Introduction to ImageJ
Session 2: Advanced image processing

Dimitri Vanhecke

March 2024

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

adolphe merkle institute
excellence in pure and applied nanoscience

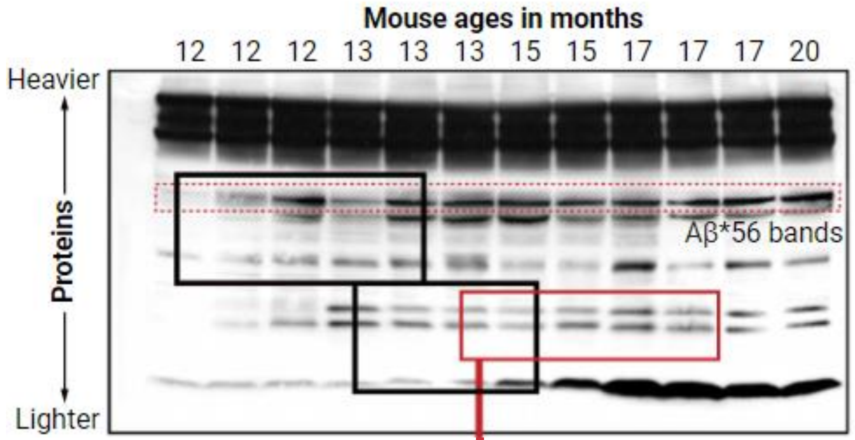
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

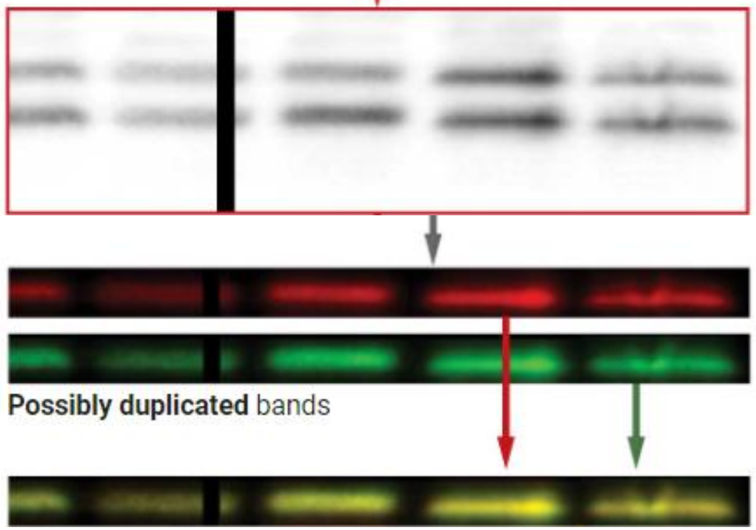
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

UNIVERSITÉ DE GENÈVE

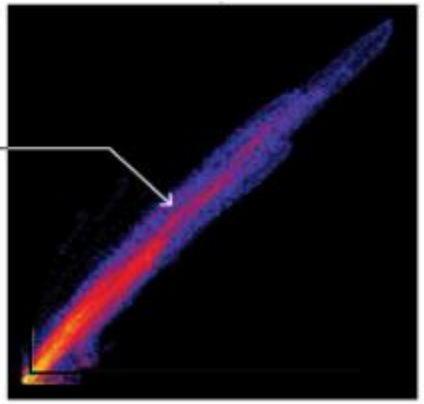
Swiss National Science Foundation

Preamble





This heat map shows one point for each group of pixels compared. Red indicates dense areas of the original image, such as the center of a band; purple indicates sparse areas.



Preamble

Associated Press
Code of Ethics for Photojournalists

AP pictures must always tell the truth. We do not alter or digitally manipulate the content of a photograph in any way.

swiss academies of arts and sciences

Code of conduct for scientific integrity

swiss-academies.ch

https://akademien-schweiz.ch/en/themen/scientific-culture/scientific-integrity-1/

THE OFFICE OF RESEARCH INTEGRITY

https://ori.hhs.gov/education/products/RIandImages/guidelines/list.html

(My) rule of thumb

- Always perform an algorithm on all pixels
- Be conservative in using filters/alogrithms/...

Overview

Part I: Transformations
Part II: Point operations
Part III: reciprocal space
Part IV: Filters
Part V: Machine learning

Part I: Transformations

Transformations

Image > transform

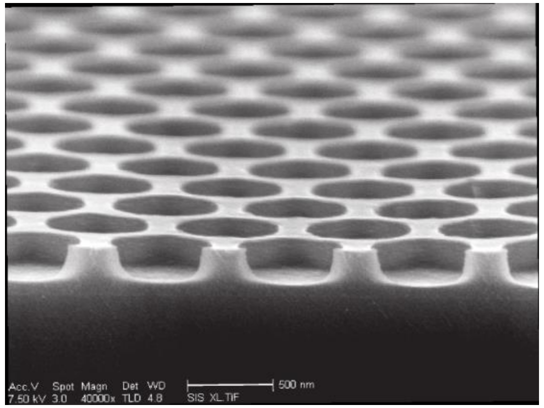
No problem	Can be a problem
Flip	Scale/bin
Rotate	Aliasing
Translate	Crop

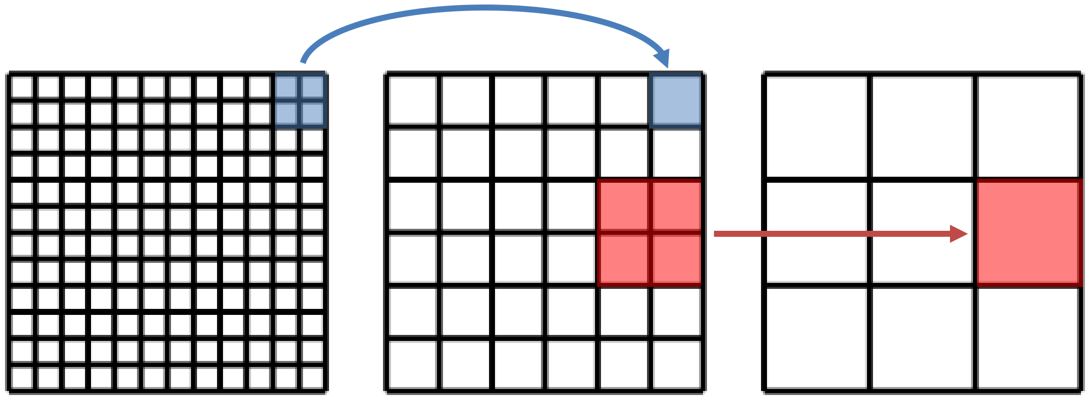
These transformations could be equally well made at the microscope

Rule (of thumb)
You must perform every function on every pixel in the image, not just on some selected pixels

Binning / scaling

	Binning	Scaling
Location	On the camera chip	In silico/postprocess
Algorithm	Integration or summation	Summation, averaging, …
Factor	2 (1,2,4,8,16, …)	Free
Interpolation	no	Yes





No binning (144 pixels)

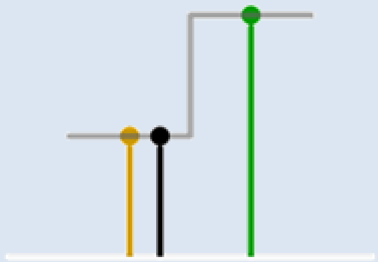
2x binning (36 pixels)

4x binning (9 pixels)

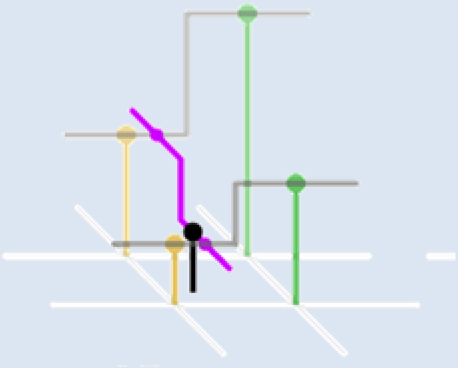
Scaling (interpolation)

Nearest Neighbour
= unweighted
Take the value of the closest voxel

1D NN: closest of two points

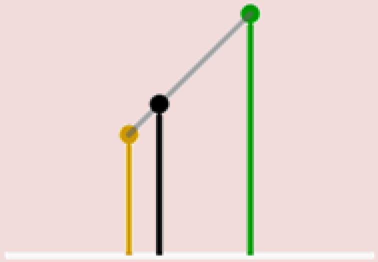


2D NN: closest pixel of four corners of a square

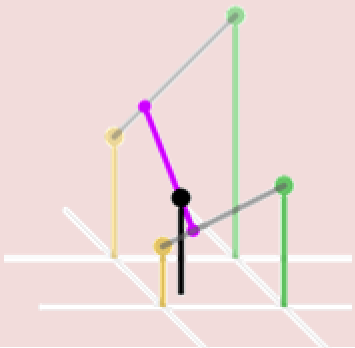


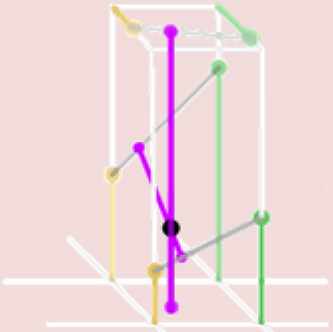
Linear
= Center of mass
Take the linear average of the two pixels the ray is intersecting

1D Linear: Center of mass of two points



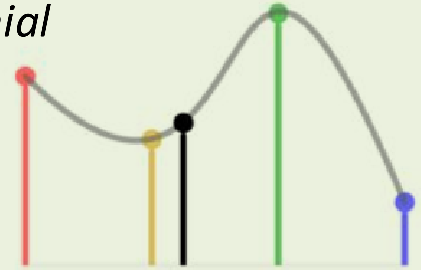
Bilinear: Center of mass of square corners
Trilinear: Center of mass of cube lattice points



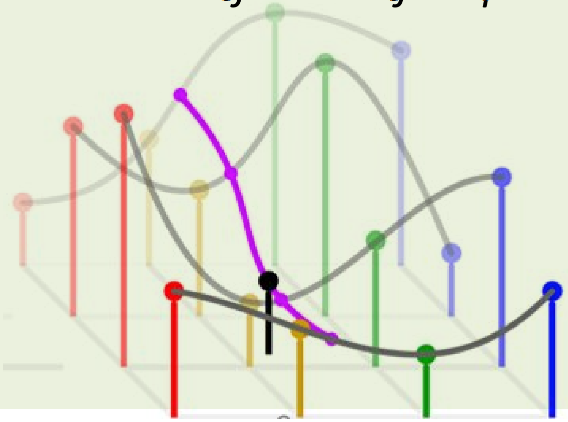


Cubic
Center of mass
= Lagrange polynomials, cubic splines or cubic convolution

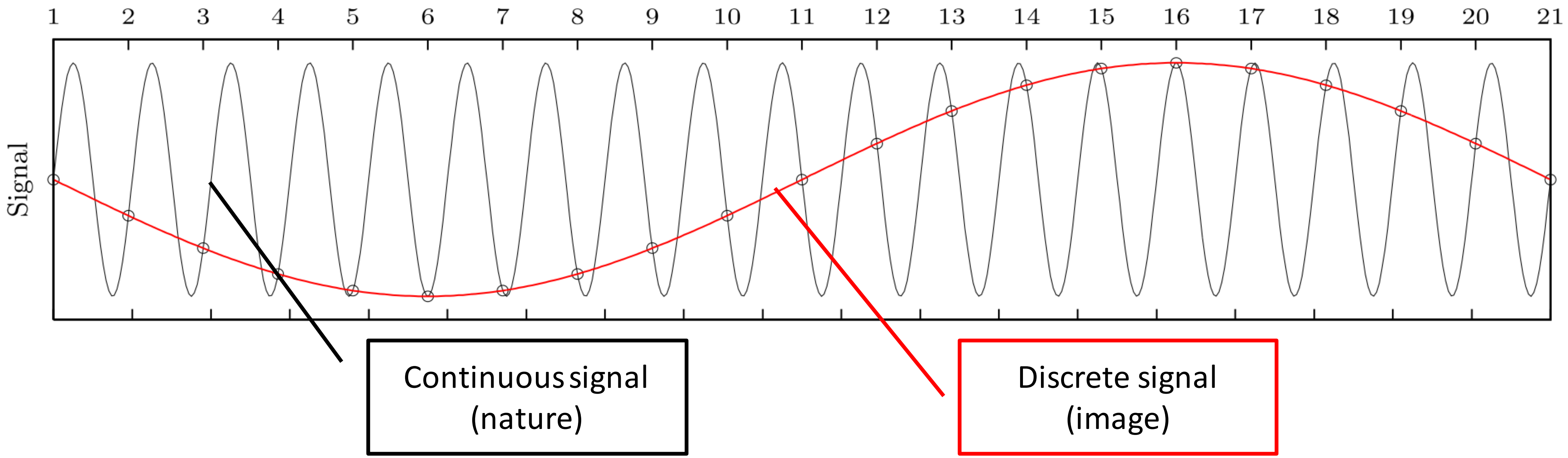
1D Cubic: Center of mass of 3^{th} degree polynomial



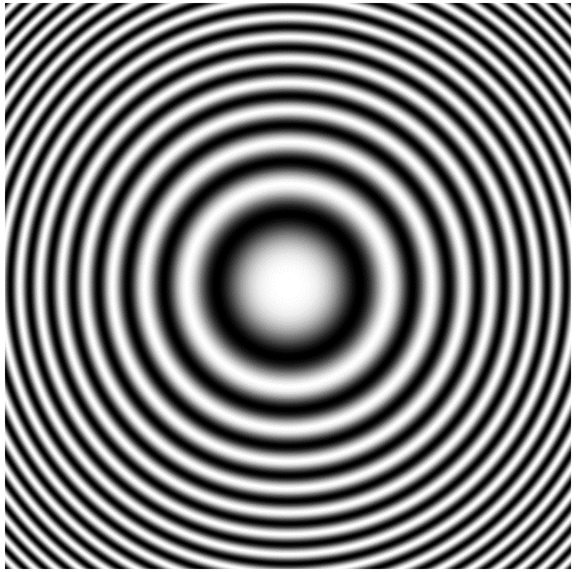
Bicubic: Center of mass of 16 pixels
Tricubic: Center of mass of 64 pixels



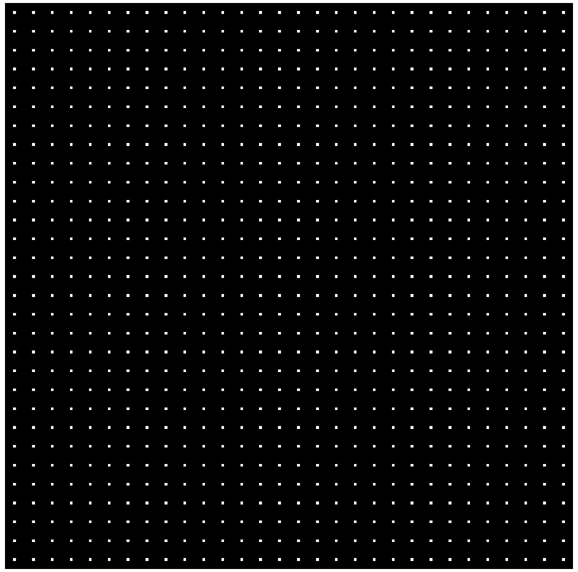
Aliasing effects



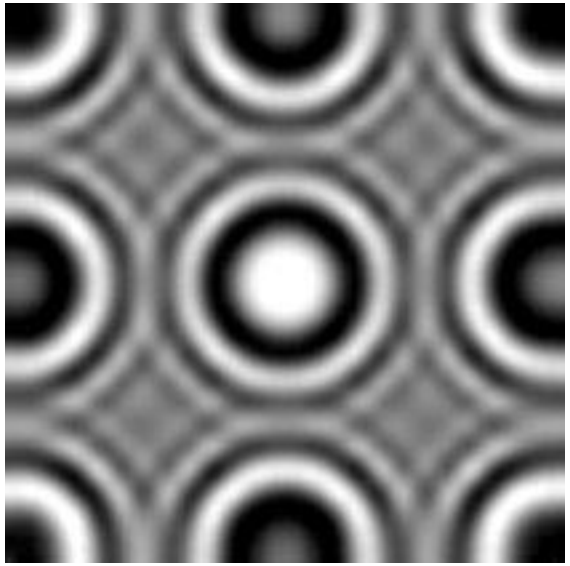
Aliasing: Example of spatial aliasing



Original (nature)



Camera / Detection grid
(= reconstruction filter)
(= “pixels” on the camera)
(= photosensitive element grid)
(= point-sampling grid)

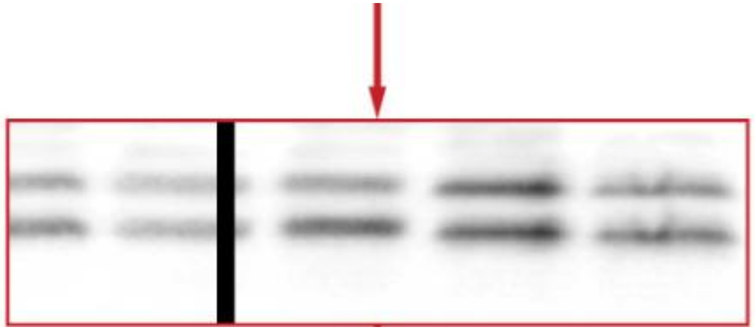


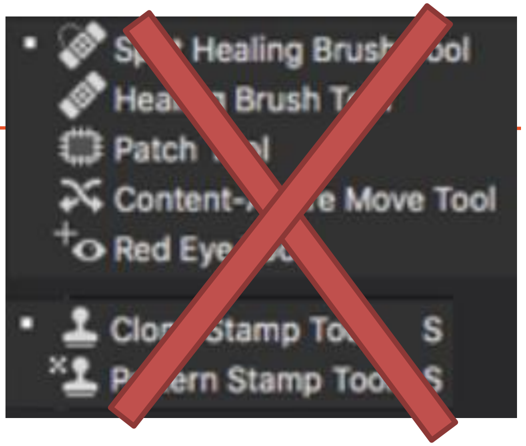
Image

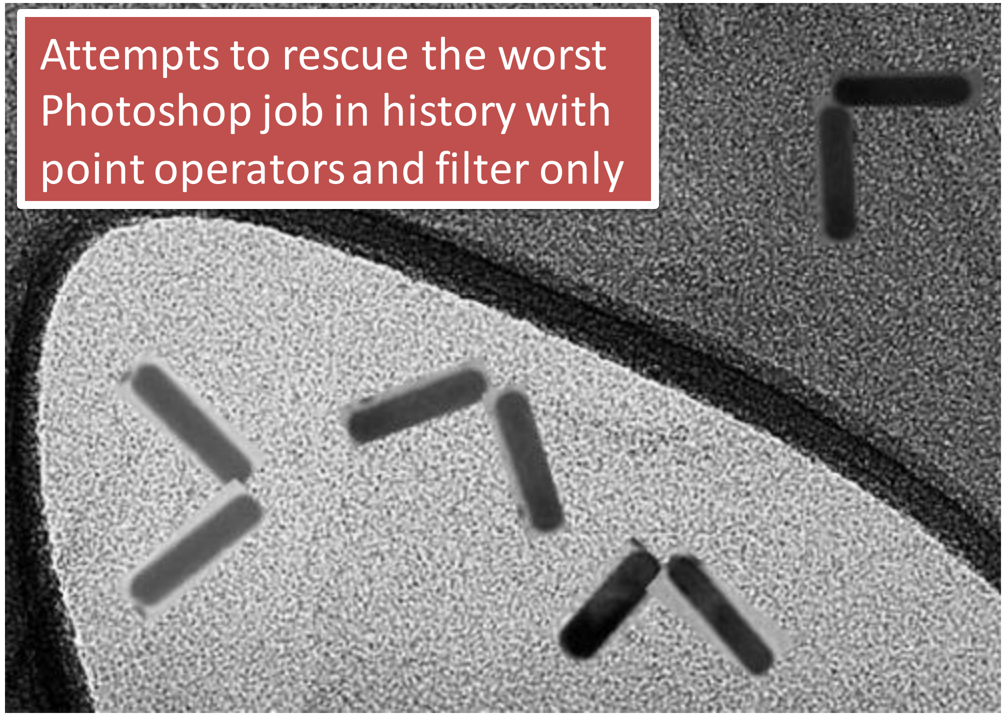
Solution: low pass filtering

See: **Shannon-Nyquist theorem**: 2x sampling otherwise one gets weird artefacts due to undersampling. However, continuous signals of nature will ALWAYS be undersampled.

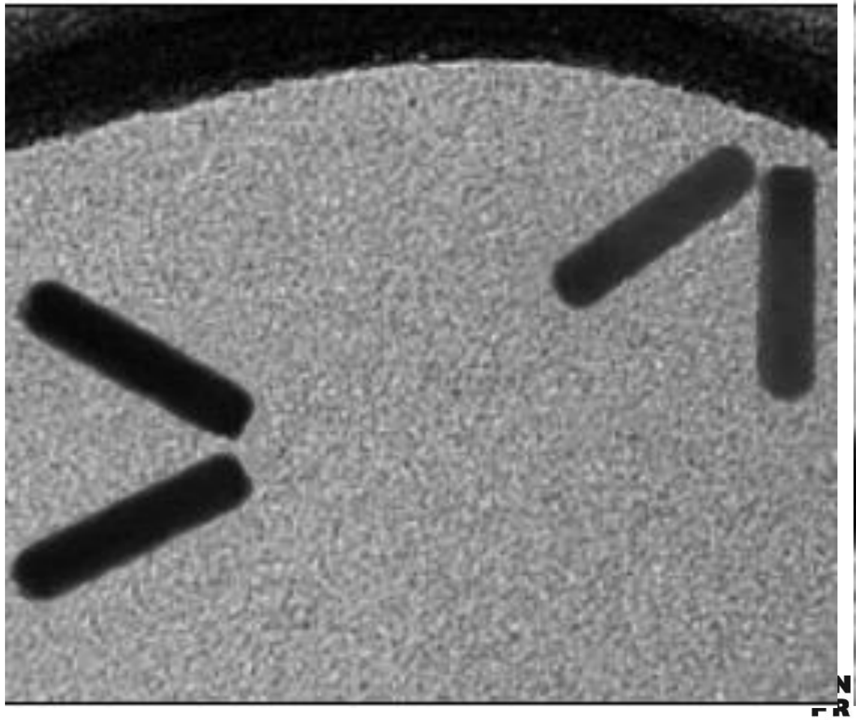
Transformations: Cropping

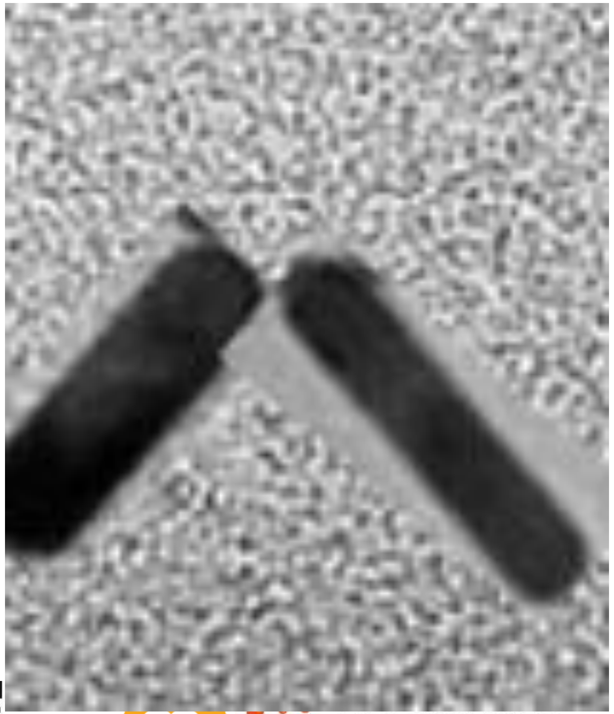






Attempts to rescue the worst Photoshop job in history with point operators and filter only





‘CryoTEM’ on Au nanorods chopsticks

Transformations: Cropping

This said...

- Cropping an image for a publication figure is usually considered acceptable.
- Consider your **motivation for cropping** the image**.**
    - Is the image being cropped to improve its "composition"
    - or to hide something that disagrees with the hypothesis?

- Legitimate reasons for cropping include:
    - Centering an area of interest
    - Trimming "empty" space around the edges of an image
    - Removing a piece of debris from the edge of the image

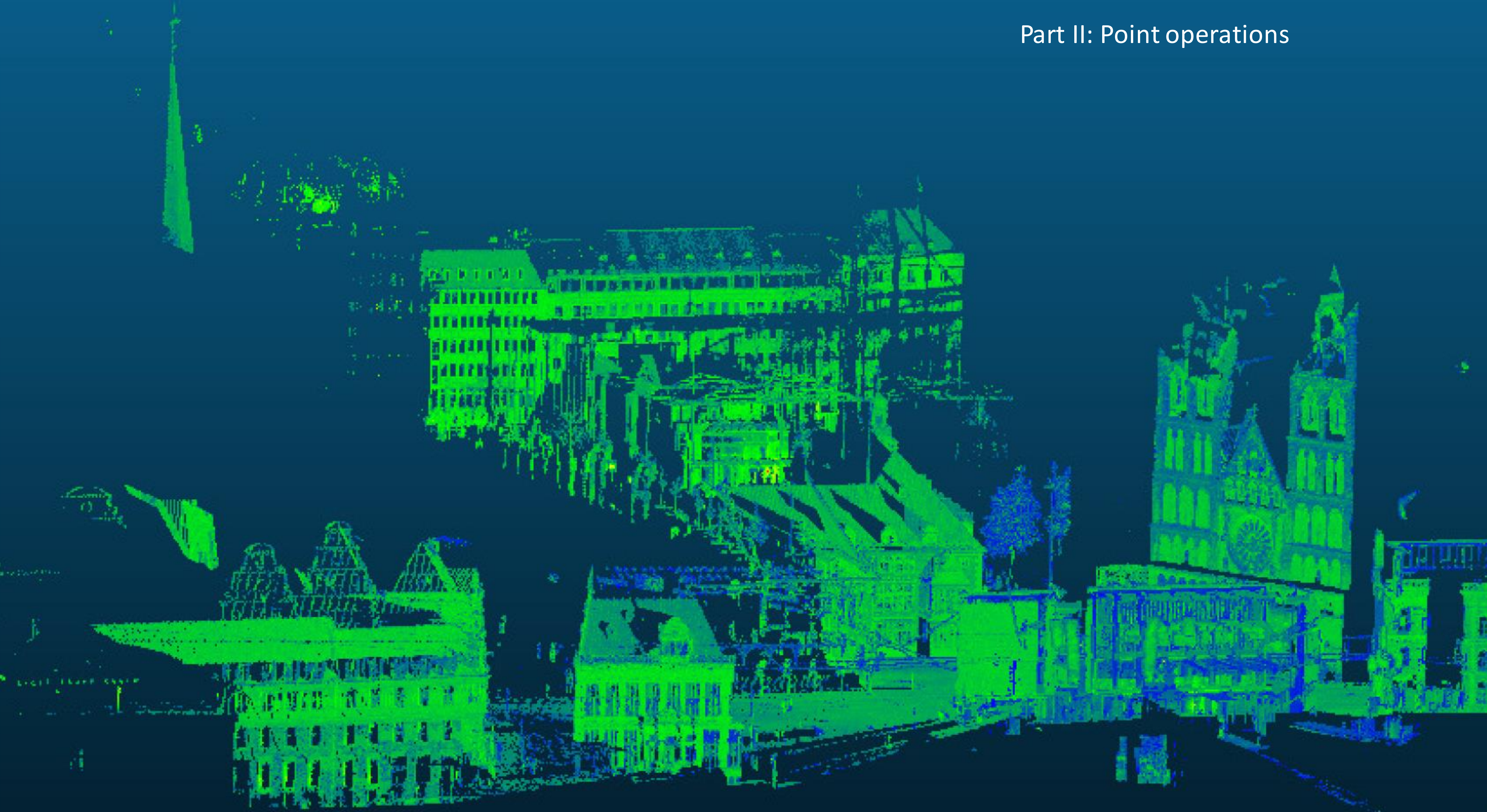
- Questionable forms of cropping: removing information in a way that changes the context. Examples:
    - Cropping out dead or dying cells, leaving only a healthy looking cell
    - Cropping out gel bands that might disagree with the hypothesis

Don't crop too much
Remember the 300 DPI requirement: you need pixels.

Do not let image manipulation ruin good science

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Part II: Point operations



Point operations

Basic concept:

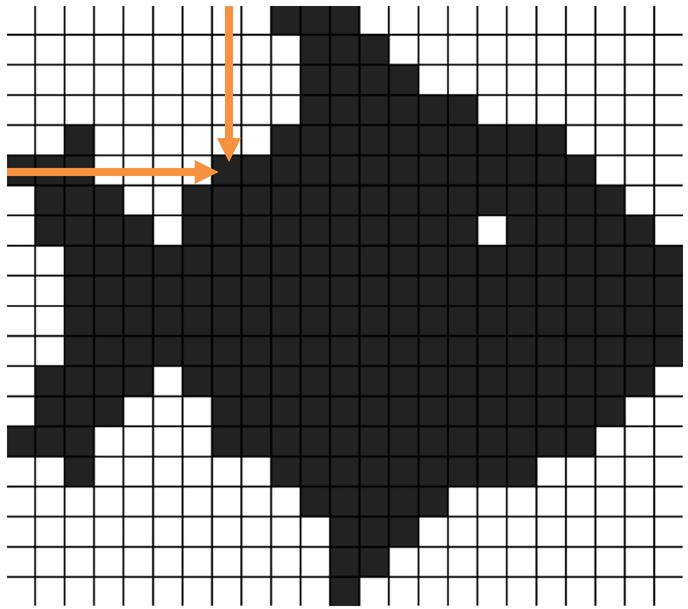
```
for a(u,v)
          a'(u,v) = f(a(u,v))
next
```

```
For each pixel in the image
          Take pixel intensity and perform a function
Go to the next pixel
```

The new (processed image) contains pixel intensities in a’

U= image width,
V= image height,
u = a given position along the horizontal axis
v= a given position along the vertical axis
a(u,v) = the grayscale value in position u, v

U= 23
V= 20
u = 8
v= 6
a(u,v) =  = 30



Point operations: Addition, subtract, multiply and divide

Basic concept:

This is called the «mapping function»

```
for a(u,v)
        a'(u,v) = f(a(u,v))
next
```

```
For each pixel in the image
        Take pixel intensity and perform a function
Go to the next pixel
```

The new (processed image) contains pixel intensities in a’

Add

`a'(u,v) = a(u,v) + B`

Adds a constant (B) to each pixel value (value increases)

Subtract

`a'(u,v) = a(u,v) - B`

Subtracts a constant (B) from each pixel value (i.e. Mean brightness decreases)

Multiply

`a'(u,v) = C x a(u,v)`

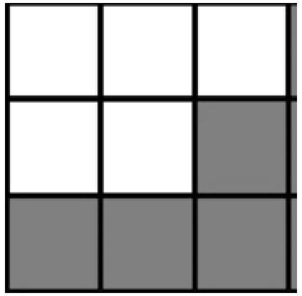
Multiplies each pixel value with a constant (C)

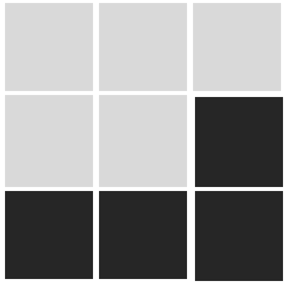
Divide

`a'(u,v) = 1/C x a(u,v)`

Divides each pixel value with a constant (C)

$$\begin{bmatrix} 255 & 255 & 255 \\ 255 & 255 & 130 \\ 130 & 130 & 130 \end{bmatrix} \xrightarrow{B=-50} \begin{bmatrix} 205 & 205 & 205 \\ 205 & 205 & 80 \\ 80 & 80 & 80 \end{bmatrix}$$





UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

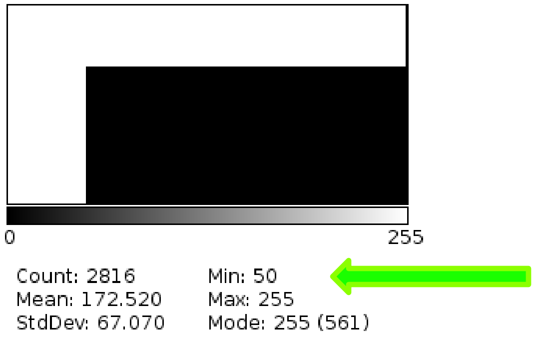
BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

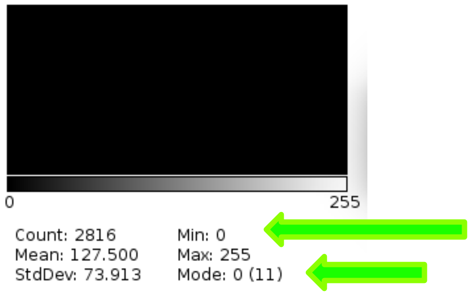
Point operations: Addition, subtract, multiply and divide

EXERCISE 1
Open Example 1 – GrayScale LUT and probe the effect of mathematical point functions add, subtract, multiply and divide on the histogram (CTRL + h or analyze > histogram)

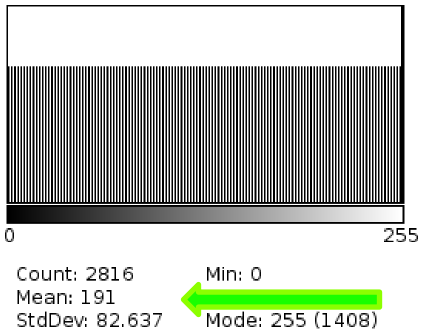
Process > Math > Add
Process > Math > Subtract
Process > Math > Multiply
Process > Math > Divide

Add 50





Multiply by 1.5

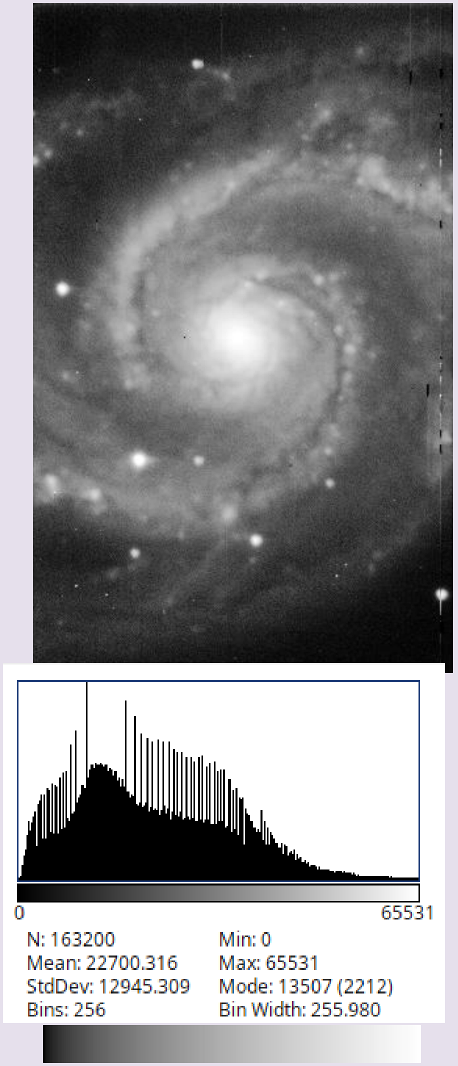


Open the Brightness/Contrast tool (auto)

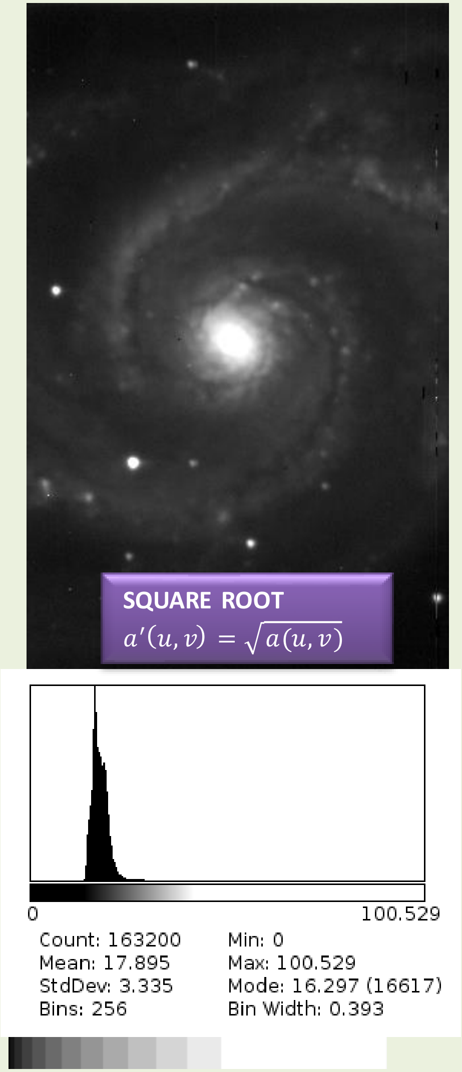
Point operations: Non-linear pixel value stretching

Normalized

Equalized



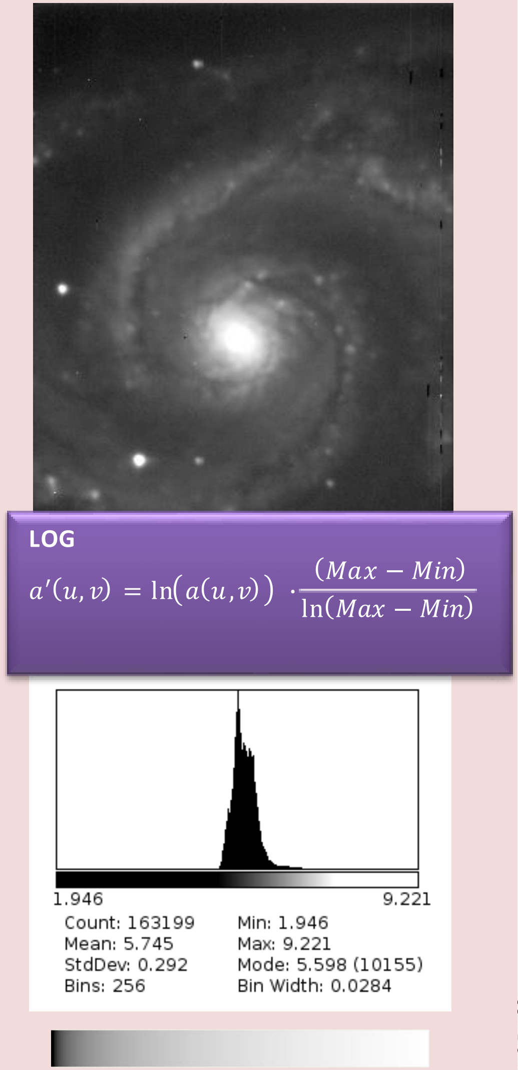
Square root



SQUARE ROOT

$$a'(u,v) = \sqrt{a(u,v)}$$

Log stretched



LOG

$$a'(u,v) = \ln\big(a(u,v)\big) \cdot \frac{(Max - Min)}{\ln(Max - Min)}$$

Point operations

EXERCISE 2
Open Example 2 (diffraction) and probe the effect of mathematical point functions (LOG, EXP,…)

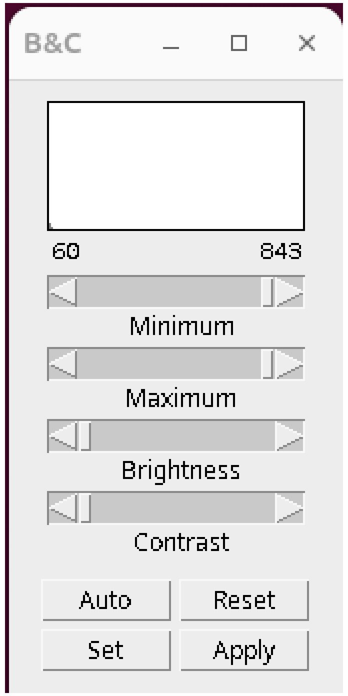
- Open the TIF image (Example 2 – Diffraction.tif)
- Adjust the brightness / contrast: Image > Adjust > Brightness / contrast (click 'Auto')

Try:

- Process > Math > log
- Process > Math > exp

Check the histograms of the processed images.

CTRL+SHIFT+d to duplicate the raw data to a new image
Be ready with the transfer function window (contrast/brightness) to adjust



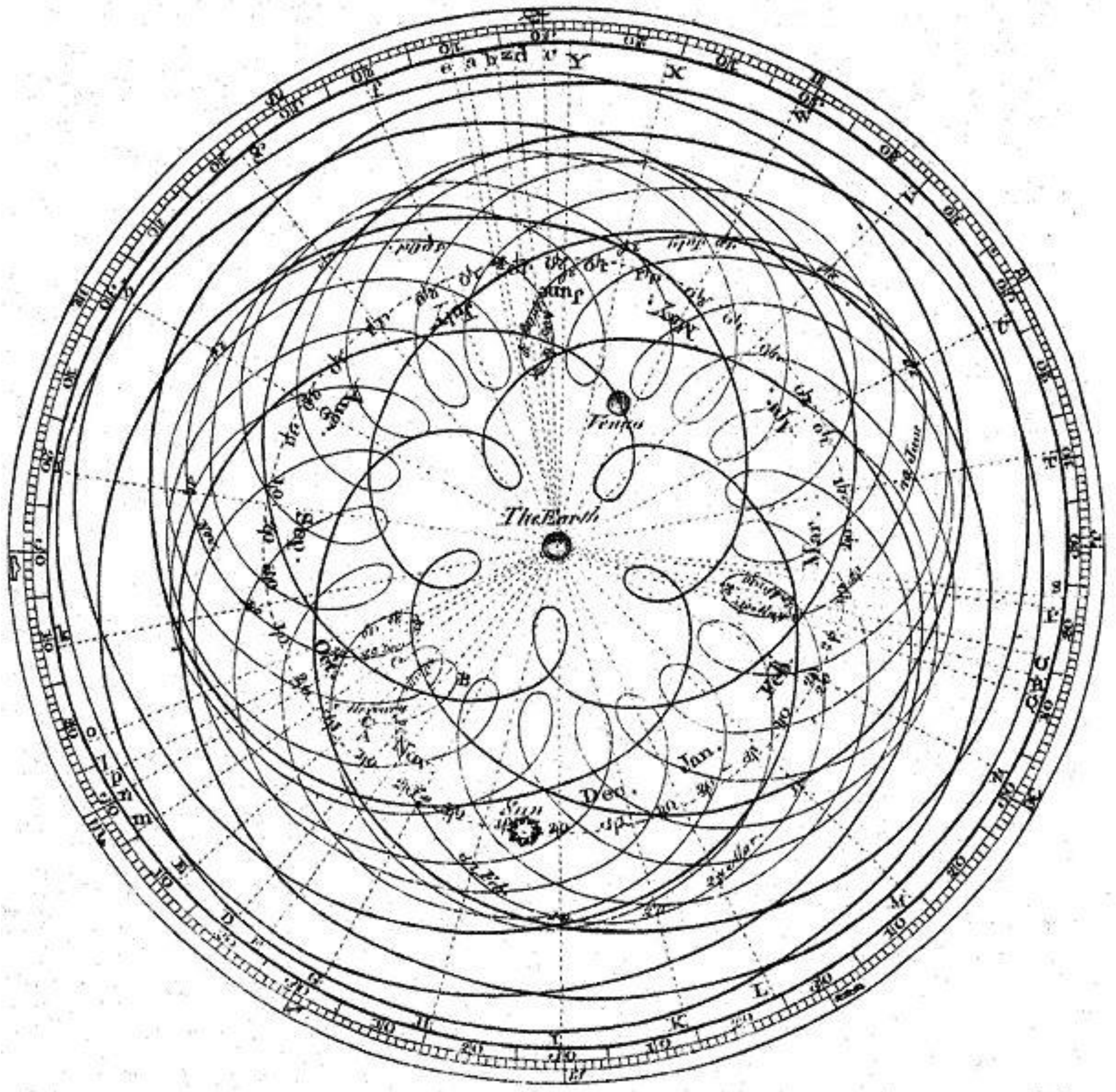
Point operations: Summary

Point operations to a defined mapping function:

Point operator processing is a simple method of image processing. This technique determines a pixel value in the processed image dependent only on the value of the corresponding pixel in the input image.

Part III: Fourier transforms

Reciprocal space



Ancient Greeks (BC)
The sun, moon, the planets move around the Earth in circles.

Ptolemy (100 AD)
Wrong: if you watch the planets carefully, sometimes they move backwards.
Therefore: The planets still move around Earth, but describe little spring-like trajectories at the same time.

Galilei (1600 AD)
Wrong: The sun is the center
(Wrong again… the church is against it)

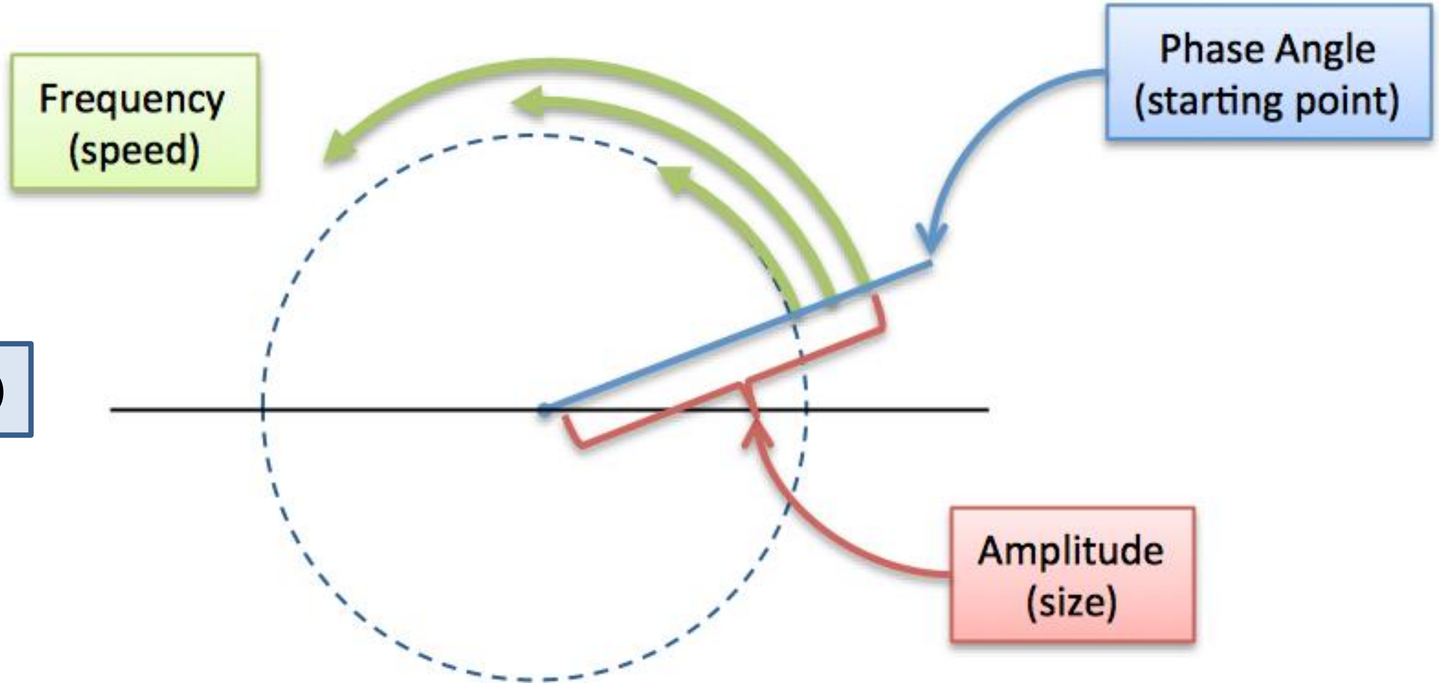
Fourier (1800 AD)
You can reconstruct any signal alias by summing a large number of smaller «epicycles»

The Fourier Transform

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

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx, \forall\ \xi \in \mathbb{R}$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x \xi} d\xi, \forall\ x \in \mathbb{R}$$

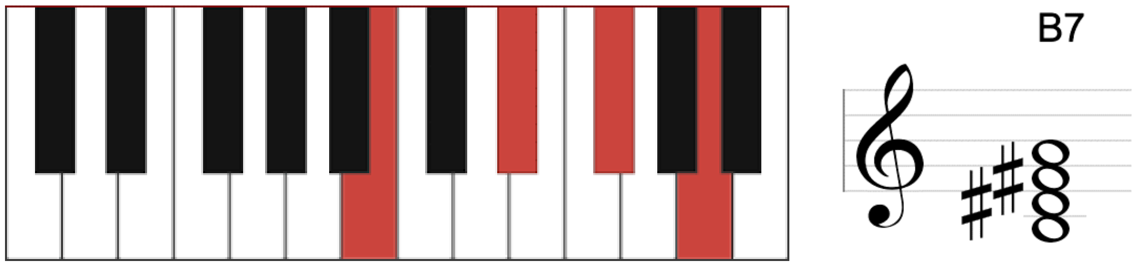


https://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform/

Fourier transform: reciprocal space

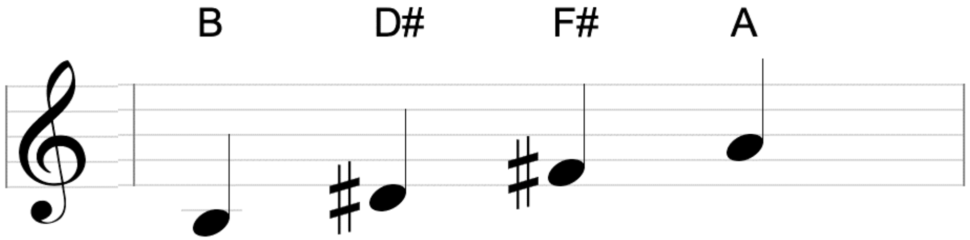
Real space

B7 chord



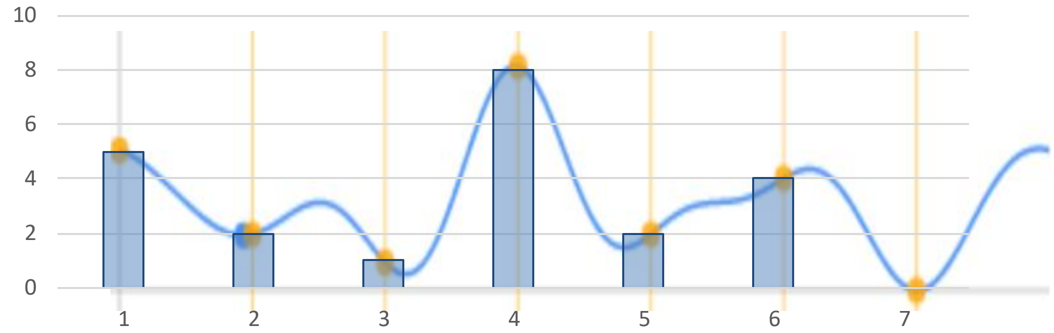
Reciprocal space

- B.
- D#
- F#
- A.

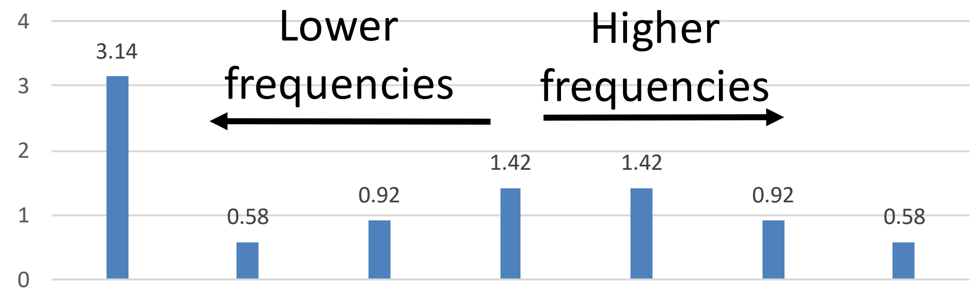


The Fourier Transform

Data Real space	5	2	1	8	2	4	0



Fourier Space
(or reciprocal space)



Phase: 0	-162.2	12.9	-65.7	65.7	-12.9	162.2
Amplitude 3.14	0.58	0.92	1.42	1.42	0.92	0.58

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx, \forall\, \xi \in \mathbb{R}$$

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x \xi} d\xi, \forall\, x \in \mathbb{R}$$

Fourier transform: when your index is continuous. (Nature)

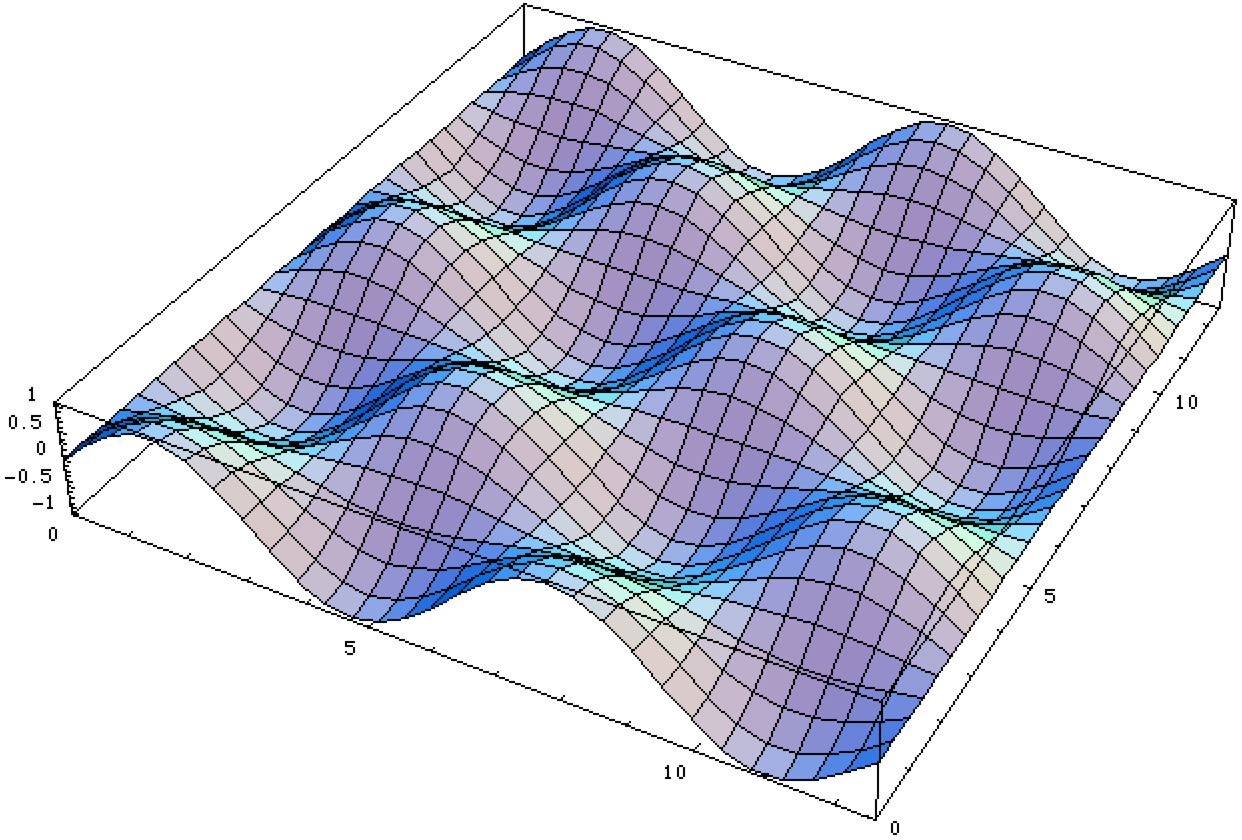
Fourier series: when your index is discrete.

Discrete Fourier series: For infinitely long but periodic signals

Discrete Fourier Transform: For general, finite length

Fast Fourier Transform: like DFT but with square images with w,h = 2^n (faster, more efficient)

The Fourier Transform – expanded to 2D

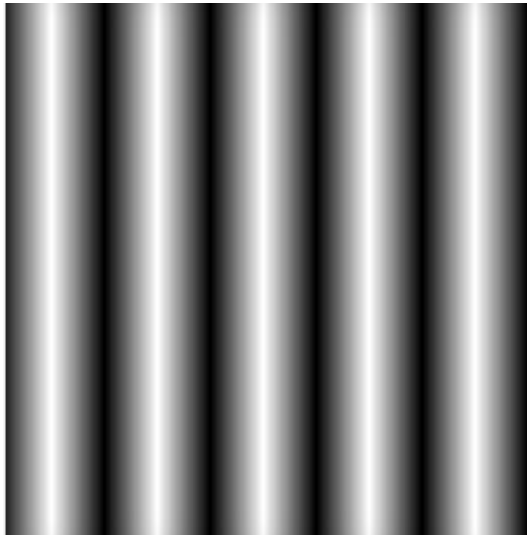


$$\hat{f}(\xi,\varrho) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(\xi x+ \varrho y)}dx\,dy\,,\forall\,\xi\,\epsilon\,\mathbb{R},\forall\,\varrho\,\epsilon\,\mathbb{R}$$

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}(\xi,\varrho)e^{2\pi i(\xi x+\varrho y)}d\xi\,d\varrho,\forall\,x\,\epsilon\,\mathbb{R},\forall\,y\,\epsilon\,\mathbb{R}$$

Fourier transform: reciprocal space

Real space



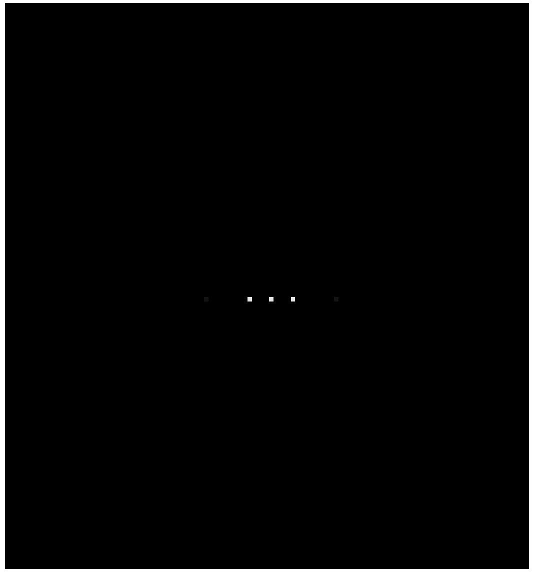
Easy: Intensity varies according to a sinoidal function

Fourier transform:

$$\hat{f}(\xi,\varrho) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(\xi x+\varrho y)}dx\,dy\,,\forall\,\xi\in\mathbb{R},\forall\,\varrho\in\mathbb{R}$$

For any real number ξ

Reciprocal space
= Fourier Space
= Power spectrum



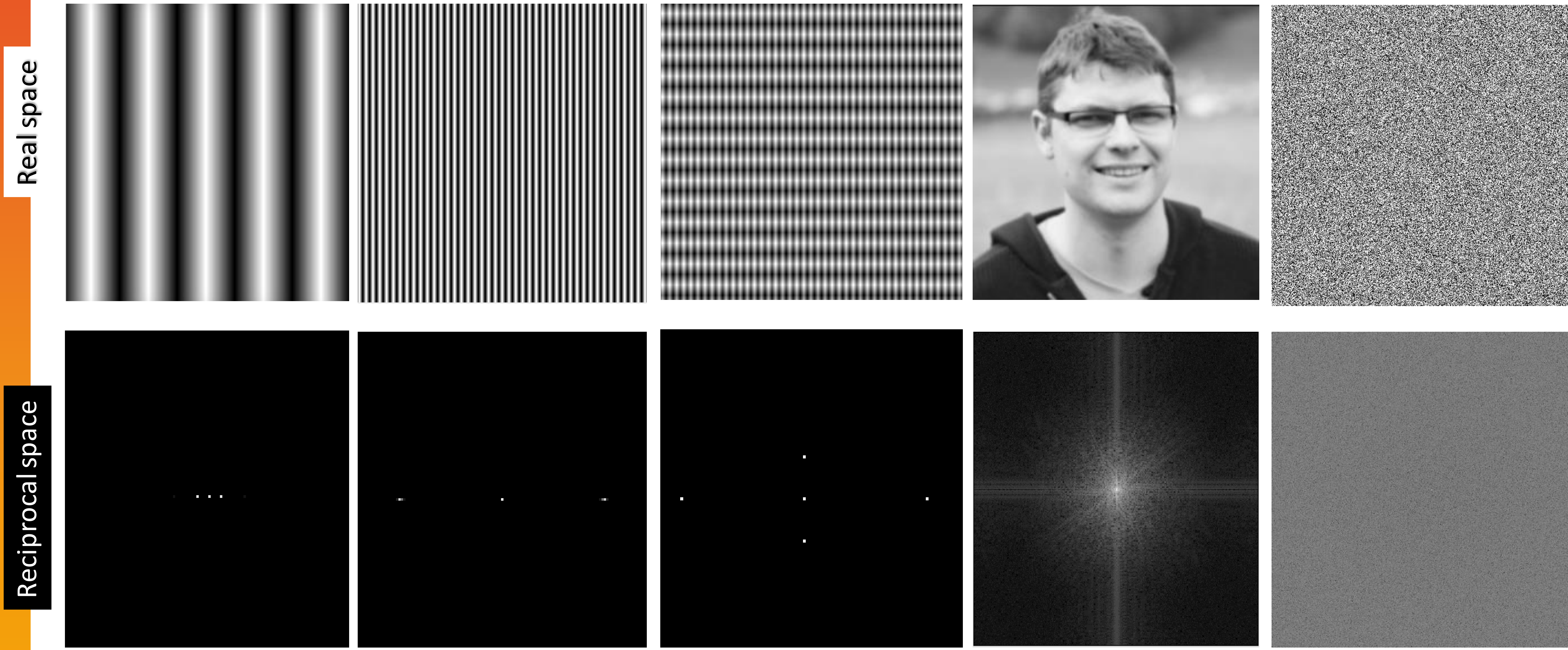
2 ‘delta functions’
And 1 central constant

Inverse Fourier transform:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}(\xi,\varrho)e^{2\pi i(\xi x+\varrho y)}d\xi\,d\varrho,\forall\,x\in\mathbb{R},\forall\,y\in\mathbb{R}$$

For any real number x

Fourier transform: reciprocal space (power spectrum)



UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transform: Note on Frequency & phase

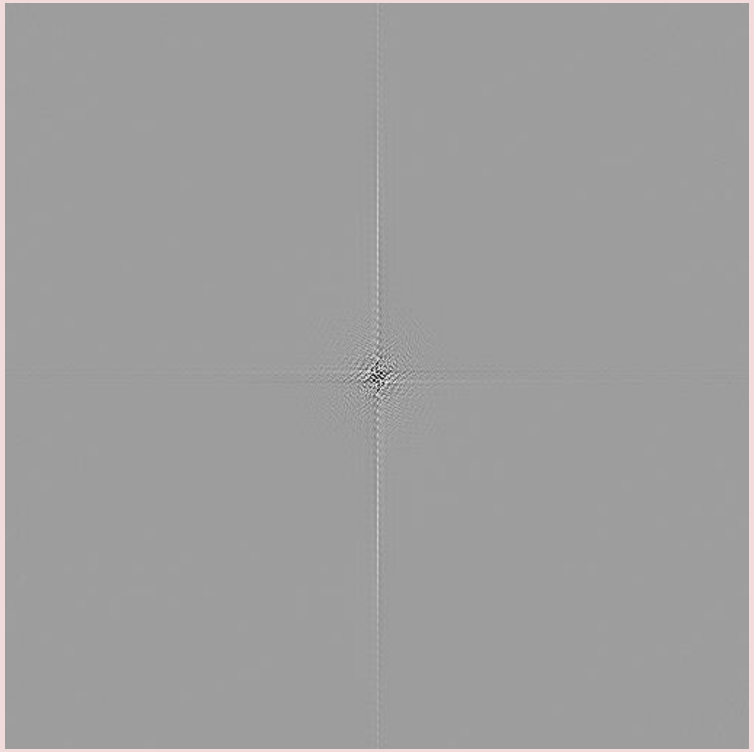
Real image $f(x,y)$

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \xi} dx, \forall\ \xi \in \mathbb{R}$$

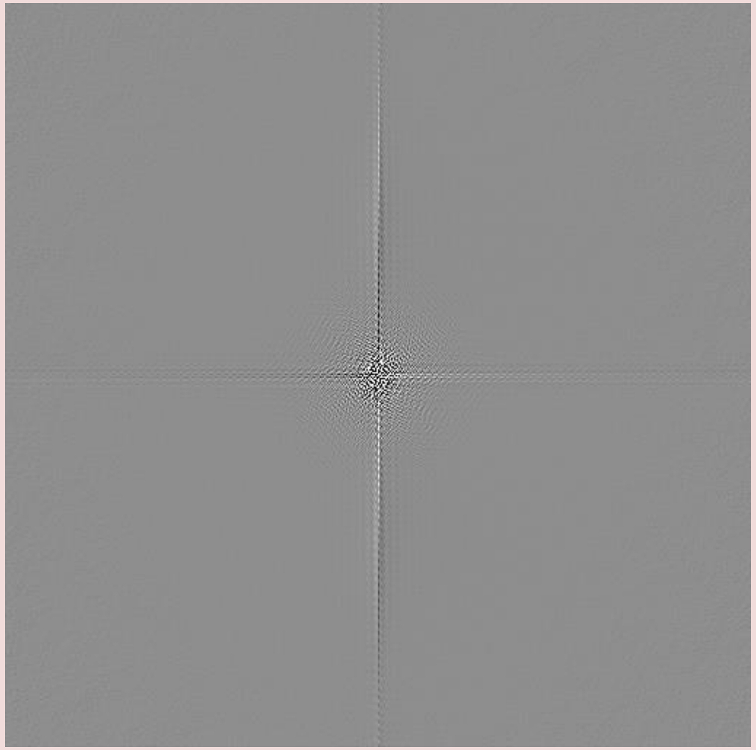
$$e^{-2\pi i x \xi} = \cos(2\pi\xi x) - i\ sin(2\pi\xi x)$$

$$\hat{f}(\xi) = R(\xi) + i\ I(\xi)$$

Fourier transform $\xi(u,v)$



Real



imaginary

Fourier transform: Note on Frequency & phase

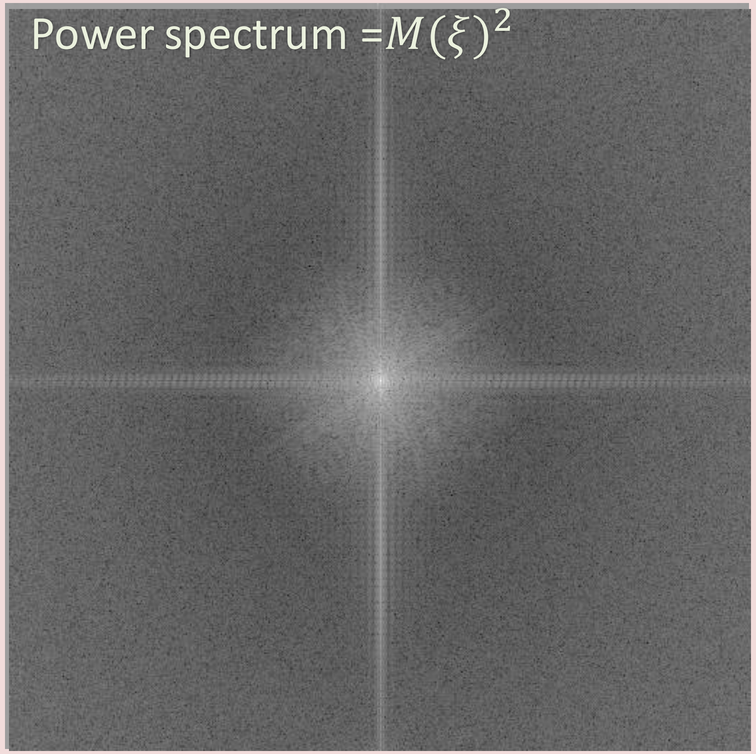
Real image $f(x,y)$

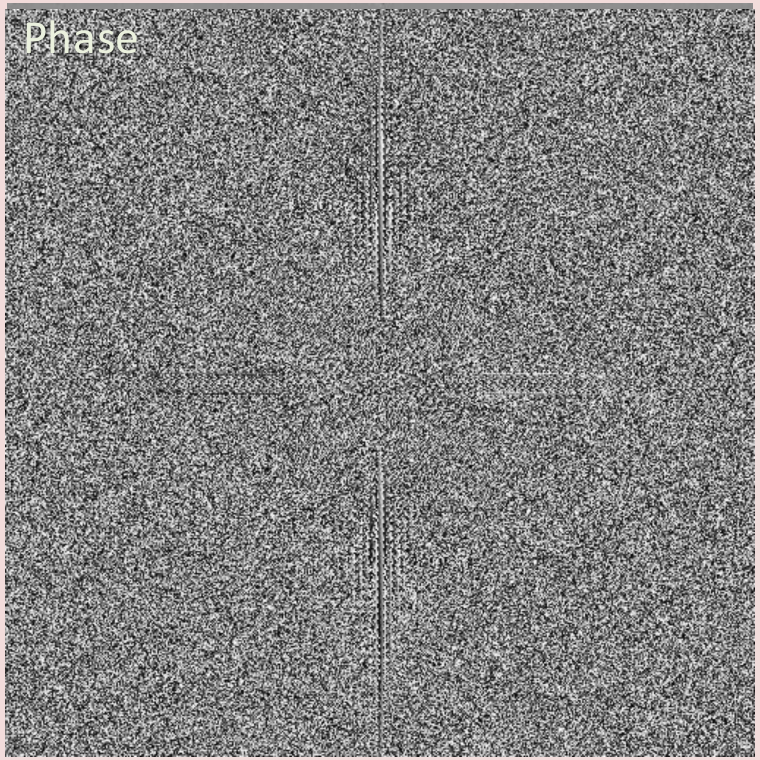
$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x\xi}dx, \forall\, \xi \in \mathbb{R}$$

$$e^{-2\pi i x\xi} = \cos(2\pi\xi x) - i\,\sin(2\pi\xi x)$$

$$\hat{f}(\xi) = R(\xi) + i\,I(\xi)$$

Fourier transform $\xi(u,v)$





Cartesian → Polar

Magnitude = $M(\xi) = \sqrt{R(\xi)^2 + I(\xi)^2}$ "how much" of a certain frequency component is present

Phase = $P(\xi) = tan^{-1}\left(\frac{I(\xi)}{R(\xi)}\right)$ "where" the frequency component starts

Fourier transformation: examples in image processing

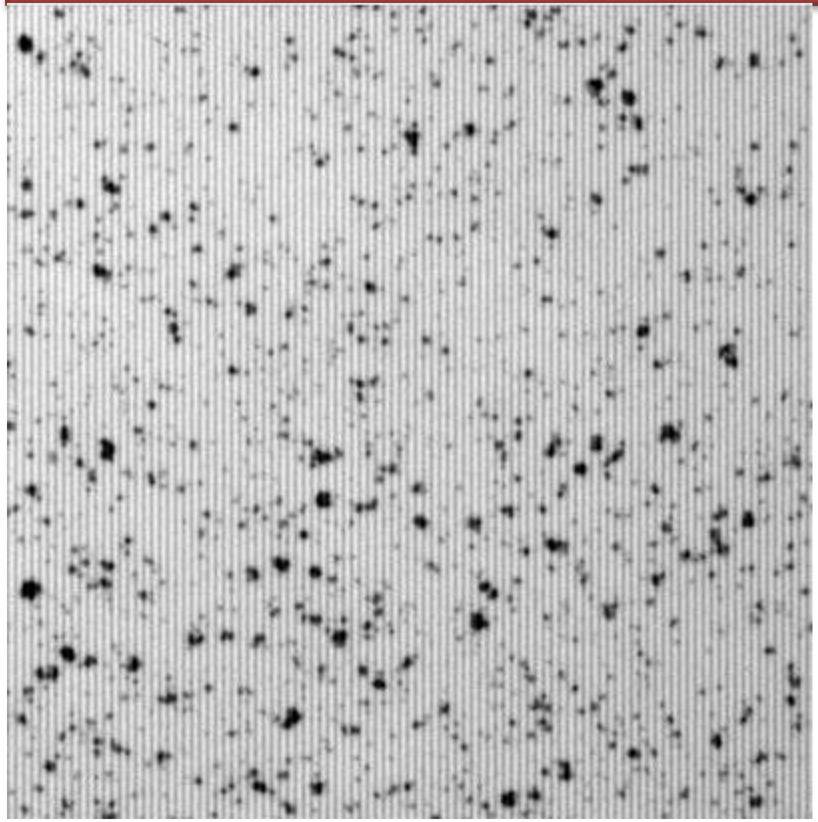
Some examples of fourier transform / image processing in reciprocal space:

- Removing repetitive noise
- Lowpass / anti-aliasing filters
- Bandpass filtering
- Assessing the resolution of an image
- Remove blur / Point spread function / motion blur
- Cross correlation

Videos and interactives (just google these):
3blue1brown Fourier Transform
Ptolemy and Homer (Youtube)
Ptolemy's spheres wolfram

Fourier transformation: filtering in Fourier space

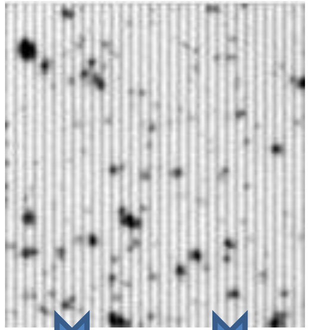
EXERCISE 3
Open Example 3A – repetitive noise (=multiplicative noise) and try to remove the repetitive noise using Fourier Filtering



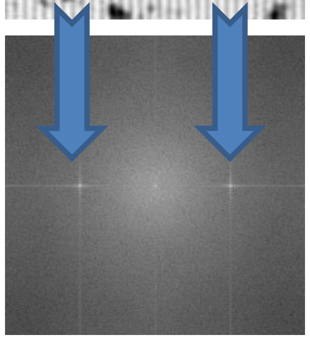
- FFT Example 3B
- Locate the 2 strong delta functions.
- Make a selection around the high frequency noise spots. Check if your foreground color is 'Black'
- Fill the area at the delta functions with black
- Inverse FFT

Fourier transformation: filtering in Fourier space

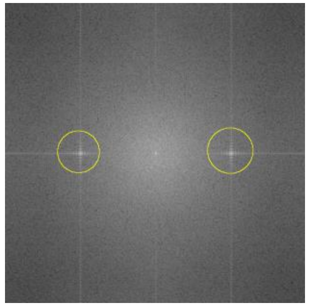
EXERCISE 3
Open Example 3 and Display an FFT. Try to remove the repetitive noise



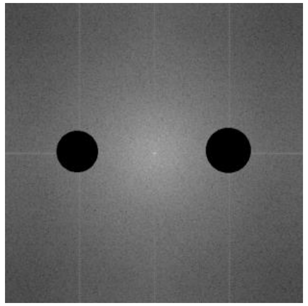
1. Open the data



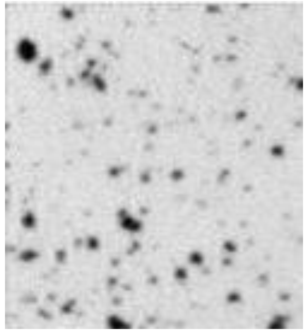
2. Make an FFT (Process > FFT > FFT)
Note the 2 strong Delta functions. These reflect the repetitive (sinoidal) noise in the image



3. Make a selection around the high frequency noise spots (hold shift to create 2 separate circles)



4. Edit > Clear
Or fill the selection with black pixels (CTRL+F), make sure that foreground color is black: edit > options > Colors...



5. Unselect the yellow selection.
6. Inverse the FFT (Process > FFT > inverse FFT)

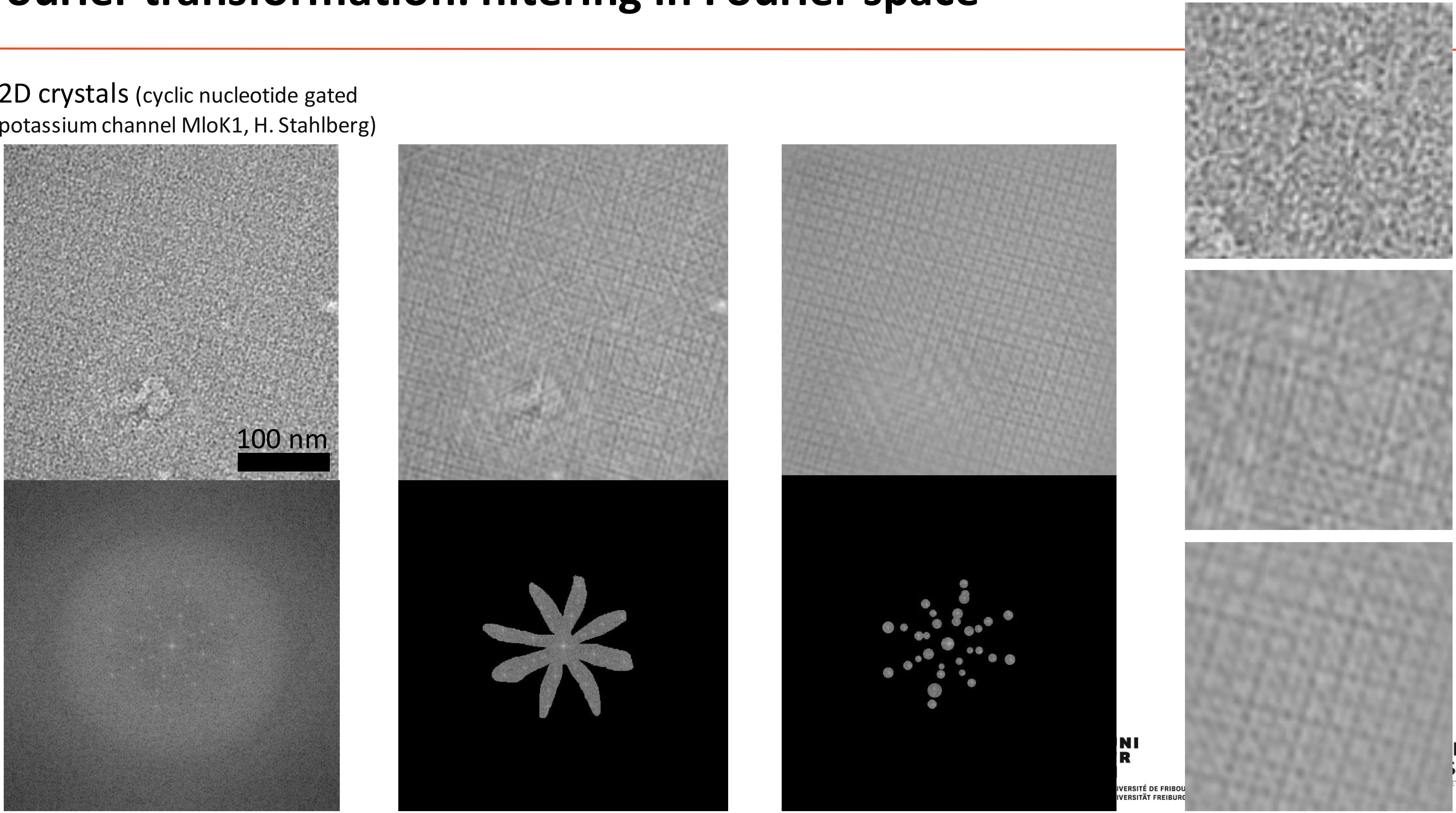
Note, in the FFT, the cursor position shows info like this: r=200 p/c (5). This is the radius of cycloid (=amplitude), the pixels per cylcoid and the frequency. Phase is not covered in this image

Remember Ethics...
Never change only part of the image... i.e. the real image

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: filtering in Fourier space

2D crystals (cyclic nucleotide gated potassium channel MloK1, H. Stahlberg)

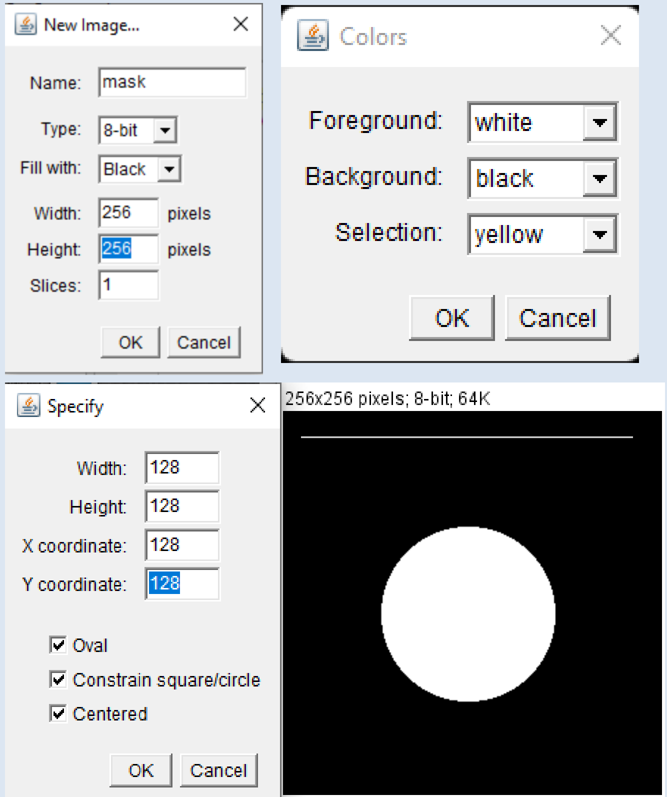


Fourier transformation: Lowpass filter

Masks in Fourier space:
Black = remove frequencies
White = pass (keep) frequencies

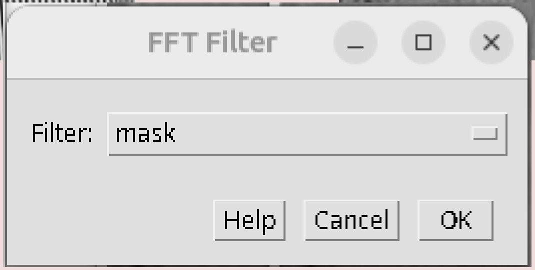
My first Fourier space filter

1. File > New > Image...
2. Pick white: Edit > Options > Colors
3. Specify a centered, round concentric circle (Edit > Selection > Specify)
4. And fill it (Edit > fill)
5. Rename the new imag"Mask" (Image > Rename...)



Apply your first Fourier space filter

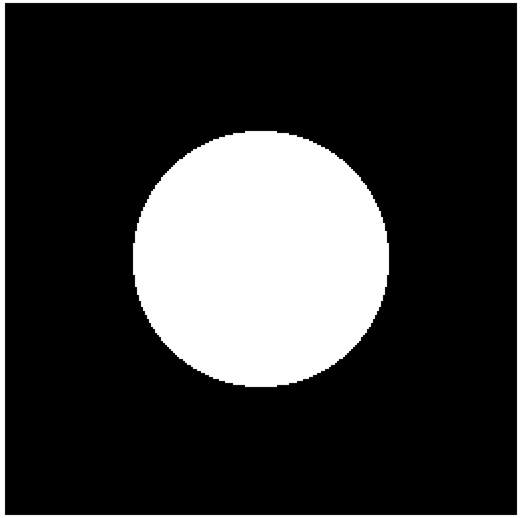
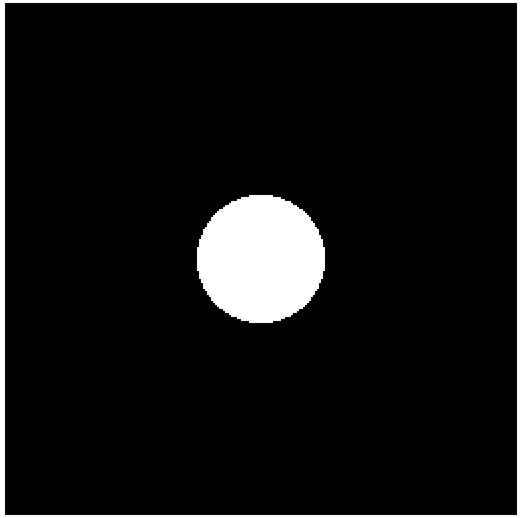
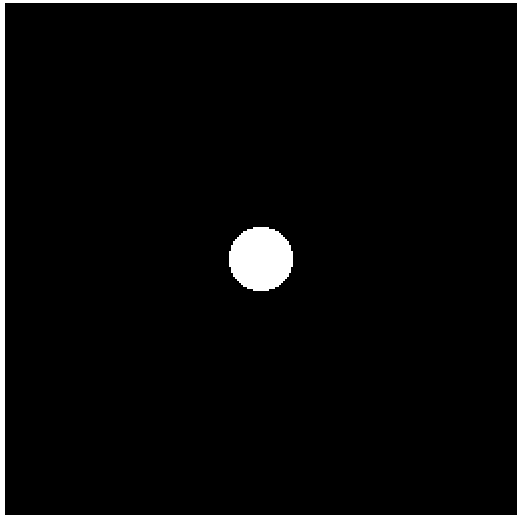
Analyze > FFT > Custom filter
Choose your mask

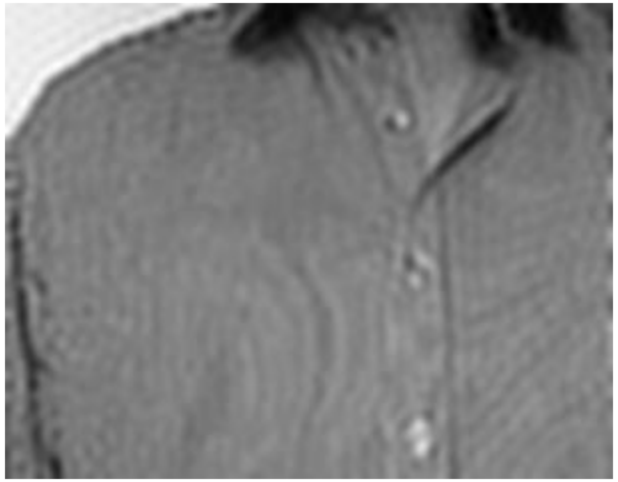
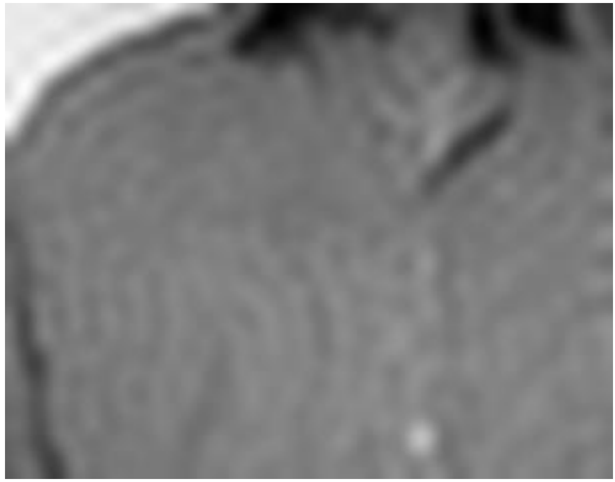


Aliasing / Moire: frequencies that are (just) above the resolution of the image

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: Lowpass filter an anti-aliasing filter

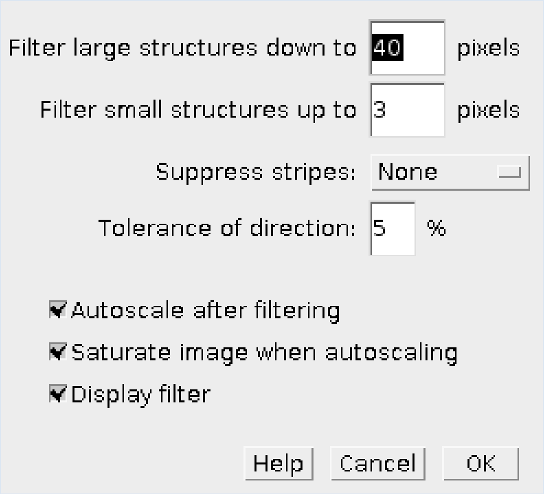
Hamming filter
Is a low pass filter with a Gaussian gradient. This reduces "ringing"

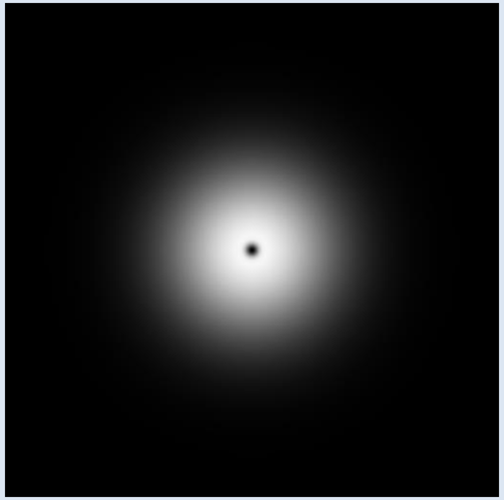
Fourier transformation: bandpass filter (Inverse notch filter)

Masks in Fourier space:
Black = remove frequencies
White = pass (keep) frequencies

Fourier bandpass filter

Analyze > FFT > Bandpass...





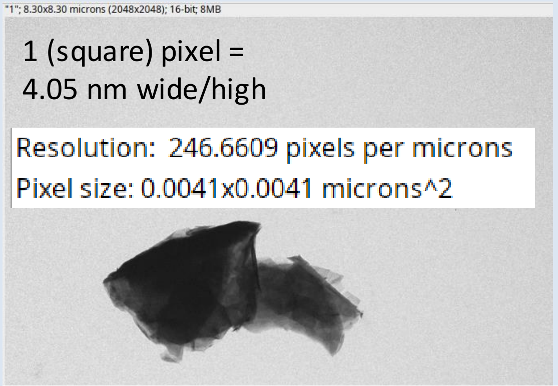
Filter large structures = high frequency cutoff (here 40 pixels / cycles)
Filter small structures = Low frequency cutoff (3 p/c)
Process > FFT > Custom filter allows to use your own filter

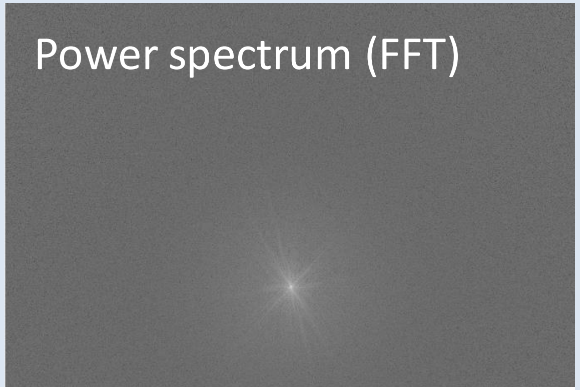
Apply your first Fourier space filter

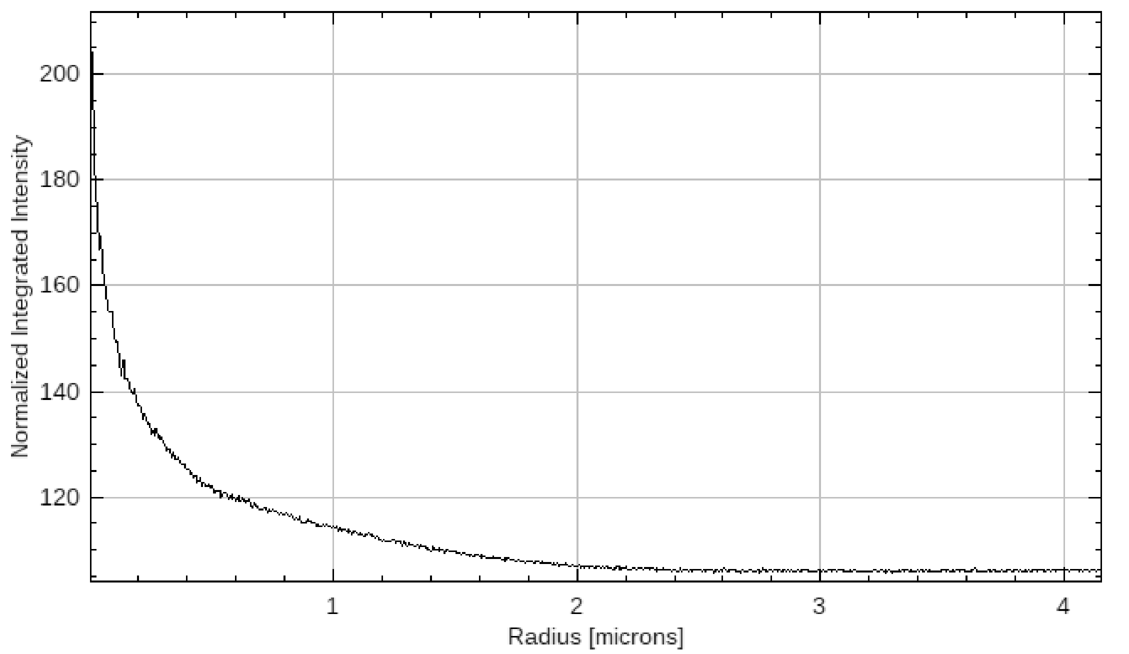
Radial profile plot:
https://imagej.net/ij/plugins/radial-profile.html

Fourier transformation: Resolution

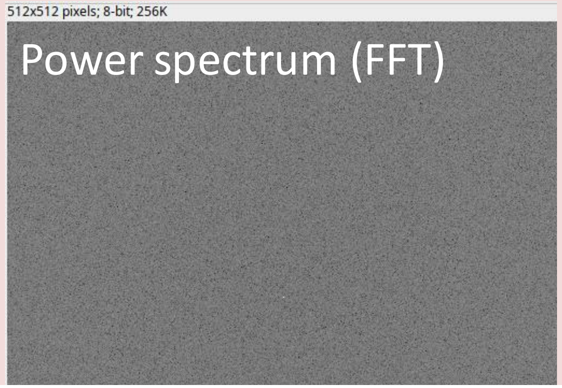
Example 1D – non-native (from 1st lecture)

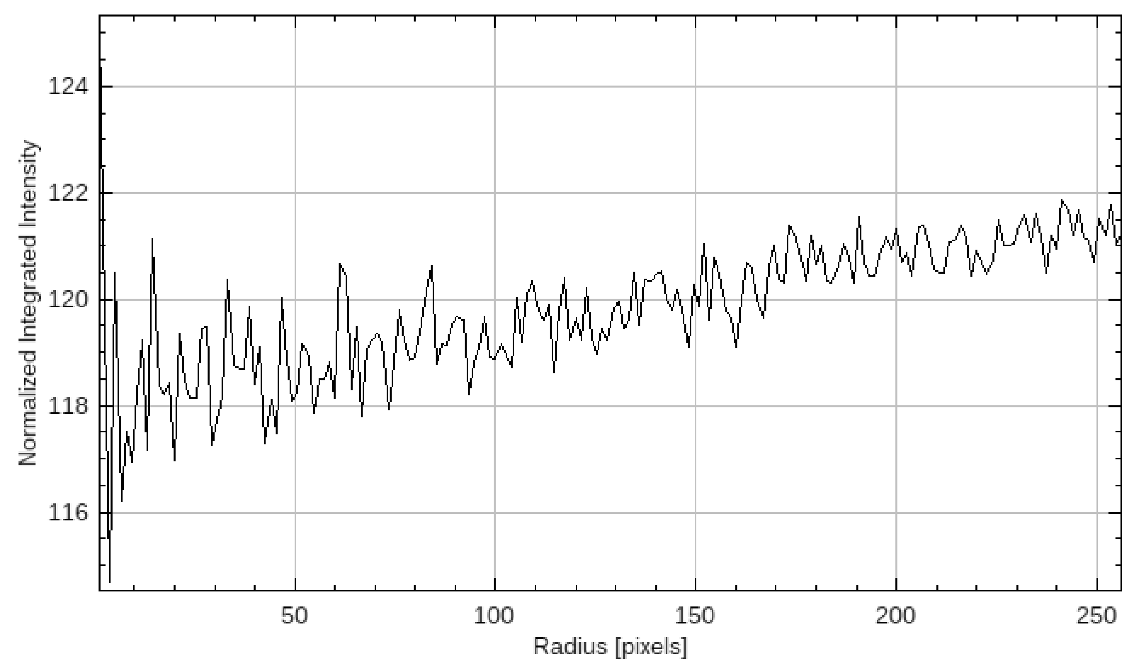






Example 3B – white noise





Fourier transformation: Resolution

Radial profile plot:
https://imagej.net/ij/plugins/radial-profile.html

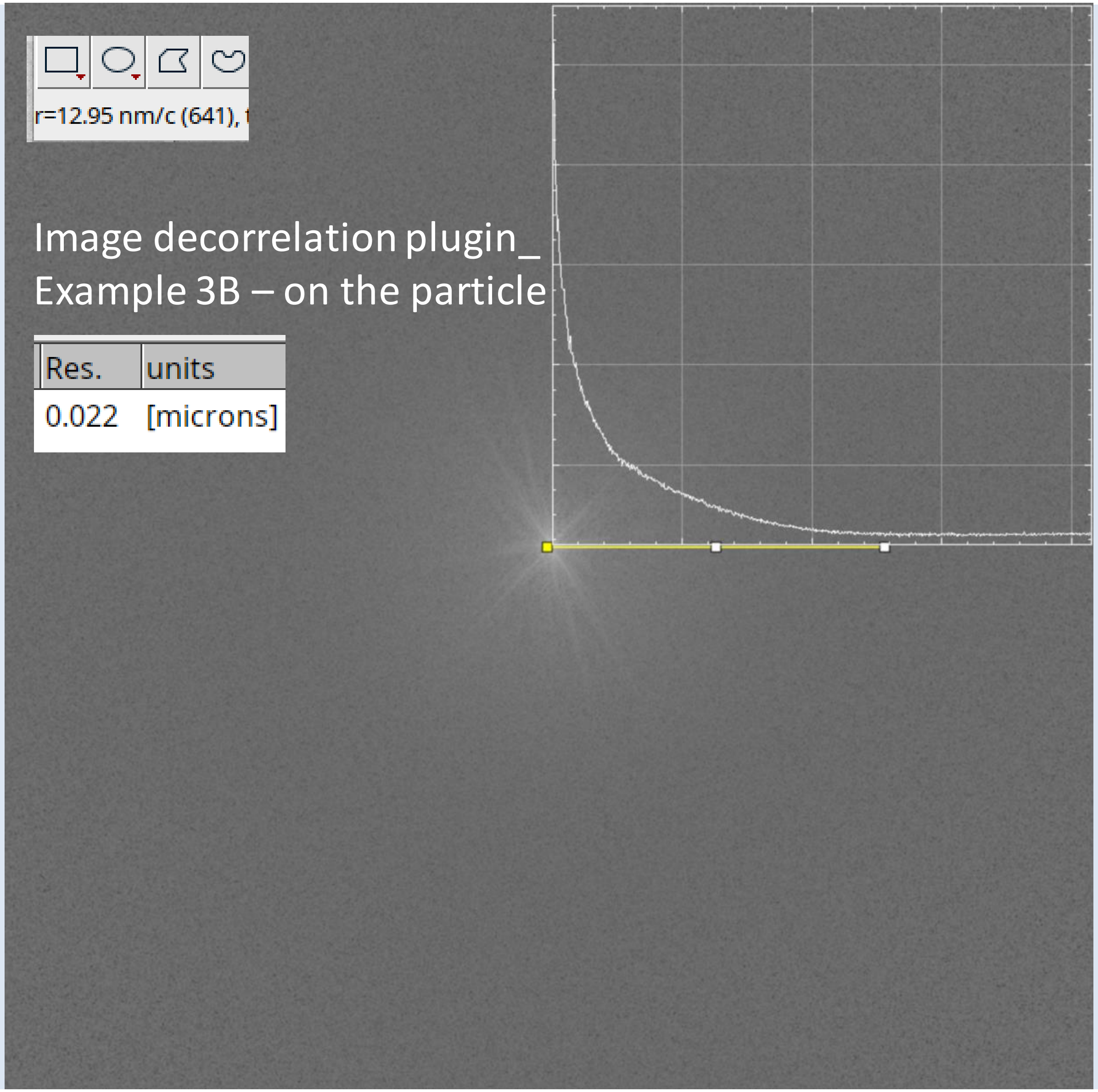
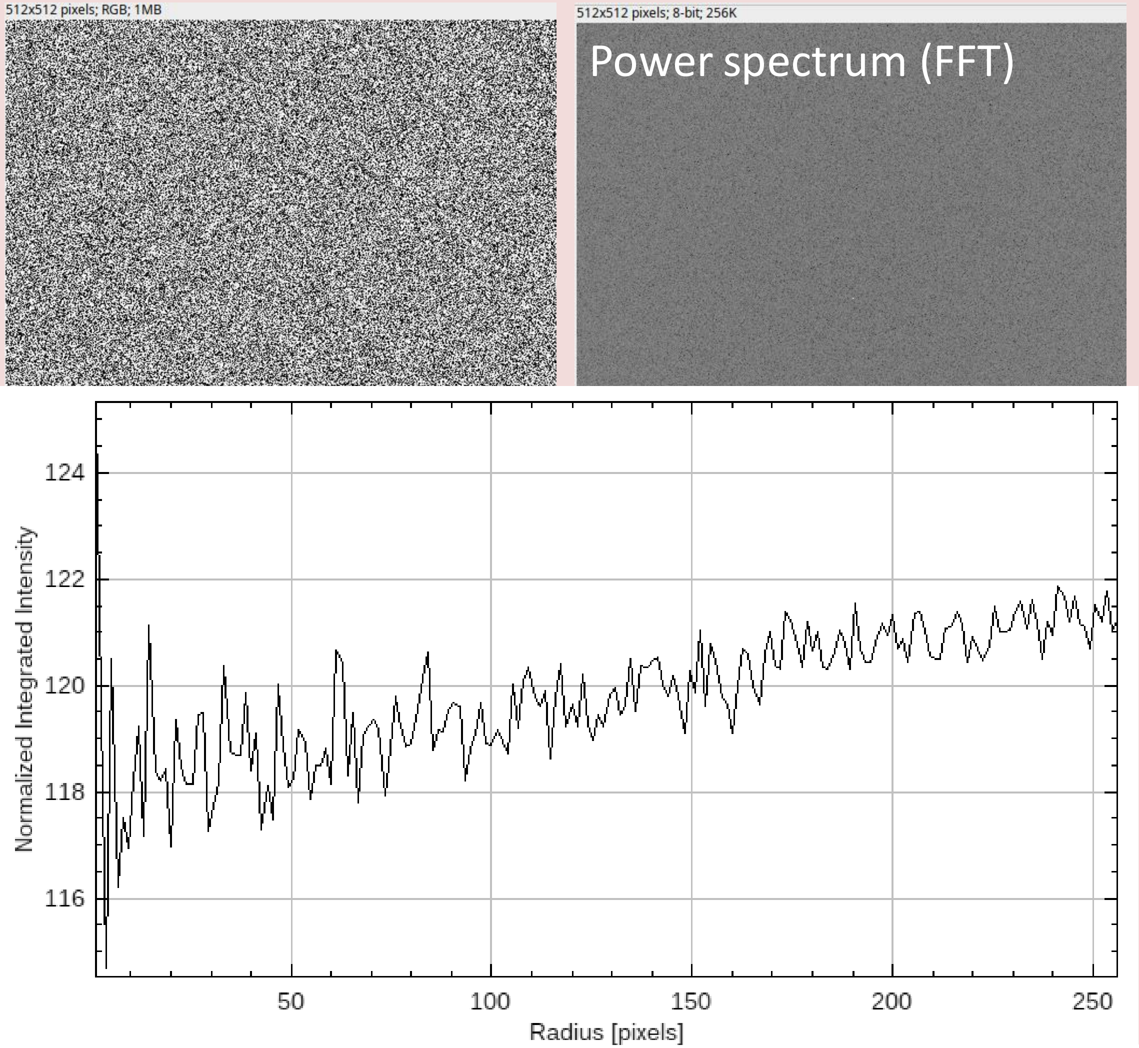


Image decorrelation plugin_
Example 3B – on the particle

Res.	units
0.022	[microns]

Example 3B – white noise



Fourier transformation: deconvolution in Fourier space

A convolution of the light source with hands

Convolution, deconvolution are DIFFICULT in real space but are simple multiplications and division in Fourier space

Can you remove the motion blur?

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: filtering in Fourier space

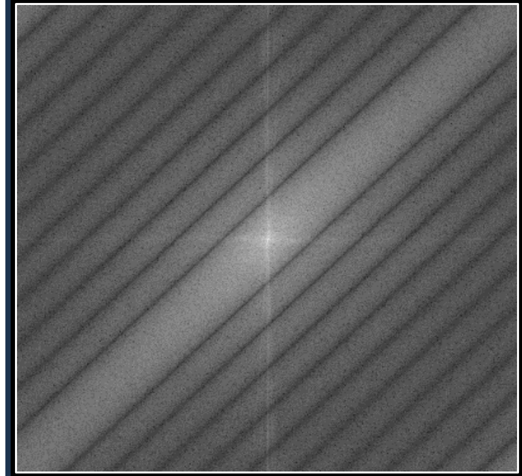
Sampling in the *temporal dimension* was not a point but a line: convolution (i.e. the camera moved....)

Convolution, deconvolution are DIFFICULT in real space but are simple multiplications and division in Fourier space

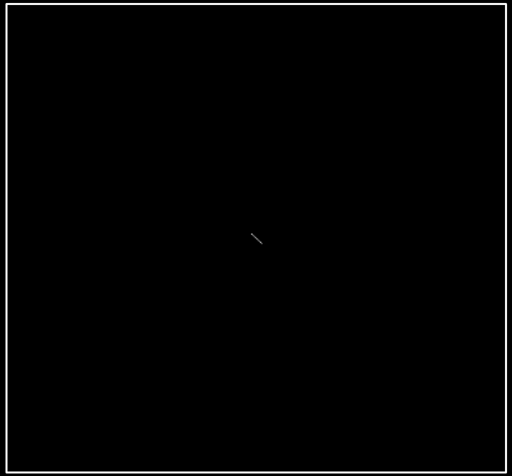
Real Space

$y(u, v)$

Fourier Space



FFT of the image



"impulse response of a linear time-invariant system"

$h(u, v)$

$$y(u, v) = (h * x)(u, v)$$

$y(u, v)$	Observed image
$x(u, v)$	Ground-truth image
$h(u, v)$	blurring vector

* denotes convolution

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: deconvolution

EXERCISE 4

Open Example 4 – Motion blurred and try to remove the motion blur

Can you remove the motion blur?

1. Open Example 4 – motion blurred, the motion blurred image.
2. Also open the point spread function of example 4.
3. Do the deconvolution: Process > FFT > FD math.
4. Image1 is the motion blurred image, Image2 is the Point spread function. Use **deconvolve** and check «Do inverse transform»

Deconvolution algorithms,
which allow to improve the resolution of an image, are exactly running these functions.

Fourier transformation: deconvolution

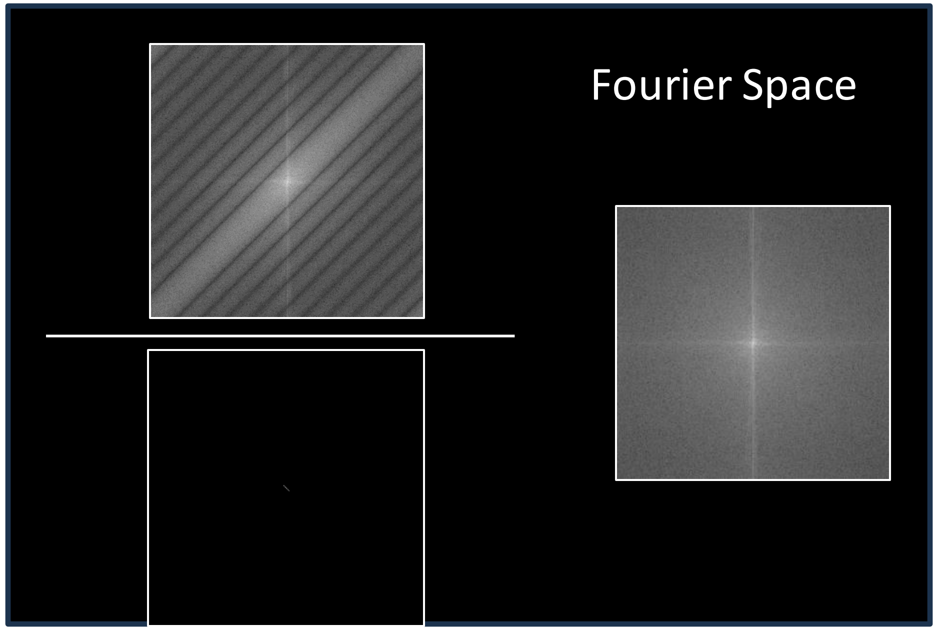
EXERCISE 4

Open Example 4 and try to remove the motion blur

Can you remove the motion blur?

1. Open Example 4, the motion blurred image.
2. Also open the point spread function of example 4.
3. Do the deconvolution: Process > FFT > FD math.
4. Image1 is the motion blurred image, Image2 is the Point spread function. Use deconvolve and check «Do inverse transform» or run the inverse FFT afterwards

$$\frac{y(u,v)}{h(u,v)}$$

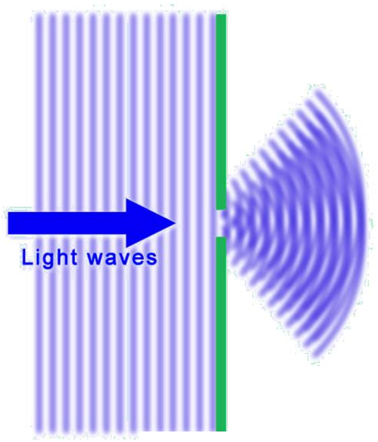


→

Real Space

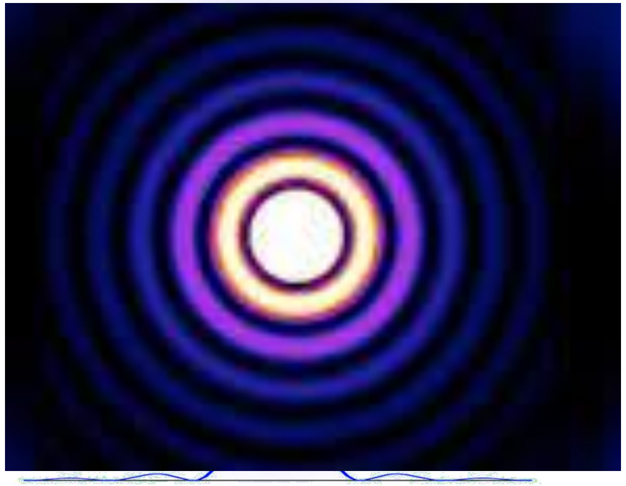
Fourier transformation: deconvolution

Deconvolution algorithms
allow the improvement of the resolution of an image. Deconvolve algorithms try to mimick the PSF (point spread function) produced through diffraction and deconvolute it to improved the image.

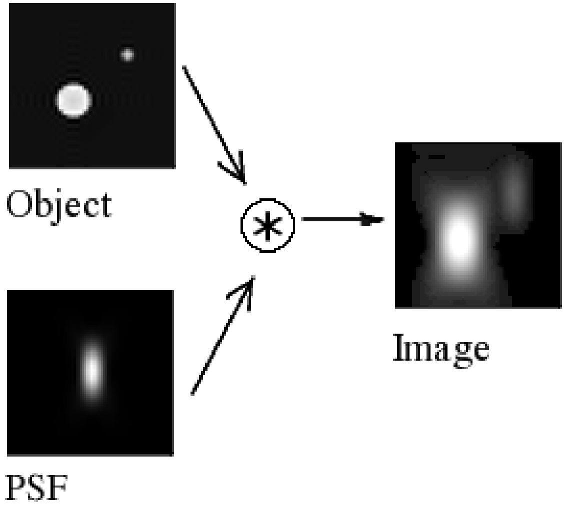


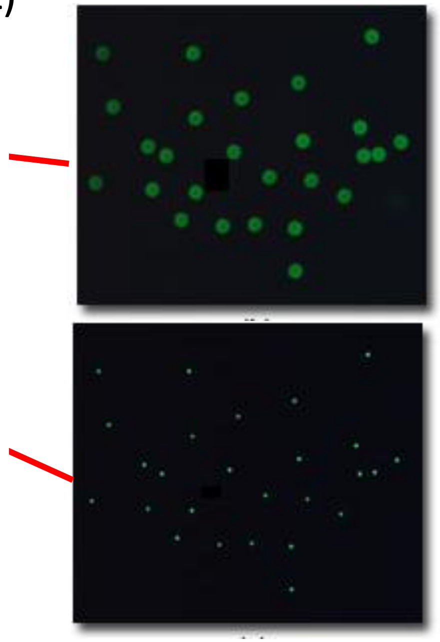
aperture

Point spread function (XY)*


Airy disc

Axial Point spread function (XZ)



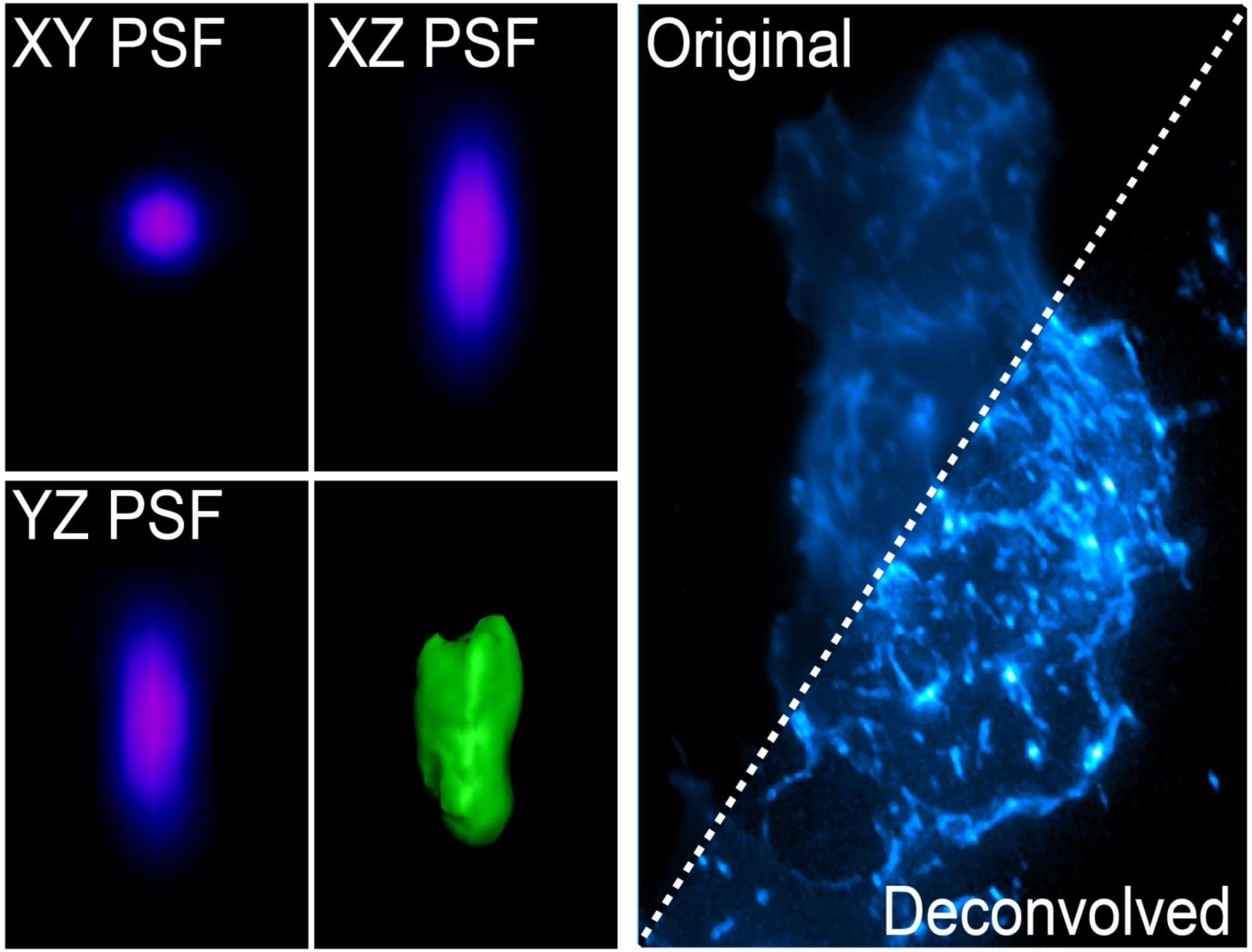


Imaged in defocus

Imaged in focus

* In electron microscopy you may see the contrast transfer function (CTF) or modulation transfer function (MDF)

Fourier transformation: deconvolution



$$y(u,v) = (h * x)(u,v) + n(u,v)$$

$y(u,v)$	Observed image
$x(u,v)$	Ground-truth image
$h(u,v)$	PSF, OTF, CTF, blurring vector…
$n(u,v)$	Unknown additive noise, independent of x(u,v)

* denotes convolution

GOAL: find g(u,v) so that:

$$\hat{x}(u,v) = (g * y)(u,v)$$

$\hat{x}(u,v)$	The estimate of $x(u,v)$ with a minimized cost function

$$\epsilon(u,v) = \mathbb{E}|x(u,v) - \hat{x}(u,v)|^2$$

$\epsilon(u,v)$	Cost function (Mean square error)
$\mathbb{E}$	Expectation

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: Wiener filter

$$y(u,v) = (h * x)(u,v) + n(u,v)$$

$$\hat{x}(u,v) = (g * y)(u,v)$$

$$G(a,b) = \frac{H^*(a,b)S(a,b)}{|H(a,b)|^2 S(a,b) + N(a,b)}$$

$G(a,b)$ Fourier transform of g(u,v)

$H(a,b)$ Fourier transform of h(u,v)

$S(a,b)$ = $\mathbb{E}|x(u,v)|^2$ the mean power spectral density of the original image x(u,v)

$N(a,b)$ = $\mathbb{E}|V(u,v)|^2$ the mean power spectral density of the noise t(u,v)

$$\hat{X}(a,b) = G(a,b)Y(a,b)$$

Rewritting this a bit:

$$G(a,b) = \frac{1}{H(a,b)}\left[\frac{1}{1 + \frac{1}{|H(a,b)|^2 SNR(a,b)}}\right]$$

$SNR(a,b) = \frac{S(a,b)}{N(a,b)}$ = signal to noise ratio

Zero noise ➔ $SNR(a,b) = \infty$ ➔ $[\quad] = 1$ = simple inverted system

noise not zero ➔ $SNR(a,b)$ drops ➔ $[\quad] > 1$ = frequencies are attenuated locally

Fourier transformation: Cross correlation (pattern matching)

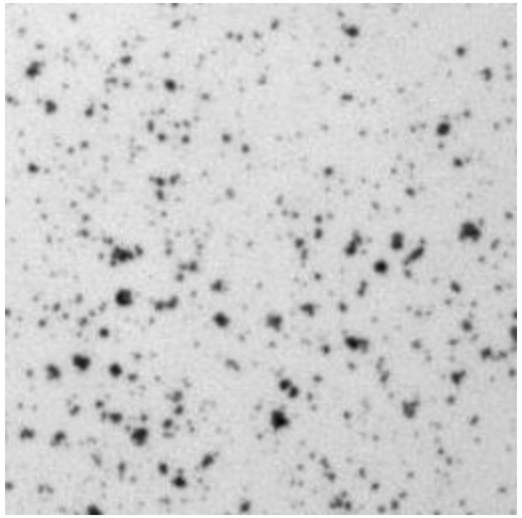


Image A

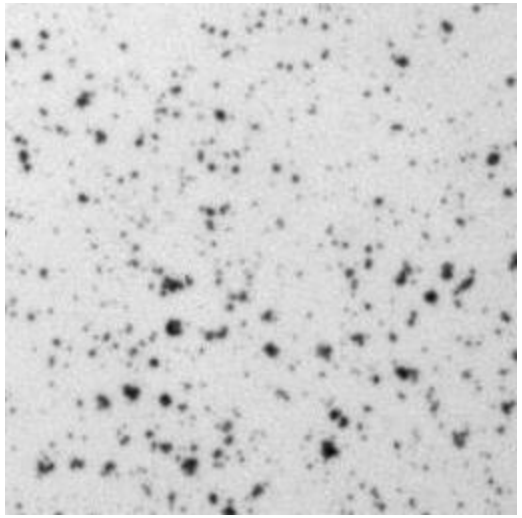
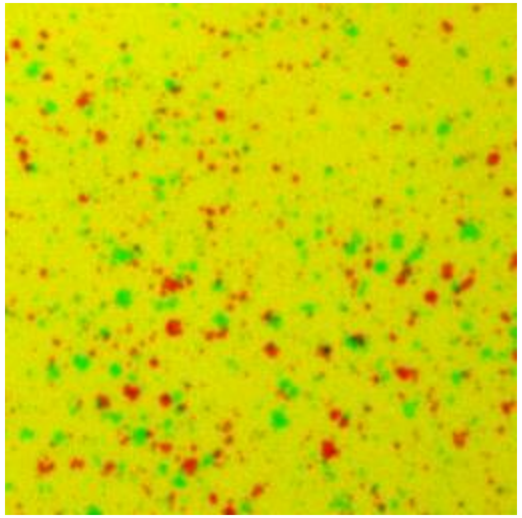


Image B



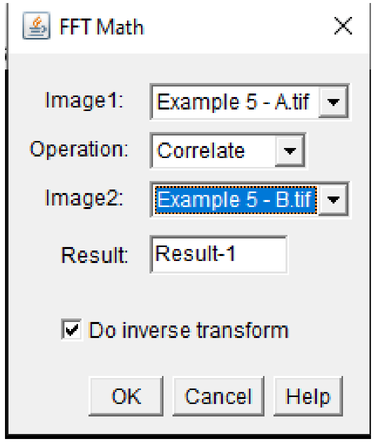
Overlayed and false color coded

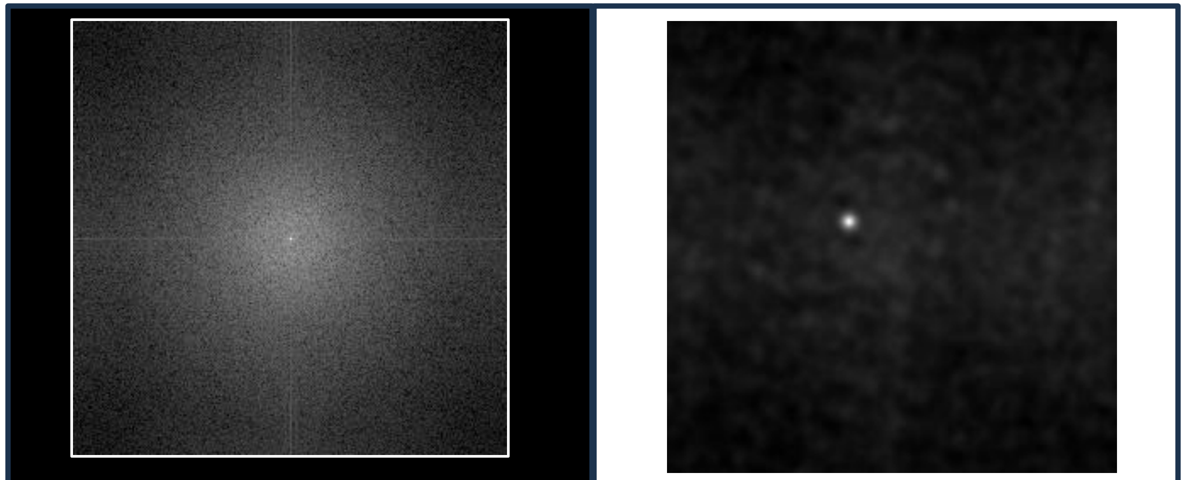
Used in:
Confocal LSM (**tiling**, …)
SEM / EDX (**drift** correction)
TEM (eucentric heigh alignment)
FIB-SEM (**tracking**)
…

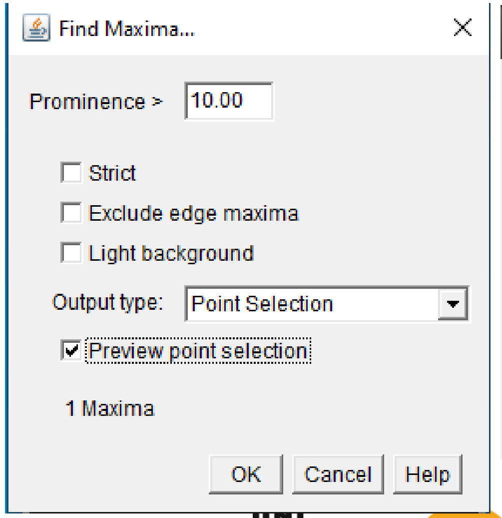
Fourier transformation: Cross correlation (advanced!)

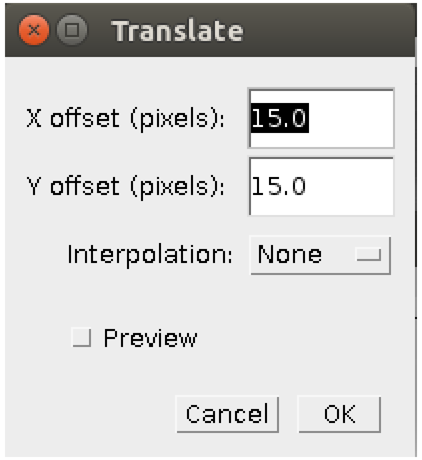
EXERCISE
Open Example 5 (both images) and try to align them

1. Process > FFT > FD math...
2. Find the position of the main peak:
    a. Process > math > Log
    b. Process > Find maxima).
    c. Analyze > Measure
3. Translate Example 5B




Fourier Space
Real space

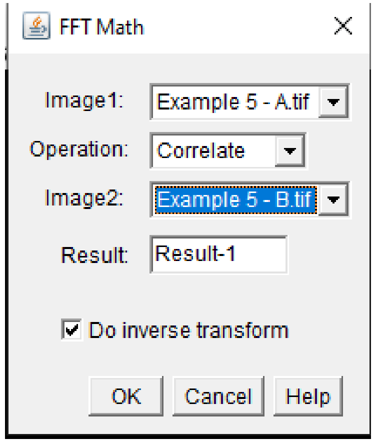




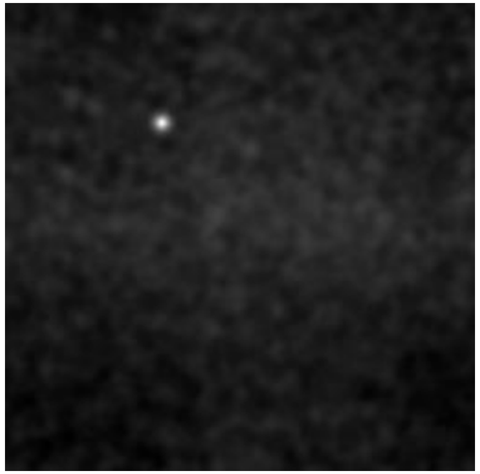
Fourier transformation: Cross correlation

EXERCISE

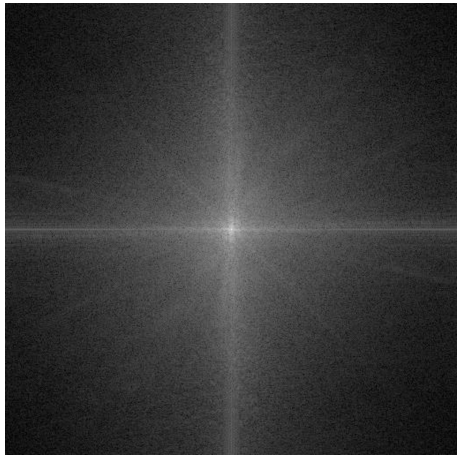
Open Example 5 and try to align the two images



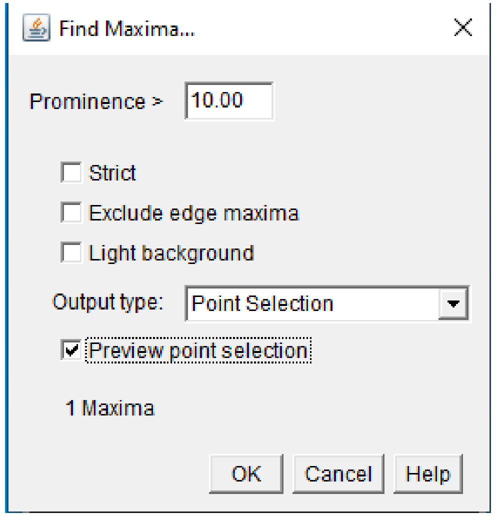
1. Make a cross correlation between the two images (Process > FFT > FD math...).



3. The result shows the cross correlation.



2. If you did not check 'Do inverse transform', do an inverse FFT



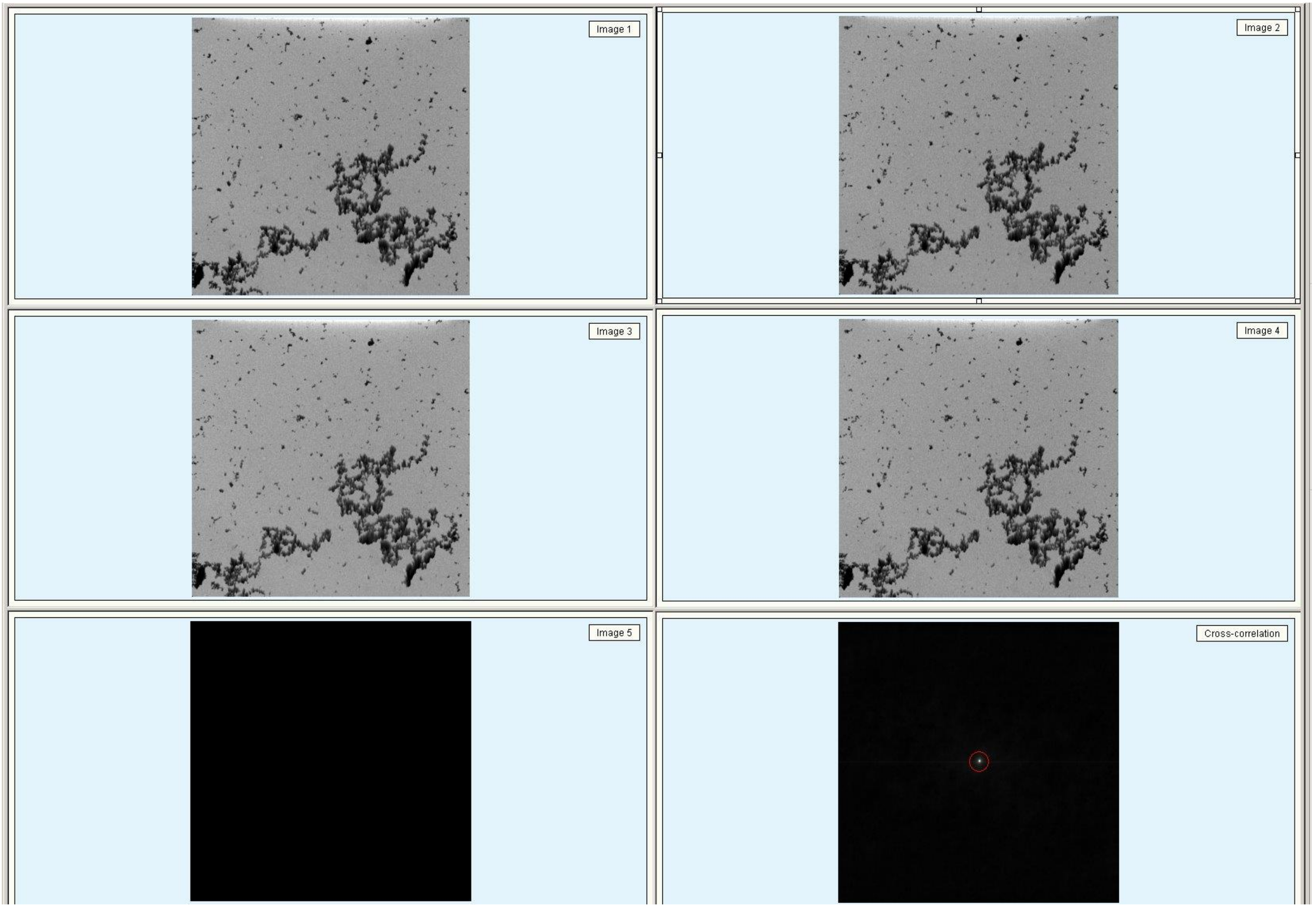
4. Find the position of the Main peak:

- Stretch the contrast (Process > math > Log). Update the B&C
- Find the peak (Process > Find maxima).
- Preview the point selection
- If needed, adjust Noise tolerance until you have 1 maximum

Fourier transformation: Cross correlation

EXERCISE

Open Example 5 and try to align the two images



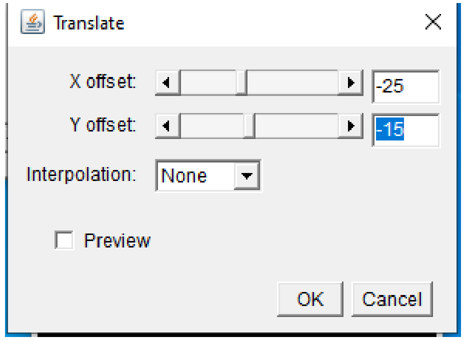
Fourier transformation: Cross correlation

EXERCISE

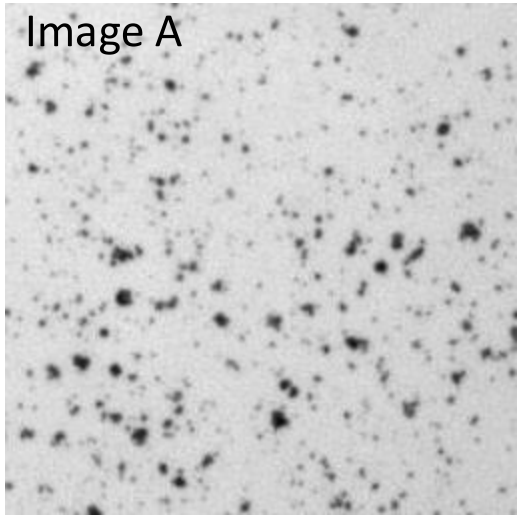
Open Example 5 and try to align the two images

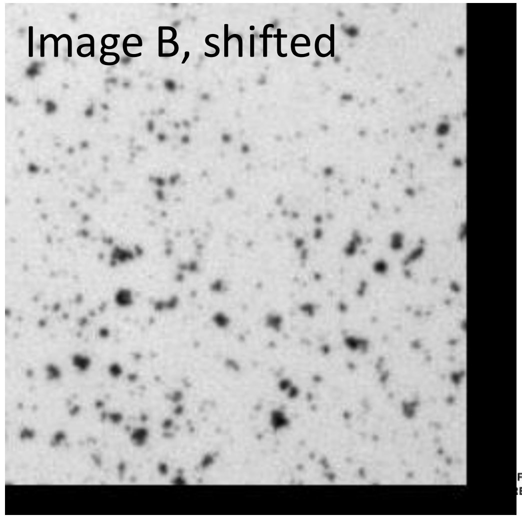
Min	Max	X	Y
21.776	21.776	103	113

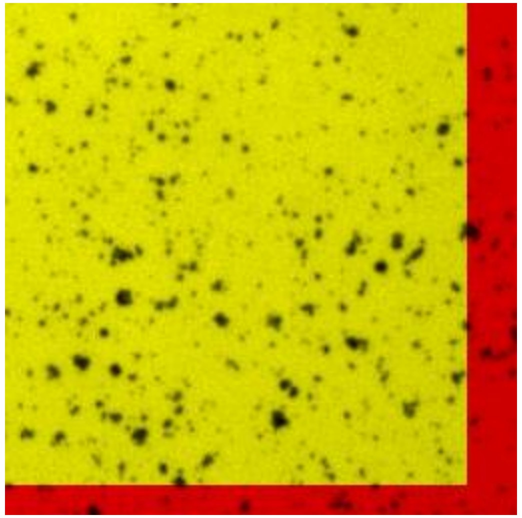
5. Measure the position of that point (Analyze > Measure)
X = 103
Y = 113
These is the translational distance seen from the center of the image (128,128) (why?)



6. Translate Example 5B
- Find the X and Y position in the Results tab, subtract 128:
- (-25, -15) (why?)
- Translate Example 5B over the found shift (Image > transform > translate…)





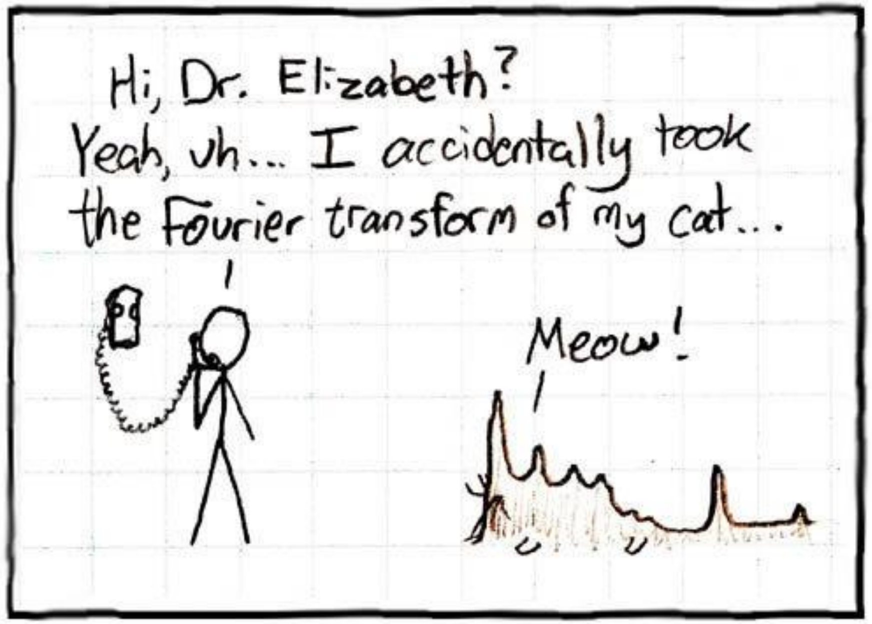


BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Fourier transformation: Summary

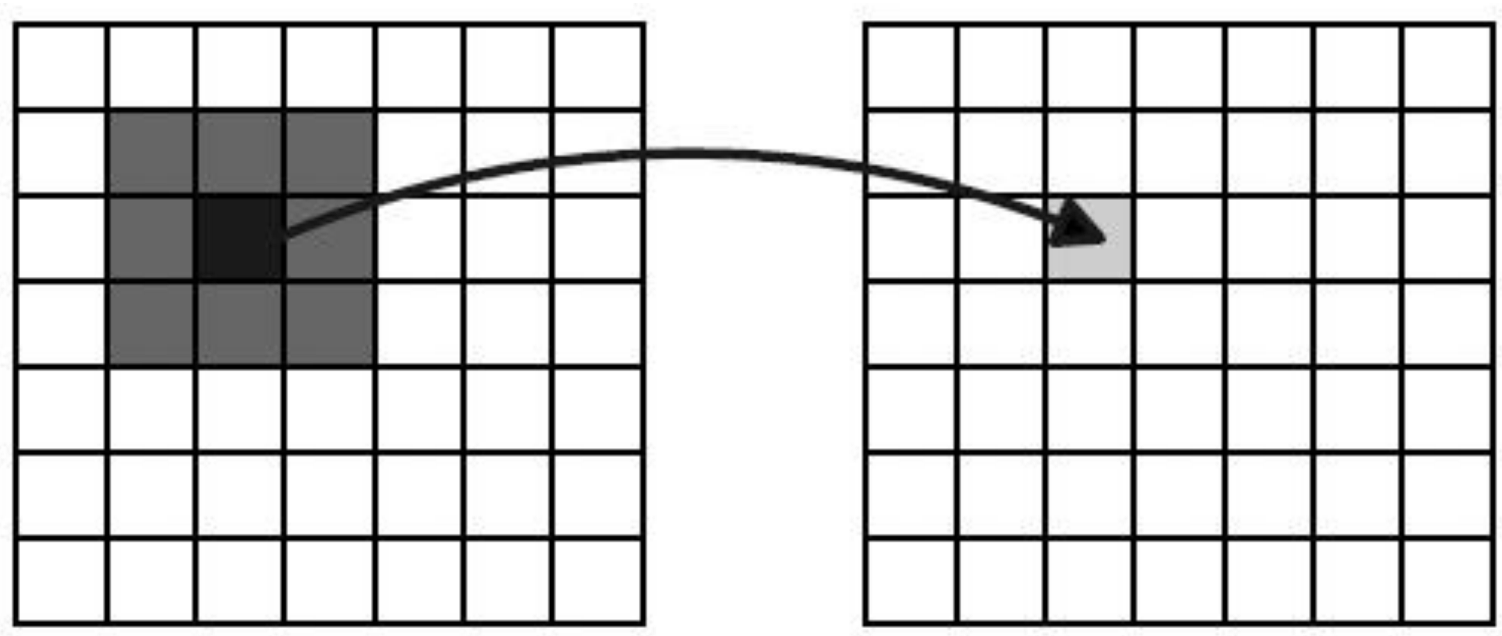
Functions in reciprocal space

In reciprocal space: convolutions become simple multiplications, deconvolutions simple divisions.



Part IV: Filters

Spatial filters



Use **surrounding pixels** to compute each new pixel intensity.

Spatial filters

Local filters

Surrounding pixel info is used: **kernel**

1x1 kernel (=point operation) $[1]$ $[2]$

3x3 kernel (=filter)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

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

Linear filters

Smoothing filters
Gaussian filters
Gradient filters
Laplacian filters

Non-linear filters

Median filter
Variance filter
Minimum filter
Maximum filter

Non-local filters

Find information similar to the current pixel, anywhere in the image. Replace it by the mean, median, … of those non-local values

Examples:
Non local means
Bilateral filter
Anisotropic diffusion

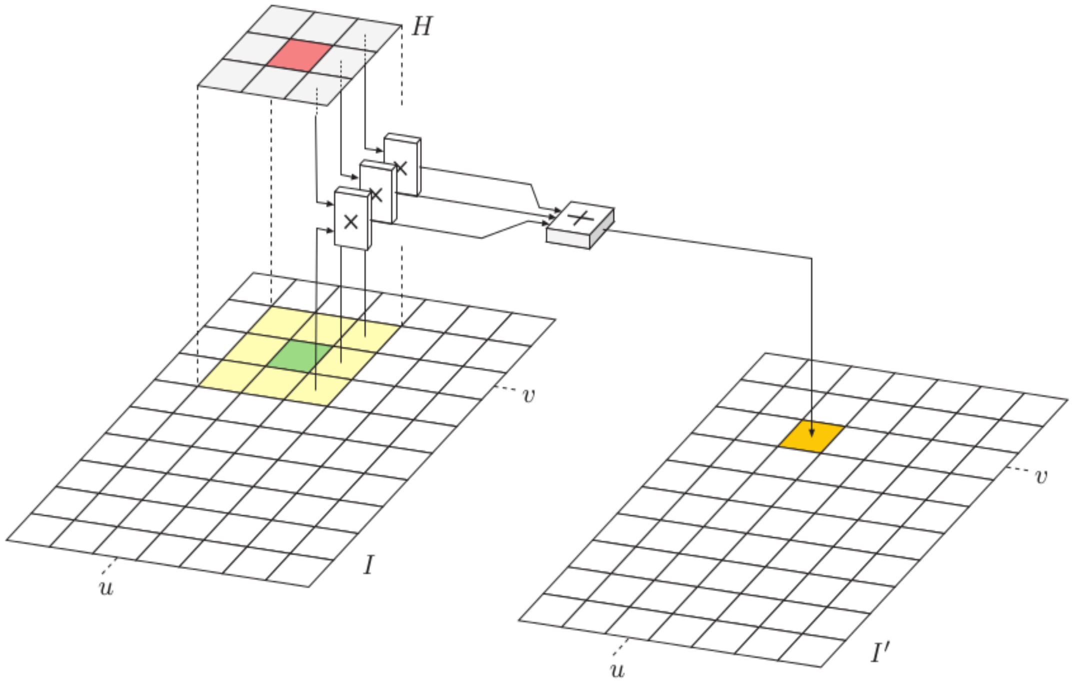
Linear filters: Box filter (or mean filter)

Basic concept:

```
for a(u,v)
    for x
       Array=(u+/-x,v+/-x))
       a'(u,v) = f(Array)
    next
next
```

u= image width, v=image height, x= kernel size

```
For each pixel in the image
   for the size of the kernel
       Put the pixel & all the surrounding pixels in an array

       perform a function. The result is the new value
       of the initial (central) pixel
   end the kernel
Go to the next pixel
```



UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

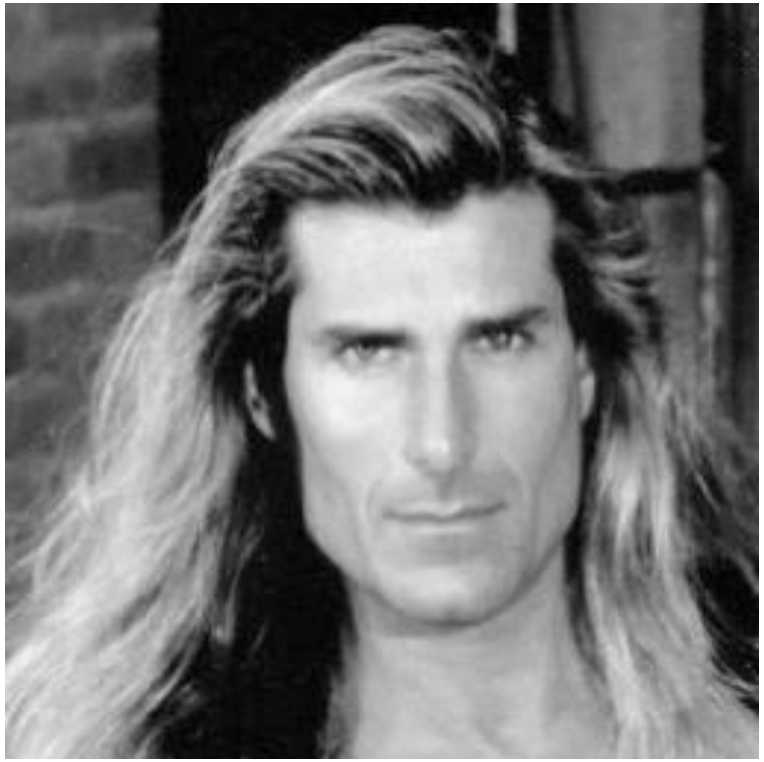
Linear filters: box filter (mean filter)

3x3 smoothing filter

Each new pixel value is the average of the pixel and its surrounding pixels (eg: a 3x3 filter is 1 central pixel and 8 surrounding pixels)

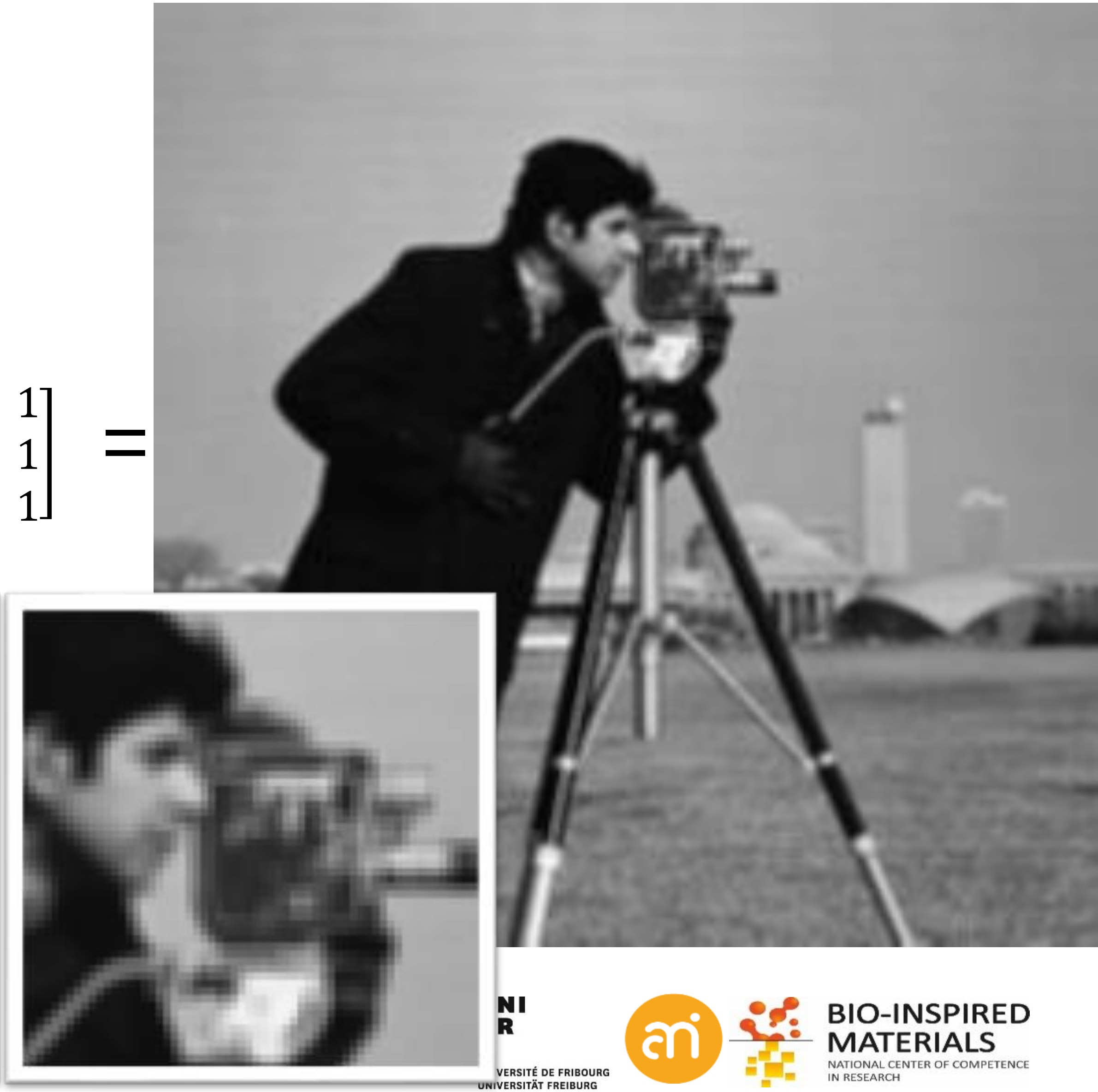
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$I = \frac{\sum \begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{01} & a_{11} & a_{12} \\ a_{02} & a_{12} & a_{22} \end{matrix}}{n}$$



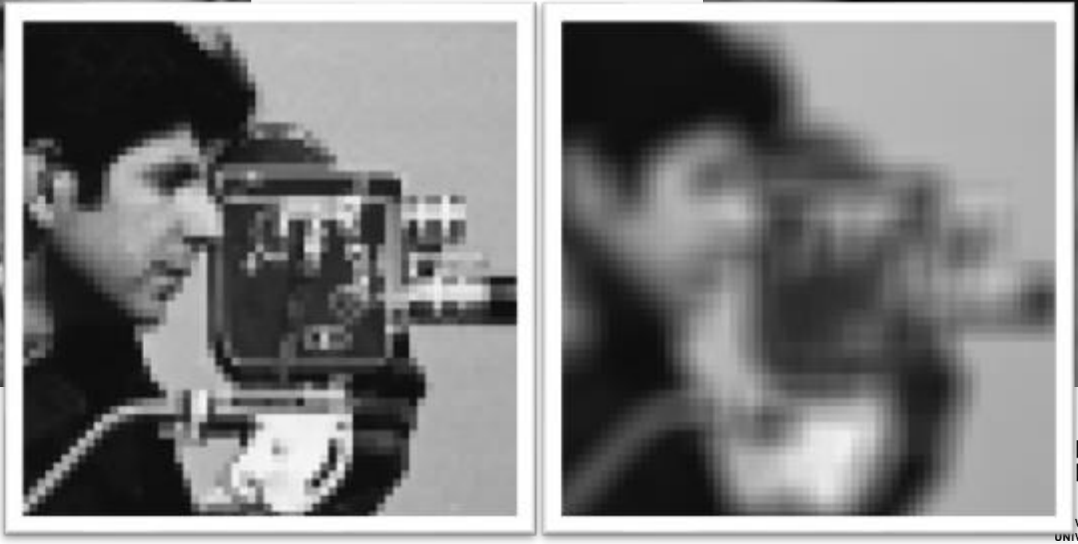
Linear filters: box filter (mean filter)

$$\otimes \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



Linear filters: box filter (mean filter)

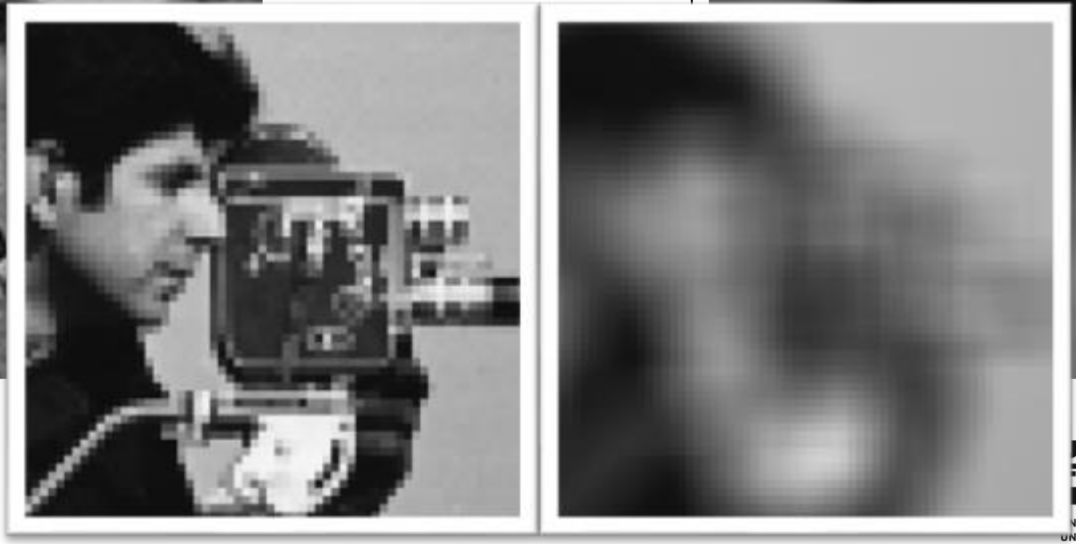
$$\otimes \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$



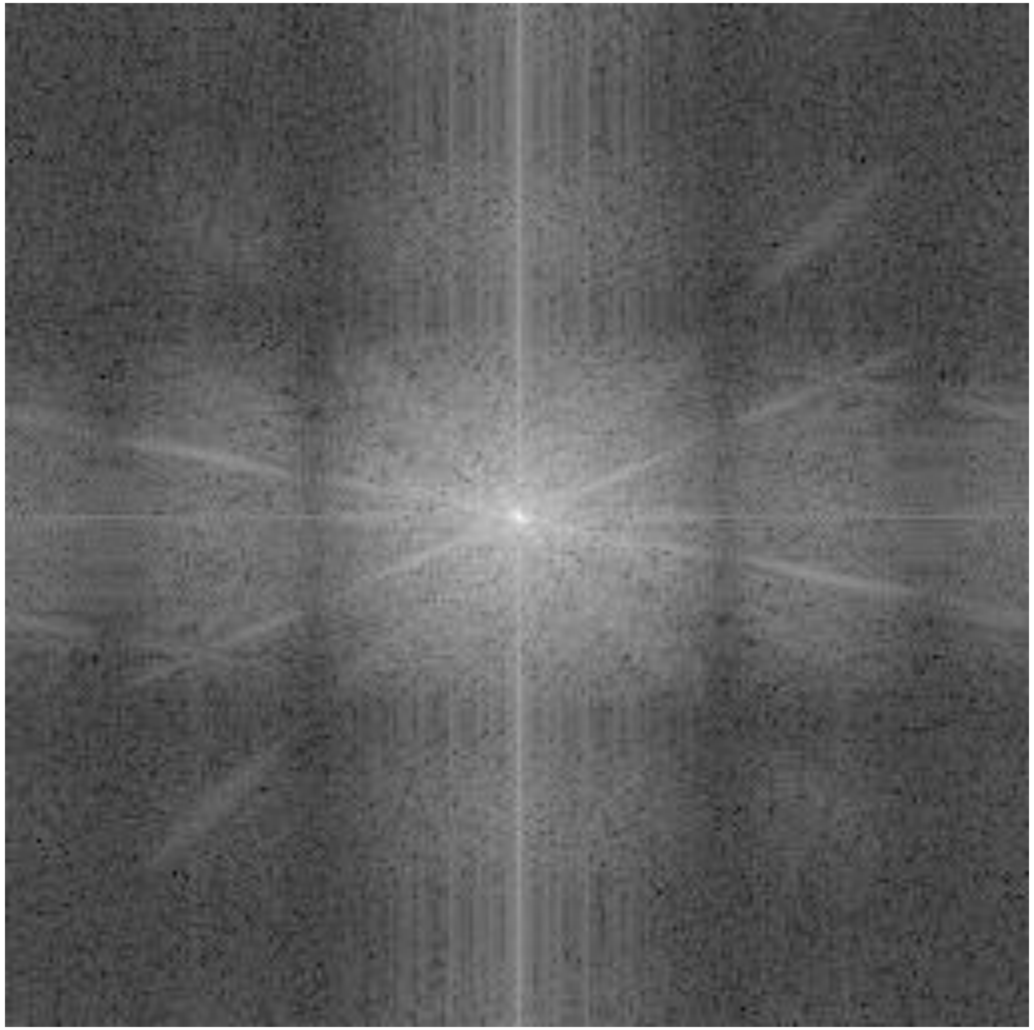
Linear filters: box filter (mean filter)

⊗

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1



Linear filters: box filter (mean filter)



UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

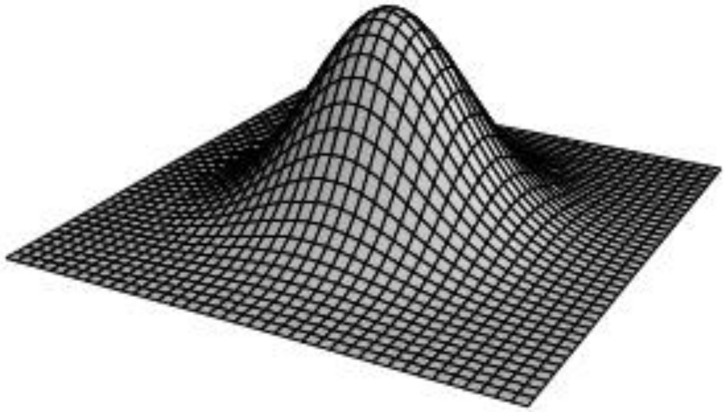
Linear filters: Gaussian

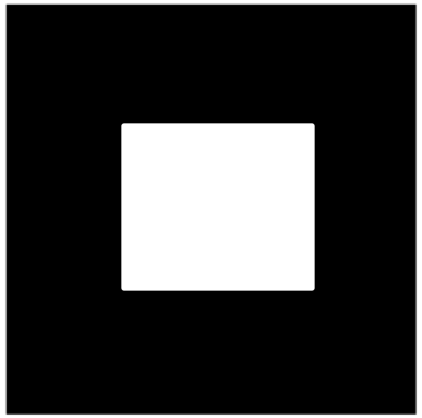
$$\frac{1}{(4\pi h^2)} e^{-\frac{|\mathbf{x}|^2}{4h^2}}$$

3x3 gaussian smoothing filter

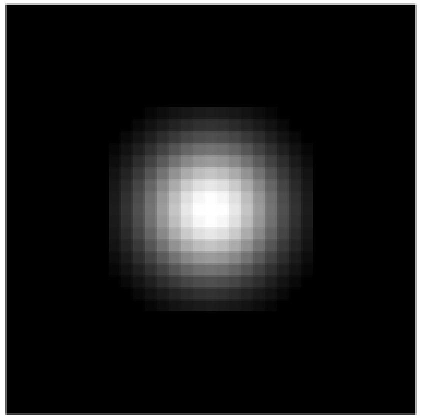
Each new pixel value is the **weighted** average of the pixel and its surrounding pixels (a 3 x 3 filter is 1 central pixel and 8 surrounding pixels or radius=1)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$





box window



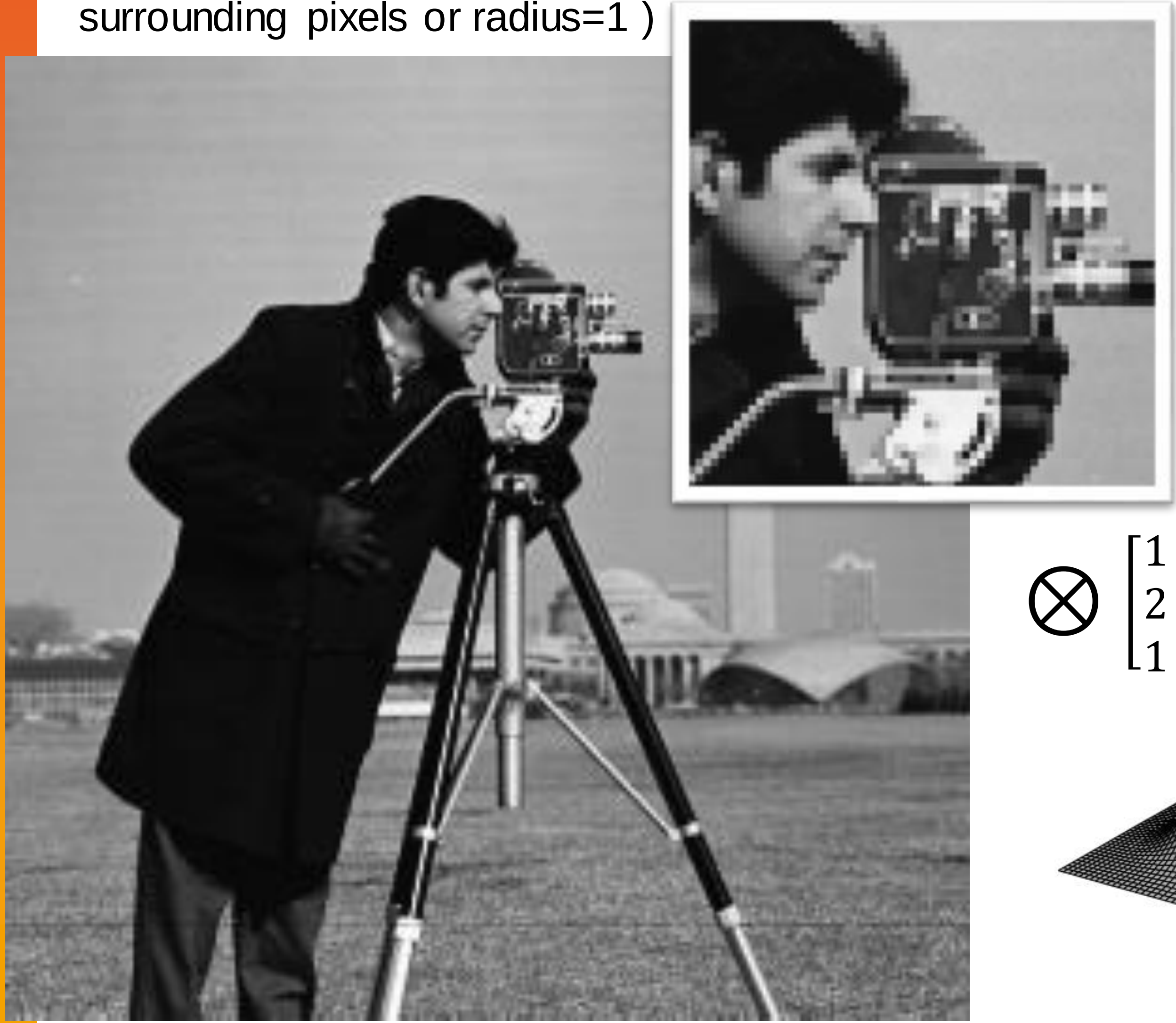
Gaussian window

Linear filters: Gaussian

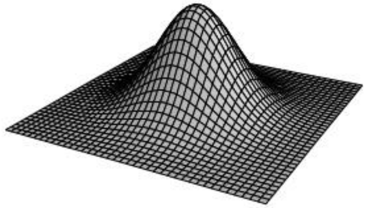
$$\frac{1}{(4\pi h^2)} e^{-\frac{|\mathbf{x}|^2}{4h^2}}$$

3x3 gaussian smoothing filter

Each new pixel value is the **weighted** average of the pixel and its surrounding pixels (a 3 x 3 filter is 1 central pixel and 8 surrounding pixels or radius=1)



$$\otimes \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} =$$



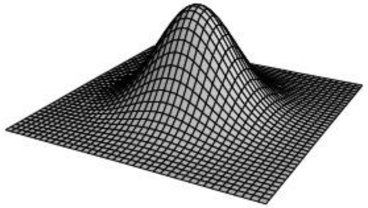
Linear filters: Gaussian

$$\frac{1}{(4\pi h^2)} e^{-\frac{|\mathbf{x}|^2}{4h^2}}$$

5x5 gaussian smoothing filter

Each new pixel value is the **weighted** average of the pixel and its surrounding pixels (a 5 x 5 filter is 1 central pixel and 24 surrounding pixels or radius=2)

$$\otimes \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} =$$

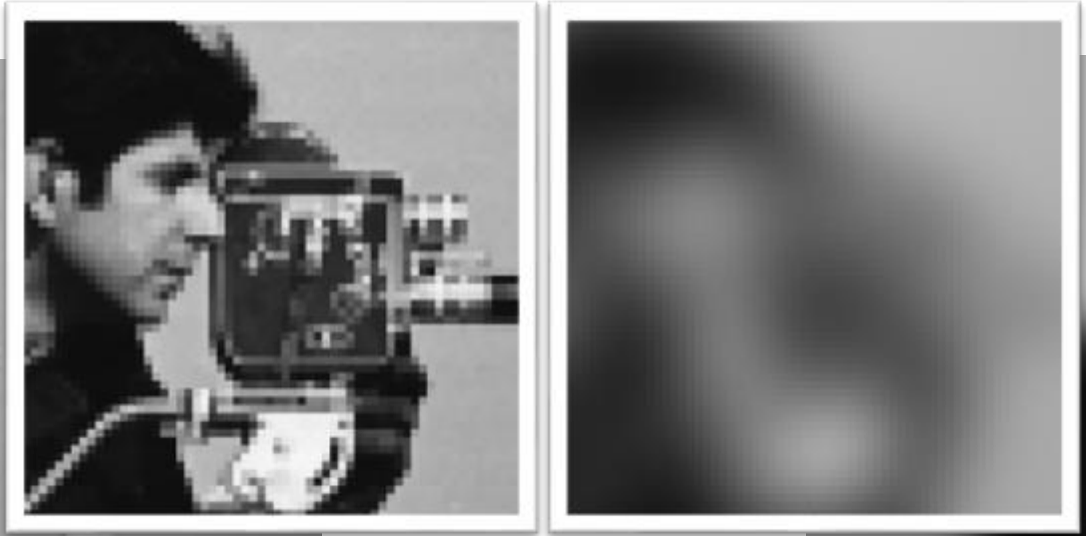


Linear filters: Gaussian

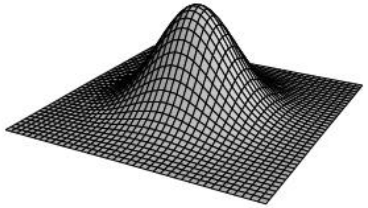
$$\frac{1}{(4\pi h^2)} e^{-\frac{|\mathbf{x}|^2}{4h^2}}$$

11x11 gaussian smoothing filter

Each new pixel value is the **weighted** average of the pixel and its surrounding pixels (a 11 x 11 filter is 1 central pixel and 120 surrounding pixels, radius=5)



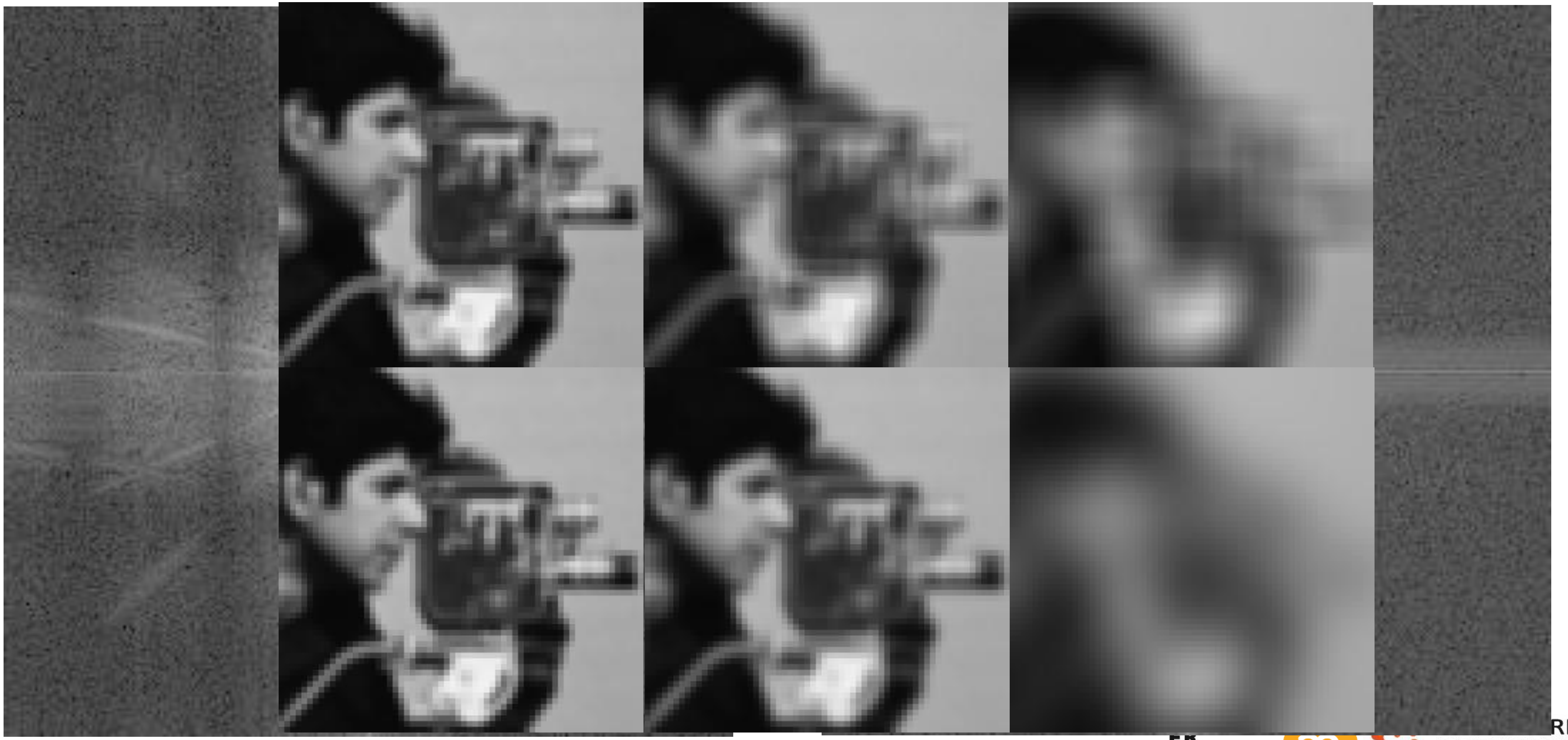
$$\otimes \begin{bmatrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1\end{bmatrix} =$$



Linear filters: Gaussian

$$\frac{1}{(4\pi h^2)} e^{-\frac{|\mathbf{x}|^2}{4h^2}}$$

Box filter, 11x11 Gaussian filter, 11x11



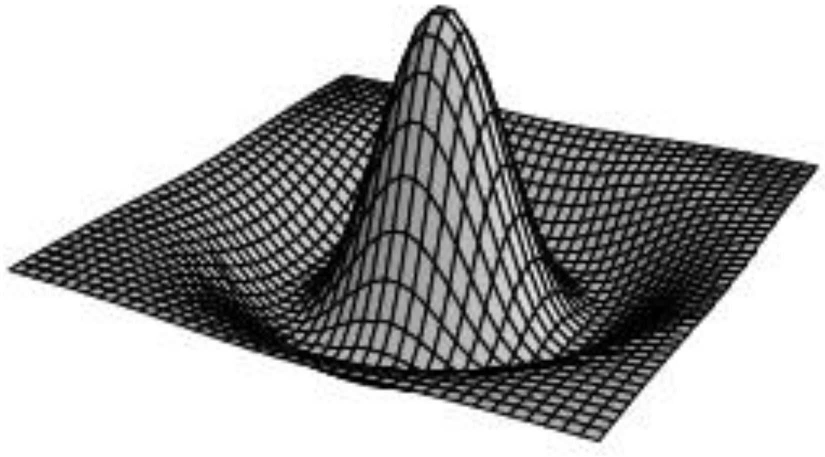
Linear filters: Mexican hat (difference)

3x3 difference filter

Coefficients of the matrix (not the central value) are < 0

➔ Differences with the central pixel are accentuated

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

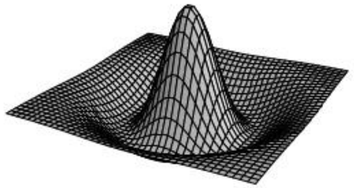


Linear filters: Mexican hat (difference)

3x3 difference filter

Coefficients of the matrix (not the central value) are < 0: differences with the central pixel are accentuated: sharpening!

$$\otimes \begin{bmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



Linear filters

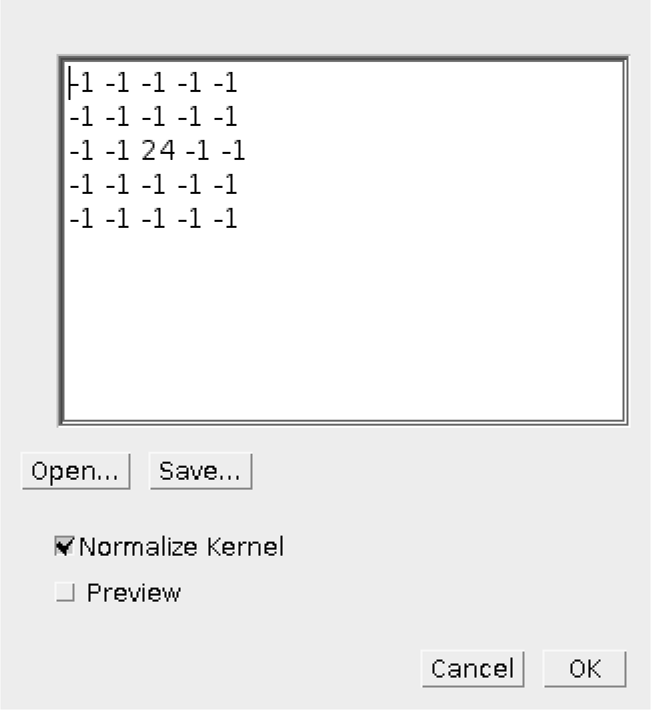
EXERCISE

Open Example 6A (Lena), Example 6B (Fabio) or camera man and try some smoothing and Gaussian filters

Process > Filters > Convolve... To design your own filter or load a premade filter (space between the coefficients)

Use the ‘Normalize kernel’ option ! (why?)

Why would you (willingly) blur your data?

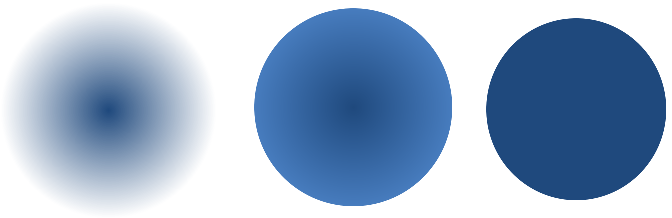


Linear filters: why?

EXERCISE
Why would you willingly blur your image? Try Example 7 - gradient

Background gradient correction

Linear filters: Image gradient magnitude



How can we express/quantify the strength of the gradient (or the «intensity» of an edge)? What about direction of a gradient?

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

How to calculate a derivative of a discrete function??

(meaning h cannot made smaller than the pixel size…)

10	60	10	200	210	250	250

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

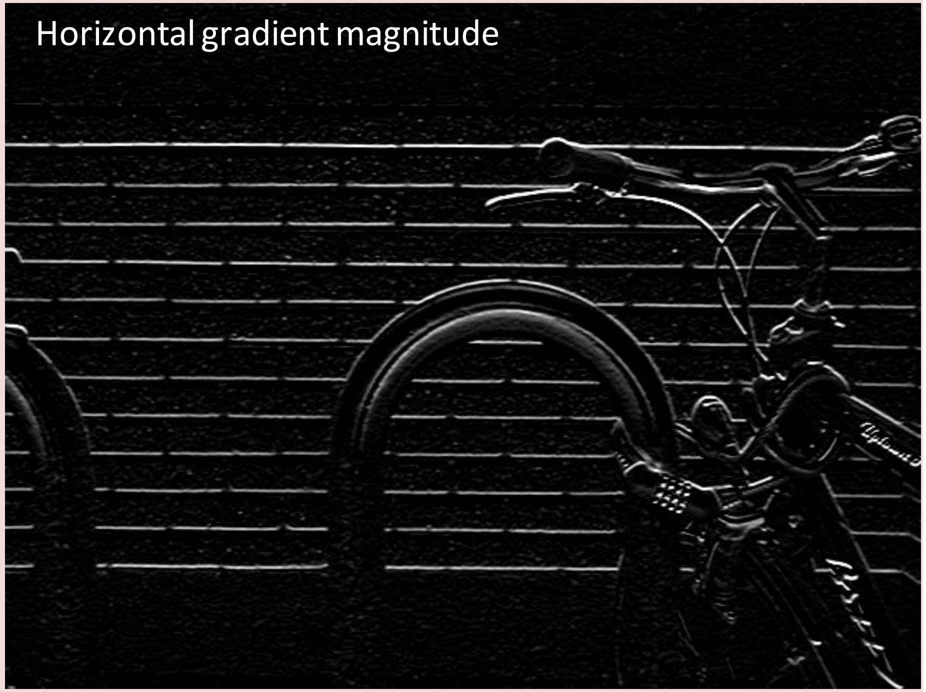
-1	0	1

1D derivative filter

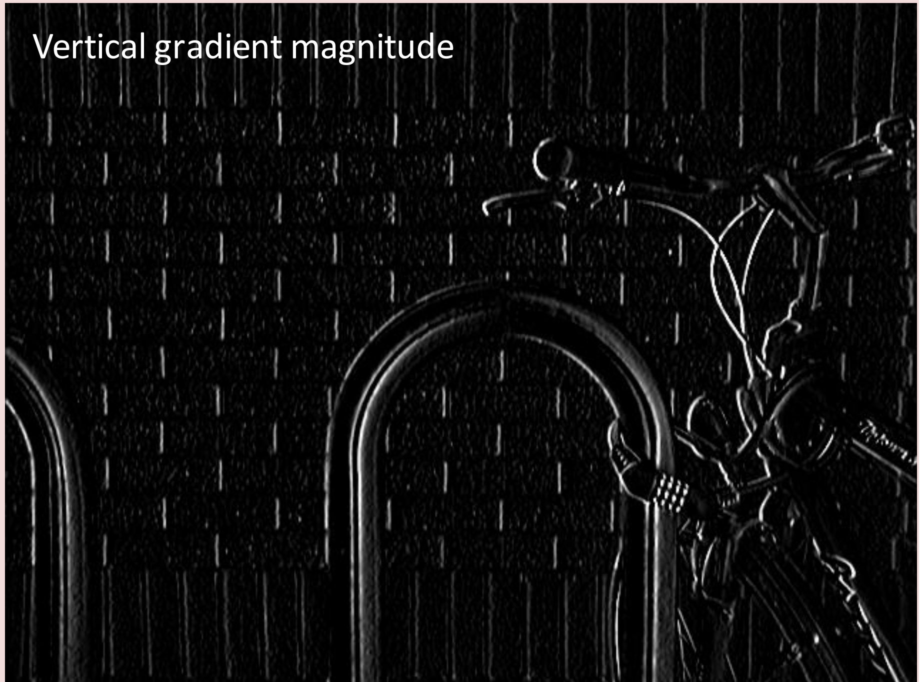
Linear filters: Prewitt gradient filter

Prewitt filter: simplest of derivative (gradient) filters
= rate of (intensity) change
= edge detection

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



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



Judith Martha Prewitt

MathSciNet

Ph.D. Uppsala Universitet 1978

Dissertation: *On some applications of pattern recognition and image processing to cytology, cytogenetics and histology*

Linear filters: Sobel gradient filter

Sobel filter: improved with a weighted average filter

x derivative

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

Weighted average

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

y derivative

$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Weighted average

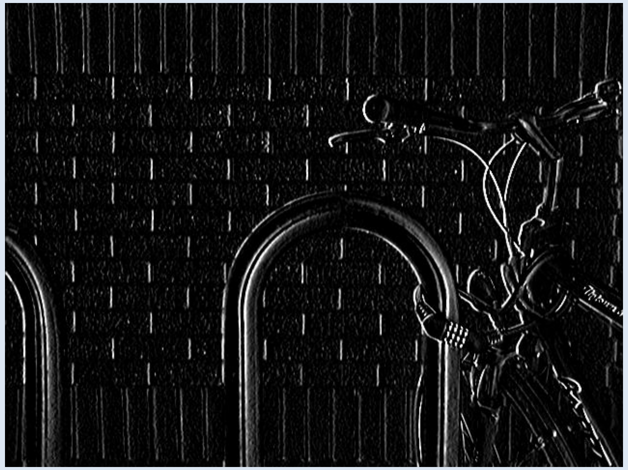
$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Linear filters: Image gradient magnitude

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

Horizontal gradient magnitude



Vertical gradient magnitude

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Gradient magnitude

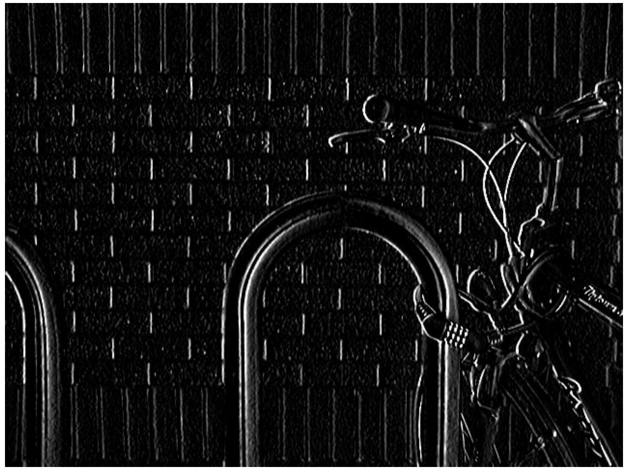
Gradient is encoded in the pixel value. High value = border

Linear filters: Image gradient magnitude

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

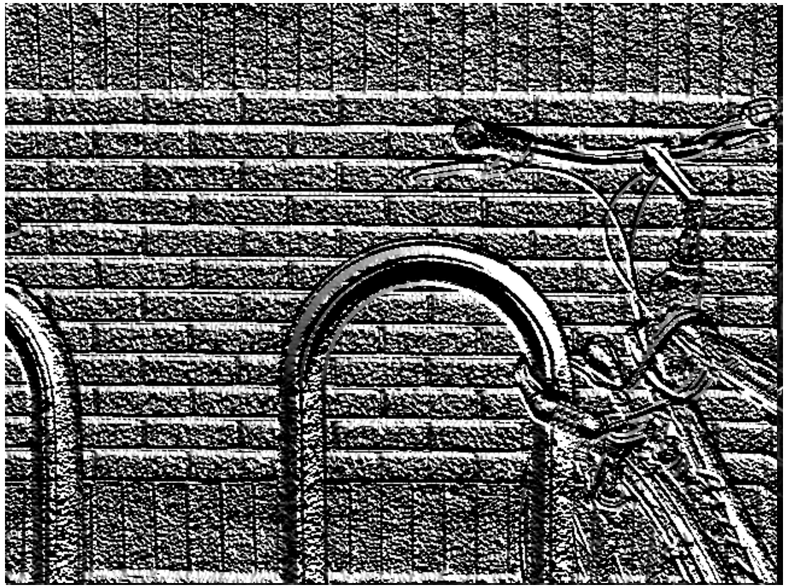
Horizontal gradient magnitude



Vertical gradient magnitude

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Gradient angle



Gradient angle is encoded in the pixel value.

Linear filters: sobel filter

EXERCISE

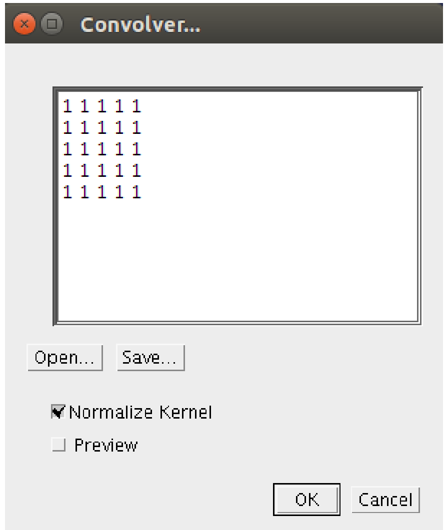
Open Example 8B or (Example 6A/B/C) and perform a Sobel filter

Process > Filters > Convolve... To design your own filter or load a pre-made filter

Linear filters: sobel filter

EXERCISE
Open Example 8 or (Example 6A/B/C) and perform a Sobel filter

1. Duplicate the image (you need an X and a Y)
2. Process > Filters > Convolve... To design your own filter or load a pre-made filter



Make sure «normalize kernel» is switched on (this causes each coefficient to be divided by the sum of the coefficients, preserving image brightness). See the live preview by clicking «preview»

Sobel edge finding filter:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

Gradient magnitude

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

3. Convert each image to 16 bit (Image > mode) – this ensures you will not overilluminate during the next steps
4. Square each of the images (Process > math)
5. Sum them up (with process > image calculator, use ‘add’, and 32-bit, new window)
6. Finally, square root the result (Process > Math)

Linear filters: Laplacian of Gaussian (LoG)

First derivative

10	60	10	200	210	250	250

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

Second derivative

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

1	-2	1

1D Laplace filter

0	1	0
1	-4	1
0	1	0

2D Laplace filter

Laplacian
= the divergence of the gradient of a function in Euclidean space
= second derivative

Linear filters: Laplacian of Gaussian (LoG)

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

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

Is another approximation of the second derivative of a discrete function and therefore also a Laplacian of Gaussian filter (LoG)

Linear filters: Laplacian of Gaussian (LoG) vs Sobel

The LoG is

- Computationally faster
- More precise

Then why using a Sobel filter?

Linear filters: Laplacian of Gaussian (LoG) vs Sobel

The LoG is

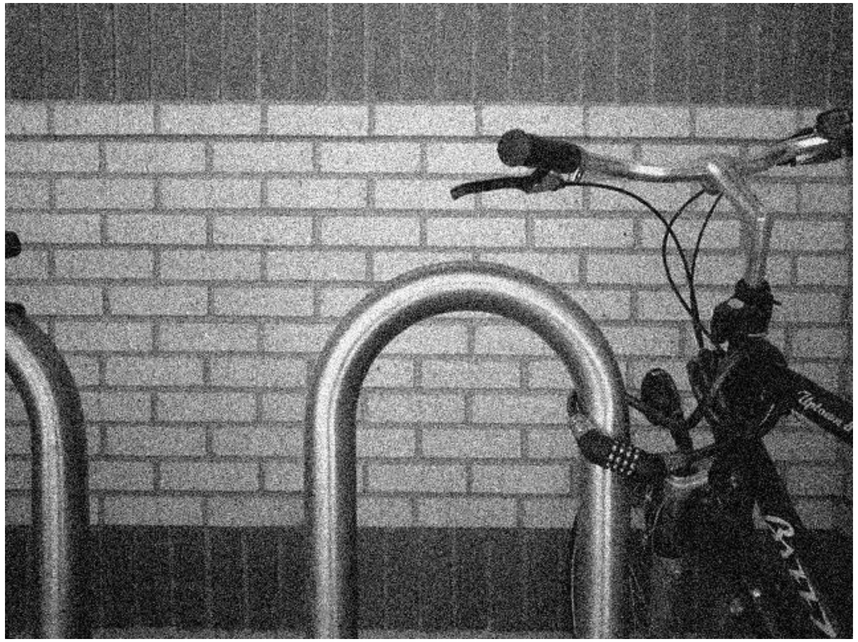
- Computationally faster
- More precise

Then why using a Sobel filter?

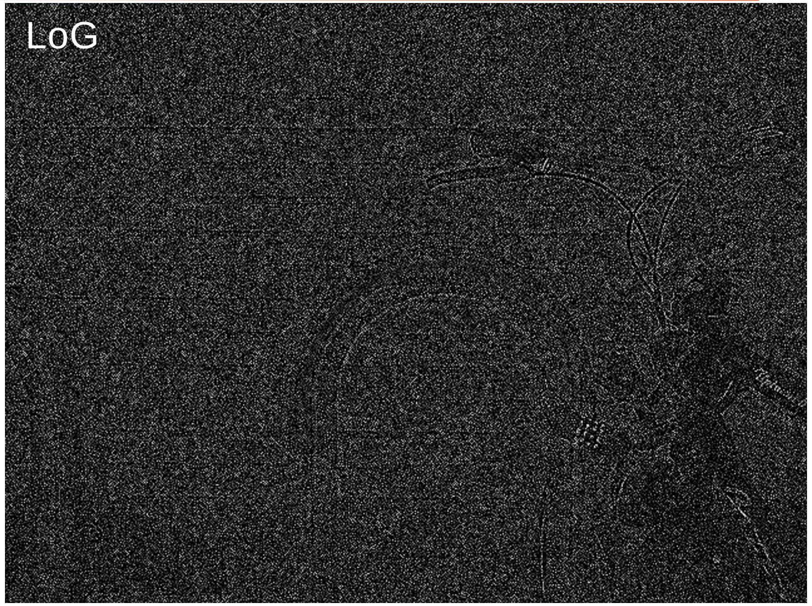
EXERCISE
Open Example 9 or 10 (A, B or C) and perform a Laplacian of Gaussian filter. Then try a Sobel filter

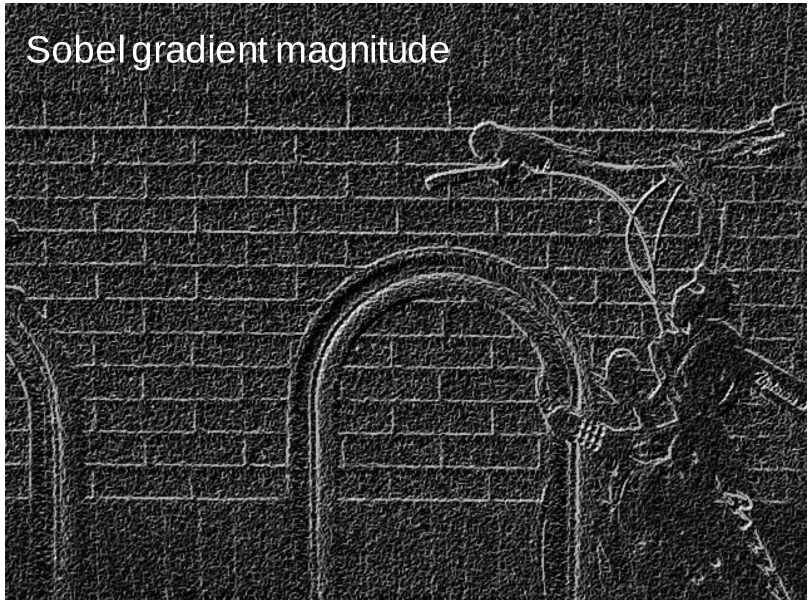
Linear filters: Laplacian of Gaussian (LoG) vs Sobel

EXERCISE
Open Example 9 (A, B or C) and perform a Laplacian of Gaussian filter. Then try a Sobel filter



The LoG is
- Computationally faster
- More precise
- Very prone to noise





Linear filters: Overview

Averaging → smoothing
(all coefficients > 0)

Difference → sharpening
(some coefficients < 0)

Gradient → edge detection
(first derivative)

Laplacian → edge detection
(second derivative)

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Spatial filters

Local filters

Surrounding pixel info is used: **kernel**

1x1 kernel (=point operation) $[1]$ $[2]$

3x3 kernel (=filter)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

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

Linear filters

Smoothing filters
Gaussian filters
Gradient filters
Laplacian filters

Non-linear filters

Median filter
Variance filter
Minimum filter
Maximum filter

Non-local filters

Find information similar to the current pixel, anywhere in the image. Replace it by the mean, median, … of those non-local values

Examples:
Non local means
Bilateral filter
Anisotropic diffusion

Non-Linear filters

Smoothing and blurring ≠ noise removal

Linear filters: **all** pixels in the kernel are used

Non-Linear filters: from all pixels in the kernel, one - the most appropriate - is **chosen**

```
for a(u,v)
   for x
      Array=(u+/-x,v+/-x))
      a'(u,v) = f(Array)
   next
next
```

minimum filter $I'(u,v) \leftarrow \min\{I(u+i,v+j) \mid (i,j) \in R\}$

Maximum filter $I'(u,v) \leftarrow \max\{I(u+i,v+j) \mid (i,j) \in R\}$

Median filter $I'(u,v) \leftarrow \text{median}\{I(u+i,v+j) \mid (i,j) \in R\}$

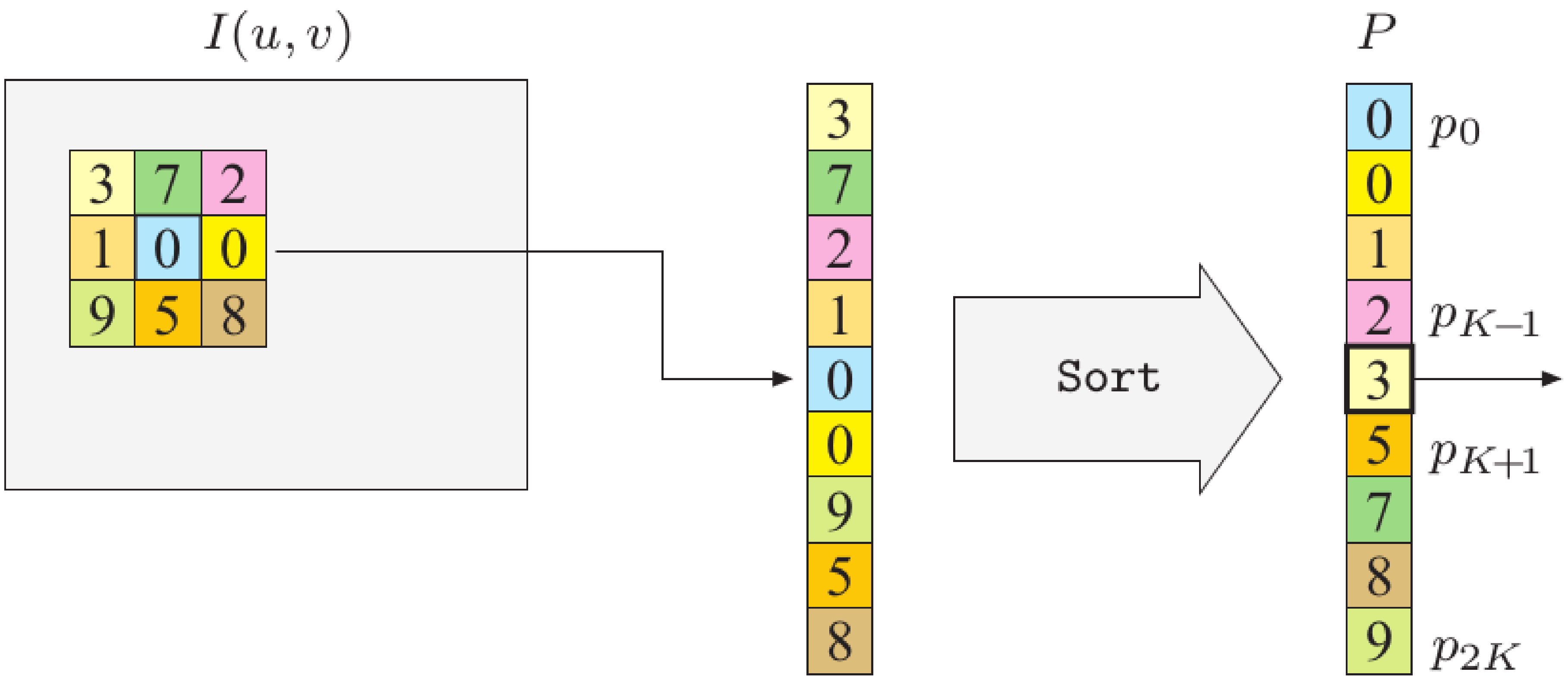
Camera man

Camera man – minimum filter 2px radius

Camera man – maximum filter 2px radius

Camera man – median filter 2px radius

Non-linear filters



Non-linear filters

EXERCISE
Noise reduction: open Example 9 or Example 10(A/B/C) and try to reduce the noise using linear filters (Gaussian smoothing) and non-linear filters (median).

Linear filter
Process > Filters > Gaussian blur

Non-linear filter
Process > Filters > Median

Non-linear filters

EXERCISE
Noise reduction: open Example 9 or example 10 (A/B/C) and try to reduce the noise using linear filters (Gaussian smoothing) and non-linear filters (median).

Process > Filters > Gaussian blur / Median

Camera man + Pepper & Salt noise
(=multiplicative noise)

Non-linear filters

Original camera man

Camera man + noise

Linear filter (Gaussian)

Non-Linear filter (Median)

Non-linear filters: Variance

EXERCISE
Exploit the relative absence of variance in the background to mask the cells (use Example 10)

Example: Bright field image of cells.

Process > Filters > Variance...

Non-linear filters: Variance

EXERCISE

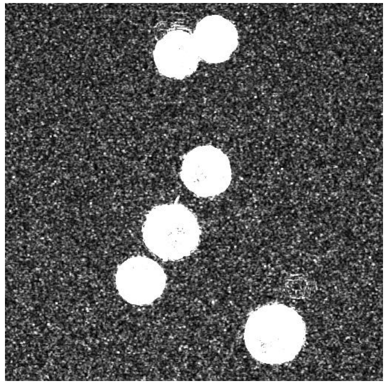
Exploit the relative absence of variance in the background to mask the cells

Example: Bright field image of cells.

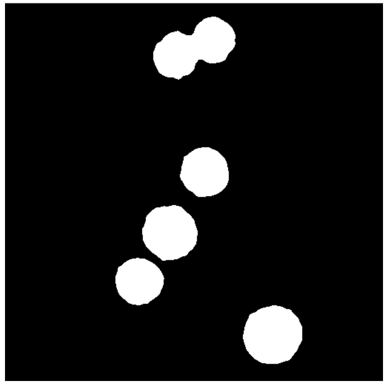
Process > Filters > Variance...

Raw data

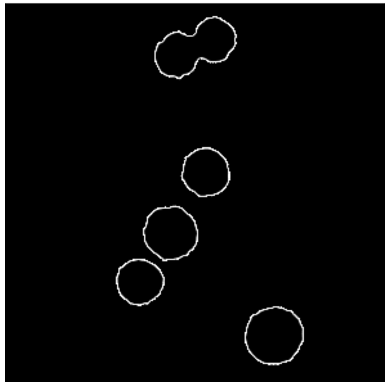
Variance filter (r=1)



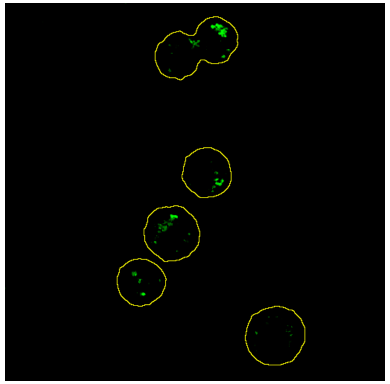
Substract 50000
Median filter

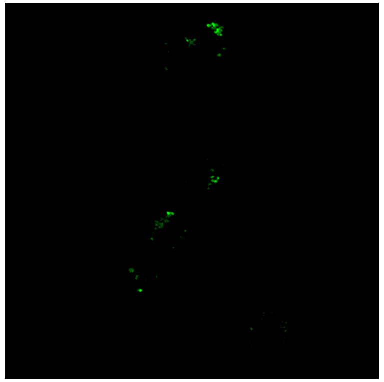


Laplacian of Gaussian



Result (overlay)





Example of a image processing pipeline!

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Spatial filters

Local filters

Surrounding pixel info is used: **kernel**

1x1 kernel (=point operation) $[1]$ $[2]$

3x3 kernel (=filter)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

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

<u>Linear filters</u>

Smoothing filters
Gaussian filters
Gradient filters
Laplacian filters

<u>Non-linear filters</u>

Median filter
Variance filter
Minimum filter
Maximum filter

Non-local filters

Find information similar to the current pixel, anywhere in the image. Replace it by the mean, median, … of those non-local values

Examples:

Non local means:
Averages neighbours with similar neighbourhoods

Bilateral filter (Adaptive smoothing):
Averages neighbours with similar intensities.
Pixel-based

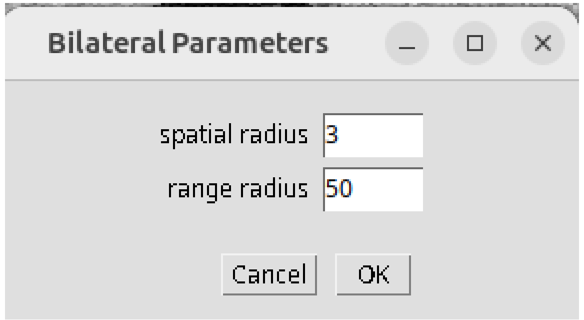
Anisotropic diffusion (adaptive smoothing):
Averages neighbours with similar intensities.
Based on *variational framework*, where some image functional (cost-function) is minimized

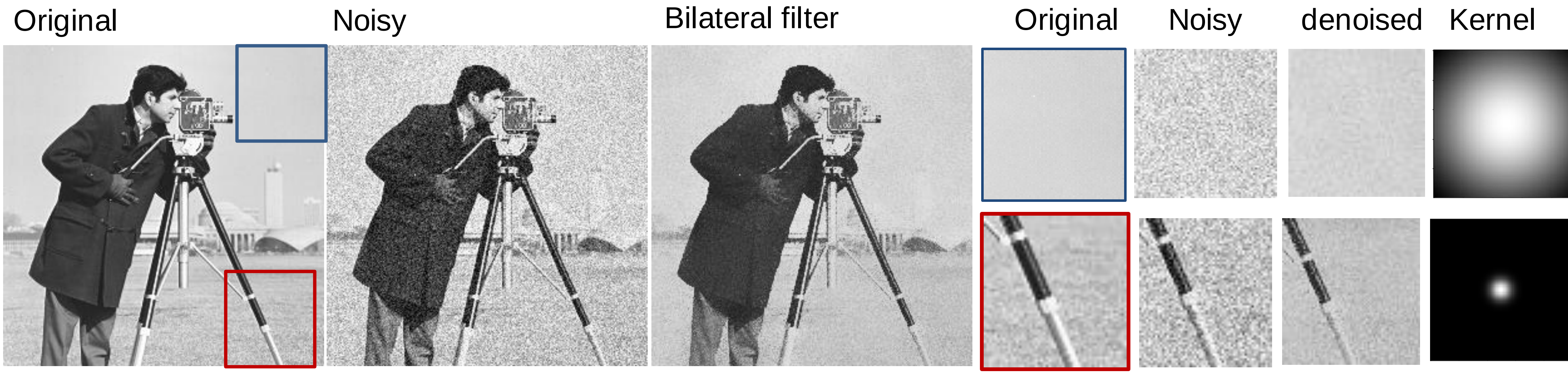
Non-local filters: bilateral filter

Concept

imagine, the kernel size of a **Gaussian filter** is variable… depending on the gradient magnitude

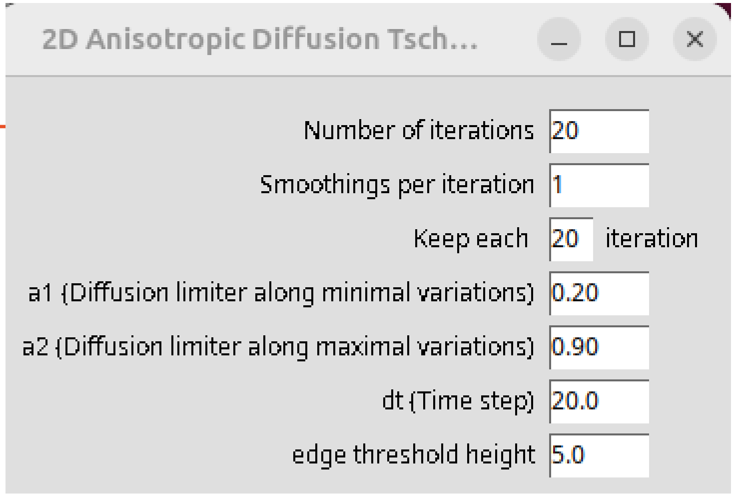
High gradient magnitude = small kernel
Low gradient magnitude = high kernel





Range: the higher range radius, the more the filter mimicks Gaussian convolution
Spatial: the higher the spatial radius, the more smoothing is applied

Non-local filters: anisotropic diffusion filter



Concept

Same concept of adaptive gaussian filters, but based on heat diffusion physics

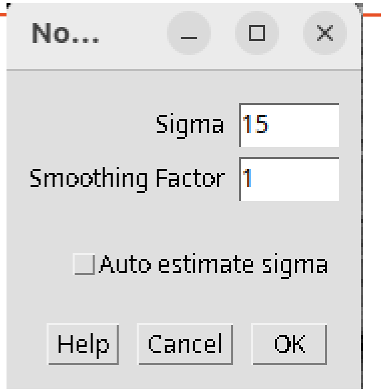
High gradient magnitude = small kernel

Low gradient magnitude = high kernel

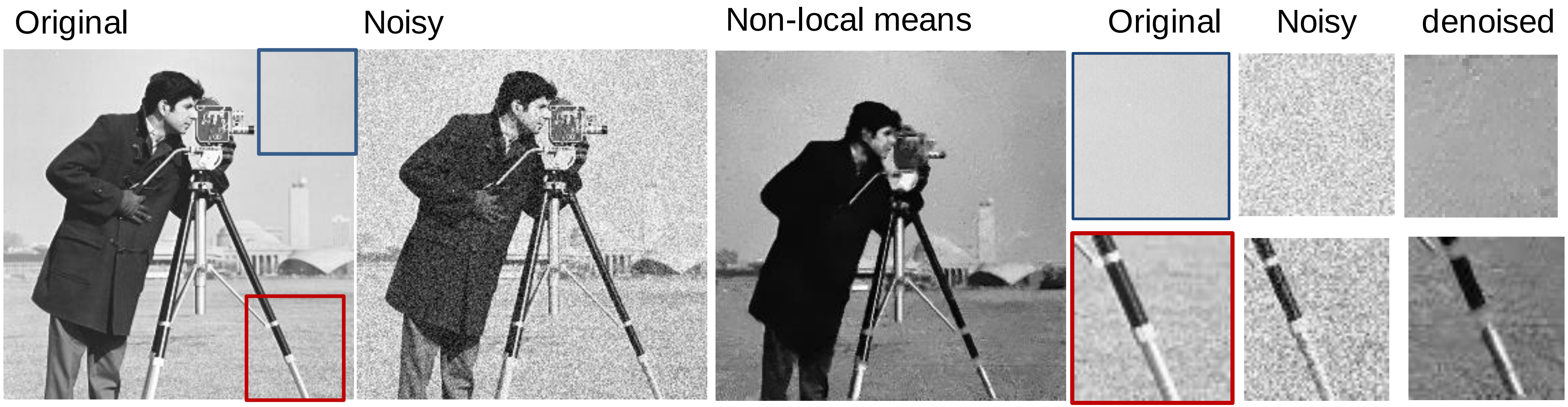
Original | Noisy | Anisotropic diffusion | Original | Noisy | denoised

Iterative!

Non-local filters: Non local means



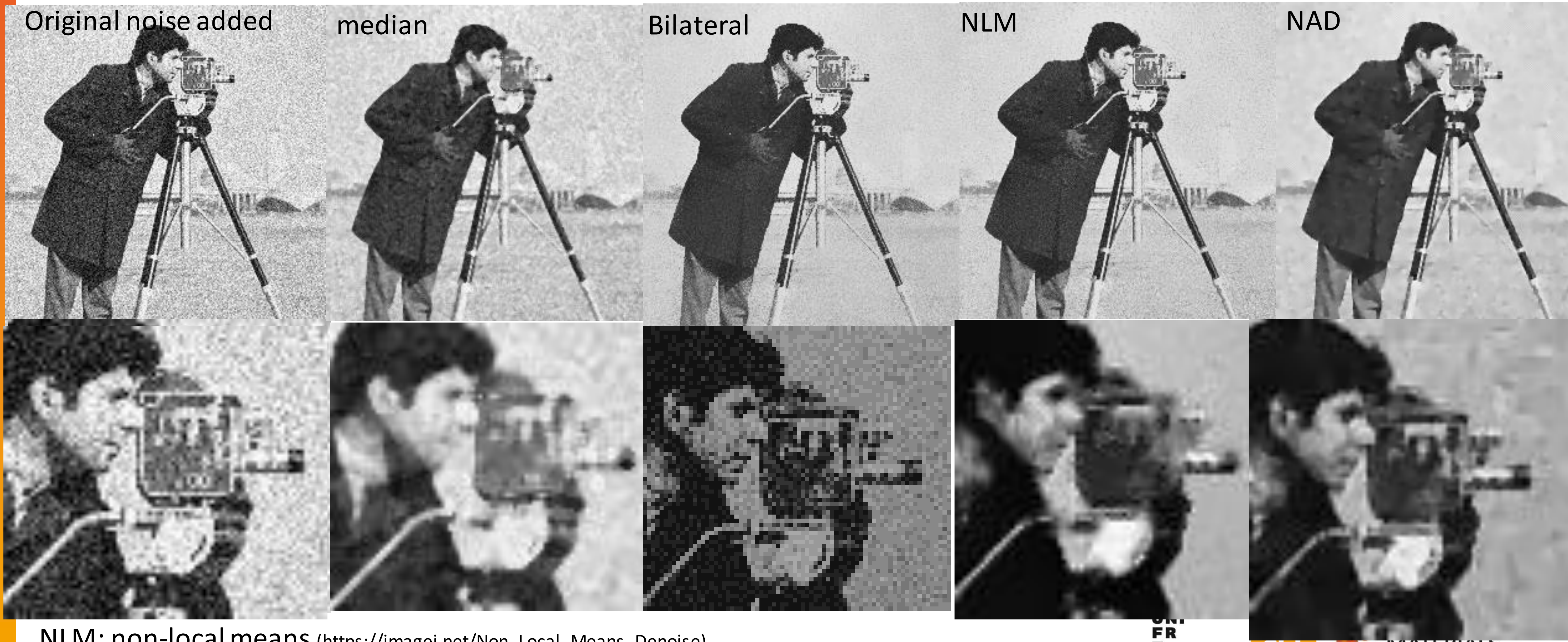
Concept
Unlike "local mean" filters, which take the mean value of a group of pixels surrounding a target pixel to smooth the image, non-local means filtering takes a mean of all pixels in the image, weighted by how similar these pixels are to the target pixel.



Sigma: "kernel size", or how far the pixels may derive from the target pixel
Smoothing factor: Additional local gaussian smoothing (1 means no smoothing)

Non local filters = noise reduction

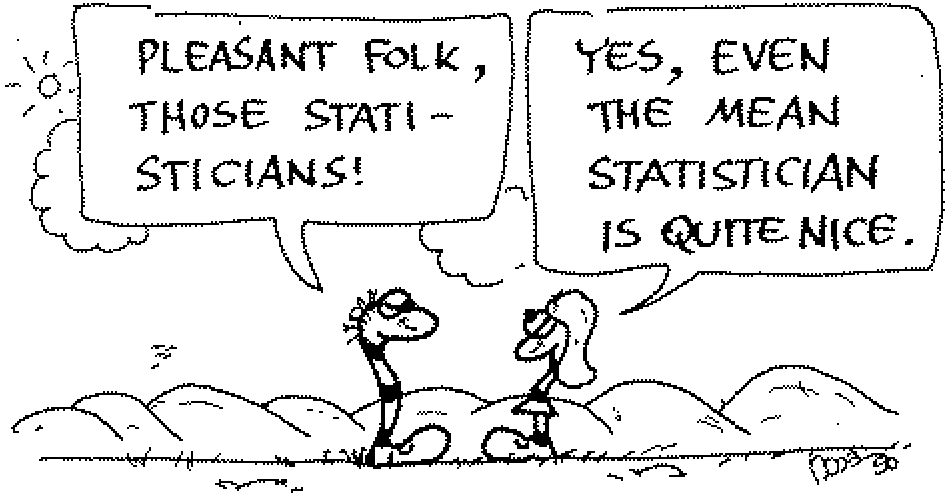
Noise = poisson noise (=shot noise), not salt and pepper



NLM: non-local means (https://imagej.net/Non_Local_Means_Denoise)

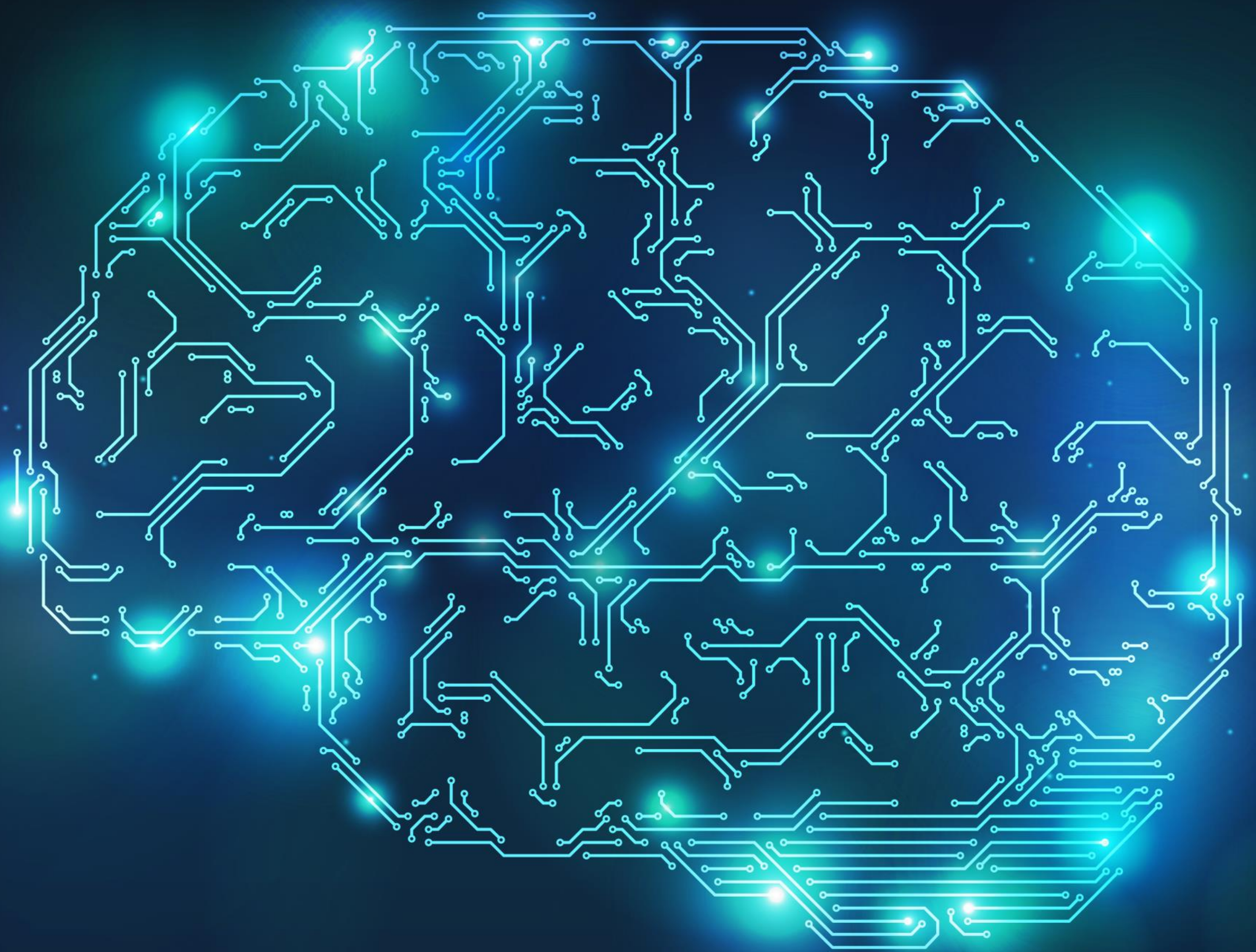
AD: non-linear anisotropic diffusion (https://imagej.nih.gov/ij/plugins/anisotropic-diffusion-2d.html)

Filters: summary

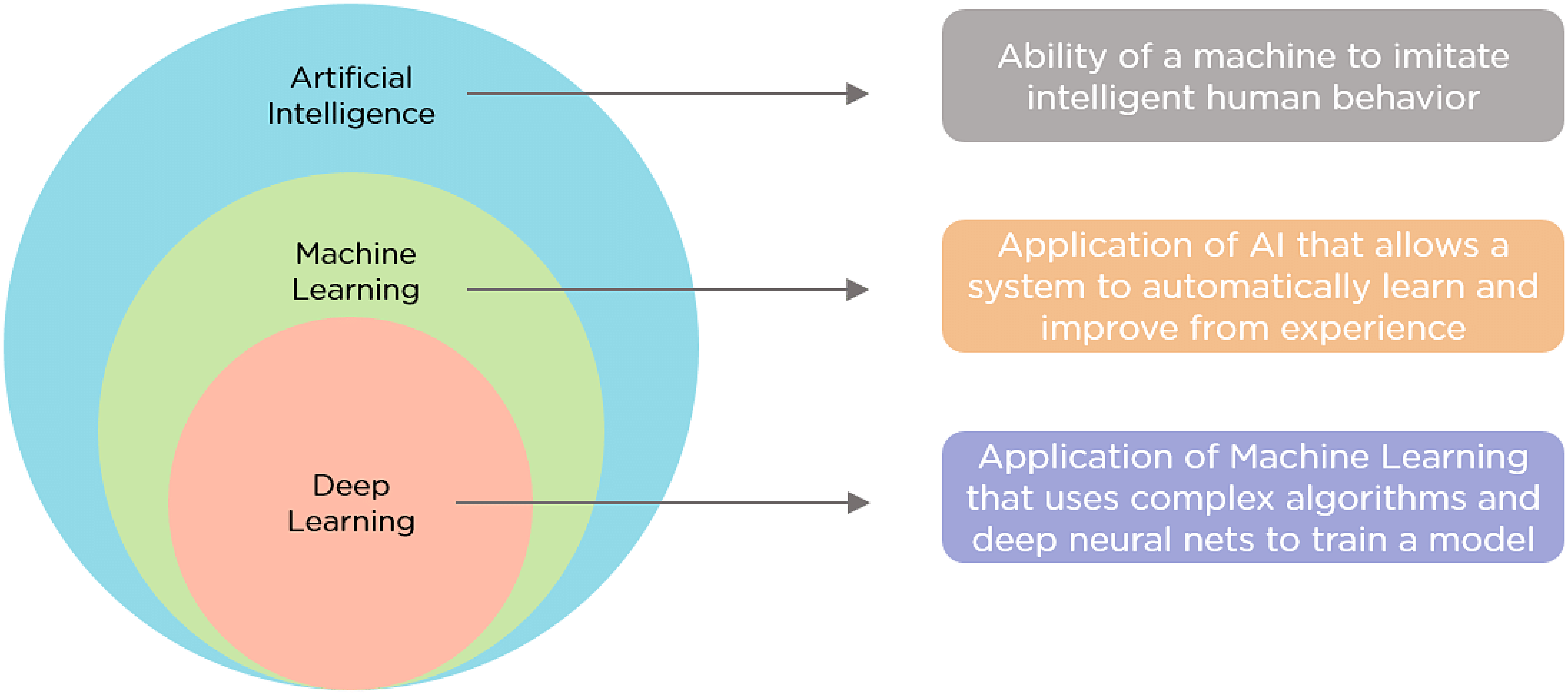


UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG
BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

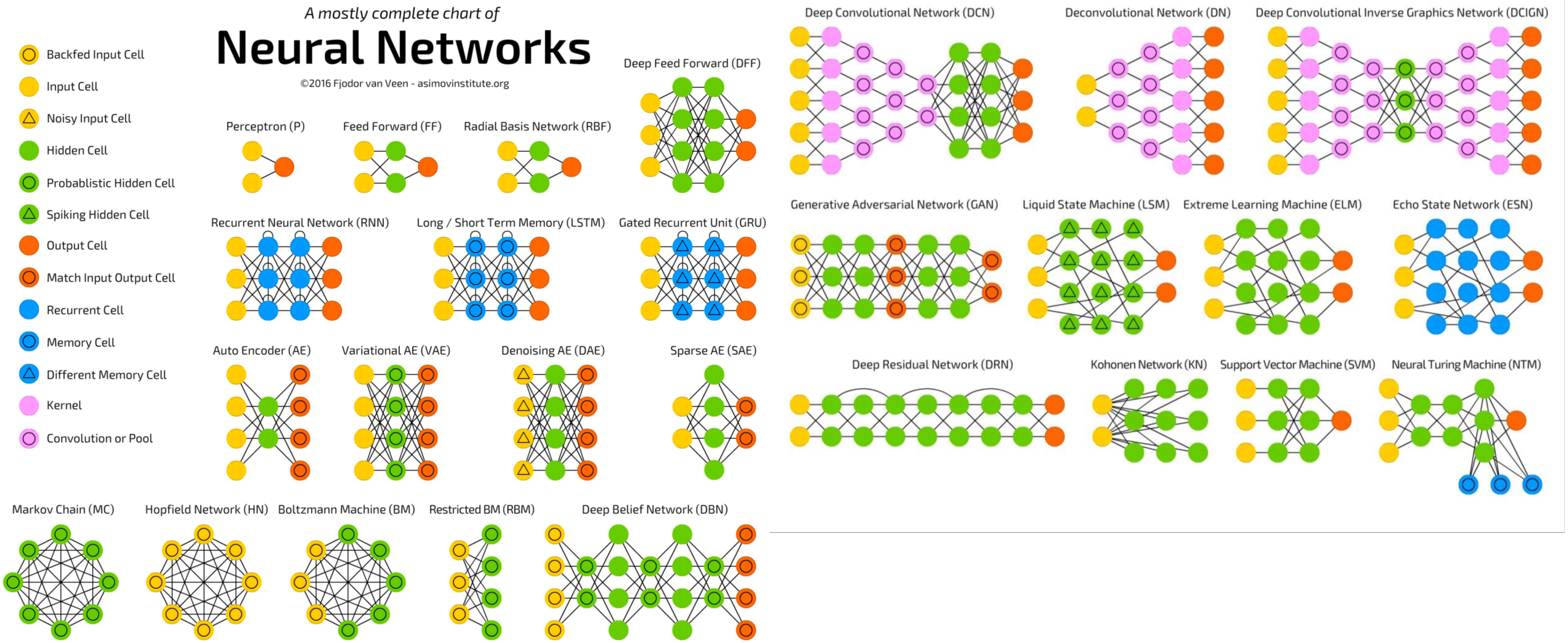
Part V: Machine learning



Machine learning



Neural networks



Deep convoluted neural networks

$$y(u,v) = (h * x)(u,v) + n(u,v)$$

$$\hat{x}(u,v) = (g * y)(u,v)$$

Assume you have

- y(u,v) (the observed image)
- x(u,v) (the image without noise)
- h(u,v) (the point spread function is 1)

Brute-force calculate g(u,v) until n(u,v) is minimal

- Input: x(u,v) and y(u,v) examples
- Stochastic gradient descent
- Iterative learning algorithm

Output

- A model (readible by software)
- That can predict how to adjust pixel intensities
- How it works: ?

Plugins > CSBDeep > N2V > N2V Train

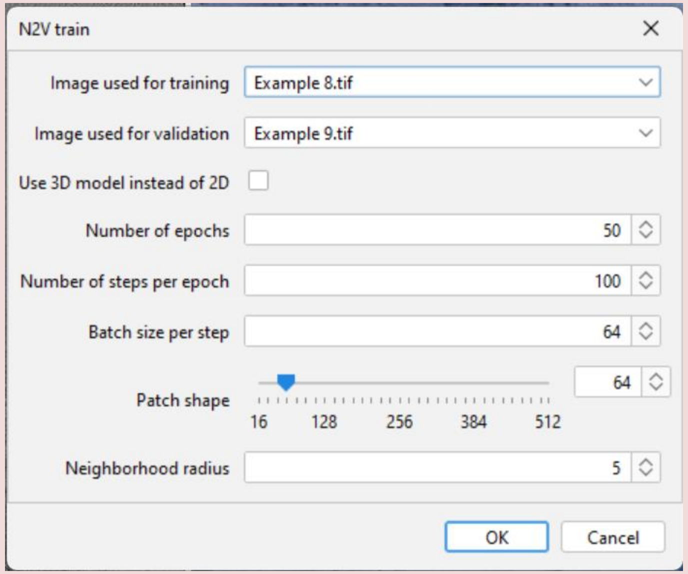


Image used for training: y(u,v)
Image used for validation: x(u,v)

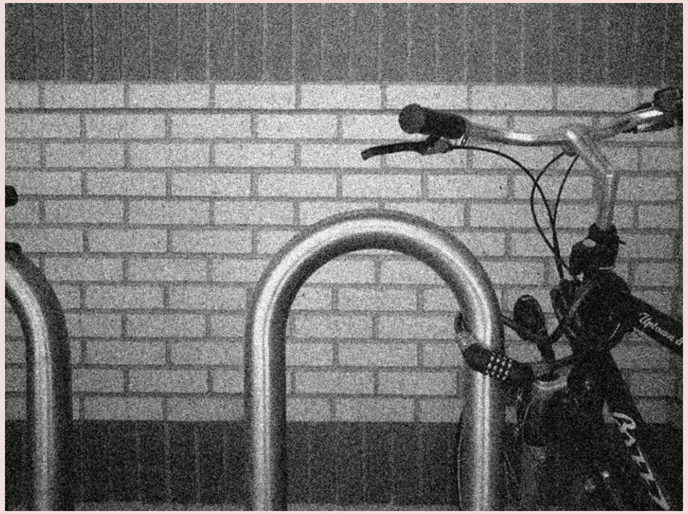
Epochs: one full cycle through the training dataset (= many iterations)

Batch size: The number of training samples (parts of an image) used in one iteration

Number of steps: Total Number of Training Samples / Batch Size

Original

Process > Noise > Add noise



Deep convoluted neural networks

$$y(u,v) = (h * x)(u,v) + n(u,v)$$

$$\hat{x}(u,v) = (g * y)(u,v)$$

Assume you have

- y(u,v) (the observed image)
- x(u,v) (the image without noise)
- h(u,v) (the point spread function is 1

Brute-force calculate g(u,v) until n(u,v) is minimal

- Input: x(u,v) and y(u,v) examples
- Stochastic gradient descent
- Iterative learning algorithm

Output

- A model (readible by software)
- That can predict how to adjust pixel intensities
- How it works: ?

Plugins > CSBDeep > N2V > N2V Train

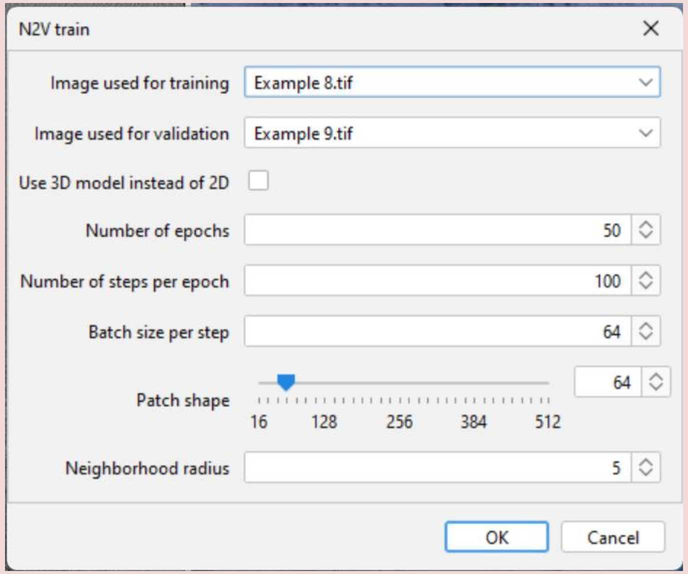


Image used for training: y(u,v)
Image used for validation: x(u,v)

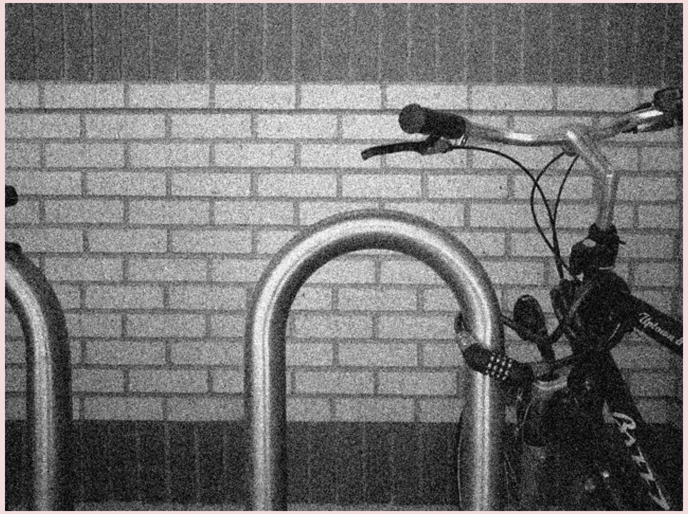
Epochs: one full cycle through the training dataset (= many iterations)

Batch size: The number of training samples (parts of an image) used in one iteration

Number of steps: Total Number of Training Samples / Batch Size

Original

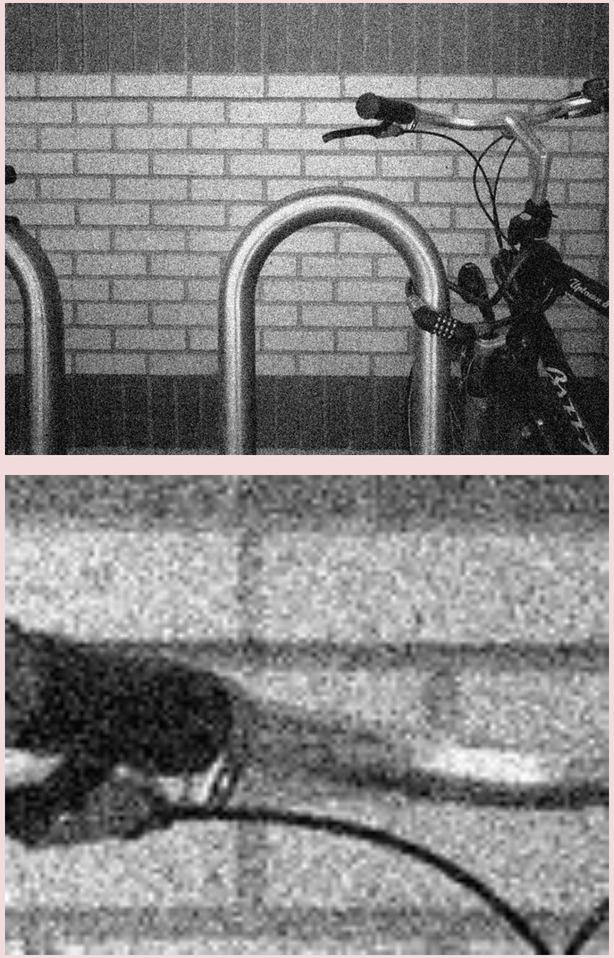
Process > Noise > Add noise



Deep convoluted neural networks: example

Original image

Noise added



Random noise is added to the image. The noise is Gaussian distributed with a mean of 0 and SD of 25.

Plugins > CSBDeep > N2V > N2V Predict

Repositories
CBSDeep and **Tensorflow** must be installed.

Deep convoluted neural networks: example

FIJI repositories

	Name	URL
☑	CSBDeep	https://sites.imagej.net/CSBDeep/
☑	TensorFlow	https://sites.imagej.net/TensorFlow/

ImageJ options: Edit > Options > Tensorflow...

TensorFlow library version management

Please select the TensorFlow version you would like to install.

Filter by.. Mode: - CUDA: - TensorFlow: -

- ( ) TF 1.15.0 CPU
- (•) TF 1.15.0 GPU (CUDA 10.0, CuDNN >= 7.4.1)
- ( ) TF 1.14.0 CPU
- ( ) TF 1.14.0 GPU (CUDA 10.0, CuDNN >= 7.4.1)
- ( ) TF 1.13.1 CPU
- ( ) TF 1.13.1 GPU (CUDA 10.0, CuDNN 7.4)
- ( ) TF 1.12.0 CPU
- ( ) TF 1.12.0 GPU (CUDA 9.0, CuDNN >= 7.2)
- ( ) TF 1.11.0 CPU
- ( ) TF 1.11.0 GPU (CUDA 9.0, CuDNN >= 7.2)

Using native TensorFlow version: TF 1.15.0 GPU (CUDA 10.0, CuDNN >= 7.4.1)

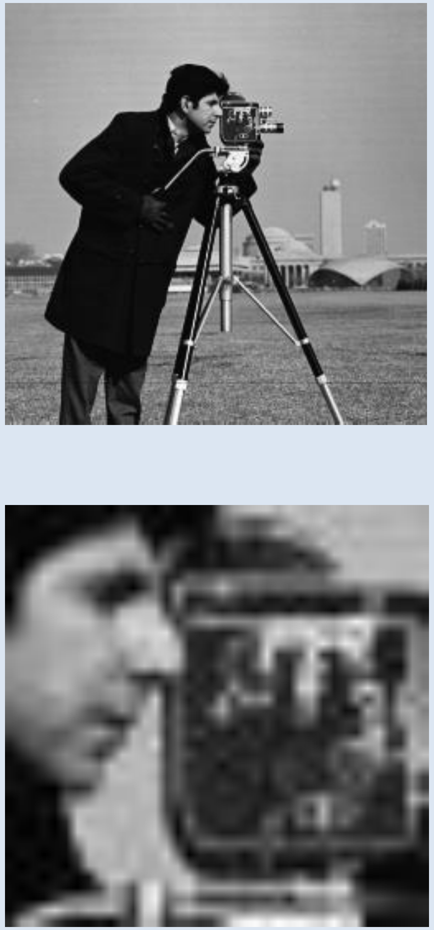
On The PC:

```
[redacted]:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Fri_Sep__8_19:17:24_PDT_2023
Cuda compilation tools, release 12.3, V12.3.52
Build cuda_12.3.r12.3/compiler.33281558_0
```

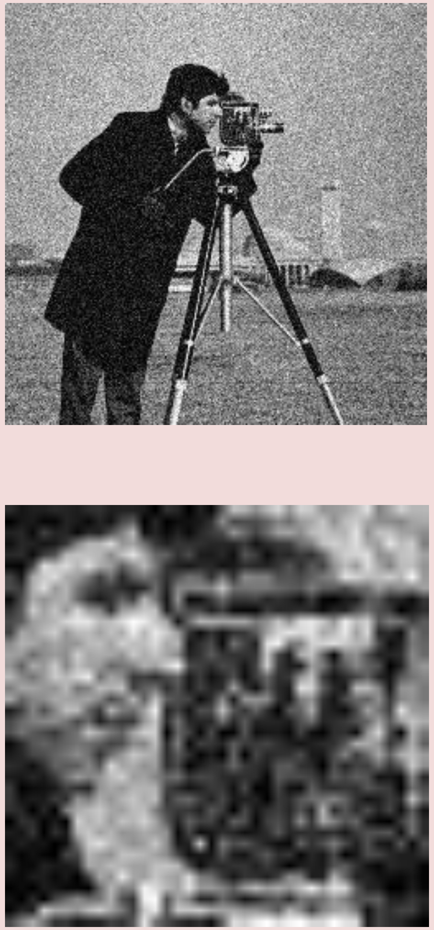
```
[redacted]:~$ nvidia-smi
Tue Mar  5 12:09:59 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 545.23.06              Driver Version: 545.23.06    CUDA Version: 12.3     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 3050 ...    On  | 00000000:01:00.0 Off |                  N/A |
| N/A   53C    P8               8W /  60W |     99MiB /  4096MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|    0   N/A  N/A      1854      G   /usr/lib/xorg/Xorg                            4MiB |
|    0   N/A  N/A      2690      G   /usr/lib/xorg/Xorg                            4MiB |
|    0   N/A  N/A     58025      C   ...linux64 New/Fiji.app/ImageJ-linux64       80MiB |
+---------------------------------------------------------------------------------------+
```

Deep convoluted neural networks: example

Original image



Noise added



N2V Prediction

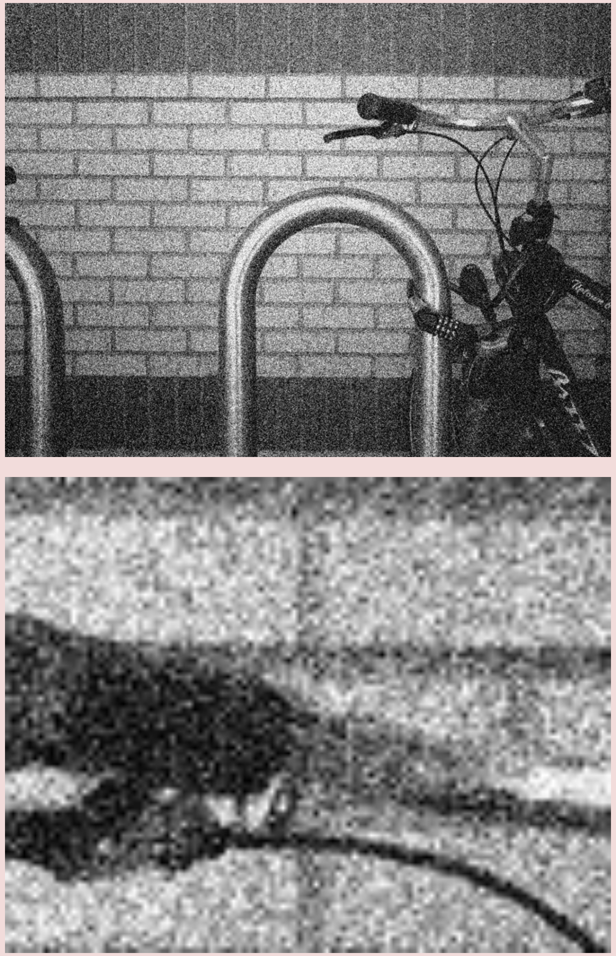
UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Deep convoluted neural networks: example

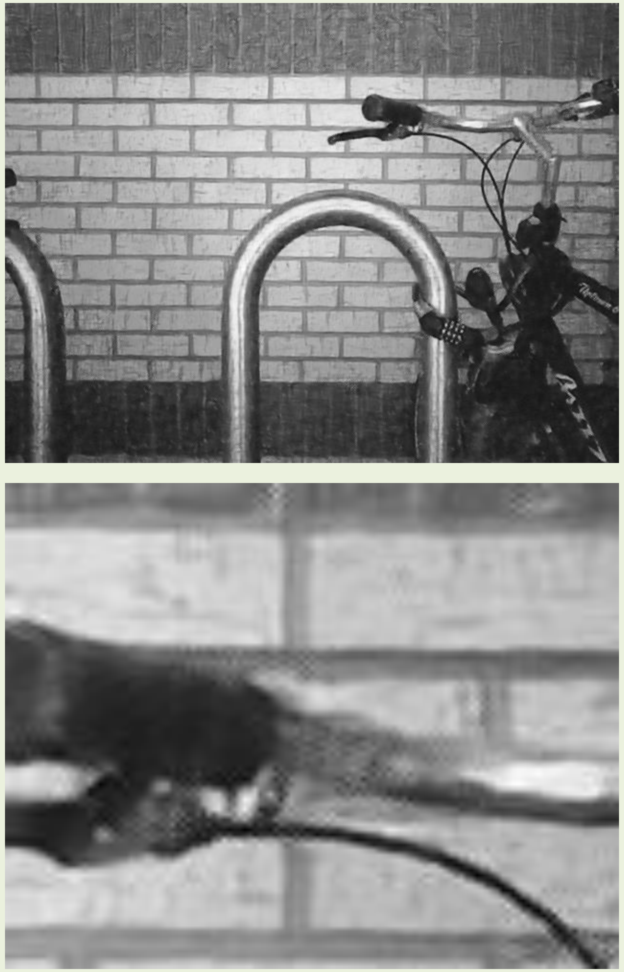
Original image

Noise added



Random noise is added to the image. The noise is Gaussian distributed with a mean of 0 and SD of 50.

Plugins > CSBDeep > N2V > N2V Predict



Congratulations,
You finished Part II, Advanced image processing

For Part III,
Install from the repos:
- DeepImageJ
- LabKit

From the internet:
Ilastik (ilastik.org)