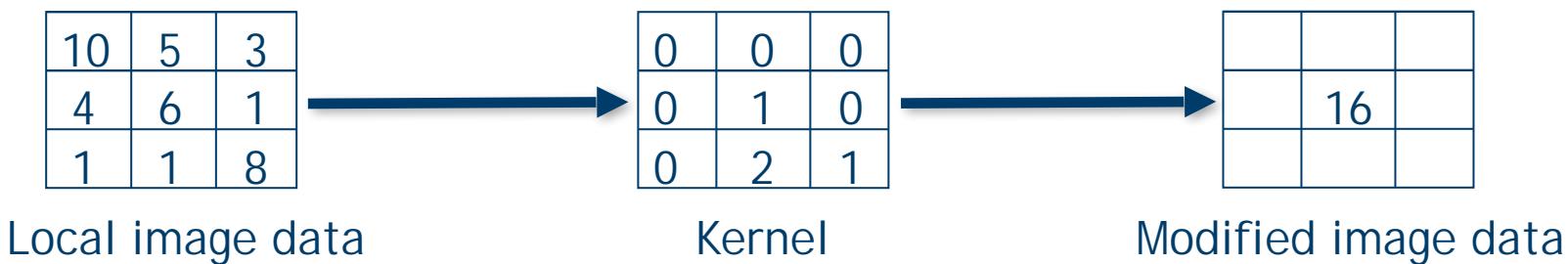
$$g[i, j]$$

	10	

Modified image data

Linear filtering

Convolution or **cross-correlation** where each pixel in the filtered image is a linear combination of the pixels in a local neighborhood in the original image:



The coefficients of the linear combination is contained in the “kernel” (filter mask).

Cross-correlation

Let f be the image, h be the kernel (of size $2k+1 \times 2k+1$), and g be the output image:

$$g[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k h[u, v] f[i + u, j + v]$$

This is called a cross-correlation operation:

$$g = h \otimes f$$

Linear filtering

$$f \otimes h = g$$

Diagram illustrating linear filtering:

- Input image f (10x10 grid):

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	0	0	0	0	0
0	0	0	90	90	90	0	0	0	0
0	0	90	90	90	90	0	0	0	0
0	0	90	90	90	90	0	0	0	0
0	90	90	90	90	90	90	90	0	0
0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
- Filter h (3x3 grid):

1	1	1
1	1	1
1	1	1
- Output image g (10x10 grid):

0	0	0	0	0	0	0	0	0	0
0	0	10	20	20	10	0	0	0	0
0	0	20							
- Filtering operation:
$$g[i, j] = \sum_{u,v} h[u, v] f[i + u, j + v]$$

Linear filtering

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	0	0	0	0	0	0
0	0	0	90	90	90	0	0	0	0	0
0	0	90	90	90	90	0	0	0	0	0
0	0	90	90	90	90	90	0	0	0	0
0	90	90	90	90	90	90	90	0	0	0
0	0	90	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0



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

=

0	0	0	0	0	0	0	0	0	0
0	0	10	20	20	10	0	0	0	0
0	0	20	40	50	30	10	0	0	0
0	10	40	70	80	50	20	0	0	0
0	20	50	80	90	70	40	10		

Linear filtering

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	0	0	0	0	0	0
0	0	0	90	90	90	0	0	0	0	0
0	0	90	90	90	90	0	0	0	0	0
0	0	90	90	90	90	90	0	0	0	0
0	90	90	90	90	90	90	90	0	0	0
0	0	90	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$$\otimes \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

0	0	0	0	0	0	0	0	0	0	0
0	0	10	20	20	10	0	0	0	0	0
0	0	20	40	50	30	10	0	0	0	0
0	10	40	70	80	50	20	0	0	0	0
0	20	50	80	90	70	40	10	0	0	0
0	40	70	90	90	80	60	30	10	0	0
10	40	70	90	90	90	70	40	10	0	0
0	30	50	60	60	60	50	30	10	0	0
0	10	20	30	30	30	20	10	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Moving average filter (box filter)



3 x 3 kernel

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



Replaces each pixel with an average of its neighborhood (smoothing effect)

9 x 9 kernel

$$g[i, j] = \sum_{u,v} h[u, v] f[i + u, j + v]$$



Sharpening filter



Enhances differences with
local average

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

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

Convolution

- Same as cross-correlation, except that the kernel is “flipped” (horizontally and vertically):

$$g[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k h[u, v] f[i - u, j - v]$$

- This is called a **convolution** operation:

$$g = h * f$$

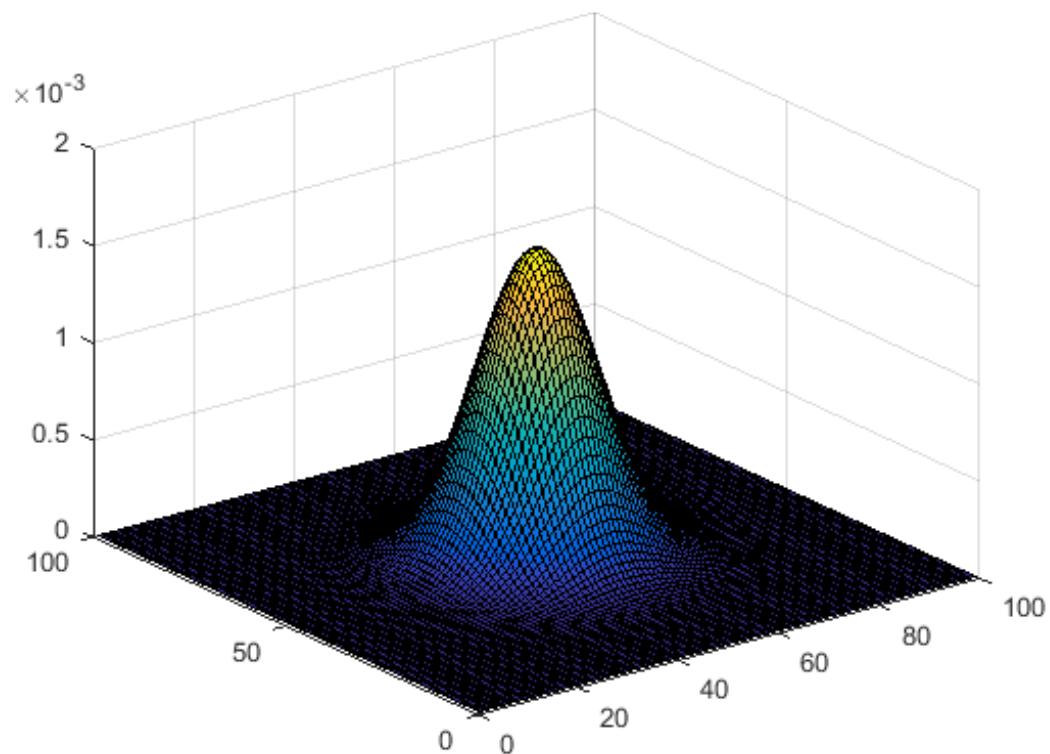
- Convolution is **commutative** and **associative** (*no difference between filter and image*):

$$a * b = b * a \quad a * (b * c) = (a * b) * c$$

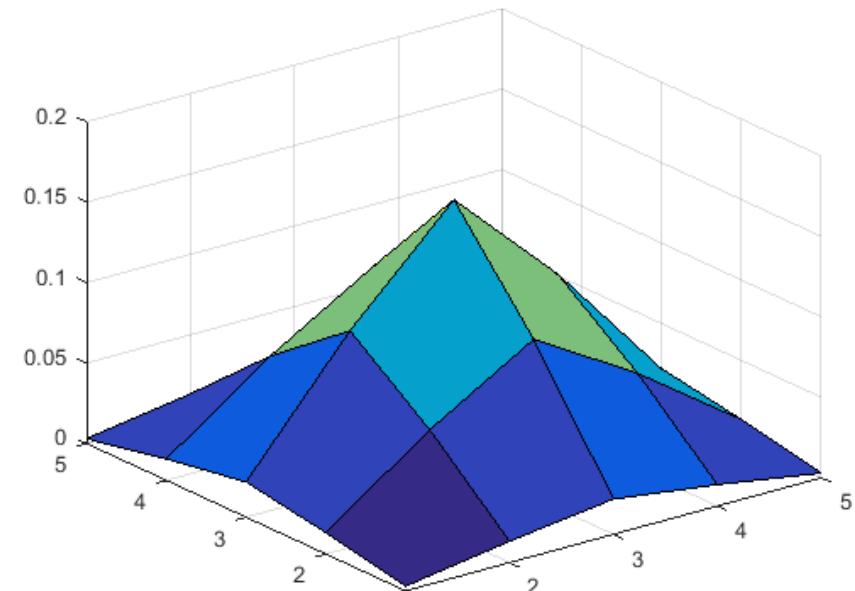
- Apply several filters, one after the other:

$$(((a * b_1) * b_2) * b_3) = a * (b_1 * b_2 * b_3)$$

Gaussian filter (smoothing)



$100 \times 100, \sigma = 10$



$5 \times 5, \sigma = 1$

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

Gaussian filtering



Original image



$\sigma = 2$ pixels



$\sigma = 4$ pixels



$\sigma = 8$ pixels

Separable filters - example

The 2D Gaussian kernel can be expressed as a product of two 1D kernels:

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

Discrete 3 x 3 approximation:

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

More efficient to perform two 1D convolutions compared to a single 2D convolution!

2D convolution

1	2	1
2	4	2
1	2	1

Filter kernel

*

1	2	2
1	4	4
3	3	5

3 x 3 image window

=

	47	

Result (center pixel only)

1D convolution along rows and columns

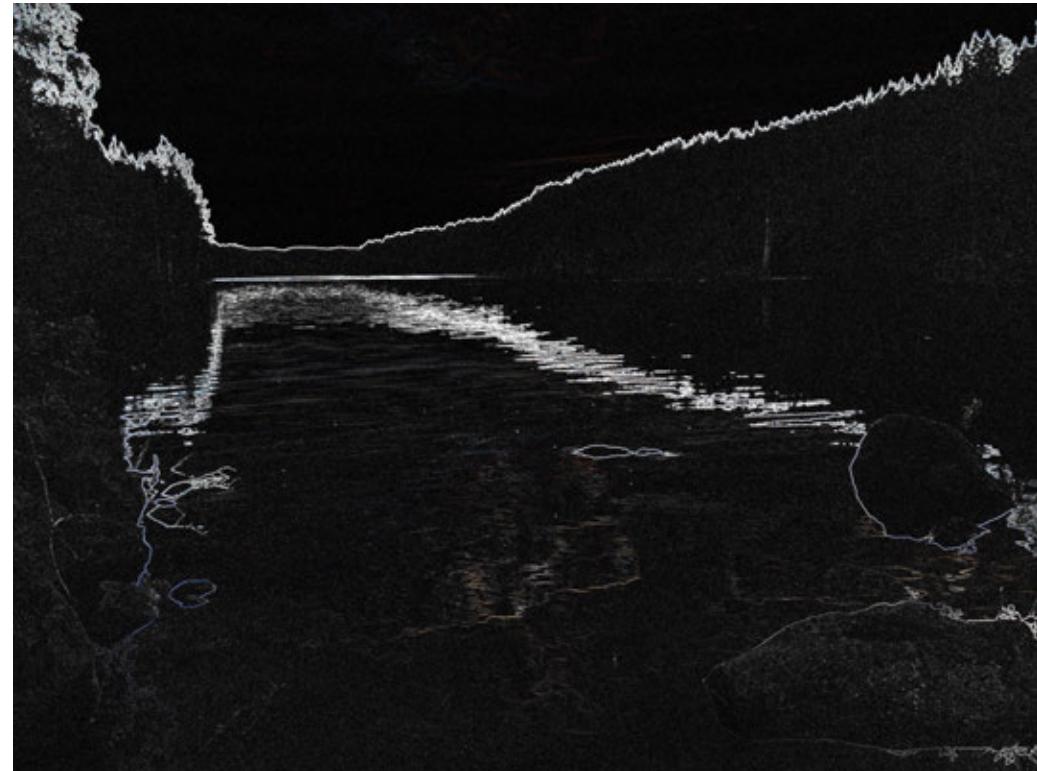
Convolution along rows:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 2 & 2 \\ \hline 1 & 4 & 4 \\ \hline 3 & 3 & 5 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & 7 & \\ \hline & 13 & \\ \hline & 14 & \\ \hline \end{array}$$

Convolution along remaining column:

$$\begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline & 7 & \\ \hline & 13 & \\ \hline & 14 & \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & & \\ \hline & 47 & \\ \hline & & \\ \hline \end{array}$$

Edge detection



Edges and image derivatives

- An edge is a place of rapid change of the image intensity function
- Corresponds to extrema of the first derivative of the image intensity function
- Discrete approximation to the image derivatives:

$$\frac{\partial f}{\partial x}[i, j] \approx f[i + 1, j] - f[i, j]$$

$$\frac{\partial f}{\partial y}[i, j] \approx f[i, j + 1] - f[i, j]$$

Image gradient:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

Gradient magnitude:

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

Prewitt operator:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Image gradient



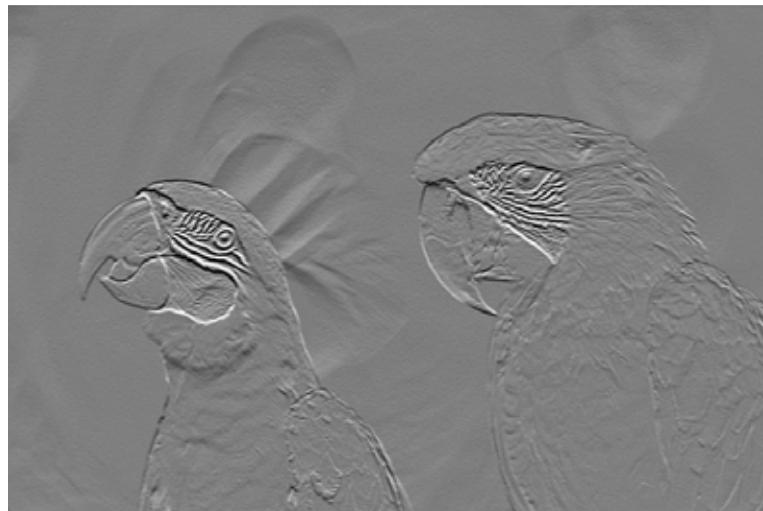
f



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

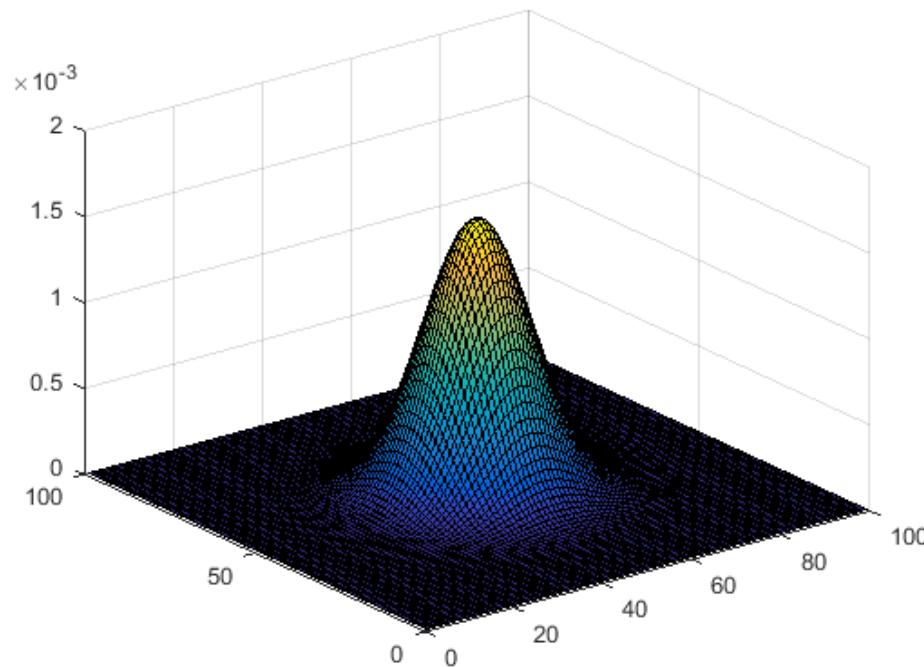


$$||\nabla f||$$

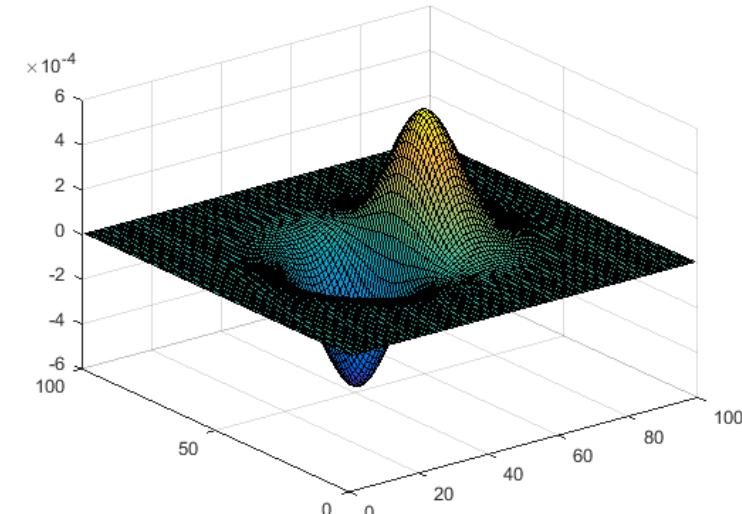


$$\frac{\partial f}{\partial y}$$

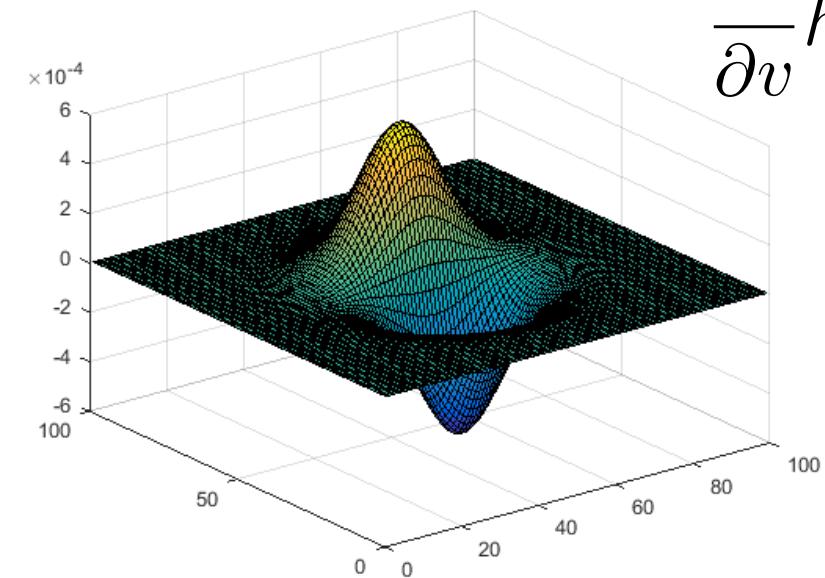
Derivative of Gaussians



$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-(\frac{u^2+v^2}{2\sigma^2})}$$



$$\frac{\partial}{\partial v} h_\sigma(u, v)$$



Sobel operator

Common approximation of the derivative of a Gaussian:

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

x-direction

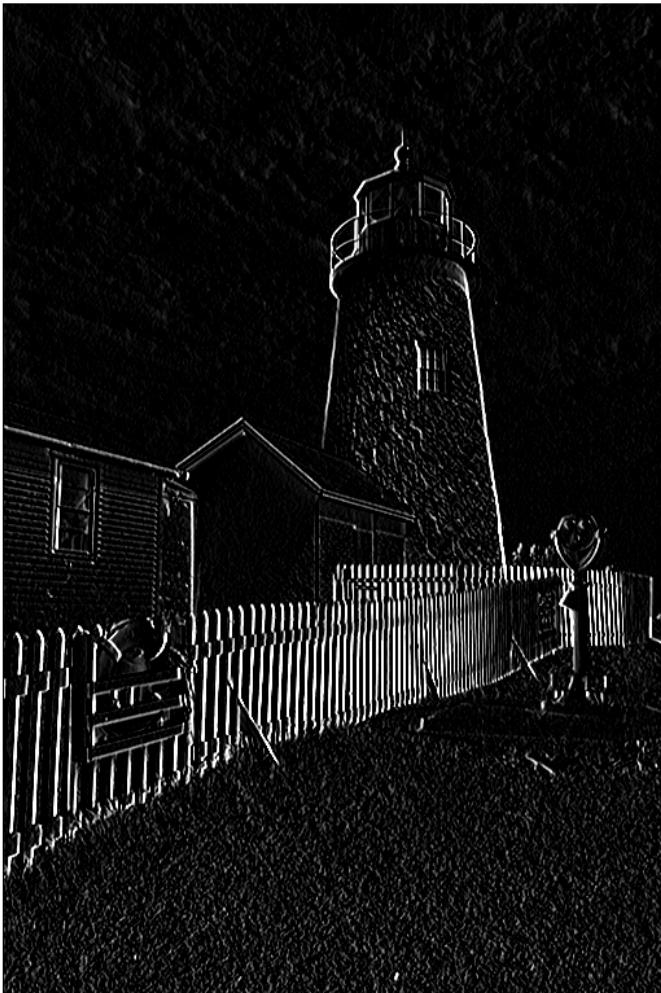
-1	-2	-1
0	0	0
1	2	1

y-direction

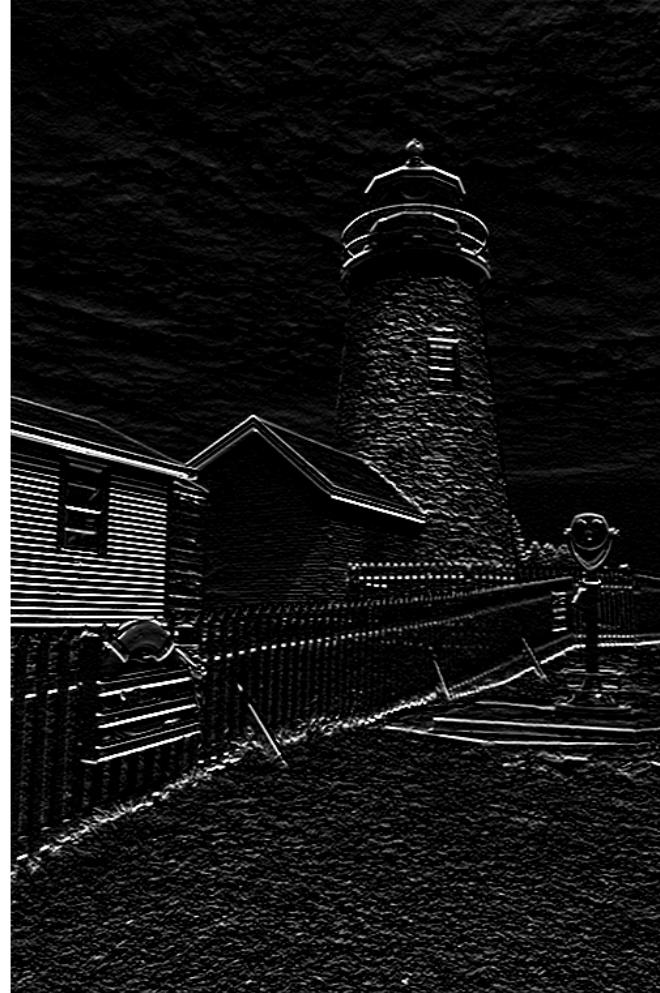


Sobel operator - example

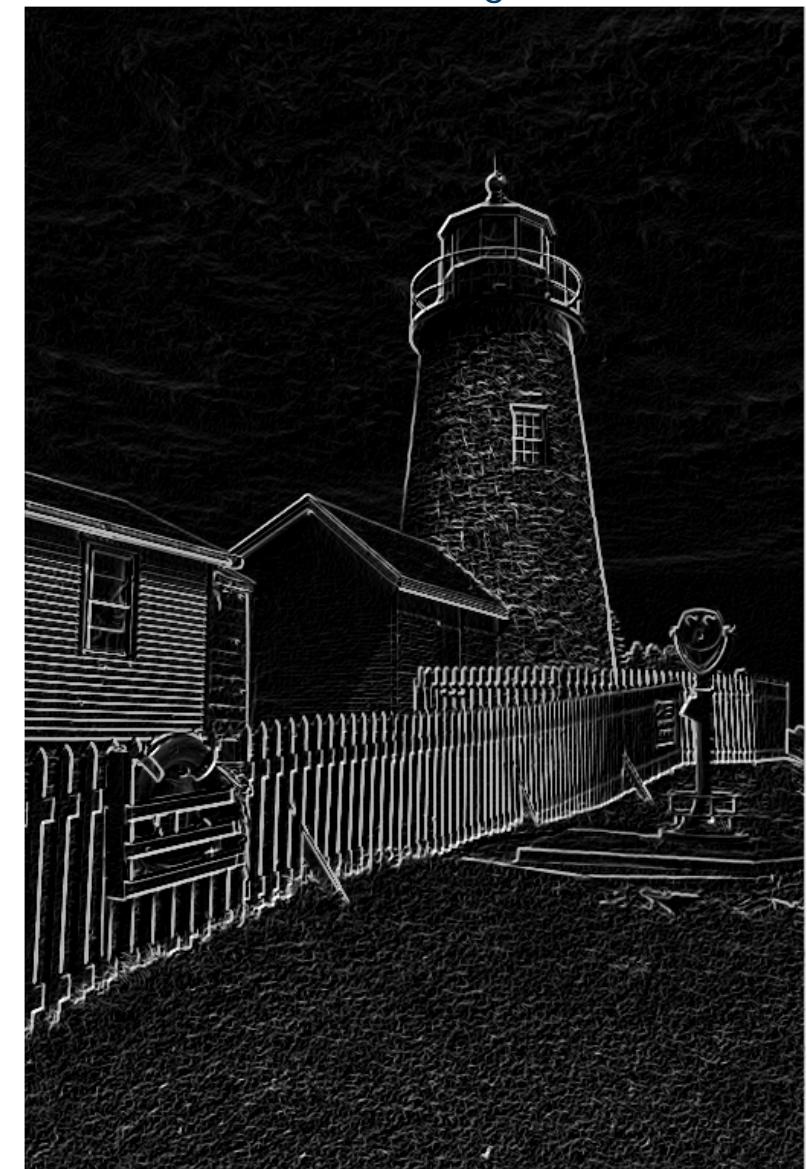
x-direction



y-direction



Gradient magnitude



Non-linear filtering - Median filter

A **median filter** operates over a neighborhood in the input image by selecting the median intensity:



Other non-linear filters:

- Bilateral filters (outlier rejection)
- Anisotropic diffusion
- Morphological operations (on binary images)
- ...

Median filtering - example

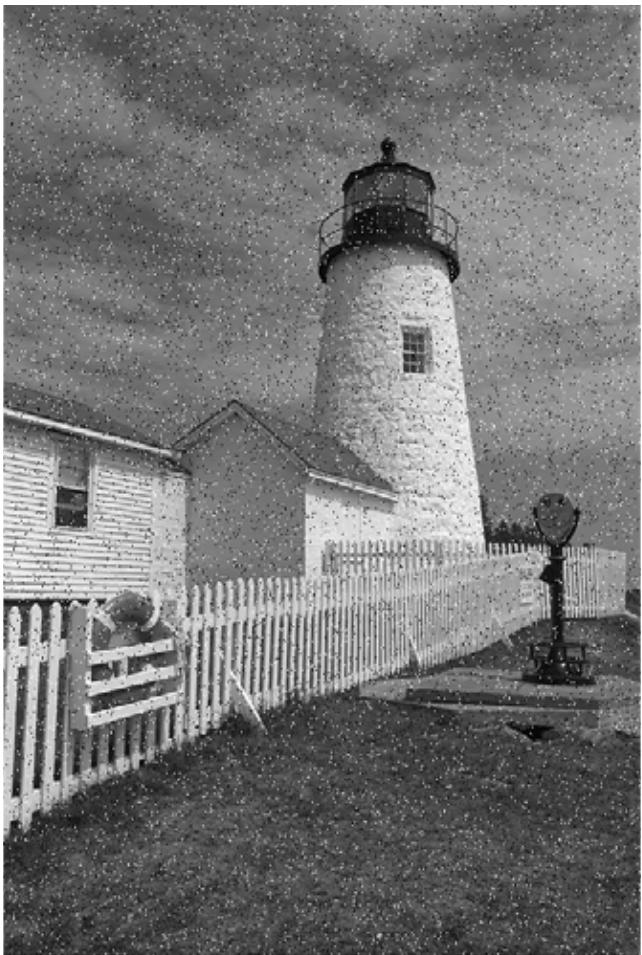


Image with Salt & Pepper noise



Image after median filtering

Morphological operations

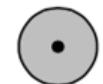
- Non-linear filtering
- Typically used to clean up binary images
- Erosion: replace pixel value with minimum in local neighborhood
- Dilation: replace pixel value with maximum in local neighborhood
- Structuring element used to define the local neighborhood:

0	1	0
1	1	1
0	1	0

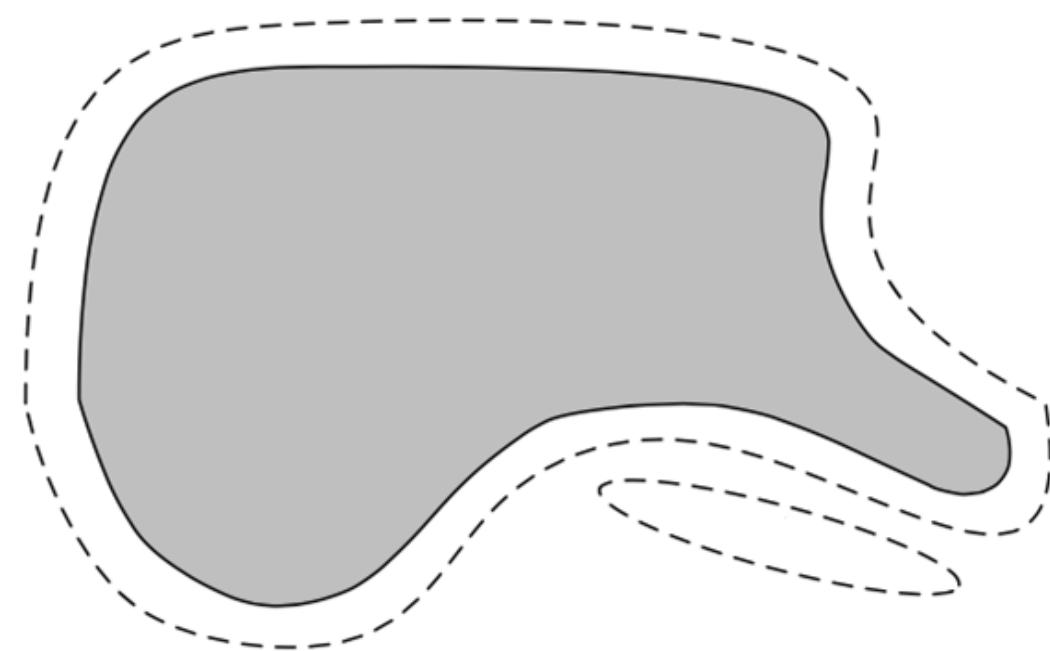
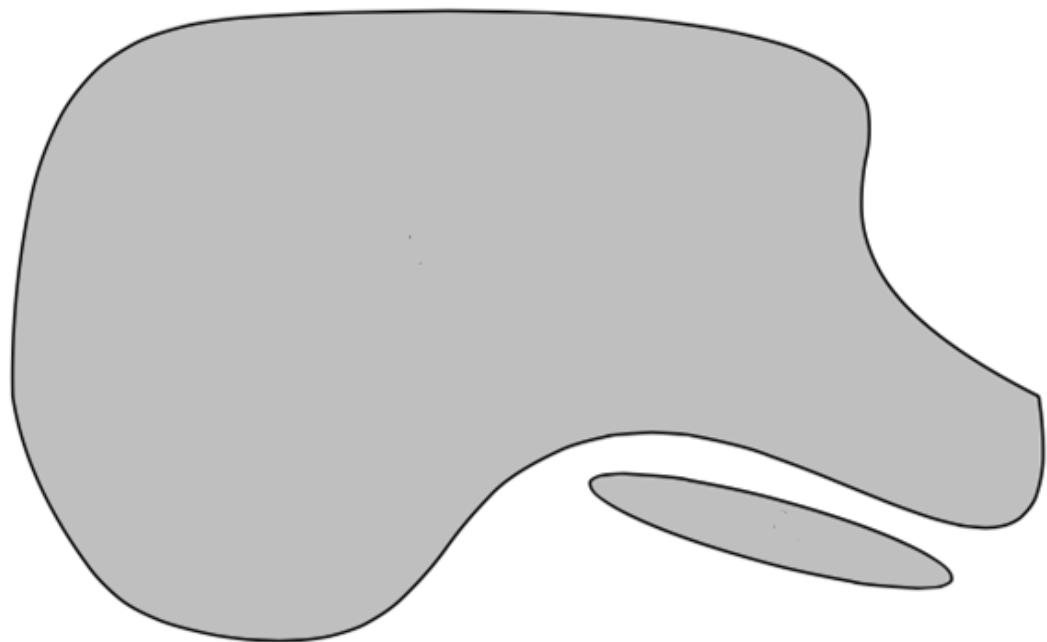


A shape (in blue) and its morphological dilation (in green) and erosion (in yellow) by a diamond-shape structuring element.

Morphological operations - Erosion



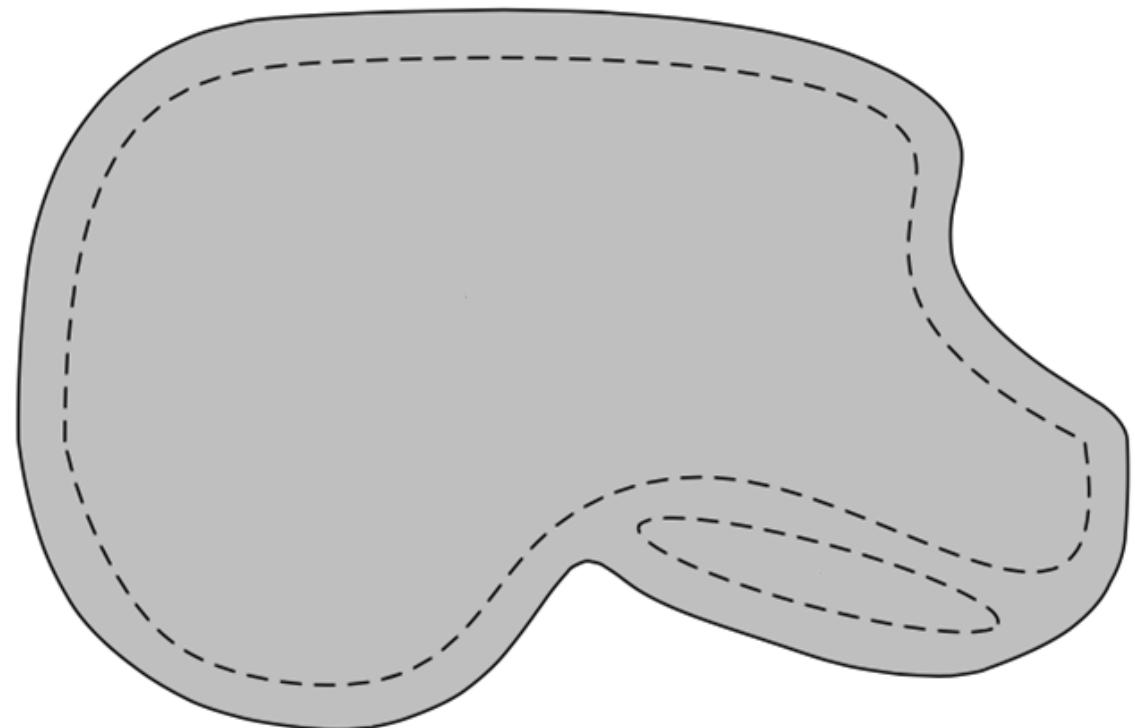
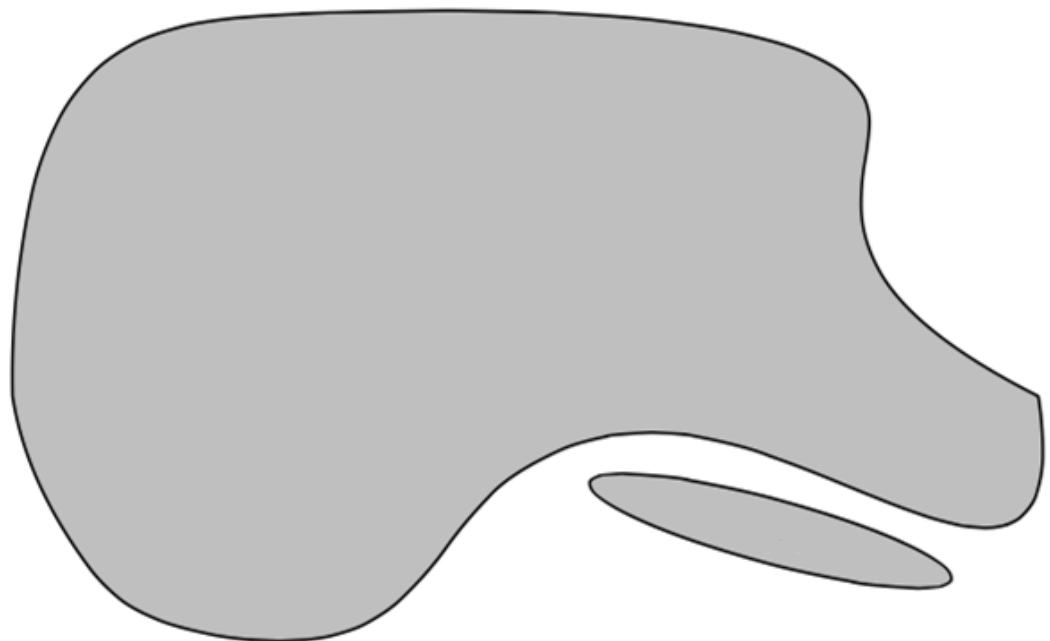
Structuring element (disk shaped)



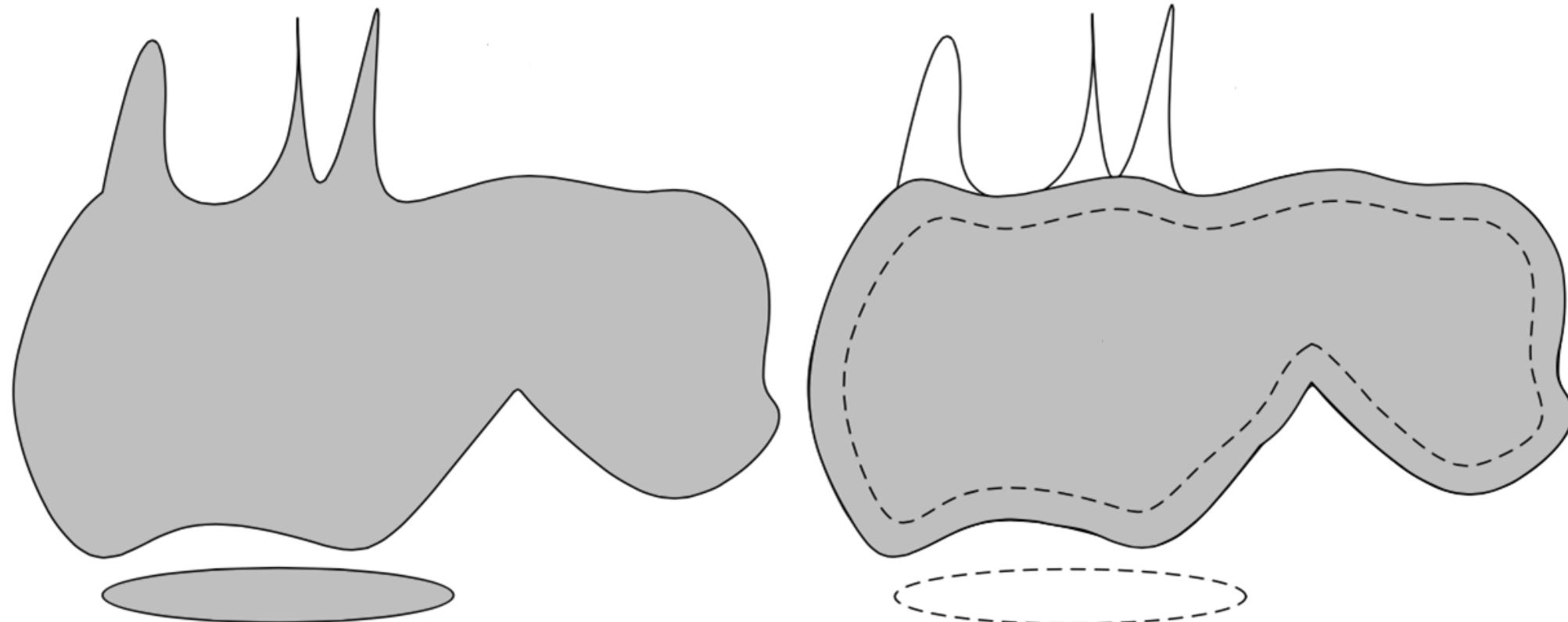
Morphological operations - Dilation



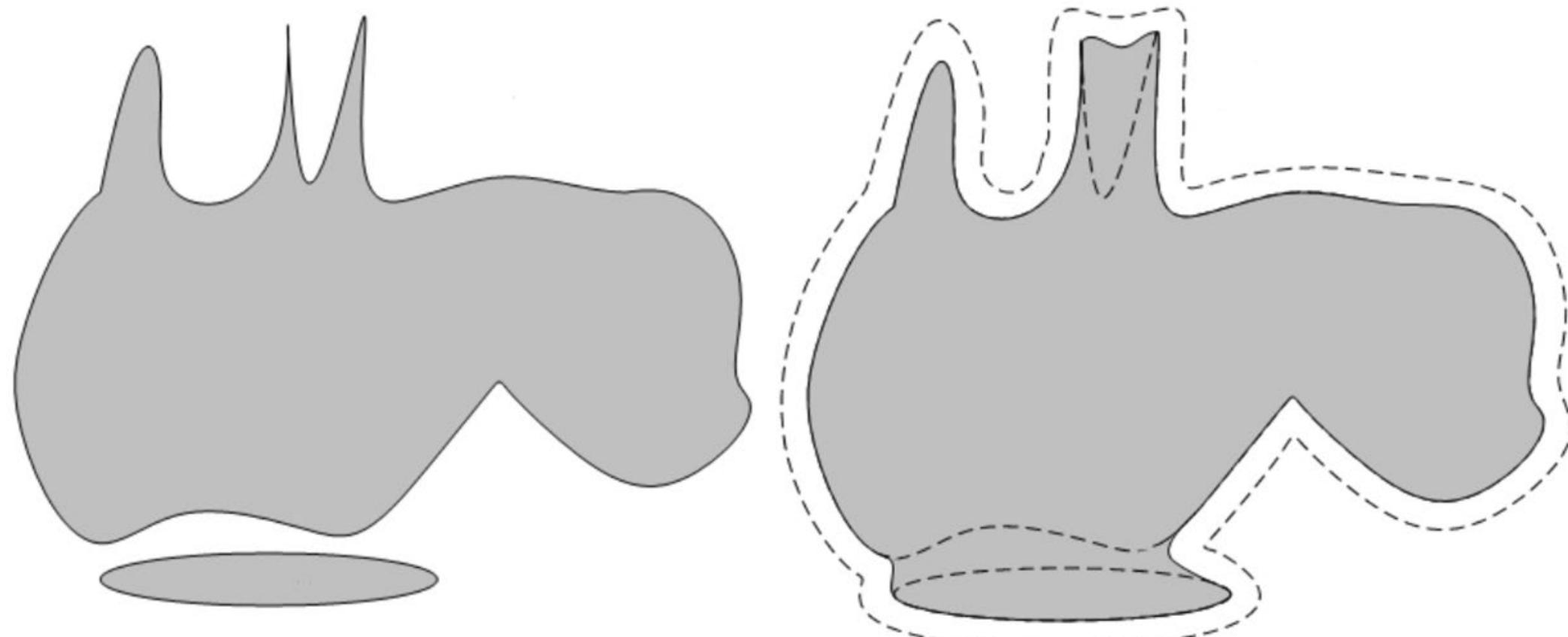
Structuring element (disk shaped)



Opening = Erosion + Dilation



Closing = Dilation + Erosion



Filtering in frequency domain

Fourier (1807):

Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies (true with some subtle restrictions).

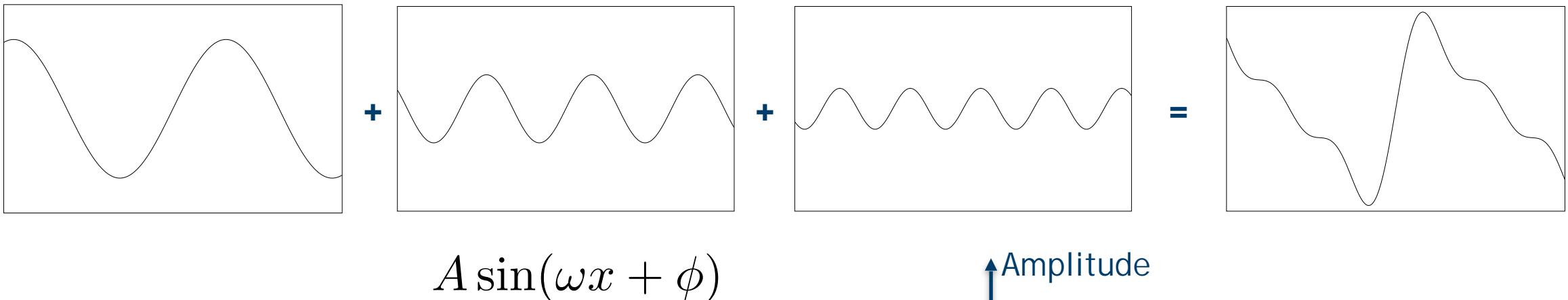
This leads to:

- Fourier Series
- Fourier Transform (continuous and discrete)
- Fast Fourier Transform (FFT)



Jean Baptiste Joseph Fourier (1768-1830)

Sum of sines



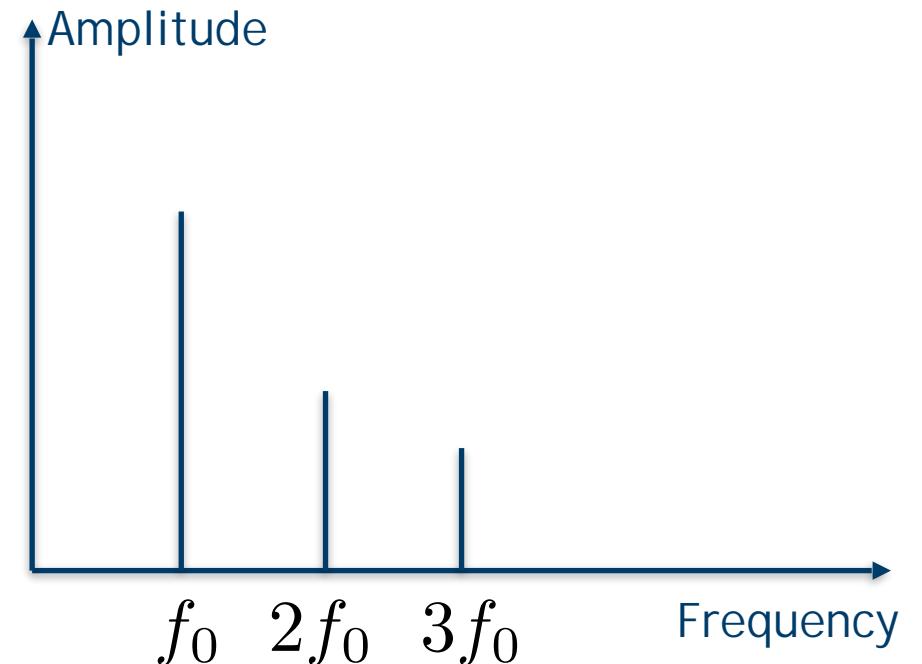
The Fourier transform stores the magnitude and phase at each frequency

Amplitude:

$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2}$$

Phase:

$$\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$



Two-dimensional Fourier transform

Continuous transform:

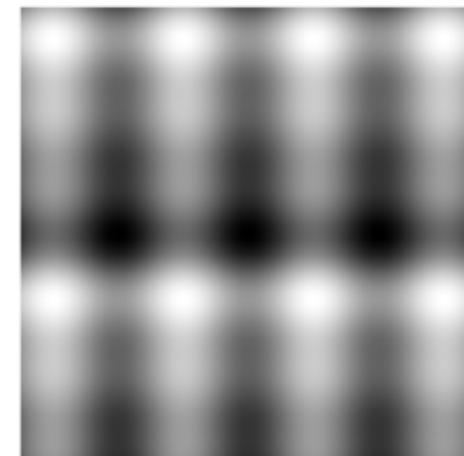
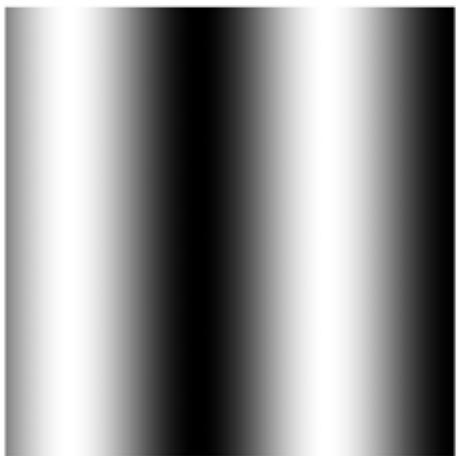
$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) e^{-j(\omega_x x + \omega_y y)} dx dy$$

Discrete transform:

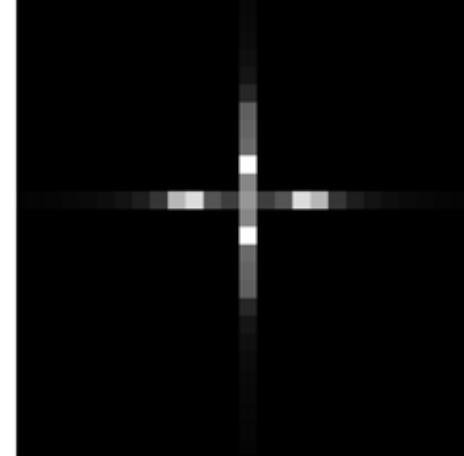
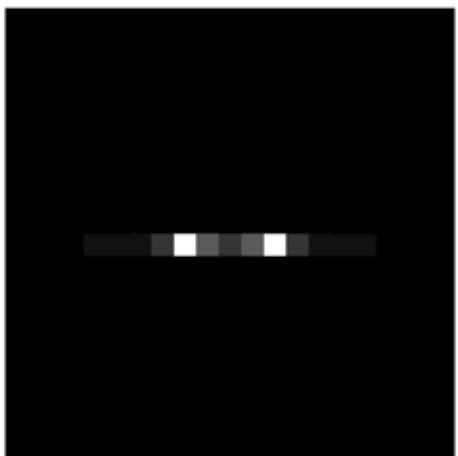
$$H(k_m, k_n) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h(m, n) e^{-2\pi j \frac{k_m m + k_n n}{MN}}$$

Fourier analysis in images

Intensity images



Fourier images



The Convolution Theorem

The Fourier transform of the convolution of two functions is the product of their Fourier transforms:

$$F[g * h] = F[g]F[h]$$

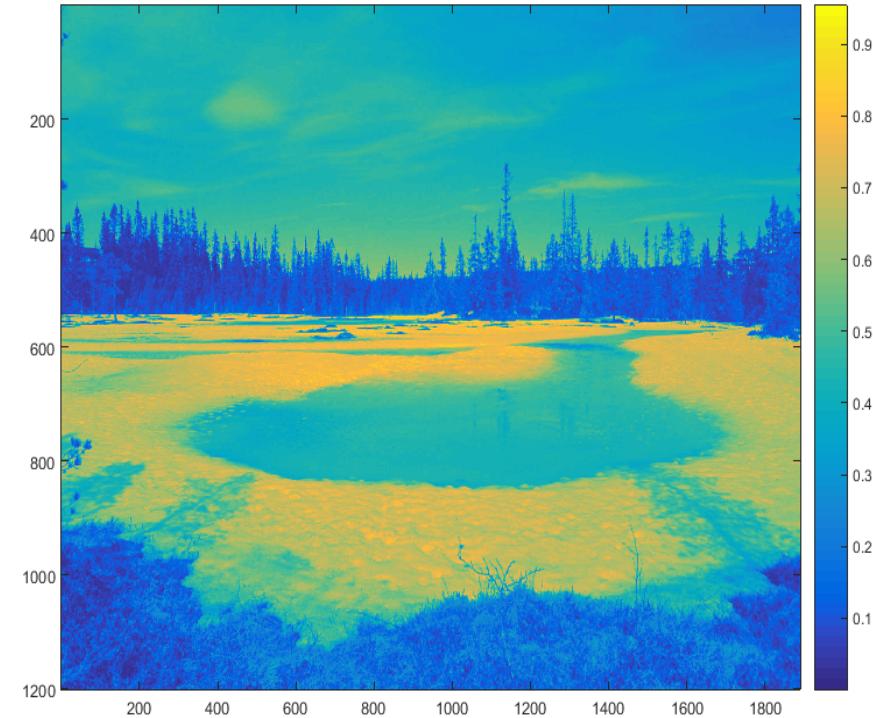
Convolution in spatial domain is equivalent to multiplication in frequency domain:

$$g * h = F^{-1}[F[g]F[h]]$$

Example – Gaussian (low pass) filtering

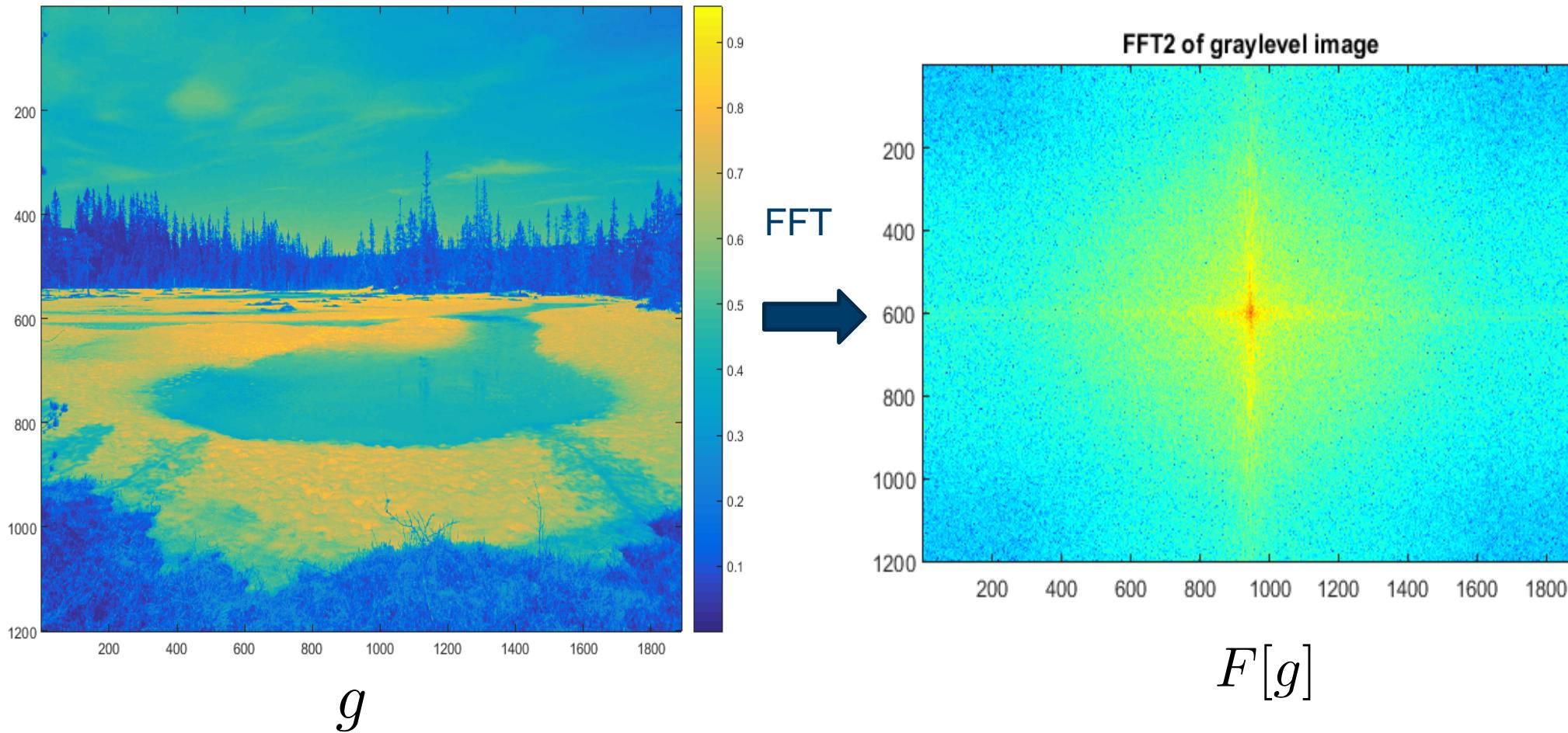


Original colour image

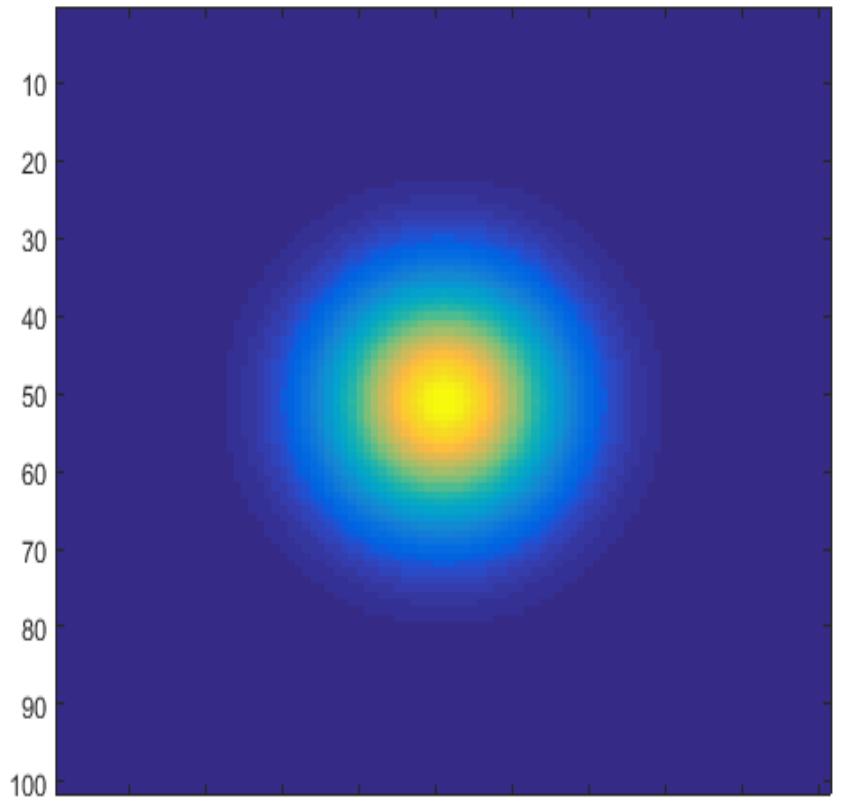


Gray level image

Example – Gaussian filtering

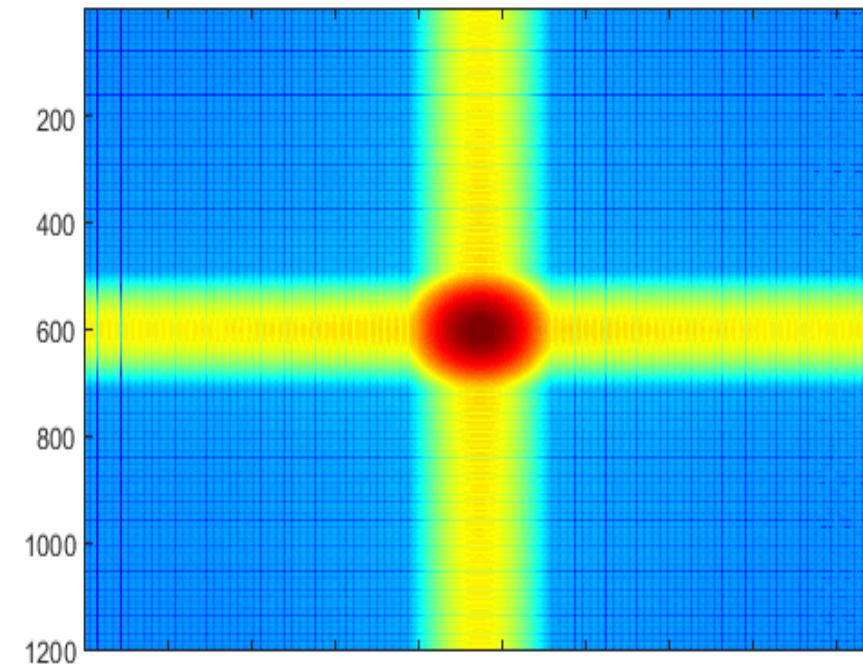


Example – Gaussian filtering



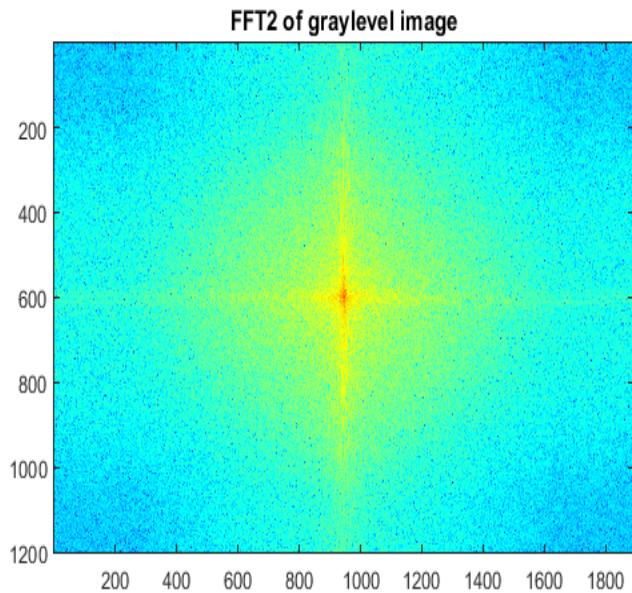
h

FFT

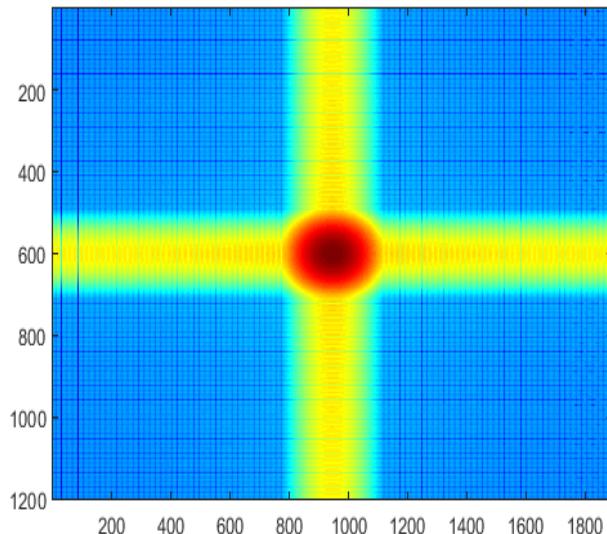


$F[h]$

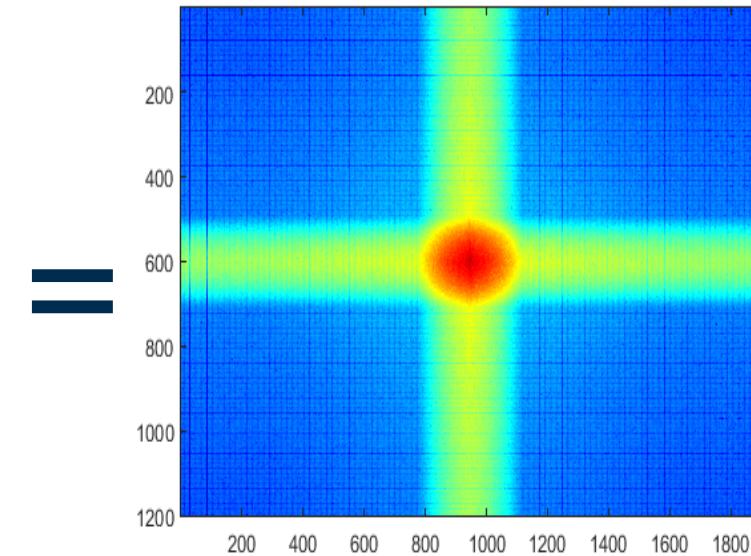
Example – Gaussian filtering



$$F[g]$$

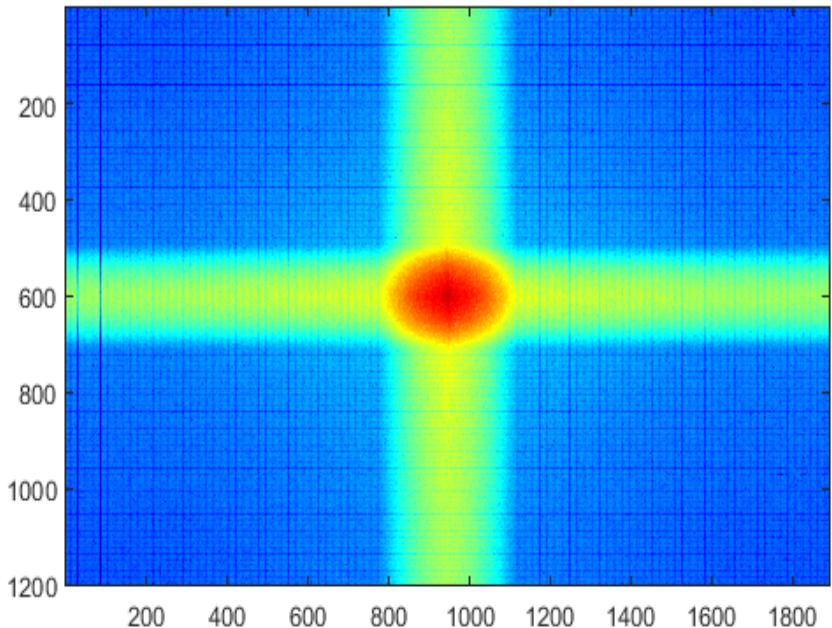


$$F[h]$$



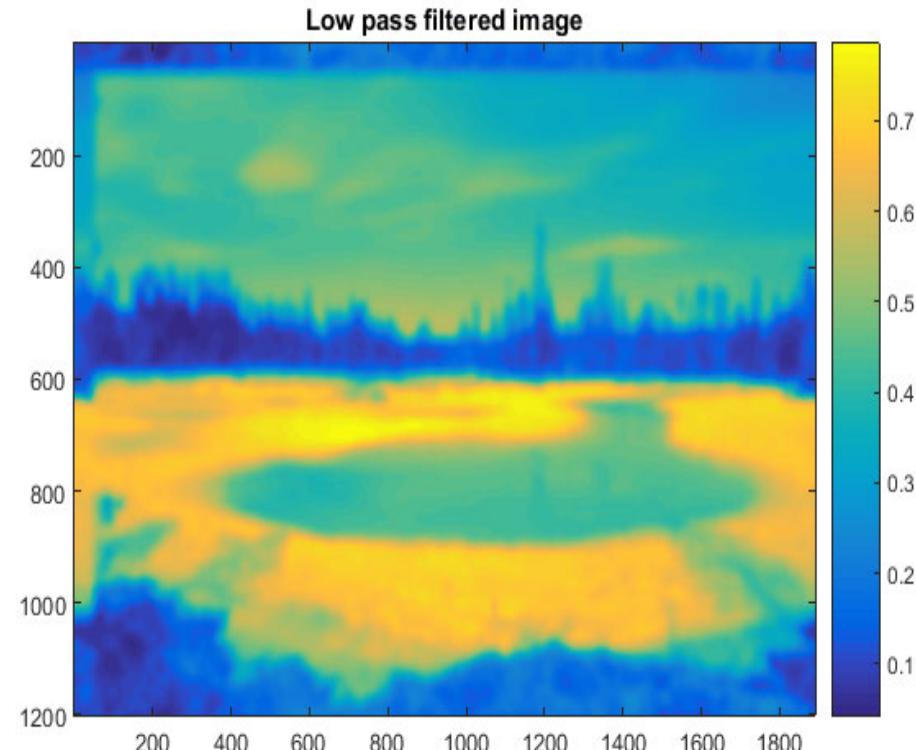
$$F[g]F[h]$$

Example – Gaussian filtering



$$F[g]F[h]$$

Inverse
FFT



$$g * h = F^{-1}[F[g]F[h]]$$

Summary

Image Processing

- Point operators
- Image filtering in spatial domain
 - Linear filters
 - Non-linear filters
- Image filtering in frequency domain
 - Fourier transforms
 - Gaussian (low pass) filtering



More information: Szeliski 3.1 – 3.4