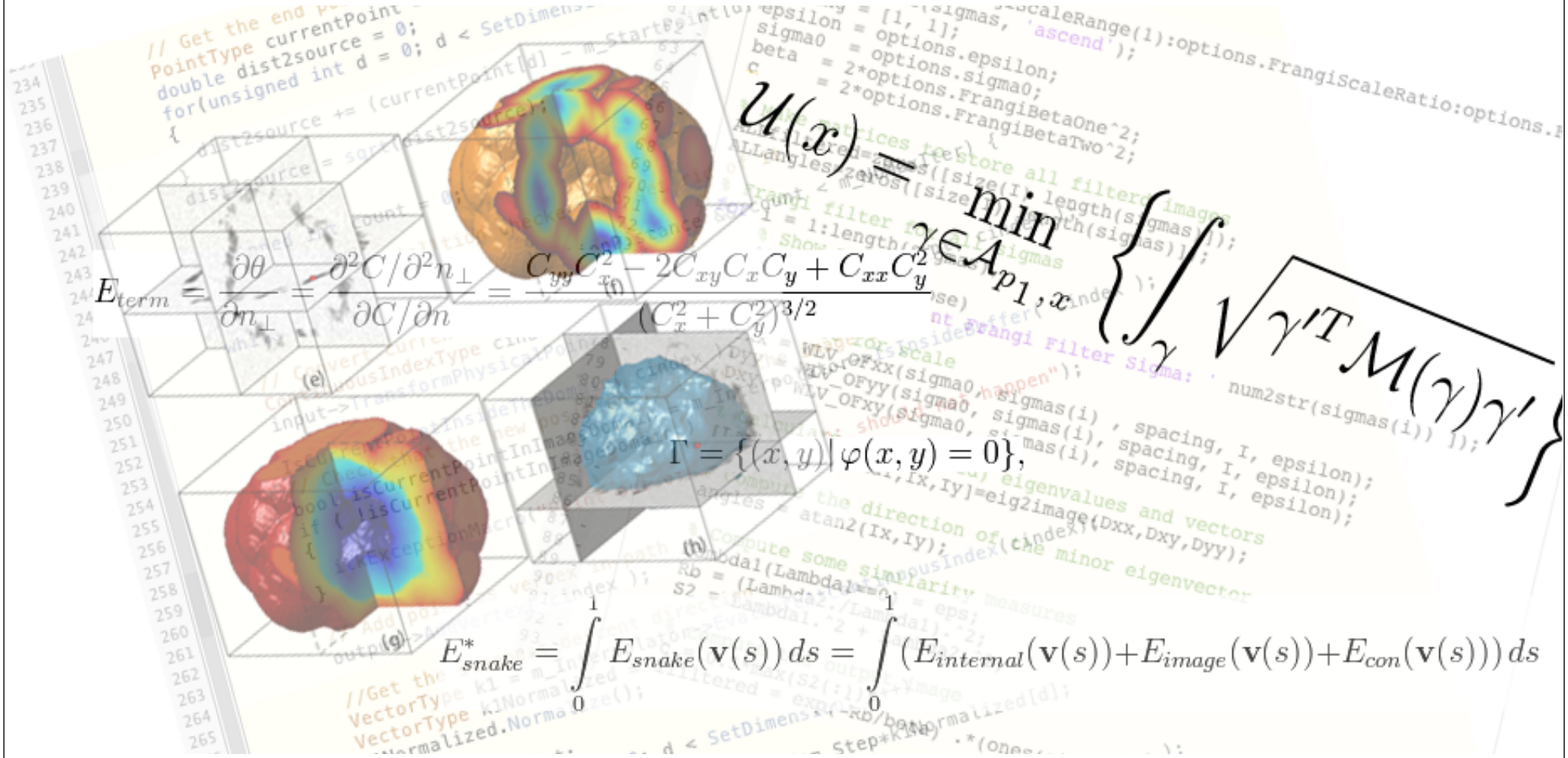
Selected Topics in Computer Vision for Biomedical Images



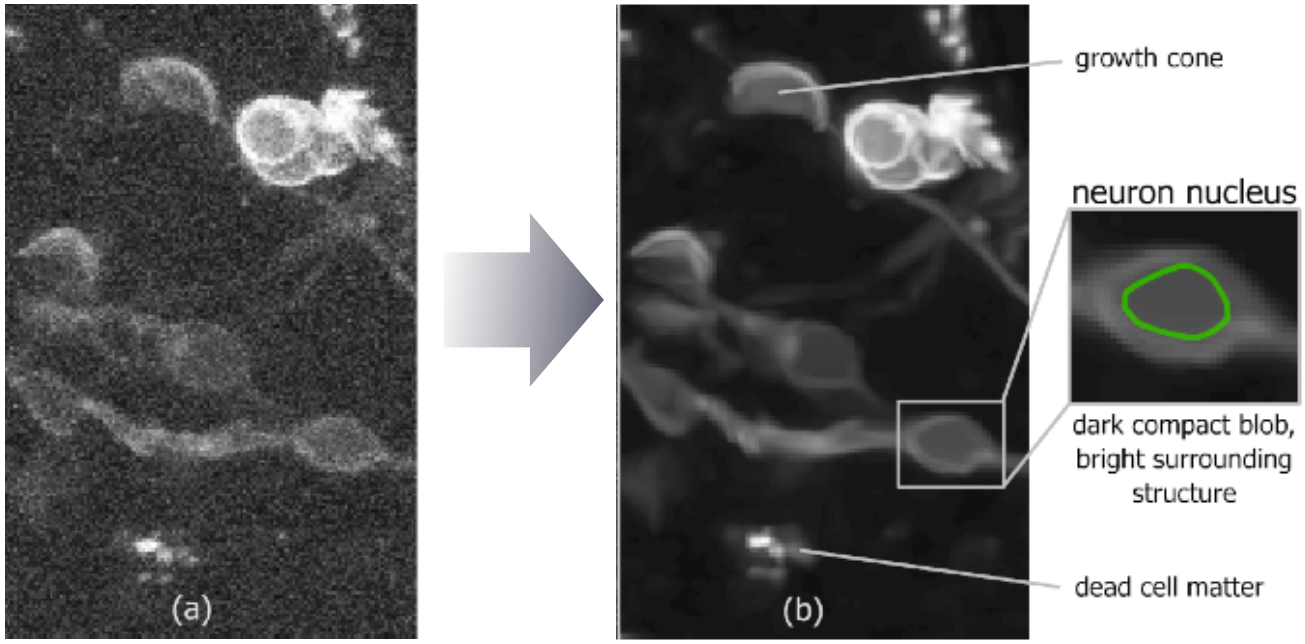
Vincent Lepetit

Class Organization

- Five different teachers, on different topics;
- Homework every week (usually implementation in Matlab, C, …);
- Each of you will study and present a paper;
- Evaluation will be based on the homeworks and the presentations.

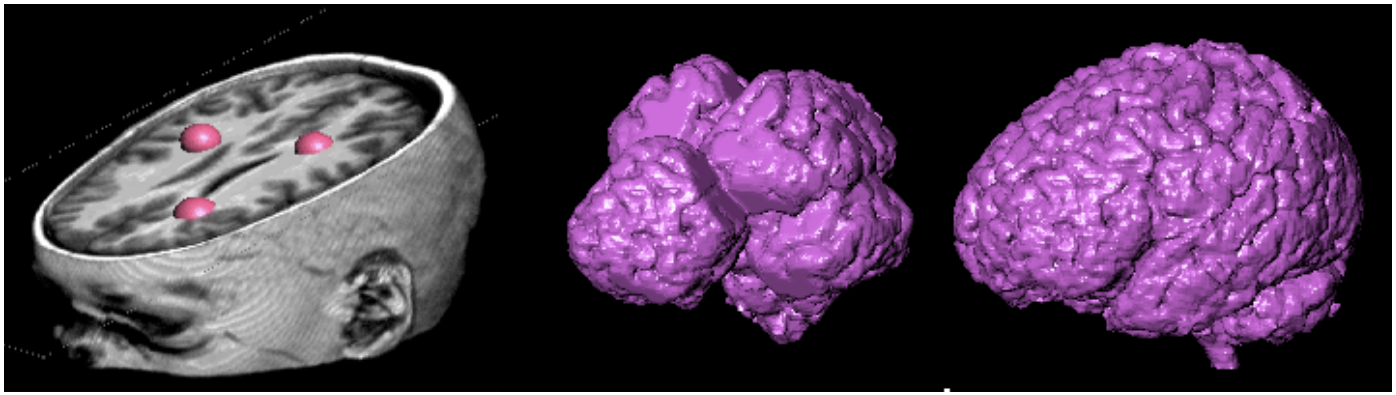
Lectures

- Image denoising (2 lectures, Dr Vincent Lepetit):

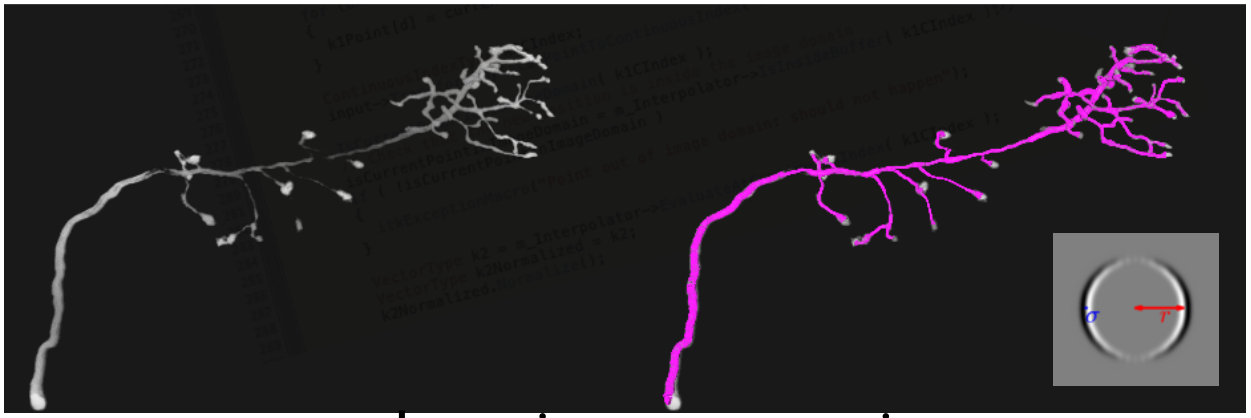


Lectures

- Image segmentation: Global methods, region-based methods, contour-based methods (2 lectures, Dr Fethallah Benmansour):



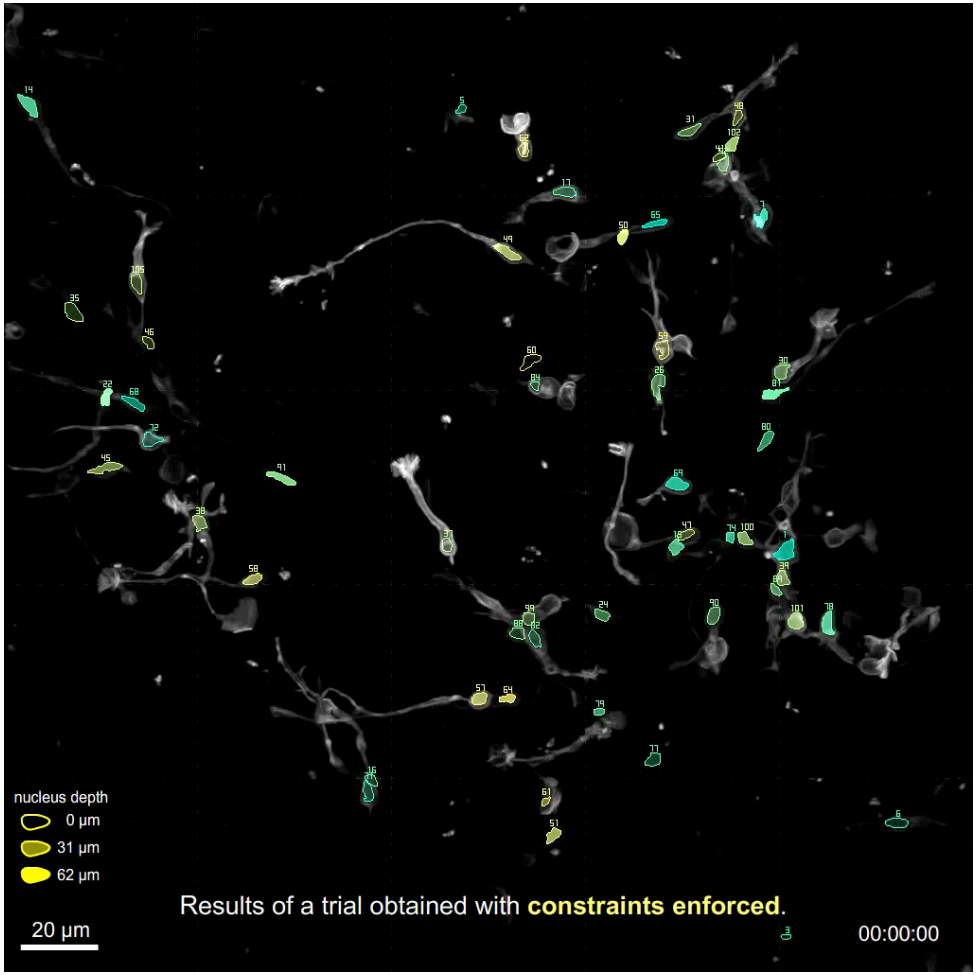
Brain segmentation with level sets



Learning segmentation operators

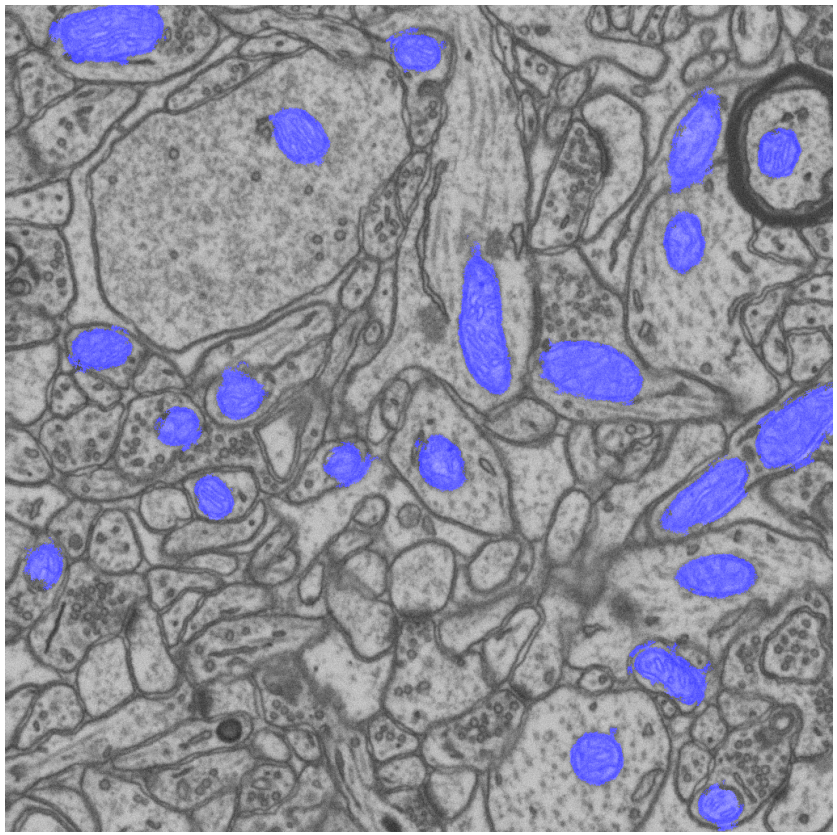
Lectures

- 2D tracking: Recursive Bayesian filtering, Batch probabilistic methods, non-probabilistic methods (2 lectures, Dr Kevin Smith):



Lectures

- Inference on graphical models: Belief Propagation and Graph-cuts (1 lecture, Dr Yunpeng Li):



Mitochondria segmentation with graph-cuts

Lectures

- Data clustering: Expectation-Minimization and Mean-shift (1 lecture, Dr Mario Christoudias):

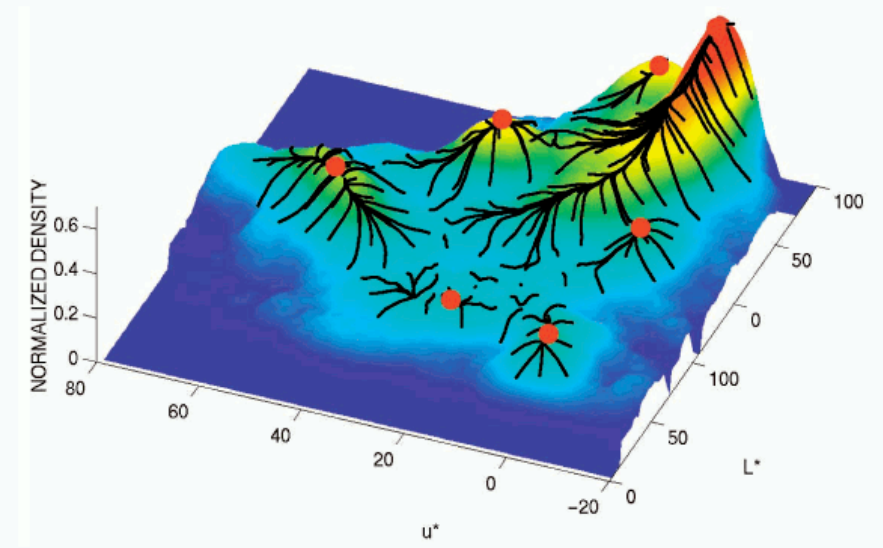
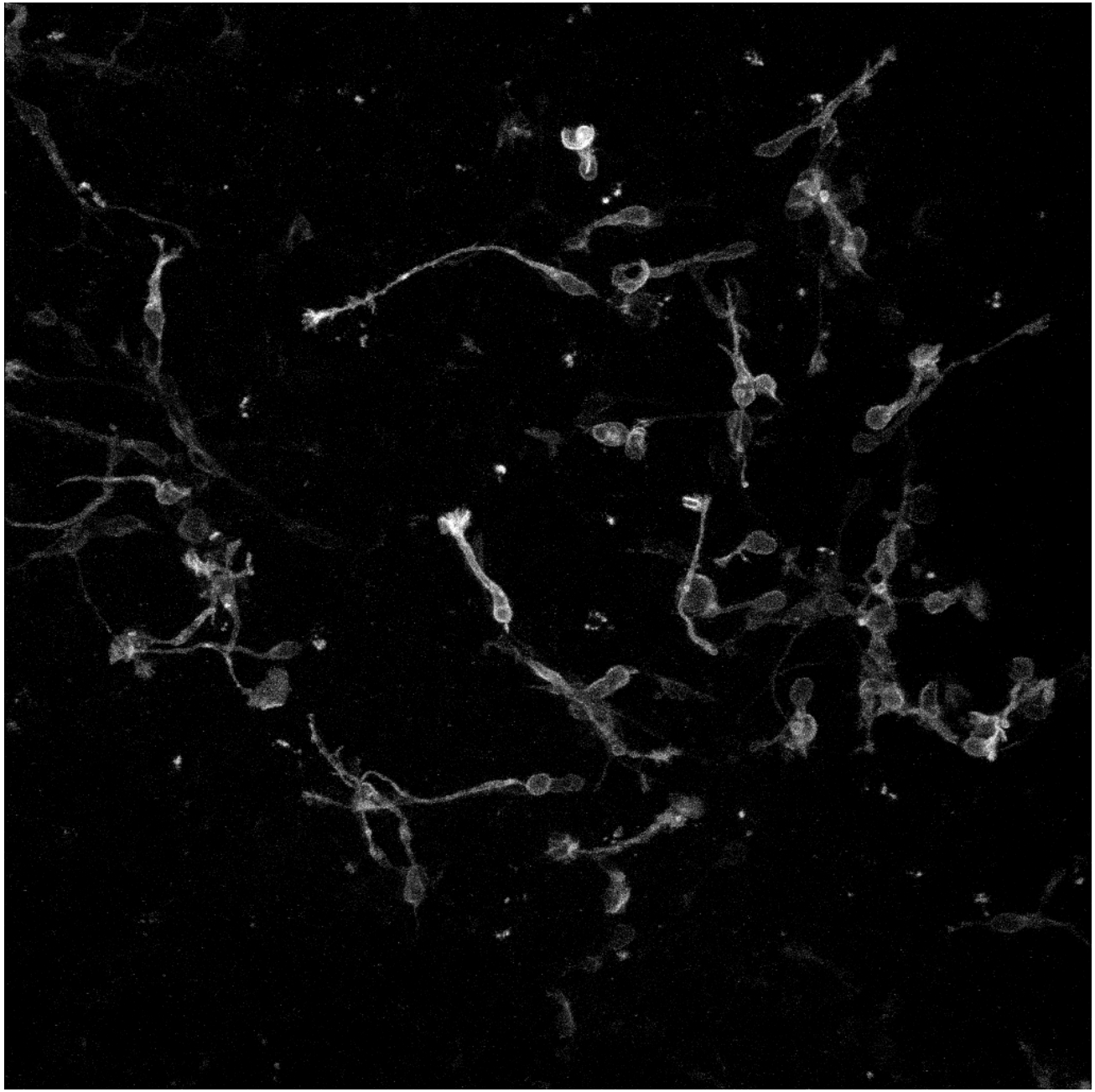
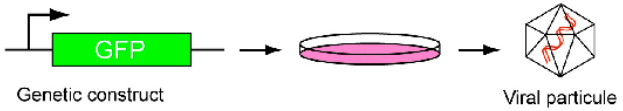
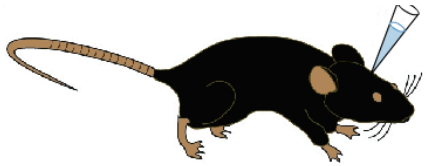


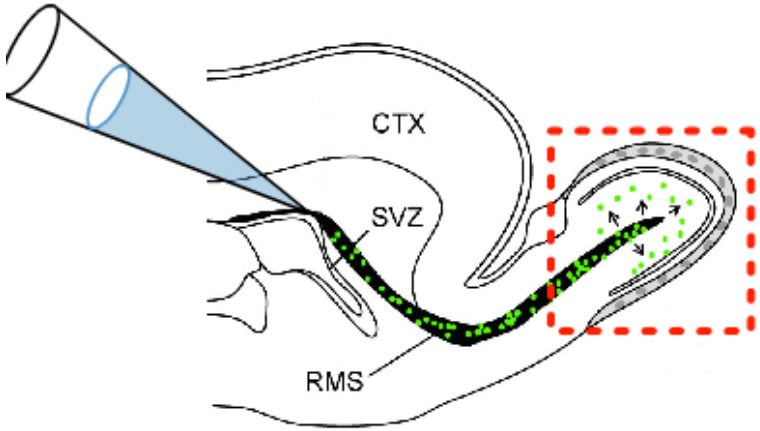
Image Denoising: introduction

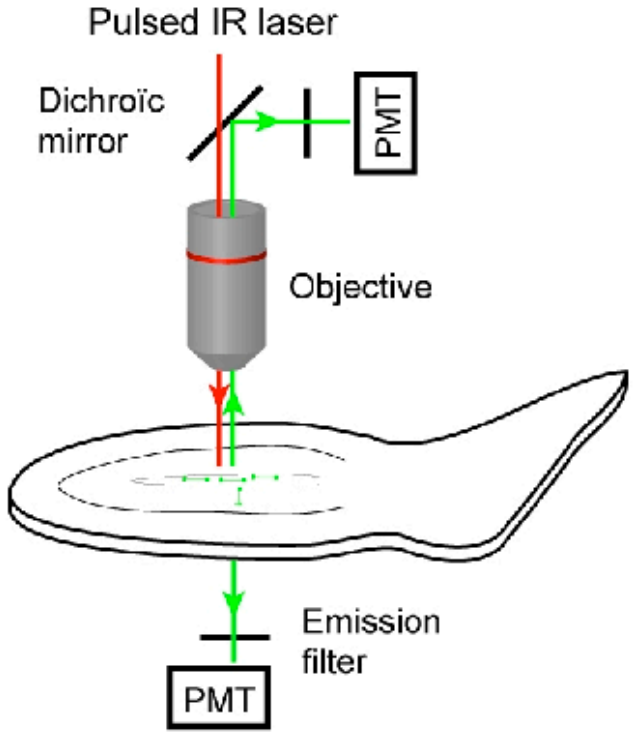


capture



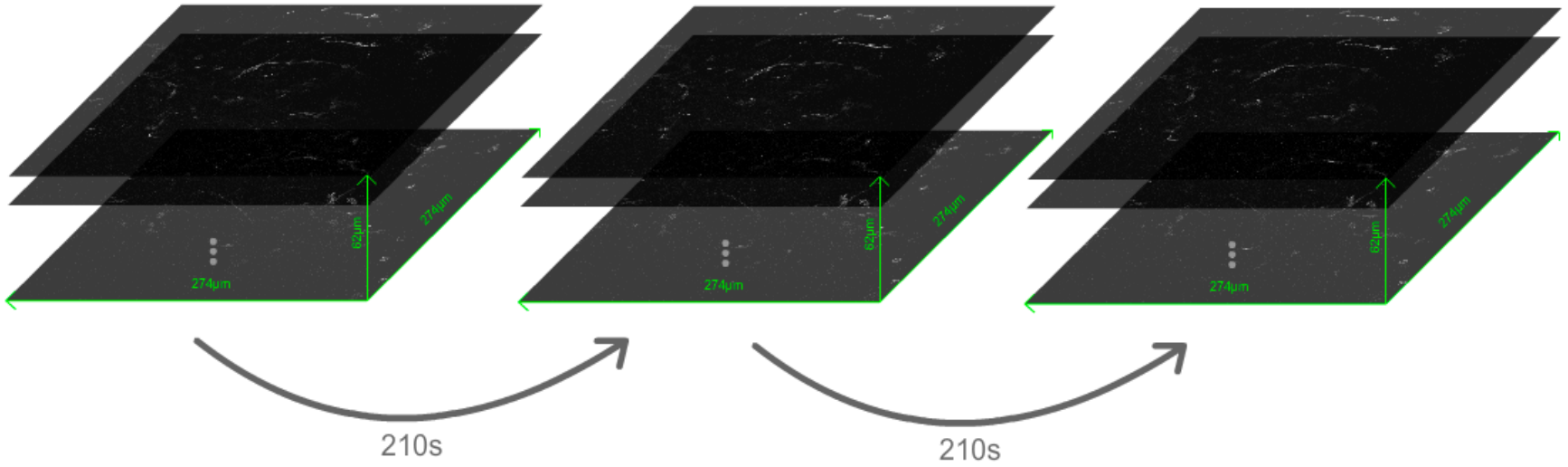


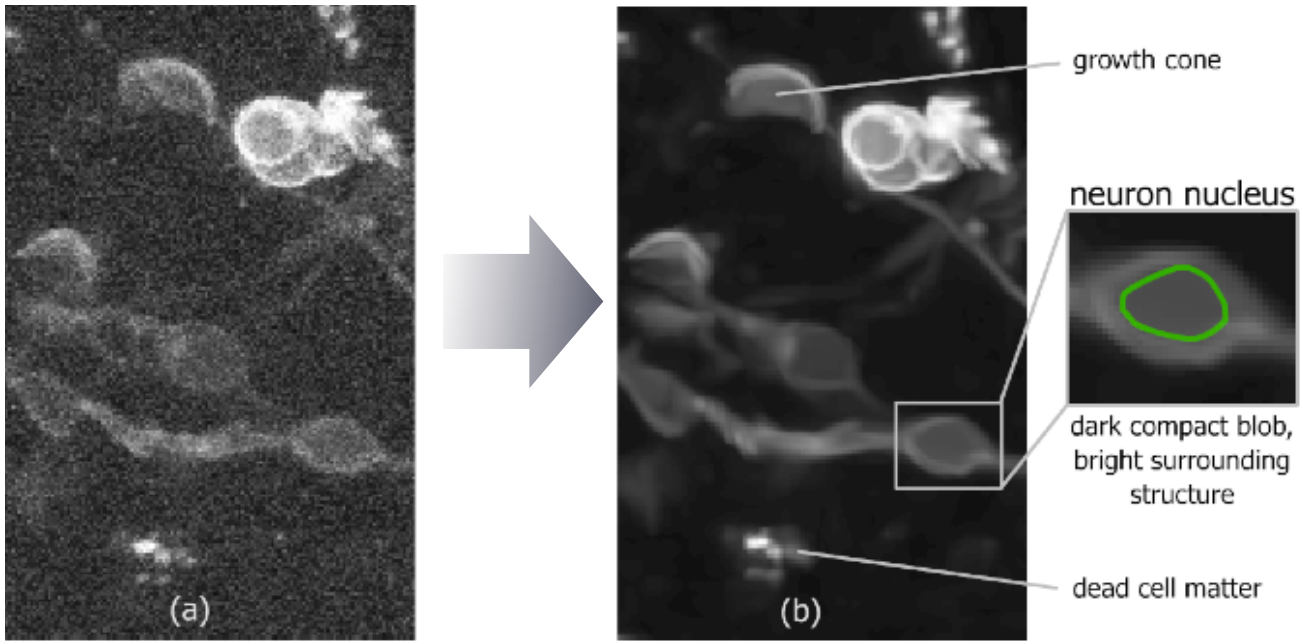


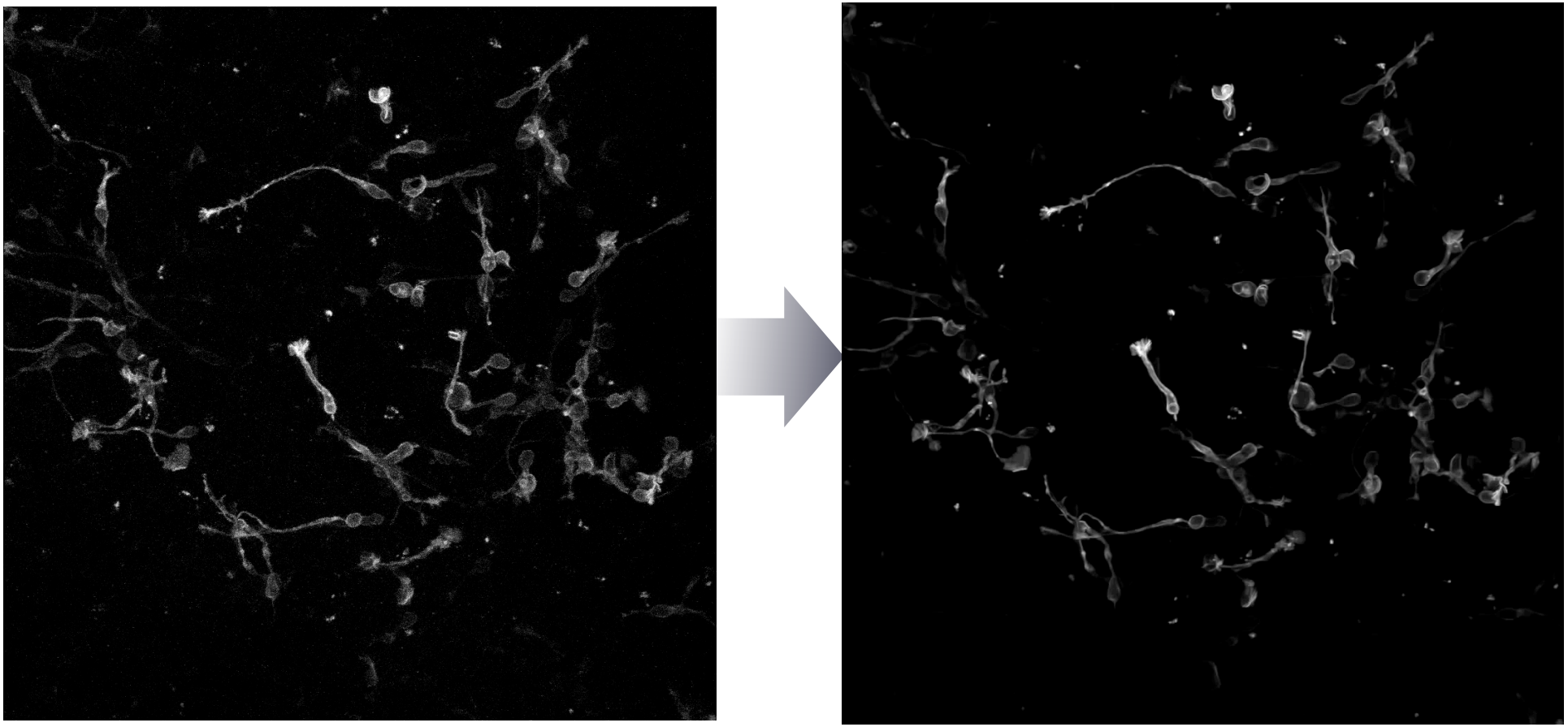


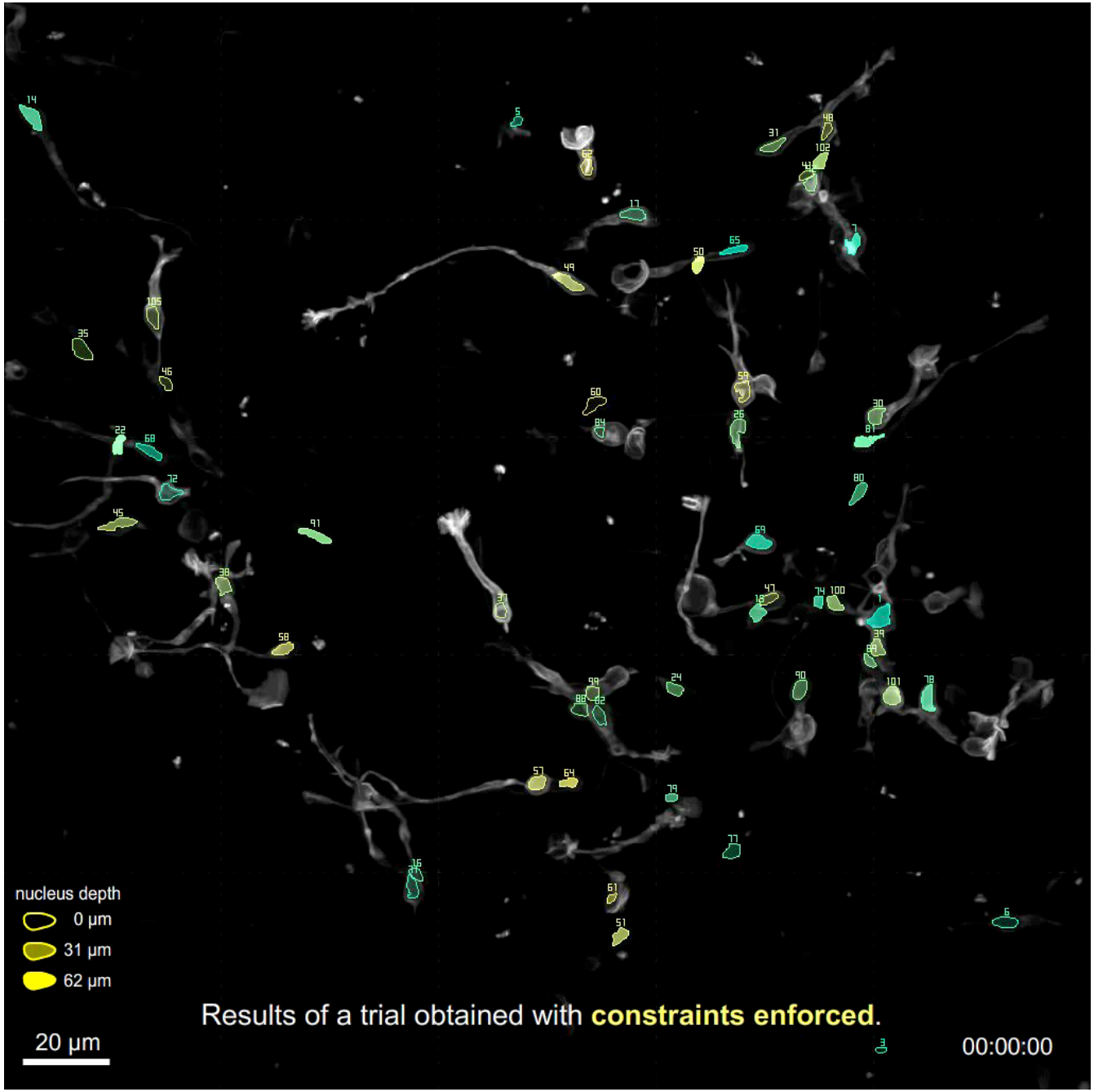
2-photon microscope

capture

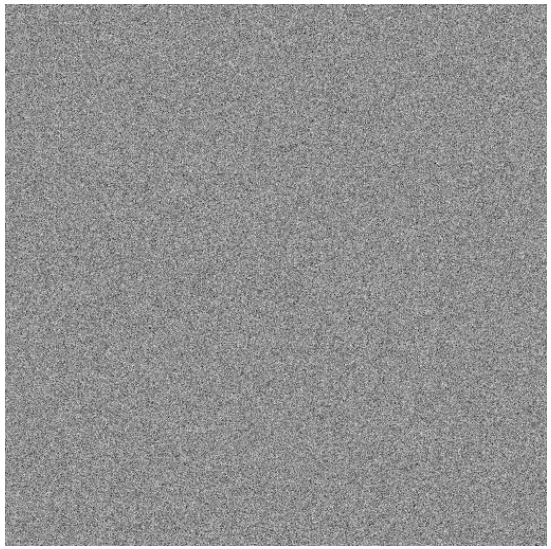

274µm
62µm
274µm
274µm
62µm
274µm
274µm
62µm
274µm
210s
210s


growth cone
neuron nucleus
dark compact blob,
bright surrounding
structure
dead cell matter
(a)
(b)




nucleus depth
0 µm
31 µm
62 µm
20 µm
Results of a trial obtained with constraints enforced.
00:00:00

$$I_{observed} = I_{ideal} + noise$$



Useful for:
making the images look better;
making further processing easier.

warning

Denoising the images is not always a good idea!

- The denoising algorithm can remove valuable data, or add artefacts.
- Denoising may take time.

Humans are very good at handling noise

SNR = 12

SNR = 2

$$\text{SNR} = \text{Signal-Noise Ratio} = \frac{\text{standard deviation(image)}}{\text{standard deviation(noise)}}$$

Image Denoising Methods

- Linear filtering (Gaussian filter);
- Anisotropic filtering;
- Non-linear filtering ([weighted] Median filter, bilateral filter);
- Wiener filtering;
- Anisotropic diffusion;
- Non-local image denoising;
- [Blind] Deconvolution;
- Variational methods.

Evaluation

- Signal-to-Noise Ratio (bad);
- Structural similarity index.

Types of Noise
Additional Noise

Gaussian noise $\sigma = 30$

Uniform noise $\in [-30; +30]$

- Gaussian (white) noise: comes from the electronics of the sensor (thermal agitation of the electrons);
- Quantization noise (uniform noise): caused by quantizing the pixels to a number of discrete levels;
- Pink noise, blue noise, Brownian noise, … : more for audio or electronics signals.

Destructive Noise

30,000 corrupted pixels

- Salt-and-pepper noise, or "impulsive" noise: destroys the signal, comes from dead pixels, conversion to digital signals error, …
- ...

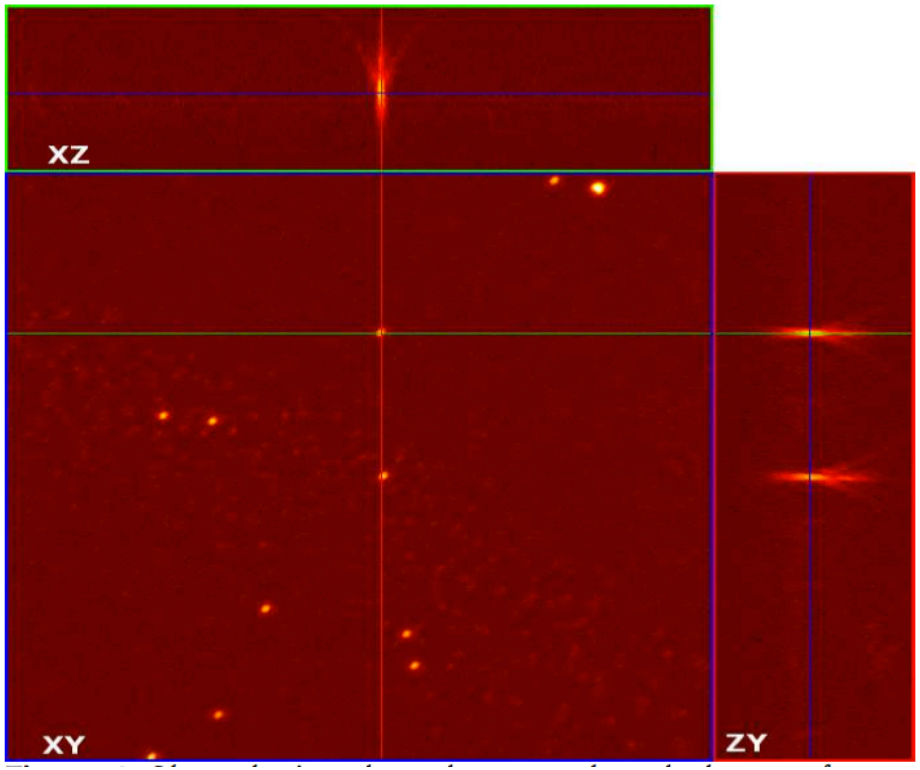
Correlation

uncorrelated noises

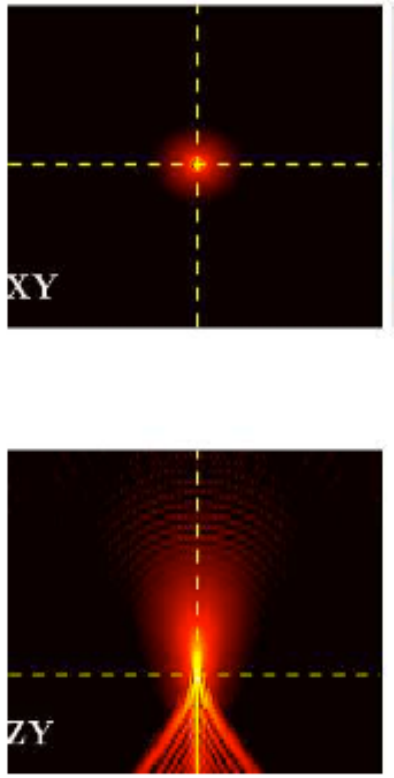
correlated noises

- Noises at different pixels can be either correlated or uncorrelated;
- Often: modeled as being independent and identically distributed, and hence uncorrelated.

Point Spread Function



captured 3D image of microspheres



point spread function (PSF)

$$\text{image} = \text{PSF} * \text{“true image”}$$

Images © Ariana-INRIA/I3S [PANKAJAKSHAN, 2010]

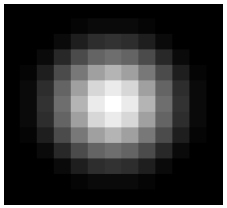
Point Spread Function

image = PSF $*$ “true image”

The true image can be approximated using

- deconvolution if the PSF is known, or
- blind deconvolution if it is not.

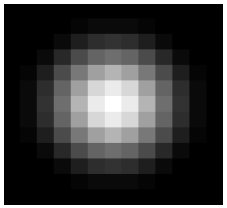
Denoising with a (linear) Gaussian filter



f *  = 

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

Denoising with a (linear) Gaussian filter



f

*

=

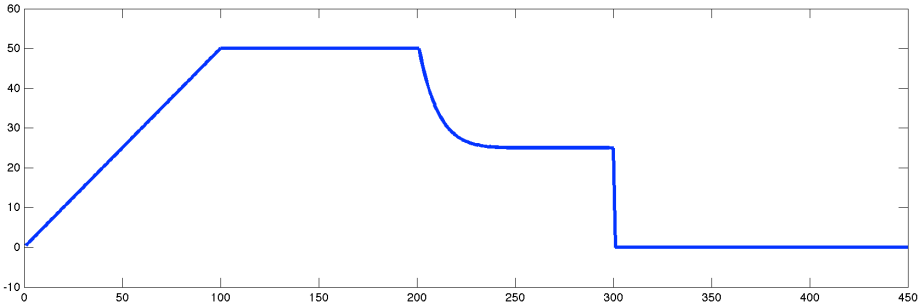
Advantages: simple, fast (separable).

Denoising with a (linear) Gaussian filter

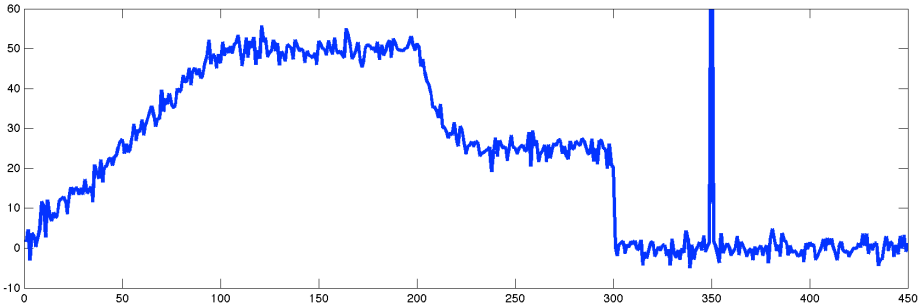
Assumptions:

- uncorrelated (Gaussian) noise with 0 mean ;
- image is (locally) linear.

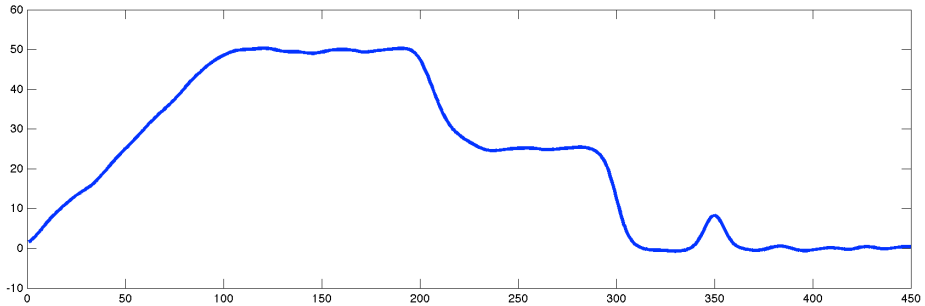
Original



Noisy



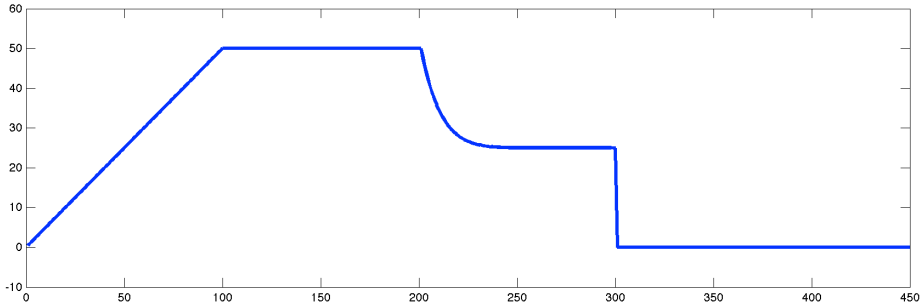
Filtered



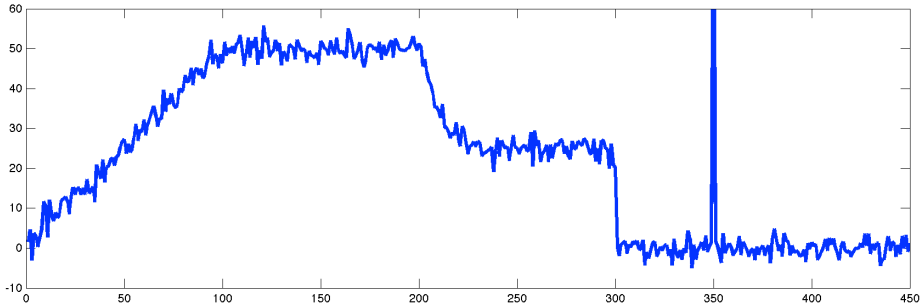
Denoising with a (linear) Gaussian filter

- Does not preserve discontinuities;
- Cannot remove salt-and-pepper noise.

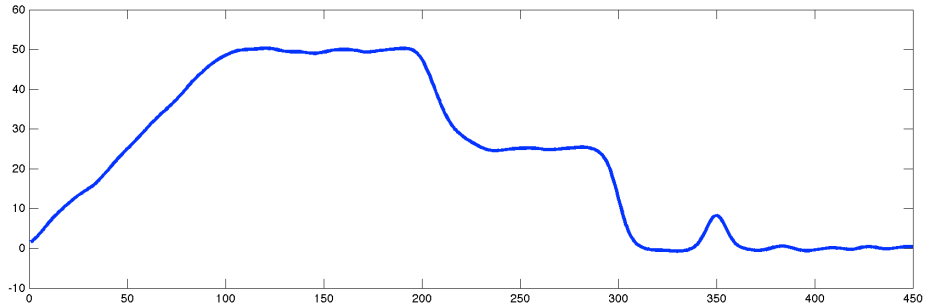
Original



Noisy

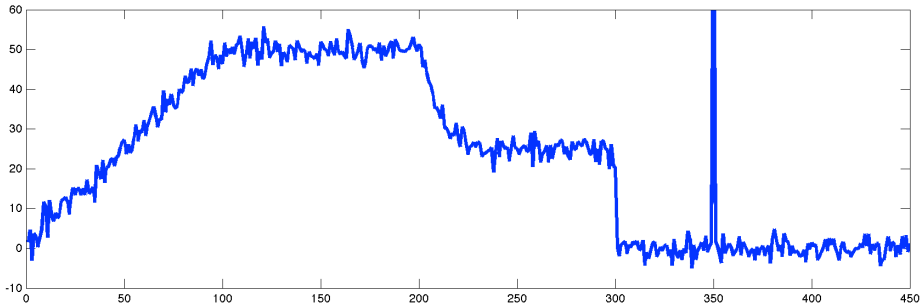


Filtered

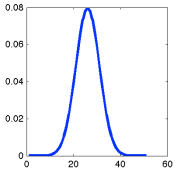


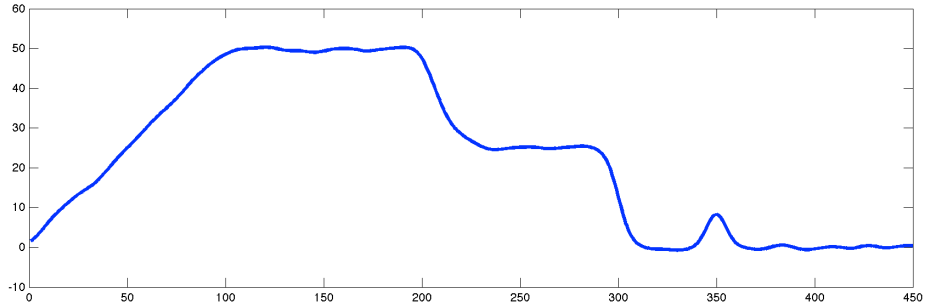
Comparison with a uniform filter

Noisy

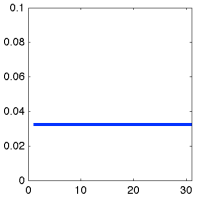


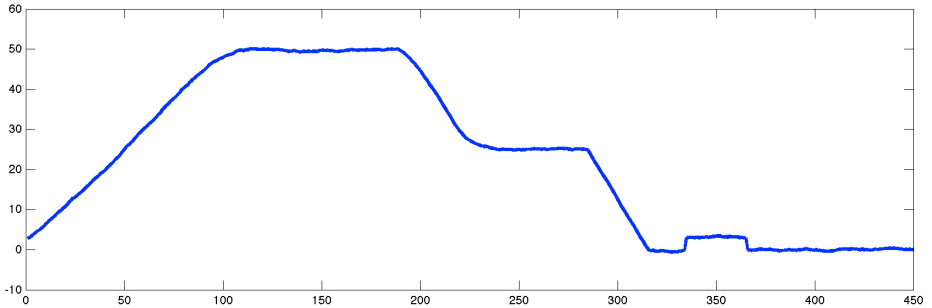
after smoothing with a Gaussian filter



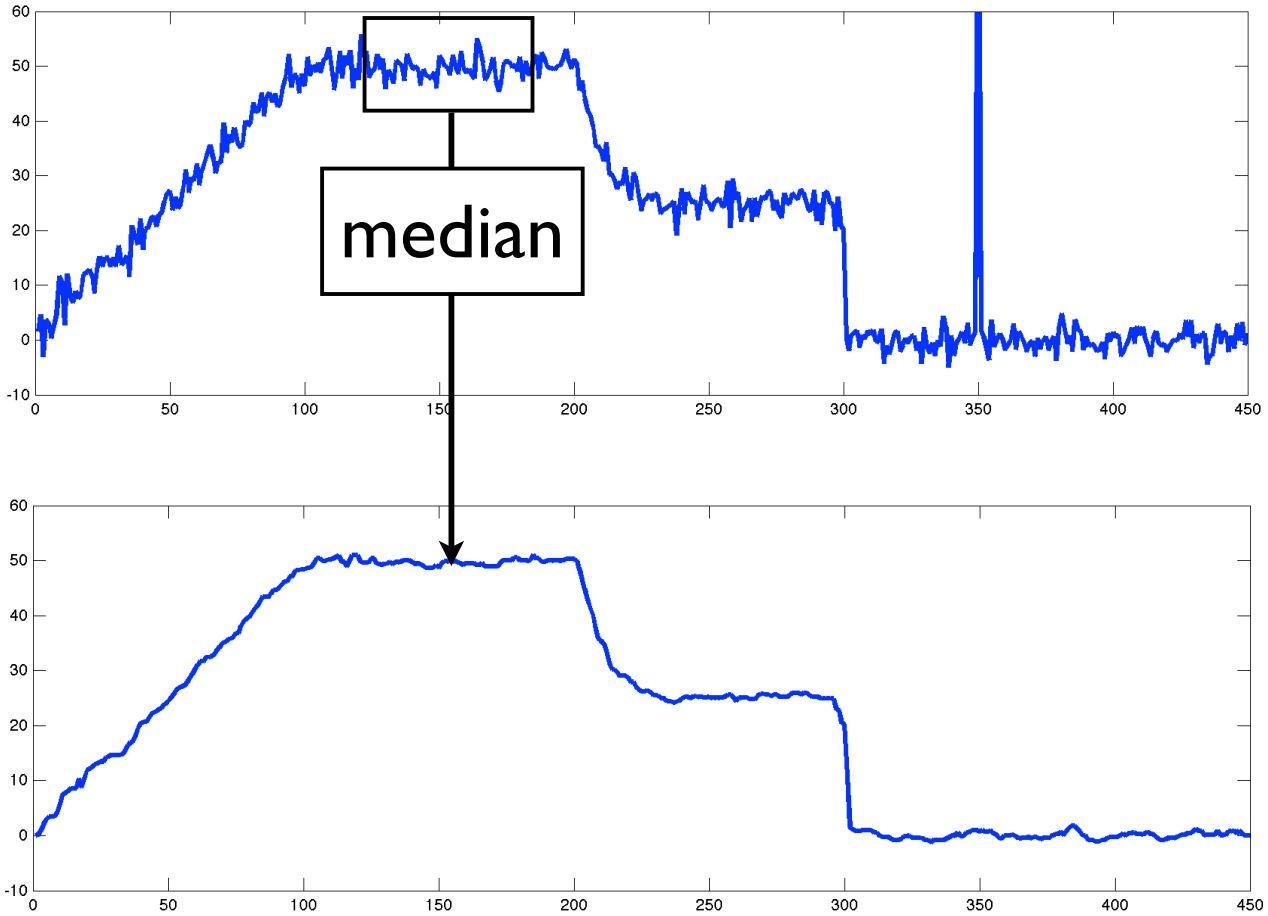


after smoothing with a uniform filter



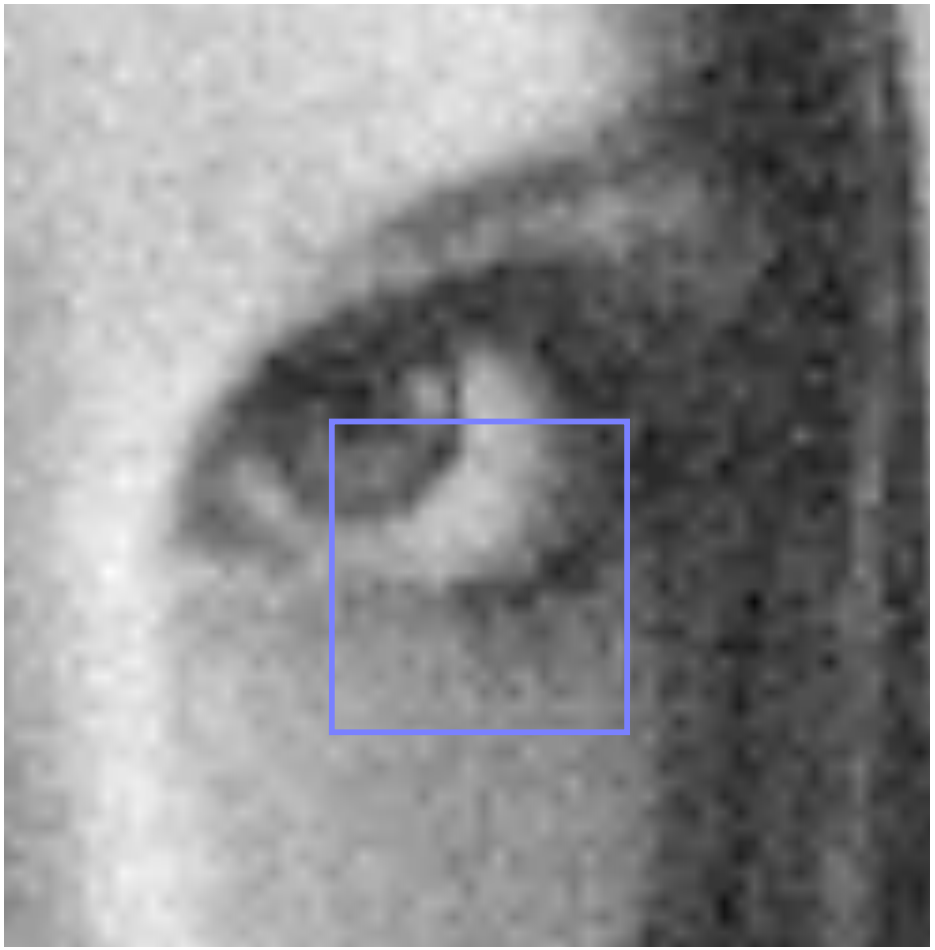


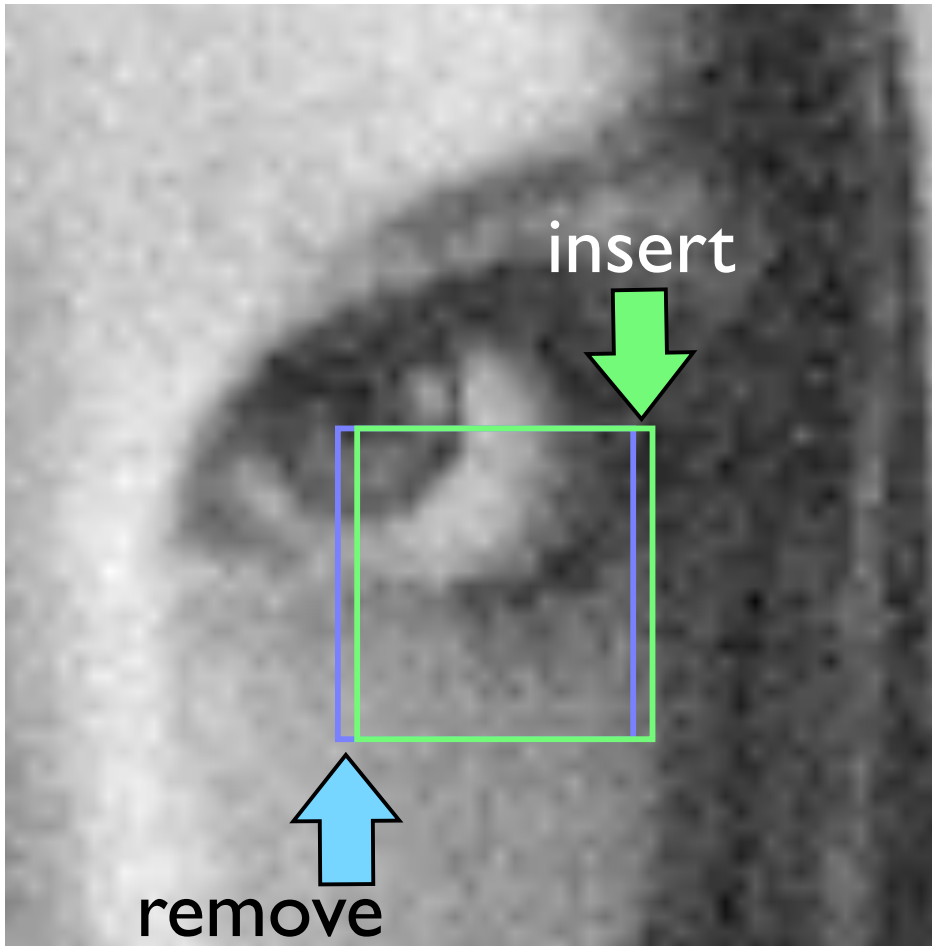
Median Filter



- Preserve discontinuities, can remove salt-and-pepper noise;
- More expensive than Gaussian smoothing.

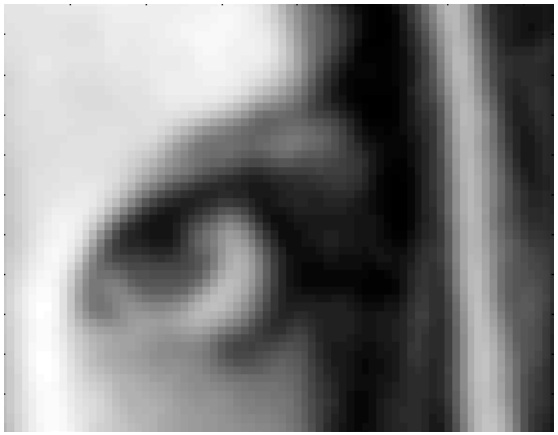
Median Filter
Efficient Implementation

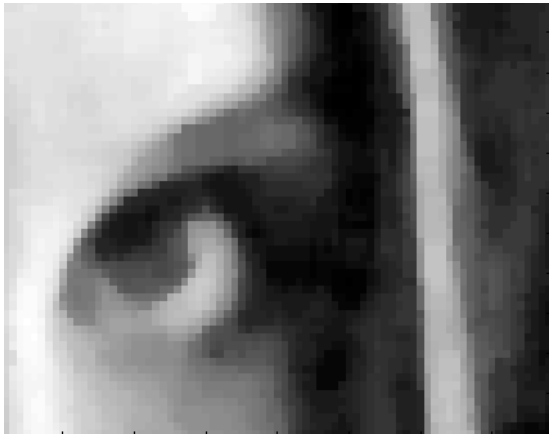




Gaussian Filter vs Median Filter on Images

Gaussian Noise



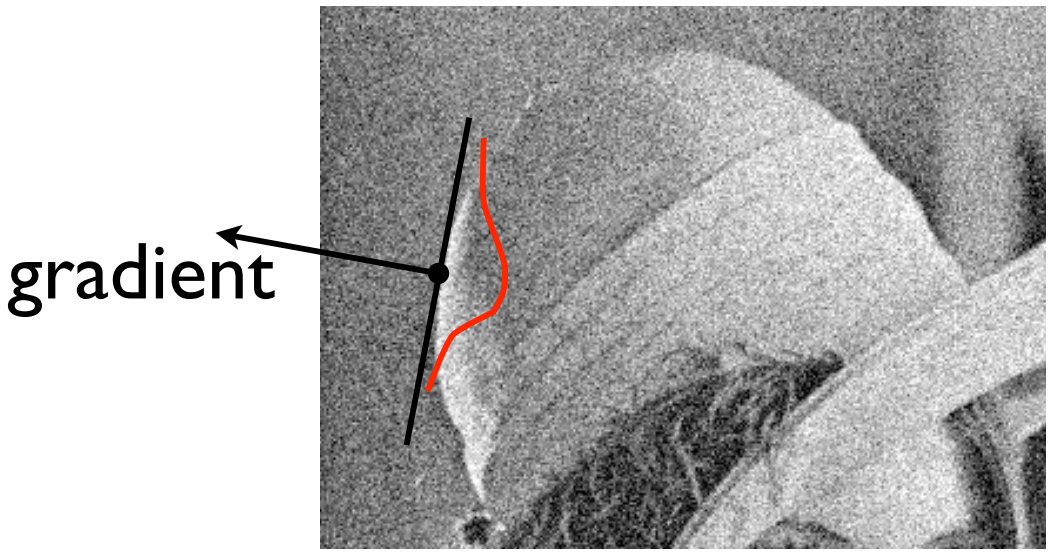


Gaussian Filter vs Median Filter on Images

Salt and Pepper Noise

Anisotropic Filters

- Avoid smoothing the discontinuities by convolving only in the direction orthogonal to the gradient.

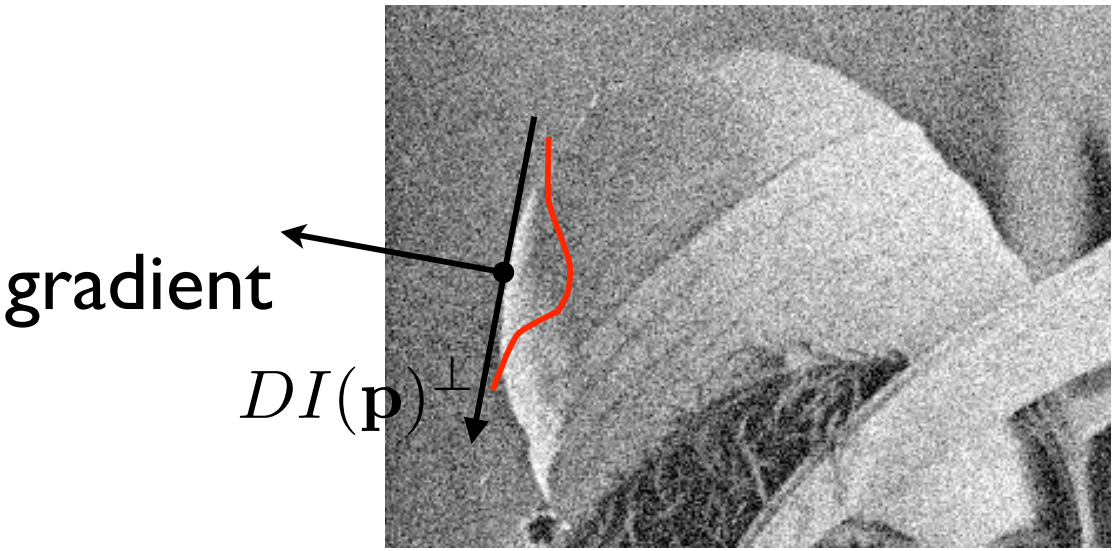


Anisotropic Filters

$$\text{Anisotropic Filter}(I, \mathbf{p}) = \int \frac{1}{\sqrt{2\pi}h} e^{\frac{-t^2}{2h^2}} I\left(\mathbf{p} + t\frac{DI(\mathbf{p})^\perp}{\|DI(\mathbf{p})\|}\right) dt$$

$DI(\mathbf{p})$: image gradient at $\mathbf{p}$

$(x, y)^\perp = (-y, x)$



Anisotropic Filters

Bad on uniform areas:

Input Image

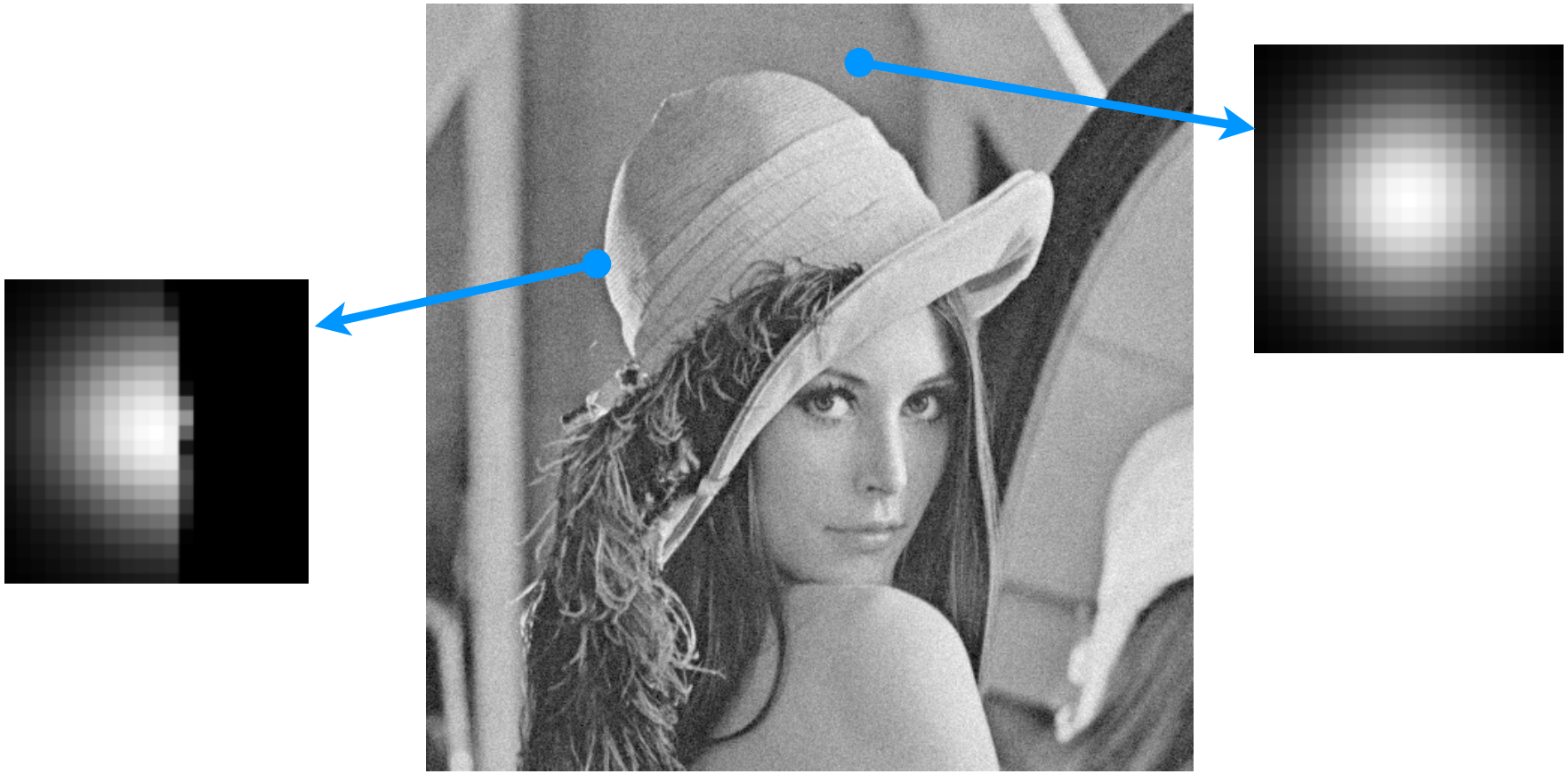
After Anisotropic Filtering

from [Buades et al, On image denoising methods]

The Bilateral Filter

([Tomasi and Manduchi, 1998], and some earlier references)

- edge-preserving and noise reducing smoothing filter:

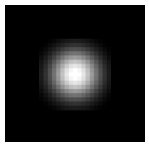


Bilateral Filter

$$\text{Gaussian Filter}(I,p) = \sum_{q \in \text{Window}(p)} G_{\sigma_S}(\|p-q\|)I(q)$$

$$\text{Bilateral Filter}(I,p) = \frac{1}{W(p)} \sum_{q \in \text{Window}(p)} G_{\sigma_S}(\|p-q\|)G_{\sigma_r}(|I_p - I_q|)I(q)$$

$$W(p) = \sum_{q \in \text{Window}(p)} G_{\sigma_S}(\|p-q\|)G_{\sigma_r}(|I_p - I_q|)$$

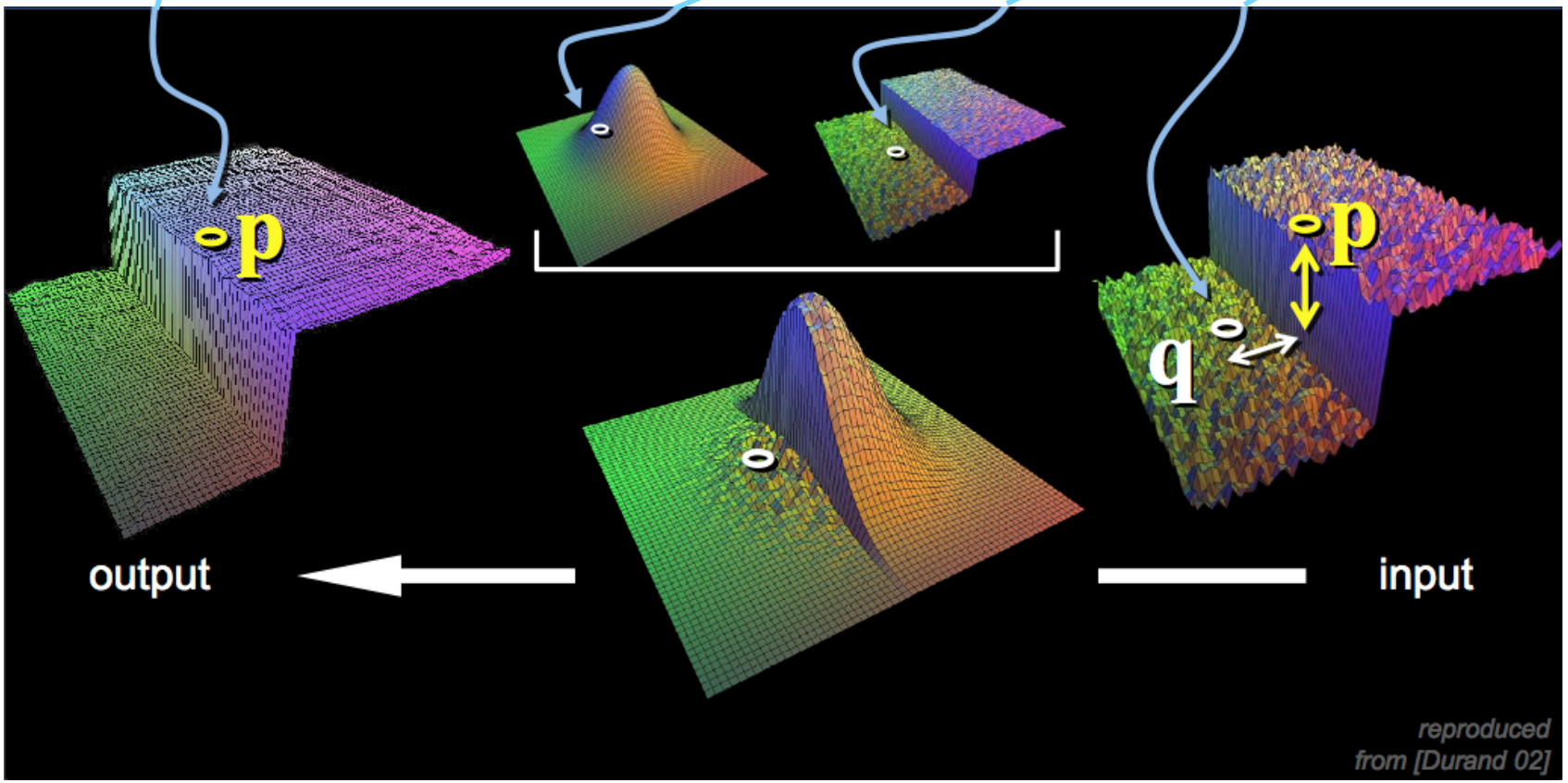


spatial weight

intensity range weight

Only pixels close in space and in intensity are considered.

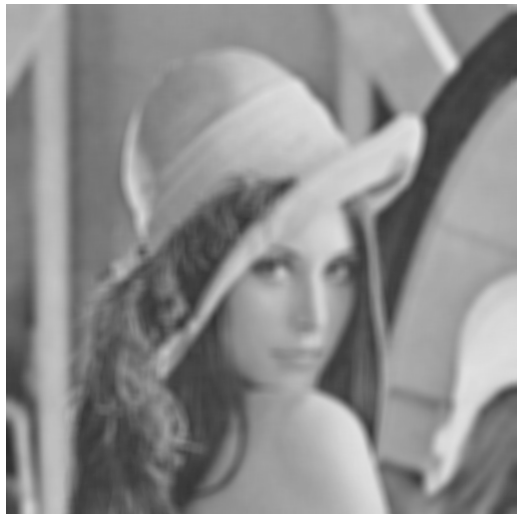
$$\text{Bilateral Filter}(I,p) = \frac{1}{W(p)} \sum_{q \in \text{Window}(p)} G_{\sigma_S}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|) I(q)$$



$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (gaussian convolution)

$\sigma_S = 2$

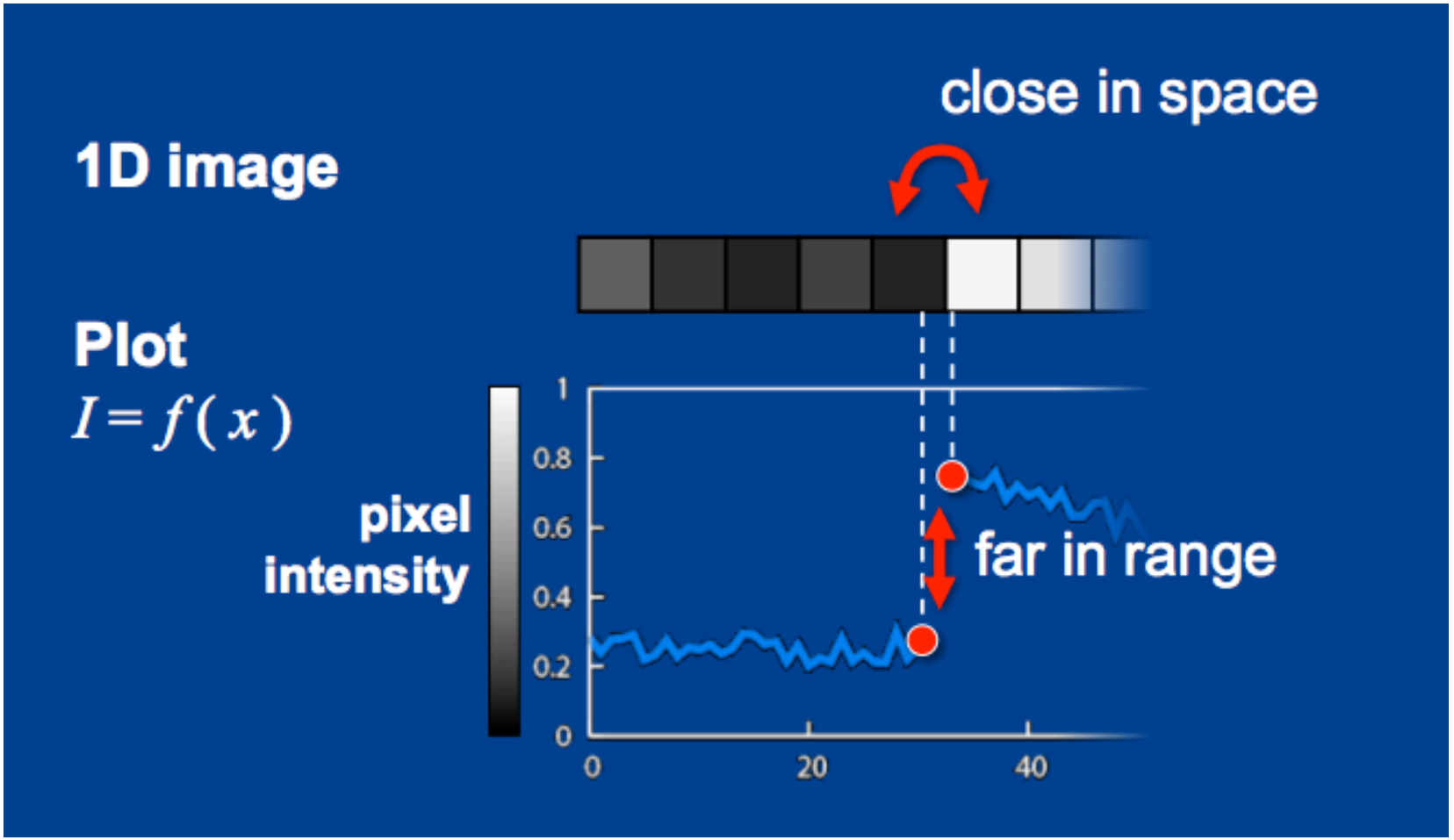
$\sigma_S = 18$ (cartoon)



Implementation

- Brutal force implementation can be slow.
- But there exists efficient implementations.

Fast Implementation of the Bilateral Filter [Paris and Durand'06]



from [A Gentle Introduction to Bilateal Filtering and its Applications", Paris et al].

Fast Implementation of the Bilateral Filter [Paris and Durand'06]

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)$$

Multiply first equation by $W_{\mathbf{p}}$

$$W_{\mathbf{p}}\, BF[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) 1$$

Fast Implementation of the Bilateral Filter [Paris and Durand'06]

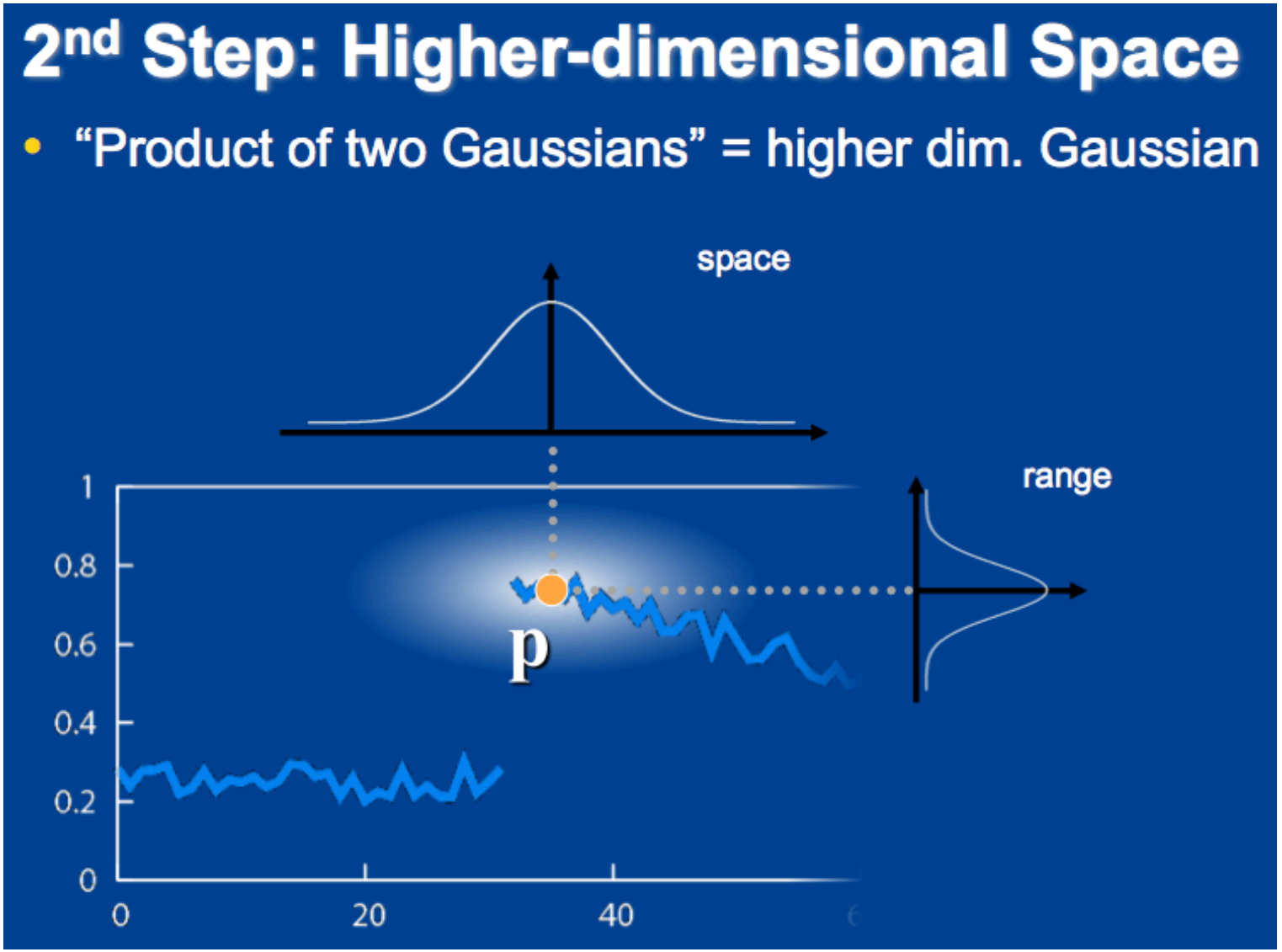
1st Step: Summary

$$W_{\mathbf{p}}\, BF[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

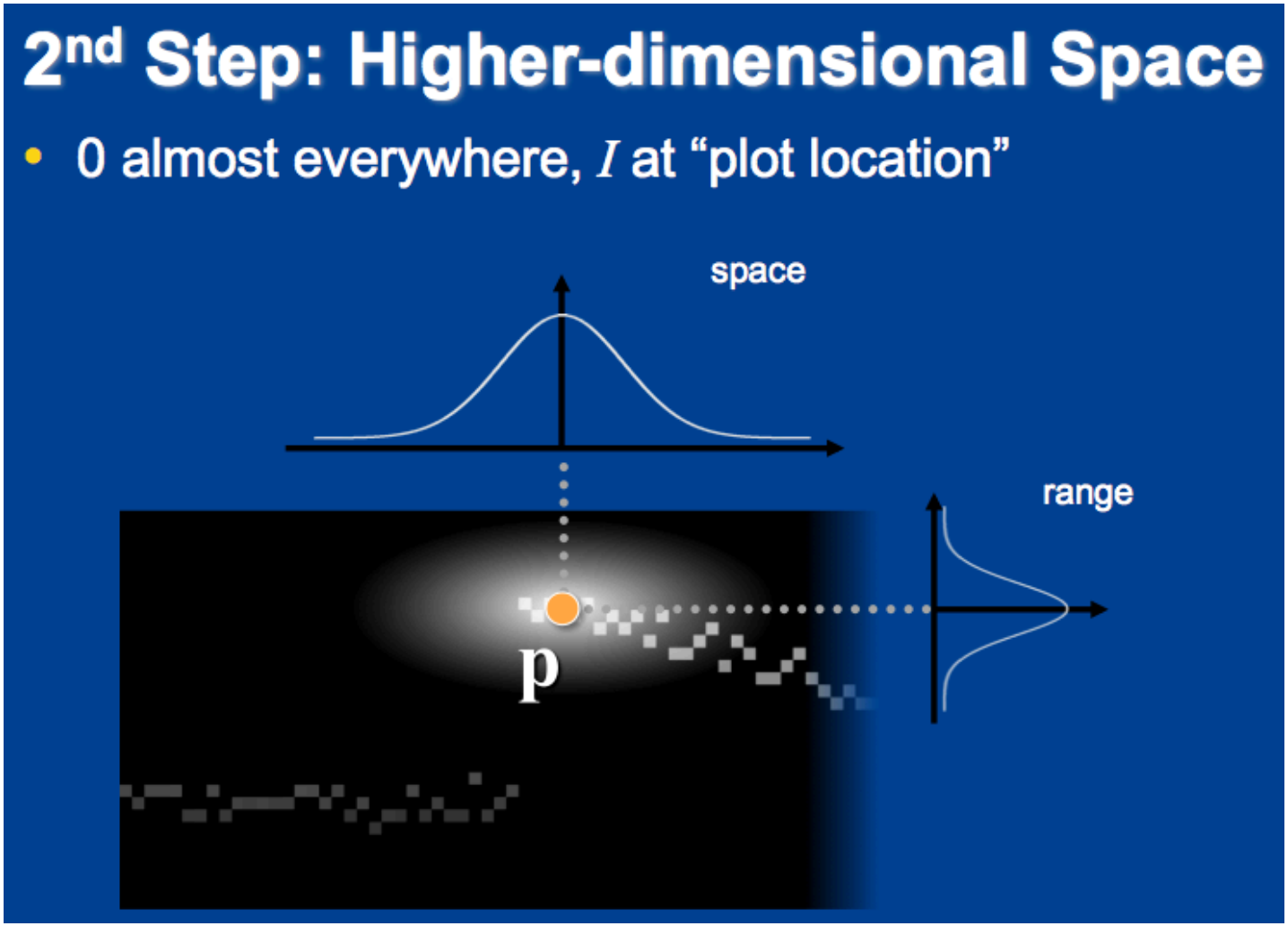
$$W_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) 1$$

- Similar equations
- No normalization factor anymore
- Don't forget to divide at the end

Fast Implementation of the Bilateral Filter [Paris and Durand'06]



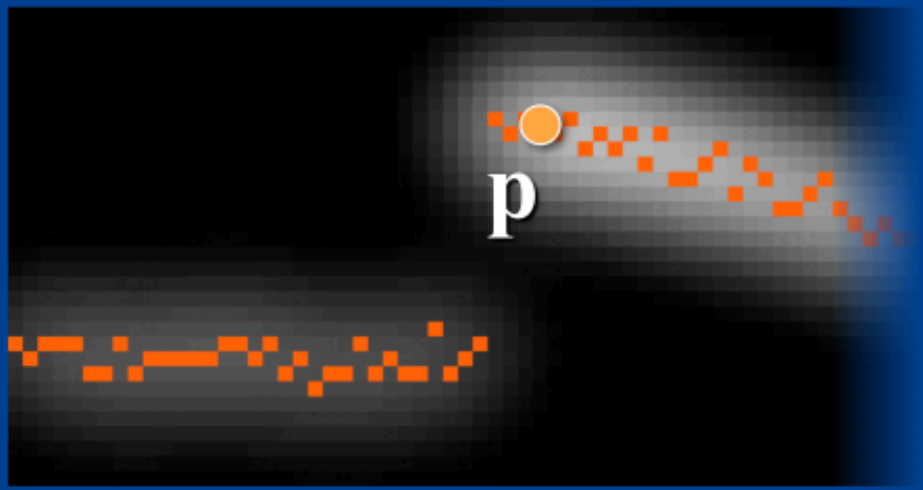
Fast Implementation of the Bilateral Filter [Paris and Durand'06]



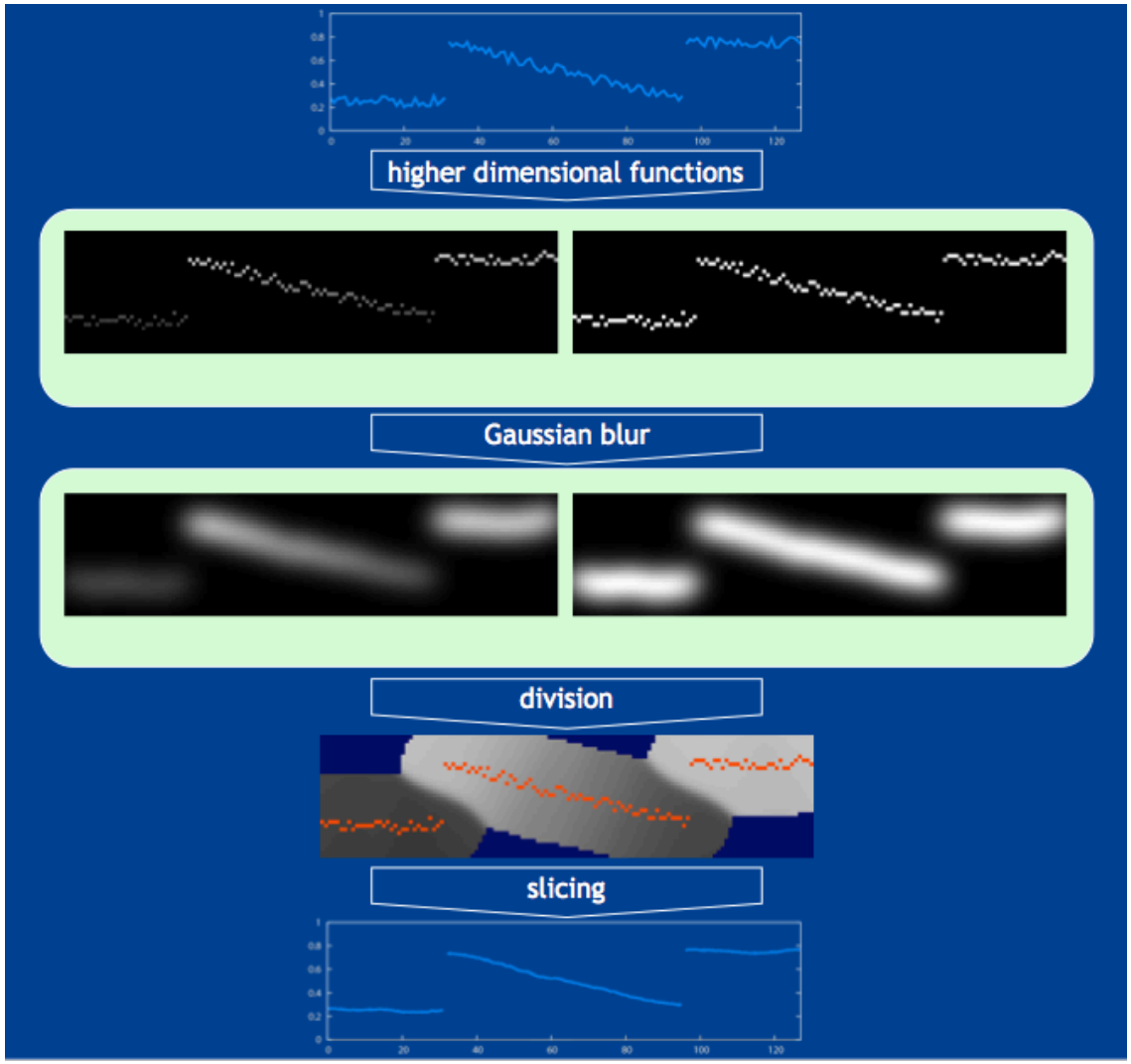
Fast Implementation of the Bilateral Filter [Paris and Durand'06]

2nd Step: Higher-dimensional Space

- 0 almost everywhere, I at "plot location"
- Weighted average at each point = Gaussian blur
- Result is at "plot location"



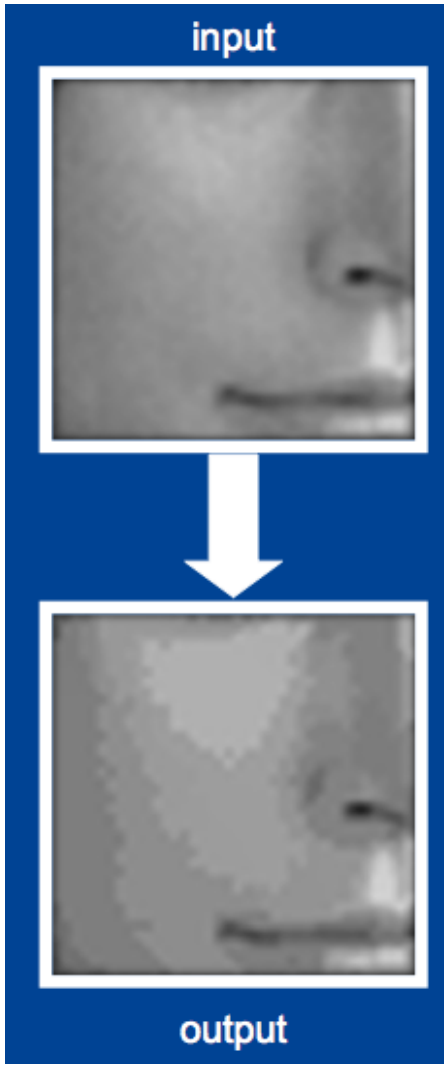
Fast Implementation of the Bilateral Filter [Paris and Durand'06]



Limitations

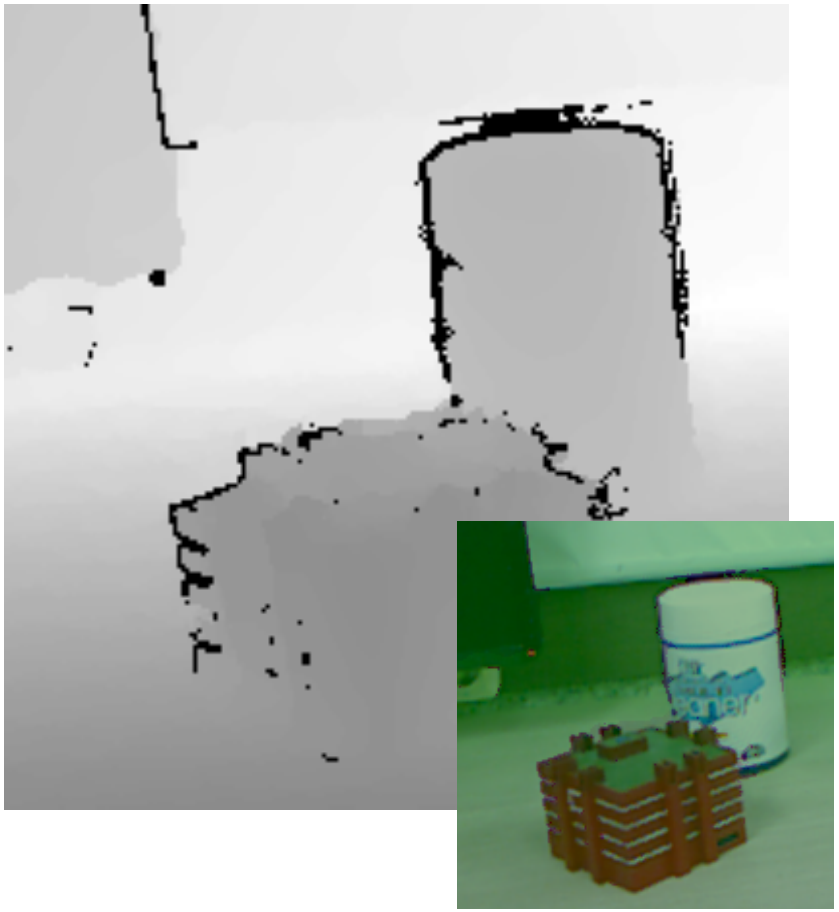
Bilateral filter tends to:

- remove texture;
- create flat intensity regions, and
- new contours.



Bilateral Filter is flexible

Adapted to smooth out the 3D output of the Kinect:



Wiener Filtering

First use of a statistical model for the images (or signals in general).

Wiener Filtering Derivation

observed image

unknown image (we are trying to recover that)

noise

$$o(x,y) = s(x,y) + n(x,y),$$

Fourier transform:

$$O(\omega_x, \omega_y) = S(\omega_x, \omega_y) + N(\omega_x, \omega_y),$$

At each frequency (ω_x, ω_y), according to Bayes' Rule:

$$p(S(\omega_x, \omega_y) \mid O(\omega_x, \omega_y)) = \frac{p(O(\omega_x, \omega_y) \mid S(\omega_x, \omega_y))p(S(\omega_x, \omega_y))}{p(O(\omega_x, \omega_y))}$$

from "Computer Vision: Algorithms and Applications", R. Szeliski.

Wiener Filtering Derivation

Dropping (ω_x, ω_y) for clarity: $p(S \mid O) = \dfrac{p(O \mid S)p(S)}{p(O)}$

Assumption 1: S has a prior distribution $p(S)$ which is a zero-mean Gaussian with variance $P_s(\omega_x, \omega_y)$:

$$p(S) = e^{-\frac{(S-\mu)^2}{2P_s}}$$

mean (=0 except at the origin)

variance at frequency (ω_x, ω_y). Can be estimated empirically from many images.

from "Computer Vision: Algorithms and Applications", R. Szeliski.

Wiener Filtering
Derivation

$$p(S \mid O) = \frac{p(O \mid S)p(S)}{p(O)} \qquad p(S) = e^{-\frac{(S-\mu)^2}{2P_s}}$$

Assumption 2: O given S has a Gaussian distribution with variance P_n :

$$p(O \mid S) = e^{-\frac{(S-O)^2}{2P_n}}$$

from "Computer Vision: Algorithms and Applications", R. Szeliski.

Wiener Filtering
Derivation

$$p(S \mid O) = \frac{p(O \mid S)p(S)}{p(O)} \qquad p(S) = e^{-\frac{(S-\mu)^2}{2P_s}} \qquad p(O \mid S) = e^{-\frac{(S-O)^2}{2P_n}}$$

negative posterior log-likelihood:

$$\begin{aligned} -\log p(S|O) &= -\log p(O|S) - \log p(S) + C \\ &= {}^1\!/_2 P_n^{-1}(S-O)^2 + {}^1\!/_2 P_s^{-1} S^2 + C \end{aligned}$$

optimal S_{opt} minimizes this quantity:

$$S_{\text{opt}} = \frac{P_n^{-1}}{P_n^{-1} + P_s^{-1}} O = \frac{P_s}{P_s + P_n} O = \frac{1}{1 + P_n/P_s} O$$

from "Computer Vision: Algorithms and Applications", R. Szeliski.

Wiener Filtering Derivation

Optimal S_{opt} minimizes this quantity:

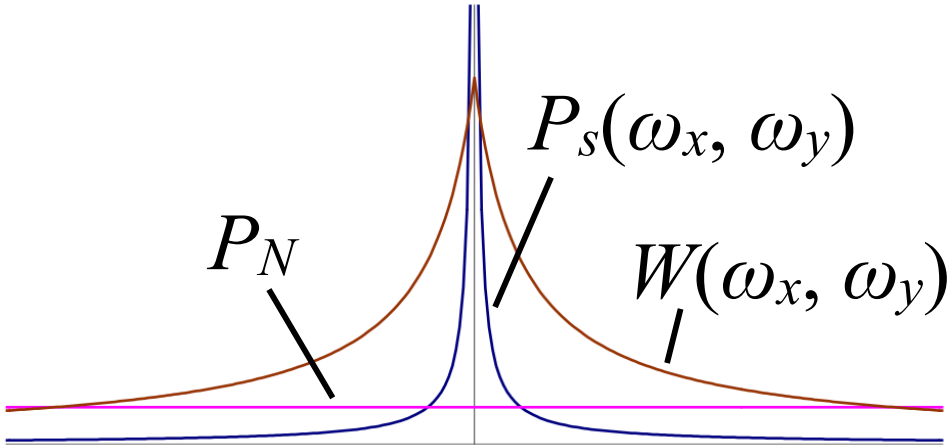
$$S_{\text{opt}} = \frac{P_n^{-1}}{P_n^{-1} + P_s^{-1}} O = \frac{P_s}{P_s + P_n} O = \frac{1}{1 + P_n/P_s} O$$

Fourier transform of the optimum Wiener filter:

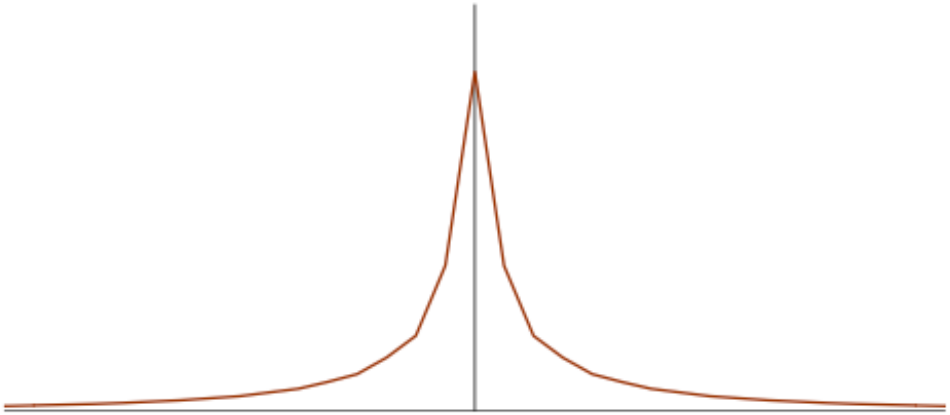
$$W(\omega_x, \omega_y) = \frac{1}{1 + P_n/P_s(\omega_x, \omega_y)}$$

from "Computer Vision: Algorithms and Applications", R. Szeliski.

Wiener Filtering in 1 Dimension



Fourier domain



Spatial domain

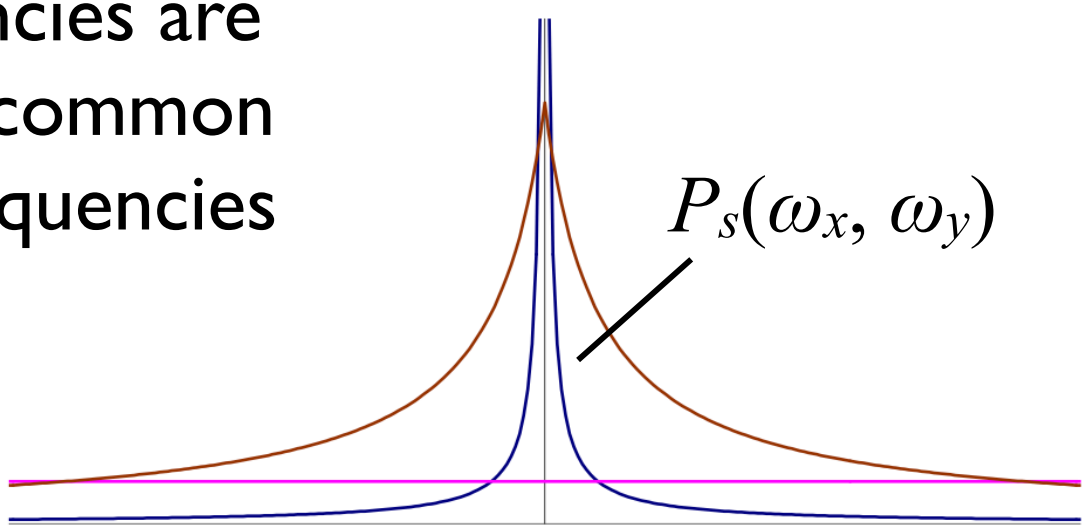
Results

Input Image

After Wiener Filtering

Remark

Low frequencies are much more common than high frequencies



Fourier domain