Introduction au traitement d'images

Enseignement intégré
IT220 | Informatique 2A | 2023-2024

Chapitre 4 : Traitements

Rémi Giraud
remi.giraud@enseirb-matmeca.fr
https://remi-giraud.enseirb-matmeca.fr/

Plan du cours

- **Introduction**
- **Formation / Acquisition**
- **Image numérique**
  - Format/Affichage/Synthèse
  - Espaces couleur caractéristiques : compression, esquisse, illusion
- **Traitements**
  - Filtrage linéaire / non linéaire : débruitage, anonymisation
  - Détection de contours : réhaussement de contraste
- **Transformée de Fourier**
  - Application : recouvrement fréquentiel
- **Compression d'images**
  - Application : algorithme JPEG
- **Transformation spatiales**

Contenu

Traitement ponctuel

- Changement de contraste, égalisation d'histogramme

Traitement sur voisinage

- Filtrage, convolution 2D, débruitage

Détection de contours

Notations

opérateur (de traitement)

$$I_e \xrightarrow{T} I_s = T(I_e)$$

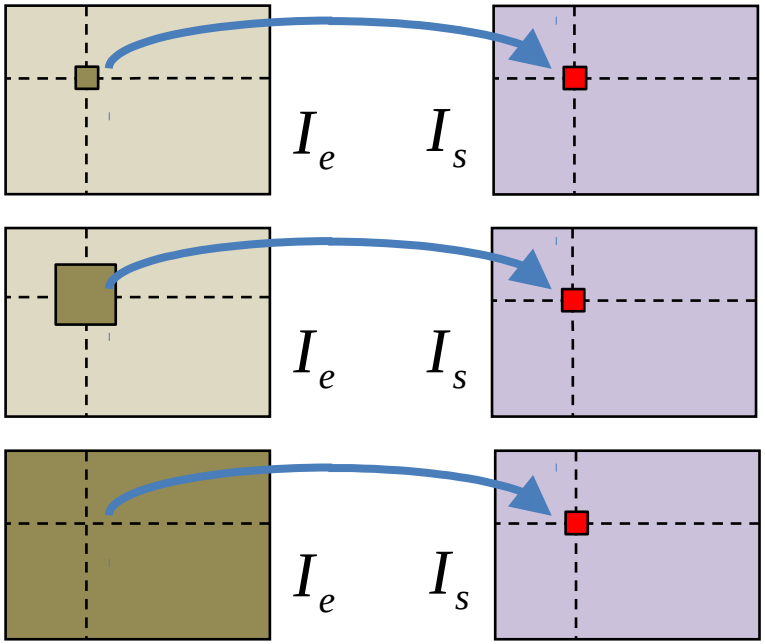
image « d'entrée »

image « de sortie »

Pixel à la position (m,n)

$$I_e(m,n)$$

Voisinage d'un opérateur



opérateur point à point (changement de dynamique, combinaisons arithmétiques ou logiques, ...)

opérateur local (filtrage, opérateurs morphologiques, détection de contours, transformations géométriques, ...)

opérateur global (transformée de Fourier, en ondelettes, ...)

Problème d'exposition

Problème d'exposition

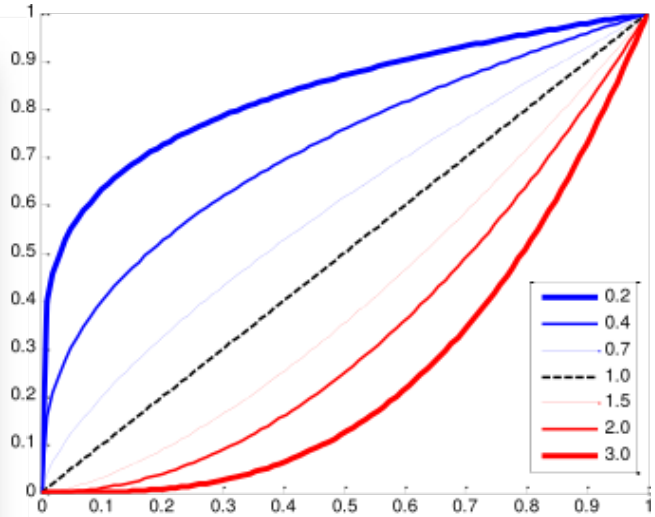
- Transposition d'intensité

$$I_e(m,n) \xrightarrow{T} I_s(m,n) = I_e(m,n) + \delta$$

Loi Gamma

- Le meilleur rendu de l'image ne correspond pas forcément à une représentation linéaire de la luminosité.

$$I_s(m,n) = I_e(m,n)^\gamma$$



Comment extraire les informations d'une image ?

- Bilan des caractéristiques de l'image
  - Dimensions spatiales
  - Codage : « vraies couleurs » couleurs indexées
  - Format numérique
  - Intervalles d'intensité
  - Répartition des intensités

```
h, w, c = img.shape
```

```
whos (dans la console)
```

```
[np.min(I(:)) np.max(I(:))]
```

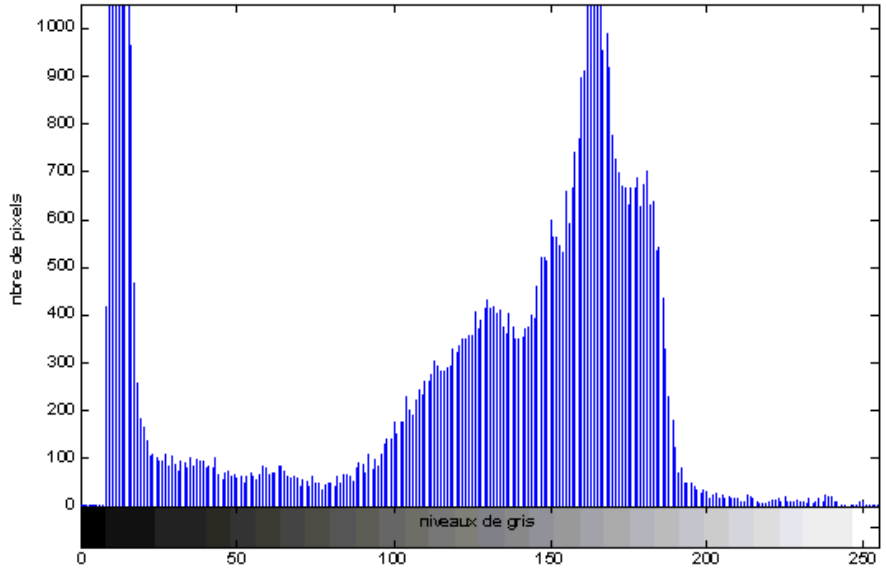
```
np.histogram(I)
```

- Identification de la fonction d'affichage
  - Déterminer le contexte
    - Informationnel
    - De fidélité
    - De comparaison
  - Choix : palette, ratio L/H, intervalles d'intensité, etc.

Définition

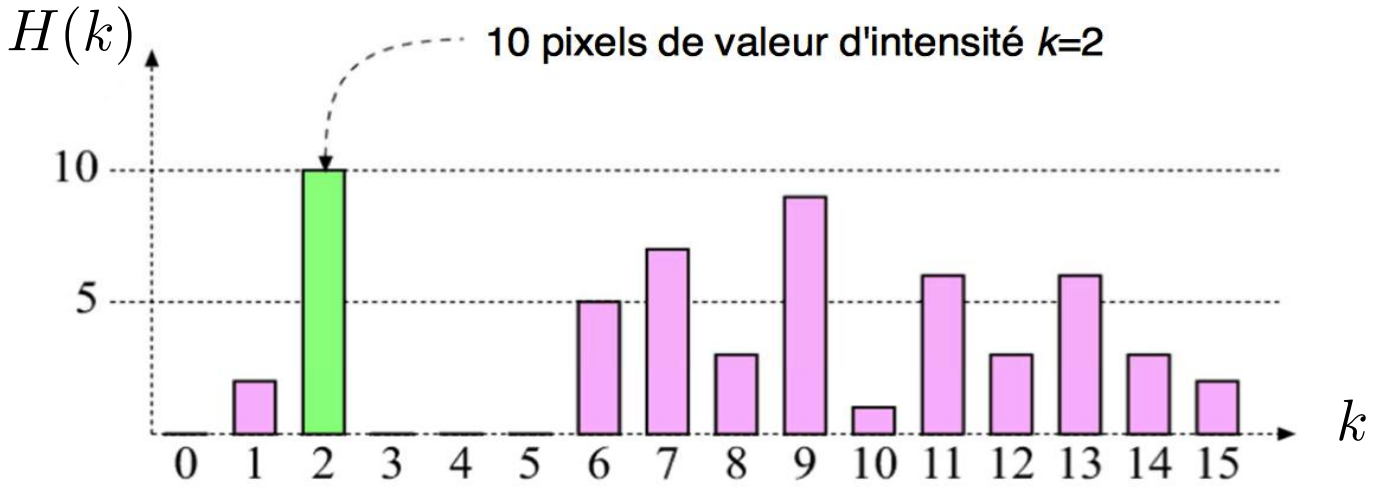
Graphique permettant de représenter la distribution des intensités des pixels d'une image (nombre de pixels pour chaque intensité)

Informations : Distribution statistique, bornes de la répartition, ...



Définition

Histogramme $H(k) = \#\{(i,j) \in M \times N : I(i,j) = k\}$

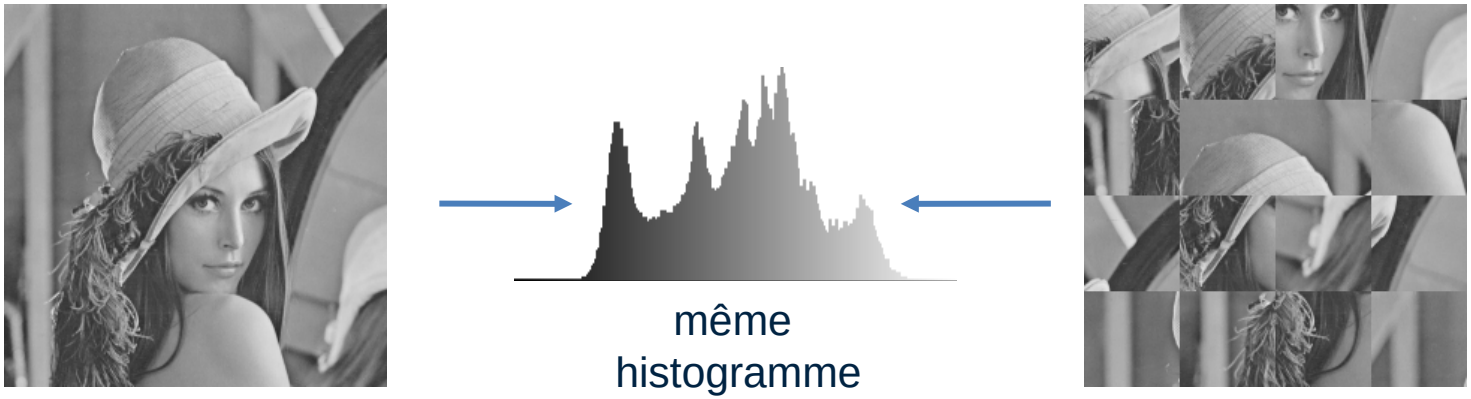


k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$H(k)$	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2

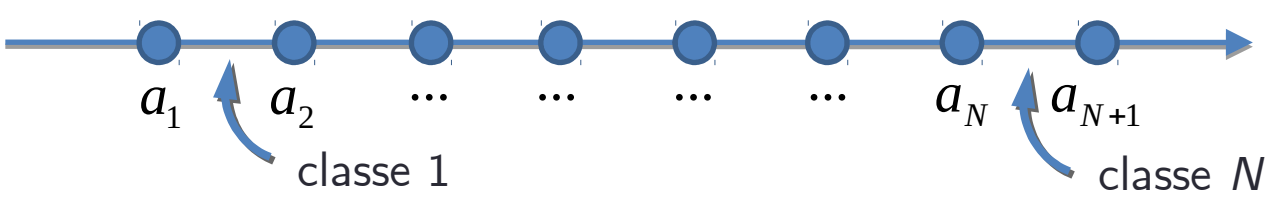
Précautions

- Choisir un nombre pertinent d'intervalles
- H ne reflète pas l'information spatiale.
- 2 images différentes peuvent avoir le même histogramme

$$H(I) = H(I') \not\rightarrow I = I'$$



Définitions des intervalles

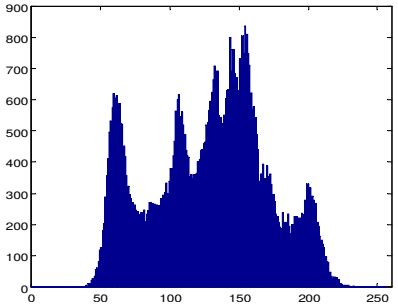


$$a \in \left[a_1, \ldots, a_{N+1}\right[$$

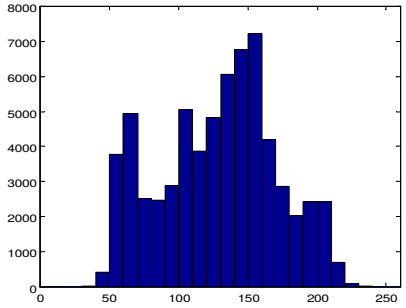
$$H(k) = \begin{cases} \#\{(i,j) \in M \times N : I(i,j) < a_{i+1}\} & i = 1 \\ \#\{(i,j) \in M \times N : a_i \leq I(i,j) < a_{i+1}\} & 1 < i < N \\ \#\{(i,j) \in M \times N : a_i \leq I(i,j)\} & i = N \end{cases}$$

amplitude de plus en plus forte

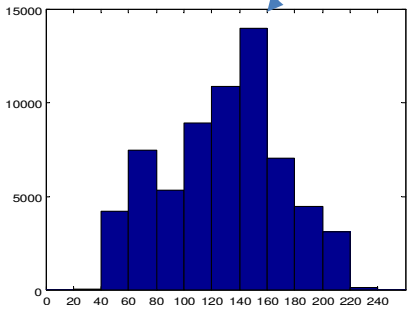
représentativité de plus en plus faible



Histogramme de « 1 en 1 »



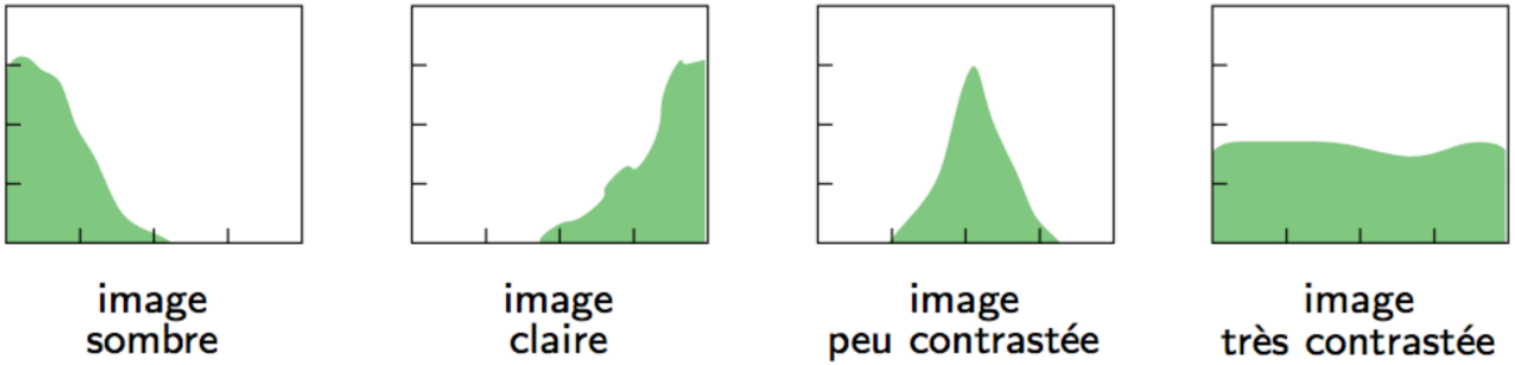
Histogramme de « 10 en 10 »



Histogramme de « 20 en 20 »

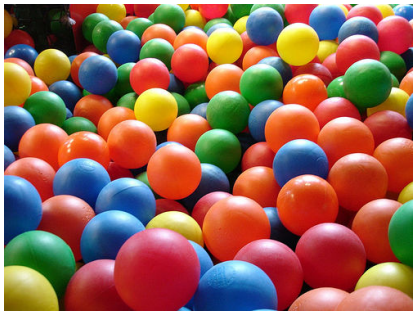
Définitions des intervalles

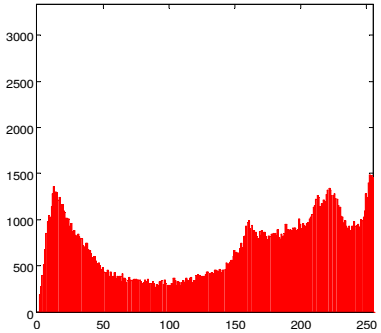
- L'histogramme d'une image nous informe sur son contraste :
  - Image sombre : valeurs proches de 0
  - Image claire : valeurs proches de 255
  - Contraste faible : valeurs tassées
  - Contraste élevé : valeur réparties



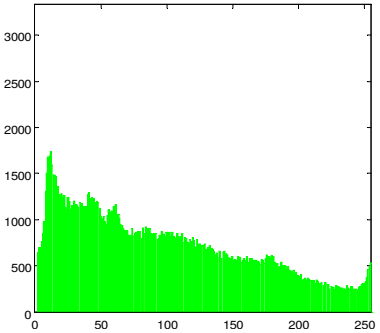
Histogramme d'une image couleur

- Histogrammes séparés pour chaque composante

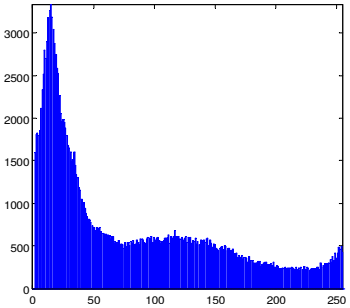




Histogramme de la composante rouge

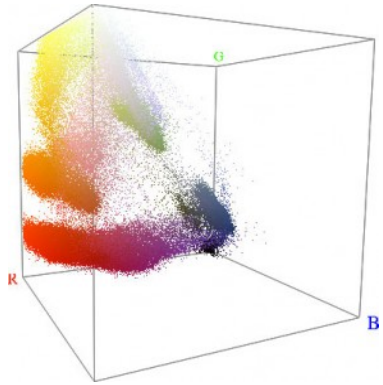


Histogramme de la composante verte



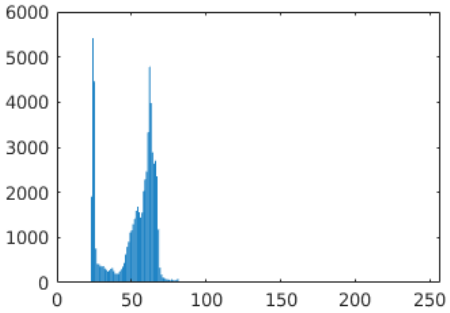
Histogramme de la composante bleue

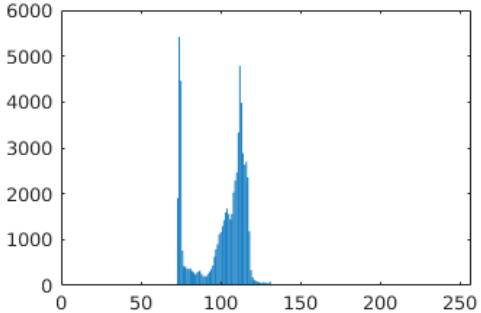
- Visualisation 3D



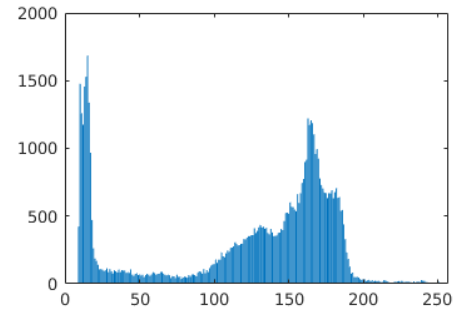
Transformations linéaires

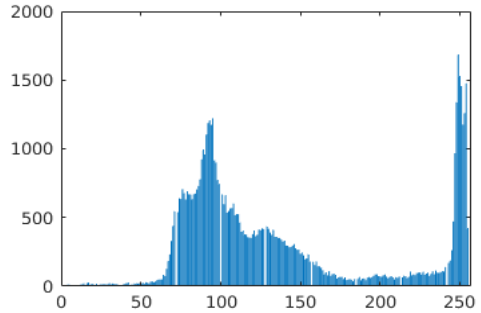
- Transposition / Gain linéaire $\{I_s(m,n)\} = \{I_e(m,n)\} + 50$





- Négatif $T(I_e) = I_{e\max} + I_{e\min} - I_e$

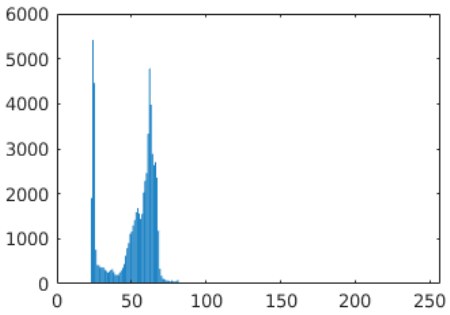


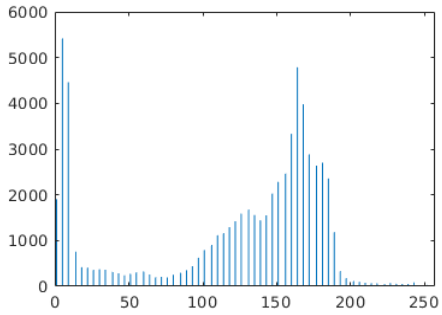


Normalisation / Étirement d'histogramme

- Modifier l'intervalle de variation $[\min, \max] \rightarrow [\min', \max']$

$$[\min, \max] \rightarrow [0, 255] \qquad I'(i,j) = \frac{255}{\max - \min}(I(i,j) - \min)$$





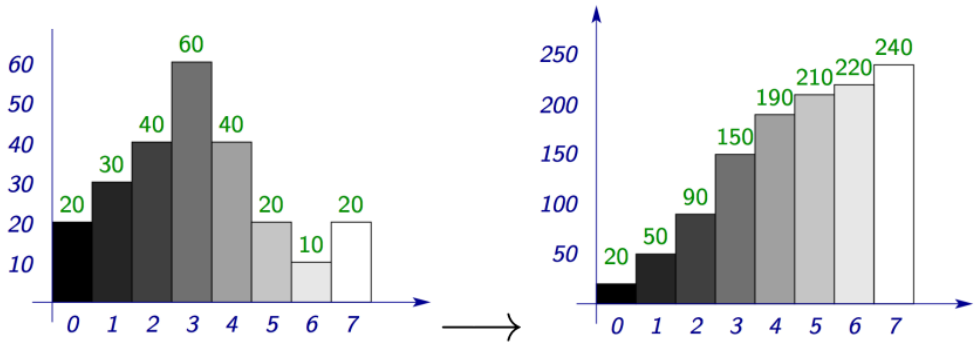
(effet produit par `imshow`)

- Quelles sont les limites de cette méthode ?

Équilibrage d'histogramme

Pour une intensité k d'une image I de taille MN à K intensités

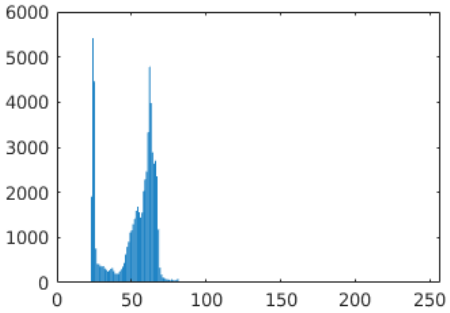
- Histogramme **normalisé** : $H_n(k) = \frac{H(k)}{MN}$; $\sum_{k=0}^{K-1} H_n(k) = 1$, avec $H_n(k) \geq 0$
  $\leftrightarrow$ loi de probabilité. Probabilité qu'un pixel ait le niveau de gris k
- Histogramme **cumulé** : $H_c(k) = \sum_{l=0}^{k} H(l)$
- Histogramme **cumulé normalisé** : $H_{cn}(k) = \sum_{l=0}^{k} H_n(l)$



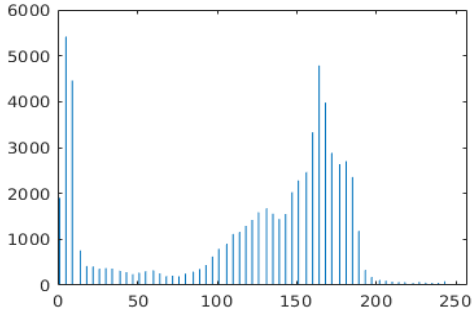
→ Nouvelle intensité k' pour un pixel (i,j), $I(i,j) = k$

$$k' = I_{\max} H_{cn}(k) = I_{\max} \sum_{l=0}^{k} H_n(l) = I_{\max} \sum_{l=0}^{k} \frac{H(l)}{MN}$$

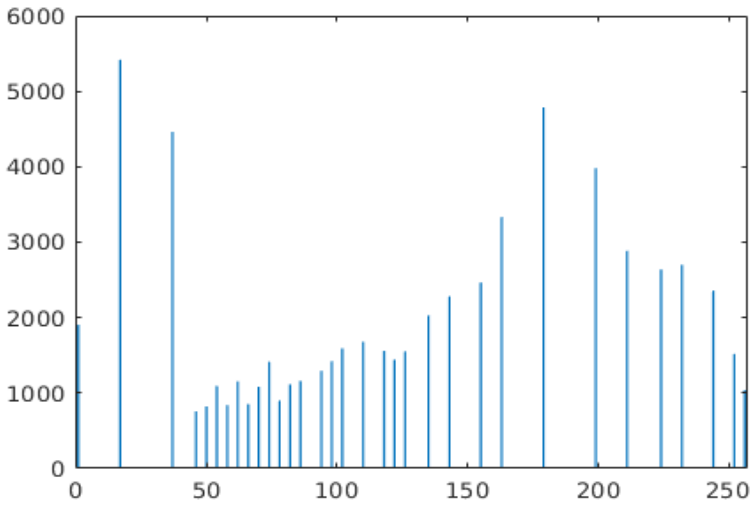
Équilibrage d'histogramme



Normalisation

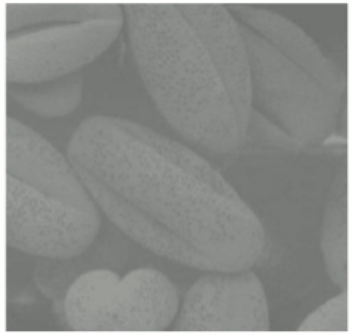


Équilibrage

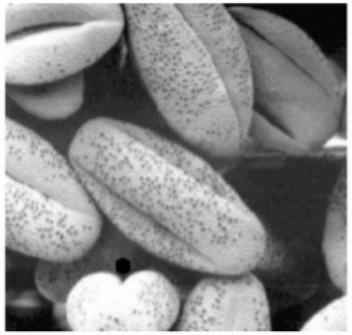


Quiz : Retrouver à qui correspondent ces histogrammes

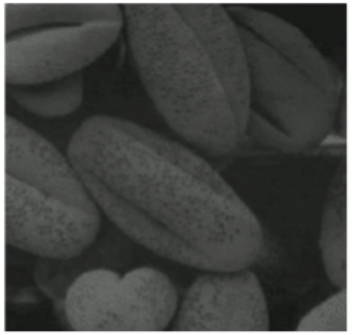
(1)



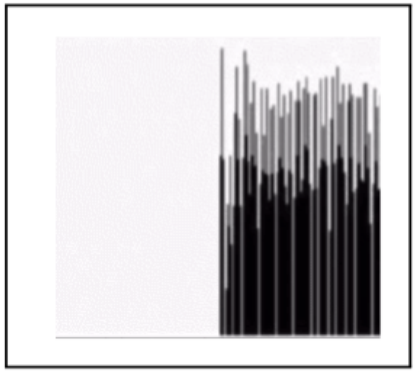
(2)



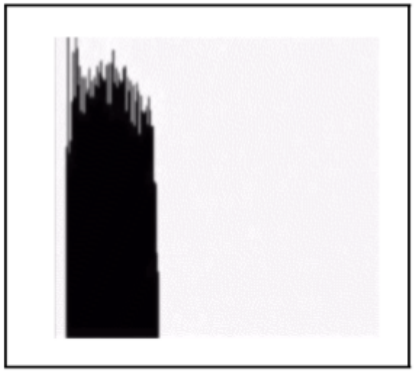
(3)



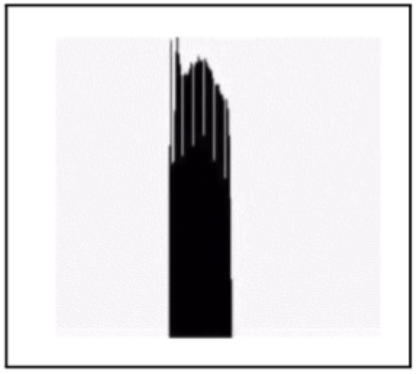
(4)



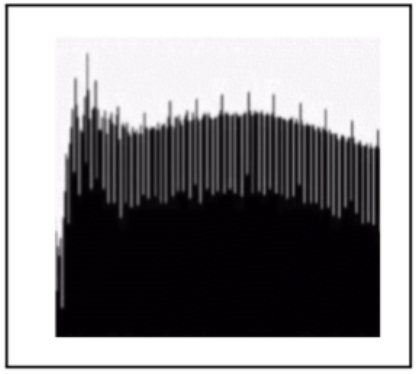
(a)



(b)



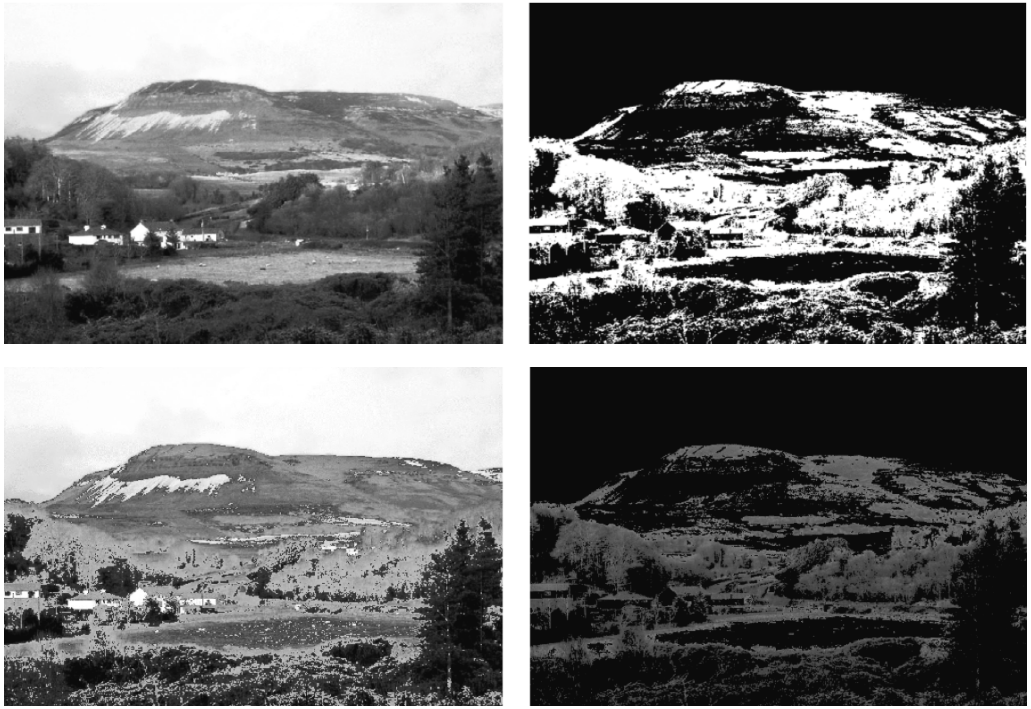
(c)



(d)

Objectifs

- Ramener l'image à quelques niveaux de gris dans une bande
- Mettre en avant des régions/détails mais n'améliore pas l'image



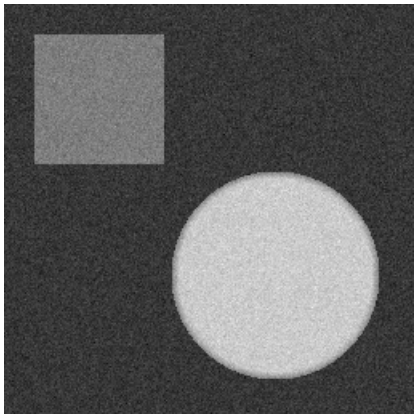
- Extraire de l'information (segmentation d'objets, chroma-keying, ...)

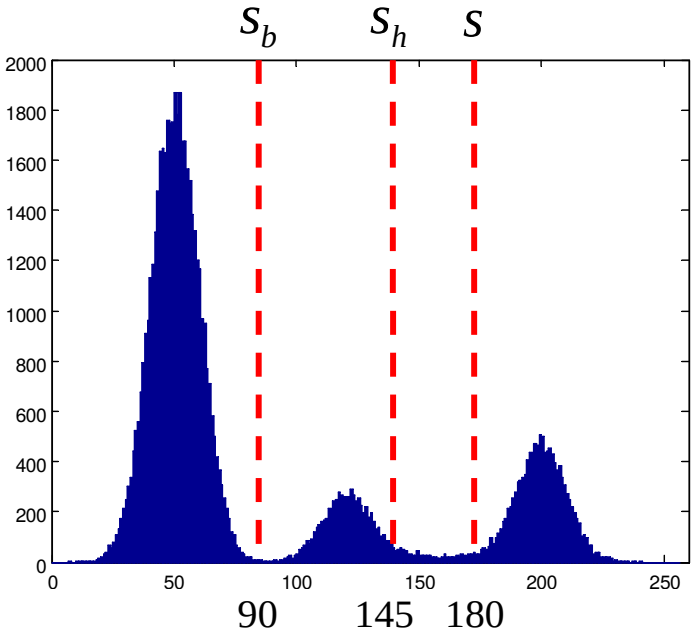
Binarisation

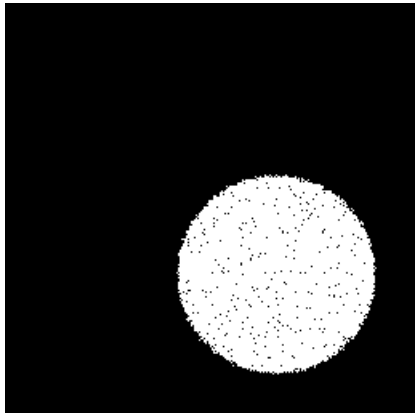
$$I_{[i_{\min},i_{\max}]}(p) = \begin{cases} 1 (\text{ou } 255), & \text{si } i_{\min} \leq I(p) \leq i_{\max} \\ 0, & \text{sinon} \end{cases}$$

En utilisant l'histogramme

$$I_s = \begin{cases} 1 & \text{si } I_e \geq s \\ 0 & \text{sinon} \end{cases}$$

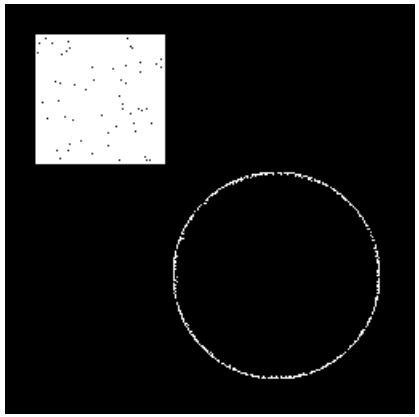






Seuillage à simple seuil

$$I_s = \begin{cases} 1 & \text{si } s_b \leq I_e \leq s_h \\ 0 & \text{sinon} \end{cases}$$

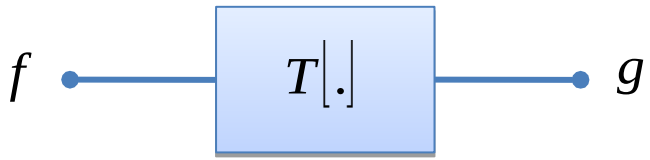


Seuillage à double seuil

TRAITEMENT SUR VOISINAGE

FILTRAGE LINÉAIRE

Système linéaire



L'entrée et la sortie d'un système linéaire T sont reliées par un produit de convolution

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

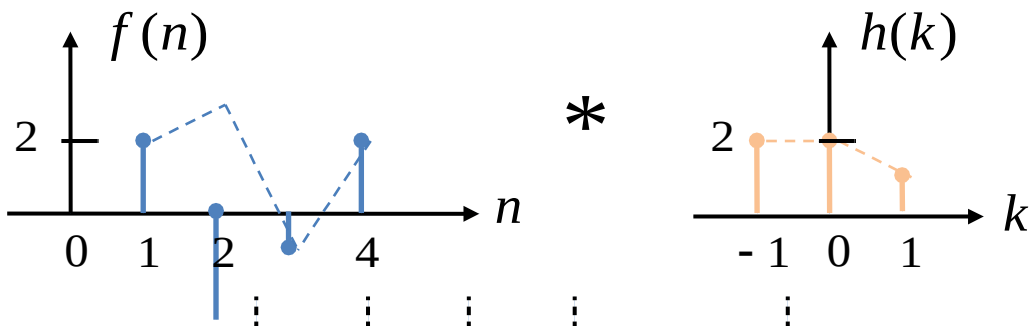
Filtre linéaire RIF

Application d'un filtre linéaire discret à réponse impulsionnelle finie

défini sur $2K+1$ échantillons

$$g(n) = \sum_{k=-K}^{K} f(n-k)h(k) \quad \Longleftrightarrow \quad I_s = I_e * H$$

Convolution discrète 1D

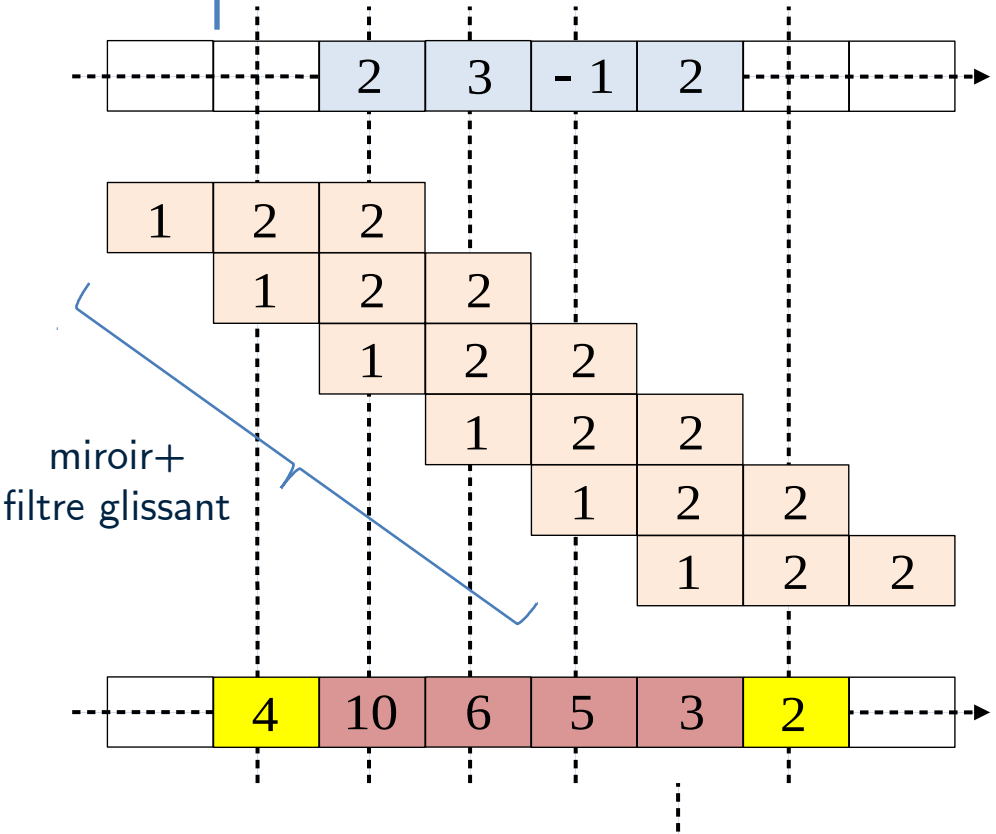


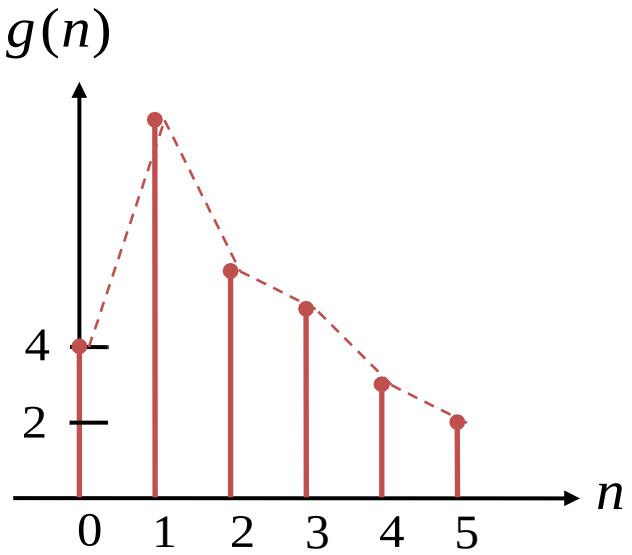
*

$$g(n) = \sum_{k=-K}^{K} f(n-k)h(k)$$

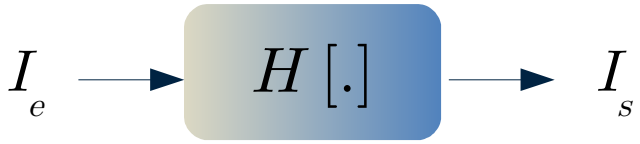
$$= \sum_{k=-K}^{K} f(n+k)h(-k)$$

miroir



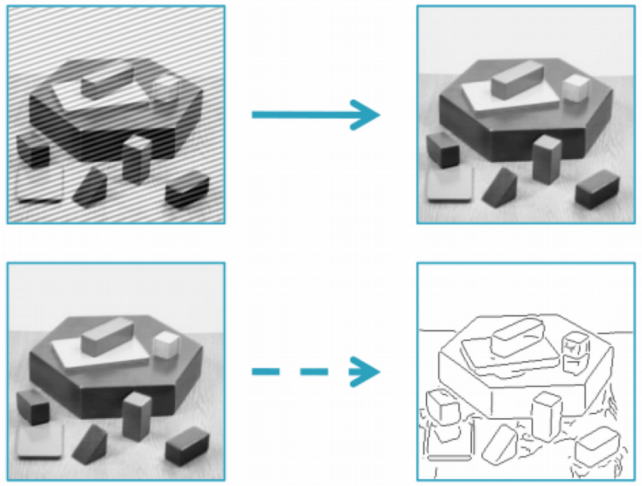


Système linéaire



Entrée et sortie reliées par un produit de convolution

$$I_s = I_e * H$$



débruitage

contours

Filtre linéaire RIF

défini sur (2K+1)x(2L+1) échantillons

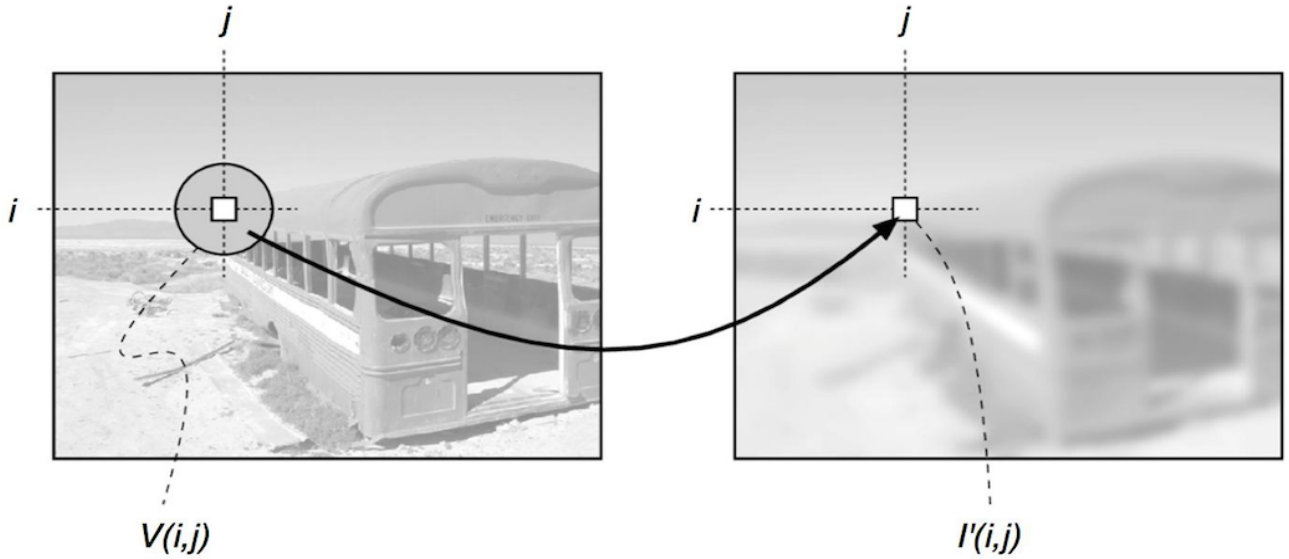
$$g(m,n) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} f(m-k, n-l) h(k,l)$$

Qu'est-ce qu'un filtre 2D ?

Opérateur local ou « de voisinage » qui permet de :

- Débruiter,
- Détecter les contours,
- Transformer géométriquement l'image, ...

Principe : Combiner la valeur du pixel I(i, j) avec son voisinage V(i, j) défini autour de (i, j). Par ex. : moyenne sur V(i, j)

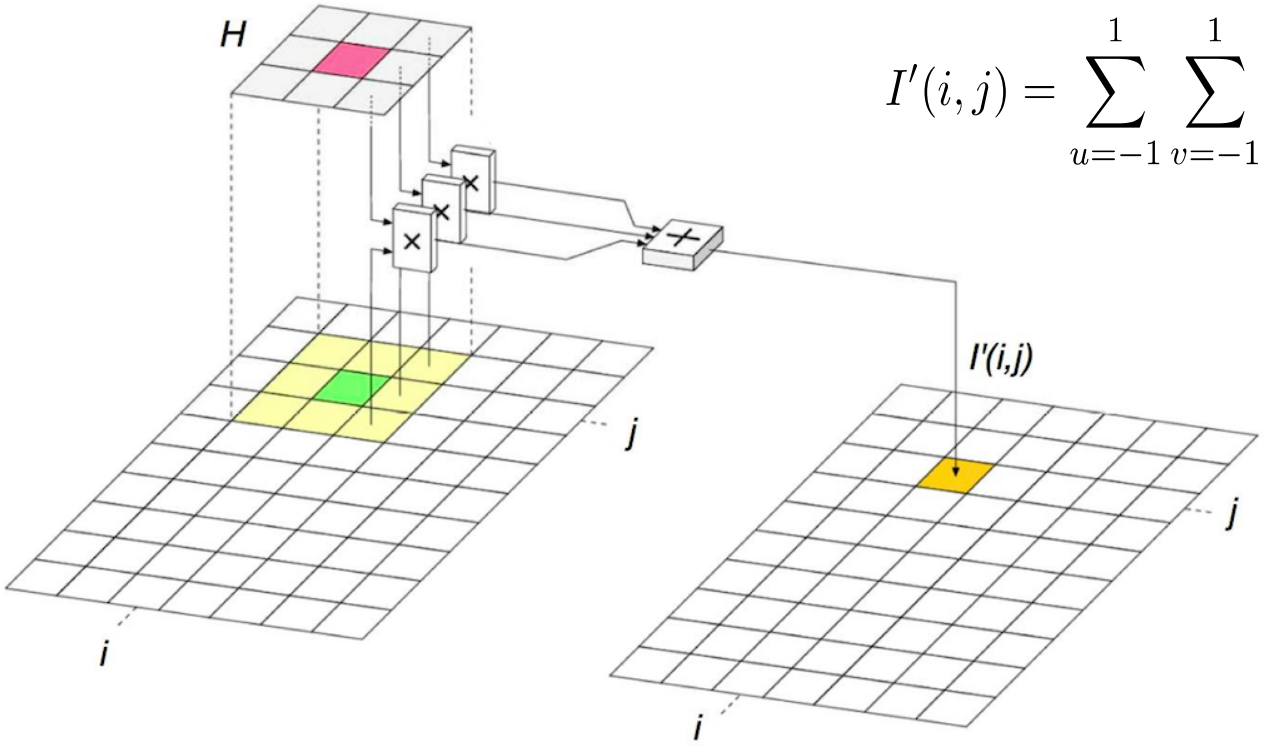


Produit de convolution 2D

$$I'(i,j) = \sum_{(u,v)\in H} I(i-u, j-v).H(u,v)$$ avec H centré sur (i,j)

Par exemple pour un filtre 3x3 :

$$I'(i,j) = \sum_{u=-1}^{1} \sum_{v=-1}^{1} I(i-u, j-v).H(u,v)$$



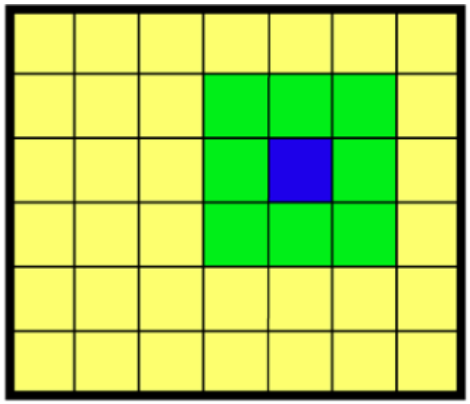
Exemple : filtre moyenneur

$$I'(i,j) = \frac{1}{|V(i,j)|} \sum_{(u,v)\in V(i,j)} I(i-u, j-v)$$

Différents paramètres :

- Taille : 3x3, 5x5, ...
- Pondération : coefficient spatiaux, souvent isotrope

Exemple : $V(i,j) \rightarrow 3 \times 3$ $\quad I'(i,j) = \frac{1}{9} \sum_{u=-1}^{1} \sum_{v=-1}^{1} I(i-u, j-v)$

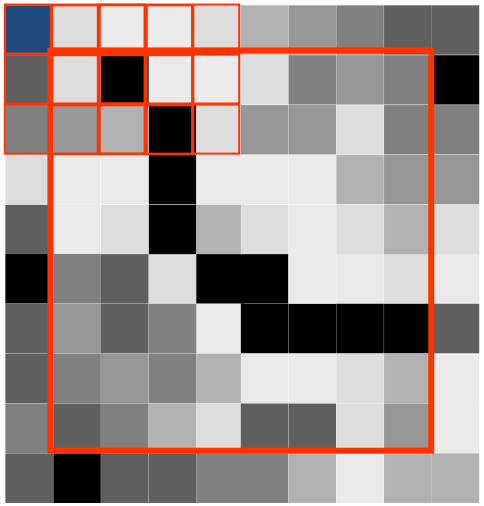


$$H(u,v) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & \underline{\mathbf{1/9}} & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \underline{\mathbf{1}} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Exemple : filtre moyenneur

h

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



A

0	5	7	6	5	4	3	2	1	1
1	5	0	7	6	5	2	3	2	0
2	3	4	0	5	3	3	5	2	2
5	7	6	0	6	6	6	4	3	3
1	7	5	0	4	5	6	5	4	5
0	2	1	5	0	0	7	6	5	6
1	3	1	2	7	0	0	0	0	1
1	2	3	2	4	6	7	5	4	7
2	1	2	3	5	1	1	5	2	6
1	0	1	1	2	2	3	6	4	4

Novelle valeur d

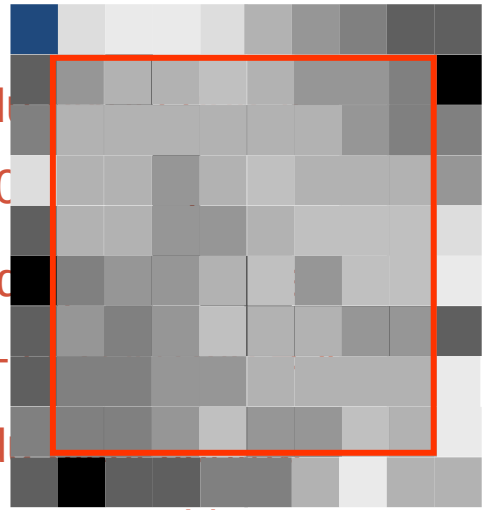
= (0+5+7+1+5+0

Novelle valeur d

= (5+7+6+5+0+

Novelle valeur d

= (7+6+5+0+7+6+4+0+5)/9 = 4

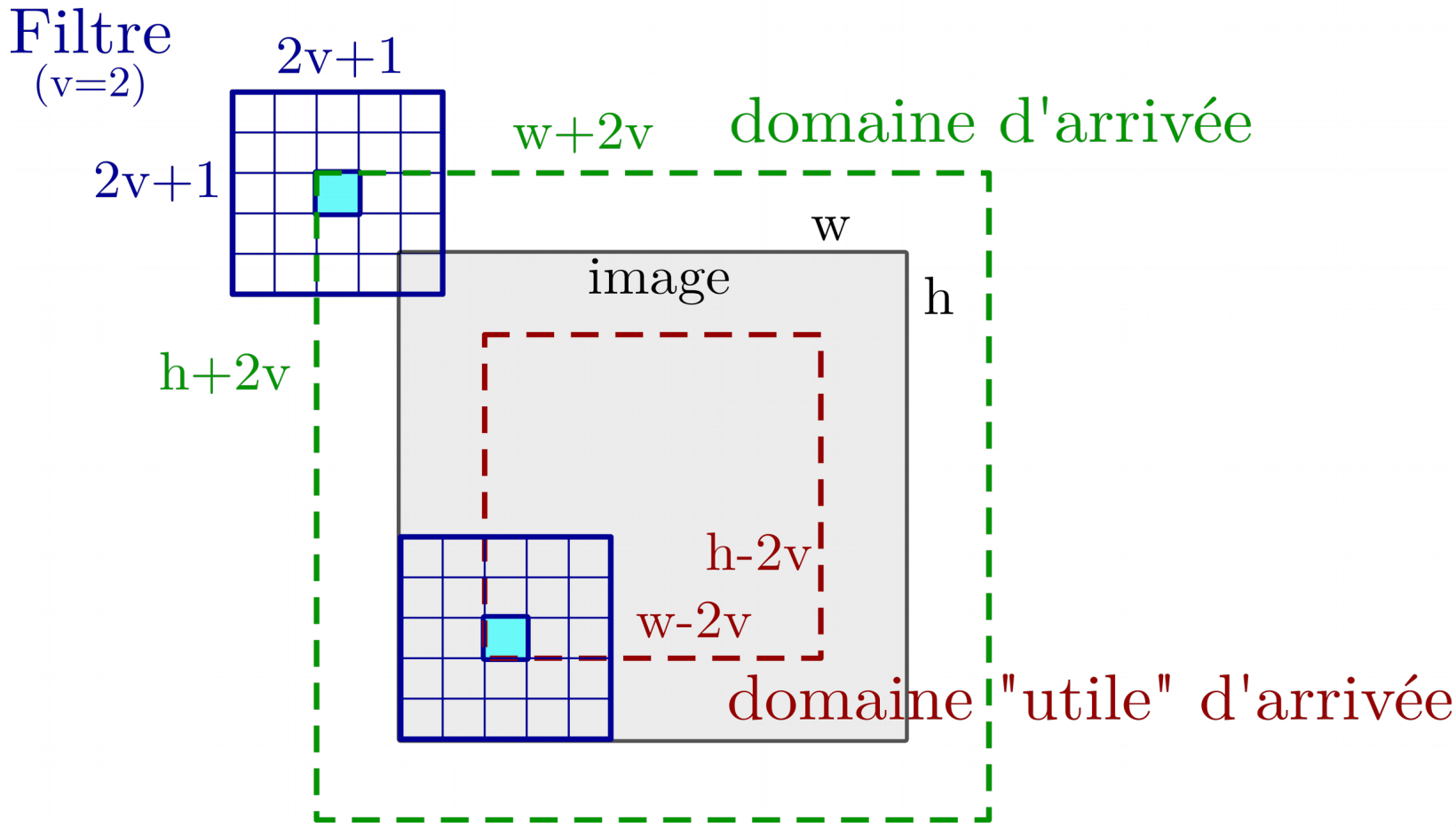


C

0	5	7	6	5	4	3	2	1	1
1	3	4	4	5	4	3	3	2	0
2	4	4	4	4	4	4	3	2	2
5	4	4	3	4	5	4	4	4	3
1	4	4	3	3	4	5	5	5	5
0	2	3	3	4	5	3	5	5	6
1	3	2	3	5	4	4	3	3	1
1	2	2	3	3	4	4	4	4	7
2	2	2	3	5	3	3	5	4	6
1	0	1	1	2	2	3	6	4	4

Gestion des effets de bords

- Différents domaines/tailles possibles en sortie de convolution



Gestion des effets de bords

- Différentes stratégies pour étendre l'image aux bords

zero-padding

extension

miroir

périodique

Filtres moyenneurs

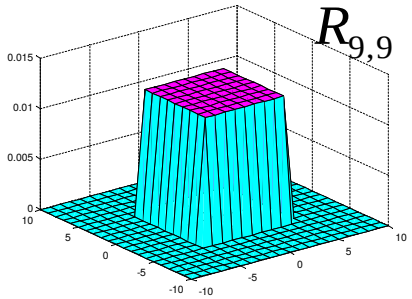
- Filtre moyenneur rectangle uniforme

normalisation

$$\begin{cases} R_{2K+1} = \dfrac{1}{(2K+1)} \underbrace{\begin{bmatrix} 1 & \dots & \dots & \dots & 1 \end{bmatrix}}_{2K+1} \\ R_{2K+1,2L+1} = R_{2K+1} * R_{2L+1}{}^{T} \end{cases}$$

$$R_3 = R_{3,1} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$R_{1,3} = R_{3,1}{}^{T} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

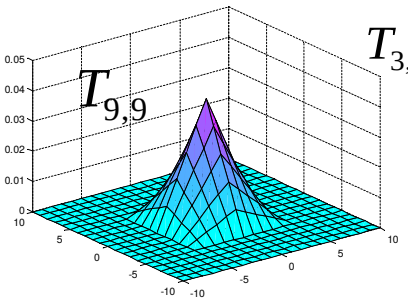


$$R_{3,3} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

plus petit filtre rectangle 2D

- Filtre moyenneur triangle

$$\begin{cases} T_{2K+1} = \dfrac{1}{(K+1)^2}\begin{bmatrix} 1 & 2 & \dots & K+1 & \dots & 2 & 1 \end{bmatrix} \\ T_{2K+1,2L+1} = T_{2K+1} * T_{2L+1}{}^{T} \end{cases}$$



$$T_{3,3} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

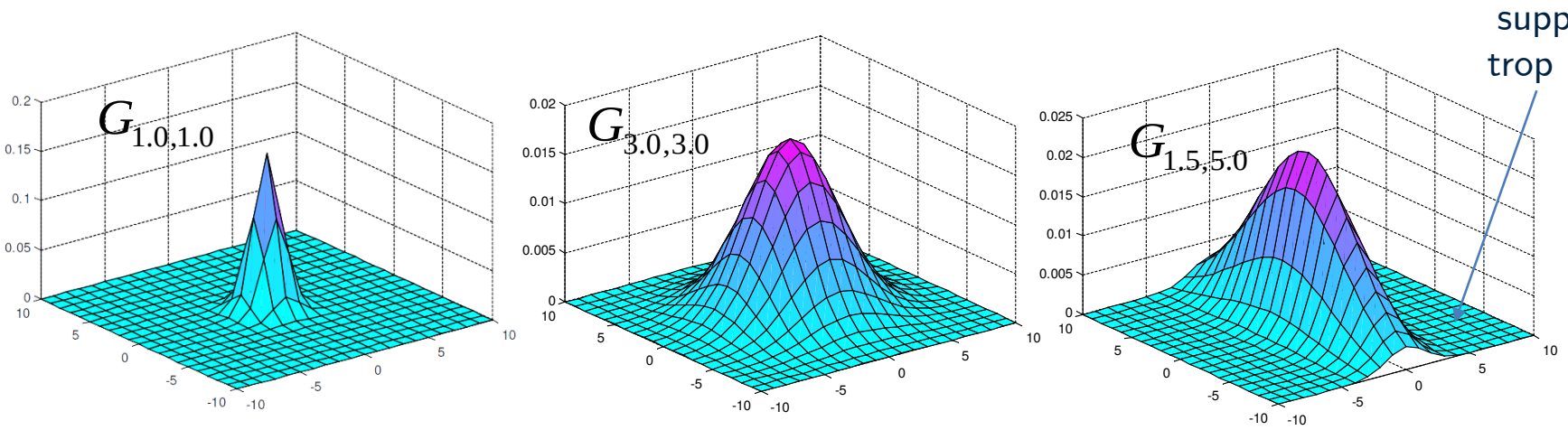
plus petit filtre triangle 2D

Filtres moyenneurs

- **Gaussien** : Pondère spatialement la contribution des pixels voisins

$$H(u,v) = \exp\left(-\left(\frac{u^2}{2\sigma_u^2} + \frac{v^2}{2\sigma_v^2}\right)\right)$$

- (u,v) exprime la distance par rapport au centre de la Gaussienne :
- Le paramètre σ détermine la largeur de la Gaussienne (écart-type) :
  → contrôle un filtrage +/- fort
  → permet de connaître la taille du support nécessaire

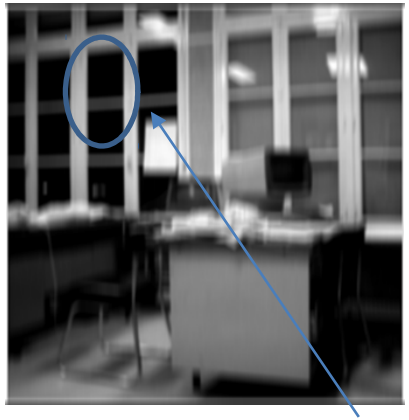


Filtres moyenneurs

$R_{11,1}$

$R_{1,11}$

$G_{3.0,3.0}$



flou vertical

flou isotrope

$G_{1.5,1.5}$

étalement

Atténuation du bruit par moyennage spatial

Bruit

Définition : Phénomène parasite (erreurs, dégradations, . . .)

- La distribution peut être (en partie) connue

Origine : capture, lumière, acquisition, lentille, mouvement, ...

Modèle : Le plus courant additif $\mathrm{I} = \mathrm{I}_0 + \mathrm{B}$

I : image observée (capturée), I_0 : image non corrompu, B : le bruit

Débruitage : Calculer I_0 depuis I avec une hypothèse sur B

- Connaissance du bruit ↔ utiliser le bon filtre
- Différents types de bruit : Gaussien, aléatoire, périodique

Bruit blanc Gaussien additif

- Affecte chaque pixel par un gain additif
- Souvent « blanc » (moyenne nul)
- Suit une loi normale de paramètre sigma σ

$\sigma = 10$

$\sigma = 30$

Bruit impulsionnel aléatoire (« poivre et sel »)

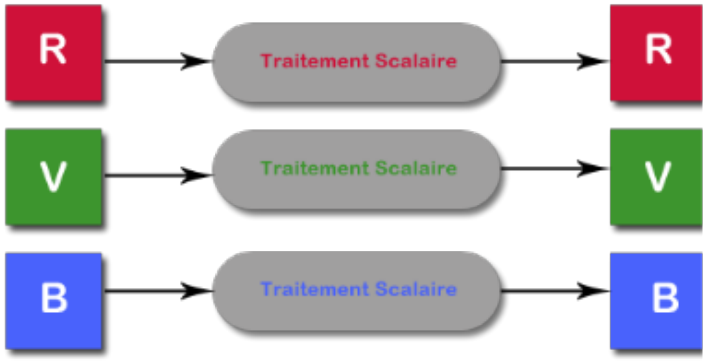
- Affecte un nombre variable de pixels
- Valeur extrême 0 ou 255
- Permet de tester les algorithmes de débruitage, restauration

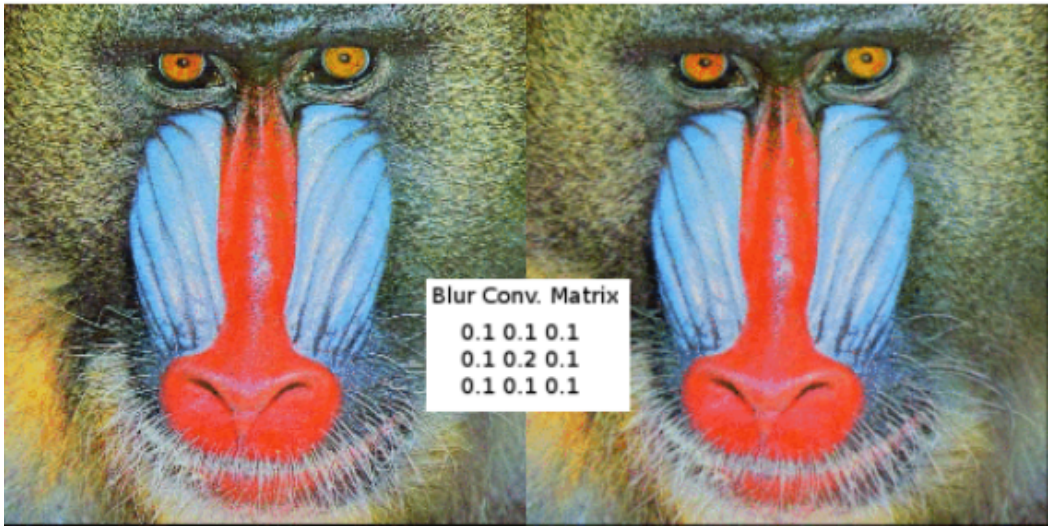
Bruit périodique

- Affecte toute l'image
- Débruitage dans l'espace fréquentiel plus adapté...

Cas d'une image couleur

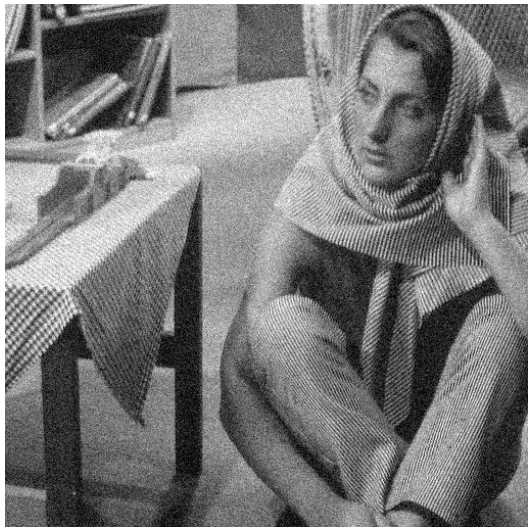
- Combinaison de chaque canal filtré indépendamment





Filtrage moyenneur (linéaire)

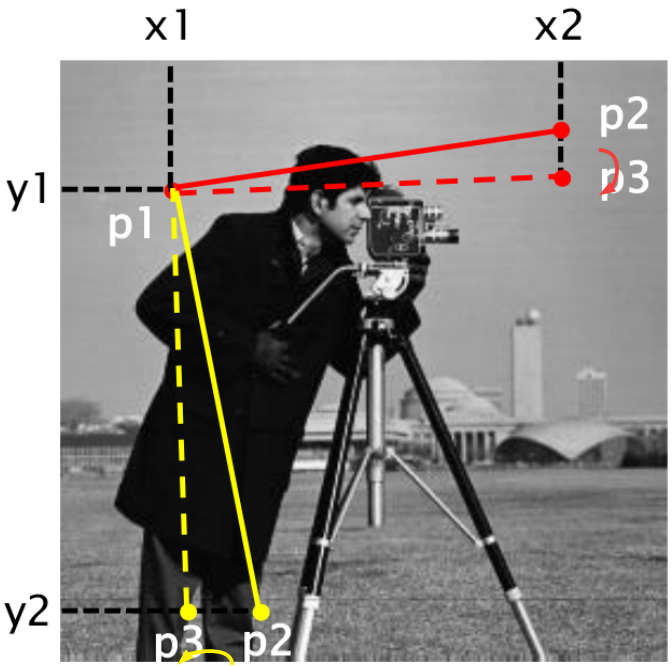
- Implémenter les filtrages linéaires suivants :
  - **Filtrage moyenneur** carré (`scipy.signal.convolve2d`)
  - **Filtrage Gaussien** (`np.meshgrid, signal.convolve2d`)
- Tester sur les images : *barbara_awgn_noise.png*, et *cameraman_sp_noise.png*

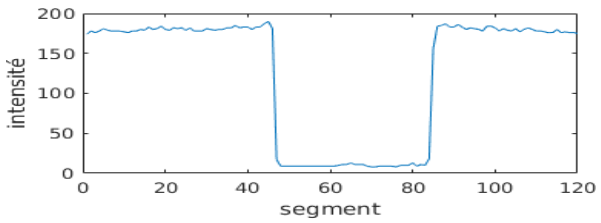


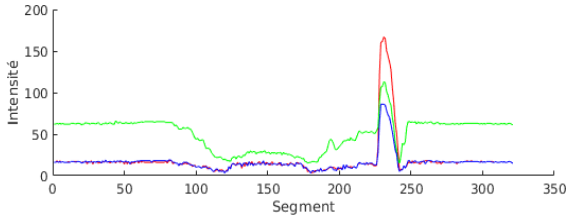
- Pour quelles images les filtrages s'avèrent-ils satisfaisants ? Pourquoi ?

Extraction de lignes

- Affichage d'une image
  (par exemple *cameraman.tif*)
- Définir un segment horizontal
  ou vertical par 2 clics (`plt.ginput`)
- Aligner les points en préférant
  le plus large segment
  `if(abs(x1-x2)>abs(y1-y2))`
- Extraire le profil 1D correspondant :
  `I[y1,x1:x2]` ou `I[y1:y2,x1]`
- Afficher le profil 1D
- Superposer les profils R, G, B
  d'une image couleur







Filtrage moyenneur (linéaire)

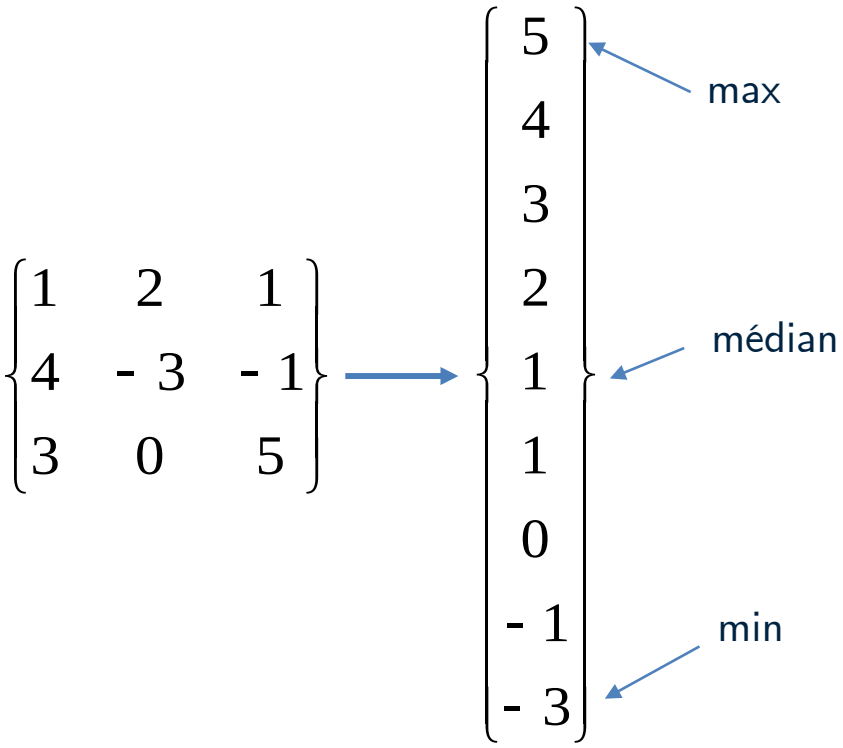
- Flouter l'image *street.png* avec un filtre moyenneur carré
- Afficher l'image et identifier la position du centre d'un visage
- Créer un masque binaire avec des 1 autour de ce centre
- Combiner les deux images grâce au masque pour obtenir une image où seuls les visages sont floutés.
- Adapter le pipeline pour l'image en couleur et plusieurs positions.

TRAITEMENT SUR VOISINAGE

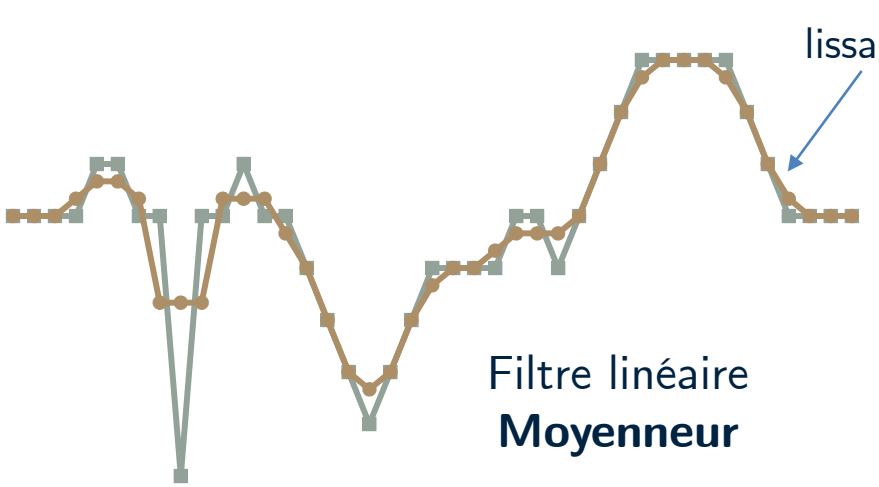
FILTRAGE NON LINÉAIRE

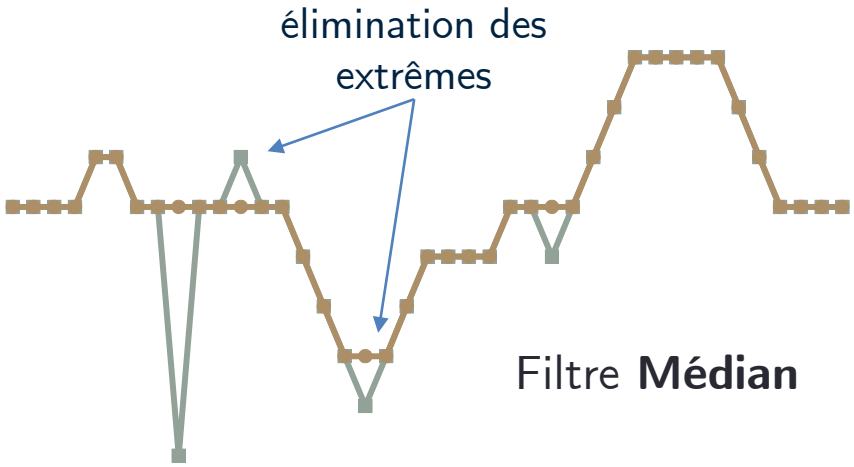
Principe

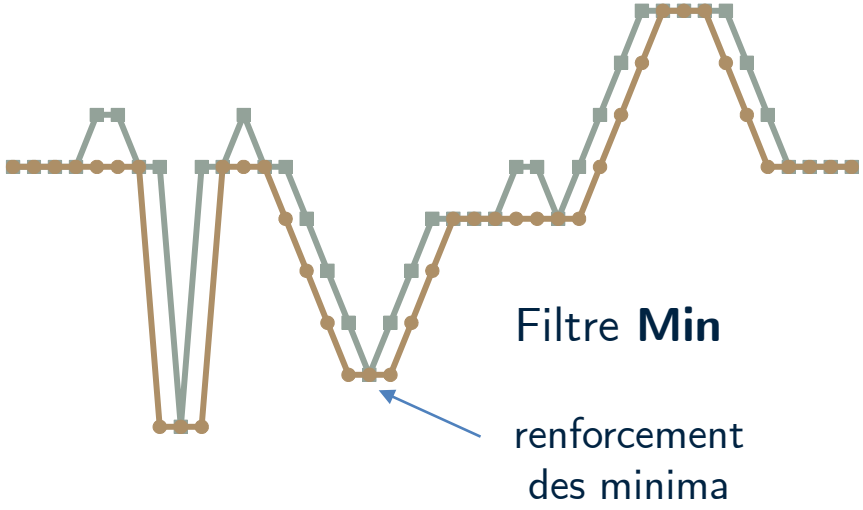
- Trier les intensités d'un voisinage et affecter l'intensité d'un rang donné

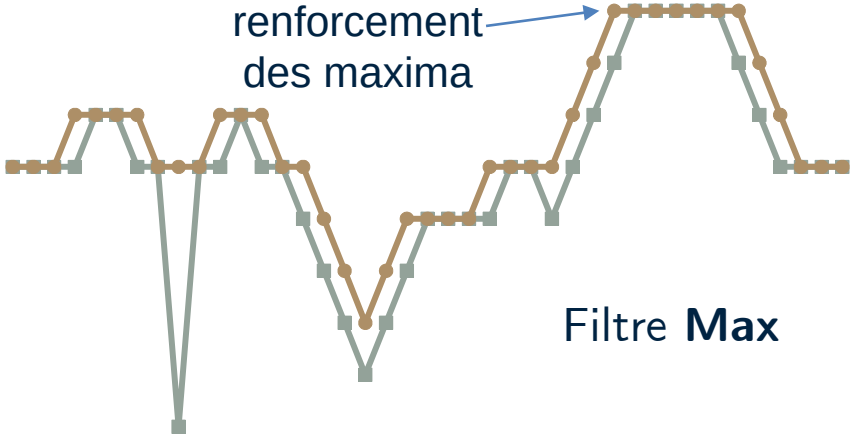


Exemples 1D : filtres de largeur 3









Exemple 2D : Comparaison passe-bas / médian

- Meilleure efficacité sur certains types de bruit et d'images

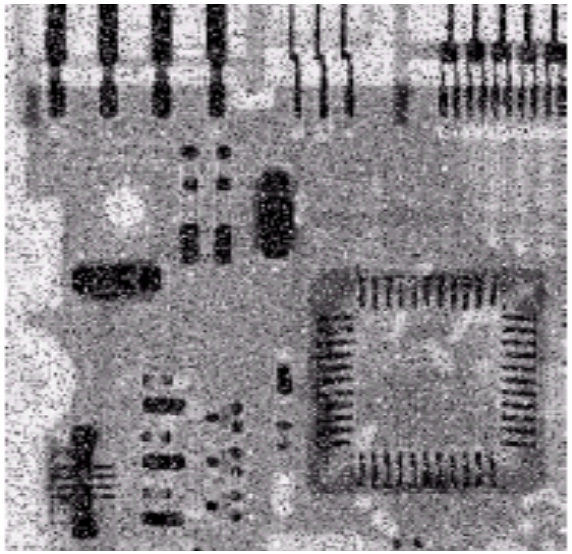
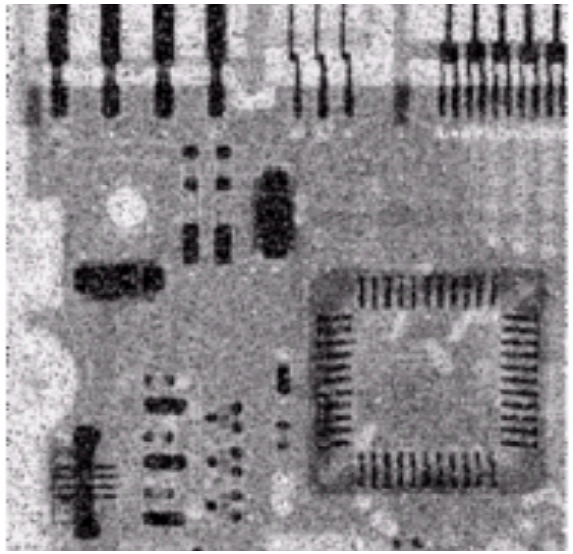
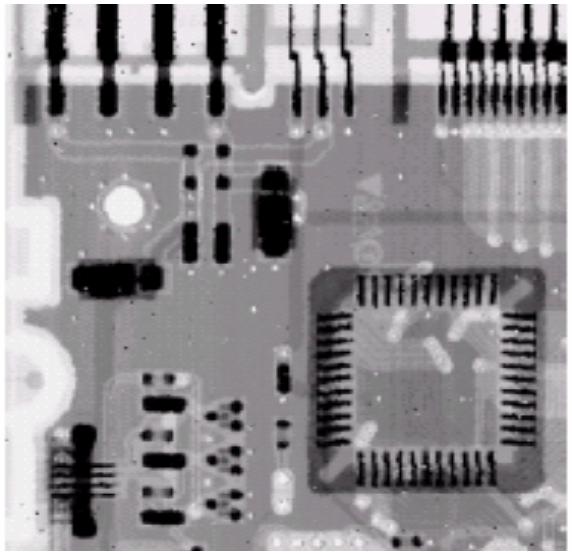


Image bruitée



Filtrage Moyenneur

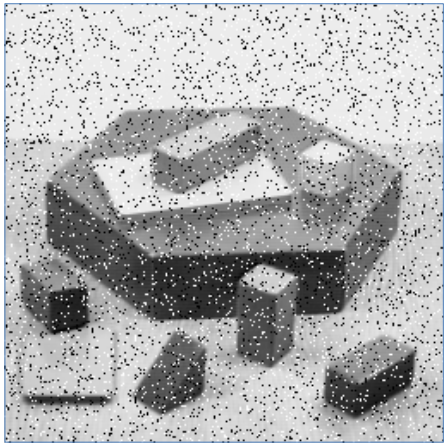


Filtrage Médian

Min / Max

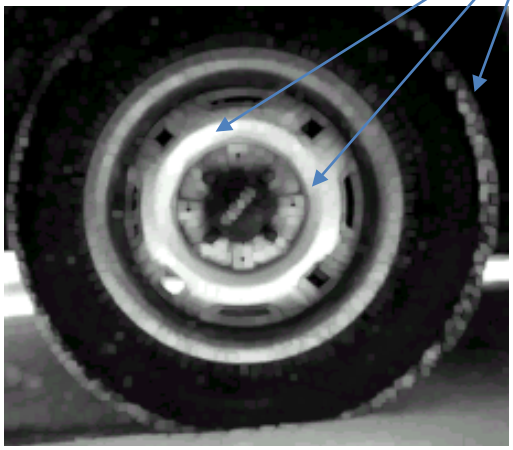
étalement des « noirs »

étalement des « blancs »



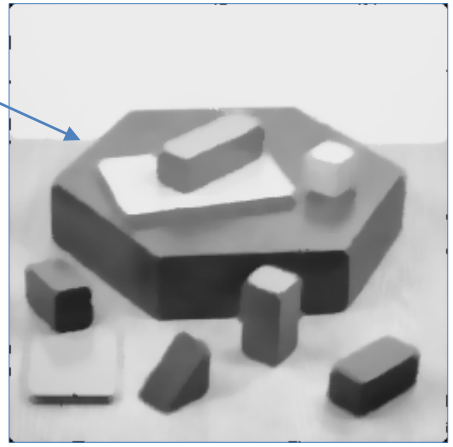
Bruit « poivre et sel »

Filtrage Min



Filtrage Max

contours préservés



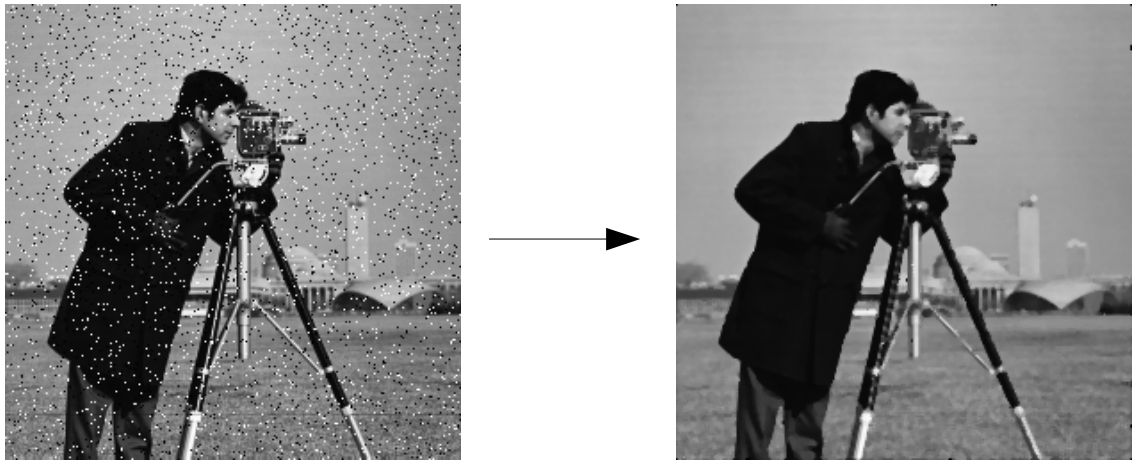
Filtrage Médian

Filtrage médian (non linéaire)

- Implémenter le filtrage médian (valeur médiane dans un voisinage)

Algo :

```
#Parcourir tous les pixels de l'image
for i in range(v, h-v):
    for j in range(v, w-v):
        #Déterminer un voisinage 2D (de taille (2v+1)*(2v+1))
        #...
        #Calculer la valeur médiane sur l'ensemble des pixels
        #...
```



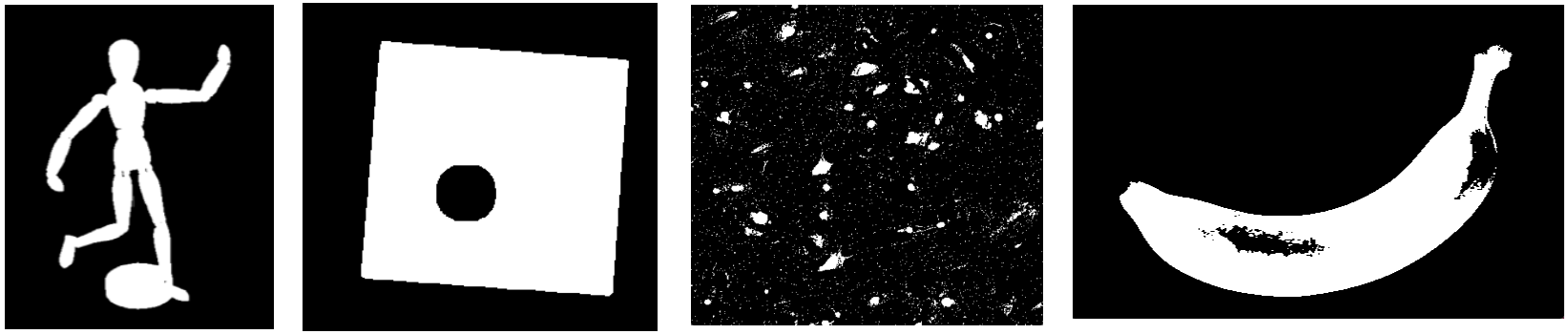
- Comparer votre résultat avec celui de `scipy.ndimage.median_filter`

OPÉRATEURS DE MORPHOLOGIE MATHÉTMATIQUE

Opérateurs de morphologie mathématique

Liés à la notion d'objets sur des images binaires :

- Pixels objets identifiés par l'intensité 1 (blanc)
- Pixels de fond identifiés par l'intensité 0 (noir)



Exemples d'images binaires

Principe : image binaire (entrée) → image binaire (sortie)

- Image de sortie obtenue par convolution, puis seuillage binaire
- Ces opérateurs peuvent permettre de :
  - Supprimer du bruit
  - Grossir/Diminuer la taille d'un objet
  - Décoller/Recoller les composantes (connexes) d'un objet

Érosion

- Un pixel d'une image I est un pixel objet si la région centrée sur ce pixel ne contient que des pixels objet → "rétrécit" l'objet
- Équivalent à convoluer, par ex. par un filtre de type 4 voisins : $h = 1/5 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$$I_c = I * h$$

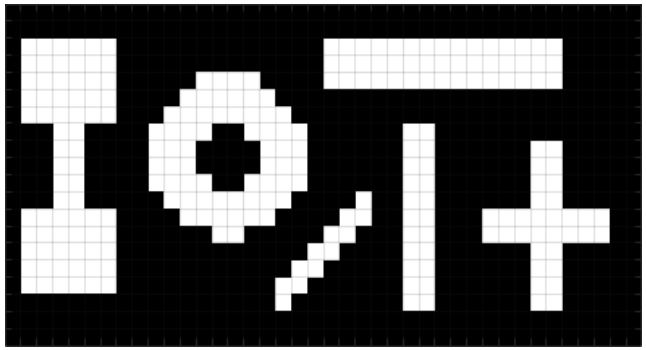


Image initiale I

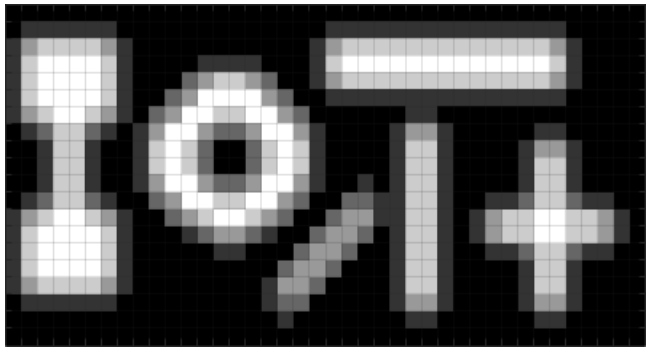


Image convoluée I_c

- Puis seuiller de manière binaire :

$$I_e(i,j) = \begin{cases} 1 \text{ si } I_c(i,j) = 1 \\ 0 \text{ sinon} \end{cases}$$

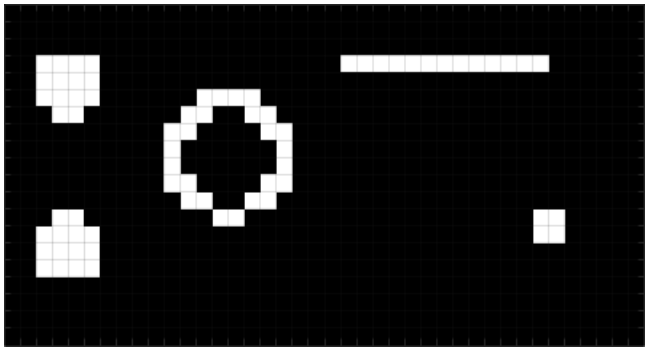


Image érodée I_e

Dilatation

- Un pixel est un pixel objet si la région centrée sur ce pixel contient au moins un pixel objet → "épaissit" l'objet
- Équivalent à convoluer, par ex. par un filtre de type 4 voisins : $h = 1/5 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

$$I_c = I * h$$

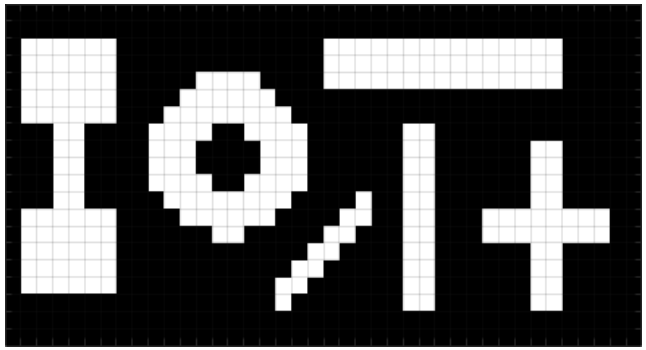


Image initiale I

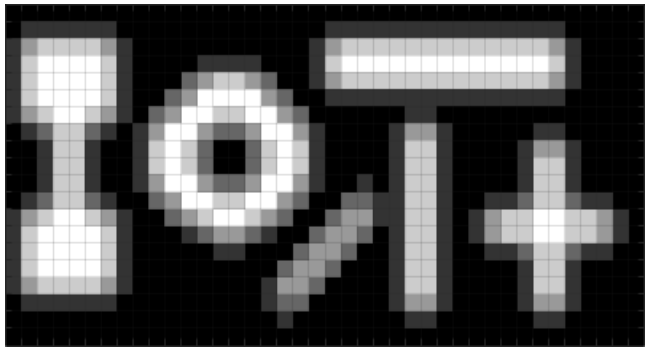


Image convoluée I_c

- Puis seuiller de manière binaire :

$$I_d(i,j) = \begin{cases} 1 \text{ si } I_c(i,j) > 0 \\ 0 \text{ sinon} \end{cases}$$

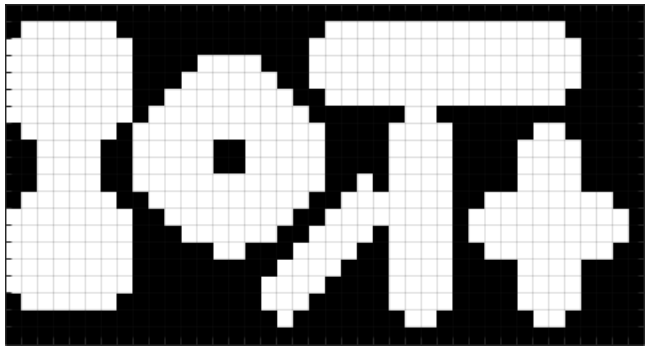


Image dilatée I_d

Fermeture

- Fusionne les structures proches, les trous sont comblés. Permet de recoller les composantes d'un objet.
- Effectuer une **Dilatation** puis une **Érosion**.

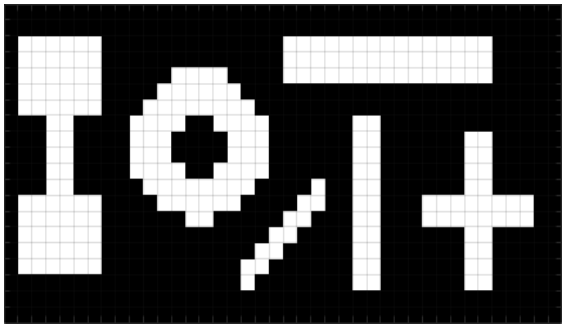


Image initiale

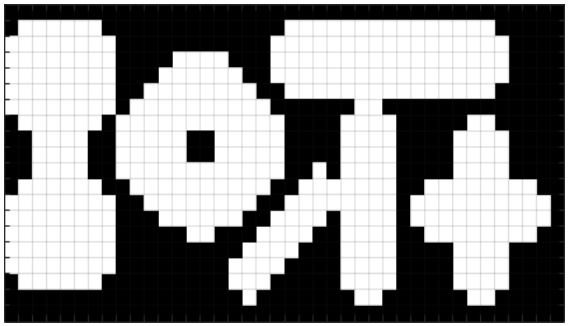


Image dilatée

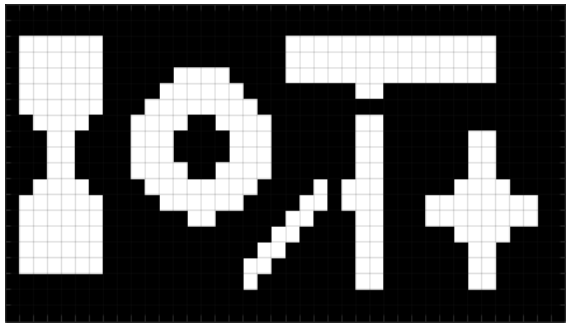


Image dilatée puis érodée

- Possibilité **d'itérer** chaque étape

Image initiale

Image dilatée (x20)

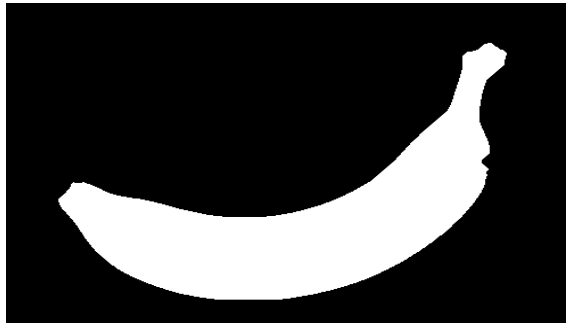


Image dilatée (x20) puis érodée (x20)

Ouverture

- Supprime les petites structures. Permet de déconnecter des objets indépendants.
- Effectuer une **Érosion** puis une **Dilatation**.

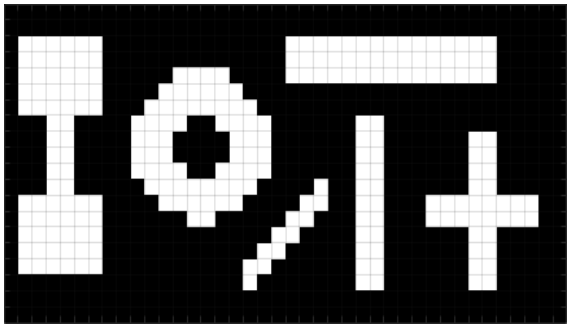


Image initiale

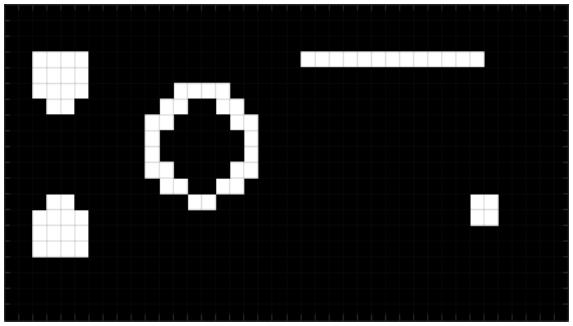


Image érodée

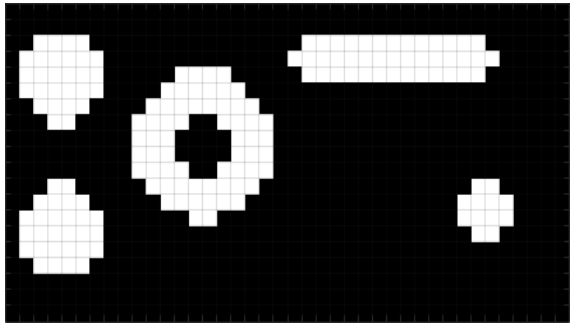


Image érodée puis dilatée

- Possibilité **d'itérer** chaque étape

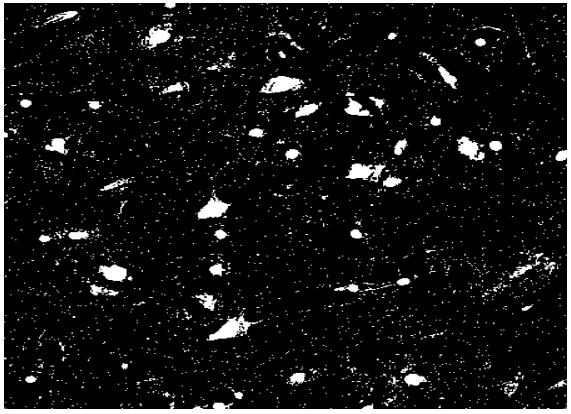


Image initiale

Image érodée (x2)

Image érodée (x2) puis dilatée (x2)

Composantes connexes

- Ensemble de pixels connectés
  - Au sens d'un critère (par ex. d'intensité 1) et d'un voisinage (4, 8, etc).
  - Possibilité d'atteindre tous les pixels de la composante sans en sortir.
- Étiquetage en composantes connexes : parcours d'une image binaire en assignant un nombre entier (étiquette/label) à chaque composante connexe de l'image. (`skimage.measure.label`)

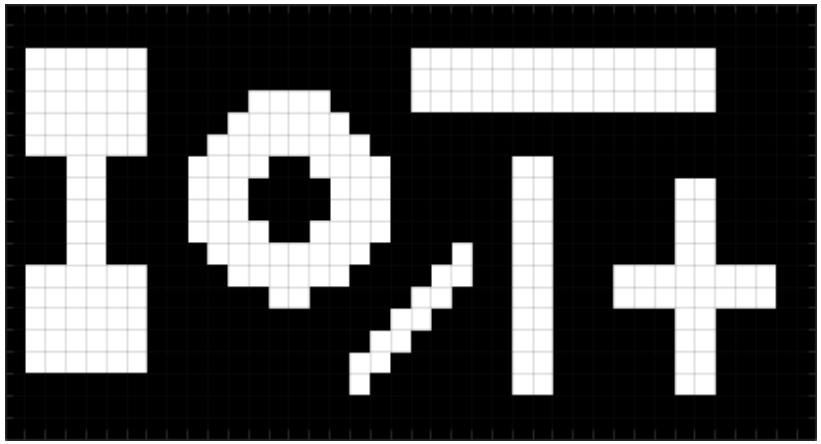
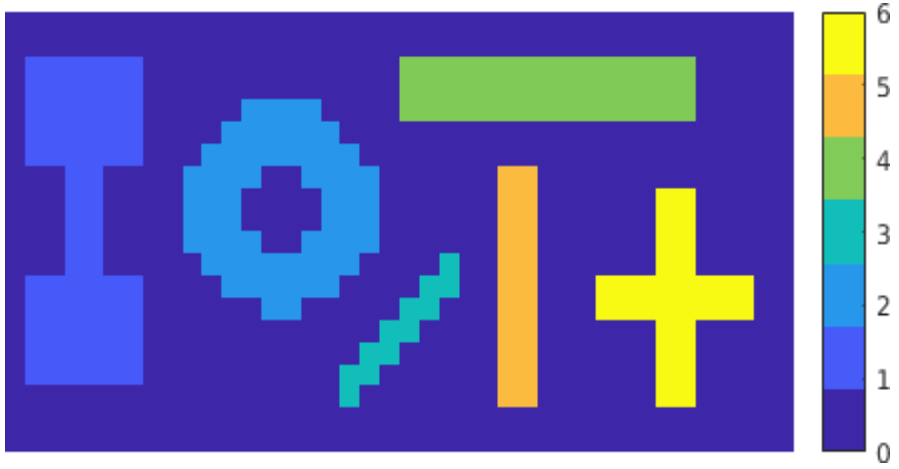


Image initiale



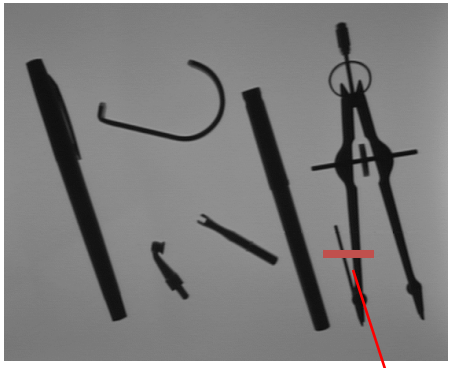
Étiquetage en composantes connexes

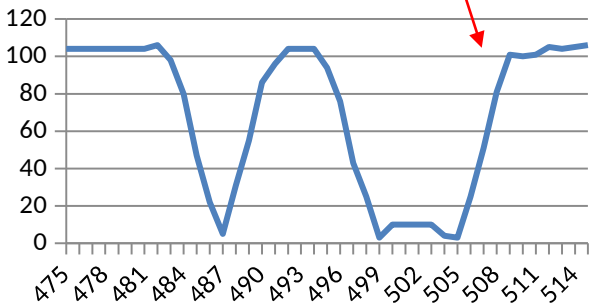
DÉTECTION DE CONTOURS

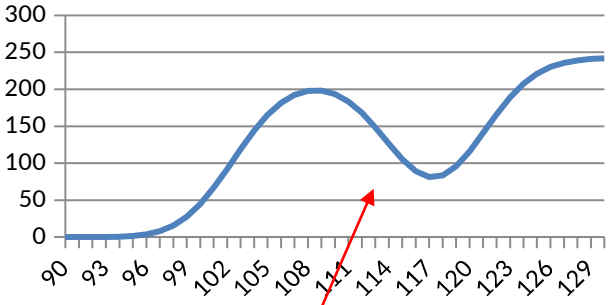
Définition

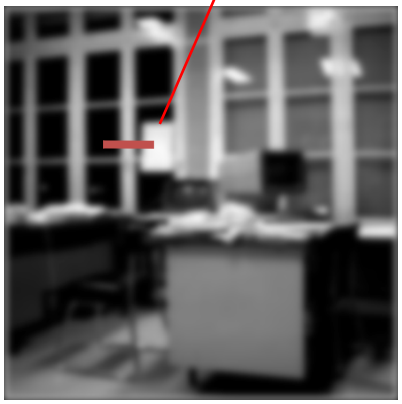
- Bord ou limite d'une région (objets, intensités)
- Séparation ou frontière entre régions
- Variation plus ou moins rapide d'un front d'intensité

Contours nets



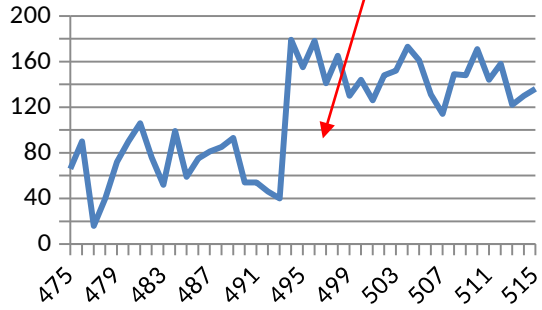






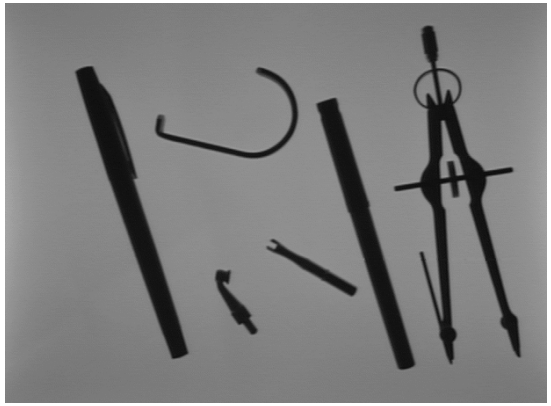
Contours flous

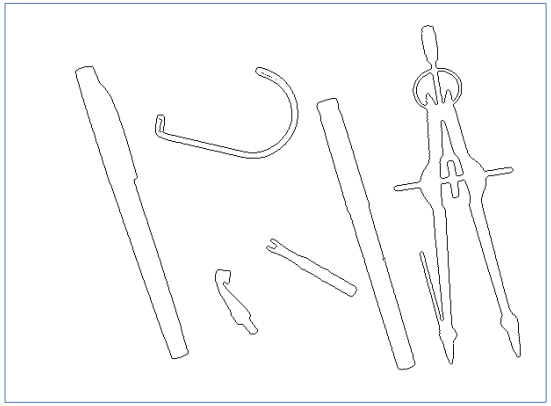
Contours bruités



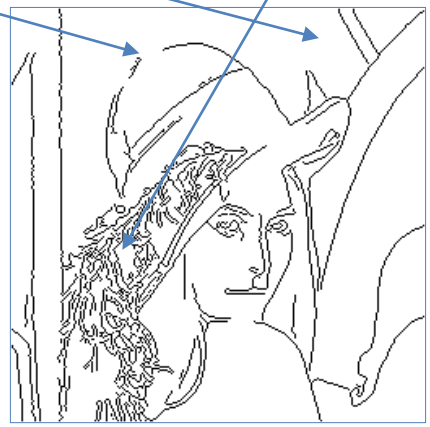
Définition

- Cas simple : contours nets et quasi fermés, rares contours superflus





- Cas complexe : contours ouverts, non définis, discontinus



Filtres de Sobel

- Deux filtres dérivateurs Sx et Sy dans les deux dimensions

moyennage (lissage) dans la direction orthogonale à la direction de dérivation

Filtres de Sobel

$$S_x = \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$$

dérivateur horizontal

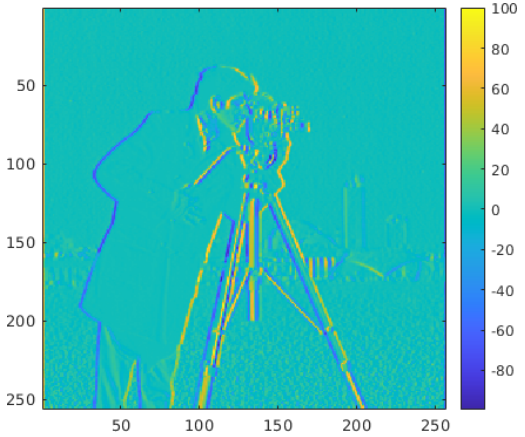
$$S_y = S_x^{\,T}$$

dérivateur vertical

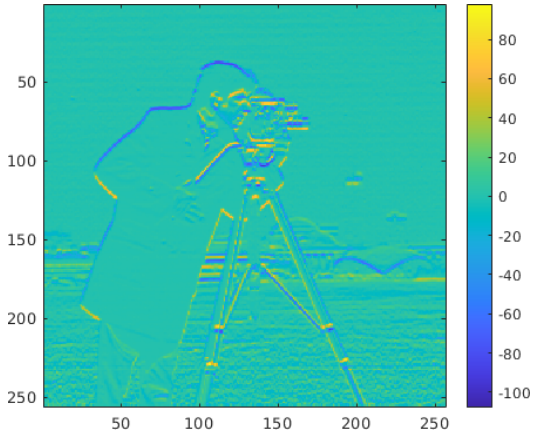
Filtres de Sobel

- Combinaison des deux détections puis seuillage pour obtenir une carte binaire de détection de contours

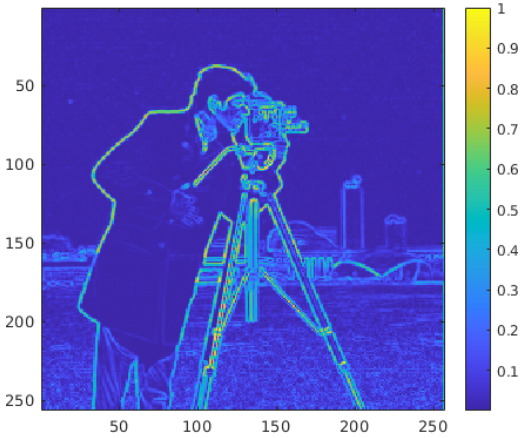
Image initiale



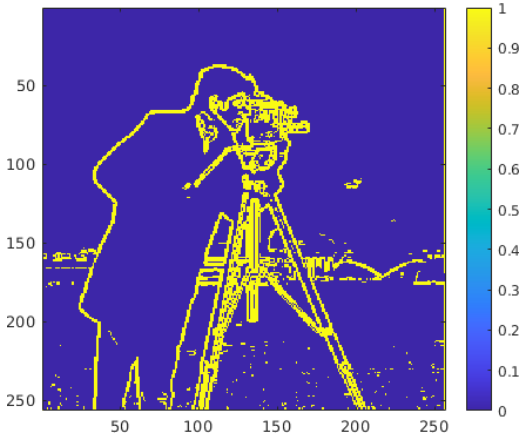
Carte de détection en X



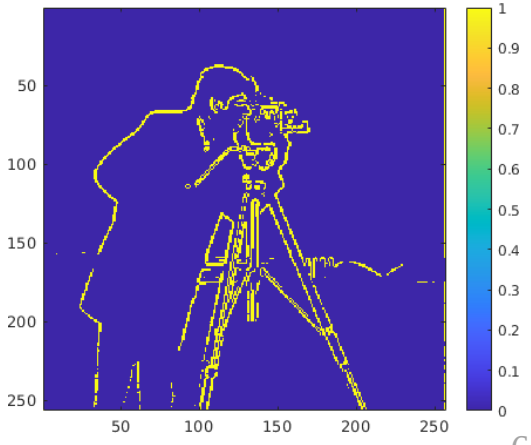
Carte de détection en Y



Carte globale normalisée



Carte seuillée (seuil=0.25)



Carte seuillée (seul=0.4)

Filtres de Sobel

- Créer les deux filtres de Sobel :

Filtres de Sobel

$$S_x = \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$$ dérivateur horizontal

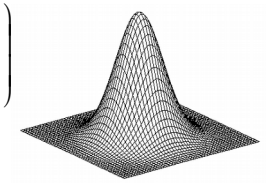
$$S_y = S_x^{\,T}$$ dérivateur vertical

- Convoluer l'image *cameraman.tif* avec ces deux filtres et observer les deux réponses.
- Calculer la norme 2 de ces deux cartes pour obtenir une unique carte d'intensité de contours.
- Normaliser cette carte entre 0 et 1.
- Seuiller cette carte pour obtenir une détection binaire des contours.

Approche de Canny

Principe : Les dérivées (d'ordre n) peuvent être approchées par la convolution avec les dérivées (d'ordre n) d'une gaussienne

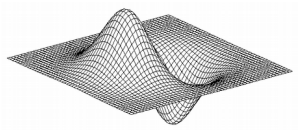
Ordre 1 $\nabla I \approx I * \nabla G$ avec $G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}$



lissage et dérivation

paramètres d'échelle contrôlant la force du lissage et de la dérivation

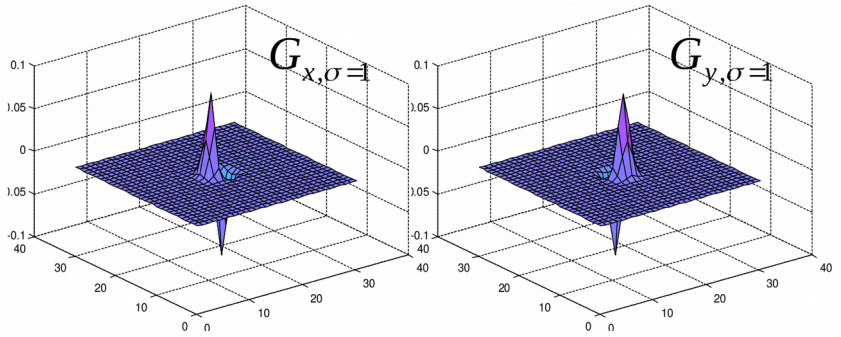
$$\nabla G = \begin{bmatrix} G_x(x,y) = -\frac{x}{2\pi\sigma_x^3\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)} \\ G_y(x,y) = -\frac{y}{2\pi\sigma_x\sigma_y^3} e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)} \end{bmatrix}$$

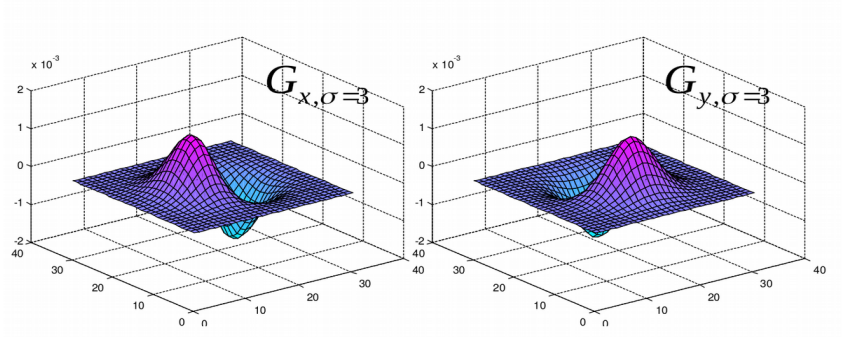


Filtres 2D RIF par échantillonnage et troncature

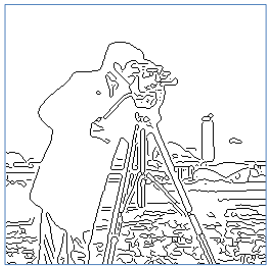
Approche de Canny

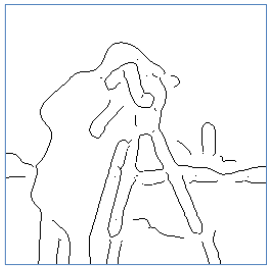
- Compromis entre sur et sous-détection





Norme du gradient
à petite échelle





Norme du gradient
à grande échelle

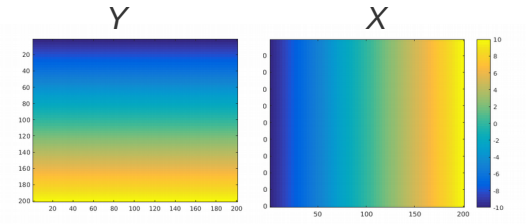
Approche de Canny

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

- Créer un filtre issu d'une dérivée de Gaussienne 2D :

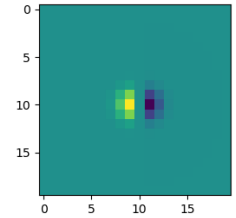
  Utiliser meshgrid pour créer un pavage d'indices :

```
P = range(-10,10)
X, Y = np.meshgrid(P,P)
```



  On retrouve l'expression des dérivées en x et y de g pour leur fournir les cartes du meshgrid et obtenir deux filtres détecteurs de contours Gx Gy :

```
sig = 1
Gx = -X/(2*np.pi*sig**4)*np.exp(-(X**2+Y**2)/(2*sig**2))
```



- Convoluer les filtres avec l'image pour obtenir les deux réponses et calculer leur norme pour obtenir une carte de détection de contours.
- Normaliser la carte entre 0 et 1.
- Utiliser cette carte dans l'application Pencil Sketch. Comparer avec le résultat de edge.

Détection de contraste

$$\Delta I = I_{xx} + I_{yy}$$

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

Filtre 1D RIF horizontal

Filtre 1D RIF vertical

$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

somme

Filtre 2D RIF

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Forme en croix (4 voisins)

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Forme diagonale (4 voisins)

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Forme cumulée « isotrope » (8 voisins)

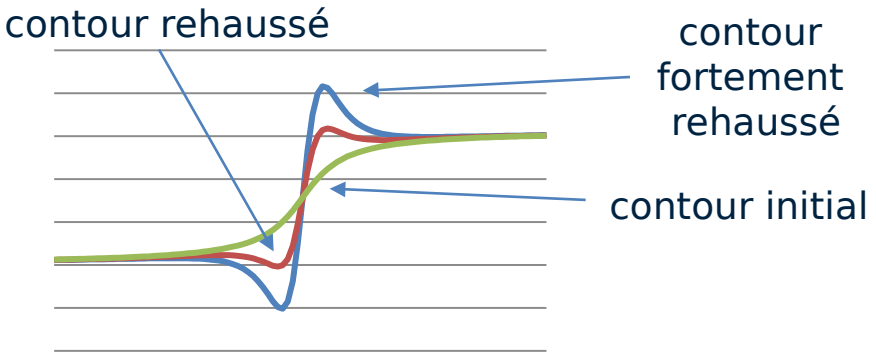
Rehaussement de contraste

- Équivalent au mode « Netteté » sur les smartphones.

$$I_s = I_s - \beta \Delta I$$

paramètre de contrôle

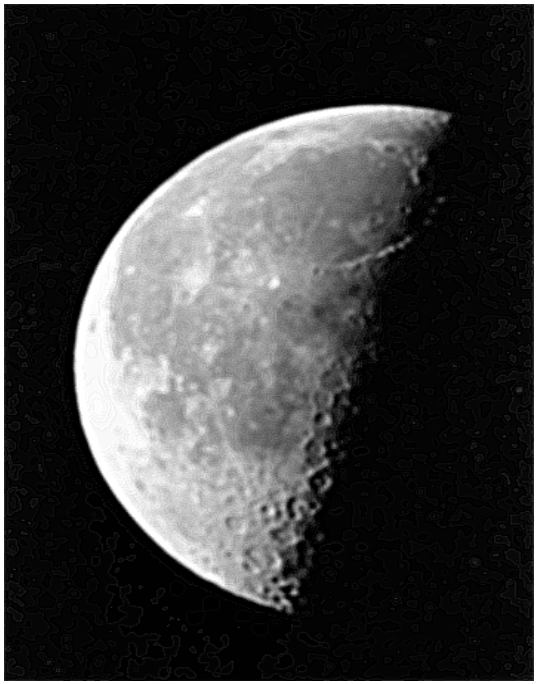
convolution de I avec la forme cumulée



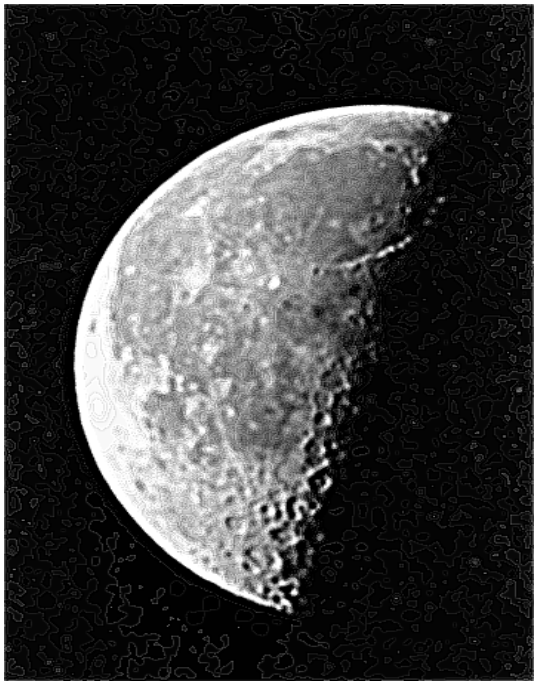
Rehaussement de contraste

- Implémenter le rehaussement en utilisant la version isotrope du filtre.
- Tester sur les images *moon.png*, *cat.jpg*.

Image initiale



Rehaussement moyen



Rehaussement fort

ANNEXES

Principe :

Moyenne pondérée des voisins selon leur proximité spatiale et couleur :

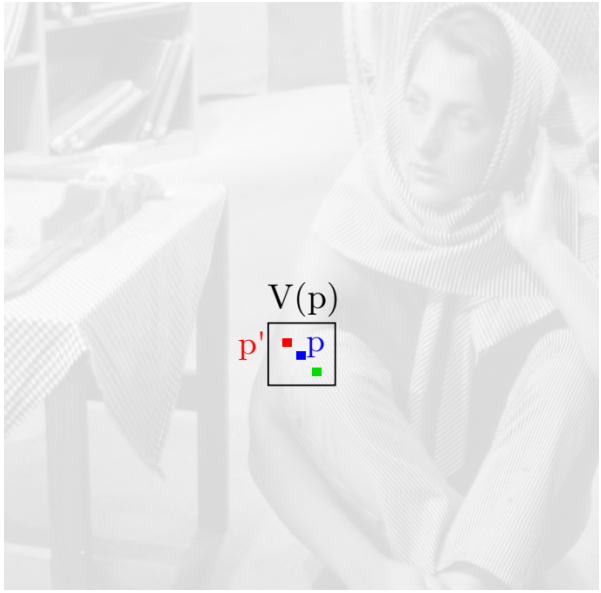
- **Spatial** : la distance des voisins influence la moyenne (= filtre Gaussien).
  Plus un pixel est proche spatialement, plus il contribue.
- **Couleur** : la différence en intensité des voisins influence la moyenne.
  Plus un pixel est de même intensité/couleur, plus il contribue.
  → Préservation des contours

$$I_F(p) = \frac{\sum_{p' \in V(p)} \omega(p,p') I(p')}{\sum_{p' \in V(p)} \omega(p,p')}$$

$$\omega(p,p') = \exp\left(-\frac{\|I(p) - I(p')\|_2}{2\sigma_c^2} - \frac{(p-p')^2}{2\sigma_s^2} \right)$$

Pondération couleur

Pondération spatiale



Résultats

Image initiale

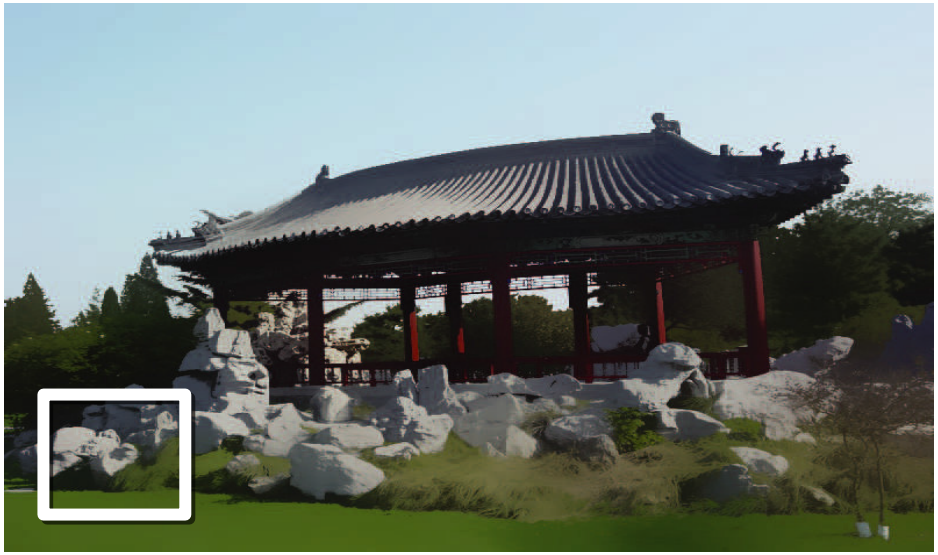


Image filtrée

Résultats

Image initiale

Image filtrée