Fundamentals of Image Processing

▶ Lecture 5: Image filtering ◀

Master of Computer Science
Sorbonne University

September 2022

Outline

Linear filtering

Linear time-invariant system (LTI)

- Continuous case, 1D
- Operator T applying on a function: $x \mapsto T[x] = y$
- x and $T[x]$ are functions: $t \mapsto x(t)$, $t \mapsto T[x](t)$
- LTI properties:
  1. Linearity: $T\left[\int_{-\infty}^{\infty} c(w)x(t)dw\right] = \int_{-\infty}^{\infty} c(w)T[x](t)dw = \int_{-\infty}^{\infty} c(w)y(t)dw$
  2. Time (translation) invariance: $T[t' \mapsto x(t'-\tau)](t) = T[x(\cdot-\tau)](t) = y(t-\tau)$
- **Impulse response** of the LTI system T: $h(t) = T[\delta](t)$
- Consequence of linearity and time invariance properties:

$$y(t) = T[x](t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = x \star h(t)$$

Linear time-invariant system (LTI)

- Let T be a LTI system, $T[x]$ writes as a convolution between x and the impulse response of T
- Proof:

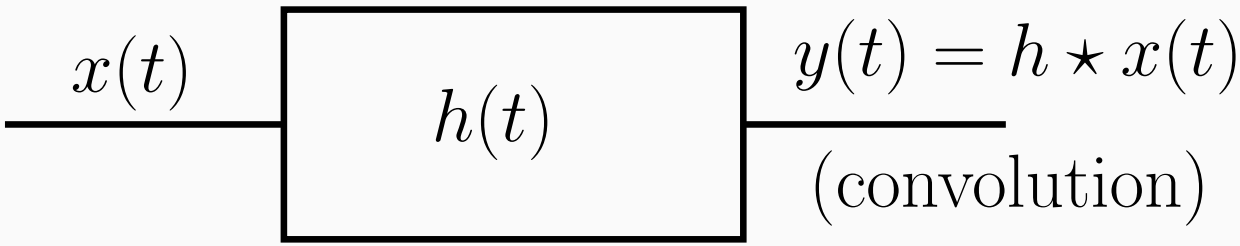
$$
\begin{aligned}
x(t) &= \int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)d\tau \\
T[x](t) &= \int_{-\infty}^{\infty} x(\tau)T[t \mapsto \delta(t-\tau)](t)d\tau && \text{(linearity)} \\
&= \int_{-\infty}^{\infty} x(\tau)T[t \mapsto \delta(t)](t-\tau)d\tau && \text{(time invariance)} \\
&= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \\
&= x \star h(t)
\end{aligned}
$$

Linear time-invariant system (LTI)

- Continuous case, in 1D:

$$y(t) = T[x](t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau = x \star h(t) \qquad (1)$$

- LTI system is fully characterized by its impulse response h
- Output y for a given input signal x: a convolution between x and h



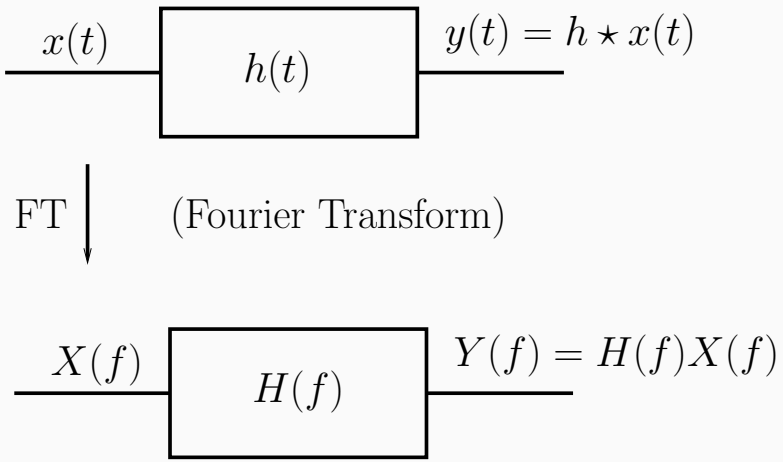
Linear filtering

Linear time-invariant system (LTI), in the frequency domain

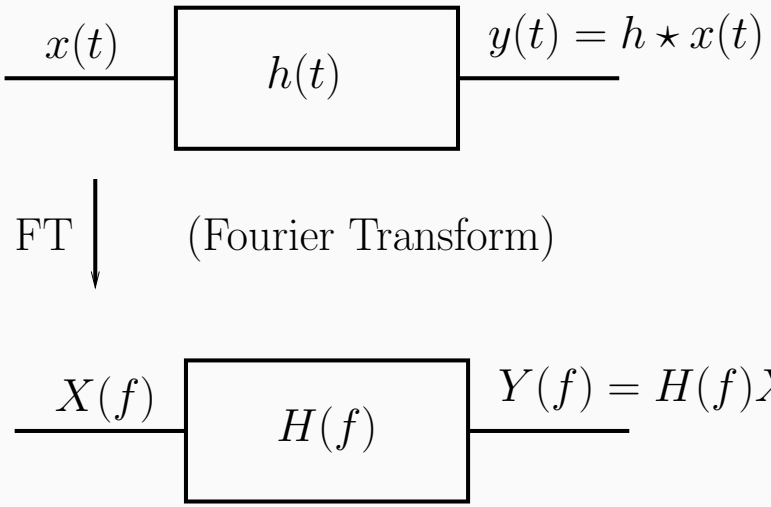
- Continuous case, in 1D: $y(t) = T[x](t) = x \star h(t)$

$$Y(f) = \mathrm{FT}\,[y]\,(f) = \mathrm{FT}\,[x \star h]\,(f) = X(f) \times H(f) \qquad (2)$$

- $h(t)$: impulse response of filter T
- $H(f) = \mathrm{FT}\,[h]\,(f)$: **transfer function** of filter T



Continuous LTI system: conclusion



filtering in temporal domain	filtering in frequency domain
$y(t) = x \star h(t)$	$y(t) = \mathrm{FT}^{-1}\left[X(f) \times H(f)\right](t)$

- Dual operations
- Pro and cons of each representation space?

Linear time-invariant system (LTI)

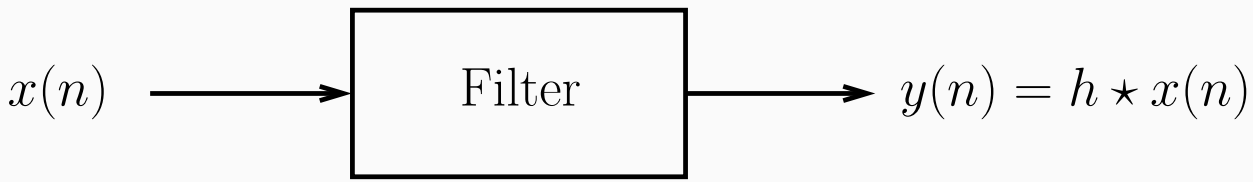
- Discrete case, in 1D
- Operator T: $x(n) \to T[x](n) = y(n)$
- LTI properties:
  1. Linearity:
     $$T\left[\sum_{k=-\infty}^{\infty} c_k x\right](n) = \sum_{k=-\infty}^{\infty} c_k T[x](n) = \sum_{k=-\infty}^{\infty} c_k y(n)$$
  2. Time invariance: $T[n \mapsto x(n-k)](n) = y(n-k)$
- $h(n) = T[\delta](n)$: impulse response of T with
  $$\delta(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases}$$ the discrete Dirac function
- Consequence: $y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x \star h(n)$
- Similar proof starting with: $x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$

Linear time-invariant system (LTI)

- Discrete case, in 1D

$$y(n) = T[x](n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x \star h(n) \quad (3)$$

- Discrete LTI system: fully characterized by the impulse response h
- Output y for an discrete input signal x: a discrete convolution between x and h

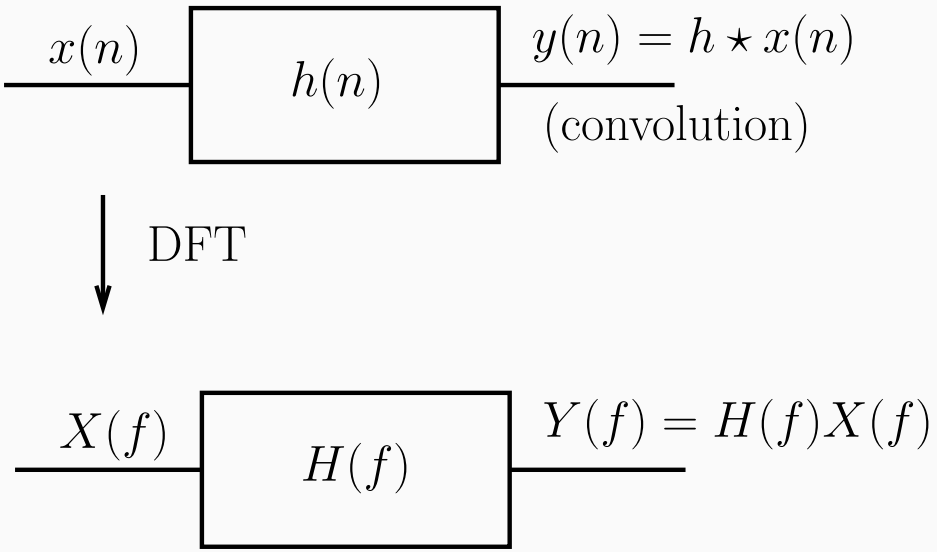


Linear time-invariant system (LTI)

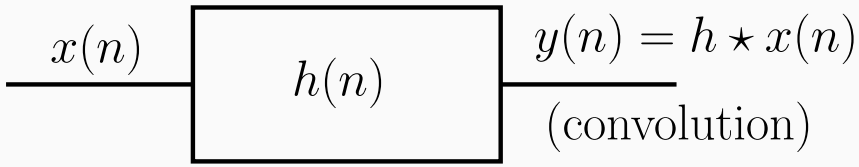
- Discrete case, in 1D: $y(n) = x \star h(n)$
- X: discrete Fourier transform of x

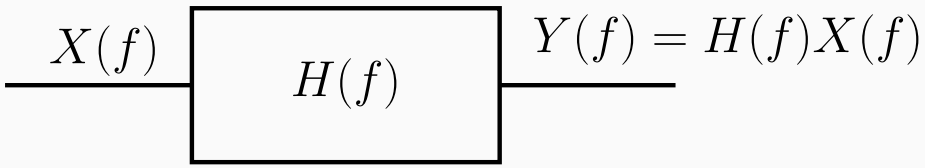
$$Y(f) = \mathrm{DFT}\,[y]\,(f) = \mathrm{DFT}\,[x \star h]\,(f) = X(f) \times H(f) \quad (4)$$

- $H = \mathrm{DFT}\,[h]$: transfer function of h



LTI system: conclusion





Time domain	Frequency domain
$y = x \star h$	$y = \mathrm{DFT}^{-1}[X \cdot H]$

- Dual operations
- Pros and cons for each space representation? Algorithmic complexity?

Linear translation-invariant system (LTI)

- Discrete case, in 2D
- Operator T on a 2D signal (image) $(n, m) \mapsto x(n, m)$
  $T : x \mapsto T[x] = y$
- LTI properties:
  1. Linearity:
     $$T\left[\sum_{k=-\infty}^{\infty} c_k x\right](n, m) = \sum_{k=-\infty}^{\infty} c_k T[x](n, m) = \sum_{k=-\infty}^{\infty} c_k y(n, m)$$
  2. Translation invariance:
     $$T[(n, m) \mapsto x(n-k, m-l)](n, m) = y(n-k, m-l)$$
- $h(n, m) = T[\delta](n, m)$: impulse response
- Consequence:
  $$T[x](n, m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k, l) h(n-k, m-l) = x \star h(n, m)$$

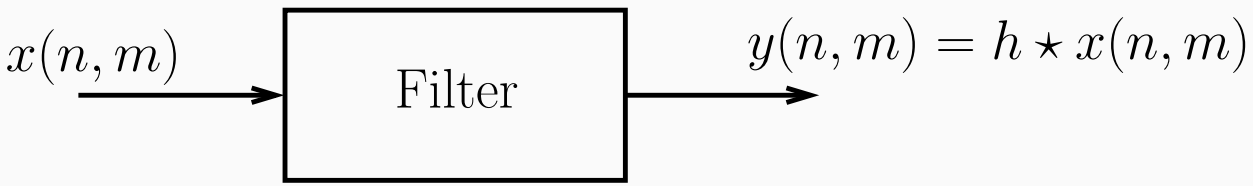
Linear translation-invariant system (LTI)

- Discrete case, in 2D: $y(n,m) = T[x](n,m)$

$$y(n,m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x(k,l)h(n-k,m-l) \tag{5}$$

$$y = x \star h$$

- LTI system: fully characterized by the impulse response $h(n,m)$
- Output $y(n,m)$ for any input $x(n,m)$: convolution between $x(n,m)$ and $h(n,m)$



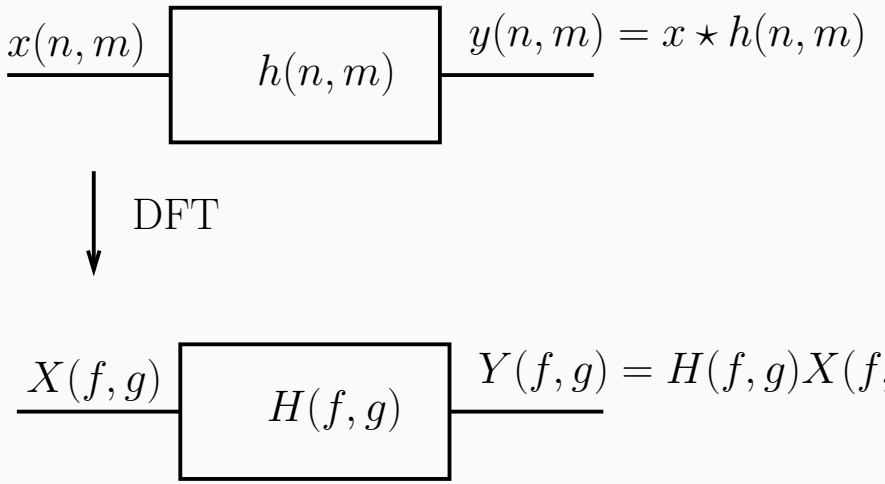
Linear translation-invariant system (LTI)

- Discrete case, in 2D: $y(n, m) = x \star h(n, m)$
- $X(f, g)$: discrete Fourier transform of $x(n, m)$

$$\begin{aligned} Y(f, g) &= \mathrm{DFT}\left[y\right](f, g) = \mathrm{DFT}\left[x \star h\right](f, g) \\ &= X(f, g) \times H(f, g) \end{aligned} \tag{6}$$

- $H(f, g) = \mathrm{DFT}\left[h\right](f, g)$: transfer function of h

LTI system: conclusion



Spatial domain	Frequency domain
$y(n,m) = x \star h(n,m)$	$y(n,m) = TFD^{-1}\left[X \times H\right](n,m)$

- Dual operations
- Pro and cons of each representation space? Algorithmic complexity?

Outline

Spatial filtering

Why filter an image?

- To reduce/remove the noise (as seen in this lecture), or irrelevant details
- To detect edges (discussed in the next lecture)
- Case of (discrete) linear space-invariant filter: a convolution between the image to process, f, and a filter h, also called convolution mask, or convolution kernel
- Linear combination of **neighboring pixel** values of image f, giving a filtered image f'
- $\boxed{\cdot \star h}$ is an operator applying on each pixel of f

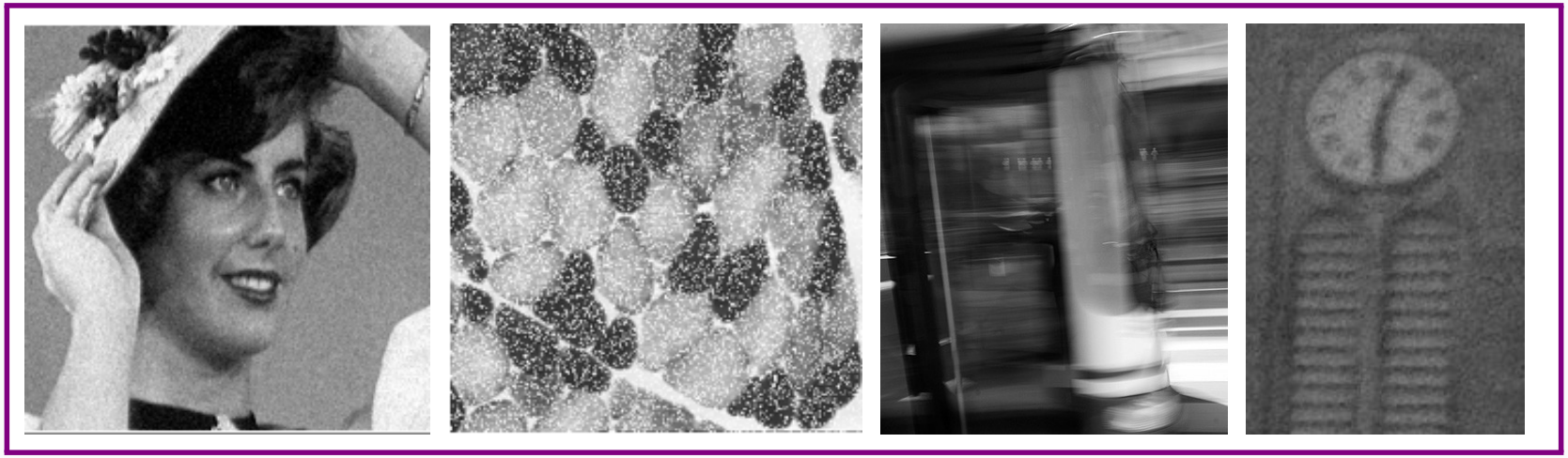
Noise reduction

Definition (noise)

- Undesirable and random process (according to a known or unknown probability distribution) occurring during the acquisition and due to various origins (lighting conditions, sensor...)
- Linear filtering can deal with **additive** noise. The noisy image I_n writes $I_n(i,j) = I(i,j) + n(i,j)$
- Typical examples: Gaussian and impulse noises
  - Gaussian: $I_n(x,y) = I(x,y) + \mathcal{N}(0,\sigma)$
  - Impulse noise (salt-and-pepper noise) of order n: select randomly n pixels and set their values to black or white
    - noise characterized by the percentage of noisy pixels
- Other common noises: blur (convolution noise), grain (multiplicative noise)

Noise reduction

Examples of noisy images

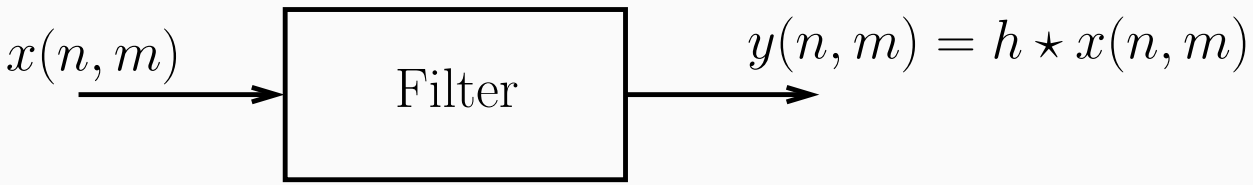


- Fundamental hypothesis for noise reduction:
  - Useful signal and noise have different frequency components
  - Useful signal $\leftrightarrow$ low frequencies
  - Noise $\leftrightarrow$ high frequencies

Linear spatial filtering: convolution

- Linear spatial filtering: convolution with the impulse response
- In the spatial domain:

$$T[x](n,m) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} x(k,l)h(n-k,m-l) = x \star h(n,m)$$



- How does it work for 2D discrete signals?

2D discrete convolution product

- As in continuous case (see lecture 3, slide 25), discrete convolution is commutative, distributive and associative, and this also holds in 2D
- Commutativity: discrete 2D convolution between an image $f(i,j)$ and a filter $h(i,j)$ also writes:

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} f(i-n, j-m) h(n,m)$$

- Two cases for h:
  1. **Finite** impulse response filters (FIR)
  2. **Infinite** impulse response filters (IIR)

FIR filter vs IIR filter

1. IIR filter: h has an unbounded support
2. FIR filter: h has a bounded support,
   $\exists \eta \mid \forall (n, m), \max(|n|, |m|) > \eta,\ h(n, m) = 0$
   - In practice h is chosen centered at (0,0) with odd size d, e.g.:

$$h(n, m) = 0 \quad \text{for } |n| > \frac{d-1}{2} \text{ and } |m| > \frac{d-1}{2}$$

For such a kernel h, the convolution writes:

$$f'(i, j) = (f \star h)(i, j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, j-m) h(n, m)$$

Convolution filtering

FIR filter

- Example with a 3×3 filter ($d = 3$)

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-1}^{1} \sum_{m=-1}^{1} f(i-n, j-m) h(n,m)$$

$$\begin{aligned} f'(i,j) &= h(1,1)f(i-1,j-1) + h(1,0)f(i-1,j) + h(1,-1)f(i-1,j+1) \\ &+ h(0,1)f(i,j-1) + h(0,0)f(i,j) + h(0,-1)f(i,j+1) \\ &+ h(-1,1)f(i+1,j-1) + h(-1,0)f(i+1,j) + h(-1,-1)f(i+1,j+1) \end{aligned}$$

- Let $g(n,m) = h(-n,-m)$, the convolution writes as a weighted sum with g:

$$\begin{aligned} f'(i,j) &= g(-1,-1)f(i-1,j-1) + g(-1,0)f(i-1,j) + g(-1,1)f(i-1,j+1) \\ &+ g(0,-1)f(i,j-1) + g(0,0)f(i,j) + g(0,1)f(i,j+1) \\ &+ g(1,-1)f(i+1,j-1) + g(1,0)f(i+1,j) + g(1,1)f(i+1,j+1) \end{aligned}$$

Convolution filtering

Algorithmic view of convolution with a FIR filter
Let h be a 2D FIR filter.

At pixel $p = (i, j)$:

1. Signal reversal (apply a central symmetry with origin as center): $h(n, m) \Rightarrow h(-n, -m) = g(n, m)$
2. Centering h on pixel p to filter
3. Compute a weighted sum between pixels values f and filter values g

Convolution filtering

Example with a 3×3 kernel

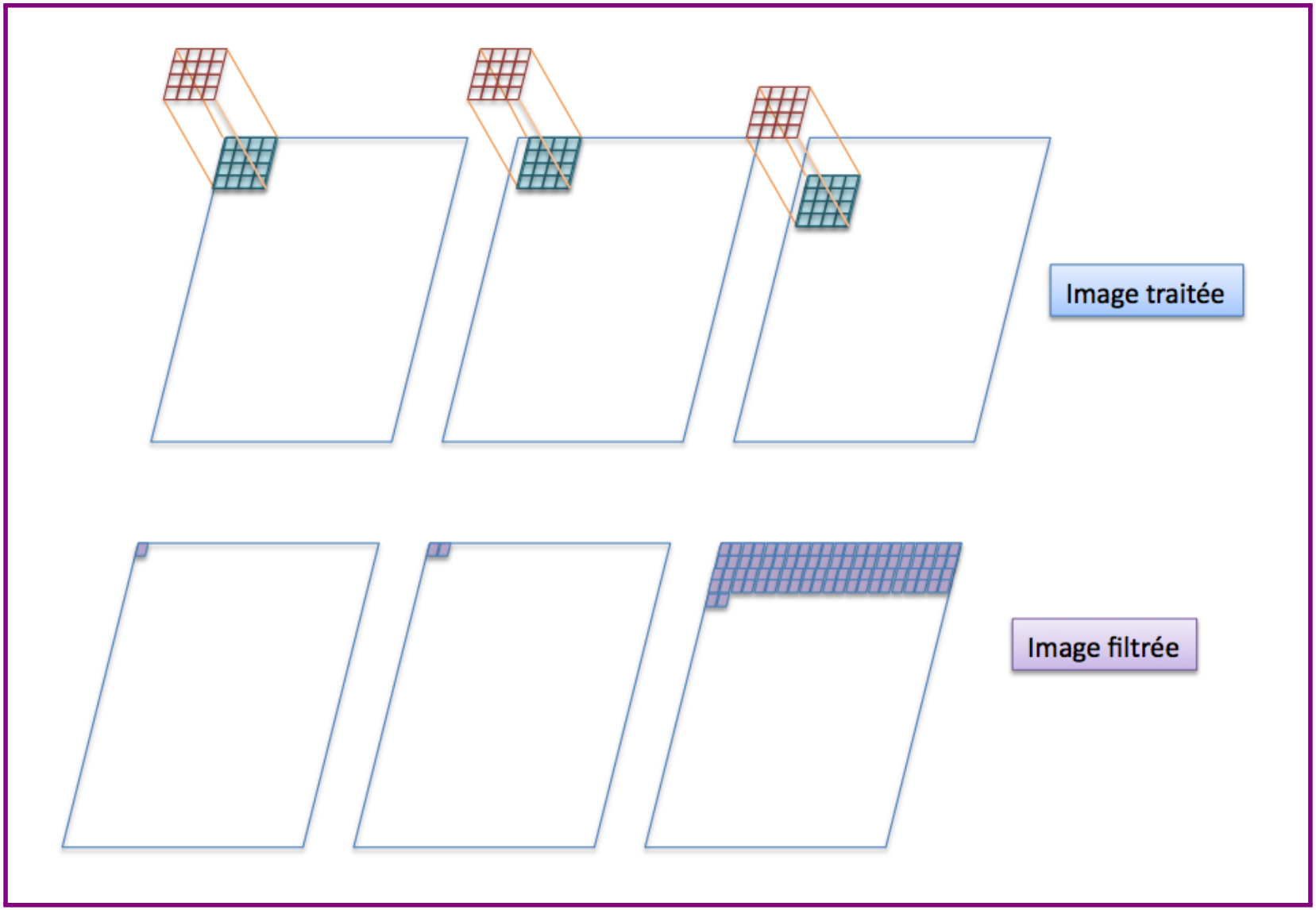
$$h = \begin{pmatrix} w_9 & w_8 & w_7 \\ w_6 & w_5 & w_4 \\ w_3 & w_2 & w_1 \end{pmatrix} \longrightarrow g = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$$

- Filtered value at pixel (i, j) is given by:

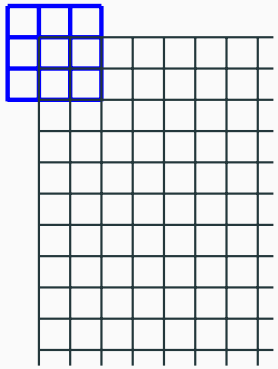
$$\begin{aligned} f'(i,j) &= w_1 f(i-1,j-1) + w_2 f(i-1,j) + w_3 f(i-1,j+1) \\ &+ w_4 f(i,j-1) + w_5 f(i,j) + w_6 f(i,j+1) \\ &+ w_7 f(i+1,j-1) + w_8 f(i+1,j) + w_9 f(i+1,j+1) \end{aligned}$$

- Filters designed to remove noise should be normalized, $\sum_i w_i = 1$, in order to preserve the image average

Sliding window principle



Border conditions



- How to apply convolution when the mask is not inside the image domain?
  - **Linear convolution**: image is padded with null values: zero-padding (image and filter have bounded support)
  - **Circular convolution**: image is repeated in two directions (bi-periodic function with unbounded support)
  - Padding with repeated border values
  - Padding with symmetry
  - Do not process borders (and reduce the image domain)

Convolution filtering: filter examples

Smoothing linear filters

- Assumption: the value of a pixel is similar to the value of its neighbors
- If the image is noisy, and under the previous assumption, a **local averaging** attenuates the noise: it is called **smoothing**
- Averaging filter, in a neighborhood of size $d \times d$, is defined by:

$$f'(i,j) = \frac{1}{d^2} \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i+n, j+m) \qquad (7)$$

- Consider an averaging filter of the same size than f: $f'(0,0)$ is the integral of f (with zero-padding)
  $\hookrightarrow$ **Smoothing is analogous to a local integration of the image**

Smoothing with averaging filter

Example

- Filter of size $d = 3$:

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

  Because filter h is symmetric, Equation (7) (previous slide) writes:

$$f'(i,j) = f \star h(i,j)$$

- In the general case, for a size d, coefficients of filter h write: $w_i = \frac{1}{d^2}$ (and then $\sum_i w_i = 1$)
- Effects of smoothing increase with the size d of the filter: smallest details (scales) disappear.

Smoothing with averaging filter: example

Figure 1: Original image

Smoothing with averaging filter: example

Figure 2: $d = 3$

Smoothing with averaging filter: example

Figure 3: $d = 5$

Smoothing with averaging filter: example

Figure 4: $d = 7$

Smoothing with averaging filter: example

Figure 5: $d = 9$

Smoothing with averaging filter: example

Figure 6: $d = 11$

Smoothing with averaging filter

Spatial filtering vs Frequency filtering

temporal/spatial	frequency
$y(n,m) = x \star h(n,m)$	$y(n,m) = TFD^{-1}\left[X(f,g) \times H(f,g)\right]$

- What is the transfer function of the averaging filter? What type of frequency filtering?
- Lowpass filter, ideal filter? (see Section 2, Filtering in the frequency domain)
- **See practical work**

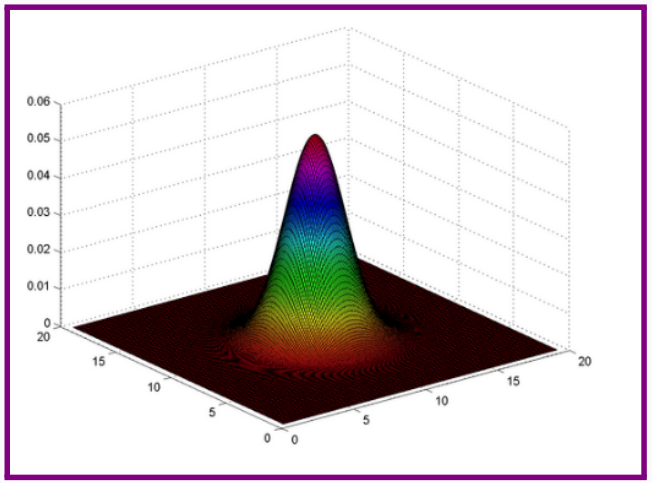
Convolution filtering: example of filters

Gaussian smoothing

- Definition: Gaussian kernel, centered, with standard deviation σ:

$$g_\sigma(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

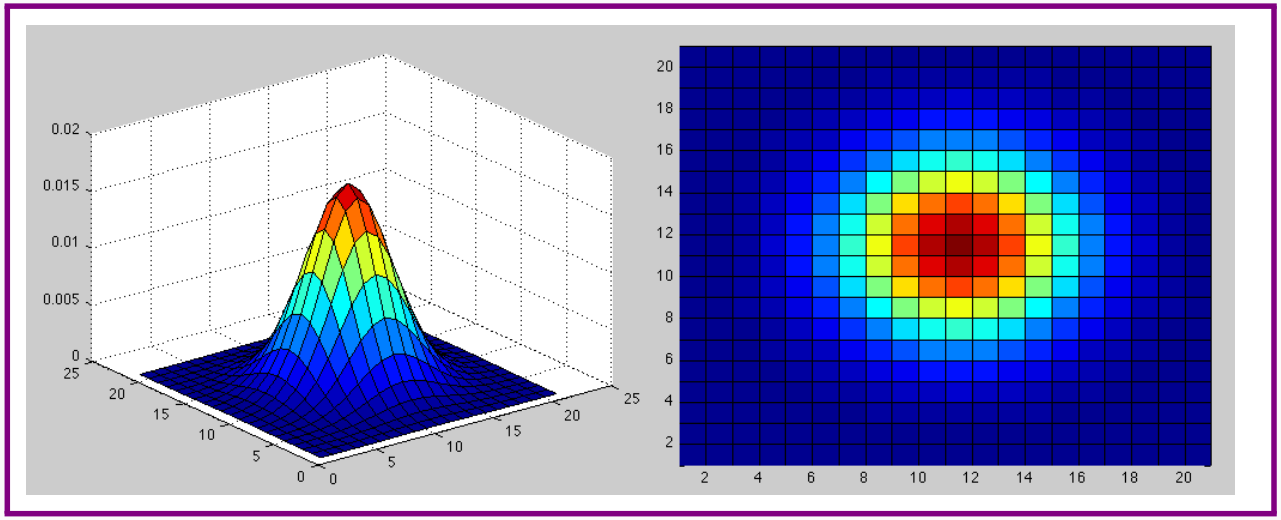
- Smoothing with a **weighted** averaging: the weight of a neighboor depends on the distance to the processed pixel



Gaussian smoothing

From continuous to discrete case

- The discrete Gaussian kernel is defined by a sampling of the continuous Gaussian function



Gaussian smoothing

Determination of filter coefficients

- Size (width and height) of filter: driven by the standard deviation σ:
  - We admit that g_σ is almost null beyond a distance of $\lceil 3\sigma \rceil$ from the center
  - To simplify, we chose a squared domain: the width kernel is $2\lceil 3\sigma \rceil + 1$ and the kernel has $(2\lceil 3\sigma \rceil + 1)^2$ coefficients.
- $\sigma \to 0$: g_σ tends towards Dirac impulse function that is the neutral element of convolution: few or no effects
- $\sigma \to \infty$: g_σ tends towards an average filter of infinite size, effects of smoothing are more and more strong, the image becomes blurred
- For higher values of σ, more image details (scales) are lost

$\hookrightarrow$ Compromise between amount of noise removed and image quality

Gaussian smoothing

Numeric example with $\sigma = 0.625$

- Filter width (and height): $2\lceil 3\sigma \rceil + 1 = 5$
- We obtain the following kernel:

$$h = 0.4 \times 10^{-2} \times \begin{pmatrix} 0.03 & 0.16 & 5.98 & 0.16 & 0.03 \\ 0.16 & 7.7 & 27.8 & 7.7 & 0.16 \\ 5.98 & 27.8 & 100 & 27.8 & 5.98 \\ 0.16 & 7.7 & 27.8 & 7.7 & 0.16 \\ 0.03 & 0.16 & 5.98 & 0.16 & 0.03 \end{pmatrix}$$

Gaussian smoothing: from smallest to largest scales

original image

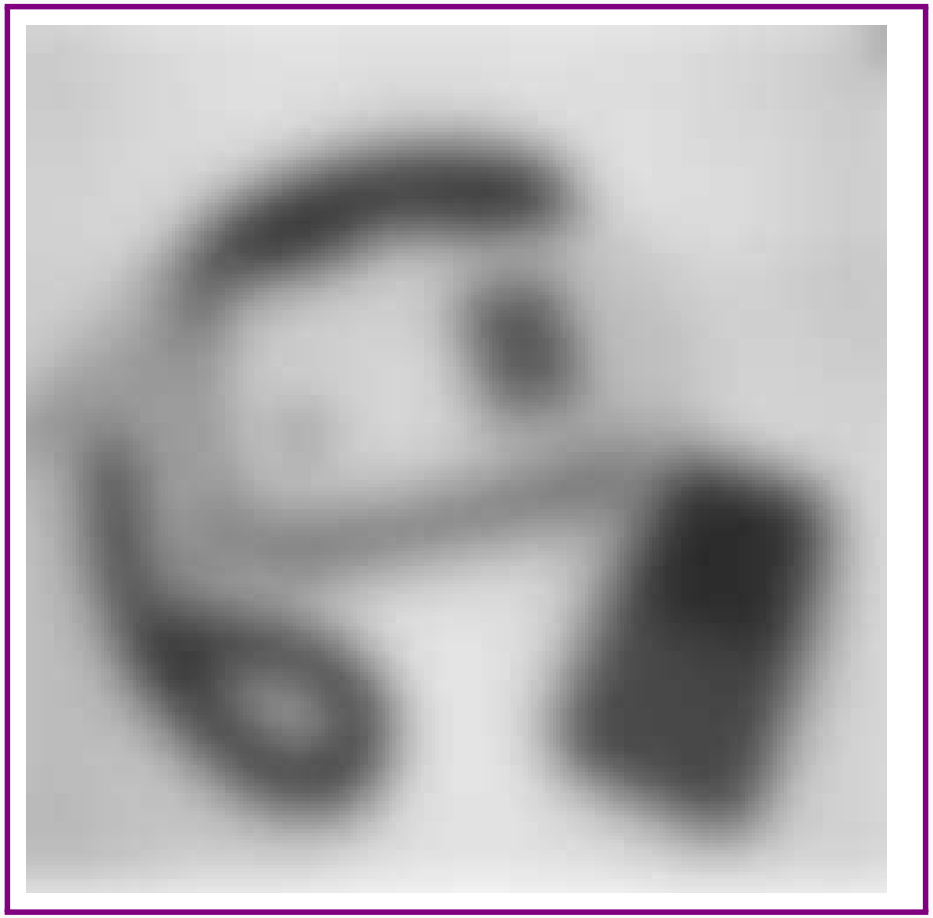
Gaussian smoothing: from smallest to largest scales

$\sigma = 2$

Gaussian smoothing: from smallest to largest scales

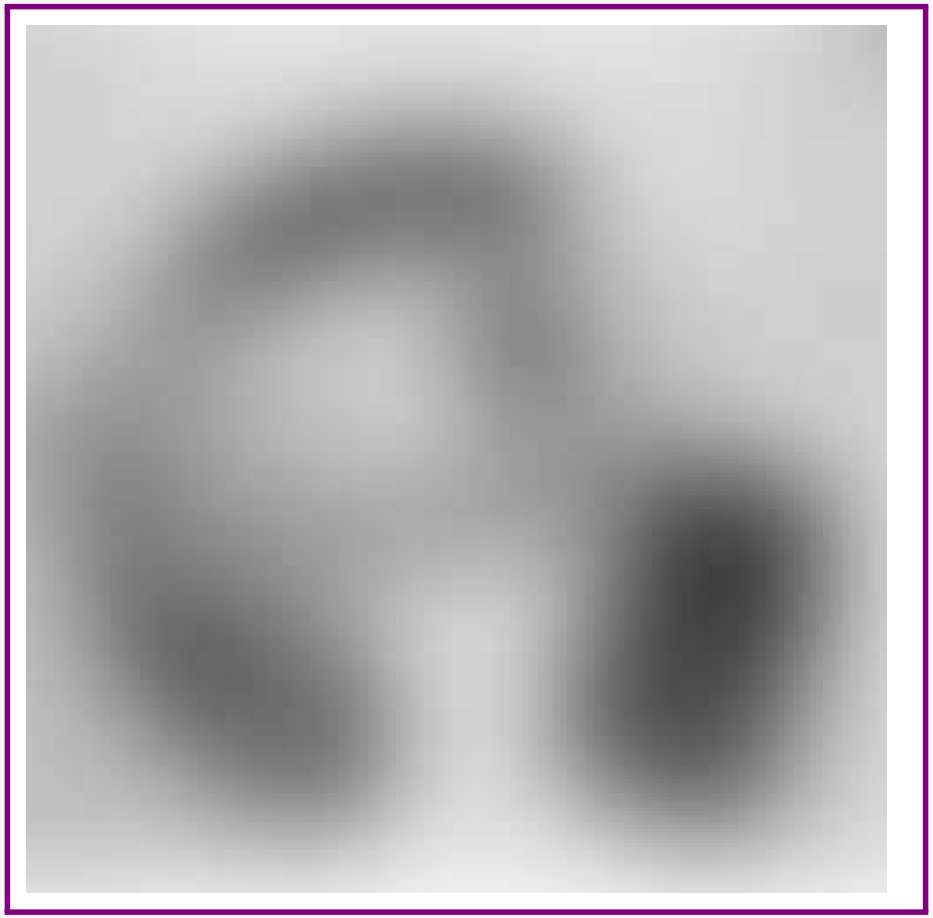
$\sigma = 4$

Gaussian smoothing: from smallest to largest scales



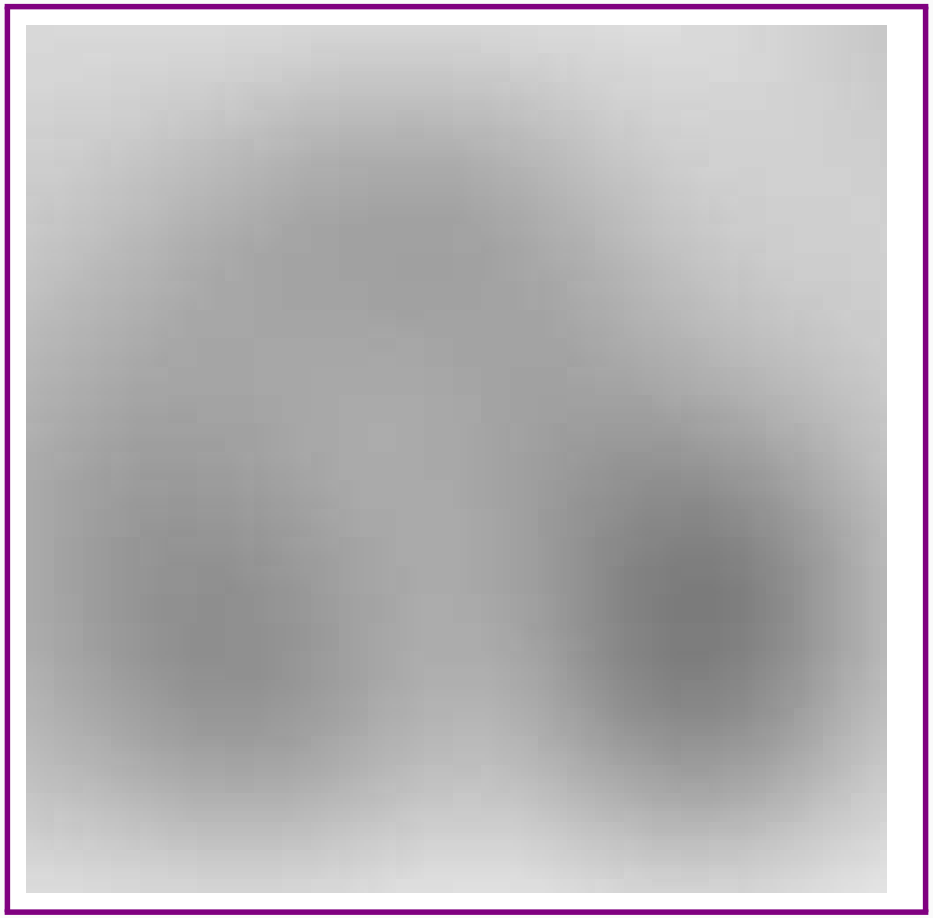
$\sigma = 8$

Gaussian smoothing: from smallest to largest scales



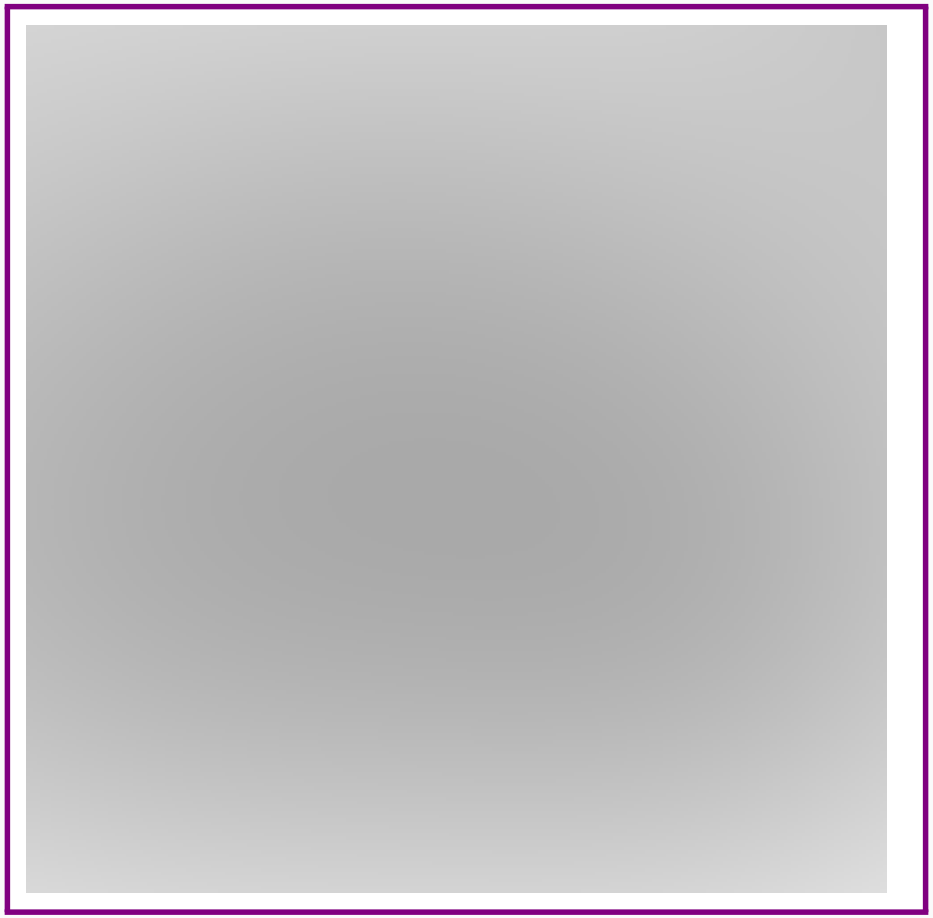
$\sigma = 16$

Gaussian smoothing: from smallest to largest scales



$\sigma = 32$

Gaussian smoothing: from smallest to largest scales



$\sigma = 64$

Gaussian smoothing

Spatial filtering vs Frequency filtering

Temporal/spatial	Frequency
$y(n,m) = x \star h(n,m)$	$y(n,m) = TFD^{-1}\left[X(f,g) \times H(f,g)\right]$

- What is the transfer function of a Gaussian kernel?
- Recall: the Fourier transform of a Gaussian function is a Gaussian function

$$\text{FT}\left[e^{-b^2t^2}\right](f) = \frac{\sqrt{\pi}}{|b|}e^{-\frac{\pi^2 f^2}{b^2}}$$

$$\text{FT}\left[e^{-b^2(t^2+u^2)}\right](f,g) = \frac{\pi}{b^2}e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

- Lowpass filter, non ideal (see Section 2, Filtering in the frequency domain)

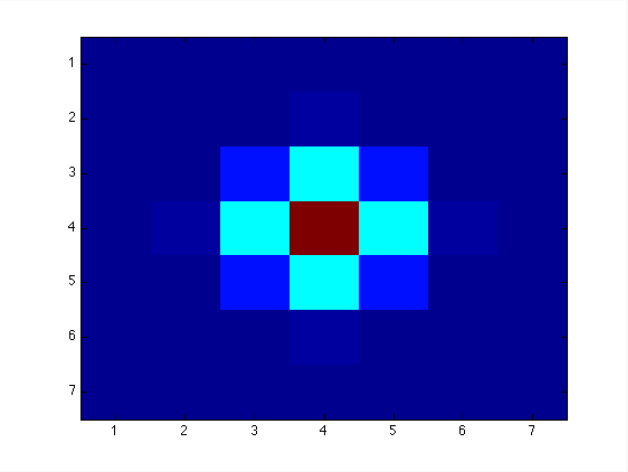
Gaussian smoothing

Spatial filtering vs frequency filtering

$$\text{FT}\left[e^{-b^2(t^2+u^2)}\right](f,g) = \frac{\pi}{b^2} e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

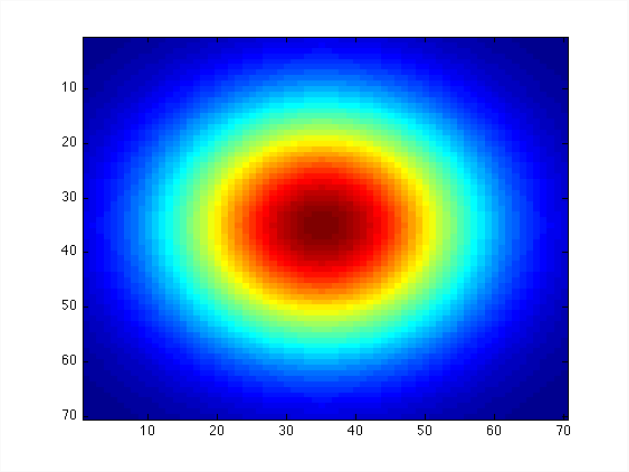
- Spatially: $\sigma_s = \frac{1}{b}$, frequency: $\sigma_f = \frac{b}{\pi} \Rightarrow \sigma_f = \frac{1}{\sigma_s \pi}$

$h(n,m)$



$\sigma_s = 0.7$, size 7×7

$H(f,g)$



$\sigma_f = 0.45$

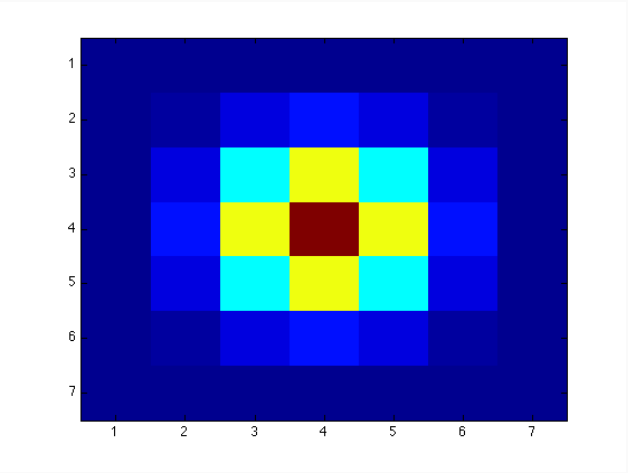
Gaussian smoothing

Spatial filtering vs frequency filtering

$$\text{FT}\left[e^{-b^2(t^2+u^2)}\right](f,g) = \frac{\pi}{b^2} e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

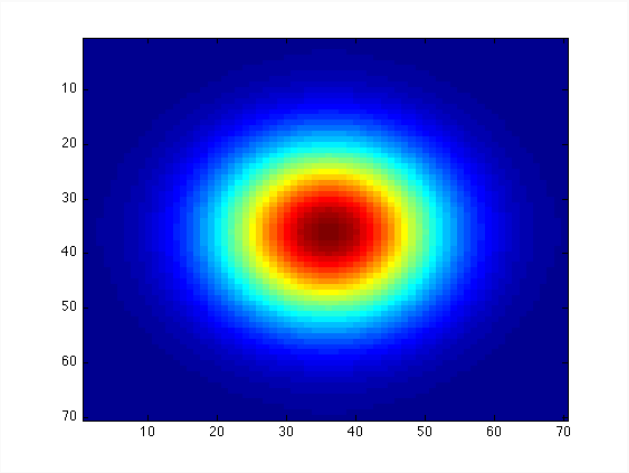
- Spatially: $\sigma_s = \frac{1}{b}$, frequency: $\sigma_f = \frac{b}{\pi} \Rightarrow \sigma_f = \frac{1}{\sigma_s \pi}$

$h(n,m)$



$\sigma_s = 1.0$, size 7×7

$H(f,g)$



$\sigma_f = 0.32$

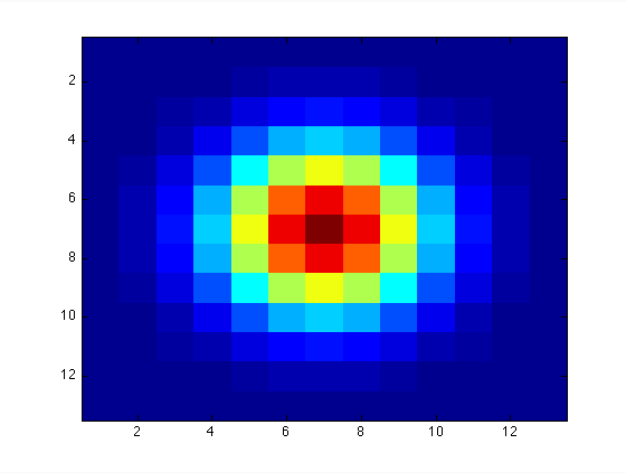
Gaussian smoothing

Spatial filtering vs frequency filtering

$$\mathrm{FT}\left[e^{-b^2(t^2+u^2)}\right](f,g) = \frac{\pi}{b^2} e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

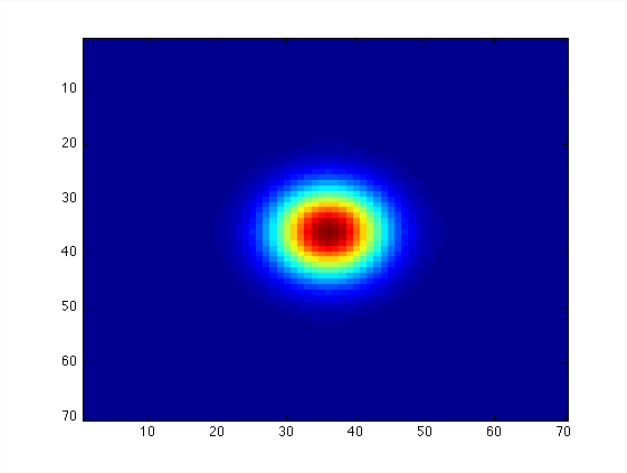
- Spatially: $\sigma_s = \frac{1}{b}$, frequency: $\sigma_f = \frac{b}{\pi} \Rightarrow \sigma_f = \frac{1}{\sigma_s \pi}$

$h(n,m)$



$\sigma_s = 2.0$, size 13×13

$H(f,g)$



$\sigma_f = 0.16$

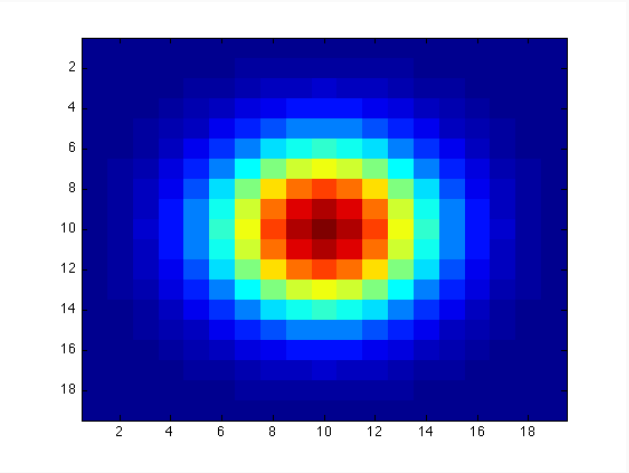
Gaussian smoothing

Spatial filtering vs frequency filtering

$$\mathrm{FT}\left[e^{-b^2(t^2+u^2)}\right](f,g) = \frac{\pi}{b^2} e^{-\frac{\pi^2(f^2+g^2)}{b^2}}$$

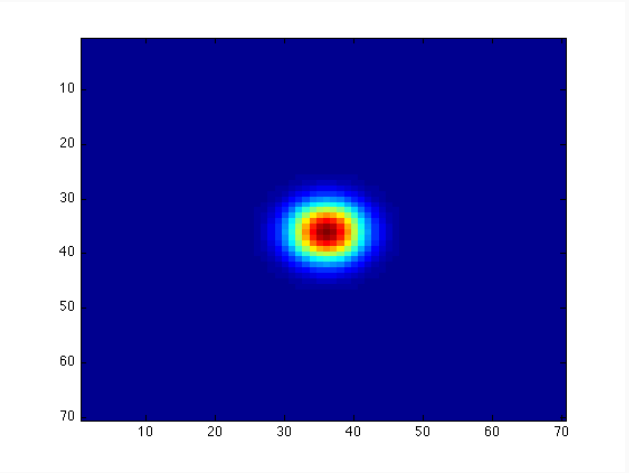
- Spatially: $\sigma_s = \frac{1}{b}$, frequency: $\sigma_f = \frac{b}{\pi} \Rightarrow \sigma_f = \frac{1}{\sigma_s \pi}$

$h(n,m)$



$\sigma_s = 3.0$, size 19×19

$H(f,g)$



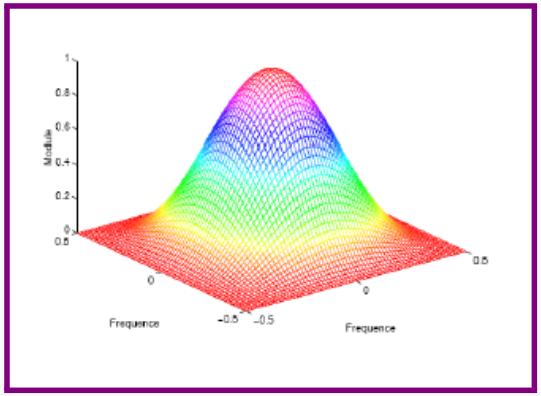
$\sigma_f = 0.10$

Convolution filtering: other filters

Binomial filter

- Coefficients derived from Newton binomial

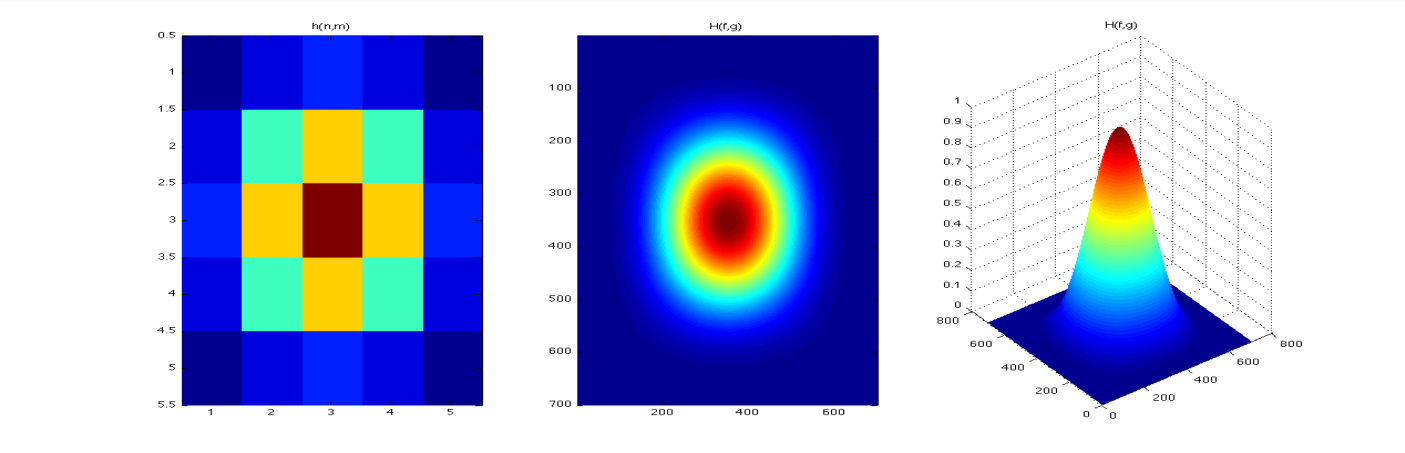
$$h = \frac{1}{256}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \frac{1}{256}\begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$



Convolution filtering: other filters

Binomial filter

$$h = \frac{1}{256}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

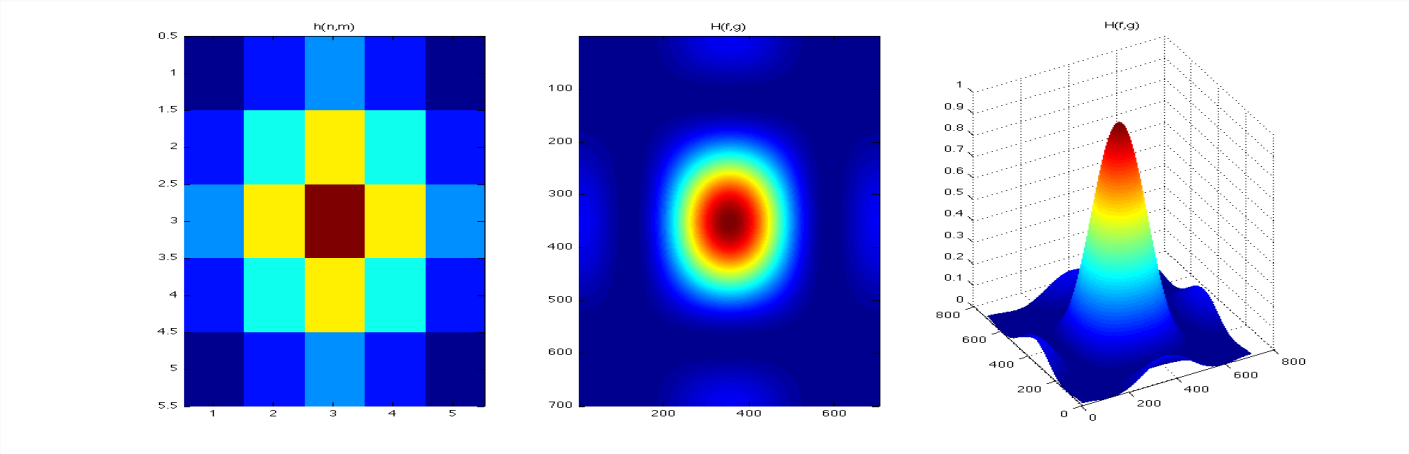


- Lowpass filter, non ideal (see Section 2, Filtering in the frequency domain)

Convolution filtering: other filters

Pyramidal filter

$$h_p = \frac{1}{81}\begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}$$

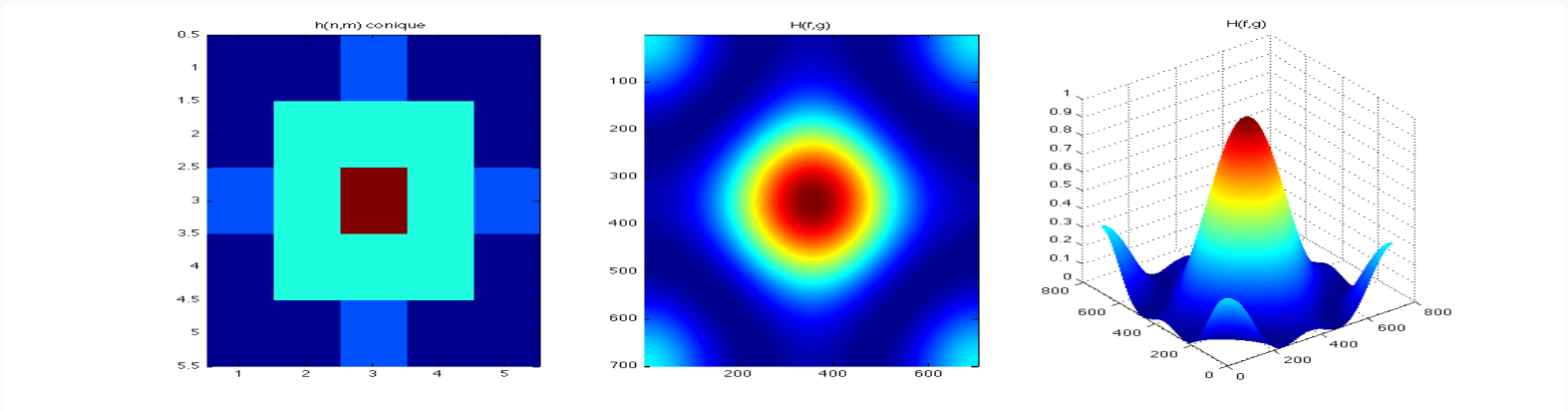


- Lowpass filter, non ideal (see Section 2)

Convolution filtering: other filters

Conic filter

$$h_c = \frac{1}{25}\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 5 & 2 & 1 \\ 0 & 2 & 2 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

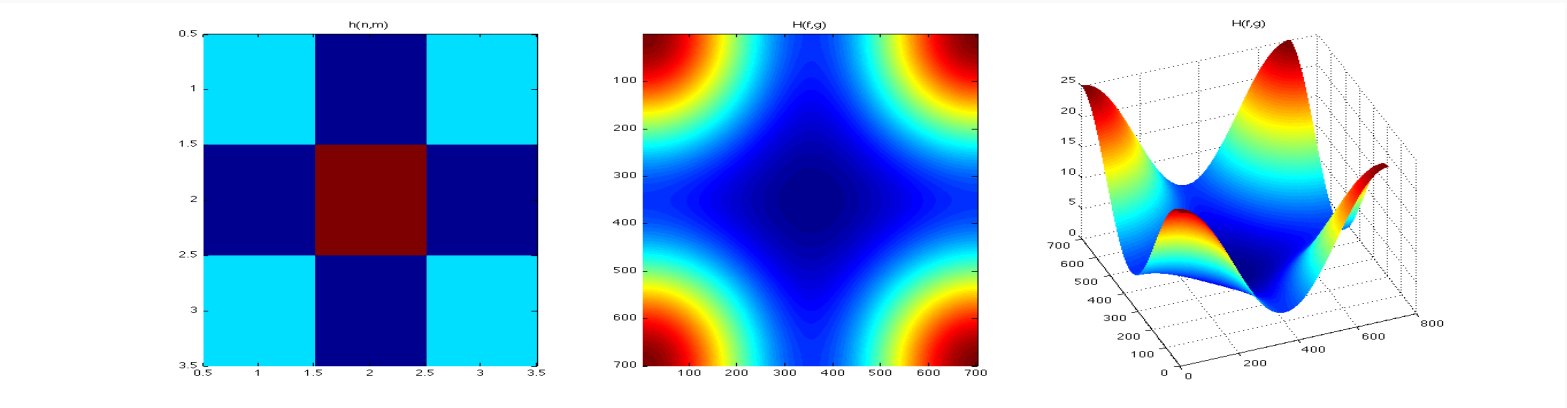


- Lowpass filter, non ideal (see Section 2)

Convolution filtering: other filters

Sharpening filter (to increase image details)

$$h_{r_1} = \begin{pmatrix} 1 & -3 & 1 \\ -3 & 9 & -3 \\ 1 & -3 & 1 \end{pmatrix}$$

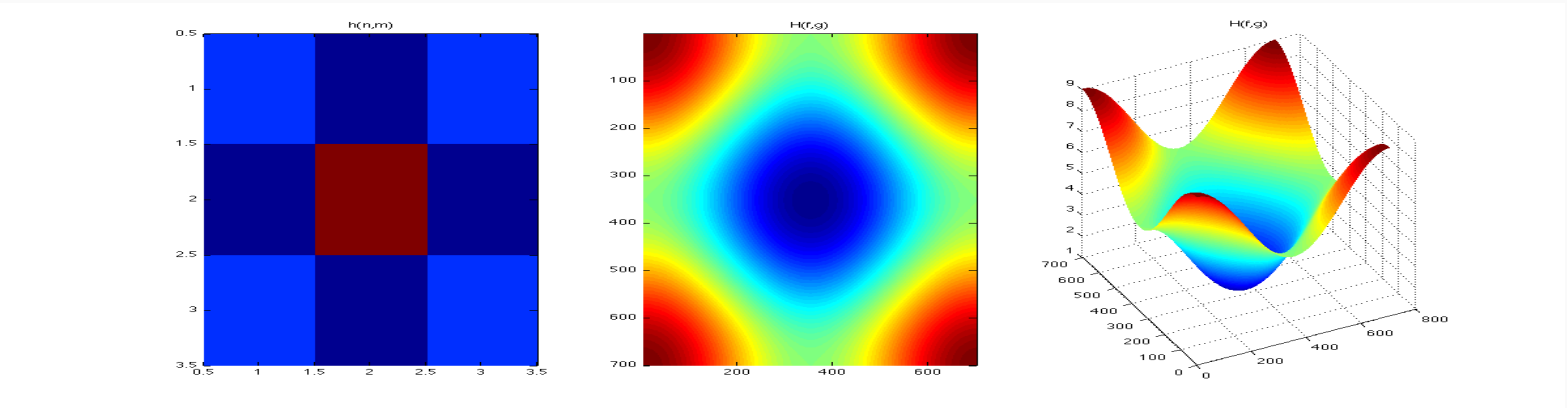


- High pass filter, non ideal (see Section 2)

Convolution filtering: other filters

Sharpening filter: second version

$$h_{r_2} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

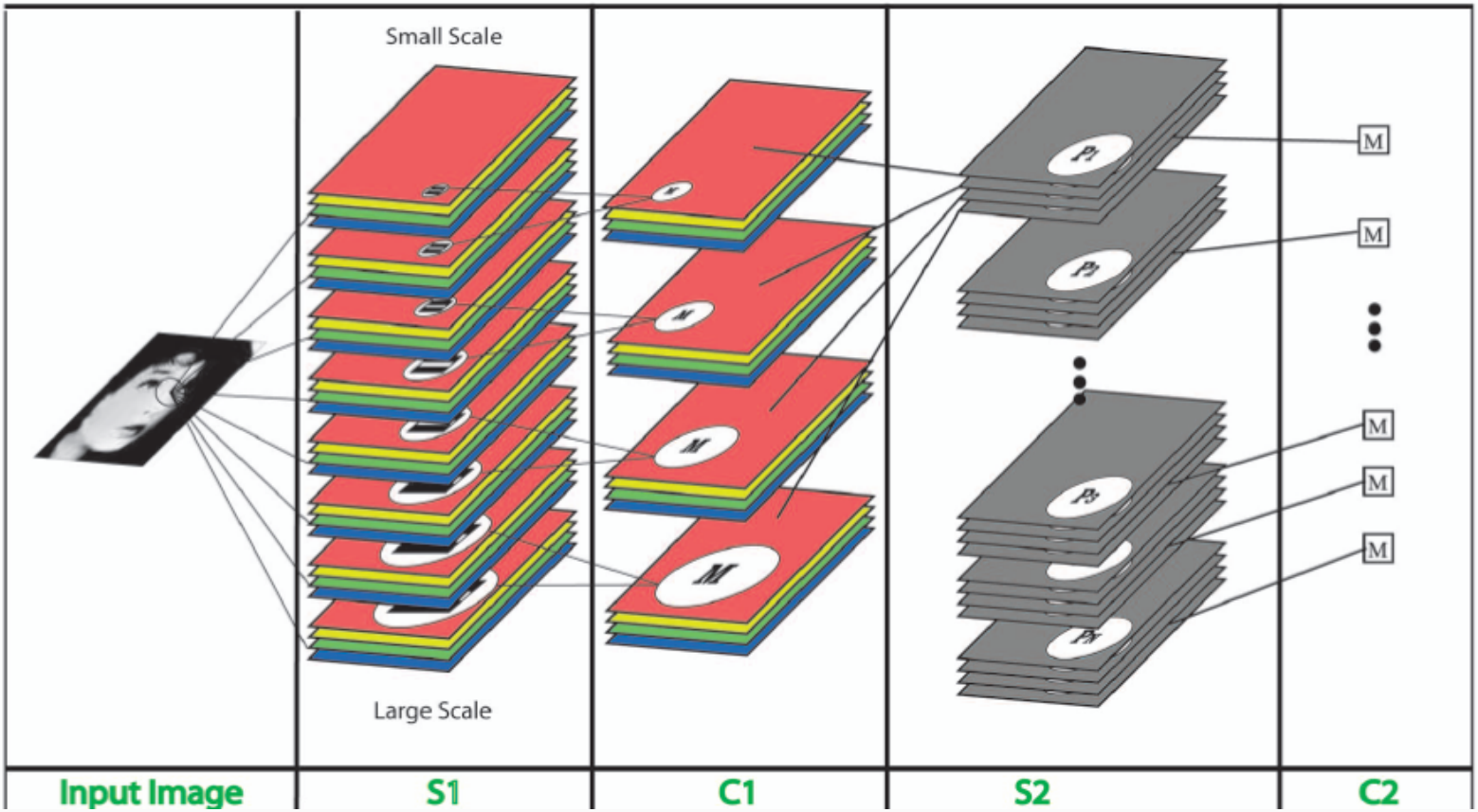


- Highpass filter, non ideal (see Section 2)

Gabor filters

Gabor filters for image classification [*Serre*2007]

- Goal: model the process inside the human visual cortex (V1)
  - Layer $S1$: convolution using a bank of Gabor filters



Gabor filters

Definition

$$h(x, y) = \exp\left(-\frac{x_0^2 + \gamma\, y_0^2}{2\sigma^2}\right)\cos\left(\frac{2\pi}{\lambda}x_0\right) \tag{8}$$

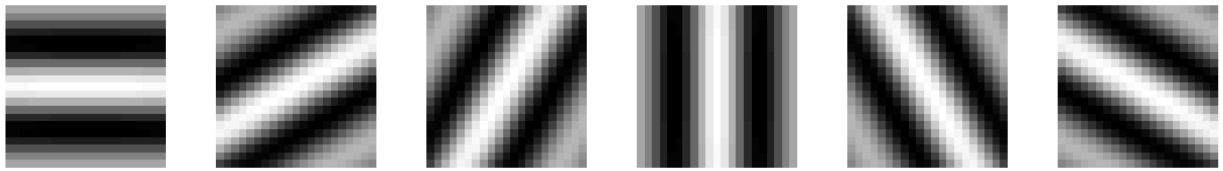
$x_0 = x\cos(\theta) + y\sin(\theta)$ and $y_0 = -x\sin(\theta) + y\cos(\theta)$

- Parameters are set according to biological considerations
- 1 aspect ratio $\gamma = 0.3$, 4 orientations $\theta = 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$, 16 values of scale ($\sigma$), wavelength ($\lambda$) and filter size ($s \times s$)
- Bandpass filters that localize **both** in space and in frequency domains
- Give response for various scales and orientations

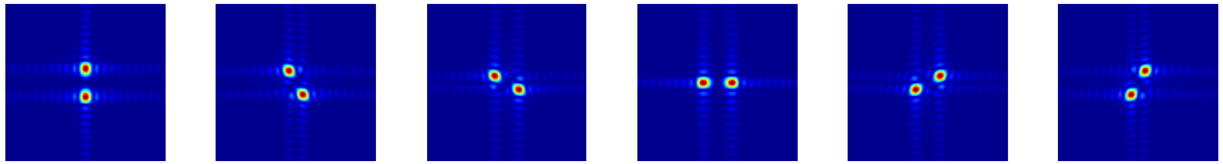
Variation of orientation

Let's vary θ:

Impulse response $h(n, n)$



Transfer function $H(f, g)$

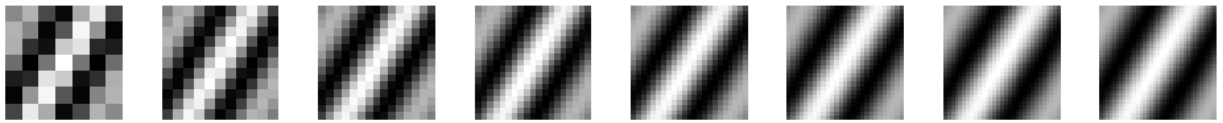


- Capture spatial frequencies at various orientations

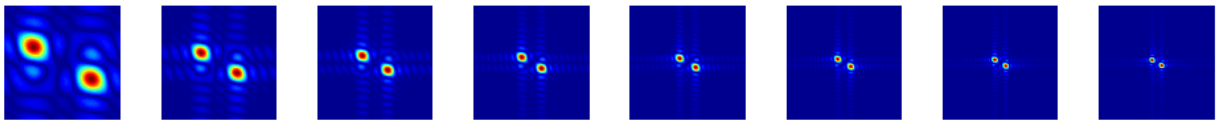
Variation of scale

Let's vary s:

Impulse response $h(n, n)$



Transfer function $H(f, g)$



- Capture spatial frequencies at various scales

Separable filter

Definition

- If $h(i,j) = h_x(i) \times h_y(j)$ (filter of size $N \times M$) the 2-D convolution may be expressed as two 1-D convolutions:

$$\begin{aligned}
(f \star h)(i,j) &= \sum_{n=-N/2}^{N/2} \sum_{m=-M/2}^{M/2} f(i-n, j-m) h(n,m) \\
&= \sum_{m=-M/2}^{M/2} \left(\sum_{n=-N/2}^{N/2} f(i-n, j-m) h_x(n) \right) h_y(m) \\
&= (f \star h_x \star h_y)(i,j)
\end{aligned}$$

Separable filter

- Averaging, Gaussian, binomial... filters are separable
- Averaging: 3×3: $h_x = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$, $h_y = \frac{1}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ and

  $h = h_y h_x$
- Gaussian: $\exp(-\frac{i^2+j^2}{2\sigma^2}) = \exp(-\frac{i^2}{2\sigma^2})\exp(-\frac{j^2}{2\sigma^2})$
- See tutorial works 4: separability
  - allows speeding-up the computation
  - allows for a recursive implementation (Deriche, 1987)

Filtering color images

Linear filtering

- Linear time-invariant filtering, with $f(i,j) \in \mathbb{R}^p$, and $h(i,j)$ matrix of size $p \times p$ ($p = 3$ for color images):

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=1}^{N} \sum_{m=1}^{M} f(i,j) h(n-i, m-j)$$

- Two cases:
  1. $h(i,j)$ is diagonal, each color channel is processed independently
  2. $h(i,j)$ is not diagonal: there are some non null crossed terms (dependence between channels)
- Choice of the color space representation is important: RGB channels can often be supposed independent

Filtering color images: example

Averaging filter

- Three scalar averaging (independence between channels)
- Kernels can be specific to each channel
- Issue: creation of "false" colors

Outline

Filtering in frequency domain

Definitions

- We consider discrete signals
- Principle: modify some signal frequency components
- Two ways:
  - In the spatial domain: convolution product between the signal and the impulse response h of the filter
  - In the frequency domain: element-wise multiplication between the signal spectrum and the transfer function H of the filter
- Three families:
  - lowpass filters
  - highpass filters
  - bandpass filters
- Ideal filter: coefficients of H are equal to 1 or 0 (keep or remove frequency components)

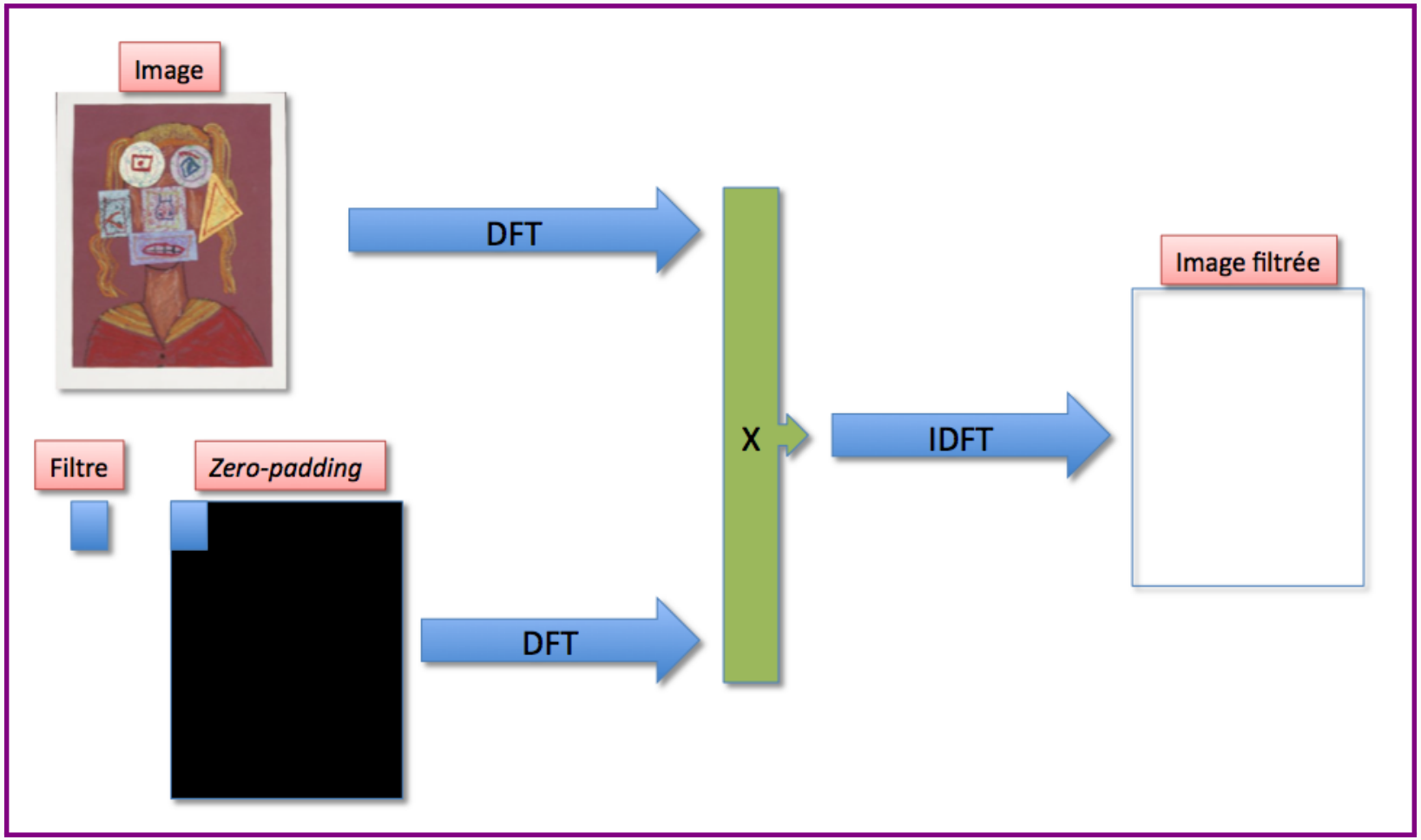
Filtering in frequency domain

Principle

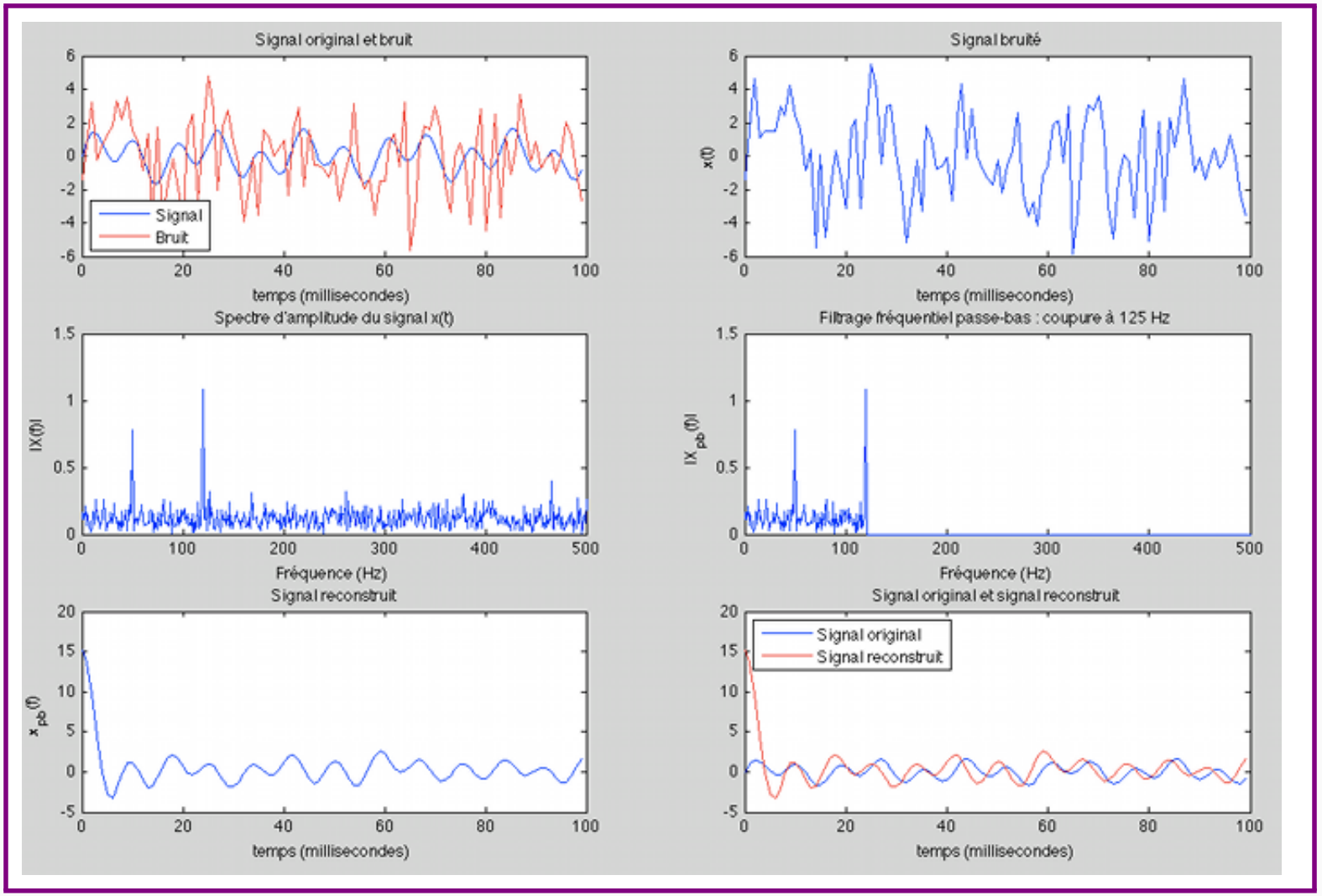
1. Compute the spectrum X as the DFT of the signal x
2. Compute the transfer function H as the DFT of the filter h
3. X and H must have the same size: use the zero-padding technique if needed
4. Element wise multiplication (element by element) of X by H: $X_{\text{filtered}}(f, g) = X(f, g) \times H(f, g)$
5. Apply the IDFT on X_{filtered} to reconstruct the filtered signal $x_{\text{filtered}}(n, m)$

Filtering in frequency domain

Principle (2-D)



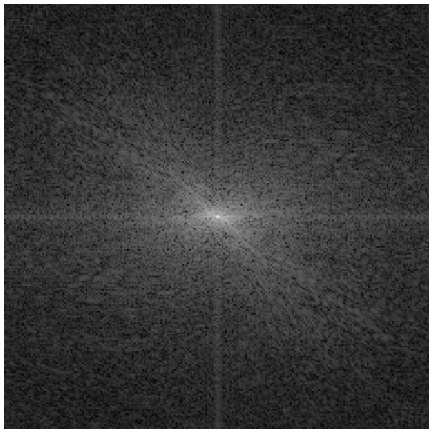
Filtering in frequency domain: example in 1-D

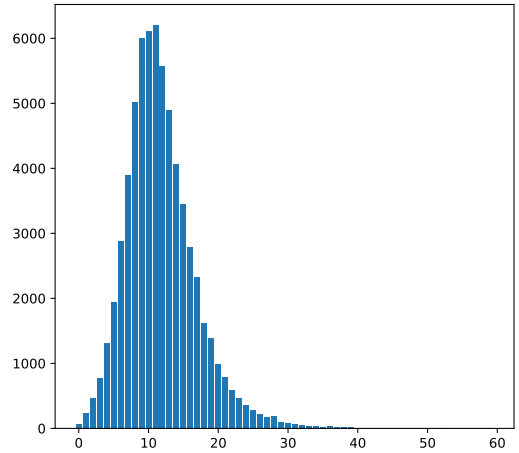


Frequency characteristics of noise

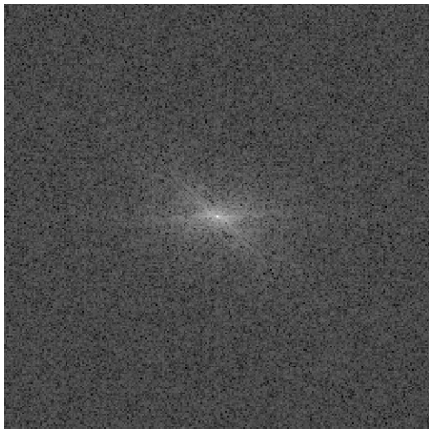
The noise is generally a high frequency (stochastic) process

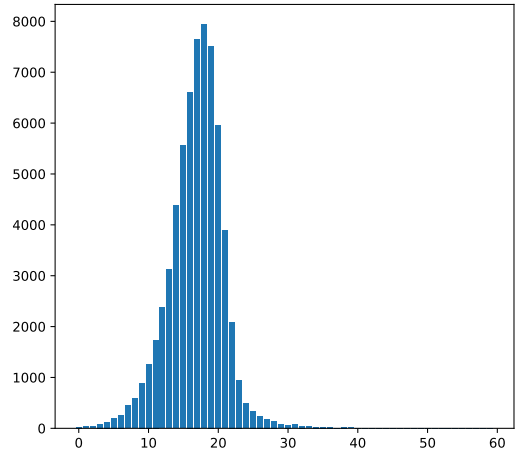
original image - spectrum - frequency components histogram





corrupted image - spectrum - frequency components histogram





Lowpass 2-D filter

Definition

- An ideal lowpass filter is a linear system that does not change low frequencies
- The support of the transfer function is the **bandwidth** and characterizes the filter
- Fundamental and low frequencies are preserved
  $\hookrightarrow$ mean and large structures are preserved after image reconstruction (IDFT)
- High frequencies are suppressed: quick spatial variations of intensity (noise, edges, textures...) are attenuated or even eliminated $\hookrightarrow$ image becomes smooth/blurred

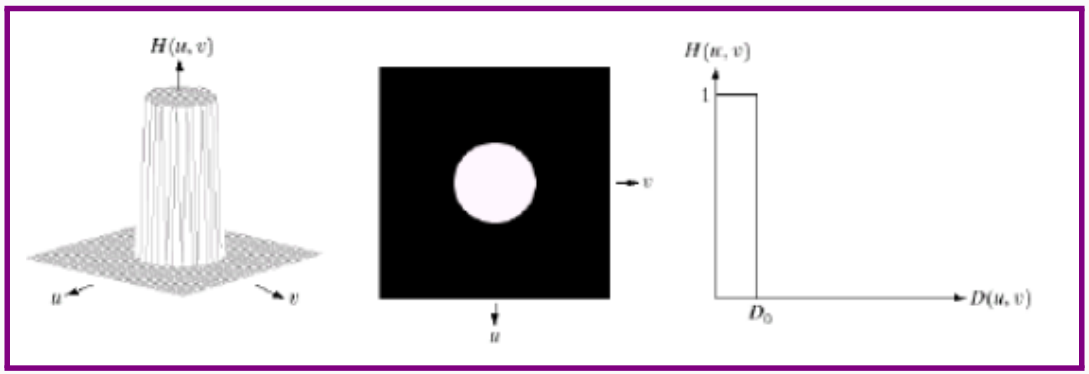
Ideal lowpass 2-D filter

Definition

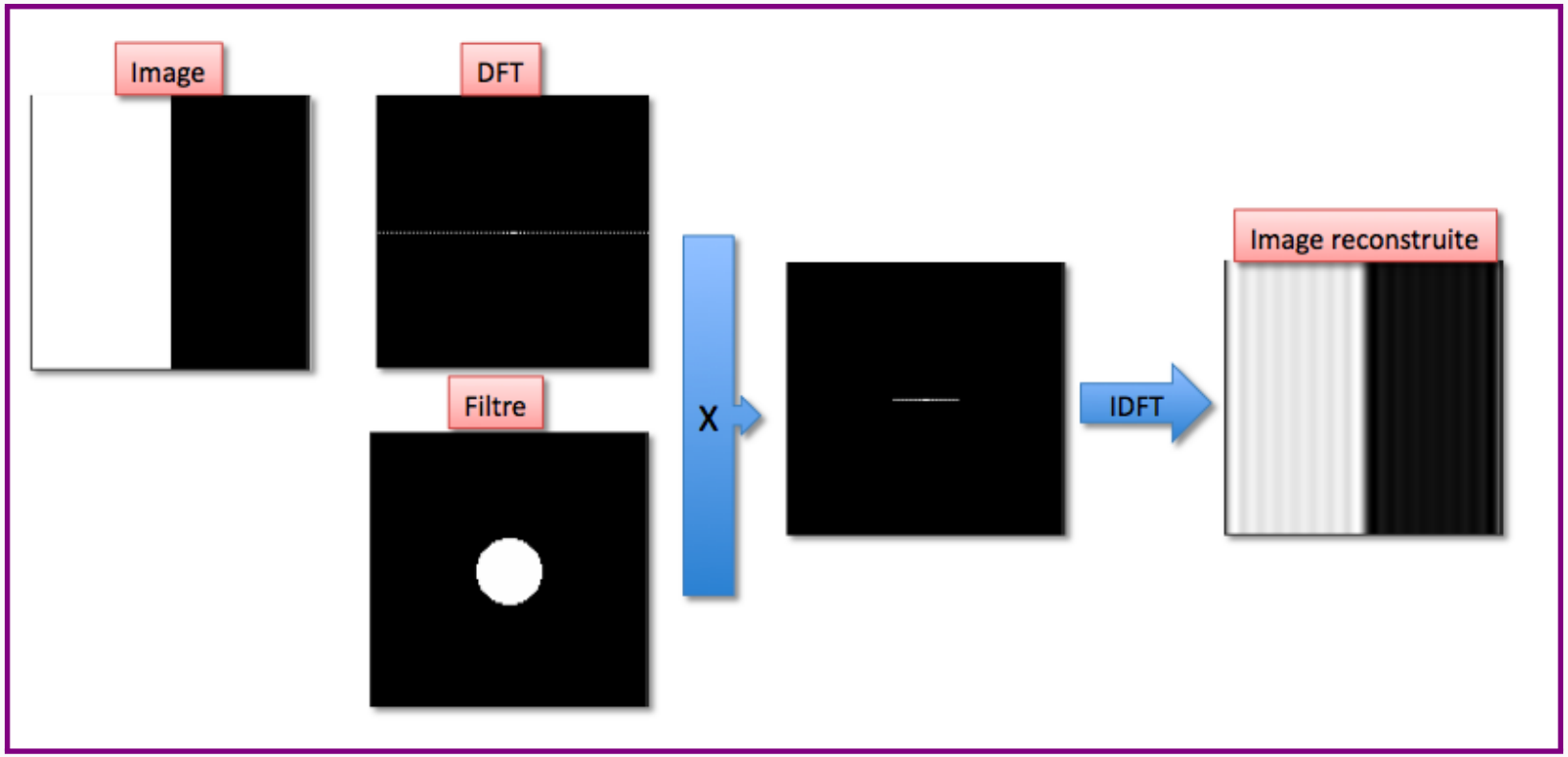
- The transfer function H of a lowpass filter with cutoff frequency D_0 writes:

$$H(u,v) = \begin{cases} 1 & \text{if } \sqrt{u^2+v^2} \leq D_0 \\ 0 & \text{if } \sqrt{u^2+v^2} > D_0 \end{cases}$$

- This filter suppresses frequency components (u, v) having a radial frequency $\sqrt{u^2+v^2}$ greater than D_0



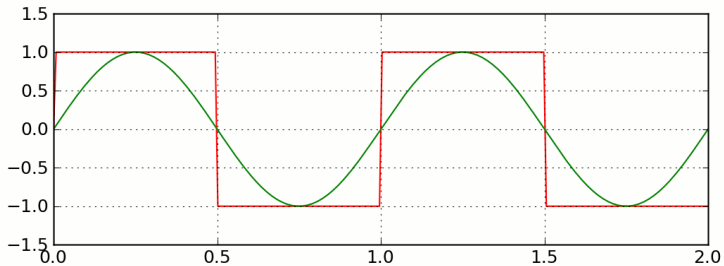
Ideal lowpass 2-D filter



Ideal lowpass 2-D filter

Consequences

- High frequencies are suppressed
- Low frequencies are preserved
- The frontier between the two regions is less sharpened, vertical lines appear in the image: *Gibbs ringing artifacts* or *Gibbs phenomenon*
  $\hookrightarrow$ high frequencies are needed to correctly represent an edge (a short spatial intensity variation)



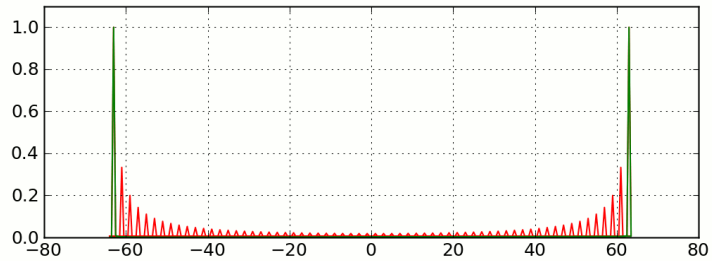
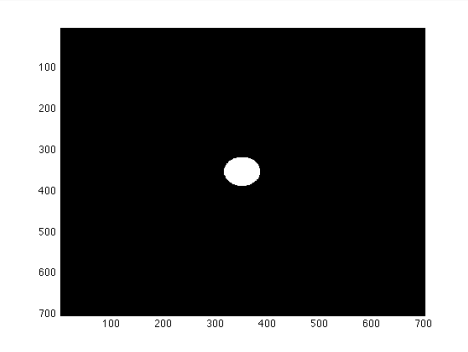


Illustration of Gibbs phenomena, in 1-D. Credit: Wikipedia

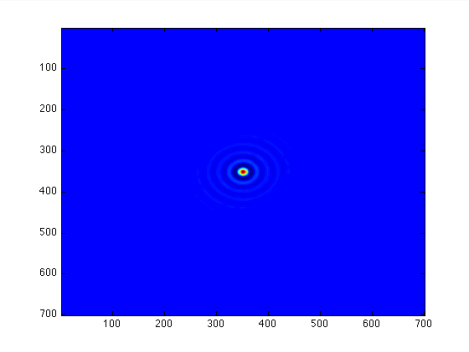
Ideal lowpass 2-D filter

Consequences

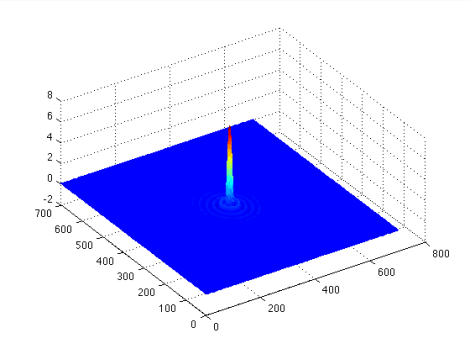
$H(f, g)$



$h(n, m)$

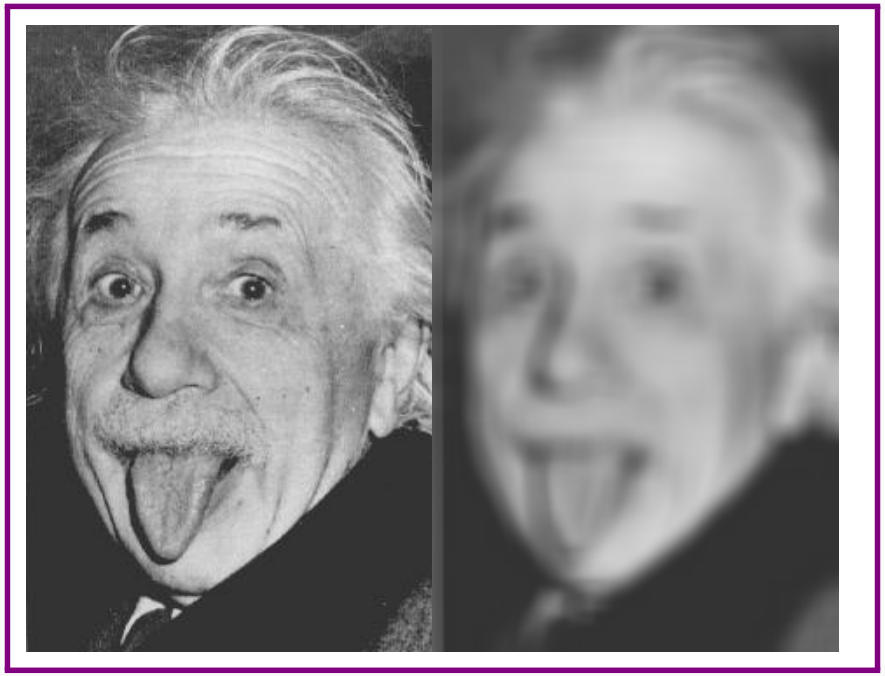


$h(n, m)$



- The ideal lowpass filter is a convolution with a first-order Bessel function (similar to a "circular" sinc function)
- H is non derivable at the cutoff frequency and induces Gibbs artifacts

Ideal lowpass 2-D filter: example on natural image

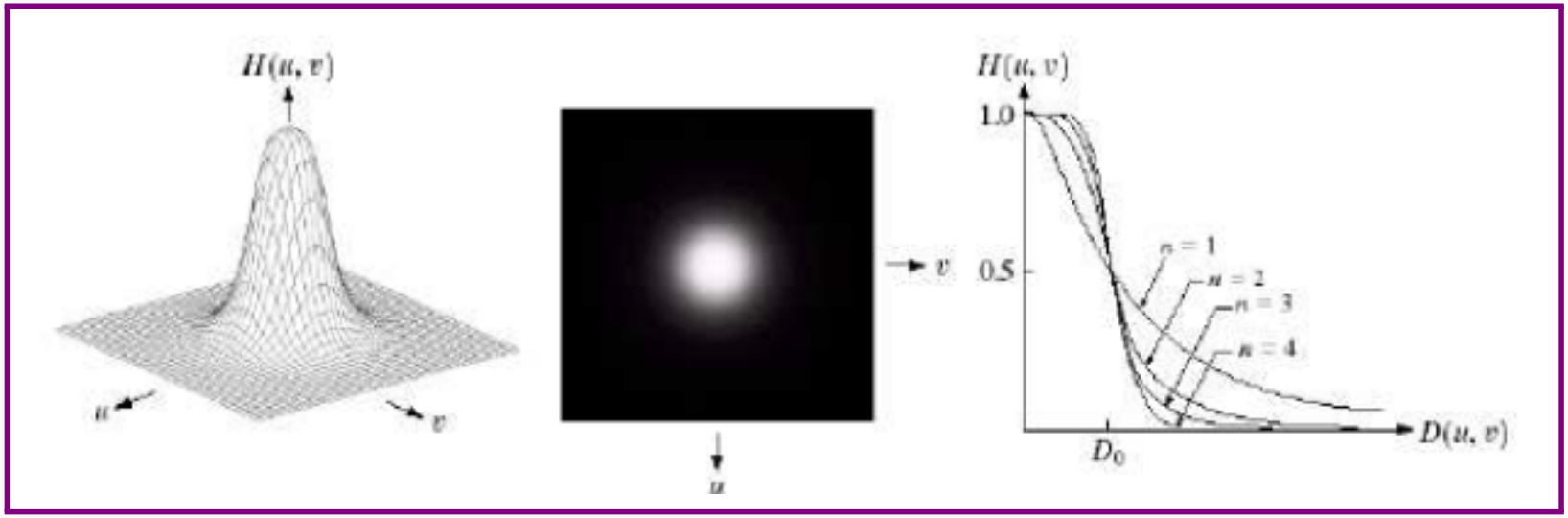


Lowpass Butterworth 2-D filter

Definition

- The lowpass Butterworth filter of order n is defined by:

$$H(u,v) = \frac{1}{1+\left(\frac{\sqrt{u^2+v^2}}{D_0}\right)^{2n}}$$



- Order n controls the slope of the transition and D_0 the cutoff frequency

Lowpass Butterworth 2-D filter of order n

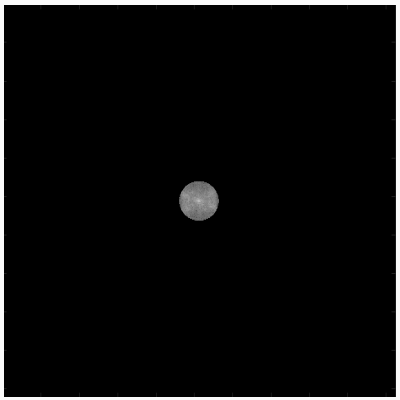
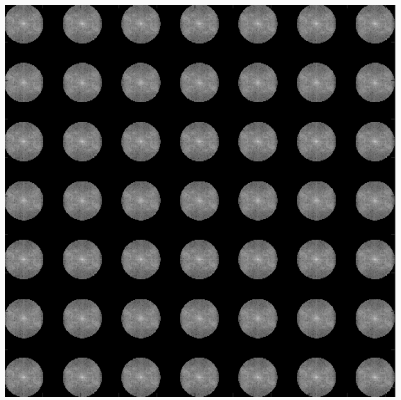
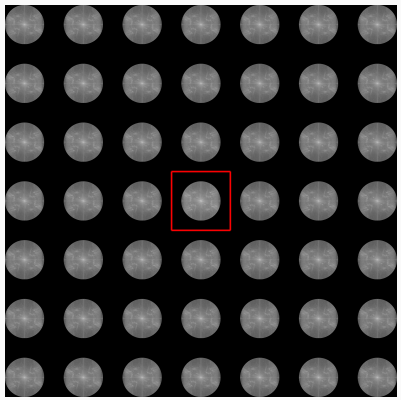
Characteristics

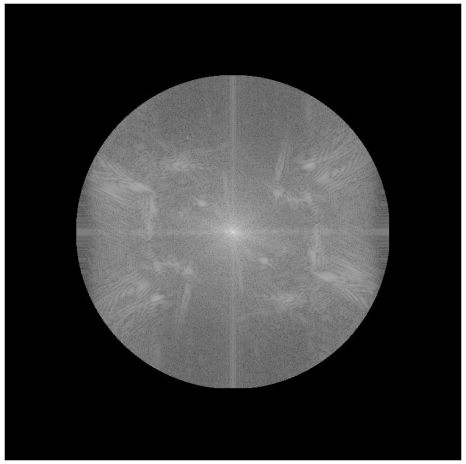
- Approximation of an ideal lowpass filter without the issue of ideal filter: no Gibbs phenomenon
- The transfer function is continuous and derivable
- High frequency components beyond the disc of radius D_0 are attenuated
- With a high value of n the filter is closer to an ideal filter and high frequency components are more attenuated
- Edges are better preserved than with an ideal filter

Aliasing: frequency interpretation (recall)

- Image correctly sampled: $f_e \geq 2f_{max}$ (Shannon)

Continuous FT | Sampled signal FT | CFT = DFT

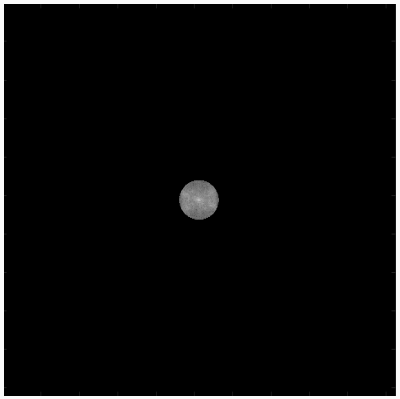



DFT=CFT | No aliasing

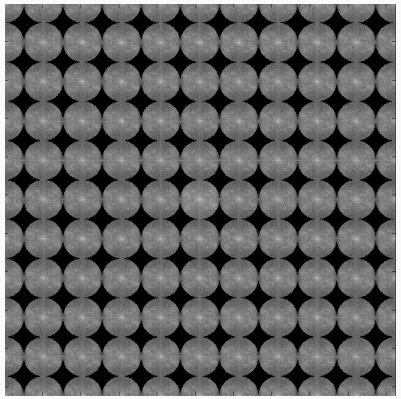
Aliasing: frequency interpretation (recall)

Image correctly sampled, limite value: $f_e = 2f_{max}$ (Shannon)

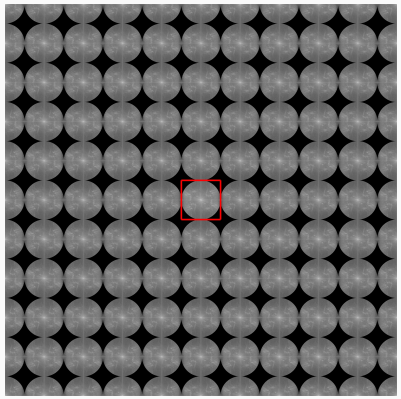
Continuous FT

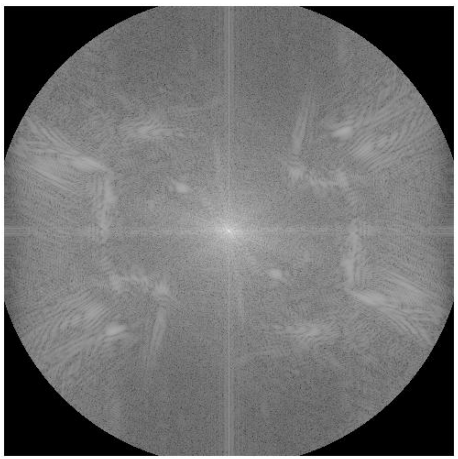


Sampled signal FT



CFT = DFT





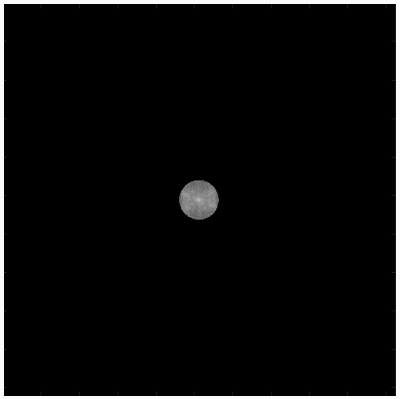
DFT=CFT

no aliasing

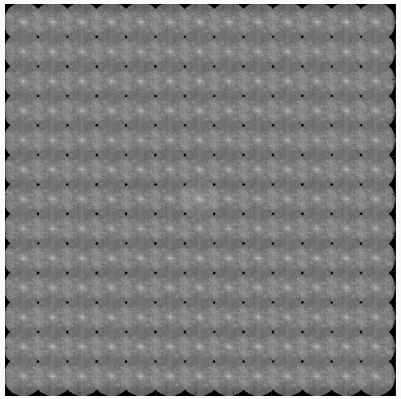
Aliasing: frequency interpretation (recall)

Image incorrectly sampled: $f_e < 2f_{max}$

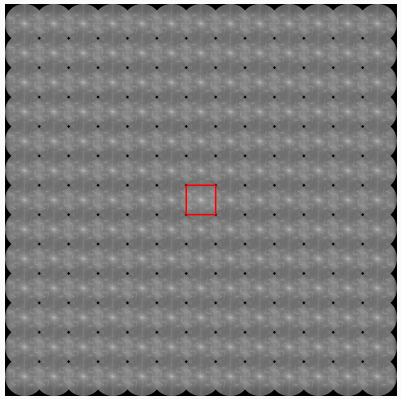
Continuous FT

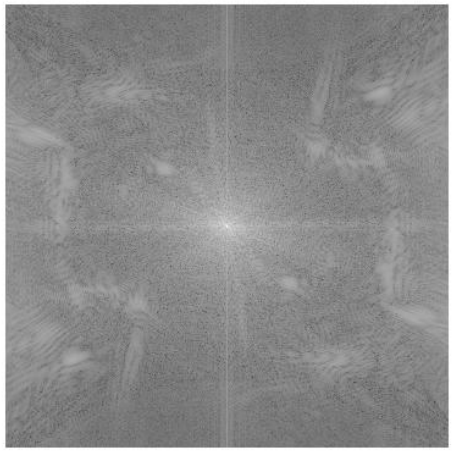


Sampled signal FT



CFT $\neq$ DFT





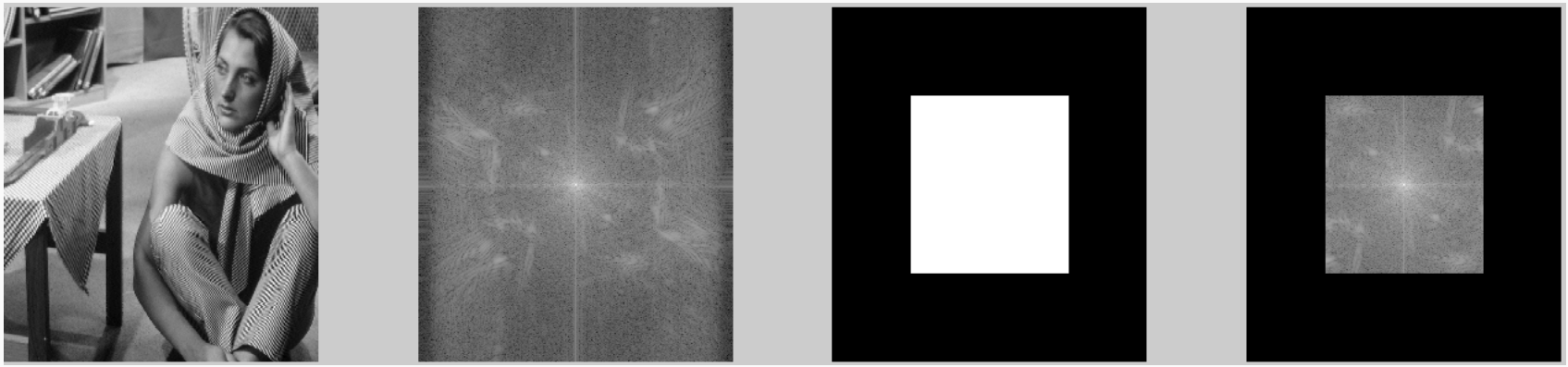
DFT $\neq$ CFT

aliasing

Digital anti-aliasing filter

Principle

- Context: a digital image of size $N \times M$ correctly sampled in Shannon's sense
- Subsampling $\rightarrow$ introduction of aliasing effects
- Anti-aliasing filter: apply a lowpass filter before subsampling

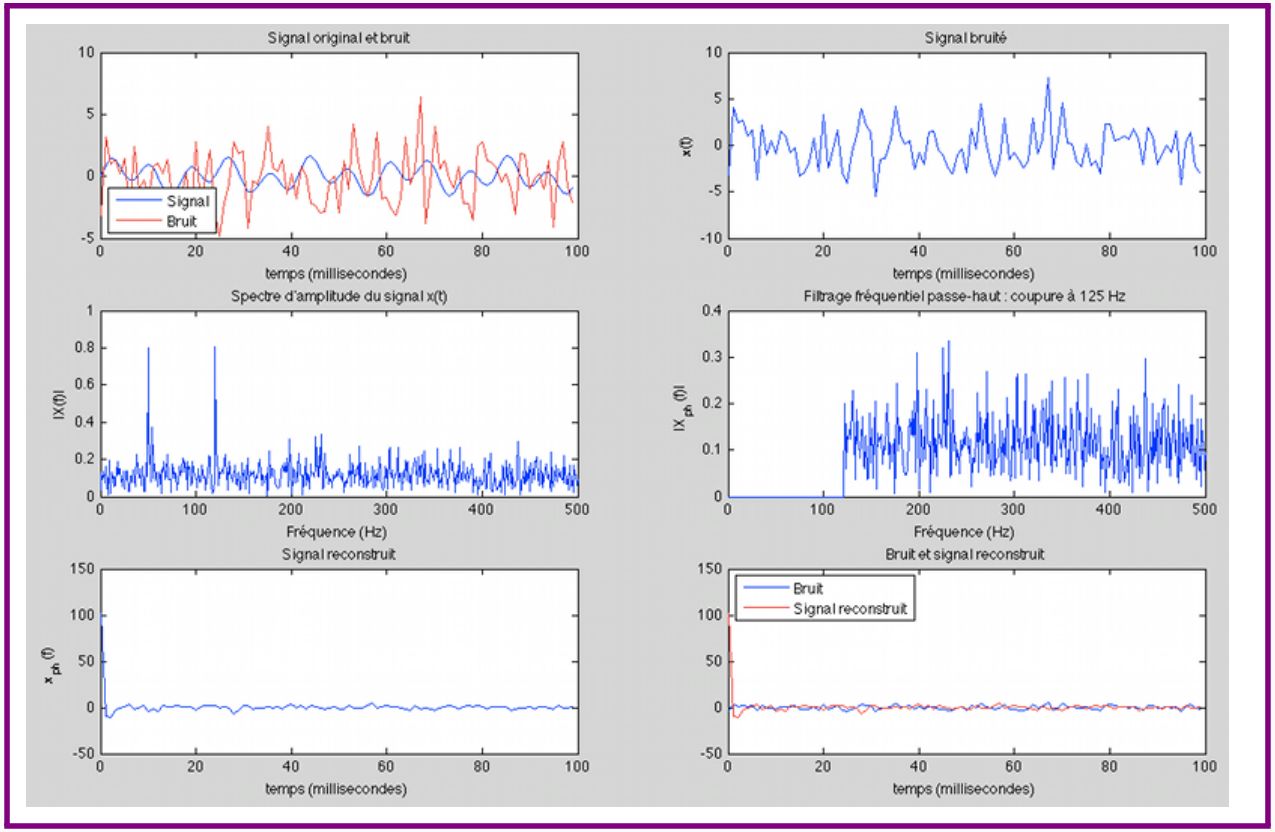


See practical works

Highpass 1-D filtering

Principle

- Preserve high frequency components, suppress low frequency components (including fundamental frequency)
- Example:



Highpass 2-D filter

Definition

- A highpass filter is a linear system that does not modify nor attenuate high frequency components
- Fundamental frequency and low frequencies are eliminated
  $\hookrightarrow$ After reconstruction (IDFT), the image mean is not preserved, as well as intensity information
- High frequencies preserved
  $\hookrightarrow$ Rapid change of intensity (noise, edges...) are highlighted

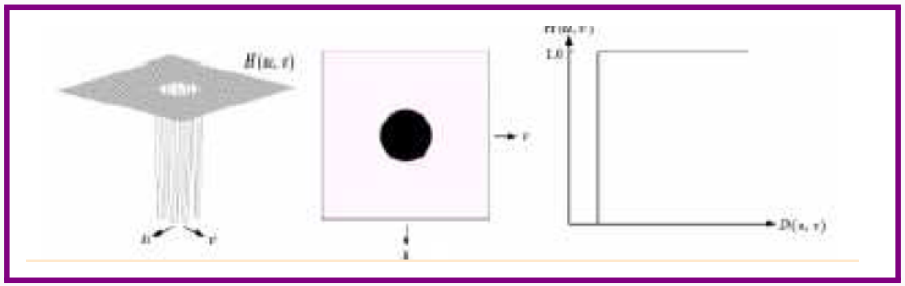
Ideal highpass 2-D filter

Définition

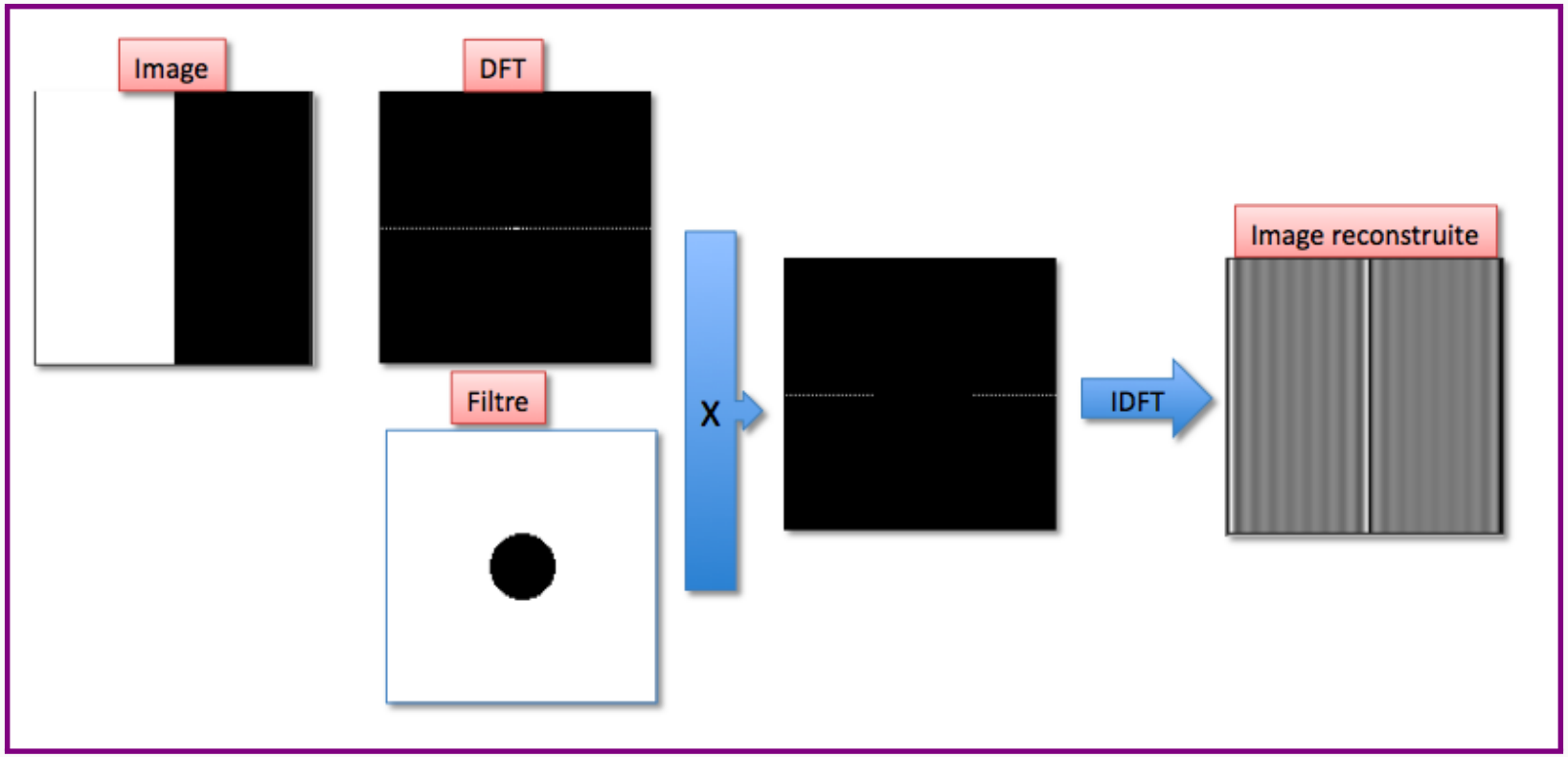
- The transfer function H of an ideal highpass filter, with cutoff frequency D_0, writes:

$$H(u,v) = \begin{cases} 1 & \text{if } \sqrt{u^2+v^2} \geq D_0 \\ 0 & \text{if } \sqrt{u^2+v^2} < D_0 \end{cases}$$

- This filter suppresses frequency components having a radial frequency $\sqrt{u^2+v^2}$ lower than D_0

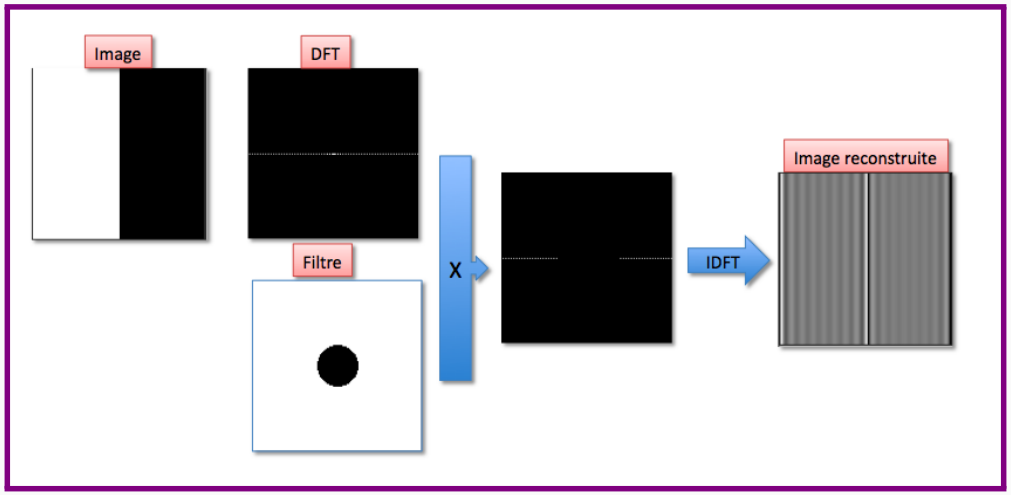


Ideal highpass 2-D filter



Ideal highpass 2-D filter

Interpretation



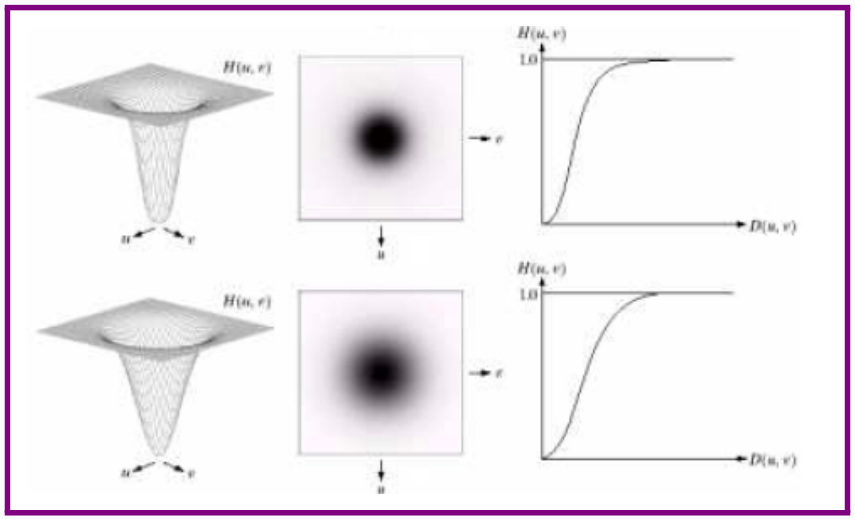
- High frequencies are preserved
- Low frequencies, including the fundamental frequency, are suppressed
- The reconstructed image has lost its intensity values but the frontier between the two regions is sharpened

Highpass Butterworth 2-D filter

Definition

- The highpass Butterworth filter, of order n and cutoff D_0, is defined by:

$$H(u,v) = \frac{1}{1+\left(\frac{D_0}{\sqrt{u^2+v^2}}\right)^{2n}}$$

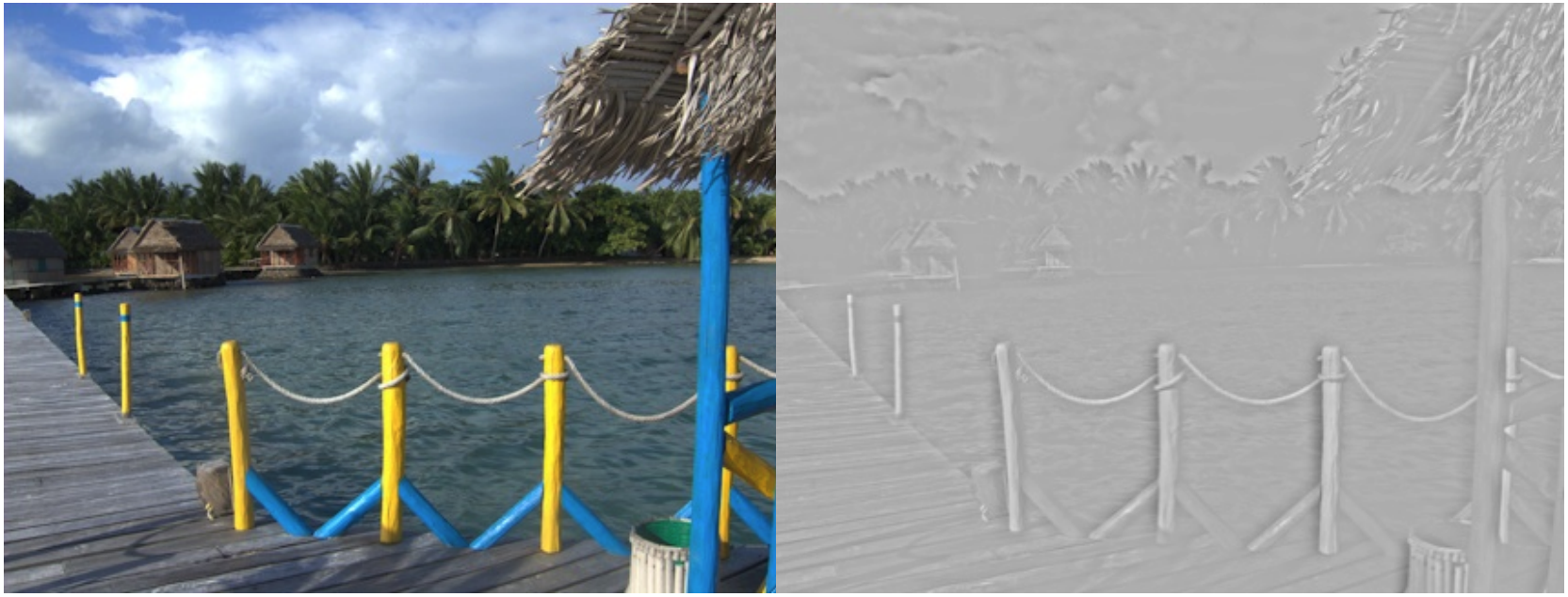


Highpass Butterworth 2-D filter

Characteristics

- Approximation of the ideal highpass filter
- Strong attenuation of low frequency components close to the origin (distance lower than the frequency cutoff)
- n controls the slope of transition between high and low frequencies
- A highpass filter has an effect similar to derivation

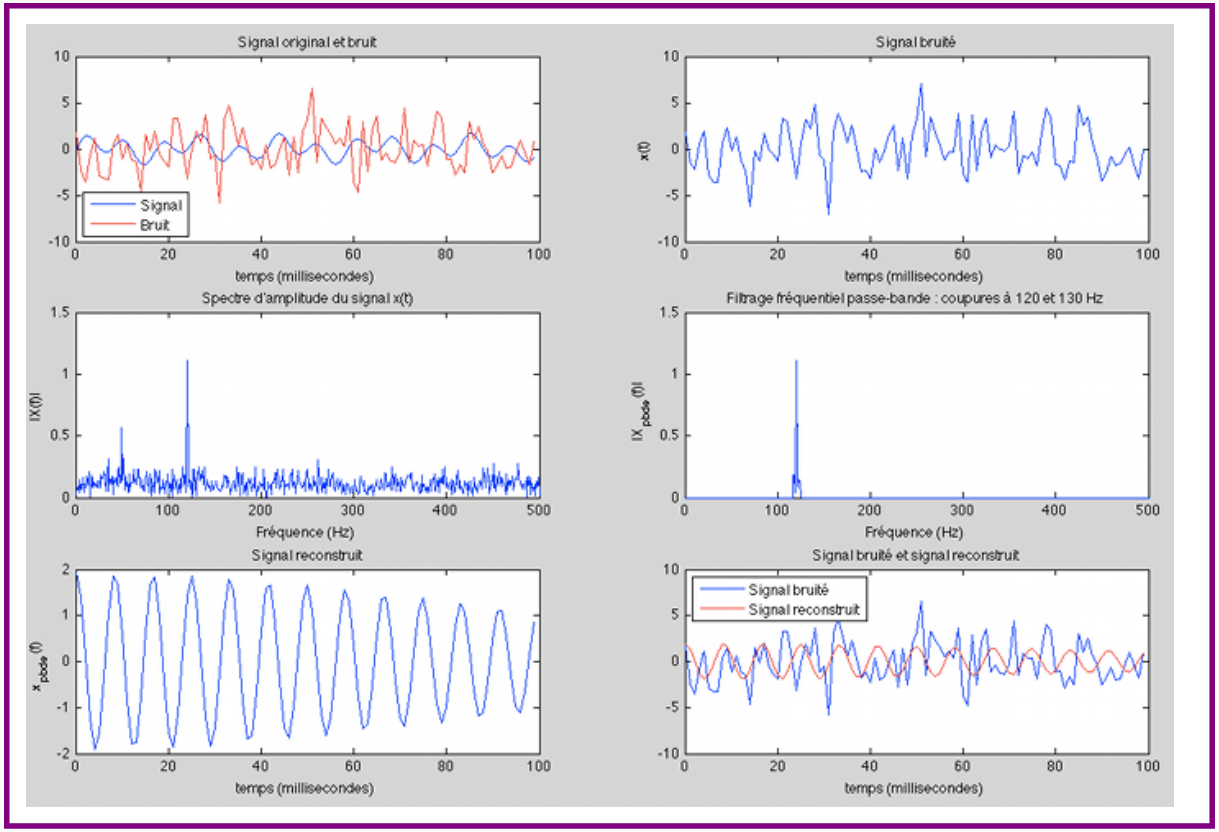
2-D highpass filter: example



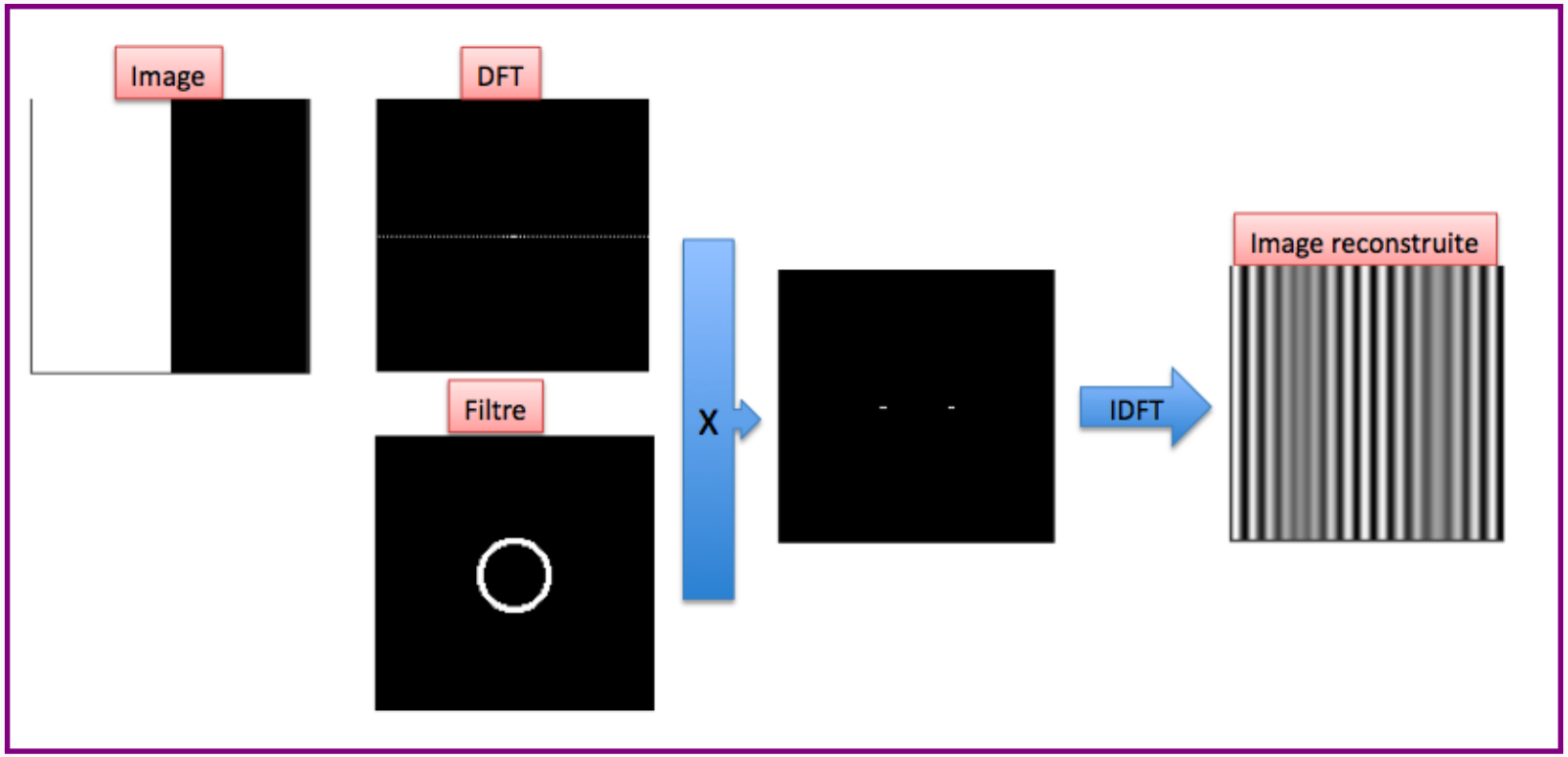
Band-pass filtering (1-D)

Principle

- Remove high and low frequency components and preserve intermediary frequencies
- Example:



Ideal band-pass filter (2-D)

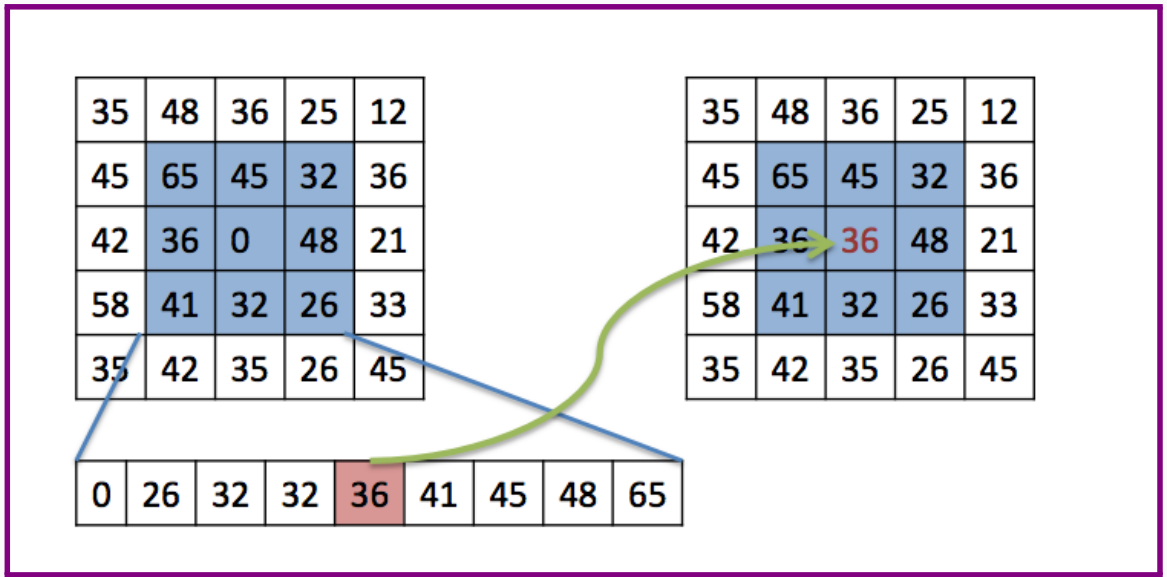


Outline

Median filter

Definition

- Let $a_1, a_2, \ldots, a_N$ (N odd) be a discrete sequence. The median value of this sequence is a_i if:
  - there exist $\frac{N-1}{2}$ elements a_j smaller than a_i
  - there exist $\frac{N-1}{2}$ elements a_j greater than a_i
- Consider a sliding window of odd size on the image support
- Replace the value of the central pixel by the median value of pixels inside the window.



Median filter

Properties

- Well suited to remove "Salt and Pepper" noise
- Becomes inefficient if a majority of pixels is corrupted inside the window
- Preserves edges (but not their localization)
- Also reduces uniform and Gaussian additive noise

Median filter

Median filter versus Gaussian smoothing

Median filter versus Gaussian smoothing

Median filter versus Gaussian smoothing

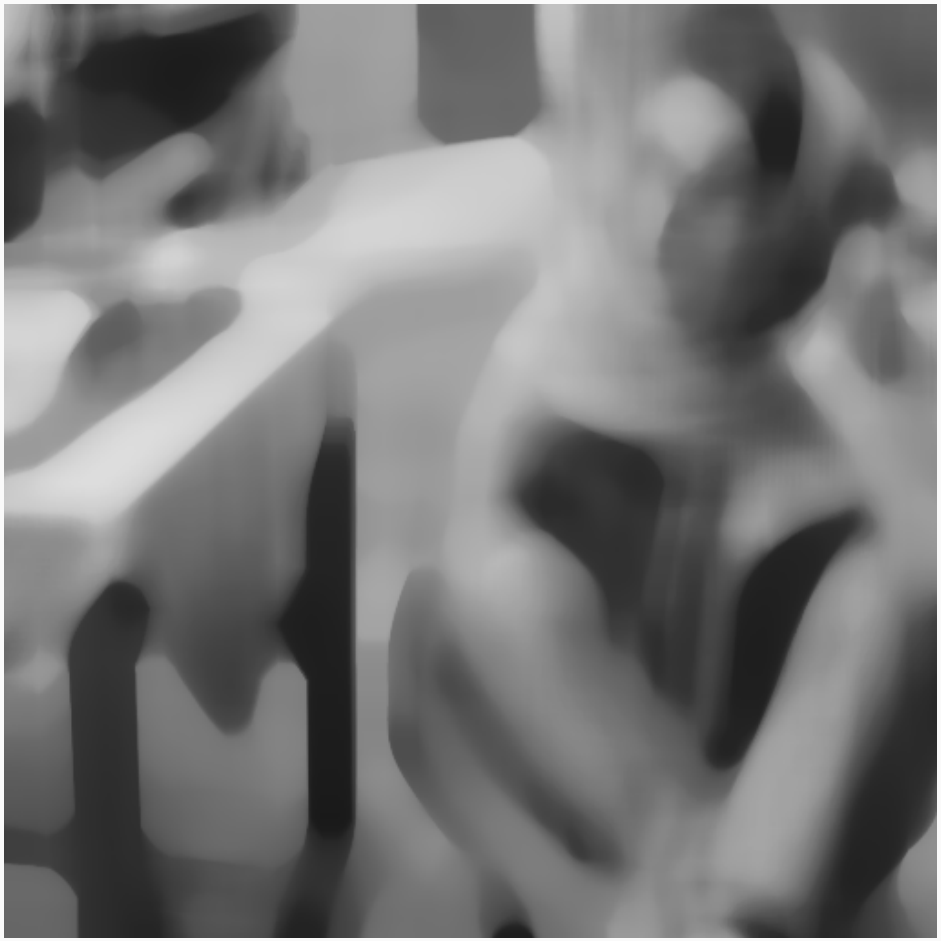
Advanced use of median filter

Original image

Advanced use of median filter

Median 15×15

Advanced use of median filter



Median 31×31

Advanced use of median filter

Median 3×3 iterated 15 times

Advanced use of median filter

Median 7×7 iterated 5 times

Median filter

Question/Exercise

- Why is the median filter non-linear?

Other non linear filter: rank filters

Definition: maximal filter

- Removes "pepper" noise:
- $f_{\max}(x, y) = \max_{(i,j) \in \text{neighborhood}(x,y)} f(i, j)$
- Morphological dilation

Definition: minimal filter

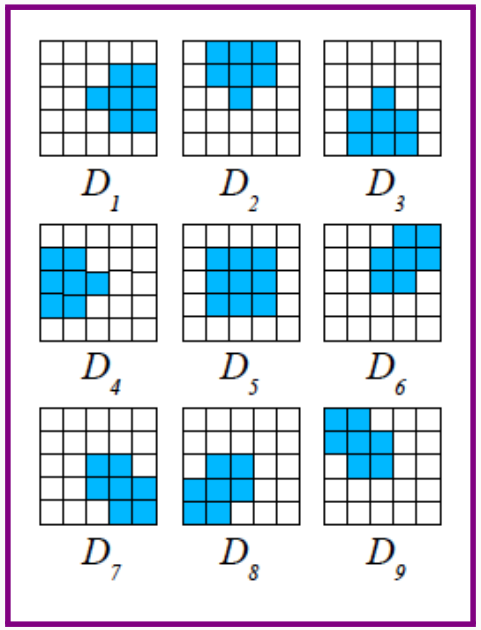
- Removes "salt" noise:
- $f_{\min}(x, y) = \min_{(i,j) \in \text{neighborhood}(x,y)} f(i, j)$
- Morphological erosion

- Median, min and max are rank filters
- Morphological filters: $f_{\min} \circ f_{\max}$, $f_{\max} \circ f_{\min}$, etc.

Nagao filter

Definition

- Window of size 5×5 centered on the pixel to process, 9 masks are defined
- For each mask D_i computes mean μ_i and variance σ_i
- Consider the minimal variance σ_i and set the pixel value to μ_i



Nagao filter: example

Original image

Gaussian noise, small σ

Gaussian noise, large σ