



A. Kaestner :: Paul Scherrer Institut

Image enhancement

Preparing the images for analysis

# Outline

- 1 Introduction
- 2 Noise and Artifacts
- 3 Basic filtering
- 4 Scale spaces
- 5 Verification
- 6 Summary
- 7 Exercises

# Introduction

# Introduction

3D and 4D imaging produce large amounts of data



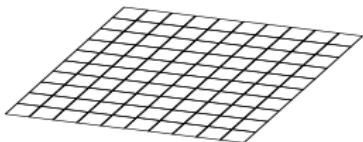
Gigabytes...  
... or even  
terabytes of data

- 3D visualization
- Sample characterization
- Process parameterization
- etc

# Different types of images

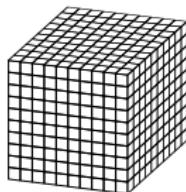
## 2D

- Pictures
- Radiographs
- CT slices



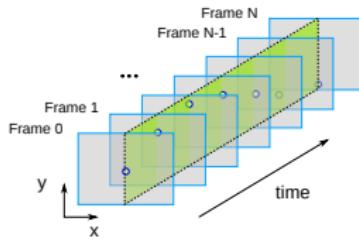
## 3D

- Volumes
- $x, y, z$
- Movies
- $x, y, t$



## 4D

- Volume movie
- $x, y, z, t$



Which information do you want to gain

## Quantitative

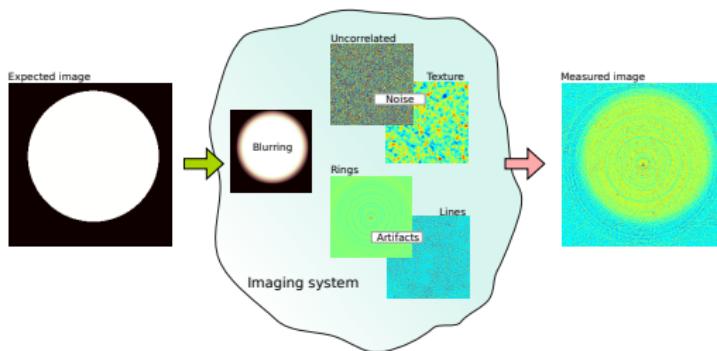
- Material composition
- Material transport

## Structure

- Identify items
- Item geometry

This will affect the choice of processing methods.

# Measurements are rarely perfect

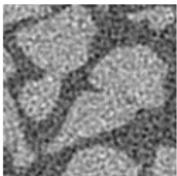


## Factors affecting the image quality

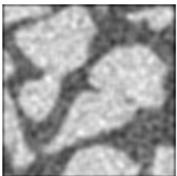
- Resolution (Imaging system transfer functions)
- Noise
- Contrast
- Inhomogeneous contrast
- Small relevant features
- Artifacts

# A typical processing chain

Image  
Acquisition



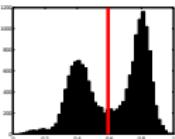
Enhancement



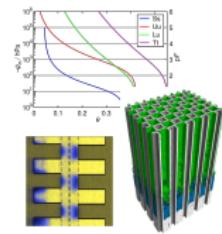
Segmentation



Post processing



Evaluation



Todays lecture will focus on the enhancement

# Noise and artifacts

The unwanted information

# Noise types

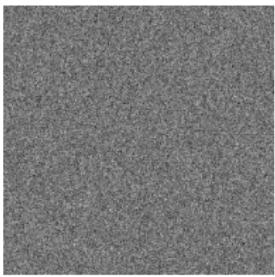
Spatially uncorrelated noise

Event noise

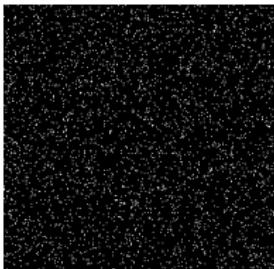
Structured noise

## Noise examples

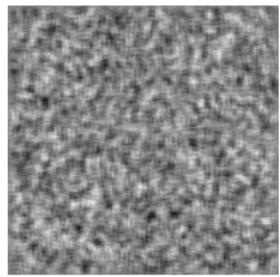
Gaussian



Salt 'n pepper



Structured



## Gaussian noise

- Additive
- Easy to model
- Other distributions obtain Gaussian shape at large numbers

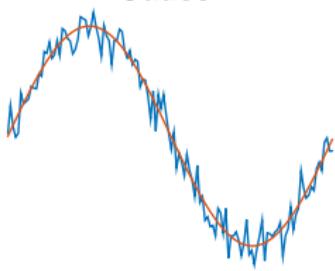
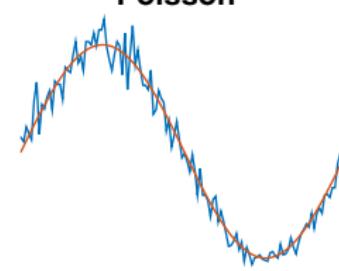
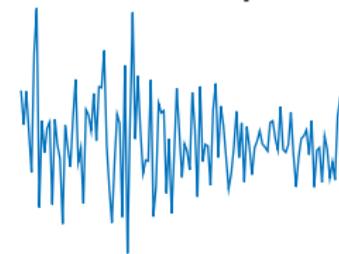
$$n(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{x-\mu}{2\sigma})^2}$$

## Poisson noise

- Multiplicative
- Physically correct for event counting

$$p(x) = \frac{\lambda^k}{k!} e^{-\lambda} x$$

# Noise examples

**Gauss****Poisson****Gaussian noise component****Poisson noise component**

- A type of outlier noise
- Noise strength give as the probability of an outlier
- Additive, multiplicative, independent replacement

## Example

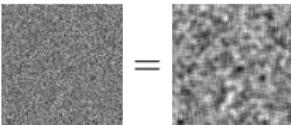
$$sp(x) = \begin{cases} -1 & x \leq \lambda_1 & x \in \mathcal{U}(0, 1) \\ 0 & \lambda_1 < x \leq \lambda_2 & \lambda_1 < \lambda_2 \\ 1 & \lambda_2 < x & \lambda_1 + \lambda_2 = \text{noise fraction} \end{cases}$$

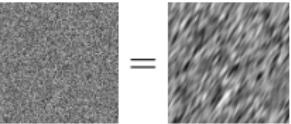
- Spatially correlated
- Example: Detector structure

## Example random field models

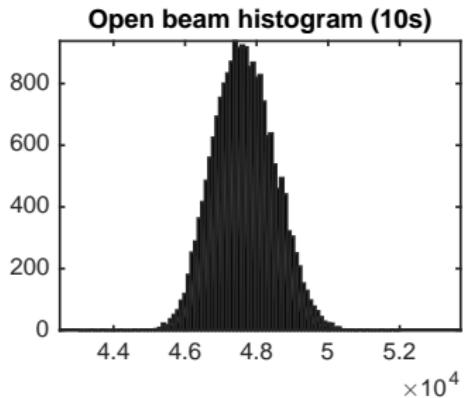
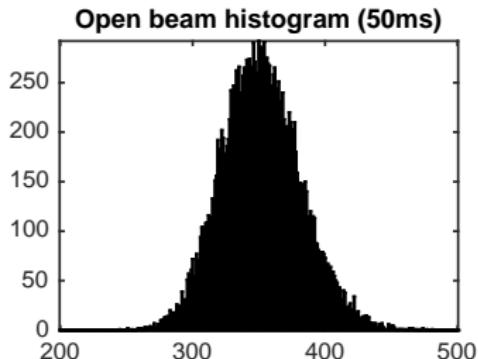
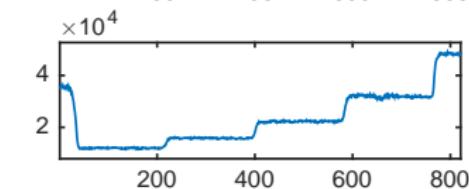
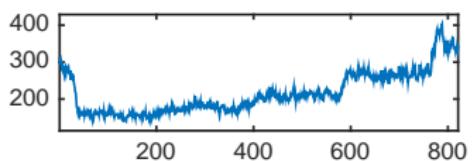
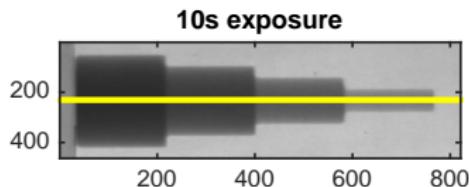
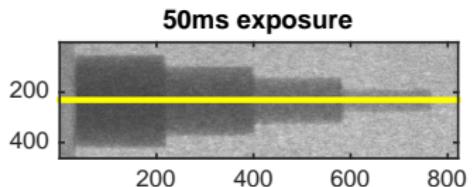
$$n(x, y) \in \mathcal{N}(\mu, \sigma)$$

$$ns = K * n \quad K = \text{convolution kernel}$$

$$U_{5 \times 5} * \quad = \quad$$


$$\quad * \quad = \quad$$


## Noisy profiles



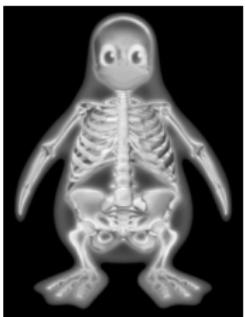
# Signal to noise ratio

A metric to describe noise strength

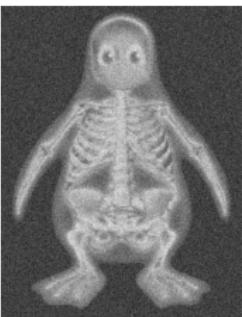
$$SNR = \frac{\mu_{image}}{\sigma_{image}} \quad (1)$$

$$SNR_{db} = 20 \log \frac{\mu_{image}}{\sigma_{image}} \quad (2)$$

- Select a region
- Compute average intensity
- Compute std deviation
- Apply eqns 1 or 2



SNR =  $\infty$



SNR = 5



SNR = 2

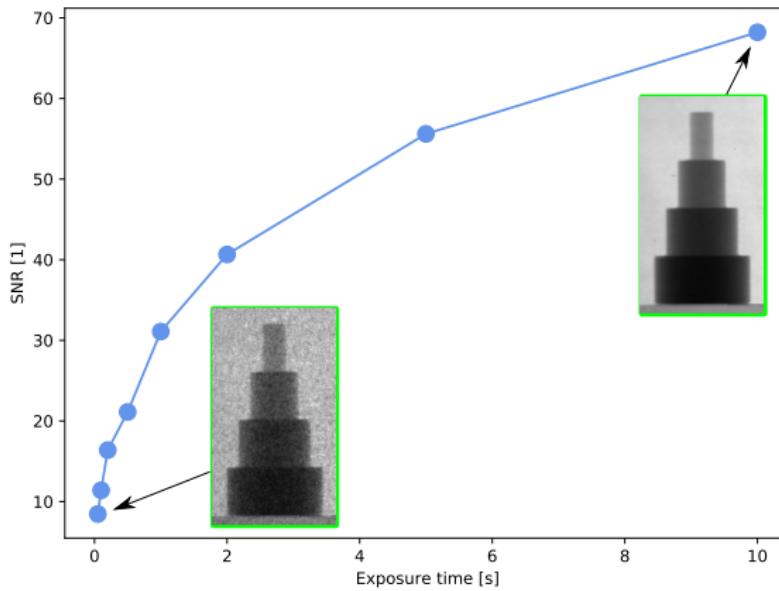


SNR = 1

## SNR for different exposure times

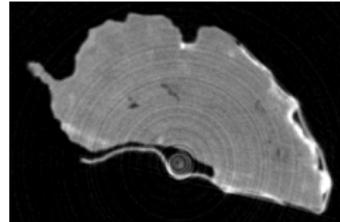
## Poisson noise

- $E[Poi(\lambda)] = \lambda$  and  $var[Poi(\lambda)] = \lambda \rightarrow \text{SNR} = \sqrt{\lambda}$
- Exposure time increase the number of events



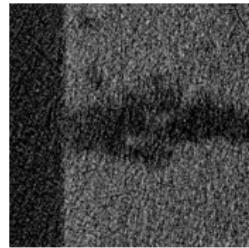
## Rings

- Appear in most CT acquisitions
- Caused by stuck pixels in the projection data
- Can mostly be suppressed during reconstruction



## Lines

- Frequent in neutron CT slices
- Caused by spots on single projections
- Can mostly be suppressed during reconstruction

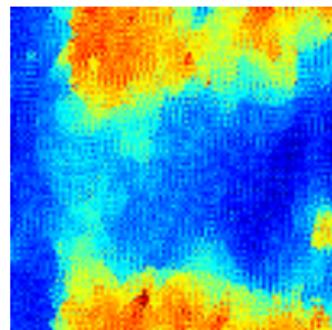


## Rounding errors

- May appear with sum operations on large data sets.
- At some point the new term is smaller than the precision.

## Instable processing

- Due to incorrect regularization
- Wrong parameterization
- Incorrect implementation... bugs etc



# Image processing with Python

Python provides several packages for matrix handling and image processing:

- `numpy` Numeric operations scalars, lists, and matrices.
- `skimage` Image processing for 2D and 3D images.
- `matplotlib` Plotting and visualization.

## Random number generators [numpy.random]

Gauss `np.random.normal( $\mu, \sigma$ , size=[rows, cols])`

Uniform `np.random.uniform(low, high, size=[rows, cols])`

Poisson `np.random.poisson( $\lambda$ , size=[rows, cols])` alt  
`np.random.poisson(img, size=[rows, cols])`

Generates an  $m \times n$  random fields with different distributions.

## Statistics

- `np.mean(f)`, `np.var(f)`, `np.std(f)` Computes the mean, variance, and standard deviation of an image *f*.
- `np.min(f)`, `np.max(f)` Finds minimum and maximum values in *f*.
- `np.median(f)`, `np.rank()`

# Basic filtering

The first approach to image enhancement

## General definition

A filter is a processing unit that

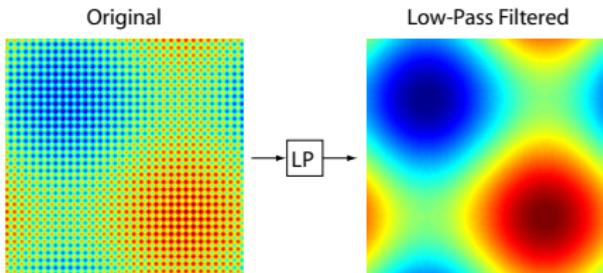
- Enhances the wanted information
- Suppresses the unwanted information

Ideally without altering relevant features beyond recognition

[Jähne, 2005]

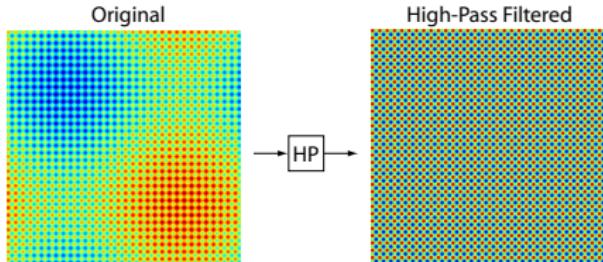
## Low pass filters

- Slow changes are enhanced
- Rapid changes are suppressed



## High pass filters

- Rapid changes are enhanced
- Slow changes are suppressed



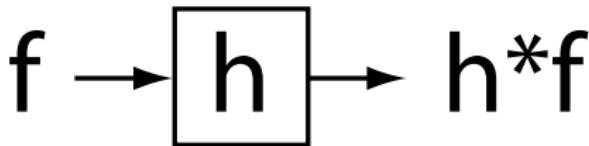
## Linear filters

Computed using the convolution operation

$$g(x) = h * f(x) = \int_{\Omega} f(x - \tau) h(\tau) d\tau \quad (3)$$

where

- $f(x)$  is the image
- $h$  is the convolution kernel of the filter



# Low-pass filter kernels

## Mean or Box filter

All weights have the same value.

Example:

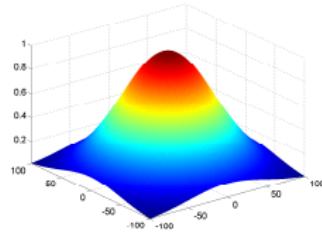
$$B = \frac{1}{25} \cdot$$

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

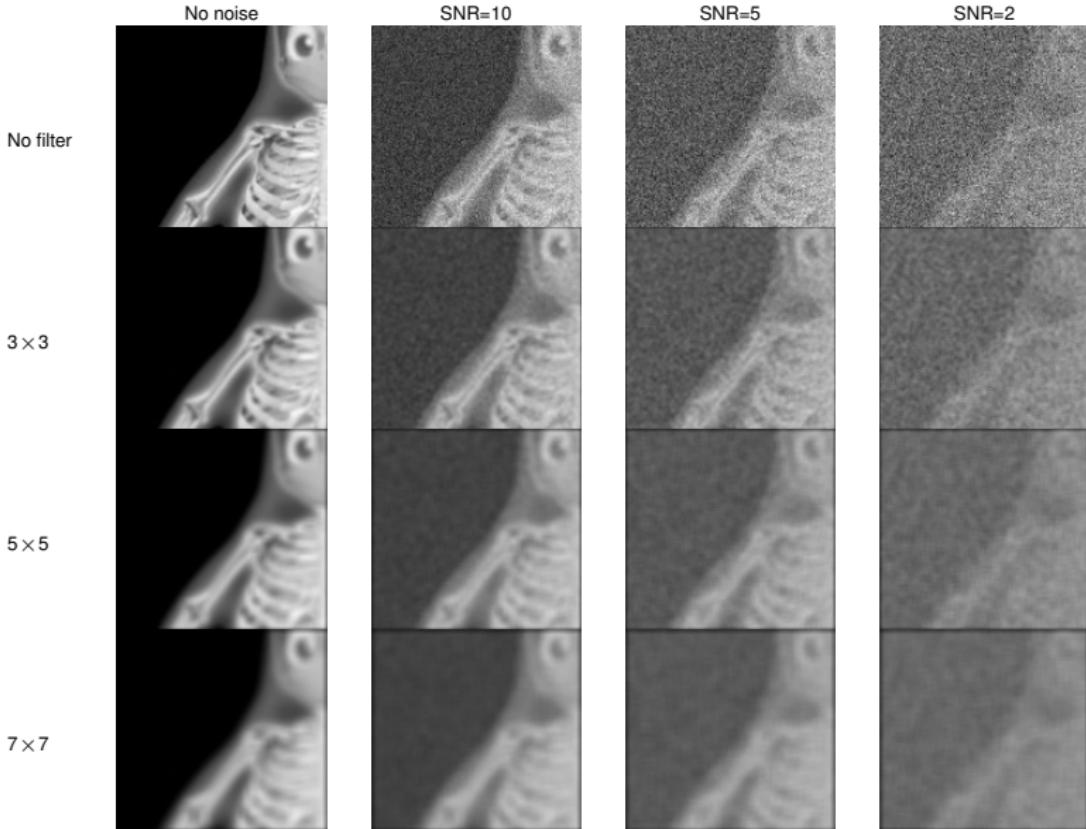
## Gauss filter

$$G = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

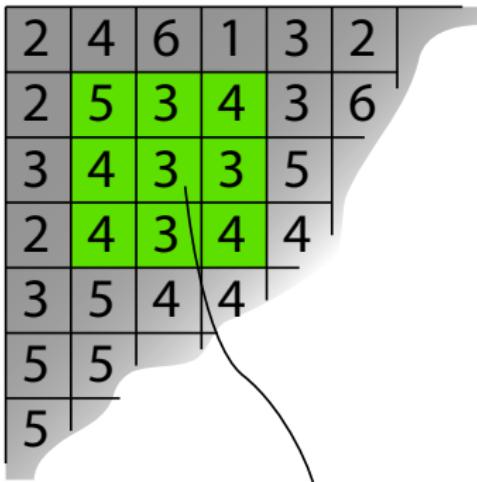
Example:



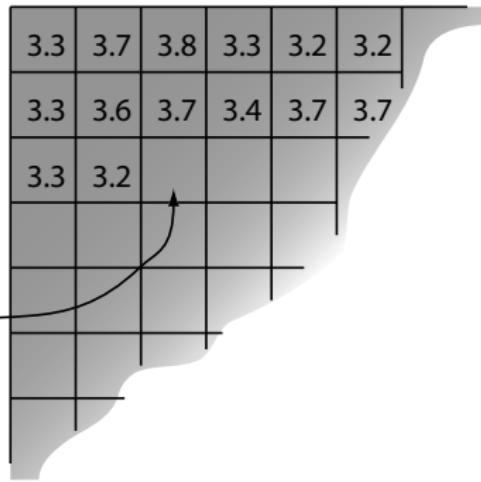
# Using a Mean filter



# How is the convolution computed



$$(5+3+4+4+3+3+4+3+4)/9=3.7$$



## Note

For a non-uniform kernel each term is weighted by its kernel weight.

The associative and commutative laws apply to convolution

$$(a * b) * c = a * (b * c) \quad \text{and} \quad a * b = b * a$$

A convolution kernel is called *separable* if it can be split in two or more parts:

$$\begin{array}{|c|c|c|} \hline \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot \\ \hline \end{array} = \begin{array}{|c|} \hline \cdot \\ \hline \cdot \\ \hline \cdot \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline \cdot & \cdot & \cdot \\ \hline \end{array}$$

## Gain

Reduces the number of computations → faster processing

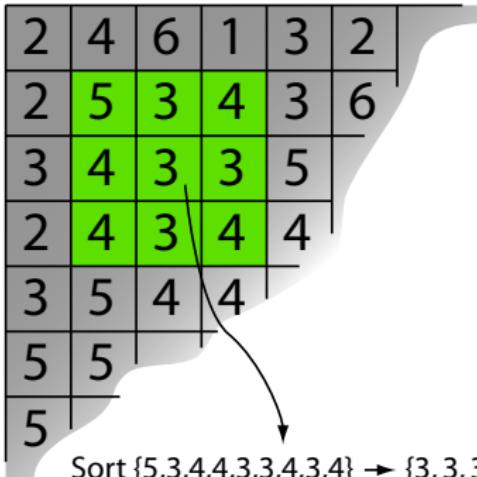
$$3 \times 3 \rightarrow 9 \text{ mult and } 8 \text{ add} \Leftrightarrow 6 \text{ mult and } 4 \text{ add}$$

$$3 \times 3 \times 3 \rightarrow 27 \text{ mult and } 26 \text{ add} \Leftrightarrow 9 \text{ mult and } 6 \text{ add}$$

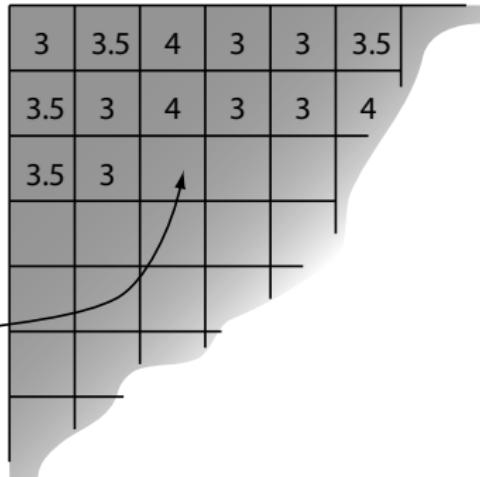
## Example

$$e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$$

# The median filter



Sort  $\{5, 3, 4, 4, 3, 3, 4, 3, 4\} \rightarrow \{3, 3, 3, 3, 4, 4, 4, 4, 5\}$



# Comparing filters for different noise types

10% salt&pepper noise



Median filtered



Mean filtered



Additive White Gaussian noise  
( $\sigma = 30$ )



Median filtered

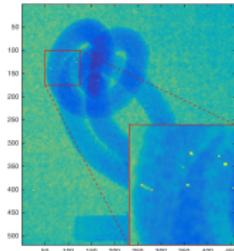


Mean filtered



## Problem

- Many neutron images are corrupted by spots that confuse following processing steps.
- The amount, size, and intensity varies with many factors.

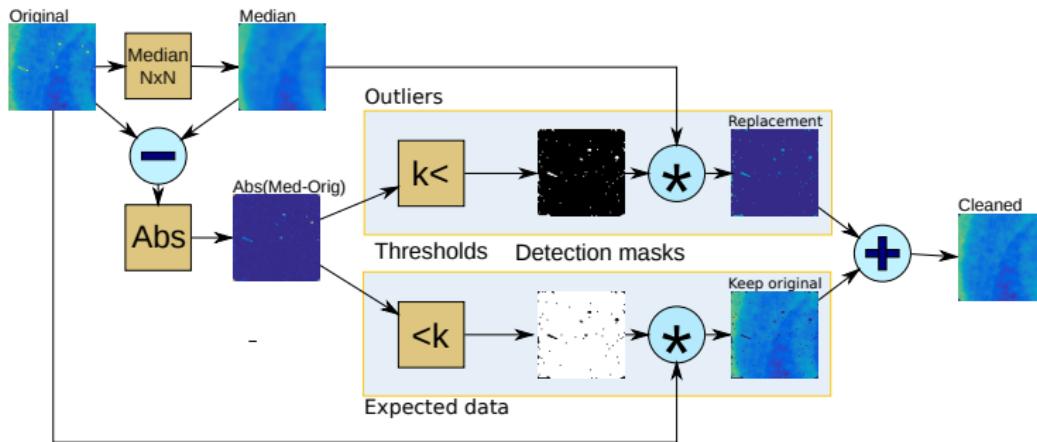


## Solutions

- :-) Low pass filter
- :-) Median filter
- :-) Detect spots and replace by estimate

# Spot cleaning algorithm

## An algorithm

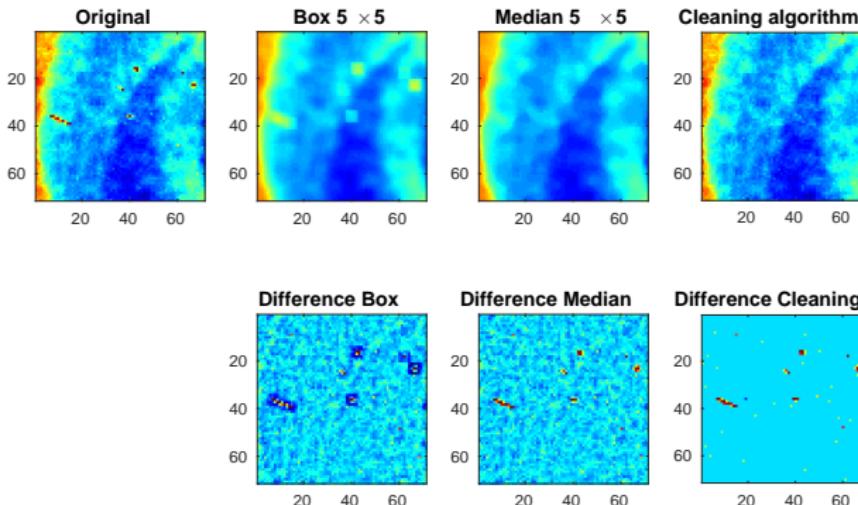


## Parameters

$N$  Width of median filter.

$k$  Threshold level for outlier detection.

# Spot cleaning – Compare performance



## The ImageJ ways

Despeckle Median ... please avoid this one!!!

Remove outliers Similar to cleaning described algorithm

# High-pass filters

High-pass filters enhance rapid changes – ideal for edge detection

Typical high-pass filters:

Gradients

$$\frac{\partial}{\partial x} = \frac{1}{2} \cdot \begin{bmatrix} -1 & 1 \end{bmatrix}$$

Laplacian

$$\Delta = \frac{1}{2} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

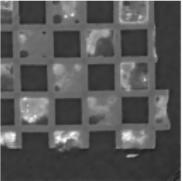
$$\frac{\partial}{\partial x} = \frac{1}{32} \cdot \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$

Sobel

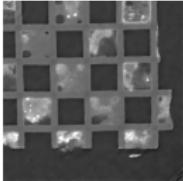
$$G = |\nabla f| = \sqrt{\left(\frac{\partial}{\partial x} f\right)^2 + \left(\frac{\partial}{\partial y} f\right)^2}$$

## Gradient example

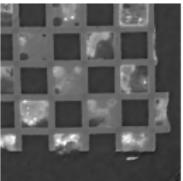
## Vertical edges

$$\frac{\partial}{\partial x} \begin{matrix} \text{Image} \\ \text{(Input)} \end{matrix} = \frac{1}{32} \cdot \begin{matrix} \text{Kernel} \\ \begin{array}{|c|c|c|} \hline -3 & 0 & 3 \\ \hline -10 & 0 & 10 \\ \hline -3 & 0 & 3 \\ \hline \end{array} \end{matrix} * \begin{matrix} \text{Image} \\ \text{(Input)} \end{matrix} = \begin{matrix} \text{Image} \\ \text{(Output)} \end{matrix}$$


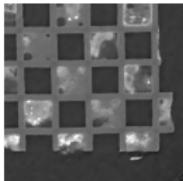
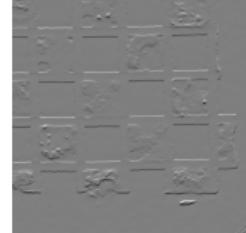
-3	0	3
-10	0	10
-3	0	3




## Horizontal edges

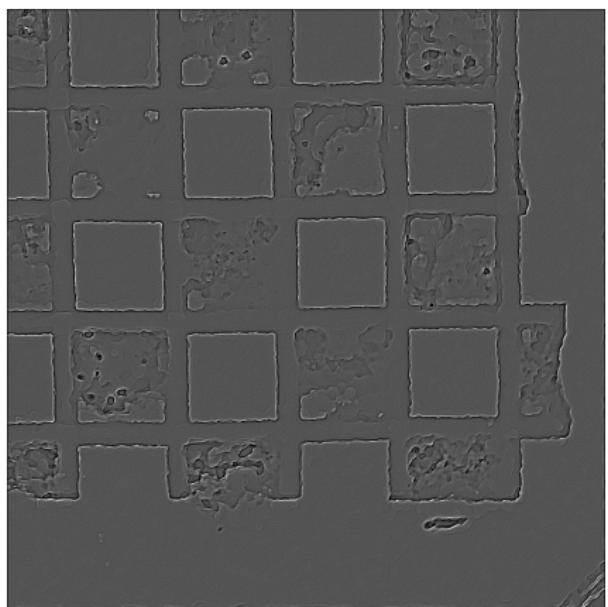
$$\frac{\partial}{\partial y} \begin{matrix} \text{Image} \\ \text{(Input)} \end{matrix} = \frac{1}{32} \cdot \begin{matrix} \text{Kernel} \\ \begin{array}{|c|c|c|} \hline -3 & -10 & -3 \\ \hline 0 & 0 & 0 \\ \hline 3 & 10 & 3 \\ \hline \end{array} \end{matrix} * \begin{matrix} \text{Image} \\ \text{(Input)} \end{matrix} = \begin{matrix} \text{Image} \\ \text{(Output)} \end{matrix}$$


-3	-10	-3
0	0	0
3	10	3

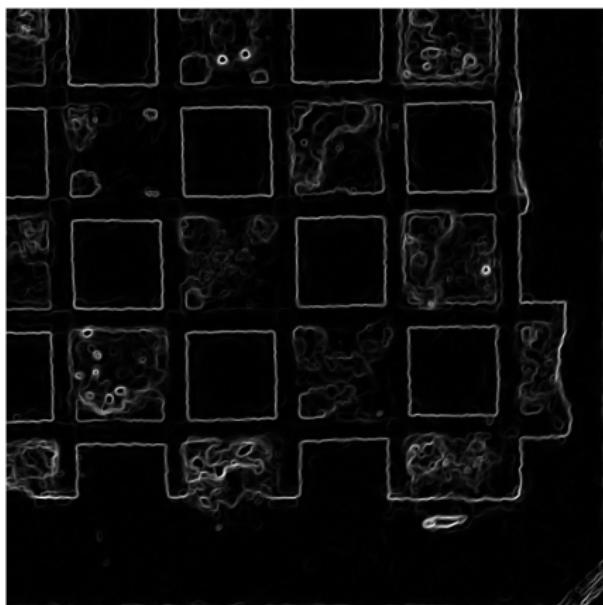
## Edge detection examples

Laplacian



Both negative and positive values

Sobel



Positive values only.

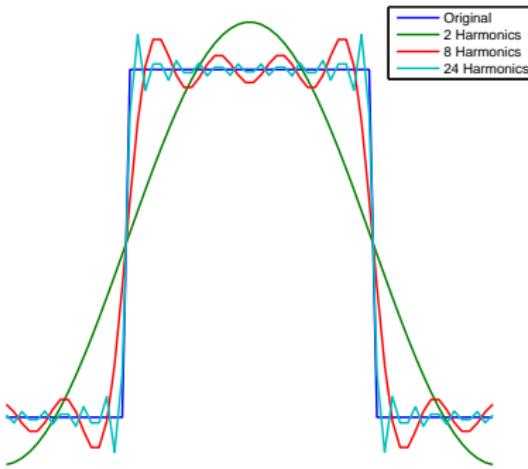
# Filters in frequency space

Applications of the Fourier transform

# What is frequency space?

## Introduction

- A signal can be decomposed into a sum of basic harmonics defined by amplitude, phase shift and frequency.
- Fine details and sharp edges require more harmonics



## Transform

$$G(\xi_1, \xi_2) = \mathcal{F}\{g\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) e^{-i(\xi_1 x + \xi_2 y)} dx dy$$

## It's inverse

$$g(x, y) = \mathcal{F}^{-1}\{G\} = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(\omega) e^{i(\xi_1 x + \xi_2 y)} d\xi_1 d\xi_2$$

## FFT

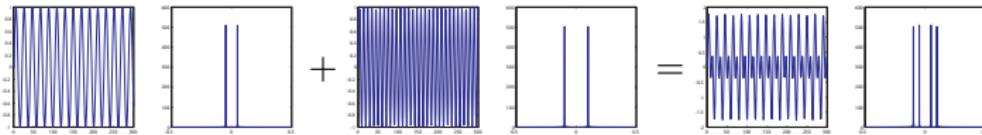
In practice – you never see the transform equations.

The Fast Fourier Transform is available in numerical libraries and tools.

[Jähne, 2005]

## Addition

$$\mathcal{F}\{a + b\} = \mathcal{F}\{a\} + \mathcal{F}\{b\}$$



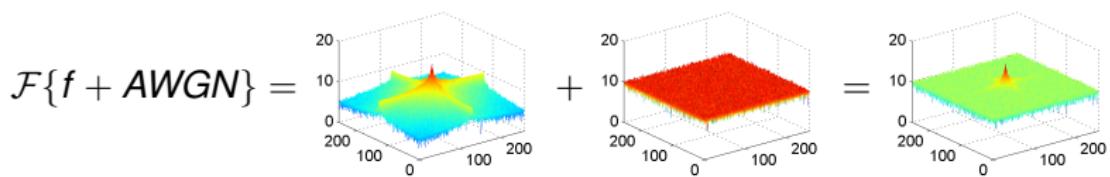
## Convolution

$$\mathcal{F}\{a * b\} = \mathcal{F}\{a\} \cdot \mathcal{F}\{b\}$$

$$\mathcal{F}\{a \cdot b\} = \mathcal{F}\{a\} * \mathcal{F}\{b\}$$

## Additive noise in Fourier space

$$f + AWGN = \text{Image} + \text{Noise} = \text{Result}$$



## Problem

How can we suppress noise without destroying relevant image features?

## Spatial frequencies and orientation

$$0^\circ \quad \mathcal{F} \left\{ \begin{array}{c} \text{Vertical stripes} \\ \text{Orientation: } 0^\circ \end{array} \right\} \Rightarrow \begin{array}{c} \text{Blue square} \\ \text{Low spatial frequency} \end{array}$$

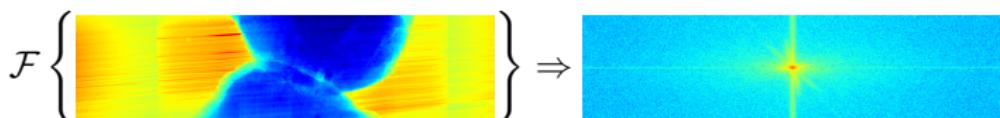
$$90^\circ \quad \mathcal{F} \left\{ \begin{array}{c} \text{Vertical stripes} \\ \text{Orientation: } 90^\circ \end{array} \right\} \Rightarrow \begin{array}{c} \text{Blue square} \\ \text{High spatial frequency} \end{array}$$

$$30^\circ \quad \mathcal{F} \left\{ \begin{array}{c} \text{Diagonal stripes} \\ \text{Orientation: } 30^\circ \end{array} \right\} \Rightarrow \begin{array}{c} \text{Blue square} \\ \text{Medium spatial frequency} \end{array}$$

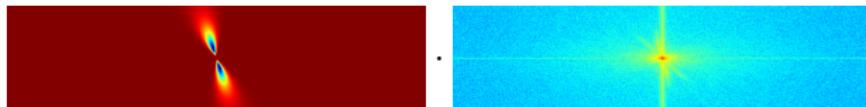
$$60^\circ \quad \mathcal{F} \left\{ \begin{array}{c} \text{Diagonal stripes} \\ \text{Orientation: } 60^\circ \end{array} \right\} \Rightarrow \begin{array}{c} \text{Blue square} \\ \text{Medium spatial frequency} \end{array}$$

## Example – Stripe removal in Fourier space

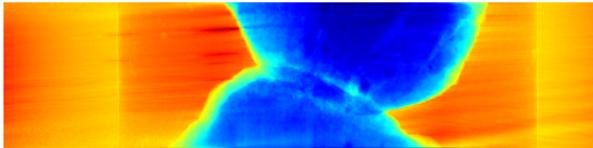
- 1 Transform the image to Fourier space



- 2 Multiply spectrum image by band pass filter

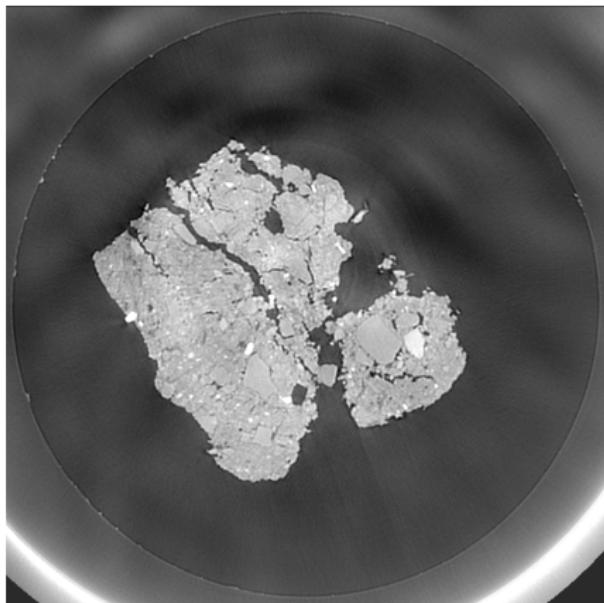


- 3 Compute the inverse transform to obtain the filtered image in real space

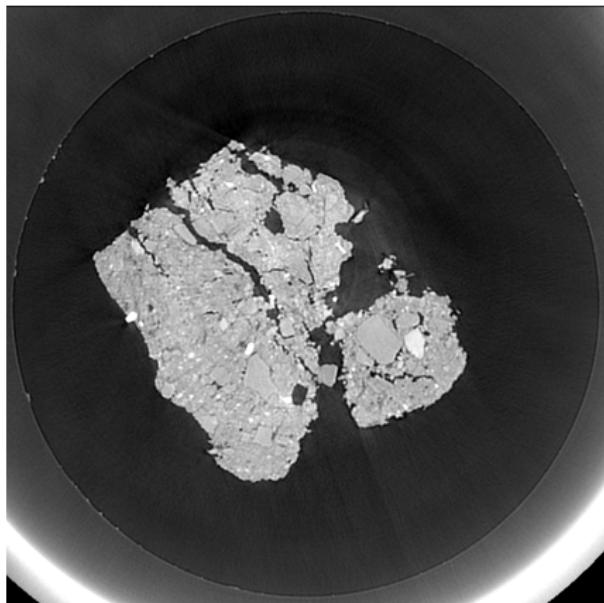


# The effect of the stripe filter

Reconstructed CT slice before filter



Reconstructed CT slice after filter



Intensity variations are suppressed using the stripe filter on all projections.

## Basic filters

Gauss `scikit.ndimage.filters.gaussian_filter(img,s)`

Box `scikit.ndimage.filters.uniform_filter(img,N)`

median `scikit.ndimage.filters.median_filter(img,3)`

general `scikit.ndimage.filters.convolve(img,[[1,2,1],[2,4,2],[1,2,1]])`

`np.fft.fft2(f), np.fft.ifft2(F)`

Computes the Fast Fourier Transform and its inverse of image  $f$ .

`np.abs(f), np.angle(f)`

Computes amplitude and argument of a complex number.

`f.real, f.imag`

Gives the real and imaginary parts of a complex number.

# Scale spaces

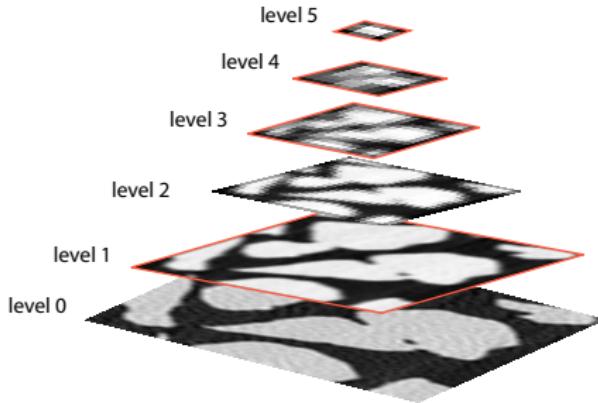
# Why scale spaces?

## Motivation

Basic filters have problems to handle low SNR and textured noise.  
Something new is required...

## The solution

Filtering on different scales can take noise suppression one step further.



# Wavelets – the basic idea

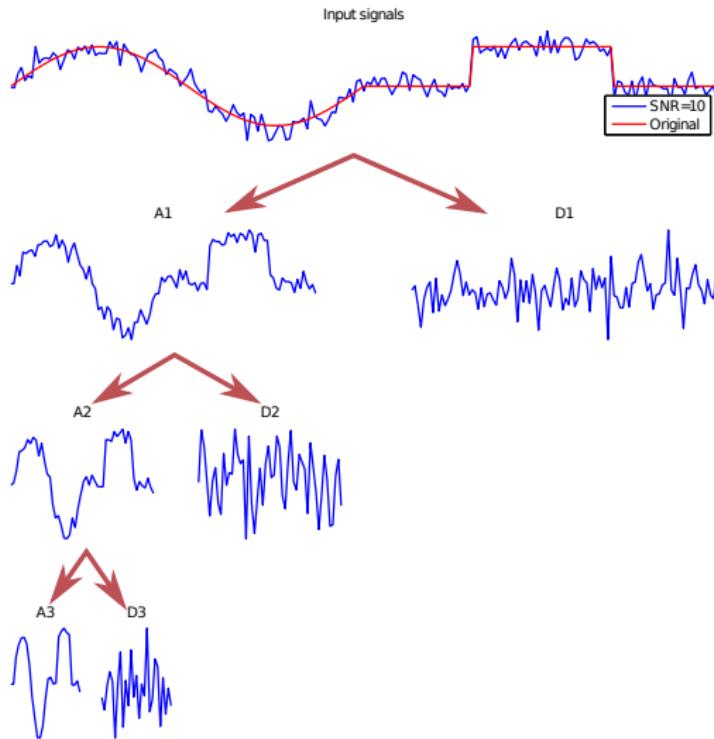
- The wavelet transform produces scales by decomposing a signal into two signals at a coarser scale containing *trend* and *details*.
- The next scale is computed using the trend of the previous transform

$$WT\{s\} \rightarrow \{a_1, d_1\}, WT\{a_1\} \rightarrow \{a_2, d_2\}, \dots, WT\{a_{N-1}\} \rightarrow \{a_N, d_N\}$$

- The inverse transform brings  $s$  back using  $\{a_N, d_1, \dots, d_N\}$ .
- Many wavelet bases exists, the choice depends on the application.
- Wavelets can have several uses:
  - Noise reduction
  - Analysis
  - Segmentation
  - Compression

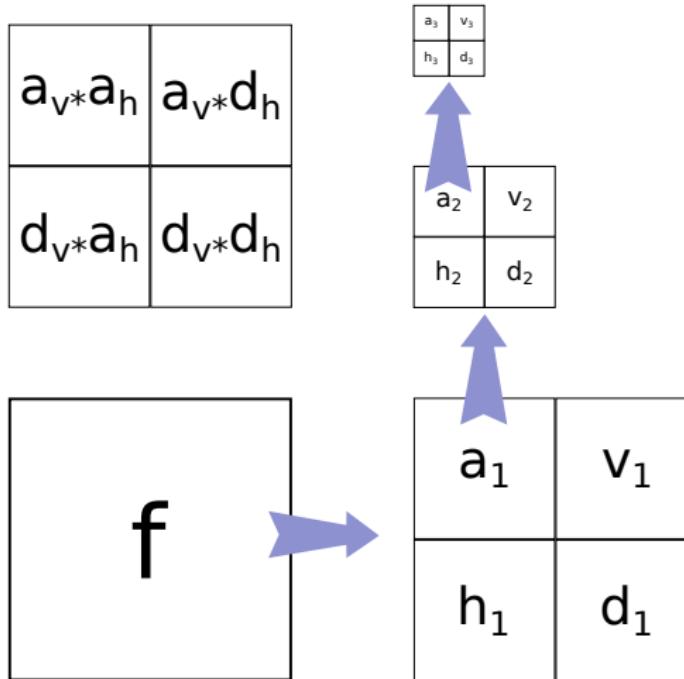
[J.C.Walker, 1999][Mallat, 2009]

# Wavelet transform of a 1D signal



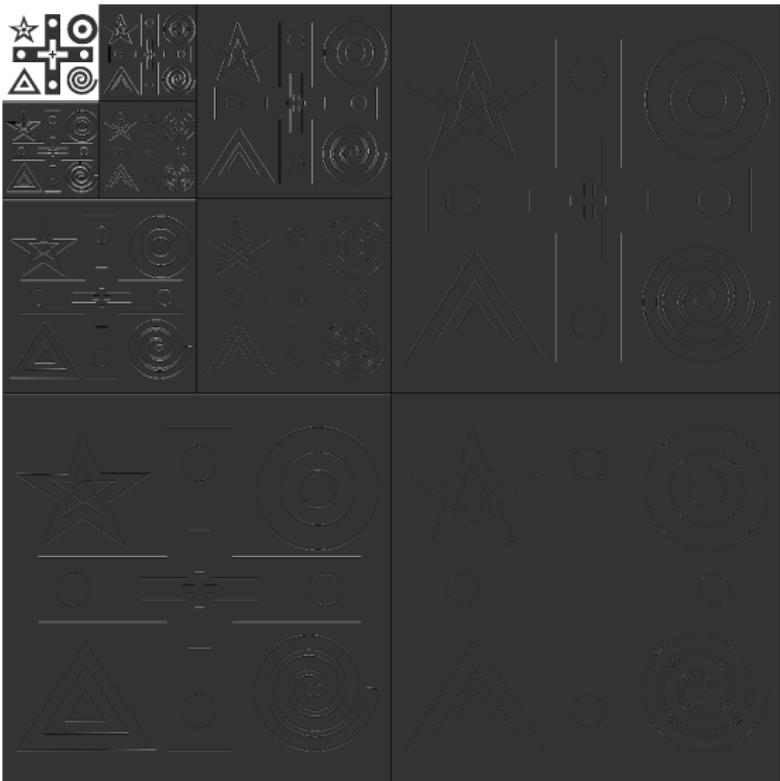
Using *symlet-4*

## Wavelet transform of an image



## Wavelet transform of an image – example

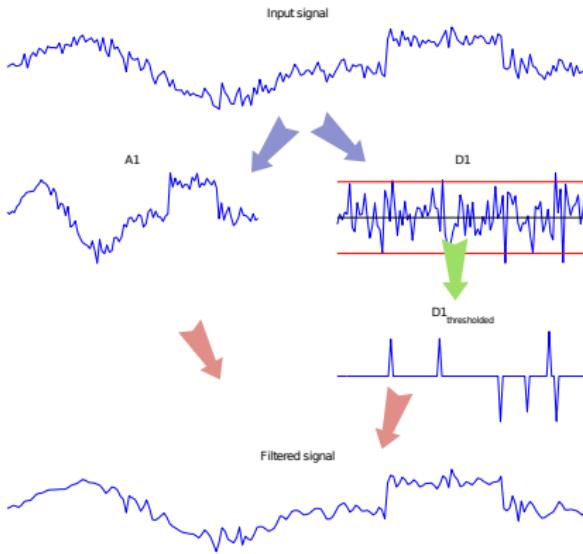
$$WT \left\{ \begin{array}{c} \star \\ \square \\ + \\ \triangle \\ \circlearrowright \end{array} \right\} =$$



# Wavelet noise reduction

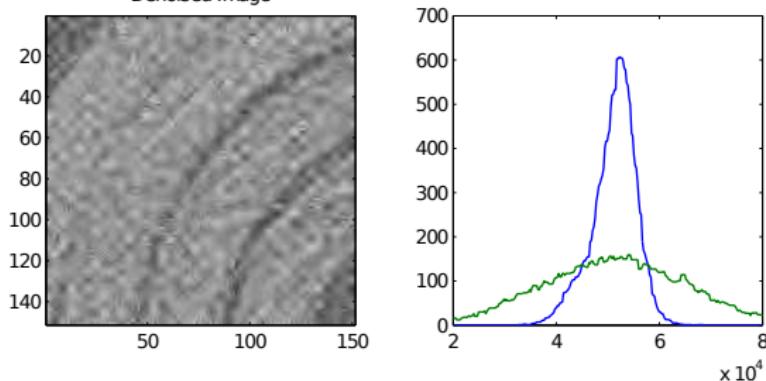
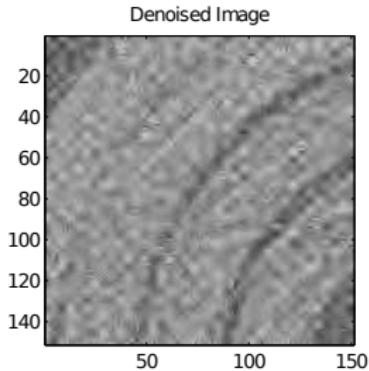
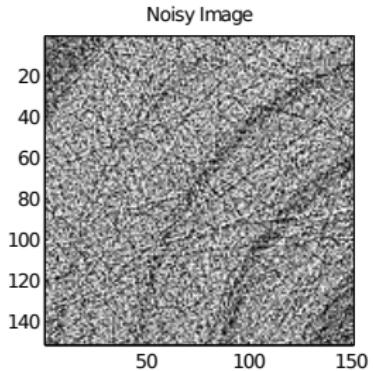
The noise is found in the detail part of the WT

- Make a WT of the signal to a level that corresponds to the scale of the unwanted information.
- Threshold the detail part  $d_\gamma = |d| < \gamma ? 0 : d$ .
- Inverse WT back to normal scale → image is filtered.



# Wavelet noise reduction – Image example

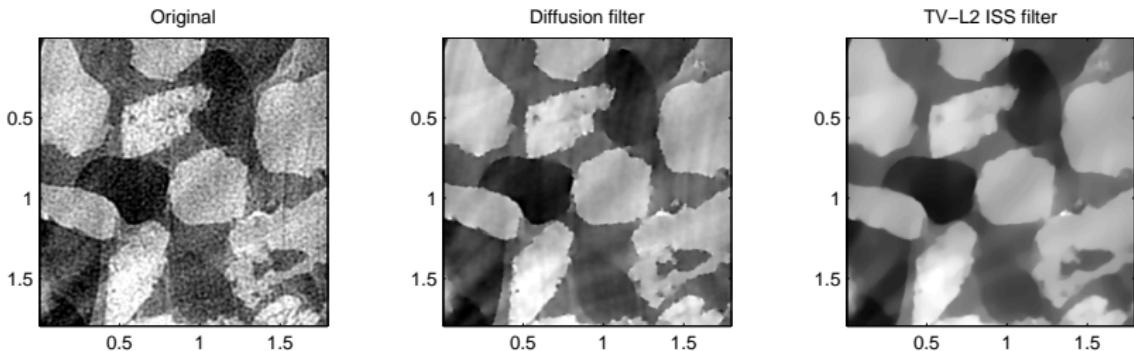
Filtered using two levels of Symlet-2 wavelet



Neutron CT of a lead scroll.

# PDE based scale space filters

Filters small features faster than larger ones.



May work for applications where Linear and Rank filters fail.

[Aubert and Kornprobst, 2002].

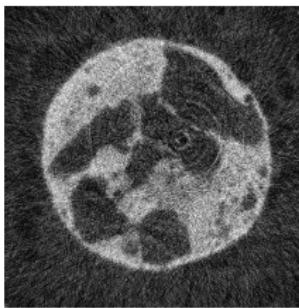
# The starting point

The heat transport equation

$$\frac{\partial T}{\partial t} = \kappa \nabla^2 T$$

$T$  Image to filter (intensity  $\equiv$  temperature)

$\kappa$  Thermal conduction capacity



Original

Intensity diffusion

The steady state solution is a homogeneous image...

# Controlling the diffusivity

We want to control the diffusion process...

Near edges The Diffusivity  $\rightarrow 0$

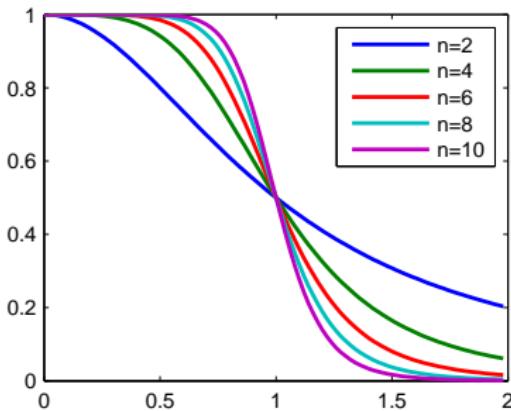
Flat regions The Diffusivity  $\rightarrow 1$

The contrast function  $G$  is our control function

$$G(x) = \frac{1}{1 + \left(\frac{x}{\lambda}\right)^n}$$

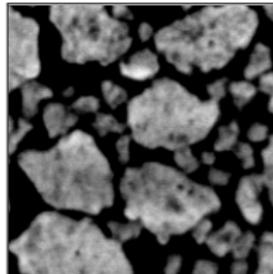
$\lambda$  Threshold level

$n$  Steepness of the threshold

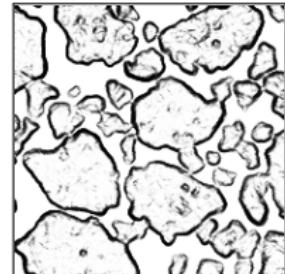


## Gradient controlled diffusivity

$$\frac{\partial u}{\partial t} = G(|\nabla u|) \nabla^2 u$$



Image



Diffusivity map

*u* Image to be filtered

*G(·)* Non-linear function to control the diffusivity

*τ* Time increment

*N* Number of iterations

This filter is noise sensitive!

A more robust filter is obtained with

$$\frac{\partial u}{\partial t} = G(|\nabla_\sigma u|) \nabla^2 u \quad (4)$$

$u$  Image to be filtered

$G(\cdot)$  Non-linear function to control the contrast

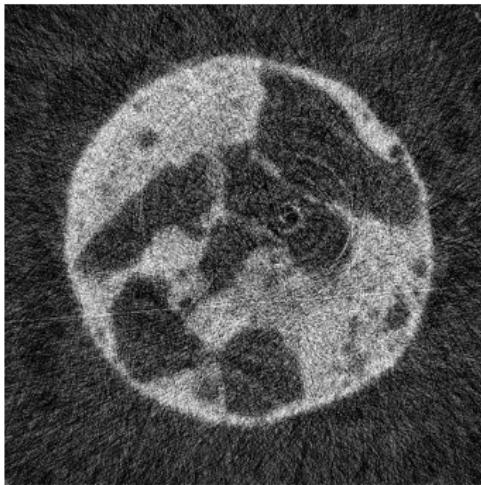
$\tau$  Time increment per numerical iteration

$N$  Number of iterations

$\nabla_\sigma$  Gradient smoothed by a Gaussian filter, width  $\sigma$

## Diffusion filter example

Neutron CT slice from a real-time experiment observing the coalescence of cold mixed bitumen.



Original

Iterations of non-linear diffusion

# The continued development

- 90's** During the late 90's the diffusion filter was described in terms of a regularization problem.
- 00's** Work toward regularization of total variation minimization.

## TV-L<sup>1</sup>

$$u = \operatorname{argmin}_{u \in BV(\Omega)} \left\{ \underbrace{|u|_{BV}}_{\text{noise}} + \underbrace{\frac{\lambda}{2} \|f - u\|_1}_{\text{fidelity}} \right\}$$

## Rudin-Osher-Fatemi model (ROF)

$$u = \operatorname{argmin}_{u \in BV(\Omega)} \left\{ \underbrace{|u|_{BV}}_{\text{noise}} + \underbrace{\frac{\lambda}{2} \|f - u\|_2^2}_{\text{fidelity}} \right\}$$

with  $|u|_{BV} = \int_{\Omega} |\nabla u|^2$

## The idea

We want smooth regions with sharp edges...

- Turn the processing order of scale space filter upside down
- Start with an empty image
- Add large structures successively until an image with relevant features appears

## The ISS filter – Some properties

- is an edge preserving filter for noise reduction.
- is defined by a partial differential equation.
- has a well defined termination point.

[Burger et al., 2006]

# The ROF filter equation

The image  $f$  is filtered by solving

$$\begin{aligned}\frac{\partial u}{\partial t} &= \operatorname{div} \left( \frac{\nabla u}{|\nabla u|} \right) + \lambda (f - u + v) \\ \frac{\partial v}{\partial t} &= \alpha (f - u)\end{aligned}\tag{5}$$

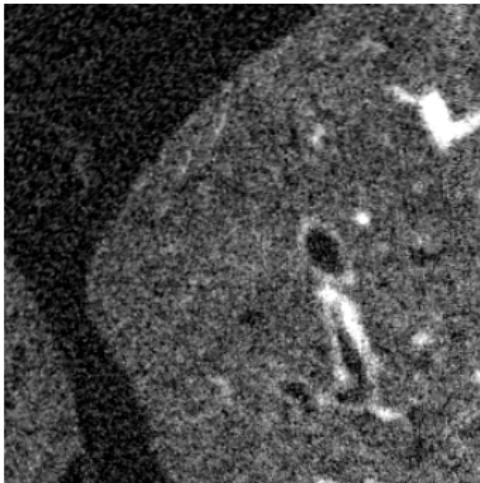
Variables:

- $f$  Input image
- $u$  Filtered image
- $v$  Regularization term (feedback of previous iteration)

Filter parameters

- $\lambda$  Related to the scale of the features to suppress.
- $\alpha$  Quality refinement
- $N$  Number of iterations
- $\tau$  Time increment

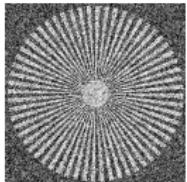
Neutron CT of dried lung filtered using 3D ISS filter



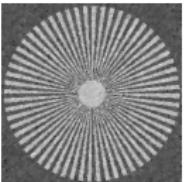
Original

Filter iterations

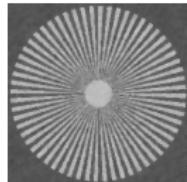
## Solutions at different times



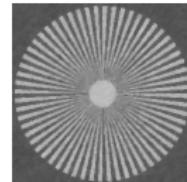
1



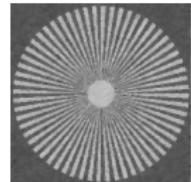
30



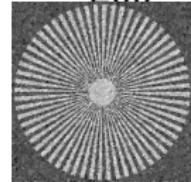
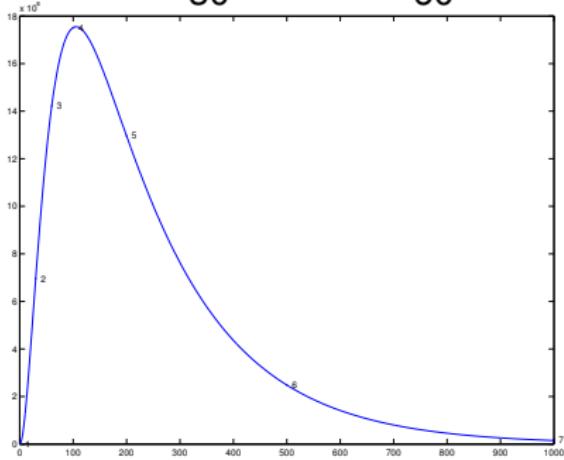
60



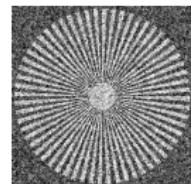
100



200



500



999

## The idea

Smoothing normally consider information from the neighborhood like

- Local averages (convolution)
- Gradients and Curvatures (PDE filters)

Non-local smoothing average similiar intensities in a global sense.

- Every filtered pixel is a weighted average of all pixels.
- Weights computed using difference between pixel intensities.

[Buades et al., 2005]

## Filter definition

The non-local means filter is defined as

$$u(p) = \frac{1}{C(p)} \sum_{q \in \Omega} v(q) f(p, q) \quad (6)$$

where

$v$  and  $u$  input and result images.

$C(p)$  is the sum of all pixel weights as

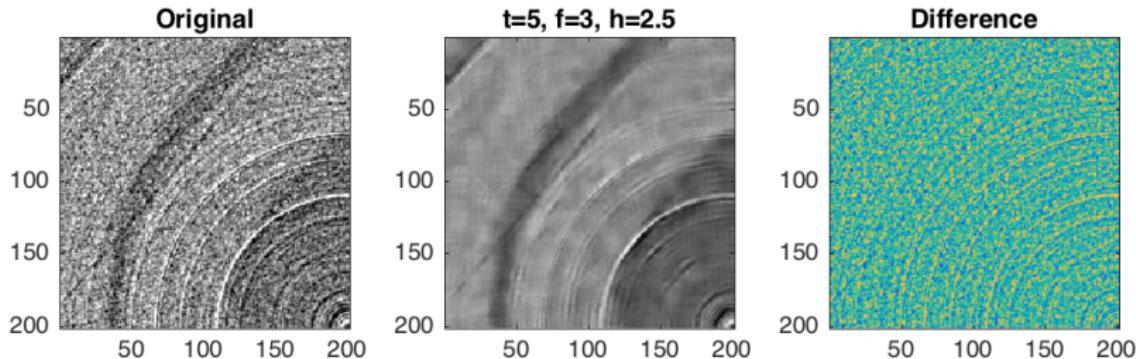
$$C(p) = \sum_{q \in \Omega} f(p, q) \quad (7)$$

$f(p, q)$  is the weighting function

$$f(p, q) = e^{-\frac{|B(q) - B(p)|^2}{h^2}} \quad (8)$$

$B(x)$  is a neighborhood operator e.g. local average around  $x$

## Non-local means 2D – Example



## Observations

- Good smoothing effect.
- Strong thin lines are preserved.
- Some patchiness related to filter parameter  $t$ , i.e. the size of  $\Omega_i$ .

## Problem

The original filter compares all pixels with all pixels...

- Complexity  $\mathcal{O}(N^2)$
- Not feasible for large images, and particular 3D images!

## Solution

It has been shown that not all pixels have to be compared to achieve a good filter effect.

i.e.  $\Omega$  in eq 6 and 7 can be replaced by  $\Omega_i \ll \Omega$

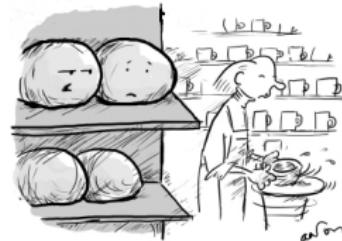
# Verification

# Verify the correctness of the method

## "Data massage"

Filtering manipulates the data...

... avoid too strong modifications otherwise  
you may invent new image features!!!



Watch that man, he'll make mugs of us all!

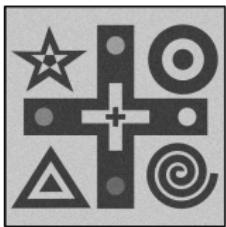
## Verify the validity your method

- Visual inspection
- Difference images
- Use degraded phantom images in a "smoke test"

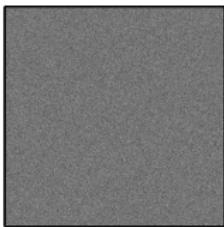
# Verification using difference images

Compute pixel-wise difference between image  $f$  and  $g$

Noisy image



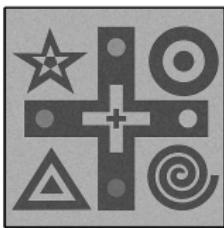
Ideal filter



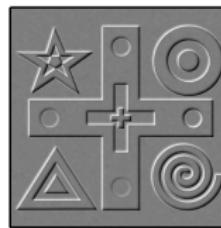
Over smoothing



Intensity scaling



Geometric shift



Difference images provide first diagnosis about processing performance

## Performance testing – The smoke test

- Testing term from electronic hardware testing – drive the system until something fails due to overheating...
- In general: scan the parameter space for different SNR until the method fails to identify strength and weakness of the system.

### Test strategy

- 1 Create a phantom image with relevant features.
- 2 Add noise for different SNR to the phantom.
- 3 Apply the processing method with different parameters.
- 4 Measure the difference between processed and phantom.
- 5 Repeat steps 2-4  $N$  times for better test statistics.
- 6 Plot the results and identify the range of SNR and parameters that produce acceptable results.

An evaluation procedure need a metric to compare the performance

## Mean squared error

$$MSE(f, g) = \sum_{p \in \Omega} (f(p) - g(p))^2$$

## Structural similarity index

$$SSIM(f, g) = \frac{(2\mu_f \mu_g + C_1)(2\sigma_{fg} + C_2)}{(\mu_f^2 + \mu_g^2 + C_1)(\sigma_f^2 + \sigma_g^2 + C_2)}$$

$\mu_f, \mu_g$  Local mean of  $f$  and  $g$ .

$\sigma_{fg}$  Local correlation between  $f$  and  $g$ .

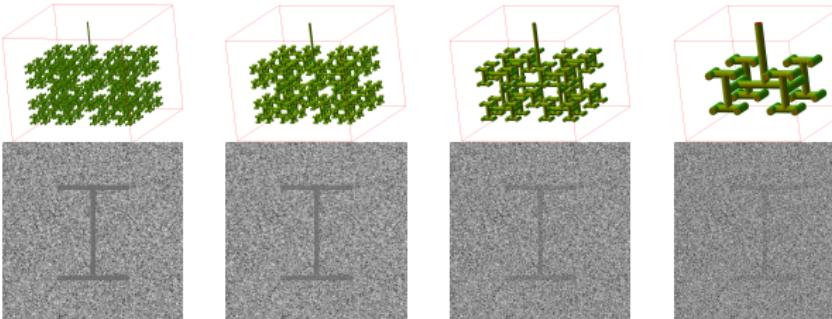
$\sigma_f, \sigma_g$  Local standard deviation of  $f$  and  $g$ .

$C_1, C_2$  Constants based on the image dynamics (small numbers).

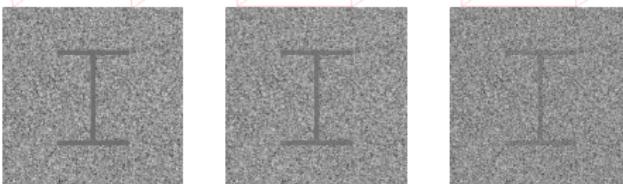
$$MSSIM(f, g) = E[SSIM(f, g)]$$

## Test run example

Phantom structures

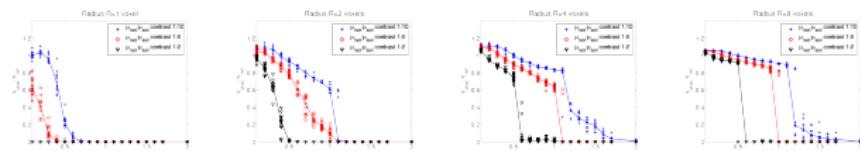


Add noise



Process

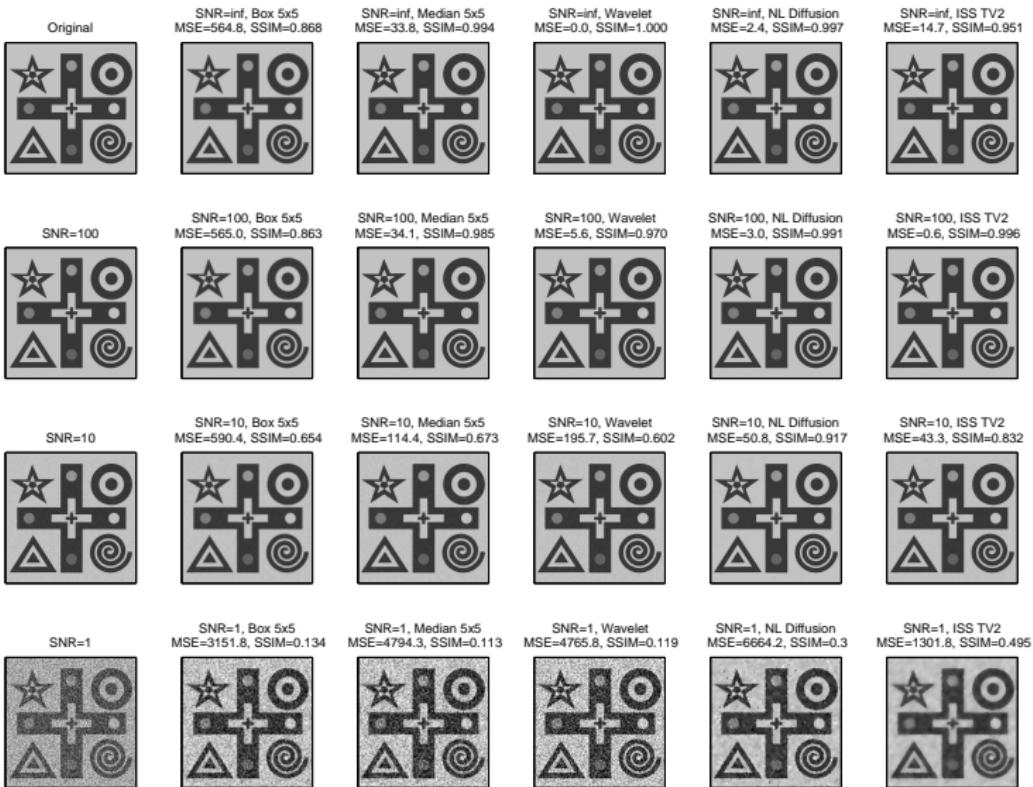
Plot results



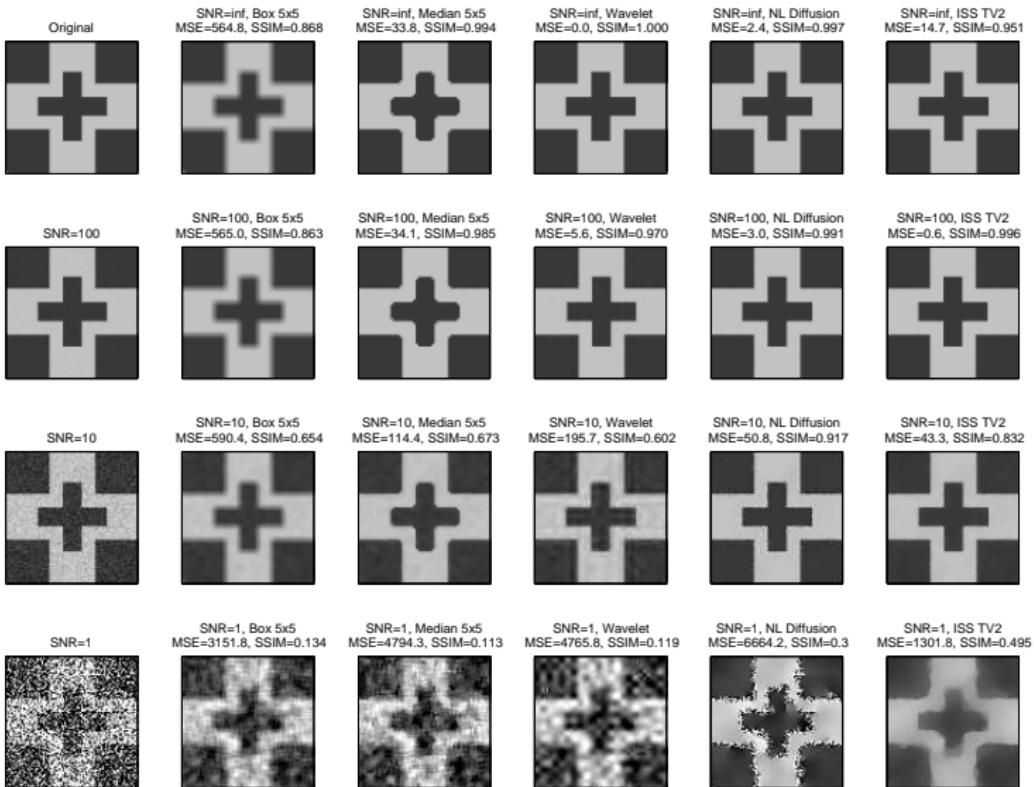
[Kaestner et al., 2006]

# Summary

# Many filters



# Details of filter performance



## Take-home message

We have looked at different ways to suppress noise and artifacts:

- Convolution
- Median filters
- Wavelet denoising
- PDE filters

Which one you select depends on

- Purpose of the data
- Quality requirements
- Available time

### Remember

A good measurement is better than an enhanced bad measurement...  
...but bad data can mostly be rescued if needed.

## Exercise links

- The exercises will be done using python and some packages.
- We will use Jupyter in the Anaconda environment.
  
- Install and start the python environment.
- Exercises 1-3
- Exercise 4

# References I



Aubert, G. and Kornprobst, P. (2002).

*Mathematical problems in image processing.*

Number 147 in Applied mathematical sciences. Springer Verlag.



Buades, A., Coll, B., and Morel, J. (2005).

A non-local algorithm for image denoising.

In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*.



Burger, M., Gilboa, G., Osher, S., and Xu, J. (2006).

Nonlinear inverse scale space methods.

*Communications in Mathematical Sciences*, 4(1):179–212.



Jähne, B. (2005).

*Digital Image Processing.*

Springer Verlag, 6 edition.



J.C. Walker (1999).

*A primer on Wavelets and their scientific applications.*

CRC Press.



Kaestner, A., Schneebeli, M., and Graf, F. (2006).

Visualizing three-dimensional root networks using computed tomography.

*Geoderma*, 136(1–2):459–469.



Mallat, S. (2009).

*A wavelet tour of signal processing: The sparse way.*

Academic Press.



Wang, Z. and Bovik, A. (2009).

Mean squared error: Love it or leave it? – a new look at signal fidelity measures.

*IEEE Signal Processing Magazine*, January:89–117.