Introduction to Computer Vision

Matthieu Baudry - Nov. 2020 matthieu.baudry@airbus.com

Download documents on the LMS

Objectives and Directives

- Understand **the basis** of Computer Vision

- Be able to
  - Modify images
  - Extract information
  - Apply Fourier Transform
  - Filter images
  - Apply deblurring

- Evaluation:
  - Report & clear code
  - Date: in 2 weeks (13th of December)
  - Notation:

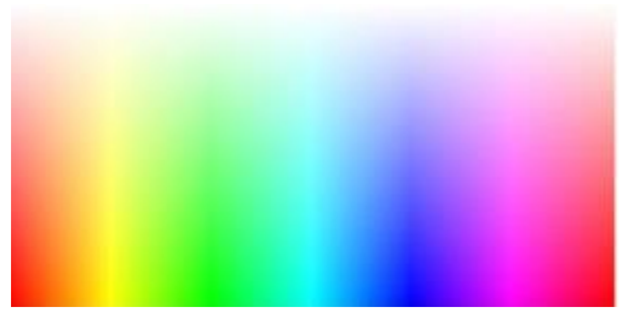
    Quality of redaction, form (put clear illustrations!)

    Quality and pertinence of comments

    “Extra work” for bonus (<u>**test other images, filters, applications**</u>, etc.)

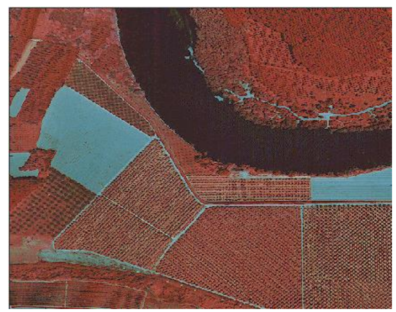
Framework

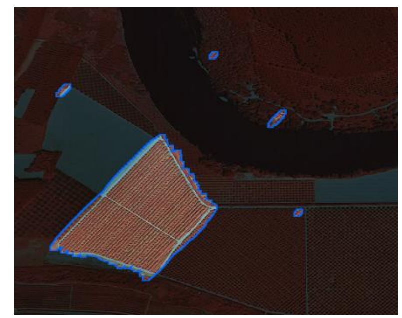
I – Basics



- GrayScale Image
- Coloured Image
- Histogram
- Filtering

II – Fourier Transform



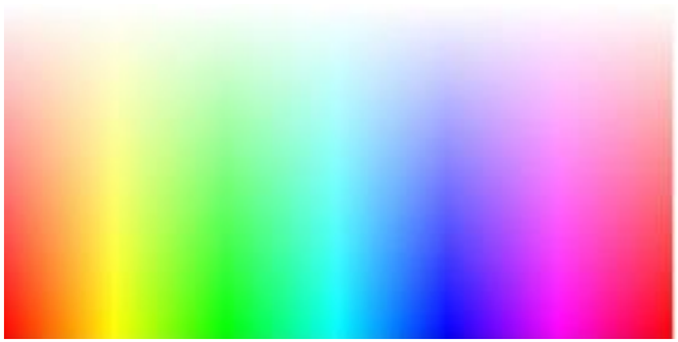


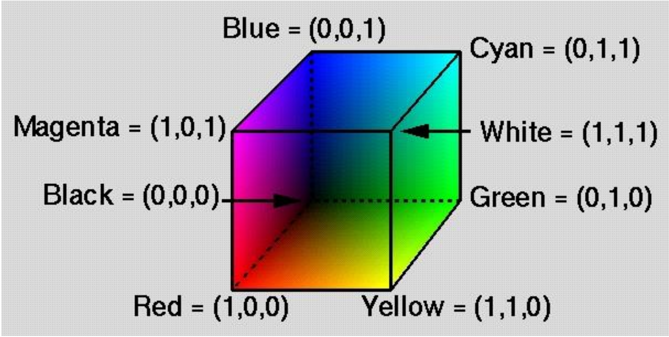
III - Deblurring

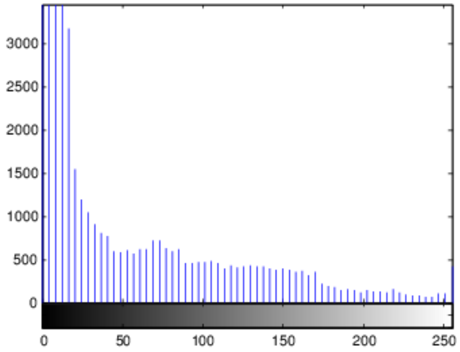
I - Basics

I – Basics

- GrayScale Image
- Coloured Image
- Histogram
- Filtering





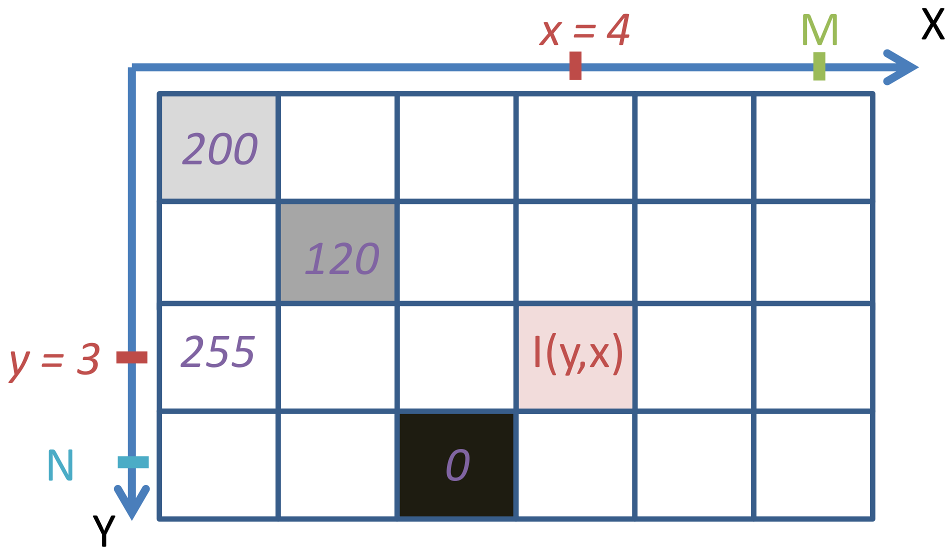


I - Basics

- GrayScale Image I of size (N*M) = Matrix of N*M*1 elements, whose values belongs to [0:255] (for UINT8, [0:1] for double)

- Lowest value = Dark, Highest value = white

- Ex: N = 4, M = 6



<u>Convention depends on languages</u>

- in C, indexes begin at 0 and end at N-1;
- In MatLab, indexes begin at 1 and end at N;
- access might be I(x,y) instead of I(y,x);
- etc.

I - Basics

Matlab commands:

- Create an image:
  - I = zeros(N,M); *% if N=M, I = zeros(N)*
  - I = ones(N,M); *% Matrix filled with 1*
  - I = eye(N); *% Diagonal matrix*
  - I = ones(N,M,3); *% N*M*3 matrix (for RGB or HSV images for instance)*

- Modify pixels value:
  - I(n,m) = new_value; *% row n, column m*
  - I = 255*I; *% Multiplies all elements of I by 255 (as a scalar)*
  - I3 = I1*I2; *% Normal matrix operation*
  - I3 = I1.*I2; *% Element per element operation*
  - I(n1:n2:m1:m2) = I2; *% Put I2 in I.* ⚠ *be careful to have correct sizes*

- Load an image:
  - I = imread('path/image.png');

- Display an image:
  - figure, imshow(I) ; or imagesc(I) ; *% imagesc scales the color levels to values of I*
  - close all

- Save an image:
  - imwrite(I,'MyImage.png','png') ;

- Help:
  - help imwrite;
  - doc imwrite;

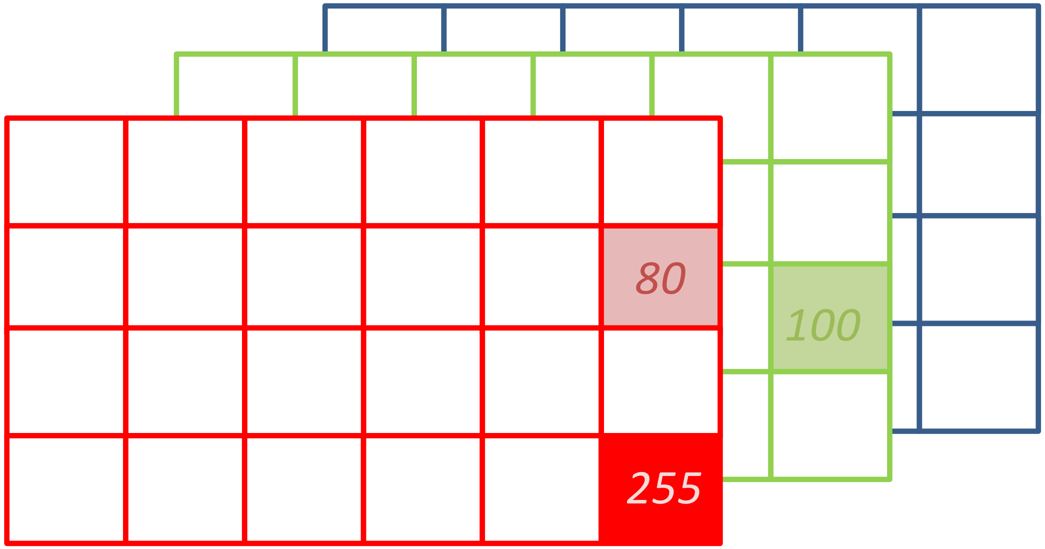
See Code section 1 (and video 1)

I - Basics

- Coloured Image I of size (N*M) = Matrix of N*M*3 elements
- Different color-spaces exist: (RGB), (HSV), (CMYK), …
- Ex: RGB

Principle: 1 Matrix for each Red, Green or Blue component

I =

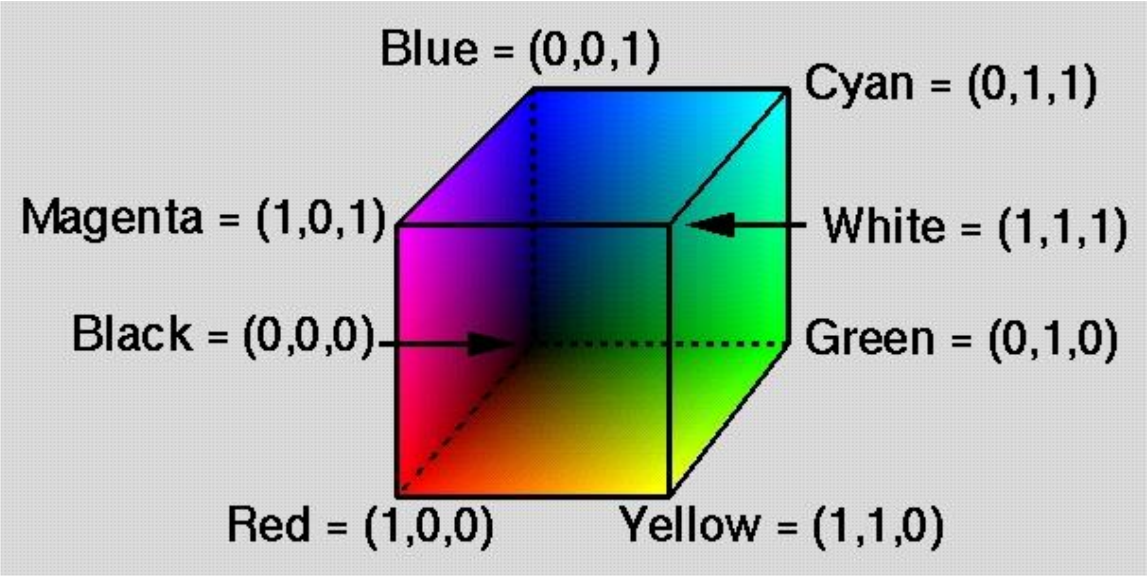


- $I_{red} = I(:,:,1)$
- $I_{green} = I(:,:,2)$
- $I_{blue} = I(:,:,3)$

I - Basics

- Coloured Image I of size (N*M) = Matrix of N*M*3 elements
- Different color-spaces exist: (RGB), (HSV), (CMYK), …
- Ex: RGB

Principle: 1 Matrix for each Red, Green or Blue component

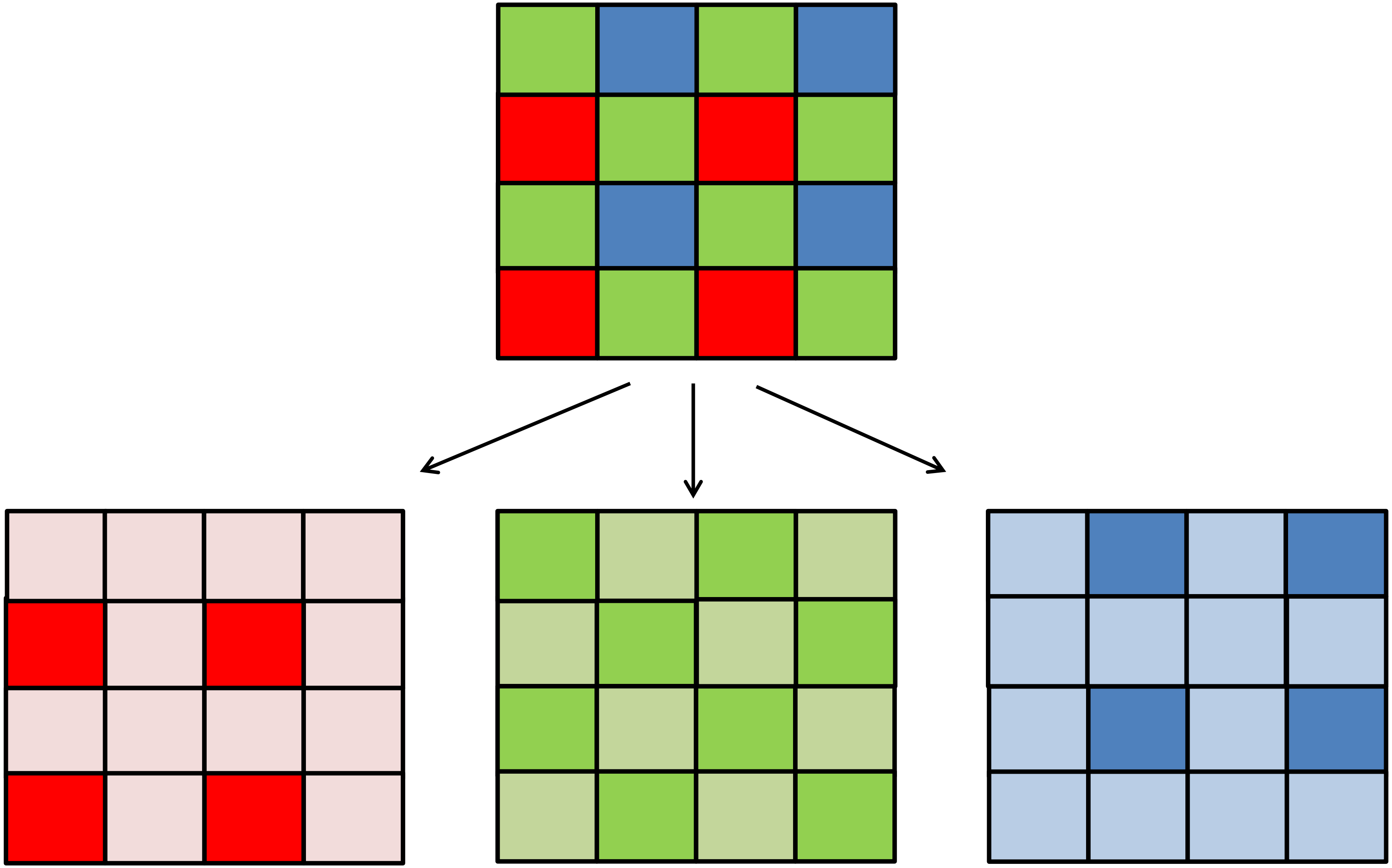


- $I_{red} = I(:,:,1)$
- $I_{green} = I(:,:,2)$
- $I_{blue} = I(:,:,3)$

See Code section 2 (and video 2)

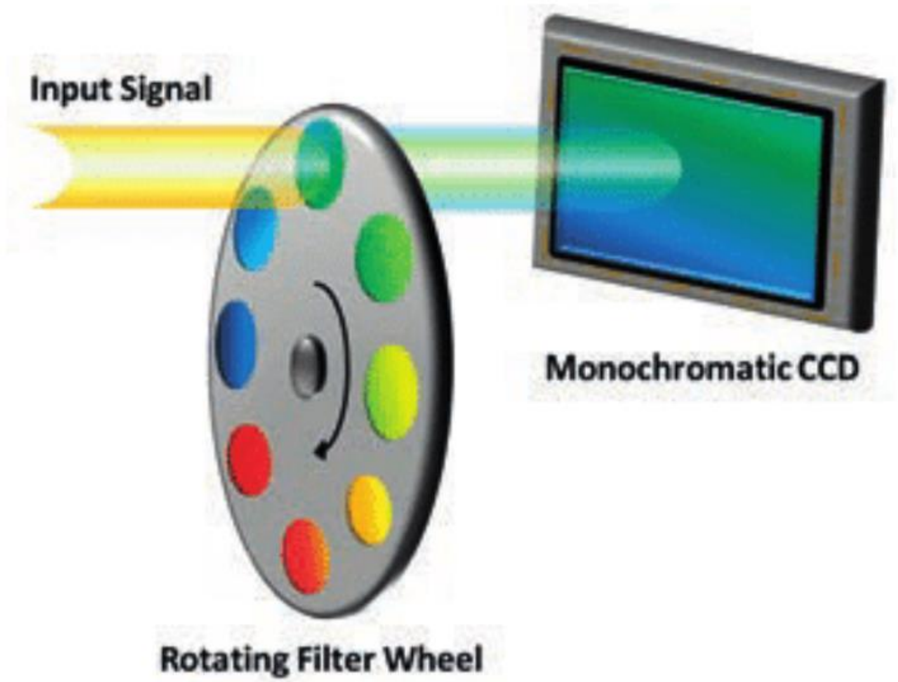
I - Basics

- Acquisition of RGB Image: Bayer matrix + interpolation



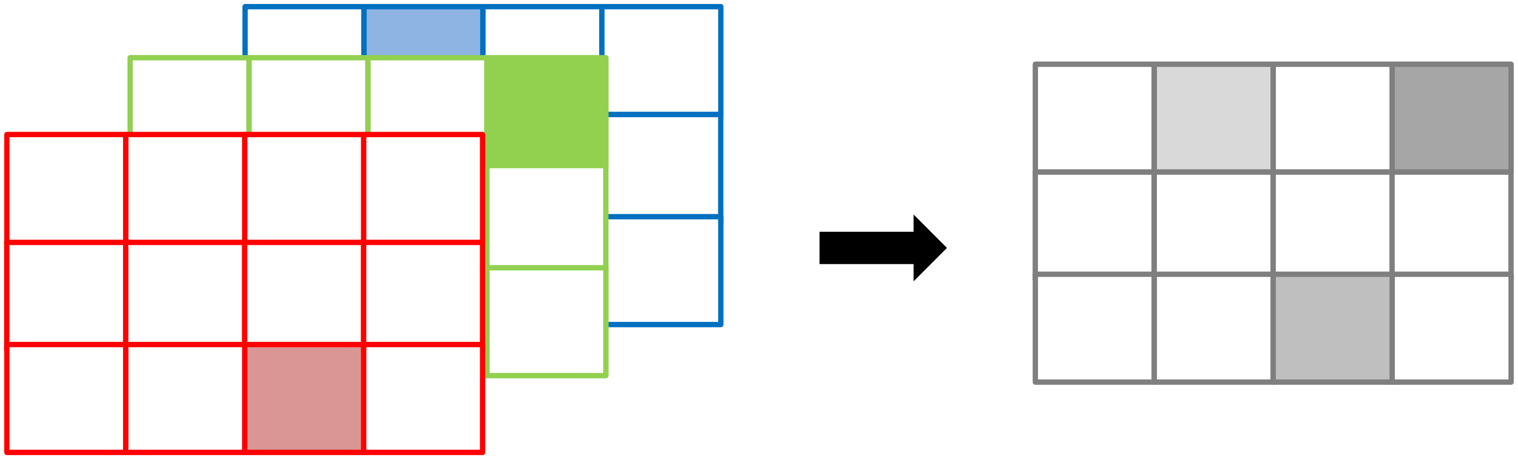
I - Basics

- Acquisition of multispectral Image: Rotating wheel



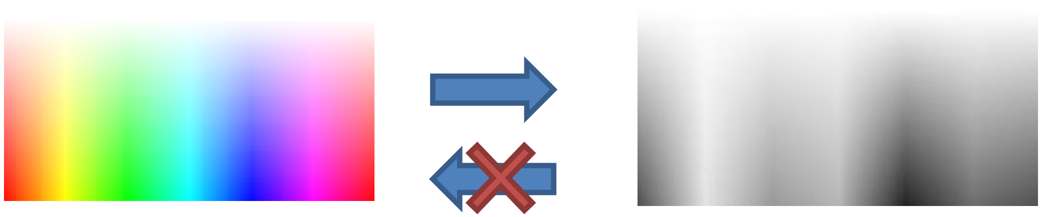
I - Basics

- RGB to Gray



$$I_g = \alpha I_{red} + \beta I_{green} + \gamma I_{blue}$$

This transformation is not bijective (loss of information)

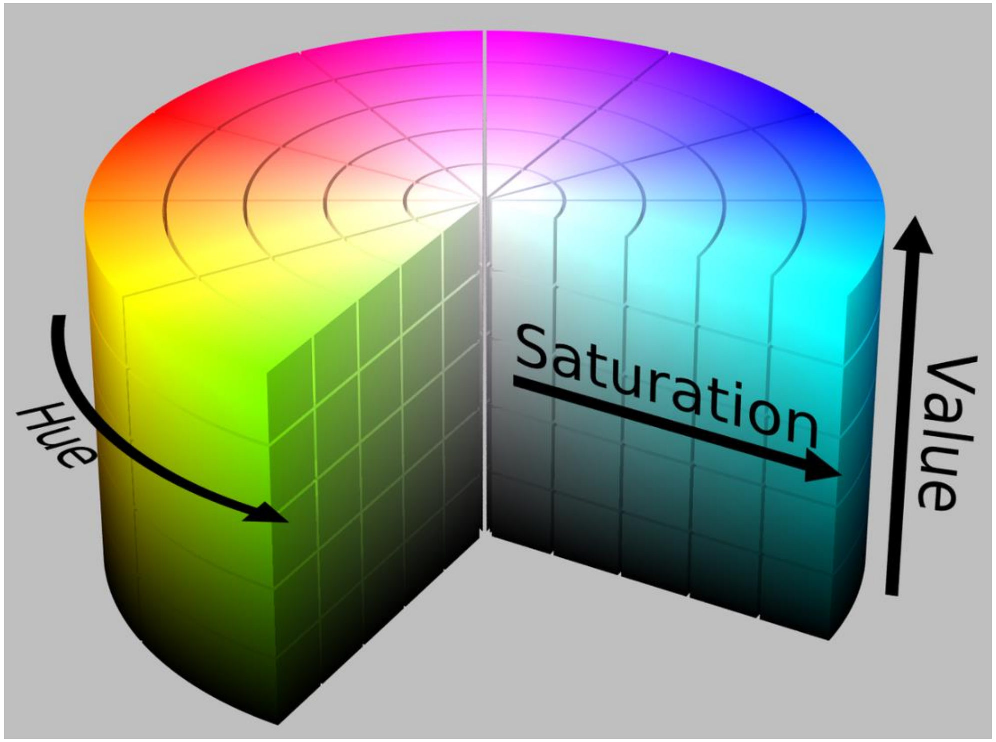


$$I_g = rgb2gray(I_{RGB})$$

I - Basics

- HSV (Hue Saturation Value) Color space

  Hue ≈ Pure color (wavelength) [*teinte* in French]
  Saturation ≈ Intensity of coloration
  Value ≈ The brightness



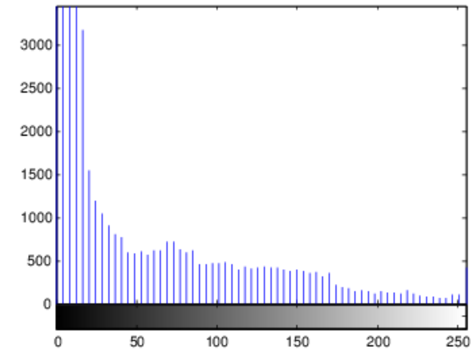
https://en.wikipedia.org/wiki/HSL_and_HSV

➢ $I_{HSV} = rgb2hsv(I_{RGB})$

I - Basics

- Histogram

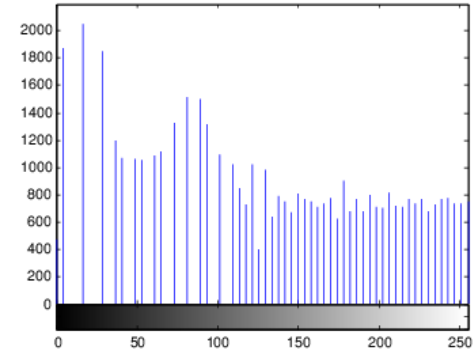
  ➢ Hist $= imhist(I_g)$

  Hist(i) = number of pixels of I that have a value equals to i

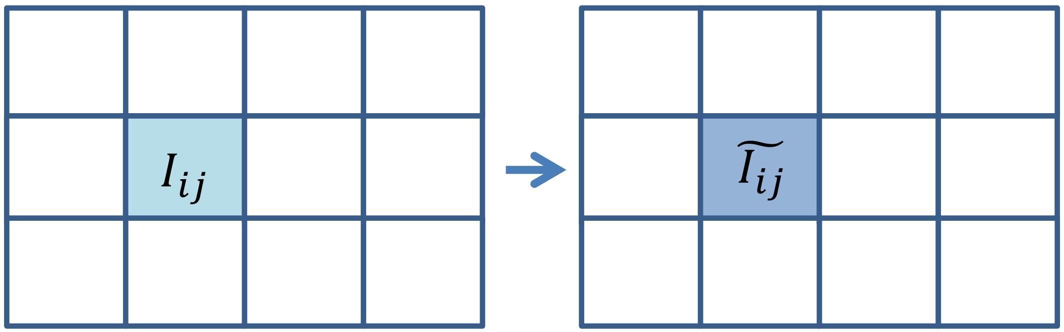
  Gives information about the contrast of the image

  ➢ $I_{eq} = histeq(I_g)$



- **Filtering**

Important part of Image Processing → modify intensity of **each pixel** in accordance with some rule (locally or globally)



Where: $\widetilde{I_{ij}} = f(I, \ldots)$

Used to remove noise, sharpen edges, blur areas, etc.

Ex:

Filtering can be *linear* or *non linear*

I - Basics

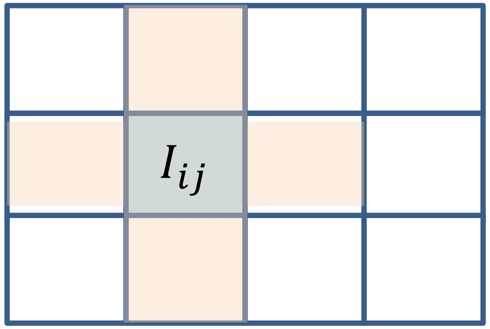
- **Non linear Filtering (1/2)**

Apply a non linear operation to the image

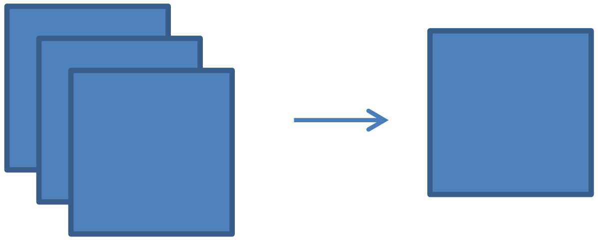
Ex:

$$I_{loc_{min}}(i,j) = \min_{(k,l)\in H}(I(i+k:j+l))$$

$$H : \{(i,j)\ \&\ 4\ closest\ neigbhours\}$$



$$I_{min}(i,j) = \min(I_1(i,j), I_2(i,j), \dots)$$



Other non linear filters:

- max,
- median,
- bilateral filter,
- etc.

I - Basics

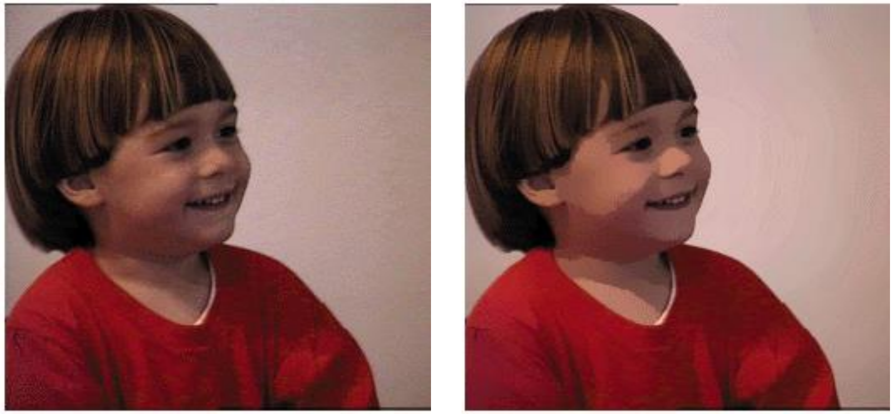
- **Non linear Filtering (2/2)**

Ex:

Median Filtering:

https://en.wikipedia.org/wiki/Median_filter

Bilateral Filtering:



http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

I - Basics

- **Linear Filtering (1/8)**

Apply a linear operation to the image: convolution by a kernel

Lets *H* be a *kernel matrix* (any matrix, size [2n+1,2m+1])

We have :

$I_{filtered} = I * H$, where $*$ is the convolution operator (2D)

Rem. Conv 1D: $(f * g)(x) = \int_{-\infty}^{+\infty} f(t)g(x-t)\,dt$

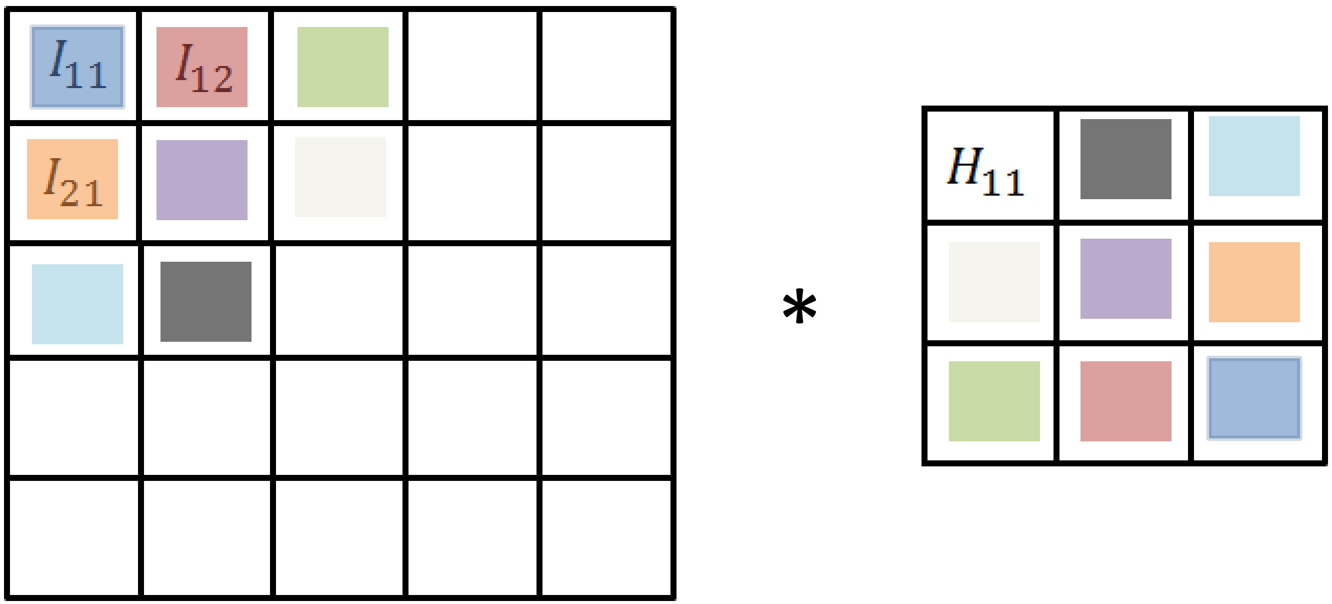
$$I_{filtered}(i,j) = \sum_{k=1}^{2n+1} \sum_{l=1}^{2m+1} H(k,l)\,.\,I(i+n-k+1, j+m-l+1)$$

➢ $I_{filtered} = imfilter(I, H, [options])$

I - Basics

- **Linear Filtering (2/8)**

Ex: $$I_{filtered}(i,j) = \sum_{k=1}^{2n+1} \sum_{l=1}^{2m+1} H(k,l)\,.\,I(i+n-k+1, j+m-l+1)$$



$$I_{filtered}(2,2) = I_{11}H_{33} + I_{12}H_{32} + I_{13}H_{31} + I_{21}H_{23} + I_{22}H_{22} + I_{23}H_{21} + I_{31}H_{13} + I_{32}H_{12} + I_{33}H_{11}$$

≠ correlation: $$I_{corr}(i,j) = \sum_{k=1}^{2n+1} \sum_{l=1}^{2m+1} H(k,l)\,.\,I(i-n+k-1, j-m+l-1)$$

Specify option 'conv' with matlab, otherwise it performs correlation (see code sections 3&4).

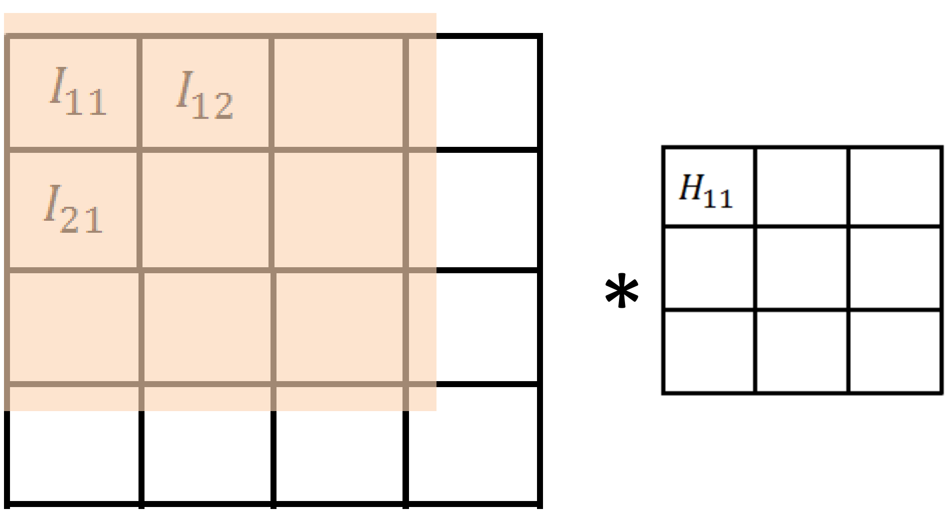
I - Basics

- **Linear Filtering (3/8)**

Ex: simple kernels

- If $H = h$ (ie scalar), then $I_{filtered} = H * I = h.I$
- If $H = \frac{1}{9}.\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, then $I_{filtered}(2,2) = \frac{1}{9}I_{11} + \frac{1}{9}I_{12} + \cdots + \frac{1}{9}I_{33}$

$$= average(I_{11}: I_{33})$$

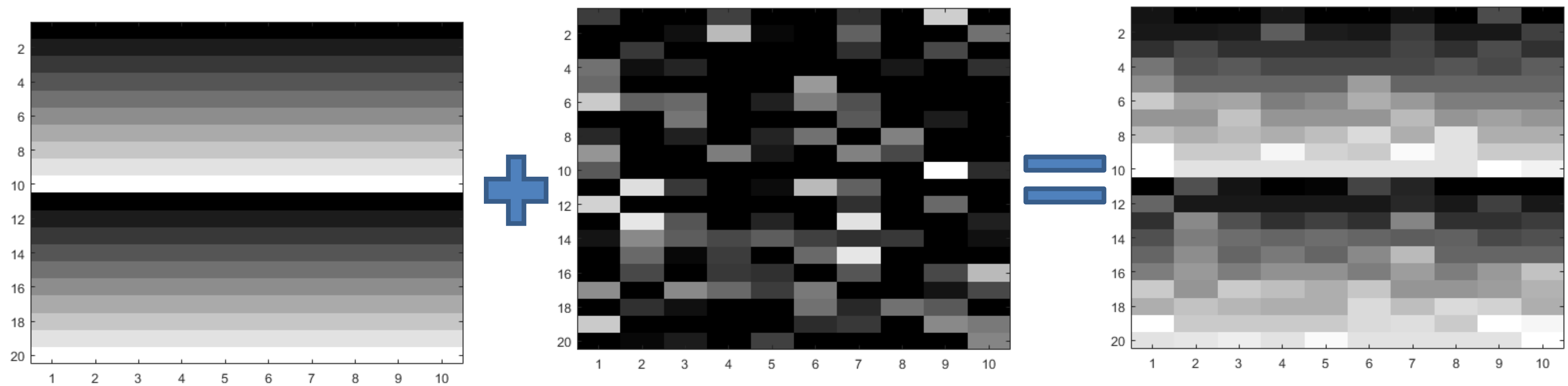


I - Basics

- **Linear Filtering (4/8)**

Ex: Smooth noise in image

$$I_{noisy} = I + I_{noise} \quad \textit{with} \quad I_{noise}(i,j) \sim \mathcal{N}(0,\sigma^2) \;\; \forall i,j$$



How to retrieve I from $\mathrm{I_{noisy}}$?

→ Use properties:

If $(x_i)_{i\in[1,m]} \sim [\mathcal{N}(0,\sigma^2)]^m$ then: $\frac{1}{m}\sum x_i \sim \mathcal{N}(0,(\sigma')^2)$ with $\sigma' = \frac{\sigma}{\sqrt{m}} \to 0 \; as \;\; m \to \infty$

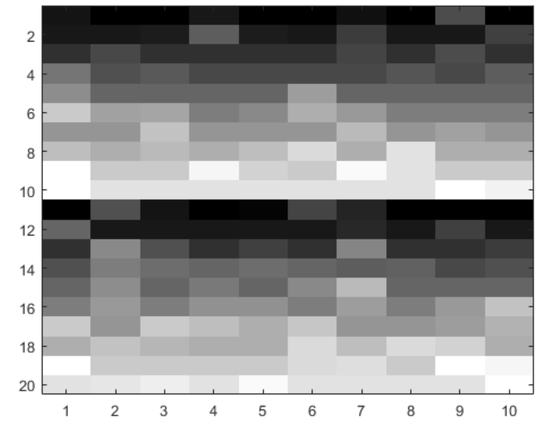
- **Linear Filtering (5/8)**

Ex: Smooth noise in image

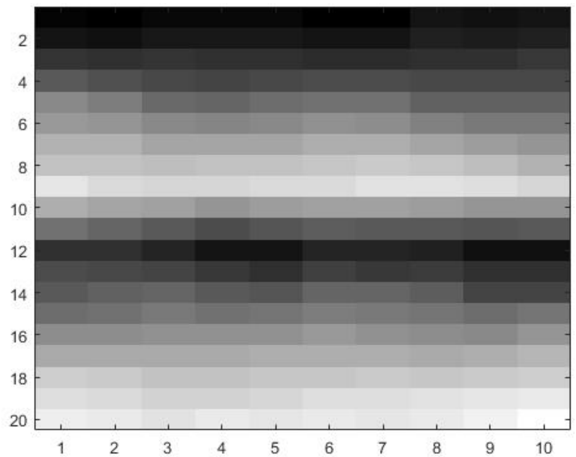
How to retrieve I from I_{noisy}?

→ Average the image, for instance with a mean operator.

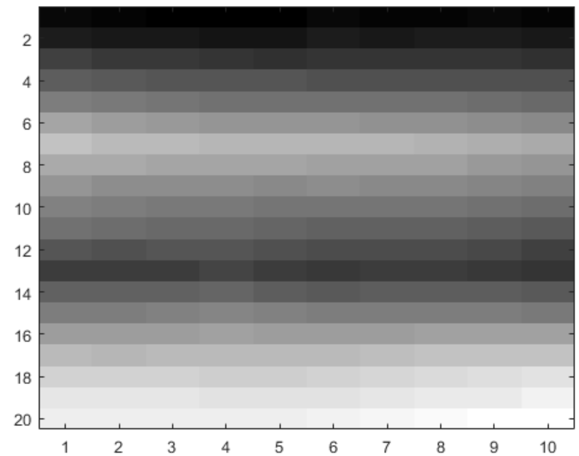
Concretely:



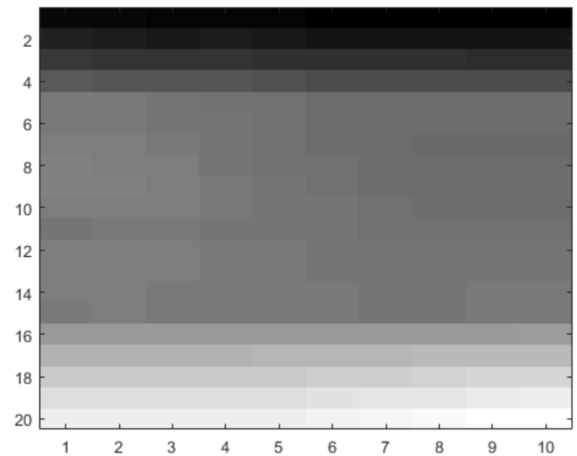
use $H = \frac{1}{m^2} * \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$



m = 3



m = 6



m = 10

→ Prefer Gaussian Kernels which better preserve edges

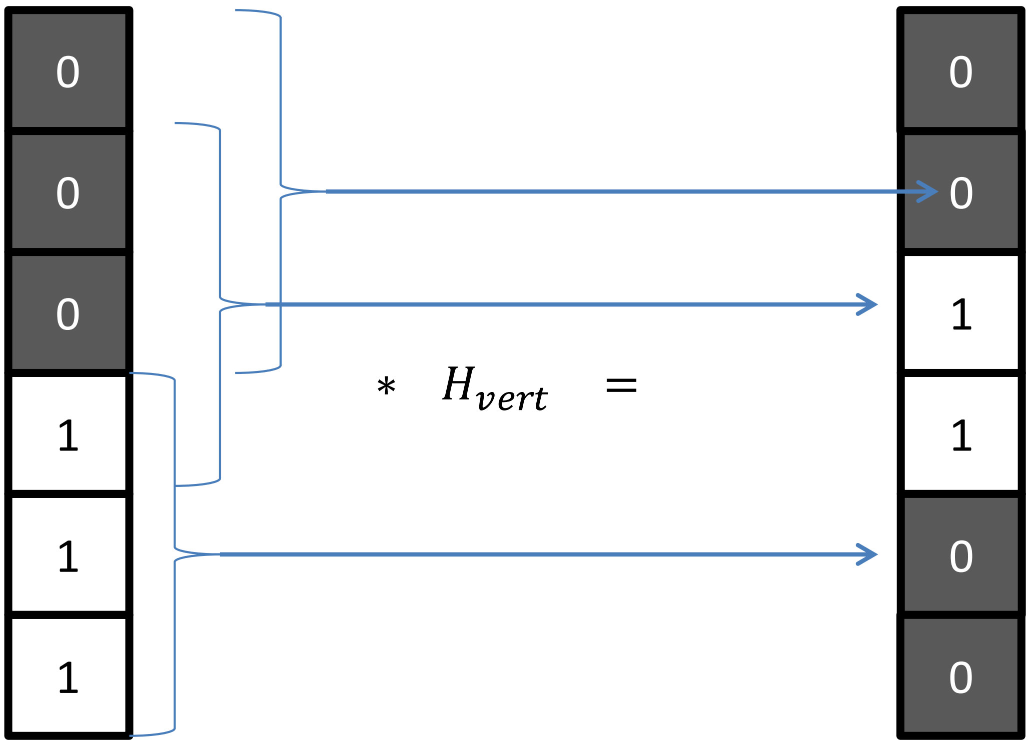
… See implemented code, section 3 (and video 3)

I - Basics

- **Linear Filtering (7/8)**

Ex 2 : Gradient of an image (vertical)

$$H_{vert} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$



$0 * (-1) + 0 * 1$

$0 * (-1) + 1 * 1 = 1$

$1 * (-1) + 1 * 1$

… See implemented code, section 4 (and videos 4 &4b)

I - Basics

- **Linear Filtering (7/8)**

Property:

$$H_1 * (H_2 * I) = (H_1 * H_2) * I$$ (here, * operator is still convolution)

For instance:

Instead of smoothing the image **and** generate gradient (which takes time), use one Kernel that combines both operations: $H_3 = H_2 \ * H_2$

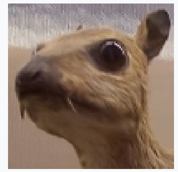
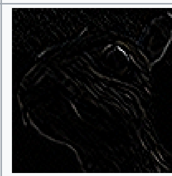
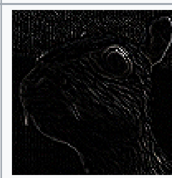
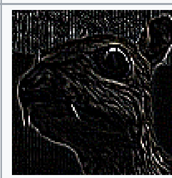
$$\left.\begin{aligned} H_1 &= [-1\ 0\ 1] \text{ (gradient)} \\ H_2 &= \frac{1}{3} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \text{(smooth)} \end{aligned}\right\} \quad H_3 = \frac{1}{3} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \downarrow$$

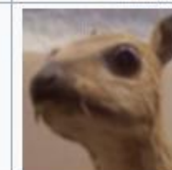
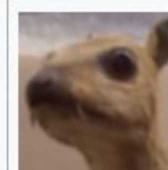
Noise Smoothing

$\longrightarrow$

Edge detection

- **Linear Filtering (8/8)**

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)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

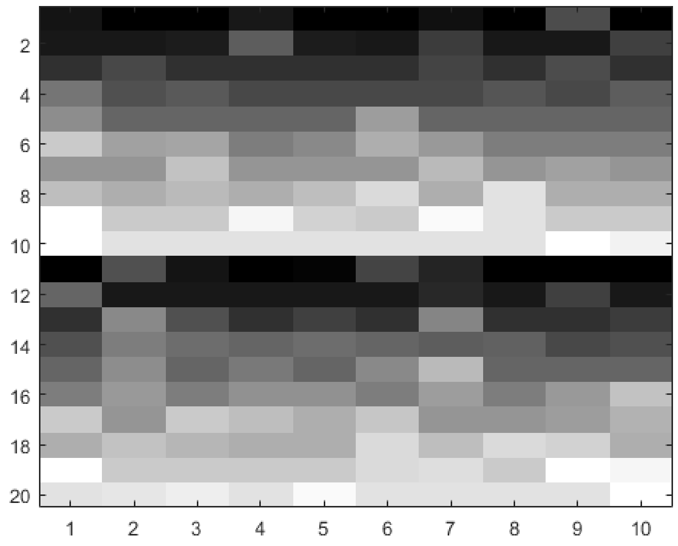
https://en.wikipedia.org/wiki/Kernel_(image_processing)

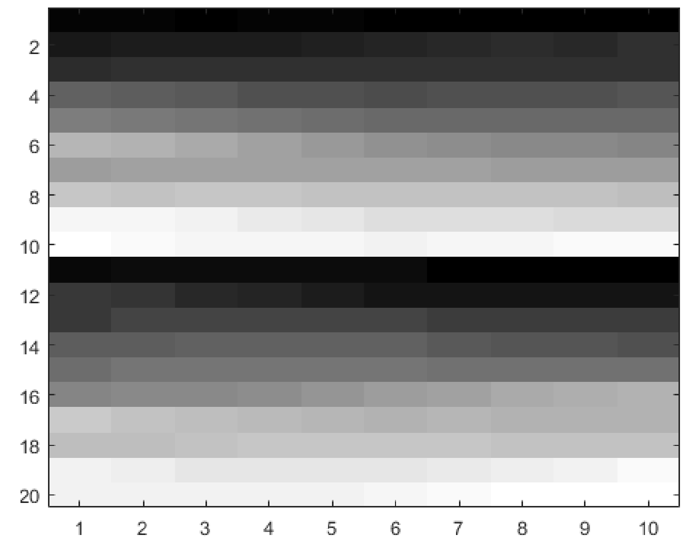
I - Basics

Adapt your filters to your problems!

Ex: Smooth noise in image

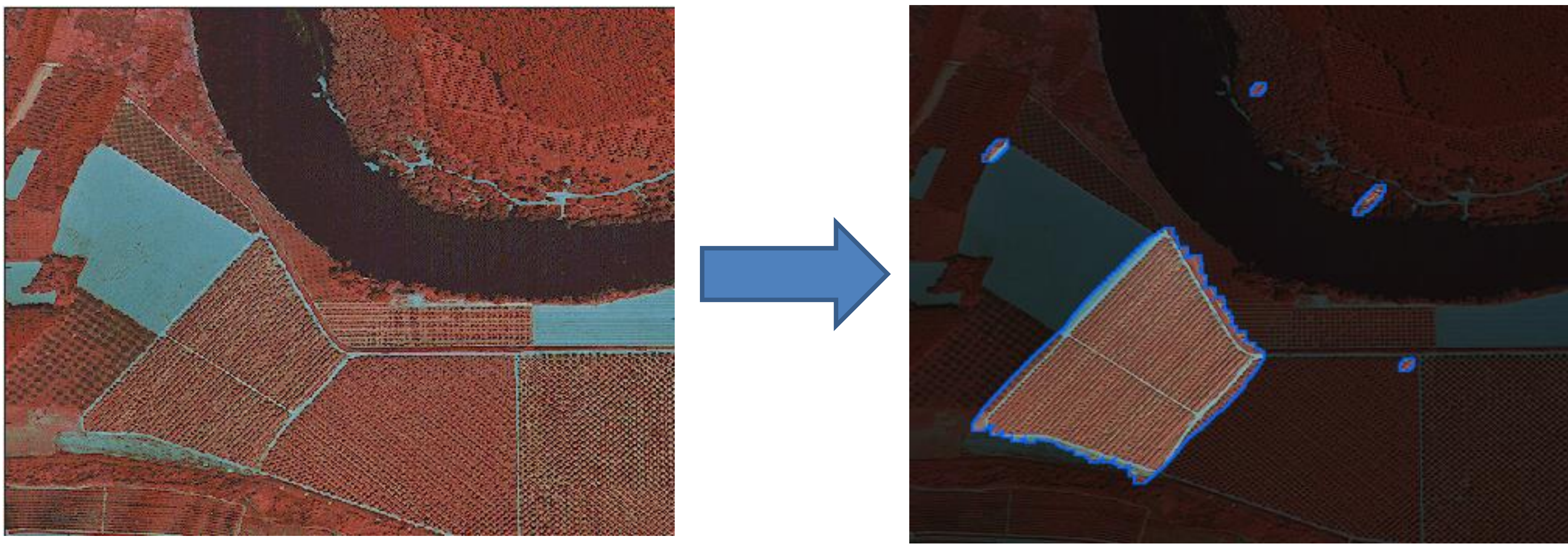
$$with\ H = \frac{1}{n} * [1, \ldots, 1]$$





II – Fourier Transform

Frequency analysis



II – Fourier Transform

Reminder: in 1 Dimension

If x is T-periodic

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}\left(a_k \cos\left(\frac{2k\pi t}{T}\right) + b_k \sin\left(\frac{2k\pi t}{T}\right)\right)$$

$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

II – Fourier Transform

Reminder: in 1 Dimension

$$< x(t), \cos\left(\frac{2k\pi t}{T}\right) >$$

If x is T-periodic

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k \cos\left(\frac{2k\pi t}{T}\right) + b_k \sin\left(\frac{2k\pi t}{T}\right))$$

$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

II – Fourier Transform

Reminder: in 1 Dimension

$$< x(t), \cos\left(\frac{2k\pi t}{T}\right) >$$

If x is T-periodic

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k \cos\left(\frac{2k\pi t}{T}\right) + b_k \sin\left(\frac{2k\pi t}{T}\right))$$

$$= \sum_{k=-\infty}^{\infty} r_k \cos(\frac{2k\pi t}{T} + \theta_k)$$

$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

II – Fourier Transform

Reminder: in 1 Dimension

$$< x(t), \cos\left(\frac{2k\pi t}{T}\right) >$$

If x is T-periodic

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k \cos\left(\frac{2k\pi t}{T}\right) + b_k \sin\left(\frac{2k\pi t}{T}\right))$$

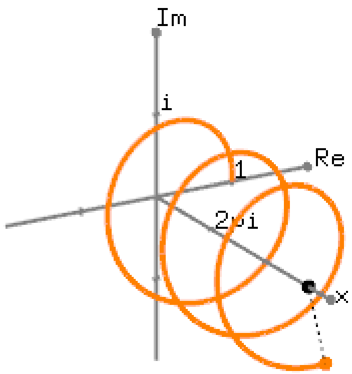
$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

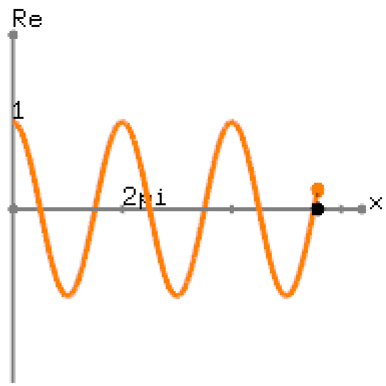
$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

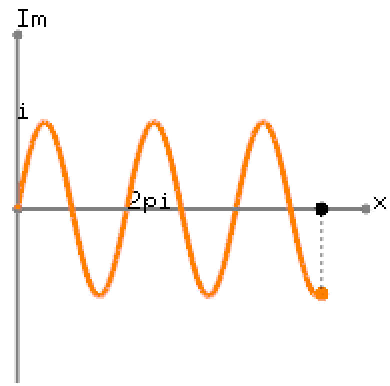
$$= \sum_{k=-\infty}^{\infty} r_k \cos(\frac{2k\pi t}{T} + \theta_k)$$

$$c_k = \frac{1}{T}\int_0^T x(t)\exp\left(\frac{-2i\pi kt}{T}\right)dt$$

$$= \sum_{k=-\infty}^{\infty} c_k \exp(\frac{2ik\pi t}{T})$$







II – Fourier Transform

Reminder: in 1 Dimension

$$< x(t), \cos\left(\frac{2k\pi t}{T}\right) >$$

If x is T-periodic

$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k\cos\left(\frac{2k\pi t}{T}\right) + b_k\sin\left(\frac{2k\pi t}{T}\right))$$

$$= \sum_{k=-\infty}^{\infty} r_k\cos(\frac{2k\pi t}{T} + \theta_k)$$

$$= \sum_{k=-\infty}^{\infty} c_k\exp(\frac{2ik\pi t}{T})$$

$$c_k = \frac{1}{T}\int_0^T x(t)\exp\left(\frac{-2i\pi kt}{T}\right)dt$$

$$=< x(t), \exp\left(\frac{-2i\pi kt}{T}\right) >$$

Scalar product measures **similarity**

II – Fourier Transform

Reminder: in 1 Dimension

$$< x(t), \cos\left(\frac{2k\pi t}{T}\right) >$$

If x is T-periodic

$$a_k = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2k\pi t}{T}\right)dt$$

$$b_k = \frac{2}{T}\int_0^T x(t)\sin\left(\frac{2k\pi t}{T}\right)dt$$

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k\cos\left(\frac{2k\pi t}{T}\right) + b_k\sin\left(\frac{2k\pi t}{T}\right))$$

$$= \sum_{k=-\infty}^{\infty} r_k \cos(\frac{2k\pi t}{T} + \theta_k)$$

$$c_k = \frac{1}{T}\int_0^T x(t)\exp\left(\frac{-2i\pi kt}{T}\right)dt$$

$$= \sum_{k=-\infty}^{\infty} c_k \exp(\frac{2ik\pi t}{T})$$

$$=< x(t), \exp\left(\frac{-2i\pi kt}{T}\right) >$$

$$= \sum_{k=-\infty}^{\infty} c_k \exp(2i\pi k f_0 t)$$

$$=< x(t), \exp(-2i\pi k f_0 t) >$$

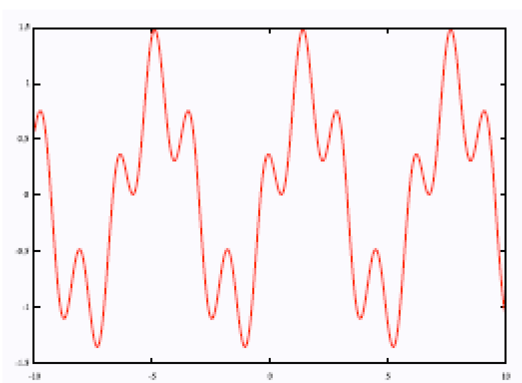
II – Fourier Transform

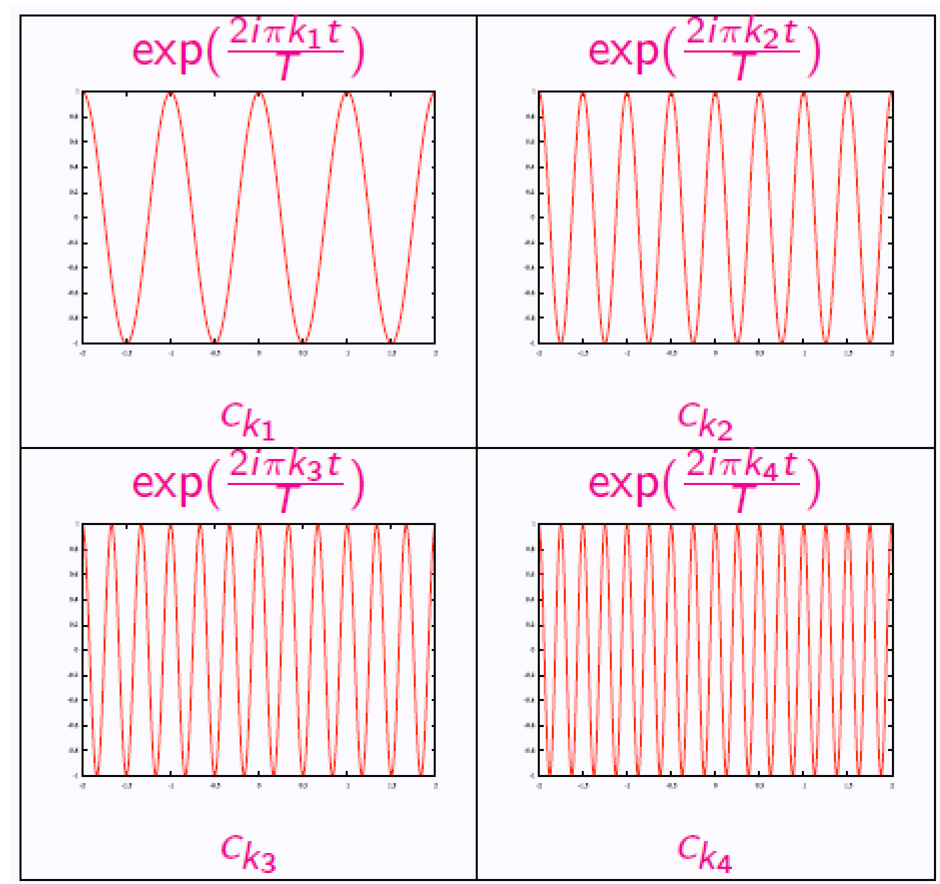
$$x(t) = \sum_{k=-\infty}^{\infty} c_k \exp(\frac{2ik\pi t}{T})$$

$$c_k =< x(t), \exp\left(\frac{-2i\pi kt}{T}\right) >$$

$angle(c_k)$ = phase between x(t) and $\exp(2i\pi k f_0 t)$

$|c_k|$ = intensity of frequency $\mathrm{k}f_0$ in x(t)





II – Fourier Transform

In 2D: $TF: I \rightarrow TF(I) = \hat{I} \in \mathbb{C}$

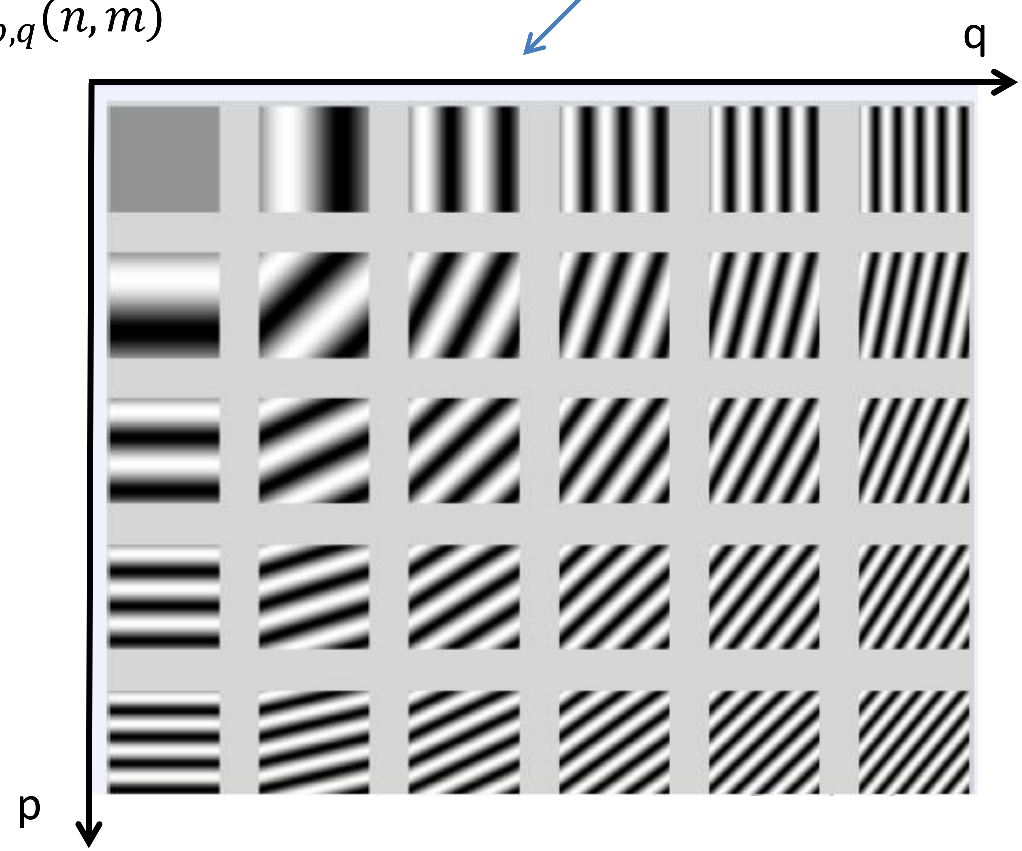
2 main frequencies: $f_{0_{rows}} = \frac{1}{N}$ and $f_{0_{cols}} = \frac{1}{M}$

$$\hat{I}(p,q) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I(n,m).e^{-j\left(\frac{2\pi}{N}\right)pn} e^{-j\left(\frac{2\pi}{M}\right)qm}$$

$$\hat{I}(p,q) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I(n,m).B_{p,q}(n,m)$$

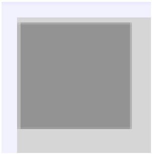
$$\hat{I}(p,q) = < I, B_{p,q} >$$

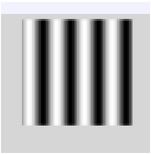
$|B_{p,q}(n,m)|$

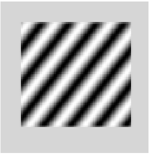


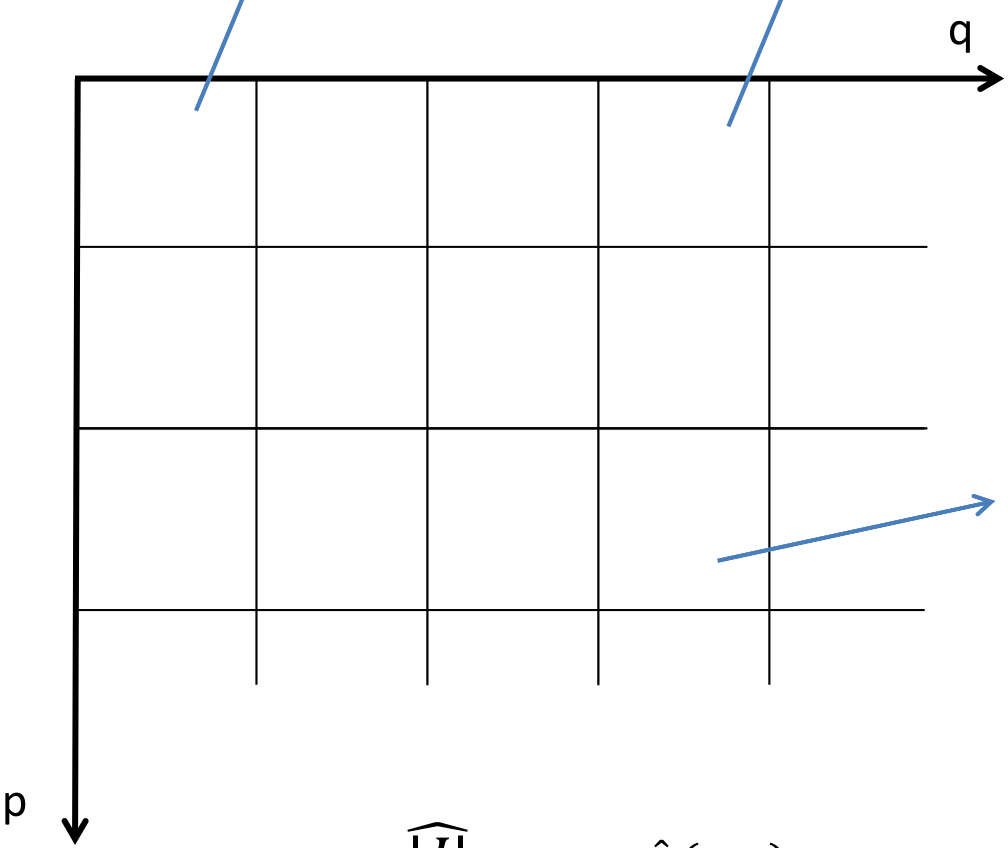
Base du traitement des images, Nicolas Thome, p34

II – Fourier Transform

$= | <I,$  $>|$

$= | <I,$  $>|$

$= | <I,$  $>|$



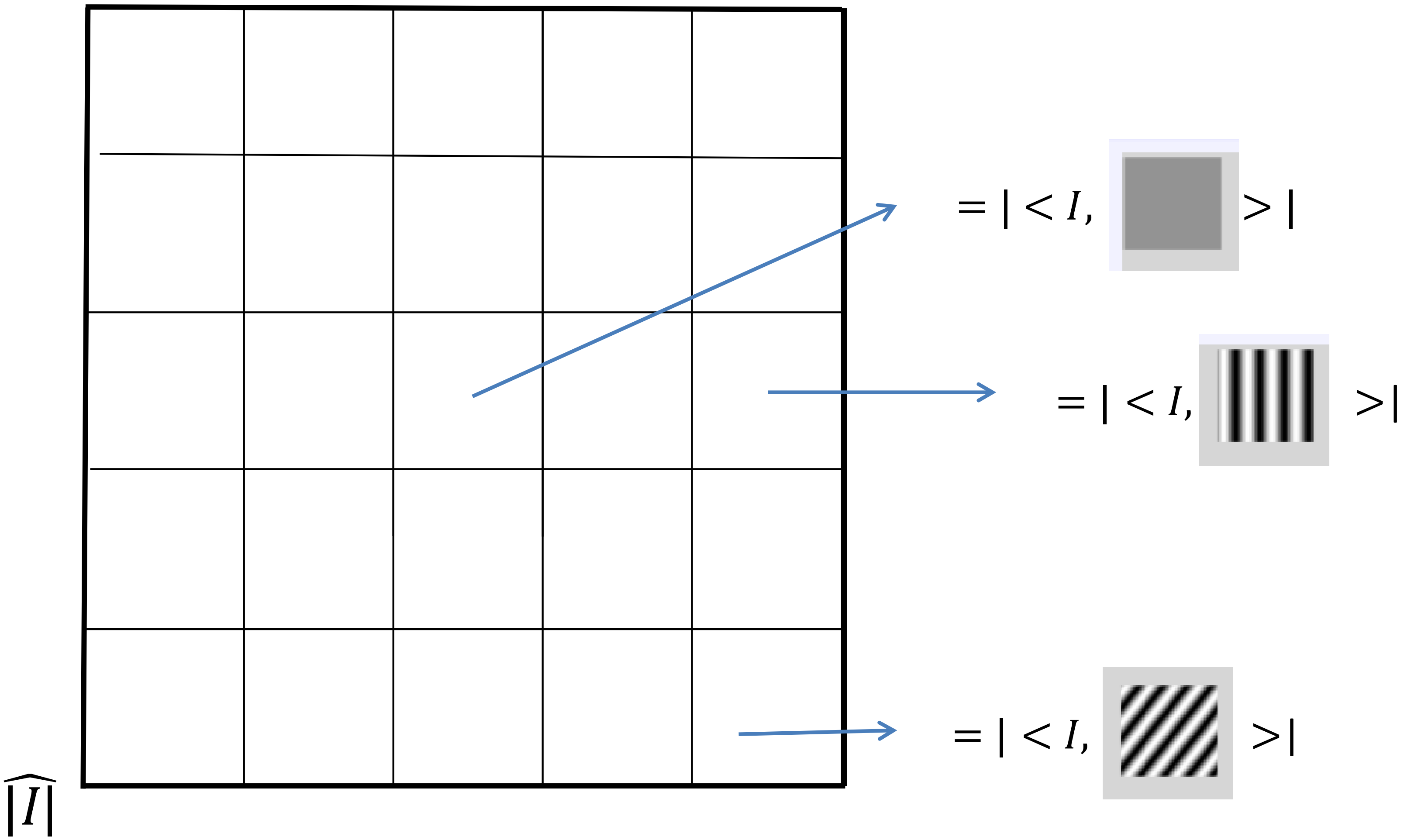
$\widehat{|I|}$

$\hat{I}\,(p,q) = <I, B_{p,q}>$

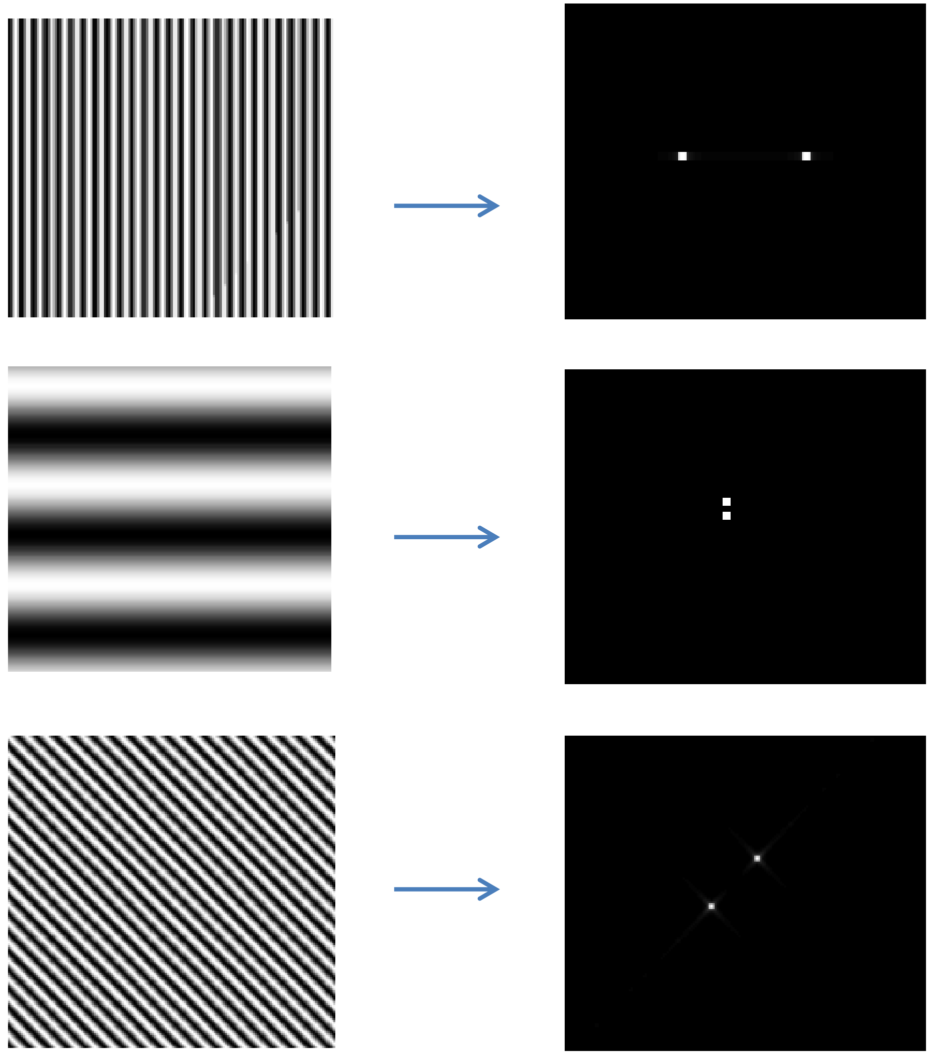
II – Fourier Transform

Warning, conventionally, the low freq are in the middle, and high freq away from the center

Done with function fftshit



II – Fourier Transform



I $\qquad\qquad$ $|\widehat{I}|$

See Code section 5 intro
(and video 5 intro)

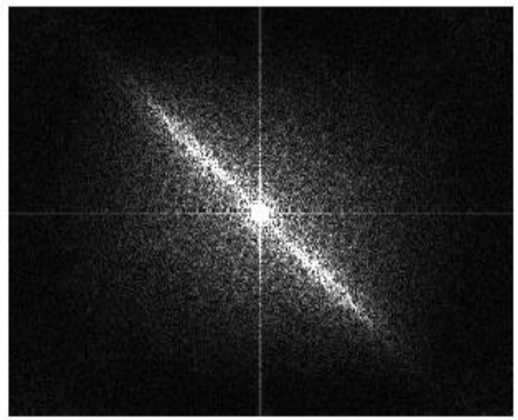
Why 2 points?
see:
https://dsp.stackexchange.com/questions/4825/why-is-the-fft-mirrored

II – Fourier Transform

$TF: I \rightarrow TF(I) = \hat{I} \in \mathbb{C}$

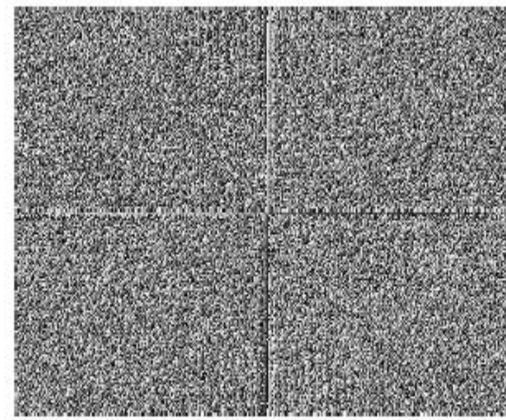
$$F(p,q) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I(n,m).e^{-j(\frac{2\pi}{N})pn}.e^{-j(\frac{2\pi}{M})qm}$$

Image u



Module $|\hat{u}|$

$\approx$ *Main directions/Patterns*



Phase $\arg(\hat{u})$

$\approx$ *Texture spatial positionning*

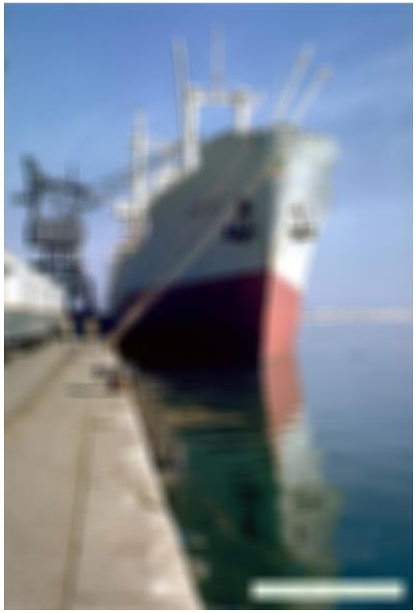
- $I_{fft} = fftshift(fft2(I))$
- $imagesc(log(abs(I_{fft})))$ % Log in order to see more details
- $I = ifft2(fftshift(I_{fft}))$

Important : $TF(H * I) = TF(H) \times TF(I)$

See Code section 5 (and video 5)

http://www.math-info.univ-paris5.fr/~bgalerne/m2mm/cours_tfd.pdf

III – Deblurring



$$I_b = H * I + N$$

Filtering: N is attenuated

$$I_{b_{filtered}} = H * I + n$$

$$I_{deblurred} = \frac{I_{b_{filtered}}}{H_{est}}$$

Problems:

- How to efficiently remove N (filtering)?
- How to correctly model H?

Annexe – TF for non periodic signals

If x is not T-periodic, replace $\frac{k}{T} = kf_0$ with f, $\sum_k$ with $\int_f$, and c_k with $\hat{X}(f)$

$$c_k = \langle x(t), \exp(-2i\pi k f_0 t) \rangle \longrightarrow \hat{X}(f) = \langle x(t), \exp(-2i\pi f t) \rangle$$

$$x(t) = \sum_{k=-\infty}^{\infty} c_k \exp(2i\pi k f_0 t) \longrightarrow x(t) = \int_{f=-\infty}^{\infty} X(f) \exp(2i\pi f t)\, df$$

$$TF: x \to TF(x) = \hat{X} \in \mathbb{C}$$

$|\hat{X}(f)|$ = intensity of frequency f in x(t)

$angle(\hat{X}(f))$ = phase between x(t) and $\exp(2i\pi f t)$