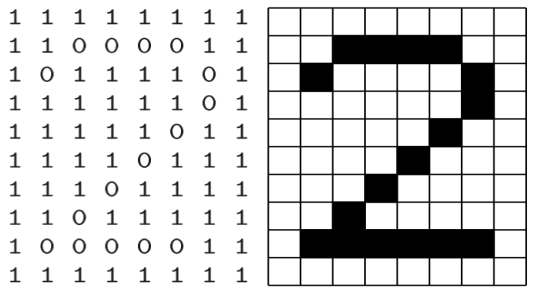
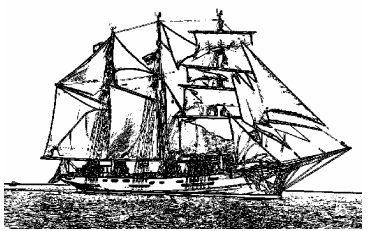
MAP201 : Découverte des Mathématiques Appliquées

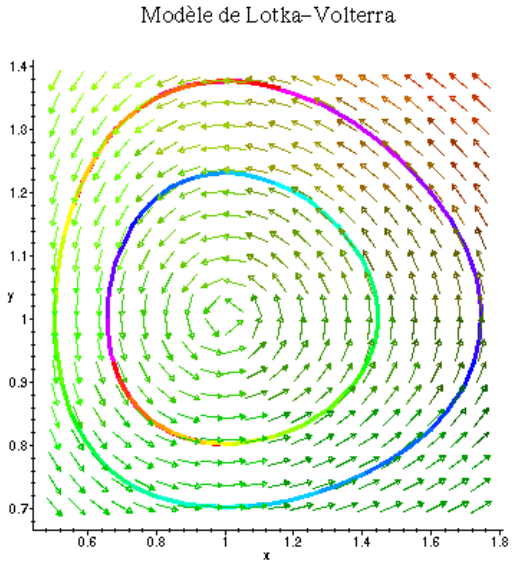
Images





Jérome Lesaint

Equa Diff



Mickael Nahon

3ÈME COURS :
INTRODUCTION AU FILTRAGE

Rappel : Histogramme et Histogramme cumulé

- Histogramme $\texttt{h(p)} =$ nombre de pixels ayant la valeur $\texttt{p}$
- Histogramme cumulé

$$\texttt{H(p)} = \text{nombre de pixels ayant une valeur } \leq p$$

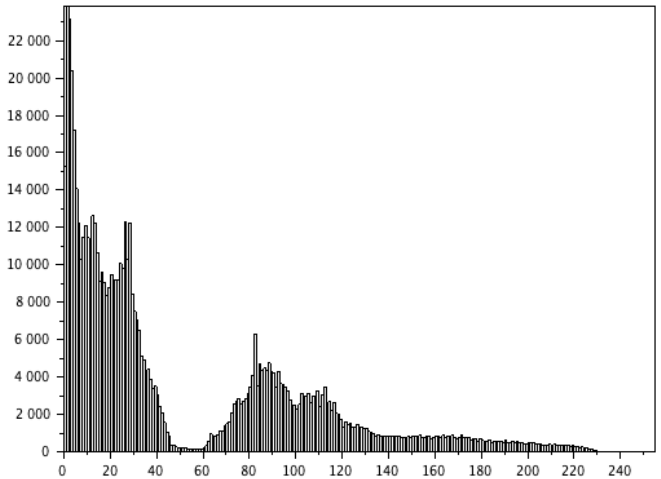
$$= \texttt{h(0)} + \ldots + \texttt{h(p)} = \sum_{\texttt{j=0}}^{\texttt{p}} \texttt{h(j)}$$

$$\Rightarrow \texttt{H(255)} = \text{nombre total de pixels}$$

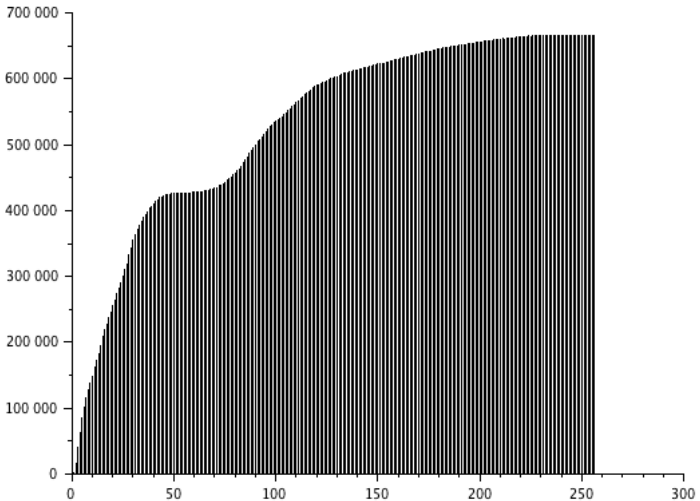
  - Définition récursive $\texttt{H(0)} = \texttt{h(0)}$ puis $\texttt{H(p+1)} = \texttt{H(p)} + \texttt{h(p+1)}$
  - Fonction Croissante $\texttt{H(p)} \geq \texttt{H(p-1)}$

Exemple

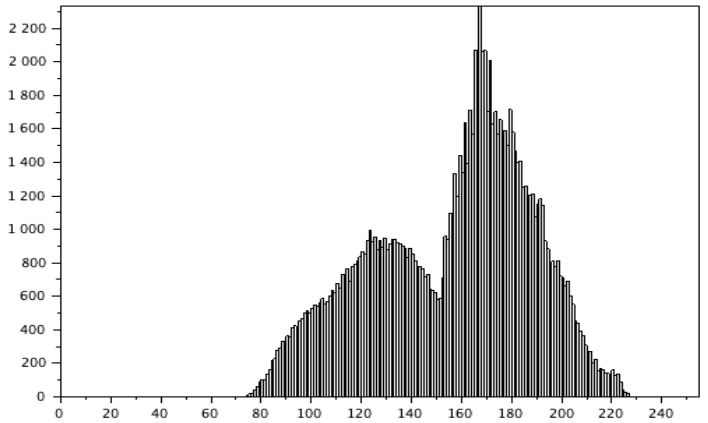
Histogramme

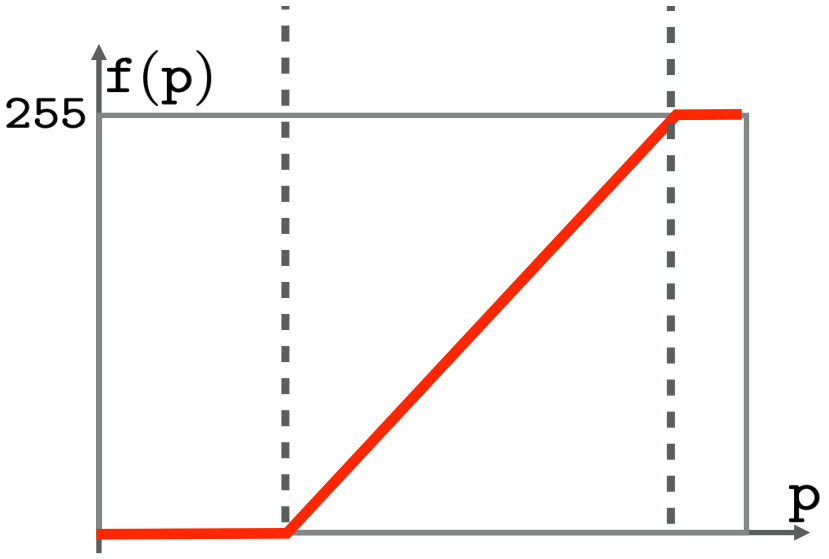


Histogramme cumulé



Correction de contraste et histogramme

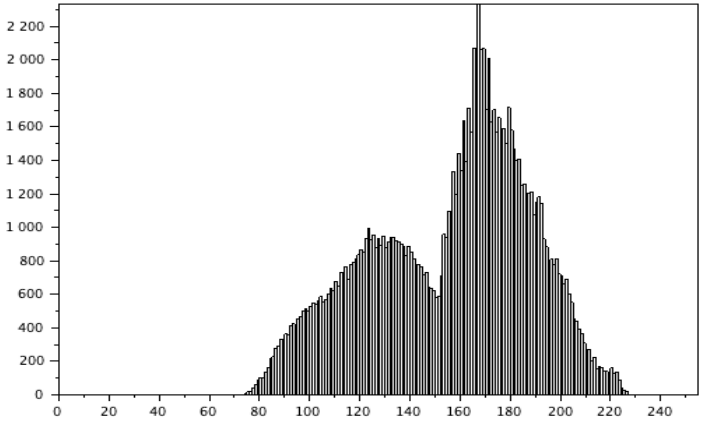


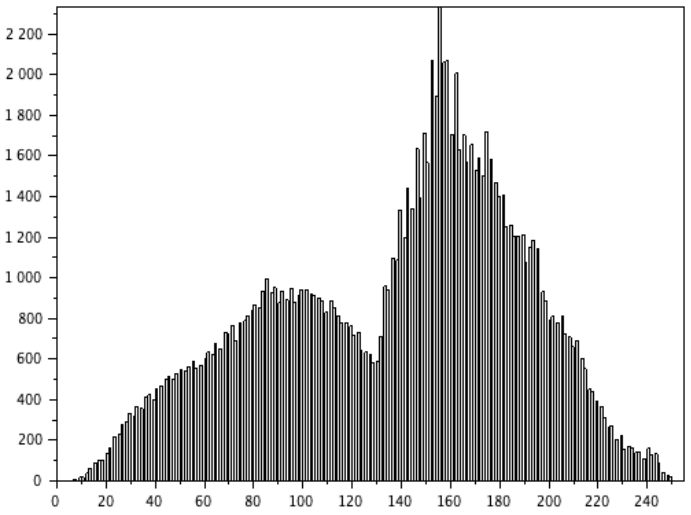


Choix des bornes :

- valeurs min et max dans h
- percentile (cf TP2)

Correction de contraste et histogramme



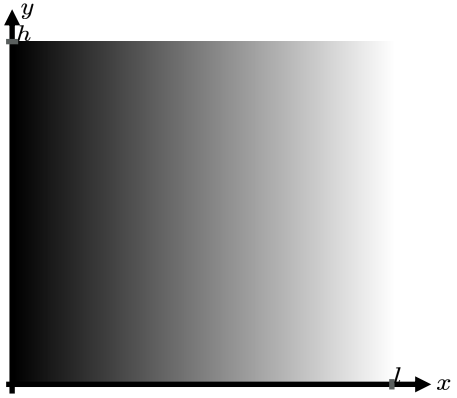


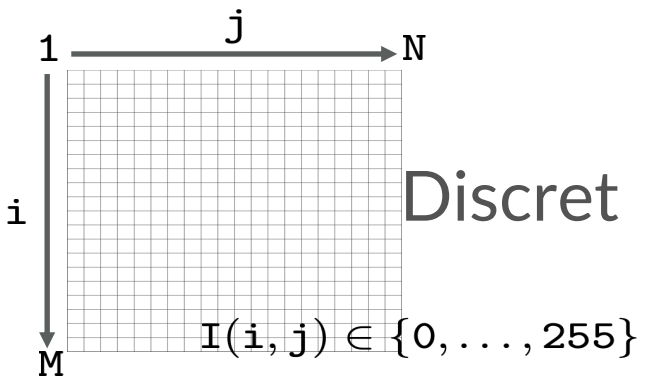
Transformations d'une image

- Ce qu'on a vu jusqu'à présent : pixel à pixel
  - Augmentation de luminosité contraste, égalisation d'histogramme
  - Chaque pixel est modifié indépendamment des autres en fonction de sa valeur (intensité lumineuse)
  - On ne regarde pas l'aspect spatial
- Ce qu'on va voir maintenant :
  - S'intéresser aux **variations spatiales**

Continu

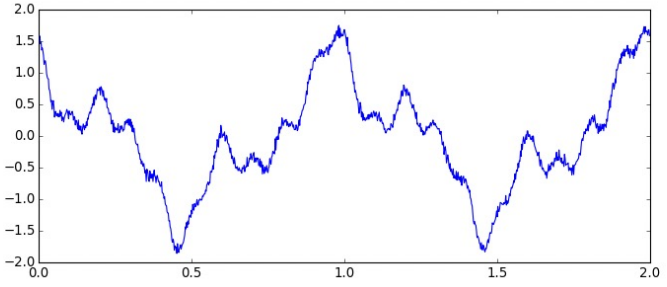
$$I : \begin{cases} \mathbb{R}^2 & \to & [0,1] \\ (x,y) & \mapsto & I(x,y) \end{cases}$$





Exemple de transformations “spatiales”

- Redimensionnement : interpolation
  - Bilinéaire, bicubique
- Lissage : élimination du bruit
  - **Filtrage** en traitement du signal



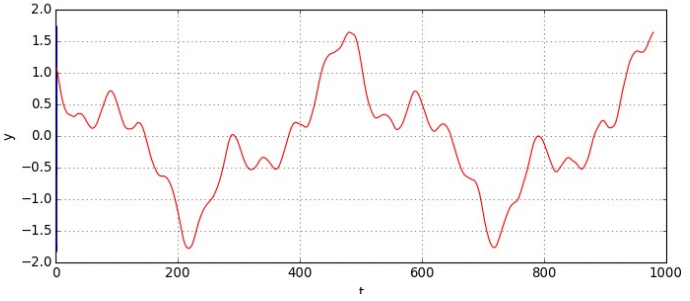
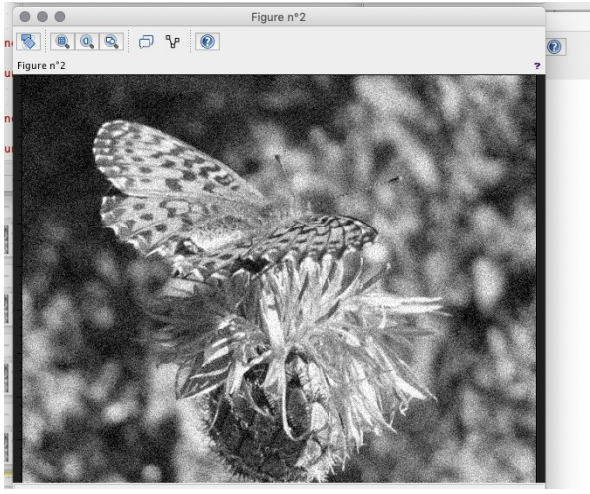
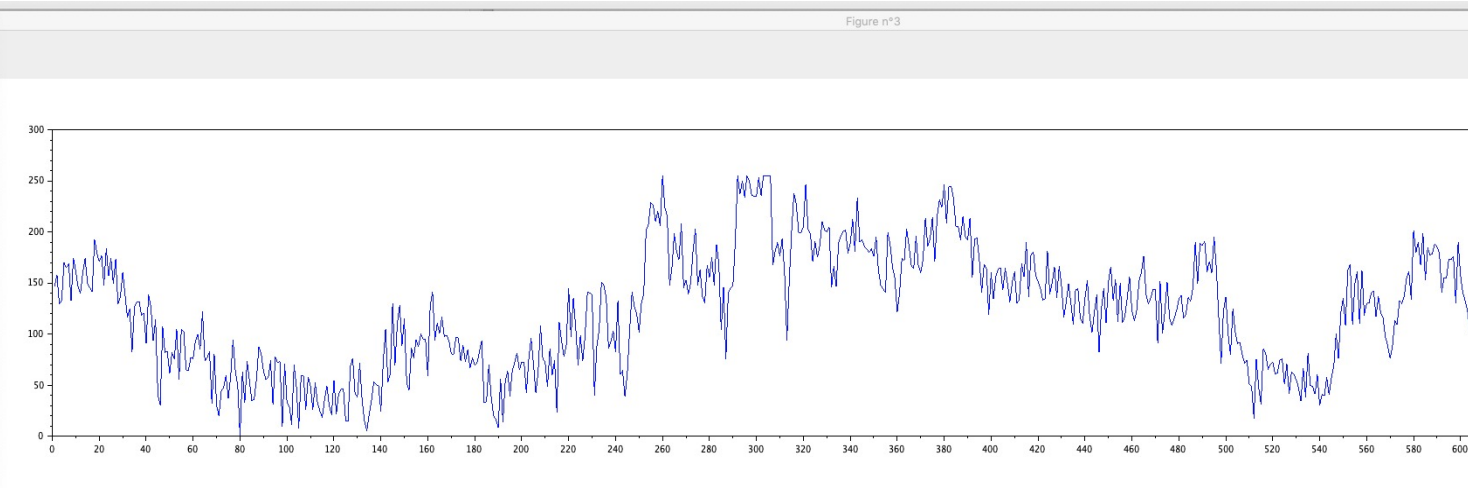
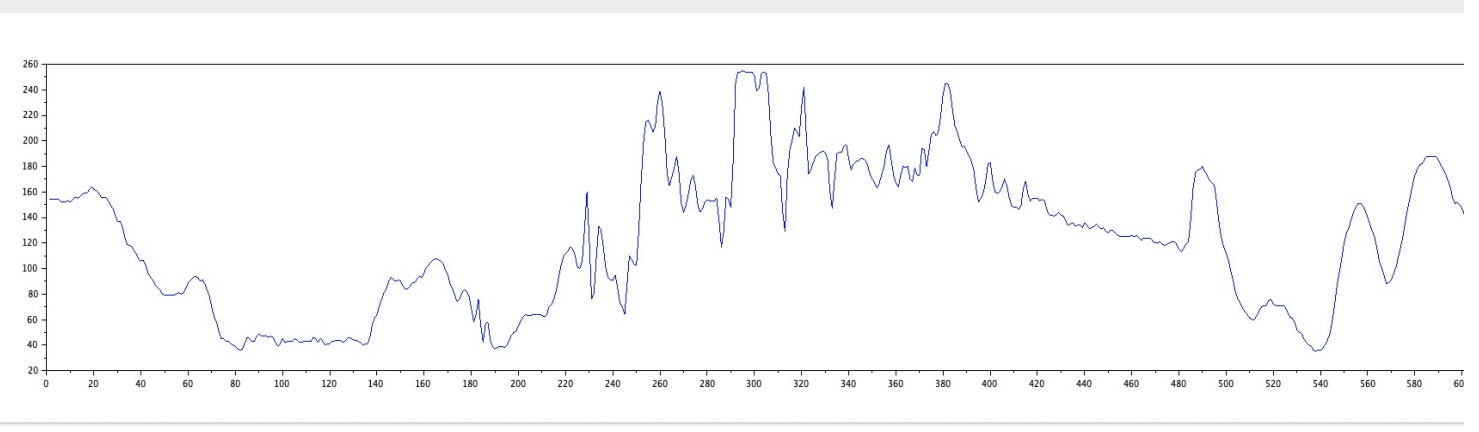


Image bruitée

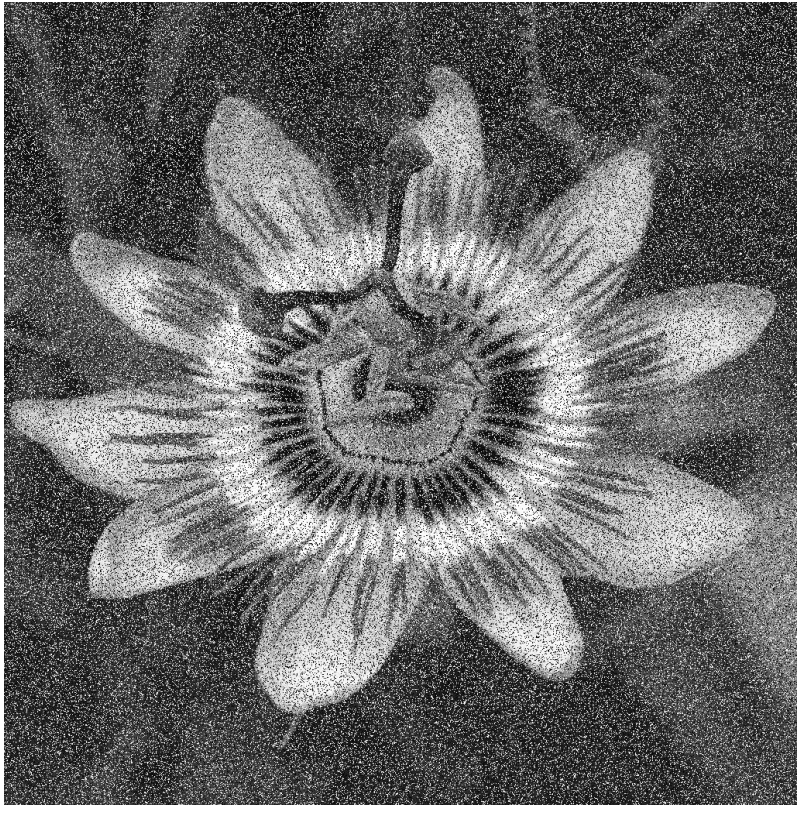






Exemple de lissage (ou débruitage)

- Lissage : élimination du “bruit”



Exemple de lissage (ou débruitage)

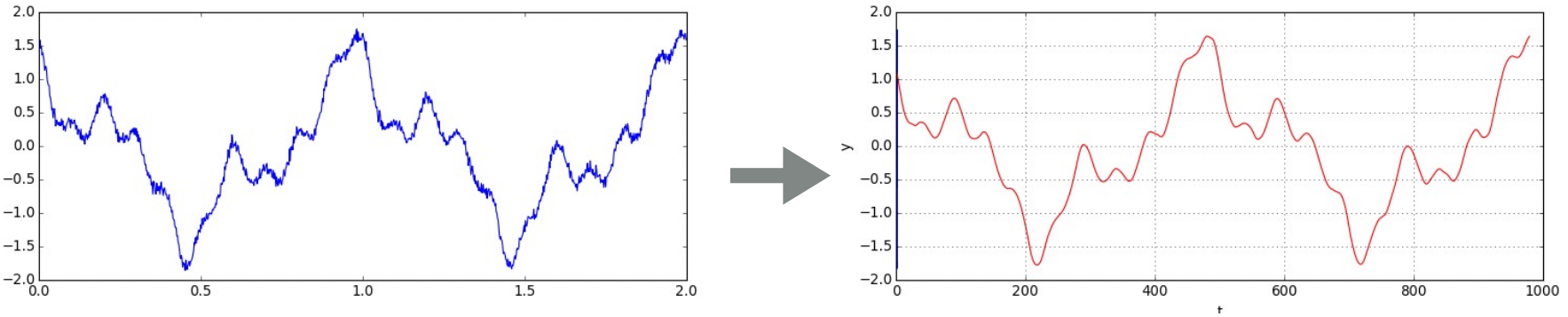
- Pourquoi ? Utile en segmentation d'images

Plan

- Cette semaine
  - Introduction au filtrage : cas 1D continu puis discret
  - Quelques filtres 2D pour éliminer le bruit
  - Produit de convolution
- La semaine prochaine
  - Fonctions de plusieurs variables : dérivabilité
  - Dérivées d'une image discrète
  - Identification des contours

Introduction au filtrage 1D

- Domaine continu **1D** : Comment réduire le bruit?



Fonction moyenne

- En 1D continu
  - Moyenne de f sur un segment $[a, b]$:

$$M([a,b]) = \frac{1}{b-a}\int_a^b f\,(t)dt$$

- En 1D discret
  - Moyenne de N valeurs consécutives $(\mathrm{f}_1, \ldots, \mathrm{f}_\mathrm{N})$

$$M(\{1,N\}) = \frac{1}{\mathrm{N}}\sum_{k=1}^{N} f_k$$

Moyenne locale

- Moyenne locale **continue** “autour” d’un point x
  - C’est la moyenne de f sur $[x-\epsilon, x+\epsilon]$

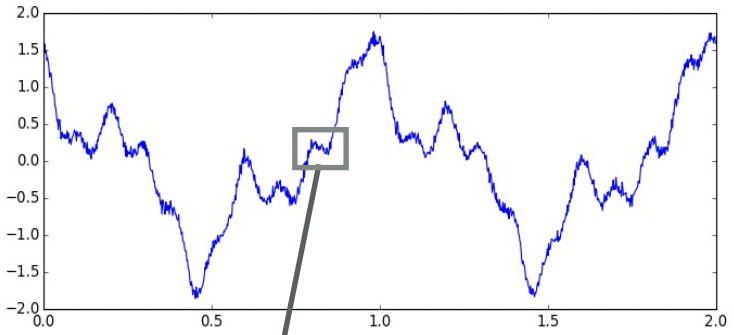
$$M_\varepsilon = \frac{1}{2\epsilon}\int_{x-\epsilon}^{x+\epsilon} f\,(t)dt$$

- Moyenne locale **discrète** « autour » d’un point x
  - C’est la moyenne des f_j sur une « fenêtre » de taille $2m+1$

$$\tilde{f}_i = \frac{1}{2m+1}\sum_{j=i-m}^{i+m} f_j$$

Reduction du bruit

Filtrage : on s'intéresse au comportement local sur une fenêtre que l'on déplace



Observation

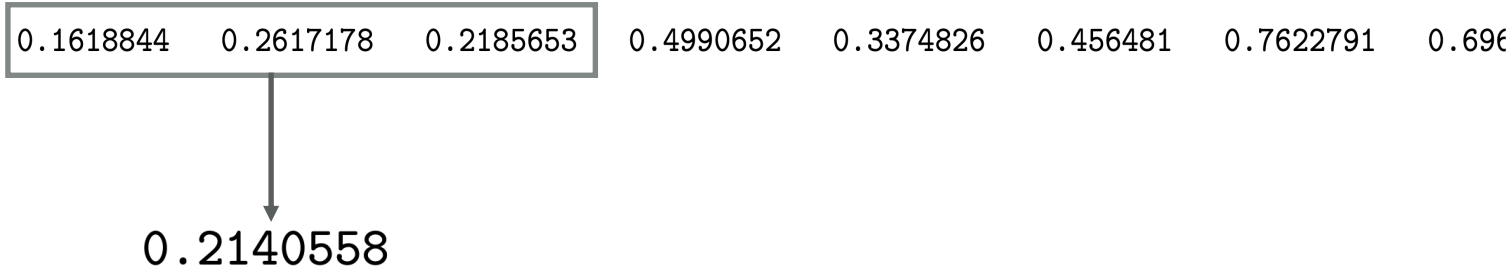
1 signal "basse fréquence"+du bruit "haute fréquence »

Idée : localement
Moyenne(signal) ~ signal
Moyenne(bruit) ~ 0
donc Moyenne(obs.) ~ signal

→ ***Appliquons une moyenne locale***

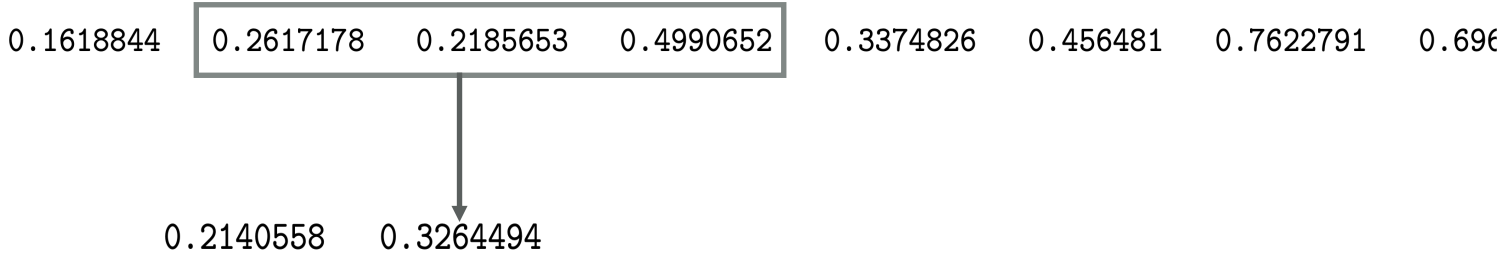
Moyenne locale : cas discret

- Idée : intégrale continue ↔️ somme discrète
  - Fonction discrète $f = [f_1, \ldots, f_N]$
  - Moyenne locale sur *2m+1 points* $\bar{f}_i = \frac{1}{2m+1} \sum_{j=i-m}^{i+m} f_j$
  - Exemple avec m=1



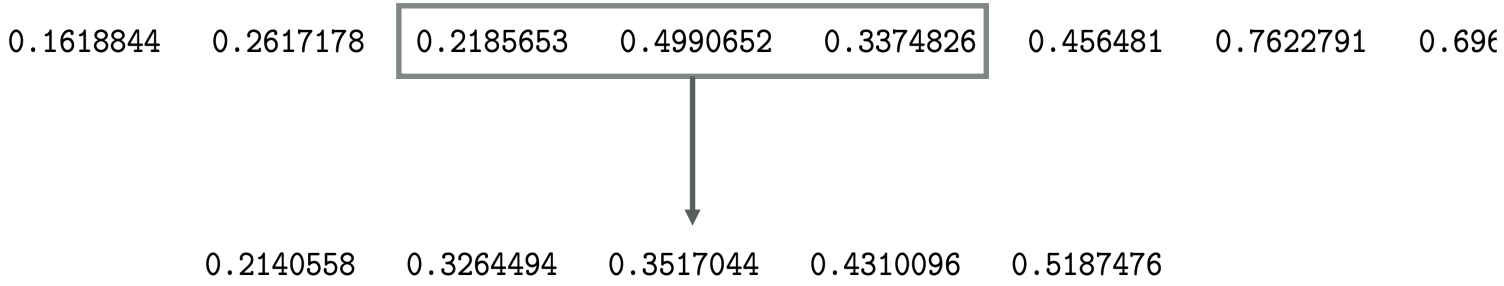
Moyenne locale : cas discret

- Idée : intégrale continue ↔️ somme discrète
  - Fonction discrète $f = [f_1, \ldots, f_N]$
  - Moyenne locale sur *2m+1 points* $\bar{f}_i = \frac{1}{2m+1} \sum_{j=i-m}^{i+m} f_j$
  - Exemple avec m=1



Moyenne locale : cas discret

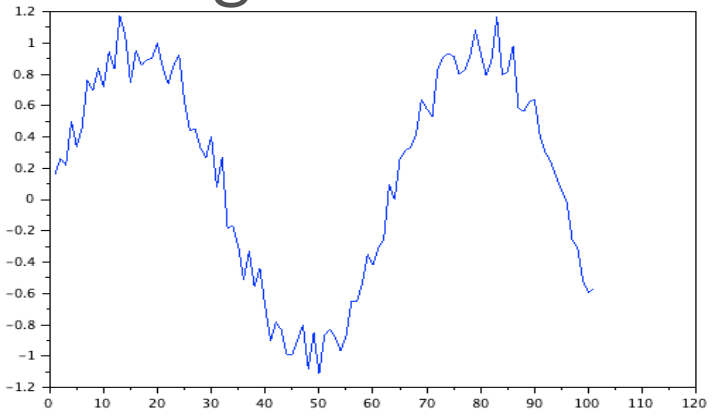
- Idée : intégrale continue ↔️ somme discrète
  - Fonction discrète $f = [f_1, \ldots, f_N]$
  - Moyenne locale sur *2m+1 points* $\bar{f}_i = \frac{1}{2m+1} \sum_{j=i-m}^{i+m} f_j$
  - Exemple avec m=1



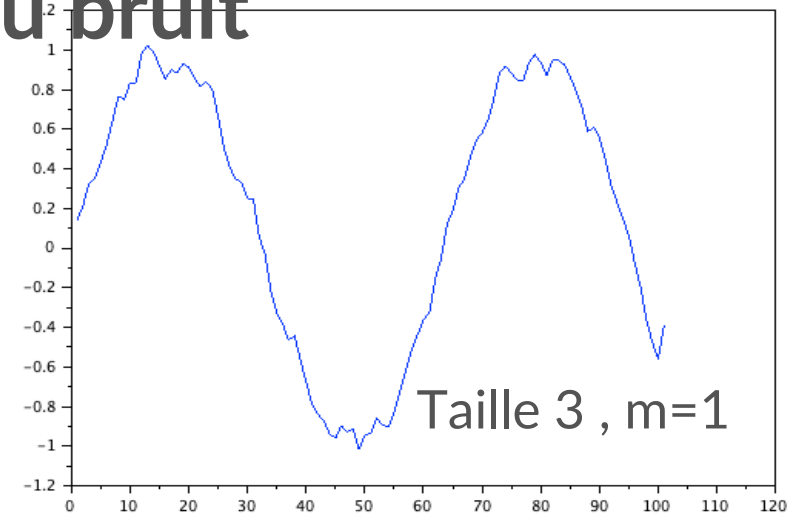
Nouveau signal filtré : → on a perdu de l'info et les valeurs au bord
→ mais le nouveau signal est "meilleur"

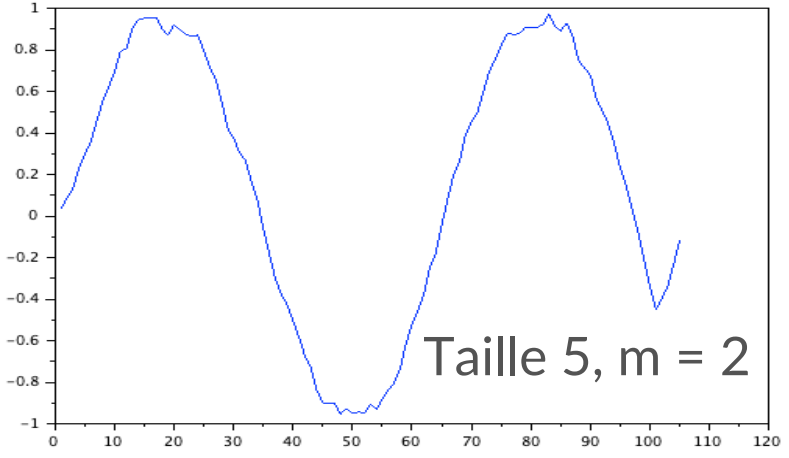
Reduction du bruit

signal bruité



Filtre **moyenne**



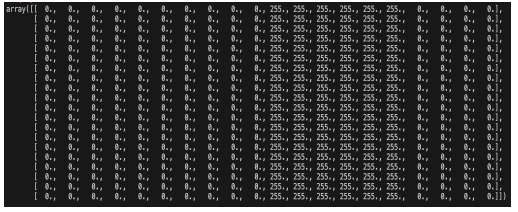


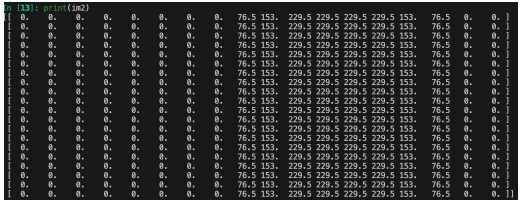
Effet de la moyenne sur une image

- Moyenne 1D sur les lignes (m=1)

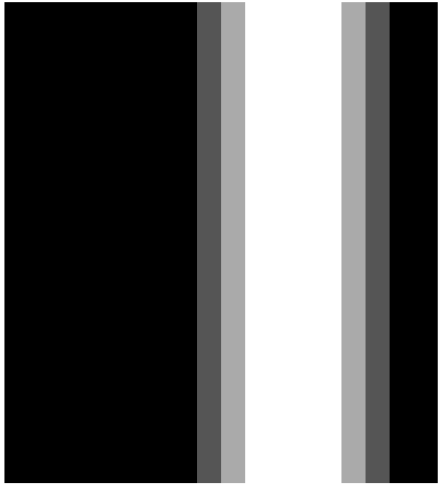
p_0	p_1	p_2

$$p'_1 = \frac{1}{3}(p_0 + p_1 + p_2)$$





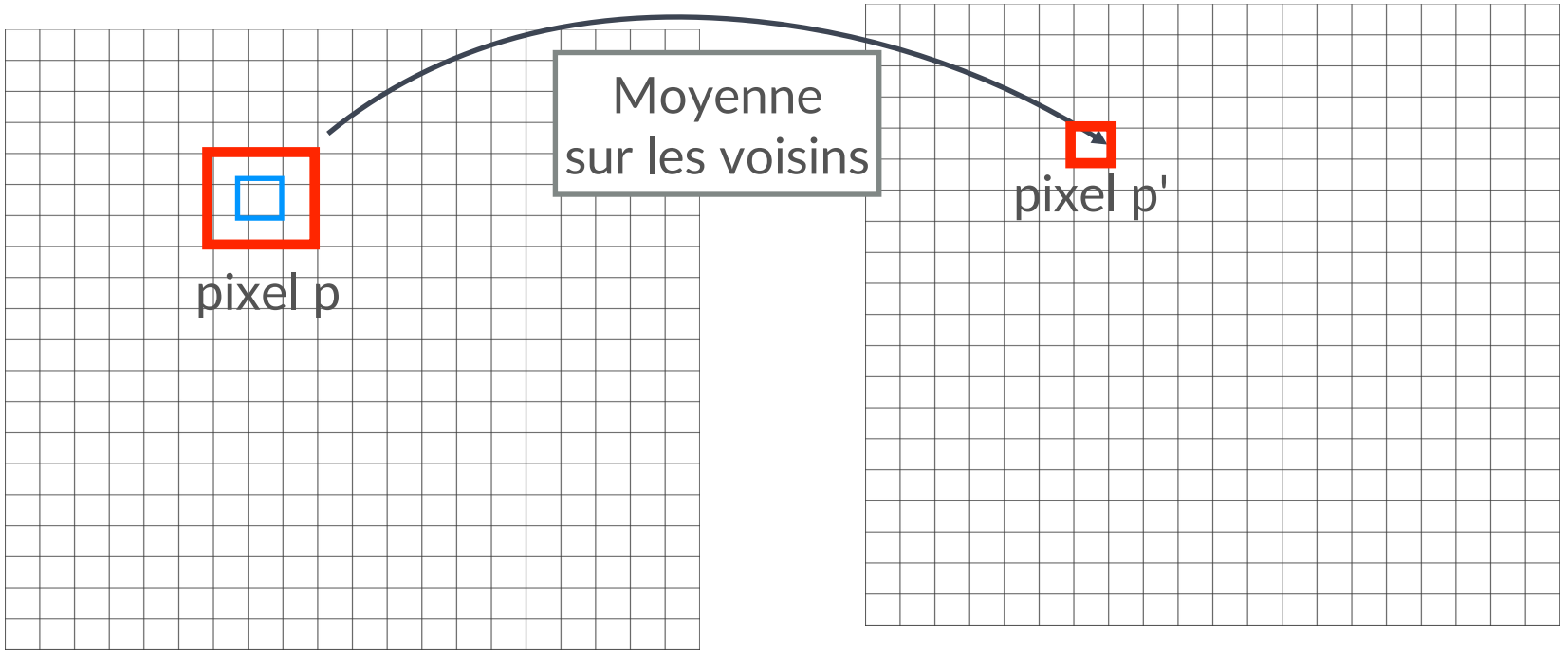
Variation brutale



- Les pixels de bord ne sont pas traités
- Variations plus douces

Moyenne 2D

- Filtre moyenne
  - La nouvelle valeur est la moyenne sur un voisinage (ou fenêtre)



En pratique

- Voisinage de taille (2m+1)x(2m+1) autour d'un pixel (u,v) :

```
In [19]: voisinage = im[u-m:u+m+1,v-m:v+m+1]
```

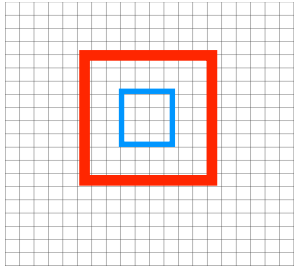
- Nouvelle valeur = moyenne du voisinage :

```
In [22]: new_val = im[u-m:u+m+1,v-m:v+m+1].mean()
```

- On applique à tous les pixels sauf ceux du bord

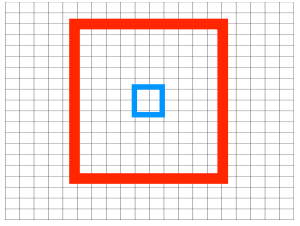
```
1 for u in range(m+1,M-m):
2     for v in range(m+1,N-m):
3         im2[u,v] = im[u-m:u+m+1,v-m:v+m+1].mean()
```

Exemple d’application du filtre moyenne



Voisinage immédiat

Exemple d’application du filtre moyenne



Voisinage plus grand

→ effet de flou

Effet sur le bruit

- Le bruit nuit à la qualité de l'image
  - Dû aux fluctuations de la lumière, capteur, échantillonnage

- Lissage par filtre moyenne
  - Éliminer le bruit sans (trop) altérer l'image

Effet sur le bruit

Moyenne
3x3

Effet sur le bruit

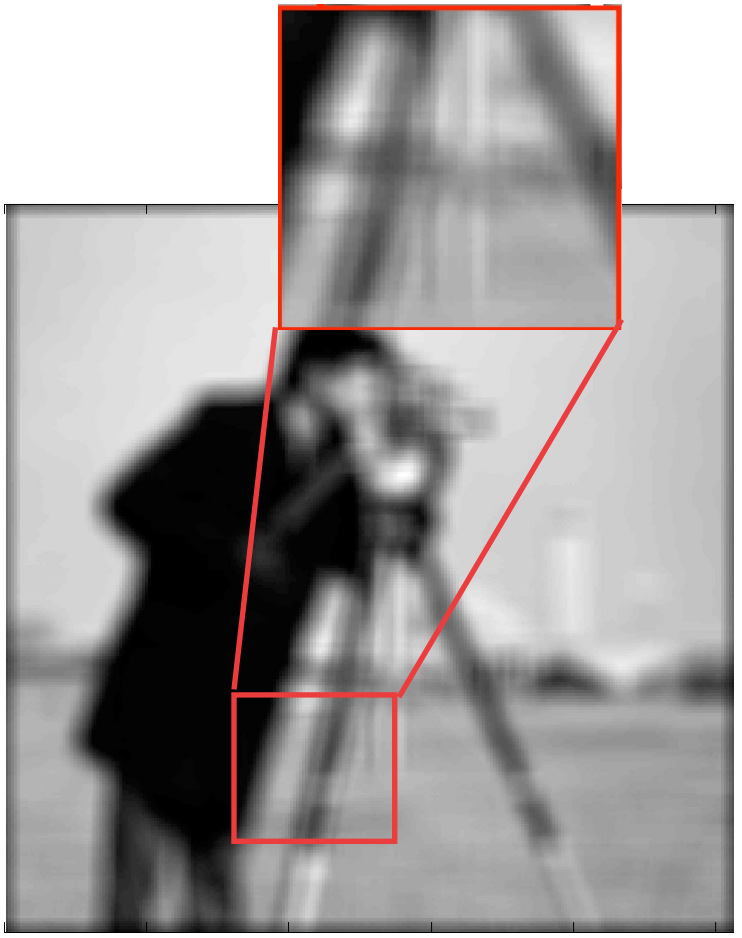
Moyenne
5x5

Effet sur le bruit

Moyenne
11x11

Inconvénients de la moyenne arithmétique

- Directions x et y privilégiées
- Introduit des artefacts



Moyenne pondérée

- Données $X = x_1, \dots, x_n$
- Poids $W = w_1, \dots, w_n > 0$

$$\bar{X}_W = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

  - La moyenne est **pondérée** par des **poids**
  - Comme un **barycentre**, les notes du bac, etc.

w = [9, 7, 5, 3, 3, 3]
notes = [...]

produit terme à terme

```
In [36]: moyenne = (notes * w).sum() / w.sum()
```

Quels poids choisir ?

- Intuitivement :
  - Plus de poids sur le pixel central
  - Symétrique
- Exemple en 1D (m=1)
  -

```
In [35]: w = [0.5, 1, 0.5]
```

$$\bar{X}_{W,i} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = \frac{0.5x_{i-1}+1x_i+0.5x_{i+1}}{0.5+1+0.5} = \frac{1}{2}(0.5x_{i-1} + 1x_i + 0.5x_{i+1})$$

Moyenne pondérée en 2D

- Poids (et fenêtre)

$$\frac{1}{40}\begin{pmatrix} 3 & 5 & 3 \\ 5 & 8 & 5 \\ 3 & 5 & 3 \end{pmatrix}$$

```
In [48]: W = 1/40 * np.array([[3, 5, 3],[5, 8, 5],[3, 5, 3]])

In [49]: W
Out[49]:
array([[0.075, 0.125, 0.075],
       [0.125, 0.2  , 0.125],
       [0.075, 0.125, 0.075]])
```

```
In [47]: im2[u,v] = (im1[u-m:u+m+1,v-m:v+m+1] * W).sum()
```

Moyenne pondérée en 2D

- Poids (et fenêtre)

$$\frac{1}{40}\begin{pmatrix} 3 & 5 & 3 \\ 5 & 8 & 5 \\ 3 & 5 & 3 \end{pmatrix}$$

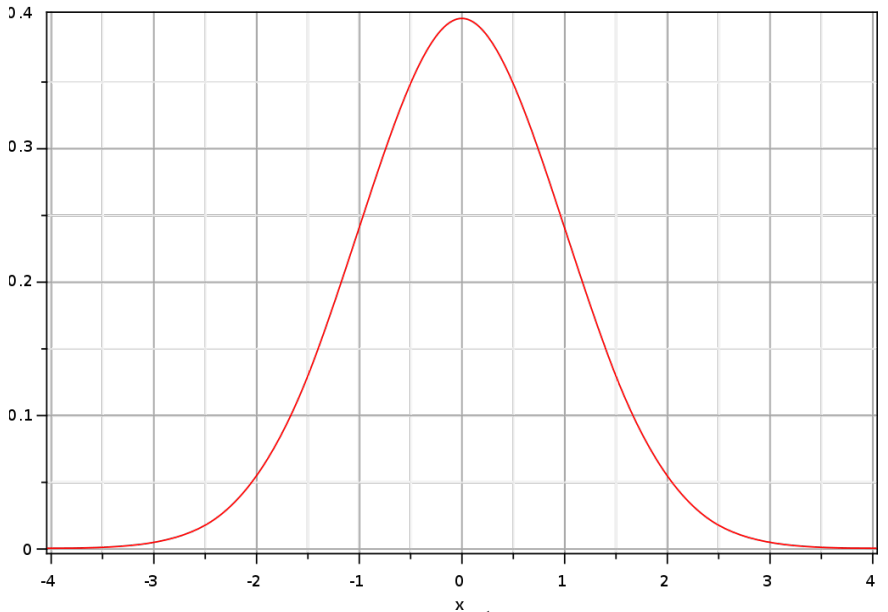
```
In [48]: W = 1/40 * np.array([[3, 5, 3],[5, 8, 5],[3, 5, 3]])

In [49]: W
Out[49]:
array([[0.075, 0.125, 0.075],
       [0.125, 0.2  , 0.125],
       [0.075, 0.125, 0.075]])
```

```
In [47]: im2[u,v] = (im1[u-m:u+m+1,v-m:v+m+1] * W).sum()
```

Filtre Gaussien en 1D

- Moyenne pondérée particulière
- Permet de générer des poids pour des tailles de fenêtre quelconque
  - Symétrique autour du point central
  - Poids sur le point central
  - Décroissance “lisse”
- Fonction Gaussienne 1D



$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

σ controle **l’étalement (la taille de la fenêtre)**

Filtre Gaussien en 1D

- Fonction Gaussienne 1D

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

Fixons : $\sigma = 1$

- Pour m = 1

$$w_1 = g(1) = e^{-1/2} = 0.607 = w_{-1}$$

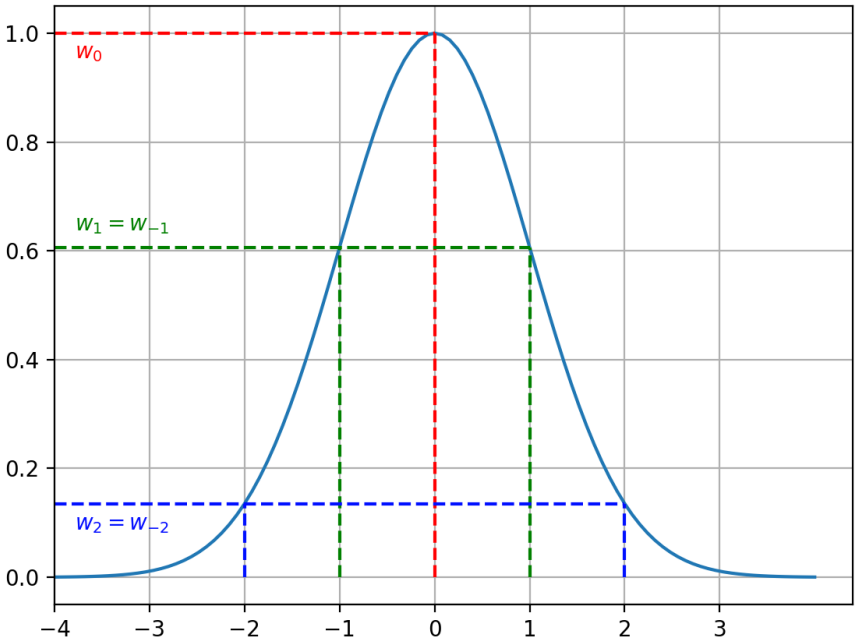
$$w_0 = g(0) = 1$$

- Pour m = 2

$$w_2 = g(2) = e^{-2} = 0.136 = w_{-2}$$

$$w_1 = g(1) = e^{-1/2} = 0.607 = w_{-1}$$

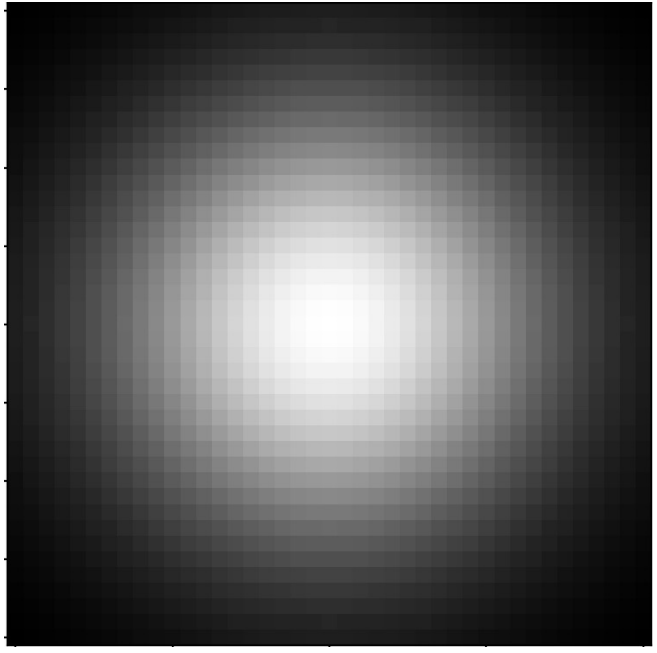
$$w_0 = g(0) = 1$$



Filtre Gaussien en 2D

- Moyenne pondérée particulière
- Permet de générer des poids pour des tailles de fenêtre quelconque
  - Symétrie radiale (idem dans toutes les directions)
  - Poids sur le point central
  - Décroissance “lisse”
- Fonction Gaussienne 2D

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



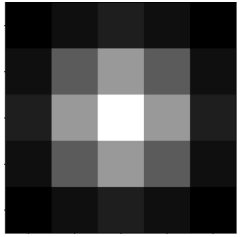
Filtre Gaussien 2D

- m = 1

```
0.36787944, 0.60653066, 0.36787944
0.60653066, 1.        , 0.60653066
0.36787944, 0.60653066, 0.36787944
```

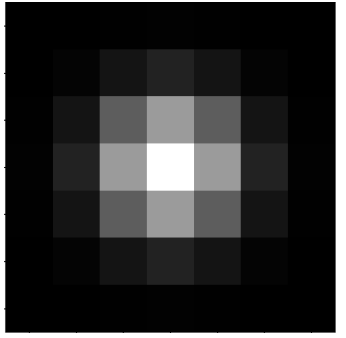
- m = 2

```
0.01831564, 0.082085  , 0.13533528, 0.082085  , 0.01831564
0.082085  , 0.36787944, 0.60653066, 0.36787944, 0.082085
0.13533528, 0.60653066, 1.        , 0.60653066, 0.13533528
0.082085  , 0.36787944, 0.60653066, 0.36787944, 0.082085
0.01831564, 0.082085  , 0.13533528, 0.082085  , 0.01831564
```



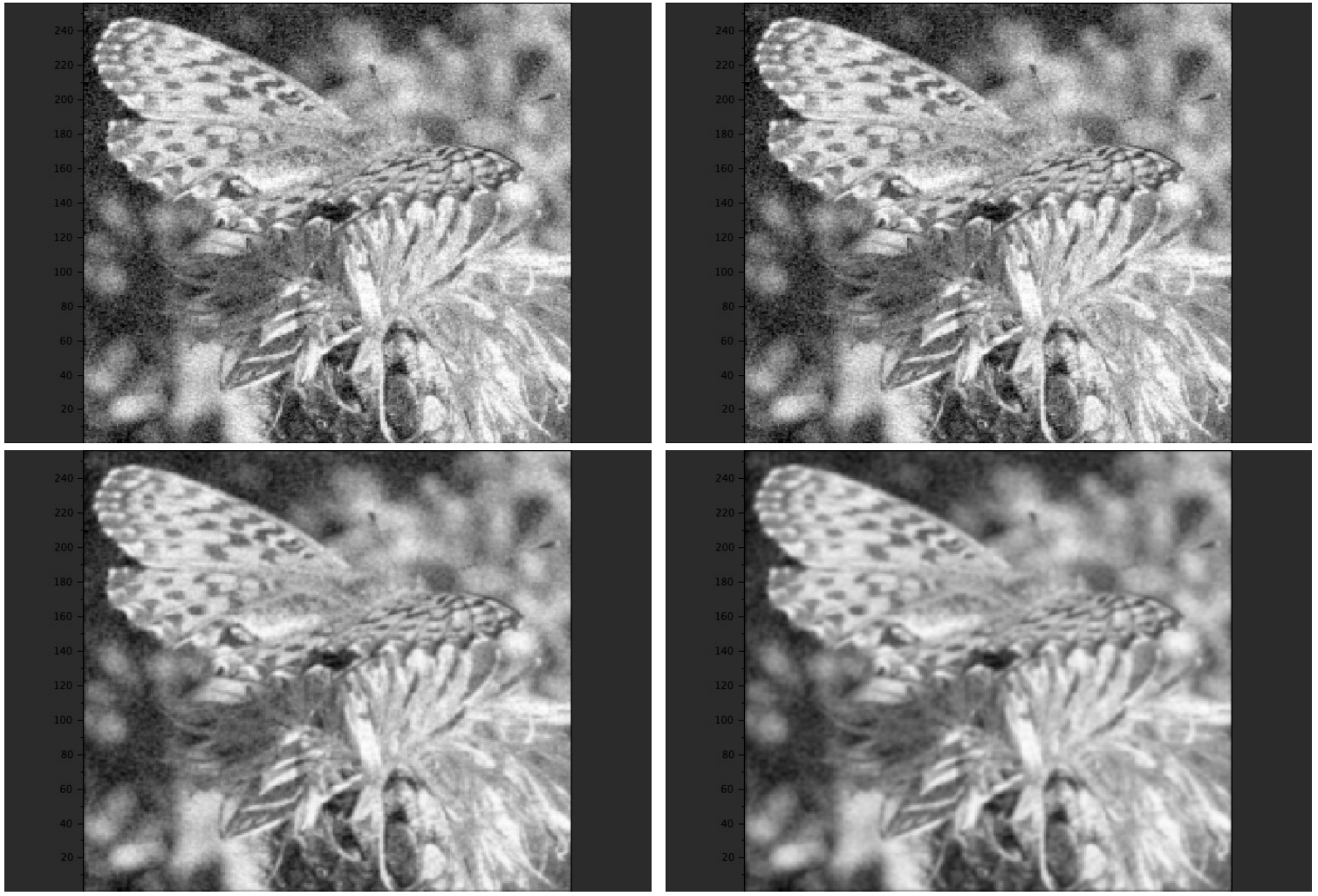
- m = 3

```
0.00012341, 0.00150344, 0.00673795, 0.011109  , 0.00673795, 0.00150344, 0.00012341
0.00150344, 0.01831564, 0.082085  , 0.13533528, 0.082085  , 0.01831564, 0.00150344
0.00673795, 0.082085  , 0.36787944, 0.60653066, 0.36787944, 0.082085  , 0.00673795
0.011109  , 0.13533528, 0.60653066, 1.        , 0.60653066, 0.13533528, 0.011109
0.00673795, 0.082085  , 0.36787944, 0.60653066, 0.36787944, 0.082085  , 0.00673795
0.00150344, 0.01831564, 0.082085  , 0.13533528, 0.082085  , 0.01831564, 0.00150344
0.00012341, 0.00150344, 0.00673795, 0.011109  , 0.00673795, 0.00150344, 0.00012341
```



Ne pas oublier de normaliser i.e. diviser par la somme des poids

Effet du filtre Gaussien



Filtrage et Convolution en 1D

- Produit de convolution entre deux fonctions

$$f * g(x) = \int_{-\infty}^{+\infty} f(t)g(x-t)dt$$

$$f * g(x) = g * f(x)$$

- Version discrete

$$f * g(n) = \sum_{-\infty}^{+\infty} f(m)g(n-m)$$

$$f * g(n) = g * f(n)$$

Convolution discrète 2D

- Opération de base du ***filtrage***

```
In [47]: im2[u,v] = (im1[u-m:u+m+1,v-m:v+m+1] * W).sum()
```

- Peut se généraliser (coefficients pas tous positifs, somme non unitaire…)
- Filtrage 2D à partir d'un filtre W en Python

```
In [57]: sp.signal.convolve2d(im,W,'same')
```

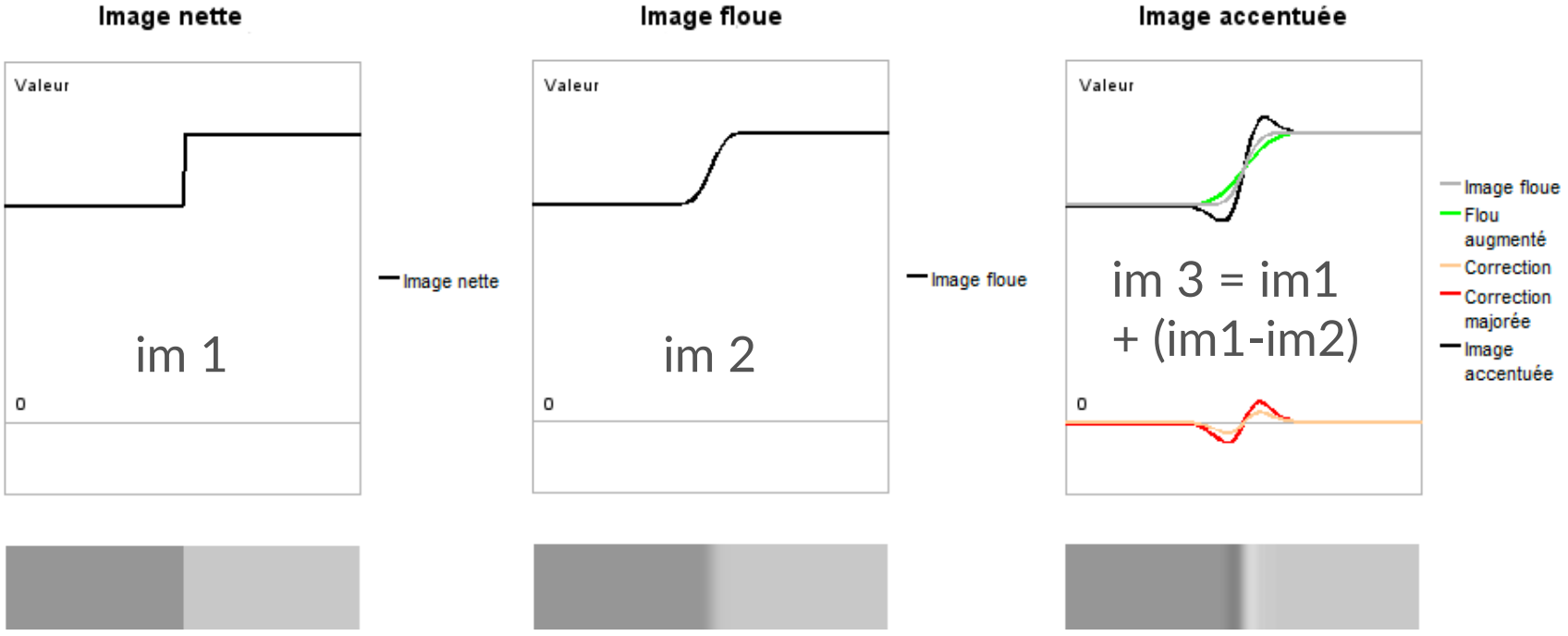
Le “masque flou” (unsharp masking)

im1 : original

im2 : filtre gaussien $\sigma = 1.5$

Que contient im1 – im2 ?

Le “masque flou” (unsharp masking)

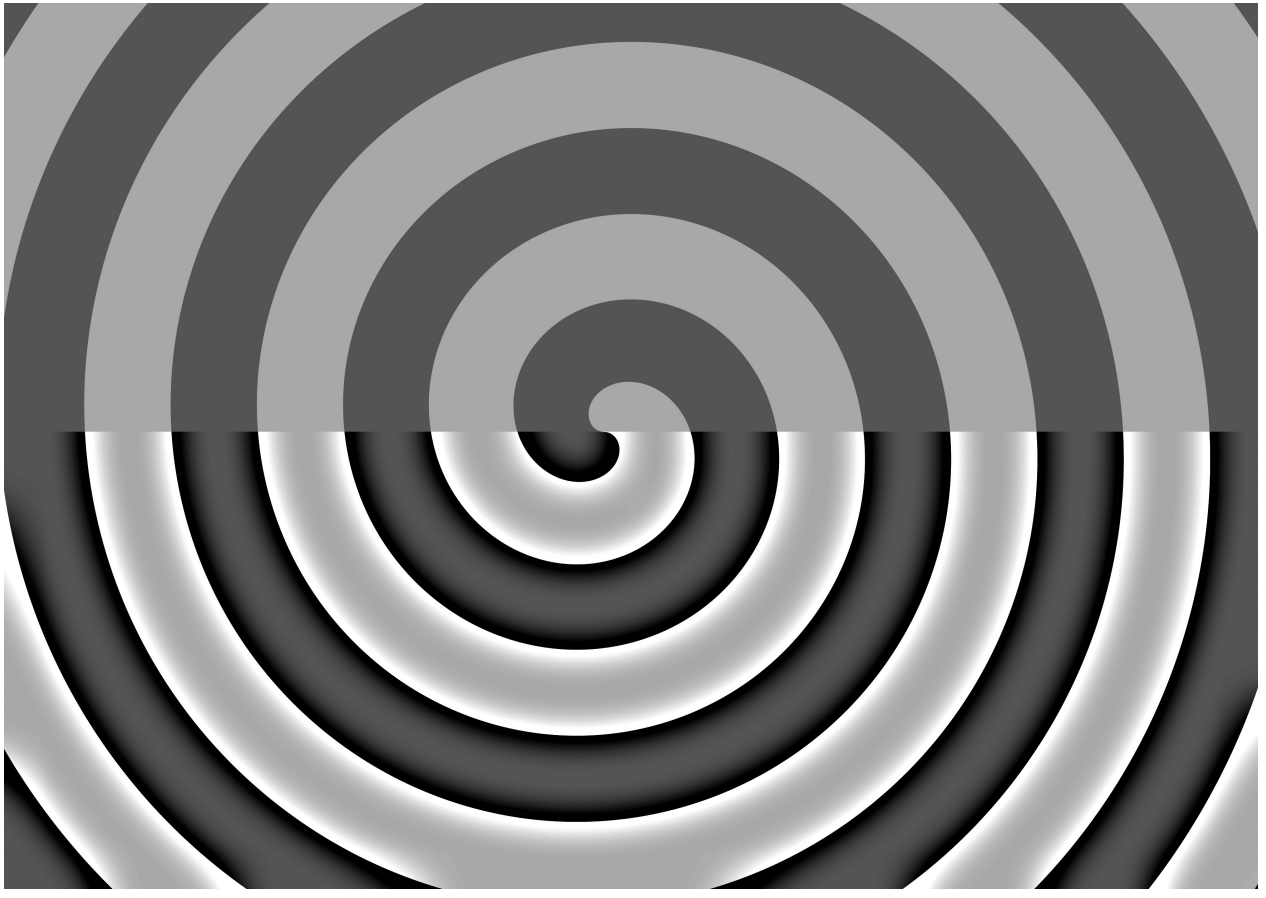


→ Augmentation de la netteté

→ Augmentation des différences **locales**

≠ correction du contraste !

Le “masque flou” (unsharp masking)



En termes de filtres

$$\mathtt{im3} = \mathtt{im1} + (\mathtt{im1} - \mathtt{im2})$$

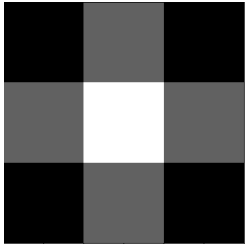
$$\mathtt{im3} = 2 * \mathtt{im1} - \mathtt{conv}(\mathtt{im1}, \mathtt{G})$$

$$\forall(\mathtt{u}, \mathtt{v}), \mathtt{im3}(\mathtt{u}, \mathtt{v}) = 2\mathtt{im1}(\mathtt{u}, \mathtt{v}) - \mathtt{sum}(\mathtt{im1}(\mathtt{u} - \mathtt{m} : \mathtt{u} + \mathtt{m}, \mathtt{v} - \mathtt{m} : \mathtt{v} + \mathtt{m}). * \mathtt{G})$$

Filtre gaussien

σ = 0.5, m=1

```
0.0113437   0.0838195   0.0113437
0.0838195   0.619347    0.0838195
0.0113437   0.0838195   0.0113437
```



Filtre équivalent pour obtenir im3 en une opération ?
im3 = conv(im1, G2)

```
-0.0113437  -0.0838195  -0.0113437
-0.0838195   1.380653   -0.0838195
-0.0113437  -0.0838195  -0.0113437
```

En termes de filtres

$$\mathtt{im3} = \mathtt{im1} + (\mathtt{im1} - \mathtt{im2})$$

$$\mathtt{im3} = 2 * \mathtt{im1} - \mathtt{conv}(\mathtt{im1}, \mathtt{G})$$

$$\forall(\mathtt{u}, \mathtt{v}), \mathtt{im3}(\mathtt{u}, \mathtt{v}) = 2\mathtt{im1}(\mathtt{u}, \mathtt{v}) - \mathtt{sum}(\mathtt{im1}(\mathtt{u} - \mathtt{m} : \mathtt{u} + \mathtt{m}, \mathtt{v} - \mathtt{m} : \mathtt{v} + \mathtt{m}). * \mathtt{G})$$

Filtre gaussien σ = 0.5

$$\mathtt{G} = \begin{matrix} 0.0113437 & 0.0838195 & 0.0113437 \\ 0.0838195 & 0.619347 & 0.0838195 \\ 0.0113437 & 0.0838195 & 0.0113437 \end{matrix}$$

Filtre équivalent pour obtenir im3 en une opération ?
im3 = conv(im1, G2)

$$\begin{matrix} -0.0113437 & -0.0838195 & -0.0113437 \\ -0.0838195 & 1.380653 & -0.0838195 \\ -0.0113437 & -0.0838195 & -0.0113437 \end{matrix}$$

Exercices récapitulatifs

Ex 1

Construisez un filtre 1D de taille 5 ($m = 2$) à partir de la fonction $f(x) = \frac{1}{1+|x|}$.

Construisez un filtre 2D de taille 5×5 ($m = 2$) à partir de la fonction $h(x, y) = \frac{1}{1+\sqrt{x^2+y^2}}$.

Ex 2

1) Nous considérons les quatre filtres 2D ci-dessous. Quel(s) filtre(s) correspond(ent) à un moyennage local ? Votre réponse devra être justifiée.

Filtres :

1:
$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

2:
$$\begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

3:
$$\begin{matrix} 0 & 0.5 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0.5 & 0 \end{matrix}$$

4:
$$\begin{matrix} 0.075 & 0.1 & 0.075 \\ 0.1 & 0.3 & 0.1 \\ 0.075 & 0.1 & 0.075 \end{matrix}$$