Department of Medical Physics and Biomedical Engineering

MPHY0030 Programming Foundations for Medical Image Analysis

Yipeng Hu

yipeng.hu@ucl.ac.uk

Filtering

Filtering

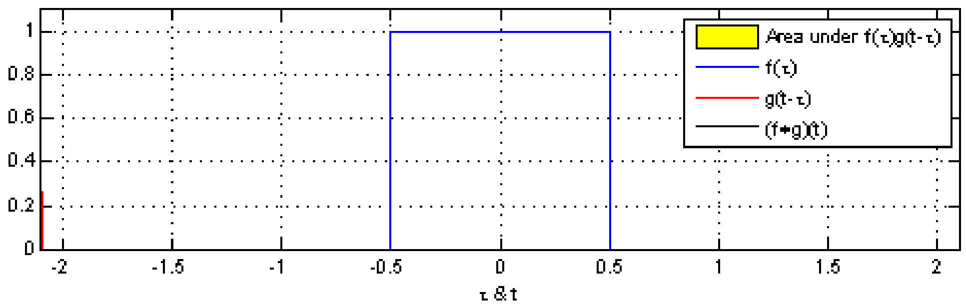
- Convolution
- Smoothing
- Differentiating
- Edge-preserving smoothing
- Restoring
- Local structuring
- Separable kernels
- Frequency multiplication
- Scale space

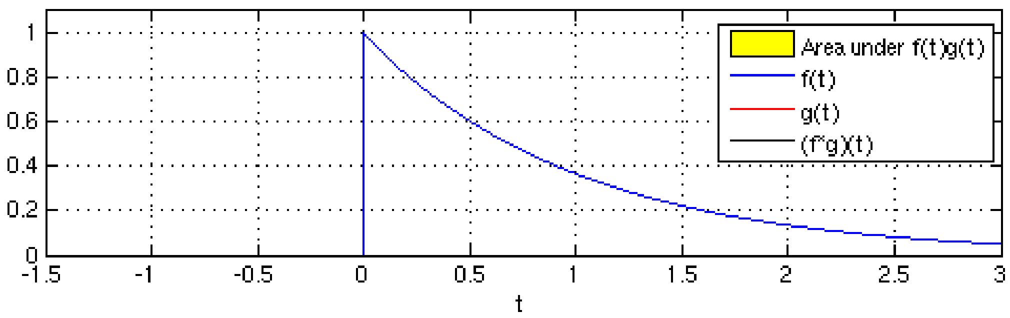
Filtering | Convolution

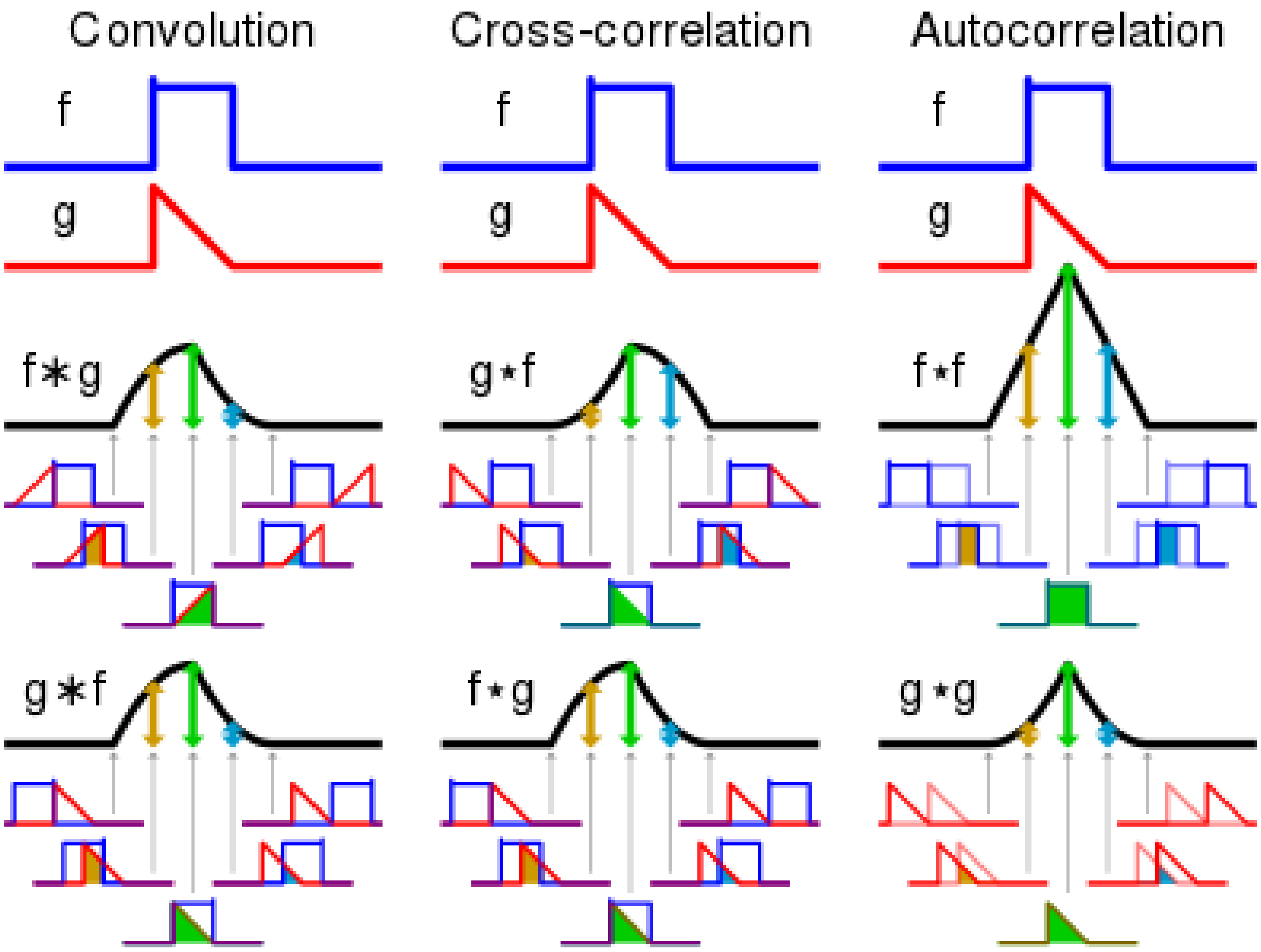
Discrete convolution

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau.$$

$$(f * g)(t) := \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$$

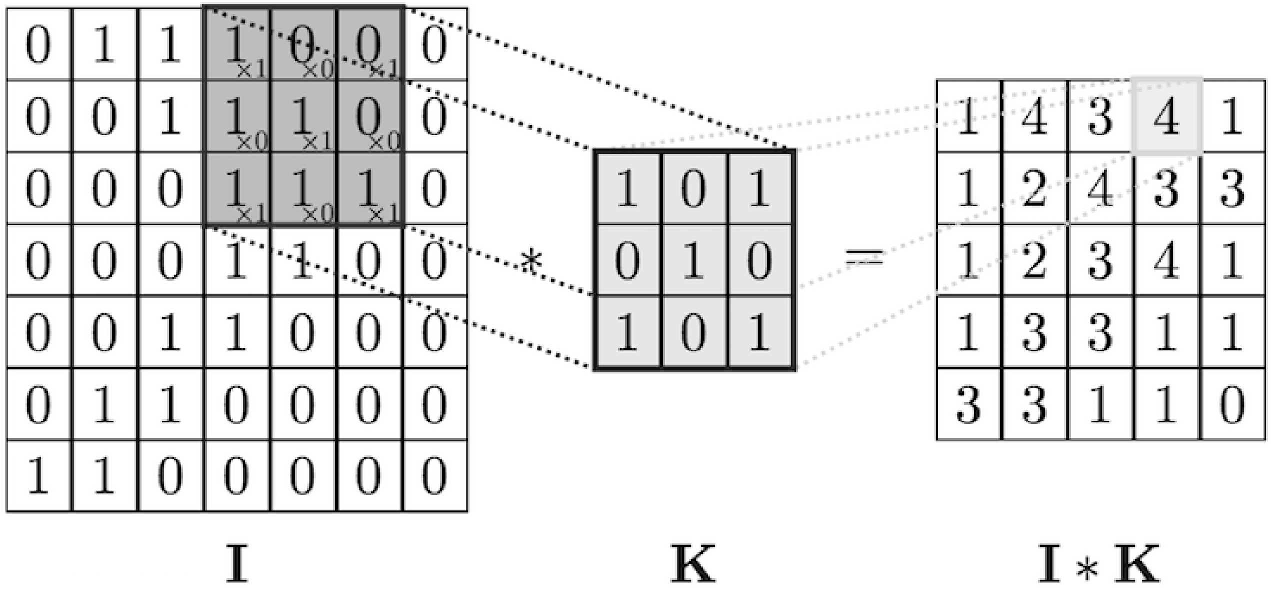


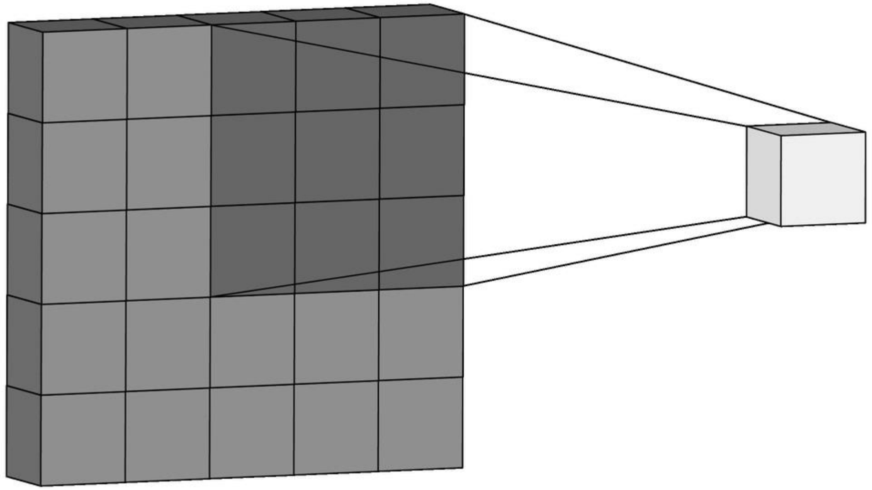



Convolution
Cross-correlation
Autocorrelation
f
g
f∗g
g∗f
g⋆f
f⋆g
f⋆f
g⋆g

Discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$





2D discrete convolution

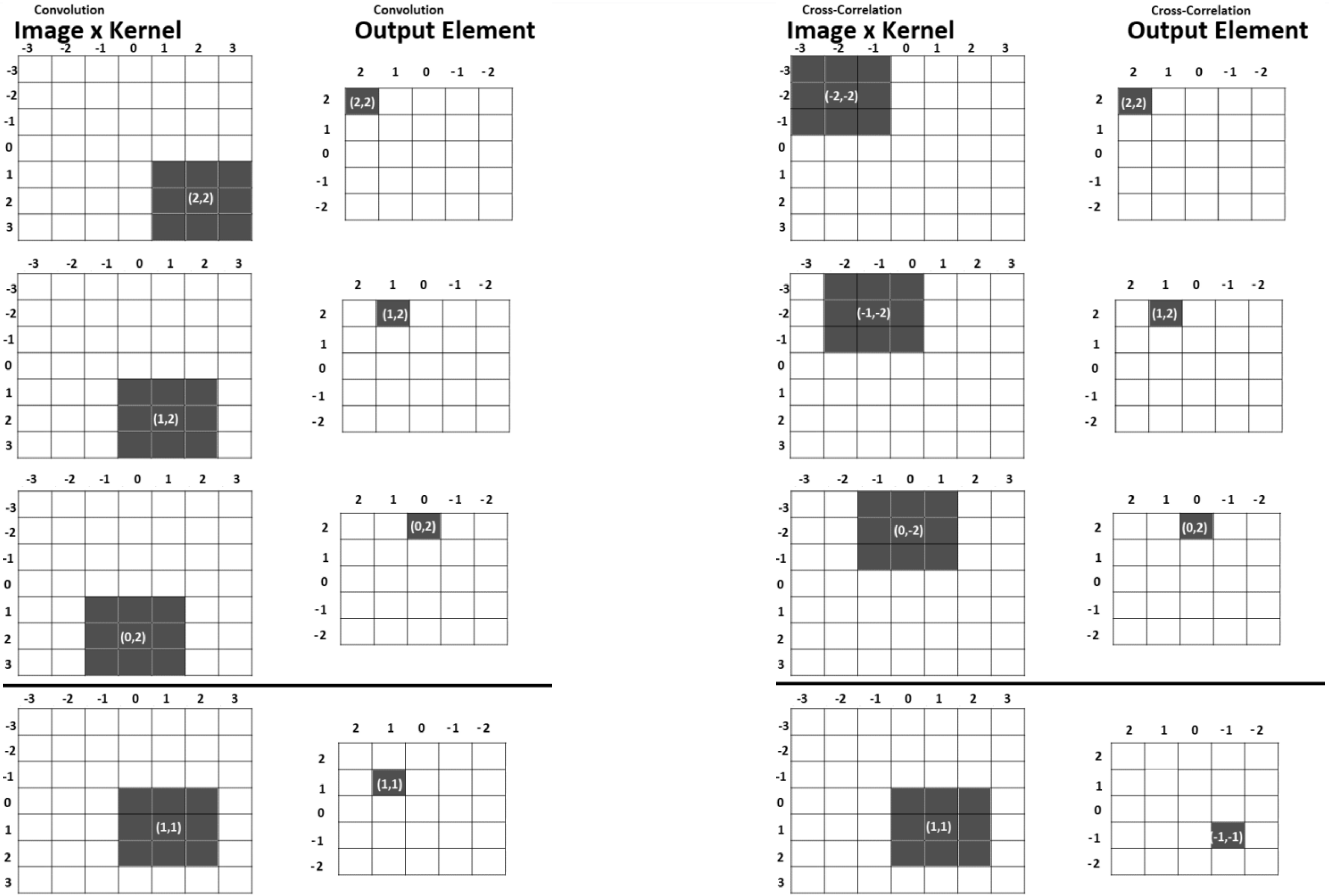
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n).$$

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n).$$

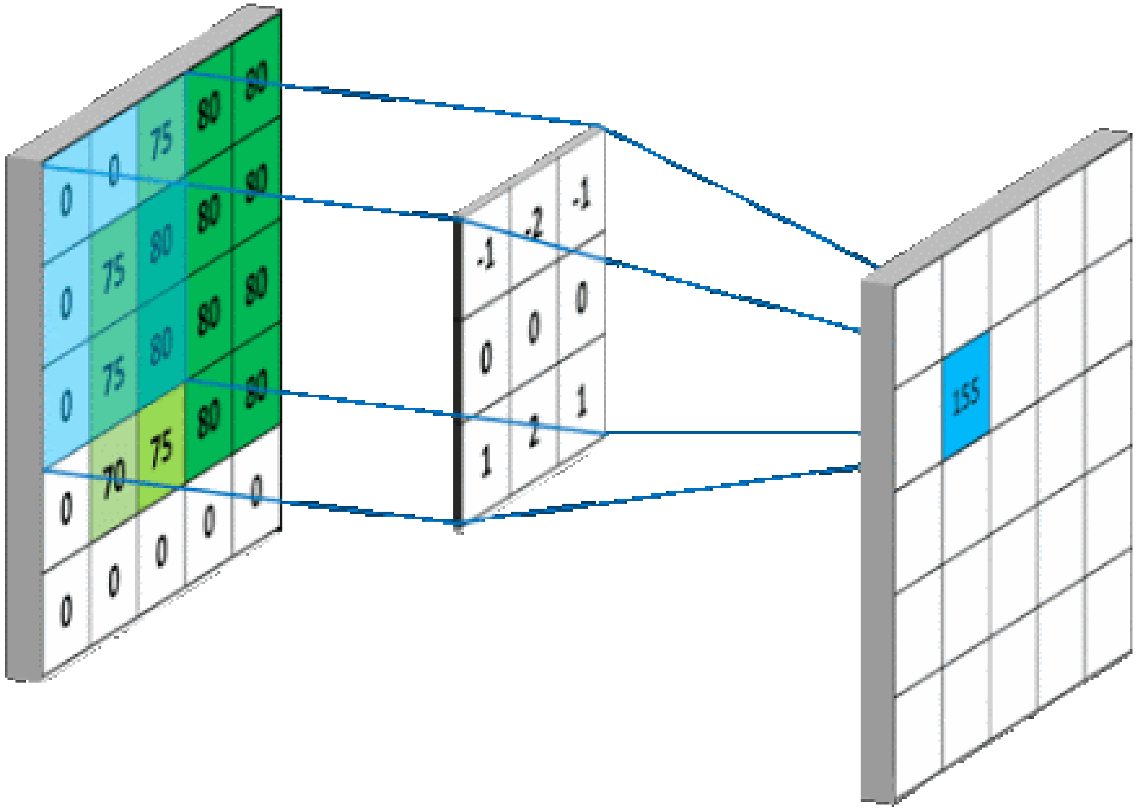
2D discrete cross-correlation

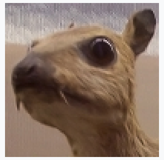
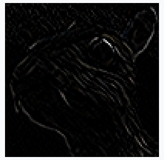
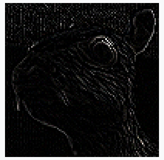
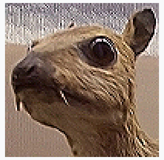
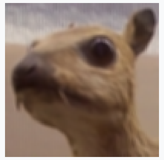
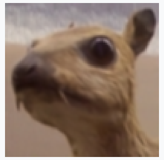
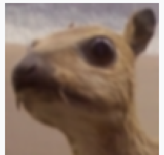
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n).$$

Filtering | Convolution



Discrete kernels



Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

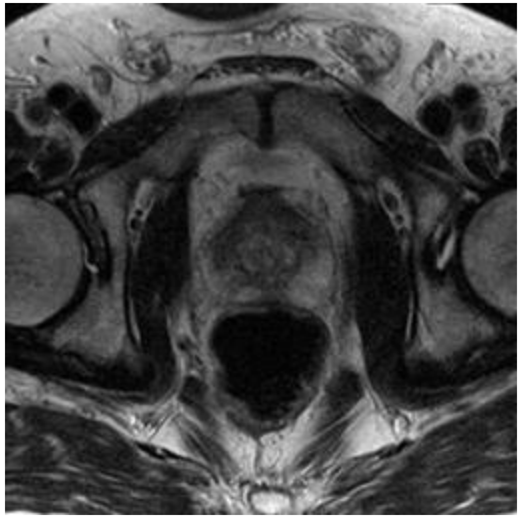
Filtering | Smoothing

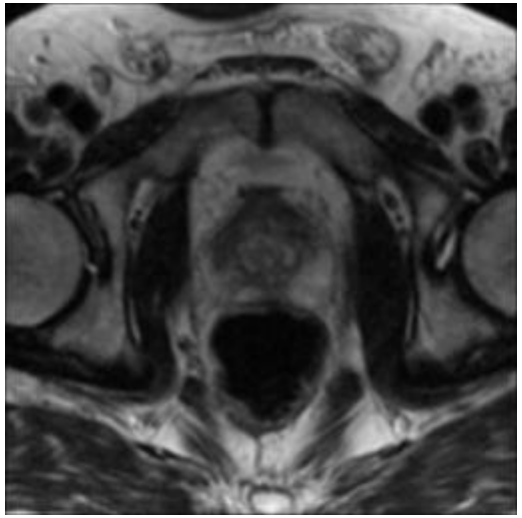
Blurring

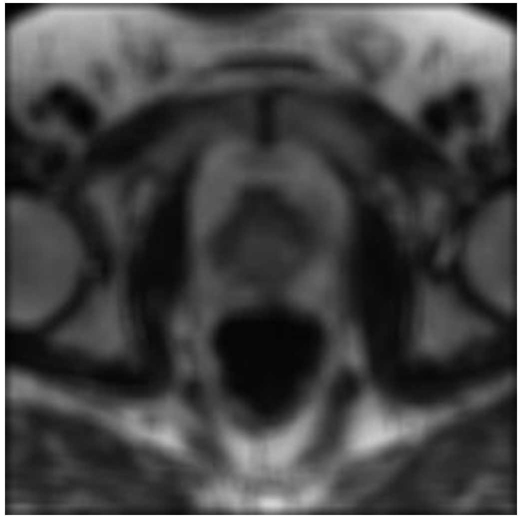
Mean filters

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

$$\frac{1}{81}\begin{bmatrix}1 & \cdots & 1\\\vdots & \ddots & \vdots\\1 & \cdots & 1\end{bmatrix}$$





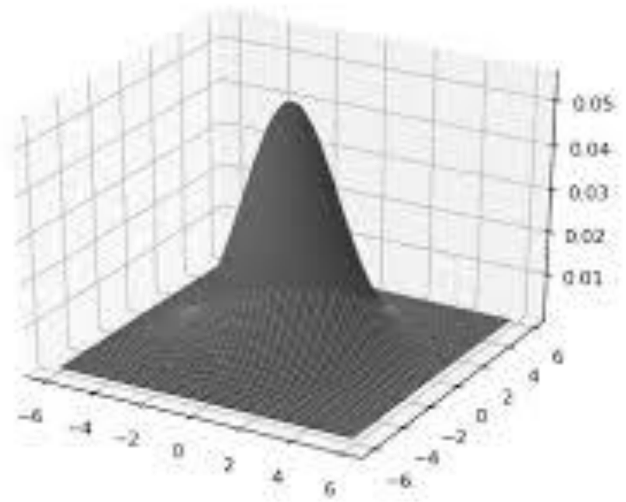


Low-pass smoothing

Gaussian filters

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$p(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{\det(\Sigma)}} exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$



1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

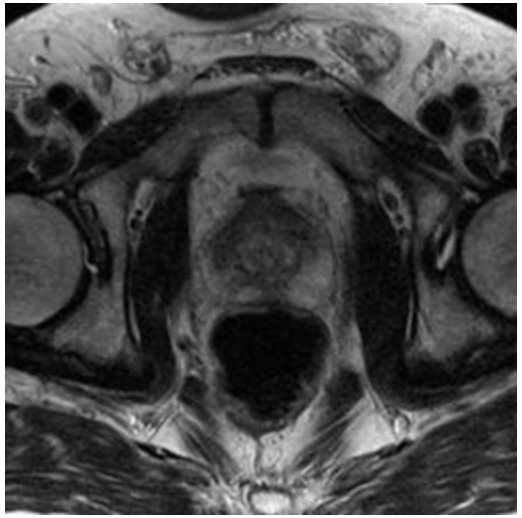
0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

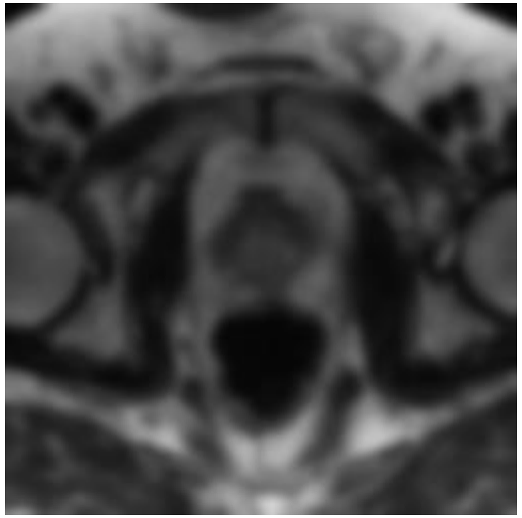
How to determine the discrete kernel size?

Low-pass smoothing

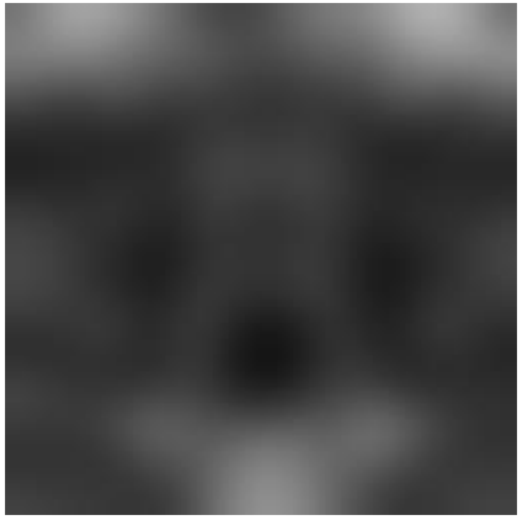
Gaussian filters

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





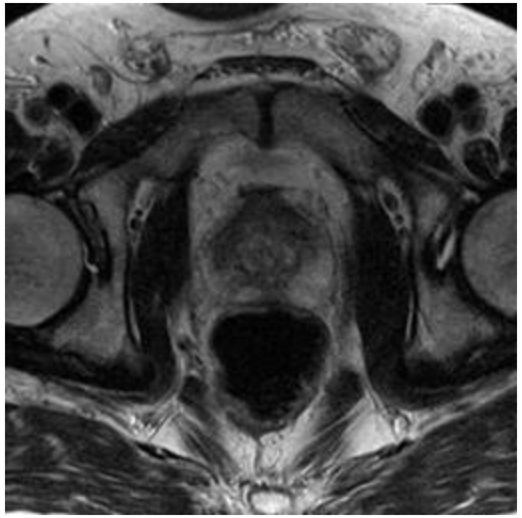
$\sigma = 3$

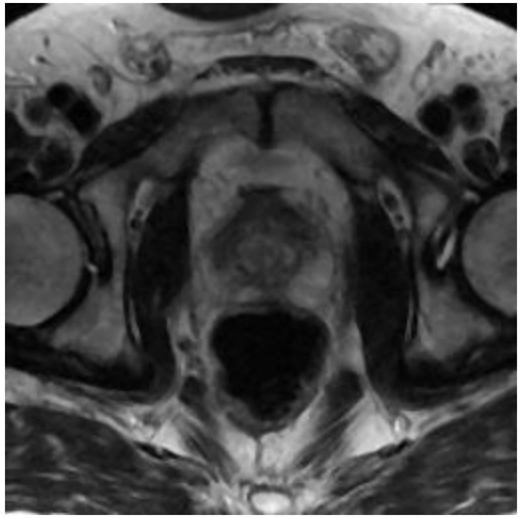


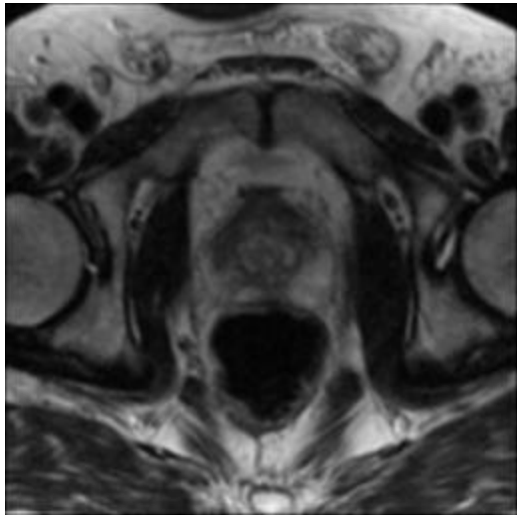
$\sigma = 15$

Nonlinear smoothing

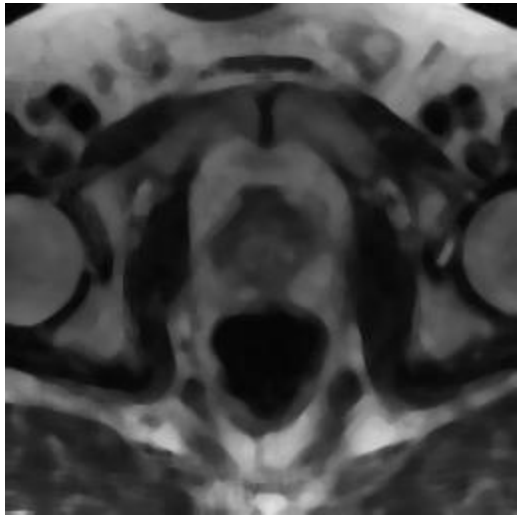
Median (percentiles) filters

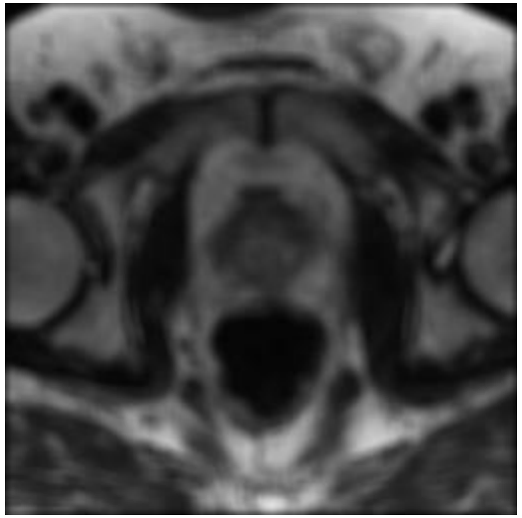






3×3





7×7

Median filters

Average filters

Purposes

- Denoising
- Edge removing
- Resizing
- Spatial transforming

Applications

Everywhere!

Filtering | Differentiating

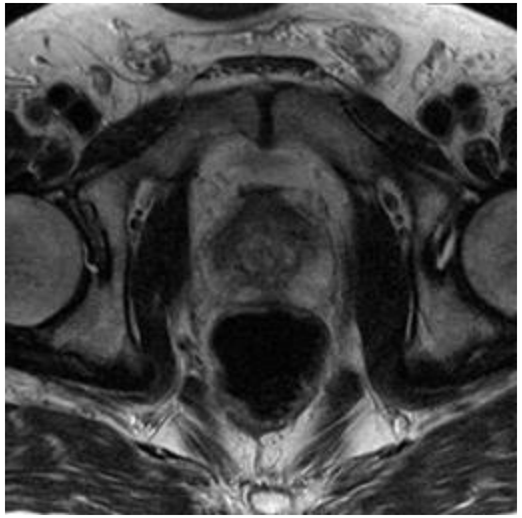
Edge detection

Prewitt operator

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

Sobel operator

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



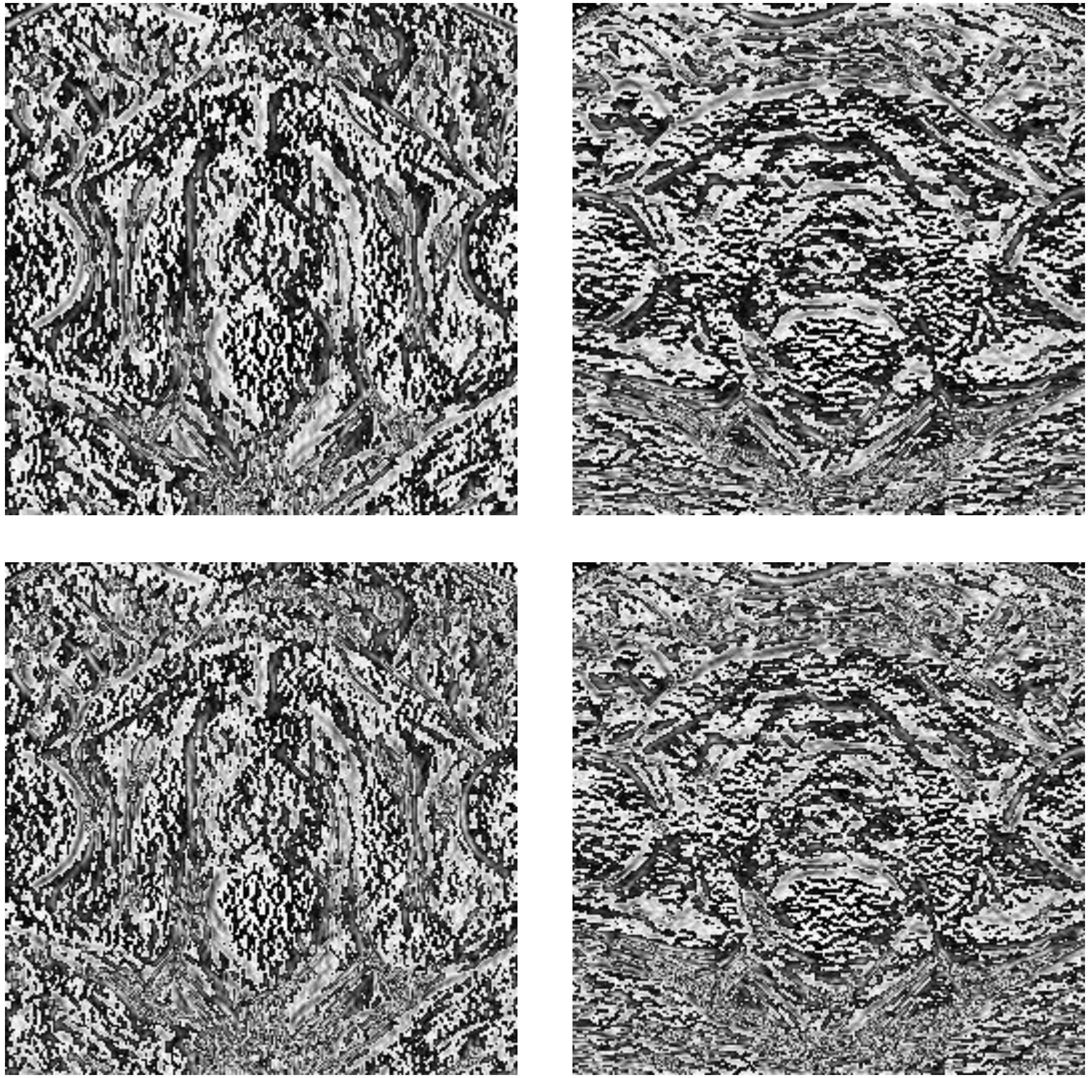


Image derivatives

First-order image derivatives (the image gradient) $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$, implemented by convolution, with finite difference kernels D_x and D_y, e.g.:

$$\frac{\partial I}{\partial x} = I * D_x = I * \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} \text{ and } \frac{\partial I}{\partial y} = I * D_y = I * \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}^T$$

Gaussian derivatives

First-order image derivatives (the image gradient) $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$, implemented by convolution, with finite difference kernels D_x and D_y, e.g.:

$$\frac{\partial I}{\partial x} = I * D_x = I * \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} \text{ and } \frac{\partial I}{\partial y} = I * D_y = I * \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}^T$$

The finite difference kernels to compute the first-order image derivatives can artificially magnifies the high frequency noise level.

Smoothing the image with a Gaussian filter $I^s = I * G$ before taking the derivatives, which can be efficiently implemented using a convolution with the derivatives of a Gaussian kernel $\nabla \mathrm{G}$:

$$\nabla I^s = \nabla(I * G) = \nabla I * G = I * \nabla \mathrm{G}$$

where $\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}^T$ and $\nabla G = \begin{bmatrix} \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \end{bmatrix}^T$, while the Gaussian derivatives are given by

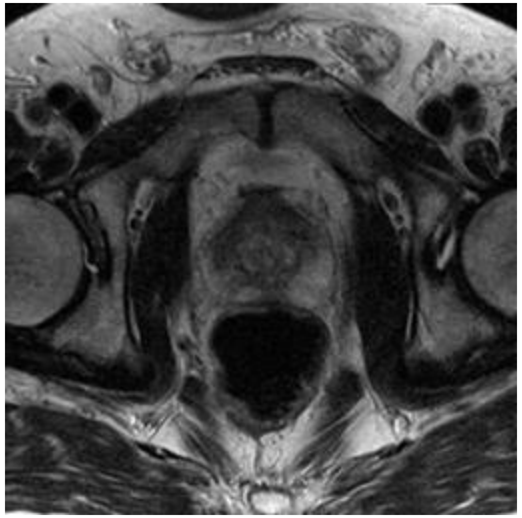
$$\frac{\partial G}{\partial x} = \frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \text{ and } \frac{\partial G}{\partial y} = \frac{y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

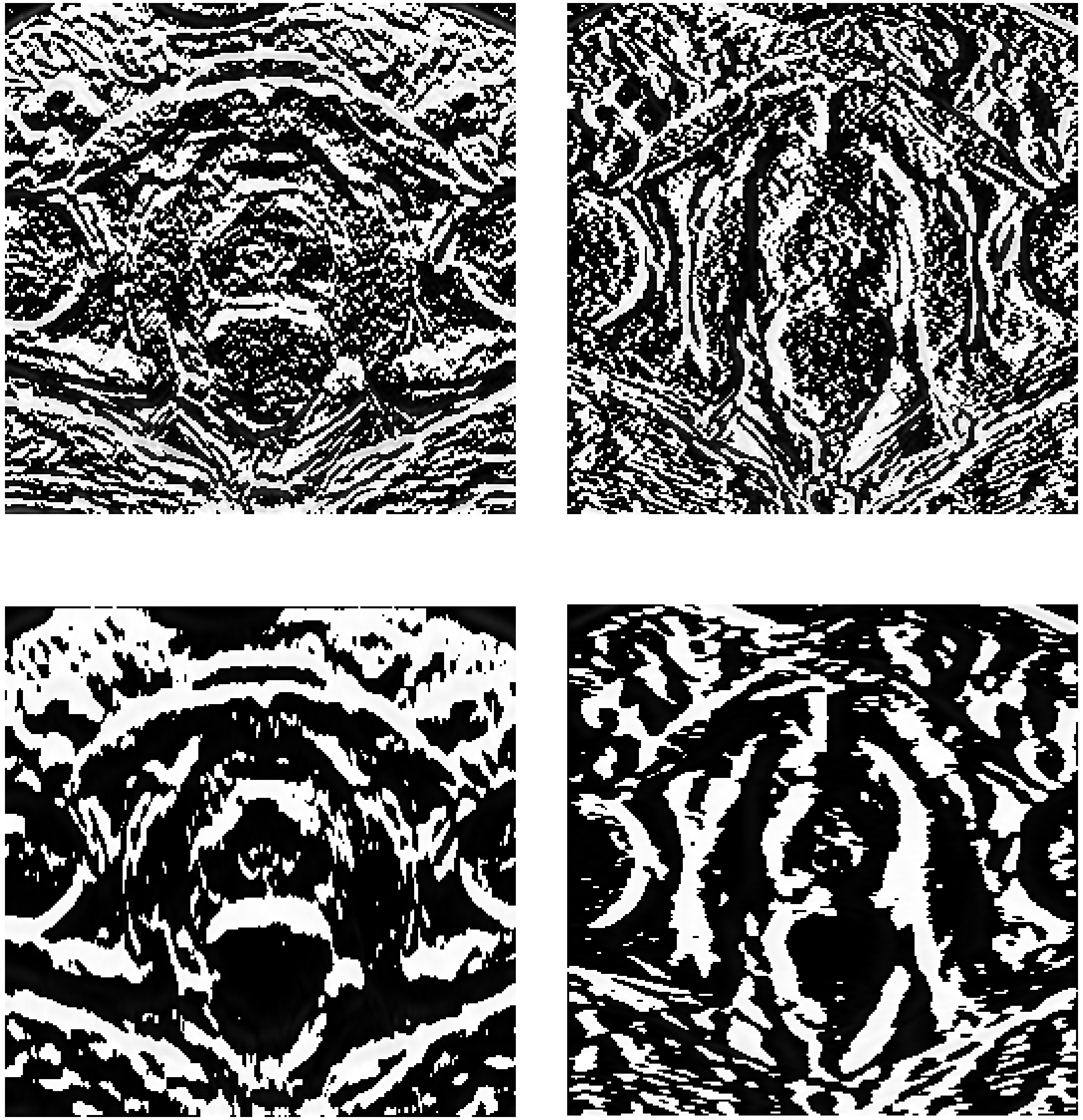
These first-order image derivatives and the magnitude $\|\nabla I^s\|$,

$$\|\nabla I^s\| = \sqrt[2]{\left(\frac{\partial I^s}{\partial x}\right)^2 + \left(\frac{\partial I^s}{\partial y}\right)^2}$$

Gaussian derivatives

Approximating Canny edge detector



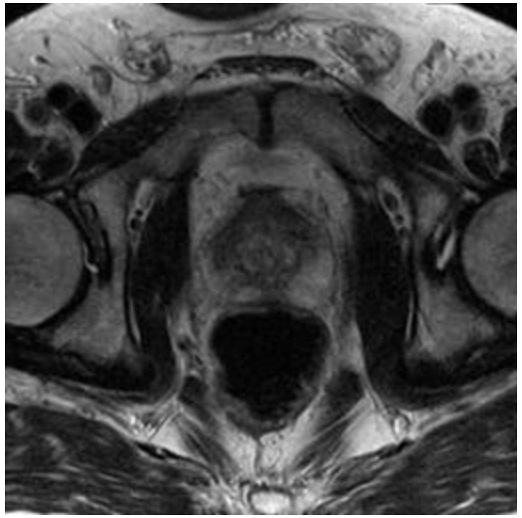


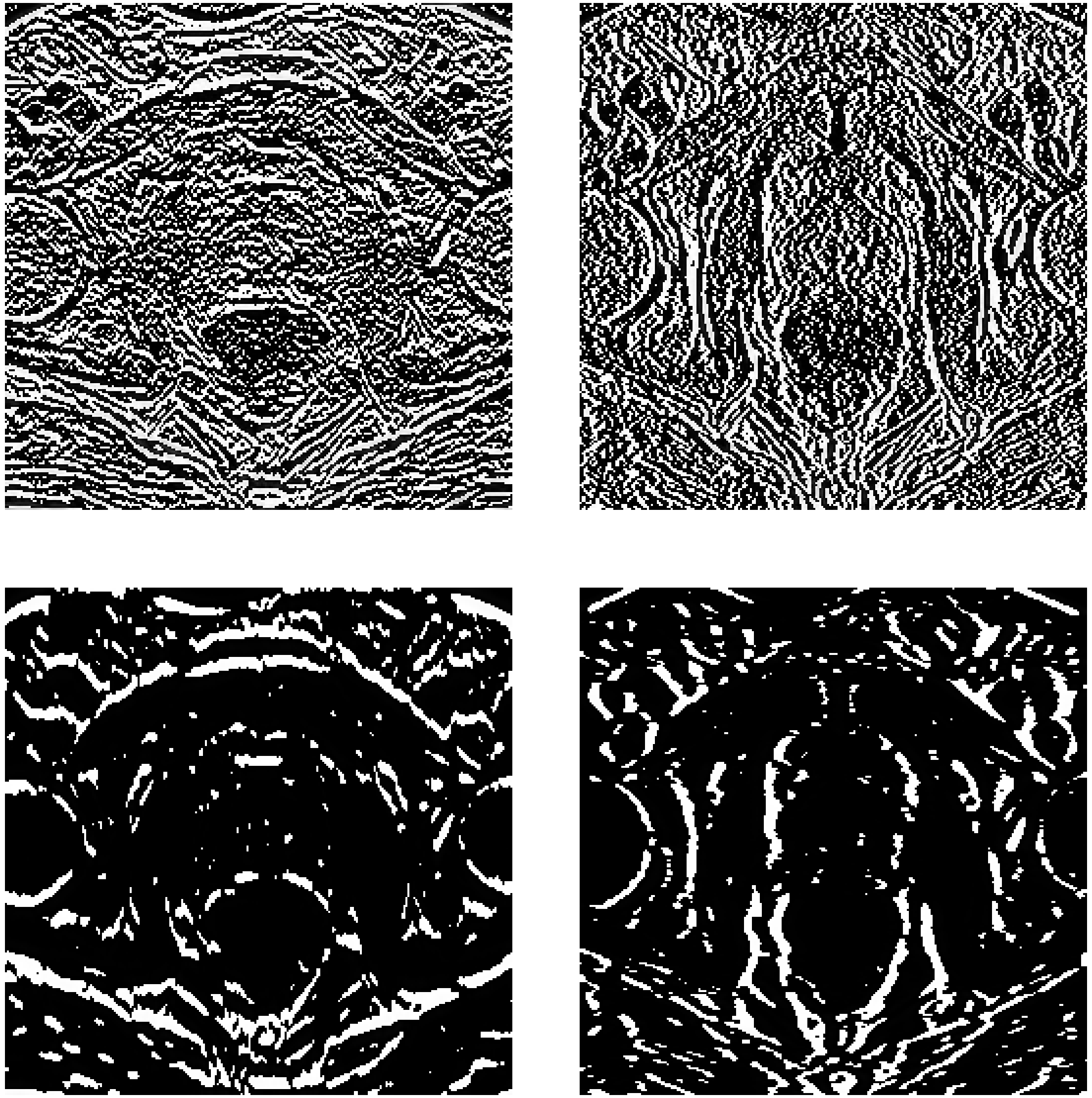
$\sigma = 1$

$\sigma = 3$

Laplacian

Second Gaussian derivatives



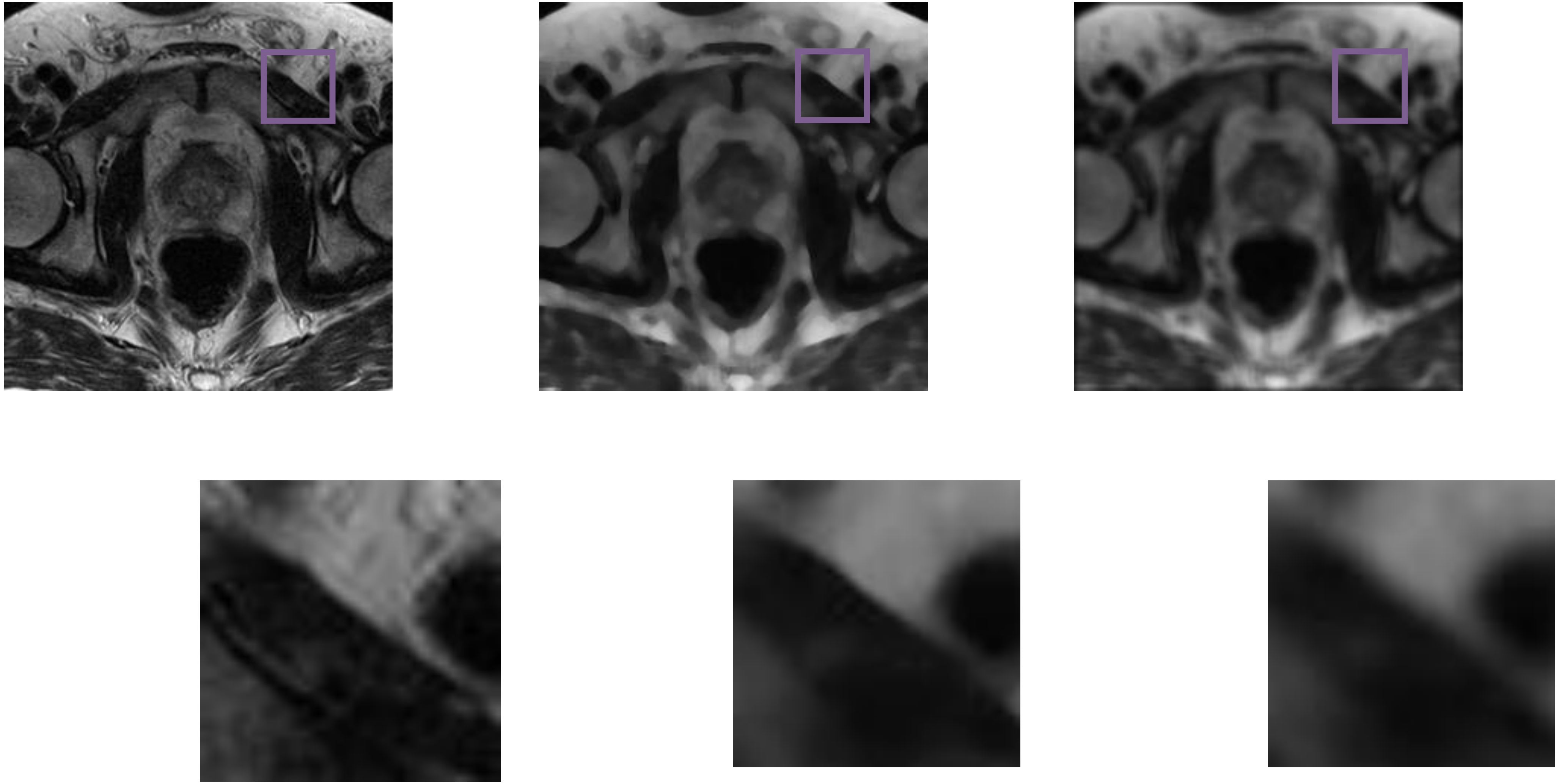


$\sigma = 1$

$\sigma = 3$

Filtering | Edge-Preserving Smoothing

Revisiting median filter



Anisotropic diffusion

Smooth the image, i.e. denoising, without removing useful edge information in the image.

Images are modelled as a time-dependent diffusion process, in which isotropic diffusion can be characterised by the heat equation, $\frac{\partial I}{\partial \mathrm{t}} = \alpha \nabla^2 I = \alpha \left(\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \right)$, where α is the diffusivity coefficient. A numerical solution to solve the heat function is the Forward-Time Central-Space (FTCS) method.

Anisotropic diffusion considers that the diffusion rates differ in different directions at a given time t,

the "flux function" $c_t(x, y)$ as an adaptive diffusivity coefficient in the heat equation, $\frac{\partial I}{\partial \mathrm{t}} = c_t(x, y) \cdot \nabla^2 I + \nabla c \cdot \nabla I$. This diffusivity coefficient function controls the diffusion rate based on how much edges in each direction, therefore it is a function of image gradient. One example diffusivity coefficient is given by:

$$c_t(x, y) = c(\|\nabla I\|) = e^{-\left(\frac{\|\nabla I\|}{K}\right)^2}$$

where K is the user-defined parameter that estimates anisotropic diffusion strength. Using the FTCS numerical scheme, an iterative process can be used to filter the original image. In each iteration, we update the intensity value $I^t(i, j)$ at time t to $I^{t+1}(i, j)$ at time $\mathrm{t} + 1$:

$$\begin{aligned} & I^{t+1}(i, j) \\ & = I^t(i, j) + \lambda \cdot [c_t(i-1, j) \cdot \nabla I(i-1, j) + c_t(i+1, j) \cdot \nabla I(i+1, j) + c_t(i, j-1) \cdot \nabla I(i, j-1) + c_t(i, j+1) \cdot \nabla I(i, j+1)] \end{aligned}$$

where, λ is the time constant, often being set between (0, 0.25] for a stable solution.

Anisotropic diffusion

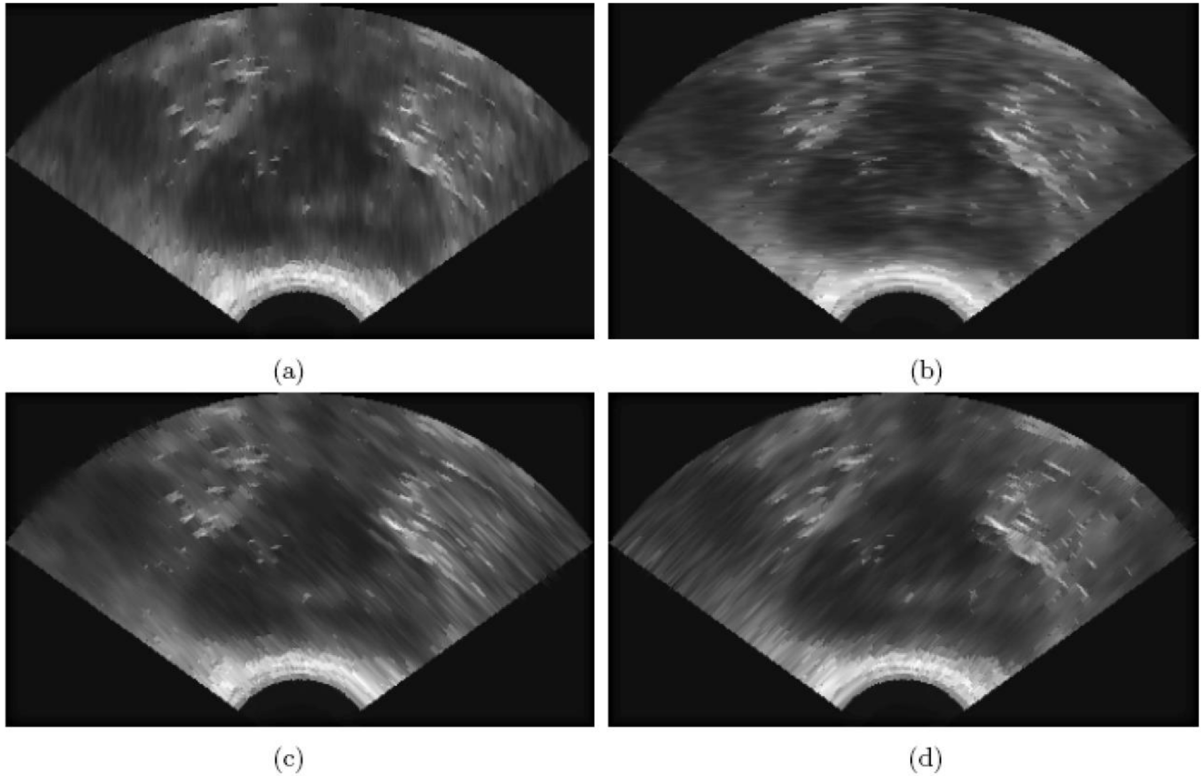


FIGURE 2: Application of extended anisotropic diffusion in eight directions. (a) North-South. (b) East-West. (c) NE-SW. (d) NW-SE.

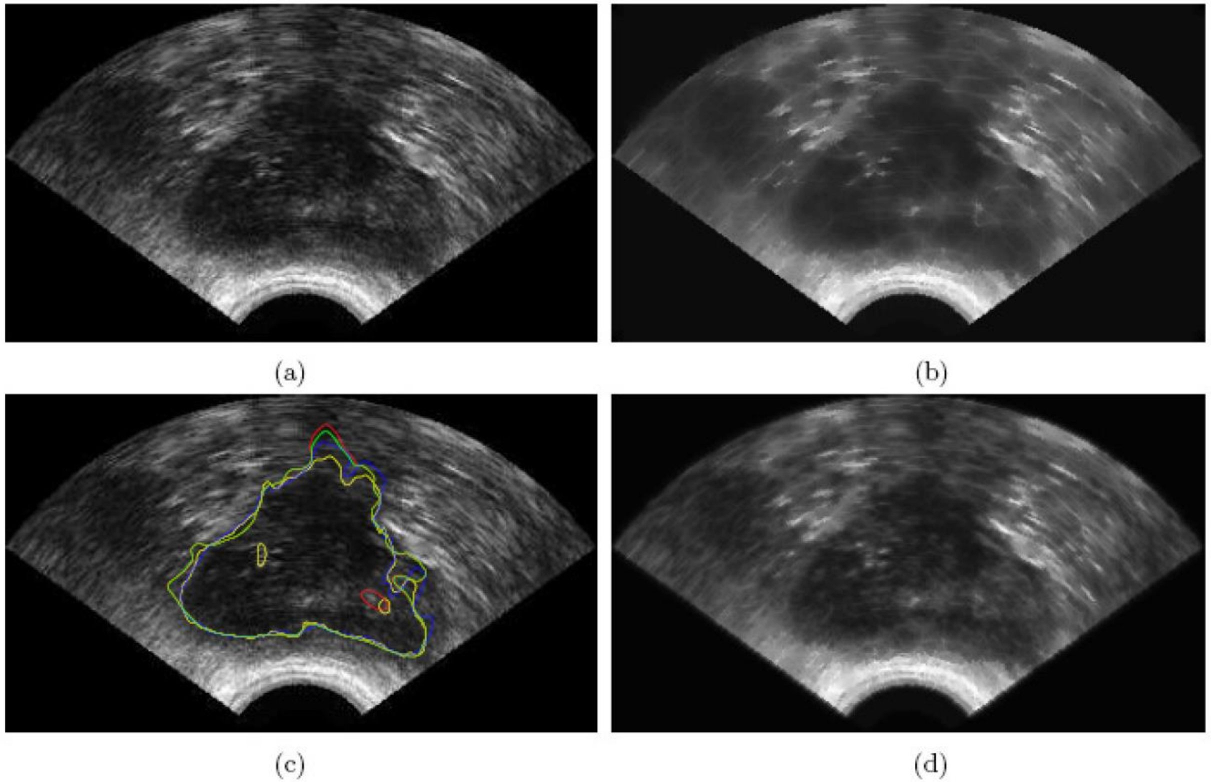


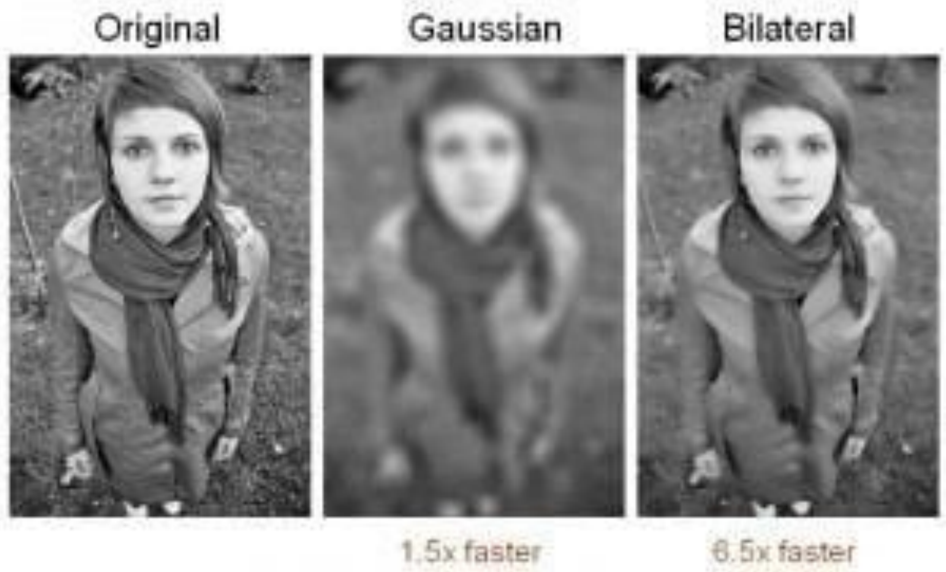
Figure 3: (a) Unprocessed TRUS image. (b) Extracted maximum diffusion over all diffused images. (c) Deformations of different contours over specific diffusion directions. (d) Application of adaptive anisotropic diffusion and extraction of maximum diffusion over all diffused images.

Kachouie, N.N., 2008. Anisotropic diffusion for medical image enhancement. *Int. J. Image Process*, *4*(4), pp.436-443.

Bilateral filter

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$



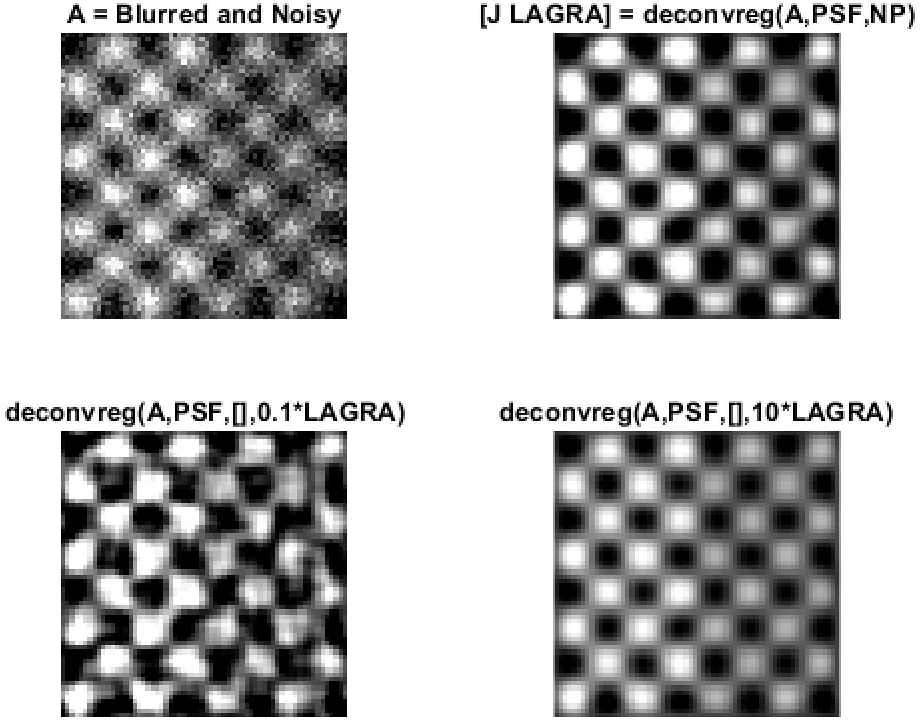
Filtering | Restoring

Motion blurring

https://uk.mathworks.com/help/images/image-deblurring.html#responsive_offcanvas

Deblurring

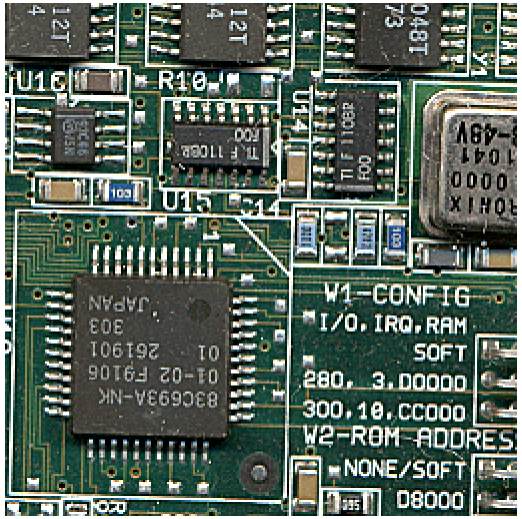
Restoration of Blurred, Noisy Image Using Estimated NSR



https://uk.mathworks.com/help/images/image-deblurring.html#responsive_offcanvas

Deblurring

Original Image

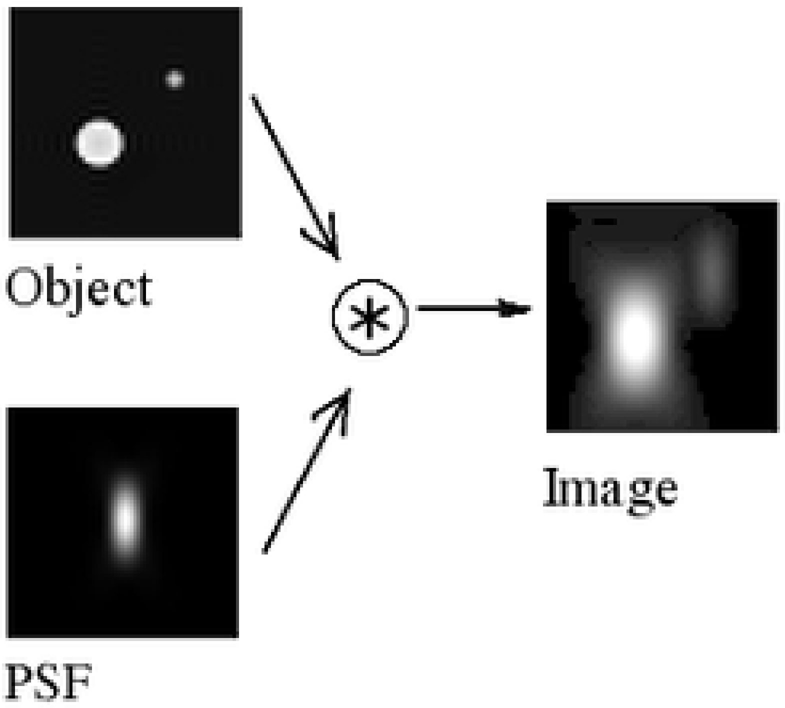


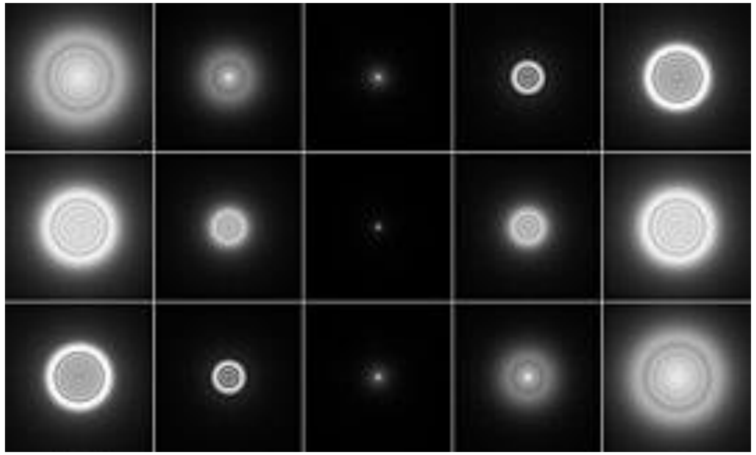
Blurred and Noisy Image

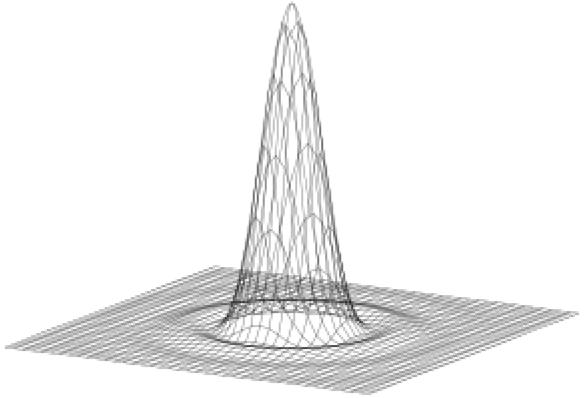
Restored Image

https://uk.mathworks.com/help/images/image-deblurring.html#responsive_offcanvas

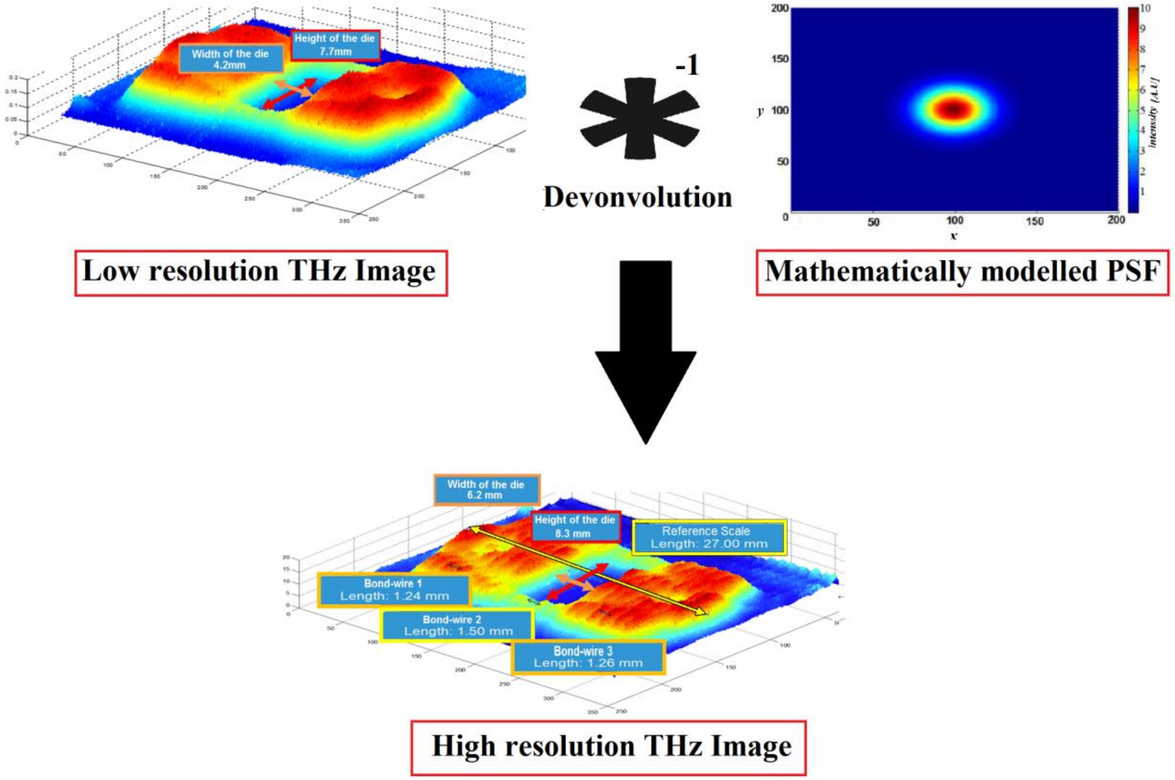
Point spread function

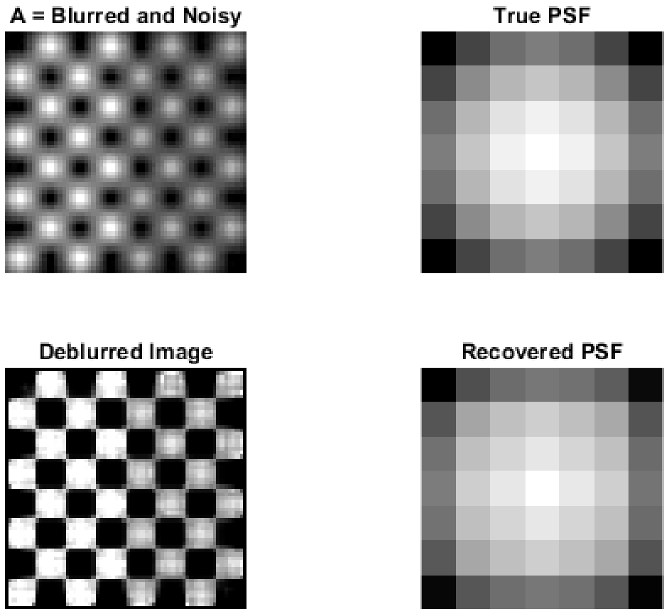






Deconvolution



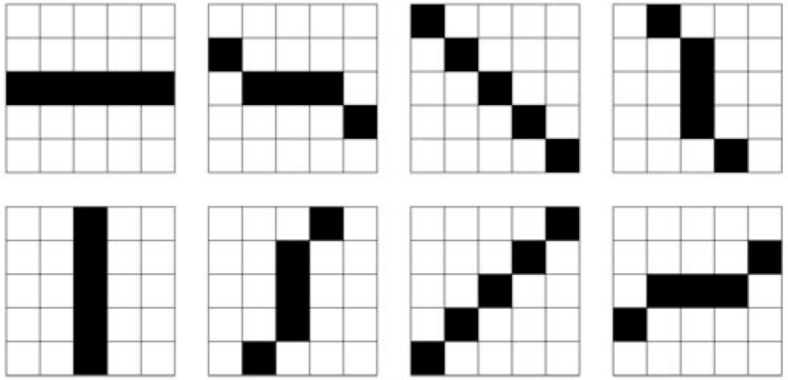


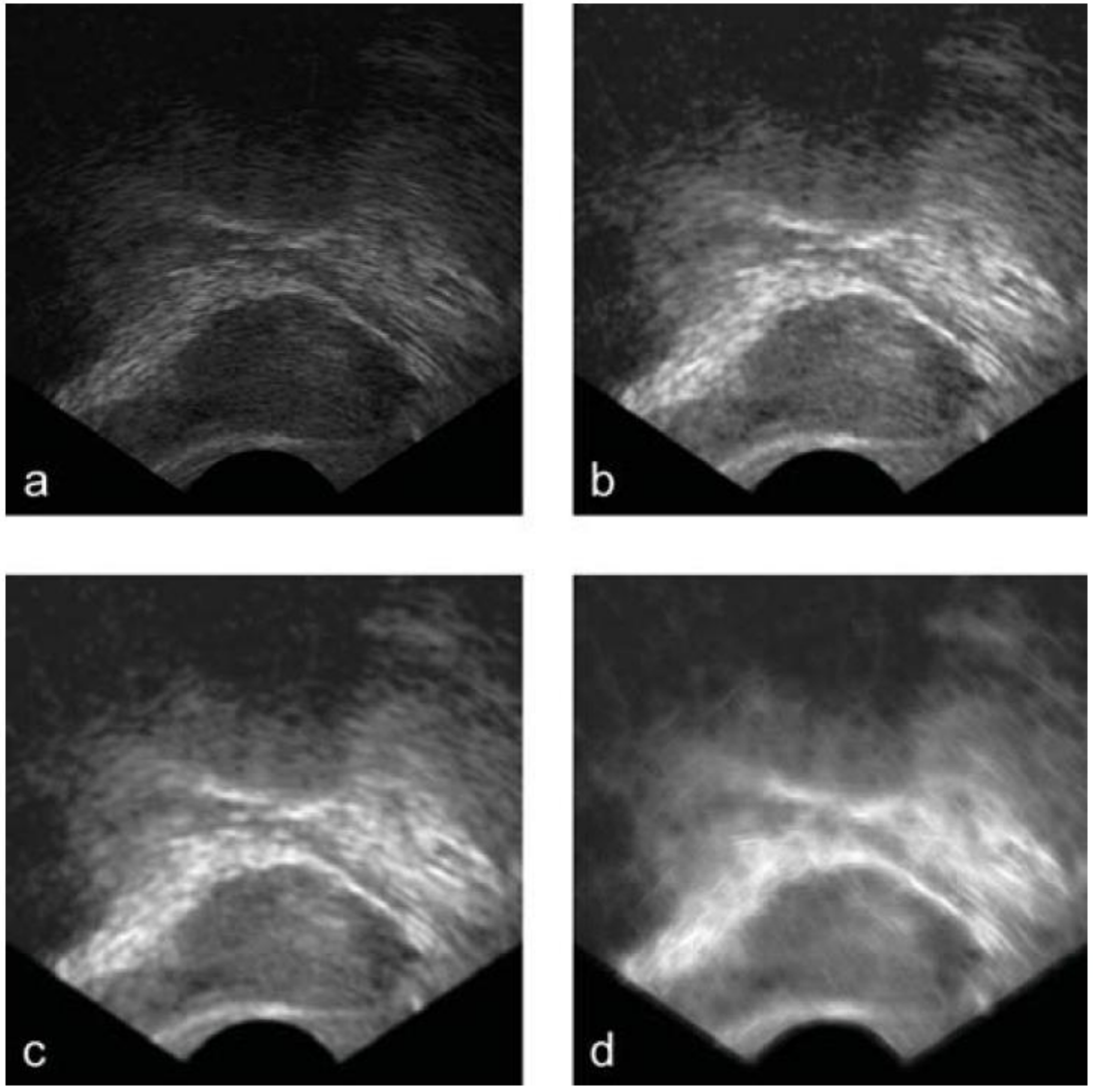
Not transpose convolution!

https://scikit-image.org/docs/dev/auto_examples/filters/plot_deconvolution

Filtering | Local Structuring

The stick filters



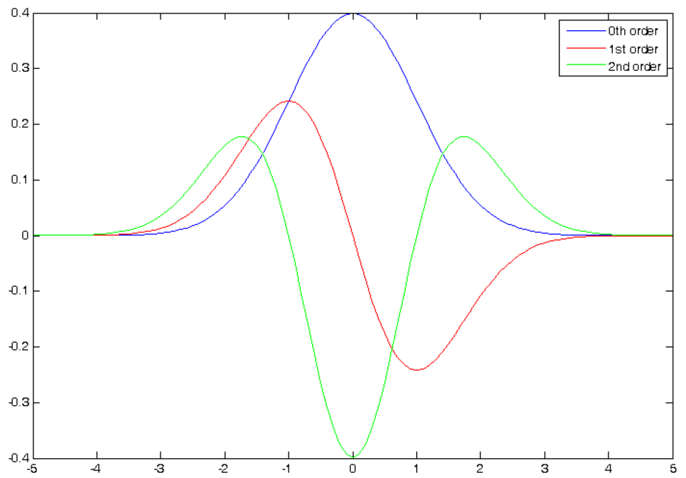


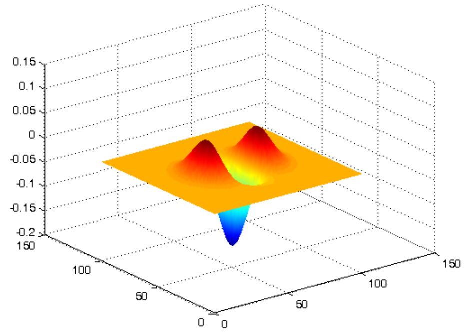
The local Jacobian tensor and Hessian matrix

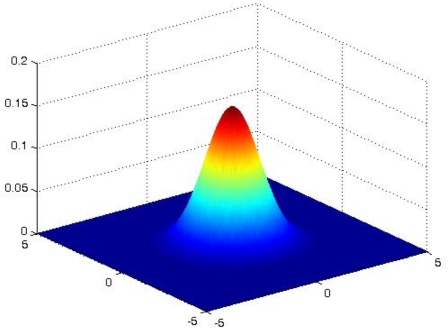
$$\mathbf{J}_I^{2d} = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \qquad \mathbf{J}_I^{3d} = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} & \frac{\partial I}{\partial z} \end{bmatrix}$$

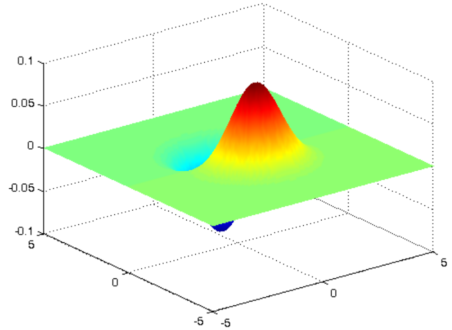
$$\mathbf{T}_I^{2d} = \left(\mathbf{J}_I^{2d}\right)^{\mathrm{T}} \mathbf{J}_I^{2d} = \begin{bmatrix} \frac{\partial I}{\partial x}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\frac{\partial I}{\partial y} \end{bmatrix} \qquad \mathbf{T}_I^{3d} = \left(\mathbf{J}_I^{3d}\right)^{\mathrm{T}} \mathbf{J}_I^{3d} = \begin{bmatrix} \frac{\partial I}{\partial x}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \frac{\partial I}{\partial x}\frac{\partial I}{\partial z} \\ \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\frac{\partial I}{\partial y} & \frac{\partial I}{\partial y}\frac{\partial I}{\partial z} \\ \frac{\partial I}{\partial z}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial z}\frac{\partial I}{\partial y} & \frac{\partial I}{\partial z}\frac{\partial I}{\partial z} \end{bmatrix}$$

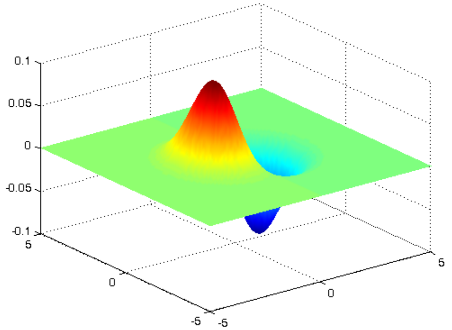
$$\mathbf{H}_I^{2d} = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \qquad \mathbf{H}_I^{3d} = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial x \partial z} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} & \frac{\partial^2 I}{\partial y \partial z} \\ \frac{\partial^2 I}{\partial z \partial x} & \frac{\partial^2 I}{\partial z \partial y} & \frac{\partial^2 I}{\partial z^2} \end{bmatrix}$$

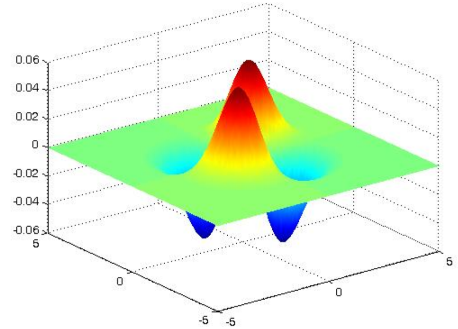


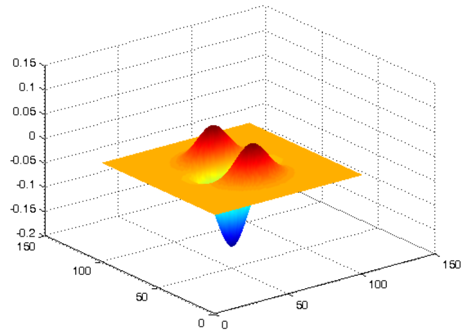








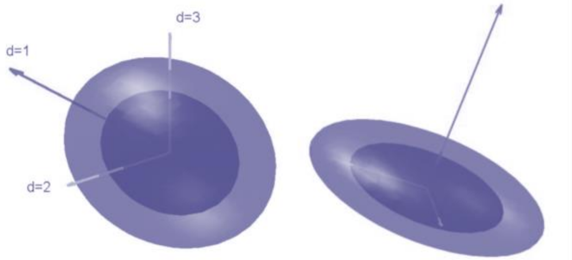




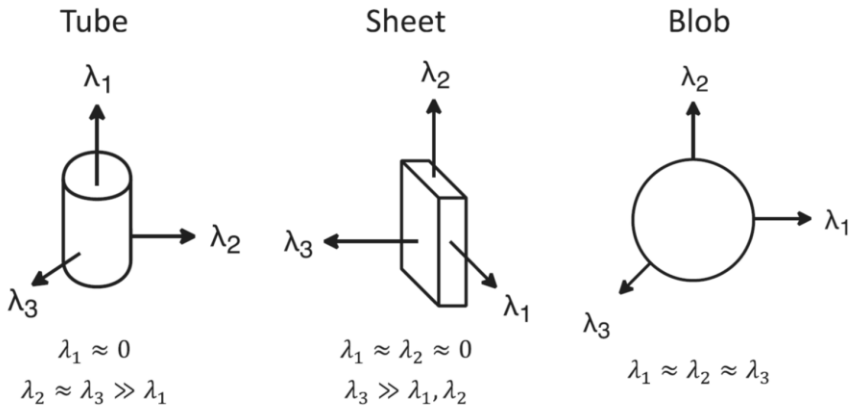
Eigenvalues and eigenvectors of Hessian matrix

$$\mathbf{H}_I^{3d} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}[\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]^{-1}$$

$$|\lambda_\mathbf{1}| \leq |\lambda_\mathbf{2}| \leq |\lambda_\mathbf{3}|$$



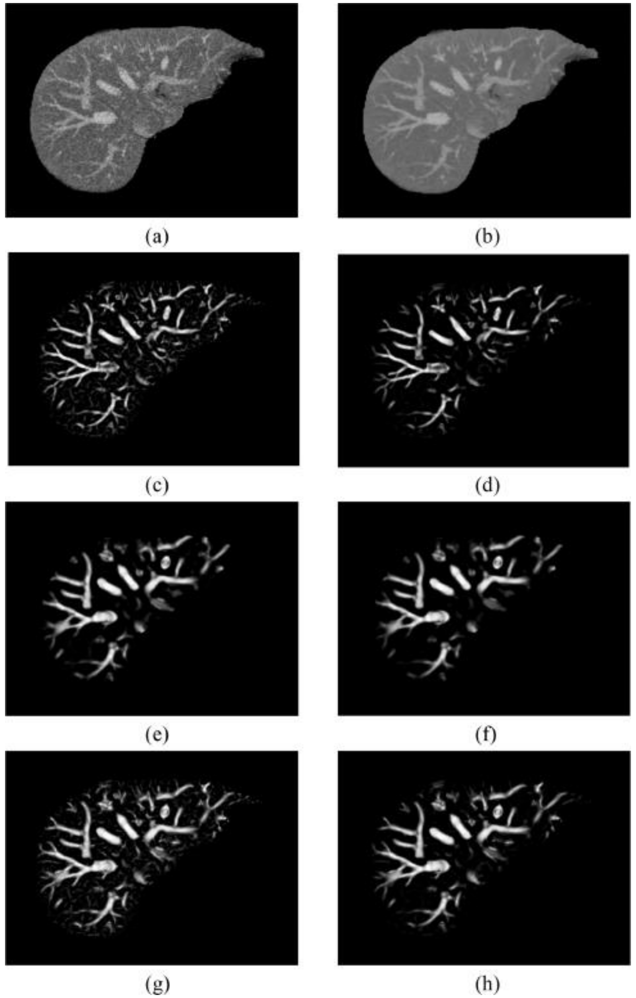
Eigenvalue-based classification

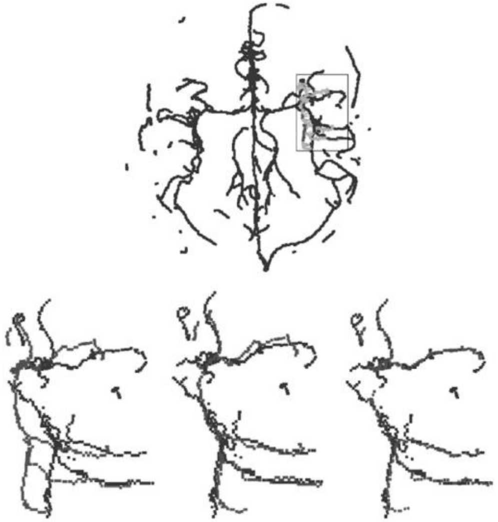


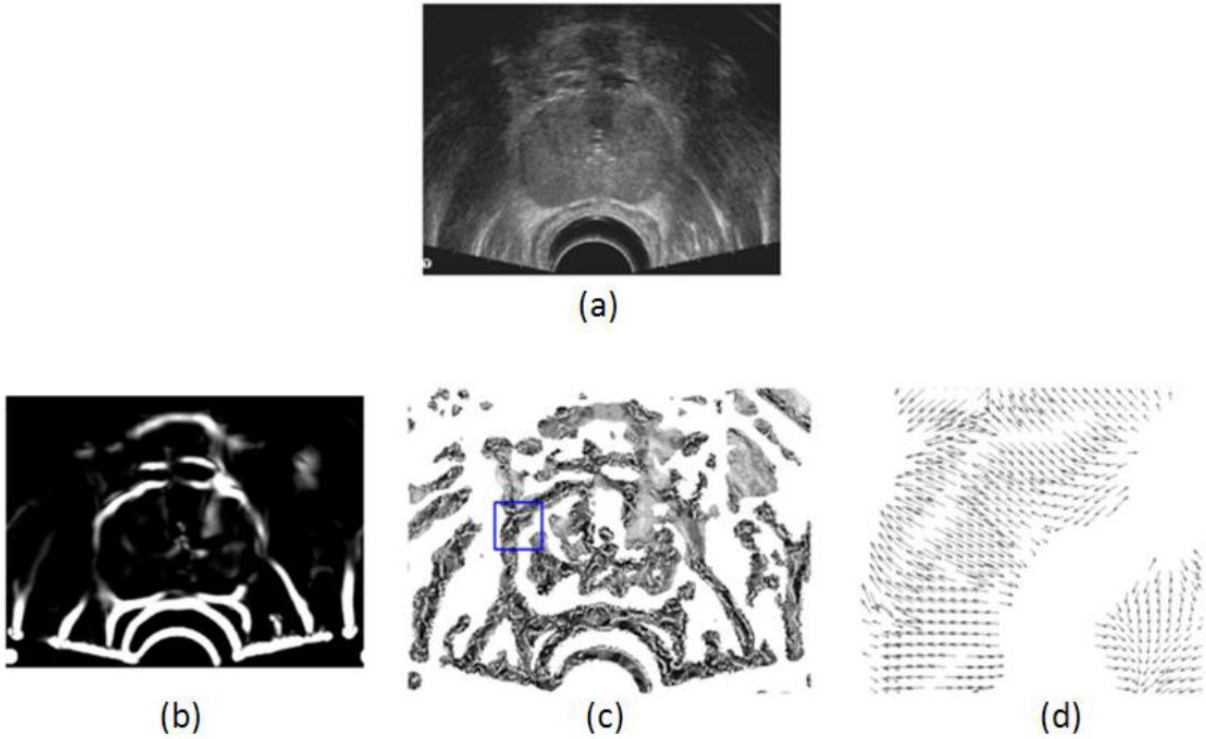
	Tube $\lambda_1 \approx 0$ $\lambda_2 \approx \lambda_3 \gg \lambda_1$	Sheet $\lambda_1 \approx \lambda_2 \approx 0$ $\lambda_3 \gg \lambda_1, \lambda_2$	Blob $\lambda_1 \approx \lambda_2 \approx \lambda_3$
$\mathcal{R}_\mathcal{B} = \frac{\lvert\lambda_1\rvert}{\sqrt{\lvert\lambda_2\lambda_3\rvert}}$	0	0	1
$\mathcal{R}_{sheet} = \mathcal{R}_\mathcal{A} = \frac{\lvert\lambda_2\rvert}{\lvert\lambda_3\rvert}$	1	0	1
$\mathcal{S} = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}$	$\sqrt{2}\lambda_3$	λ_3	$\sqrt{3}\lambda_3$
$R_{blob} = \frac{\lvert(2\lvert\lambda_3\rvert - \lvert\lambda_2\rvert - \lvert\lambda_1\rvert)}{\lvert\lambda_3\rvert}$	1	2	0
$\mathcal{R}_{tube} = \frac{\lvert\lambda_1\rvert}{\lvert\lambda_2\lambda_3\rvert}$	0	$\frac{1}{\lambda_3}$	$\frac{1}{\lambda_3}$
$\mathcal{R}_{noise} = \lvert\lambda_1\rvert + \lvert\lambda_2\rvert + \lvert\lambda_3\rvert$	$2\lvert\lambda_3\rvert$	$\lvert\lambda_3\rvert$	$3\lvert\lambda_3\rvert$
$\mathcal{R}_{bone} = \frac{\lvert\lambda_1\lambda_2\rvert}{\lvert\lambda_3\rvert^2}$	0	0	1

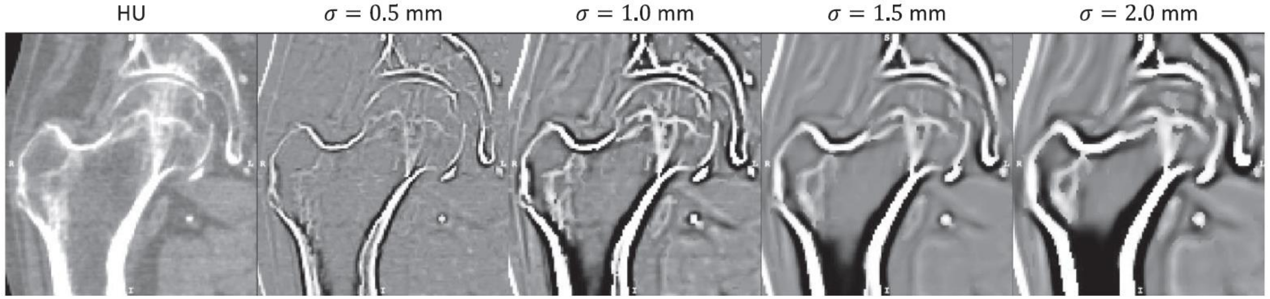
Besler, B.A., Michalski, A.S., Kuczynski, M.T., Abid, A., Forkert, N.D. and Boyd, S.K., 2021. Bone and joint enhancement filtering: Application to proximal femur segmentation from uncalibrated computed tomography datasets. Medical Image Analysis, 67, p.101887.

Vessel-ness and sheet-ness filters









Besler, B.A., Michalski, A.S., Kuczynski, M.T., Abid, A., Forkert, N.D. and Boyd, S.K., 2021. Bone and joint enhancement filtering: Application to proximal femur segmentation from uncalibrated computed tomography datasets. Medical Image Analysis, 67, p.101887.

Filtering | Separable Kernels

Sobel operator

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

$$\mathbf{K} = \mathbf{u} * \mathbf{v}$$

$$\mathbf{I} * \mathbf{K} = \mathbf{I} * \mathbf{u} * \mathbf{v}$$ (associativity)

Computational complexity

$$O(M \times N \times 3 \times 3) \rightarrow O\big(M \times N \times (3 + 3)\big)$$

$$O(M \times N \times m \times n) \rightarrow O(M \times N \times (m + n))$$

Singular value decomposition

$$\mathbf{K} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3] \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{bmatrix} [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]^{\mathrm{T}}$$

If $\mathbf{rank}(\mathbf{K}) = \mathbf{1}$ (test the number of non-singular values / linearly-independent vectors)

$$\mathbf{K} = S_1 \mathbf{u}_1 \mathbf{v}_1^{\mathrm{T}} = S_1 \mathbf{u}_1 * \mathbf{v}_1^{\mathrm{T}}$$ (definition of convolution)

Separable Gaussian kernels

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\cdot\sigma} e^{-\frac{y^2}{2\sigma^2}}$$

So are its derivatives

Filtering | Frequency Multiplication

Convolution theorem

$$f(x,y) * h(x,y) \quad \Leftrightarrow \quad F(u,v)H(u,v)$$

$\boxed{\text{Space convolution = frequency multiplication}}$

Fourier transform

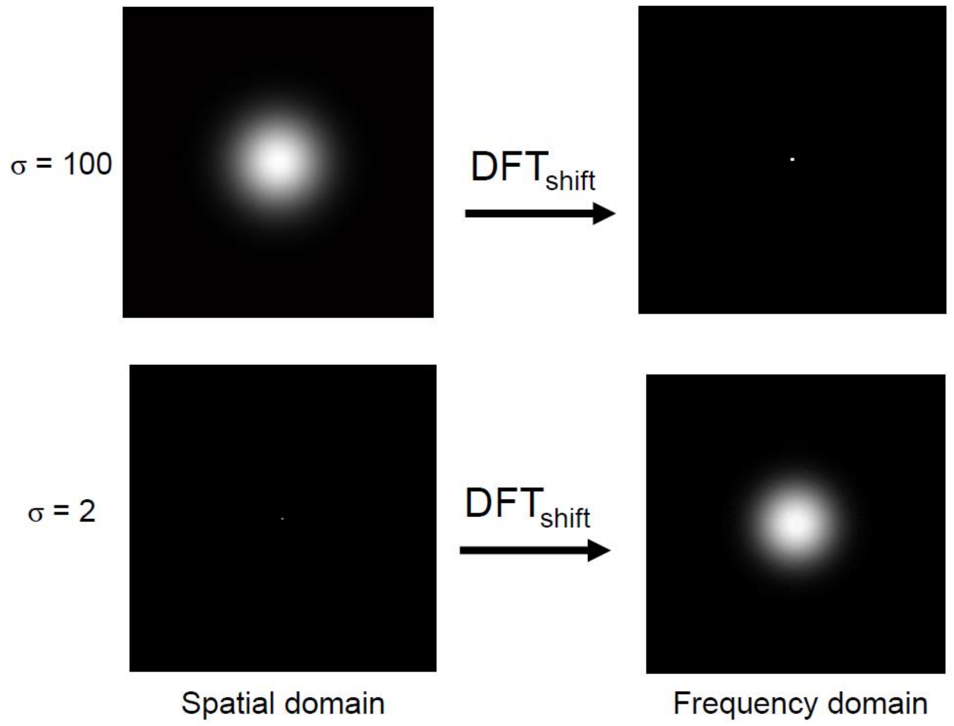
Discrete Fourier transform

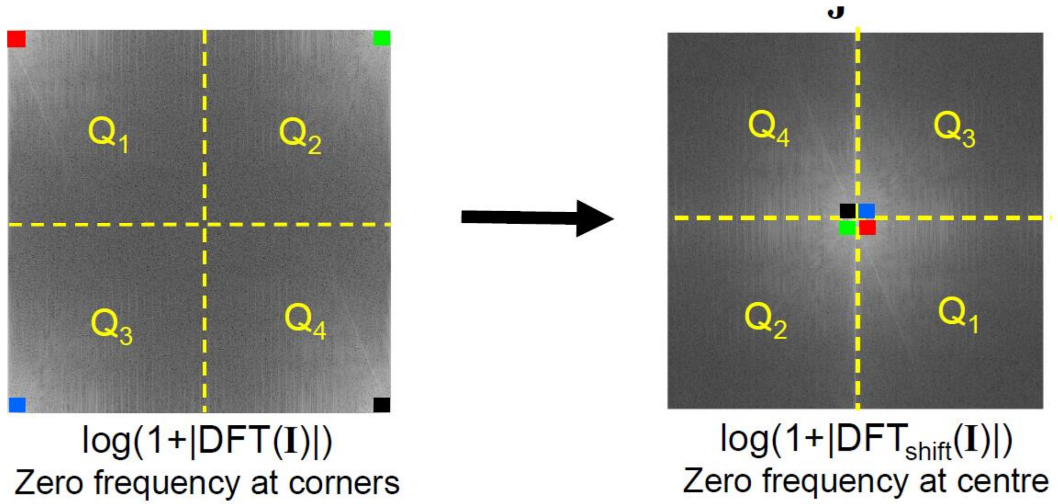
Precision, sensitive to kernel size

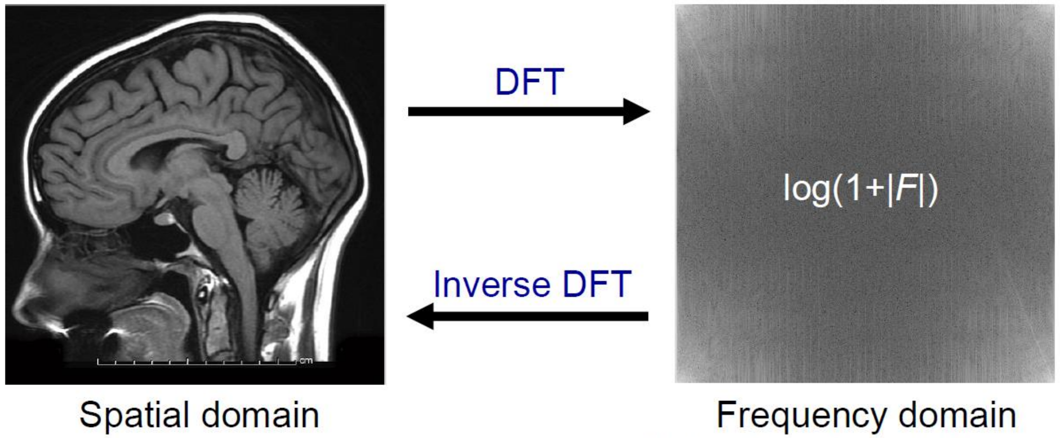
Fast discrete Fourier transform

FFT shift

Inverse fast discrete Fourier transform

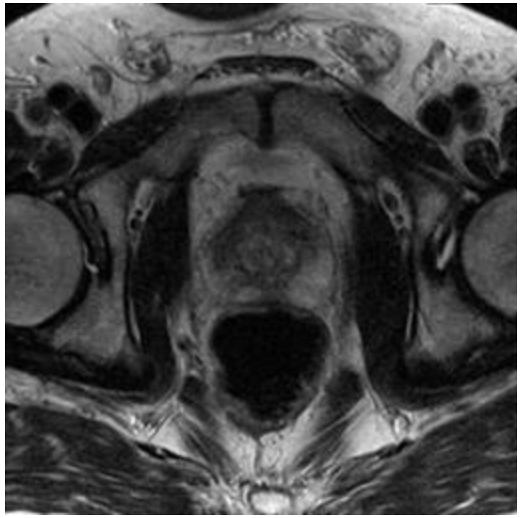


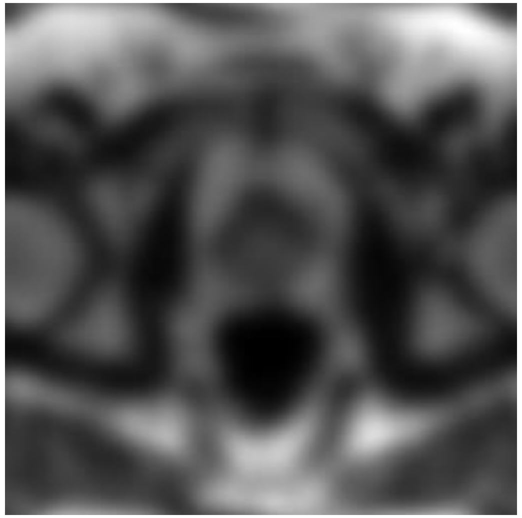




Filtering | Frequency Multiplication

Filtering in frequency domain





script_filtering.py M   script_fft.py U

script_fft.py > ...

```python
import numpy as np
from PIL import Image


# read an image
IMG_FILE = '../data/mri_prostate.dat'
img0 = np.genfromtxt(IMG_FILE,delimiter=',',dtype='uint8')
M, N = img0.shape

# build a Gaussian kernel
s = 0.01  # scale
x, y = np.linspace(-M/2,M/2,M), np.linspace(-N/2,N/2,N)
grid_x, grid_y = np.meshgrid(x, y)
kernel = np.exp(-(grid_x**2+grid_y**2)*s)
kernel = kernel / kernel.sum()  # normalisation

# filtering
img0_fft = np.fft.fft2(img0) # FFT
img0_fft = np.fft.fftshift(img0_fft) # zero-freq. locations
img1_fft = img0_fft * kernel # multiplication in frequency domain
img1 = np.fft.ifft2(img1_fft) # inverse FFT
img1 = np.abs(img1) # real part

# save to files
img1 = (img1-img1.min()) / (img1.max()-img1.min()) *255 # to uint8
Image.fromarray(img1.astype('uint8')).save('fft_s1e-2.png')
```

Filtering | Scale Space

Motivation

Processing images at multiple scales

Hierarchical world

Physics

Biological vision

Implementation

Linear (Gaussian) scale space

Applications

– Multiscale filtering,

e.g. smoothing, edge detection, (variable-size) vessel detection

– Multiscale similarity measures

$$\mathcal{S}_{multiscale} = \frac{1}{Z}\sum_{\sigma} \mathcal{S}(f_\sigma(\mathbf{x}), f_\sigma(\mathbf{y}))$$

- f_σ is a 3D Gaussian filter with an isotropic standard deviation $\boldsymbol{\sigma}$.
- The number of scales Z is application-specific, e.g. $\sigma \in \{0, 1, 2, 4, 8, 16, 32\}$.
- $f_{\sigma=0}$ is equivalent to filtering with a Dirac delta function, i.e. unfiltered images.

– Multiscale image registration*

- Convolution
- Smoothing
- Differentiating
- Edge-preserving smoothing
- Restoring
- Local structuring
- Separable kernels
- Frequency multiplication
- Scale space