Université Constantine 2 -Abdelhamid Mehri
Faculté des **N**ouvelles **T**echnologies de l'**I**nformation et de la **C**ommunication
Département **I**nformatique **F**ondamentale et ses **A**pplications

Master Sciences et **T**echnologies de l'**I**nformation et de la **C**ommunication **STIC**

Matière : **Imagerie et Vision Artificielle ImVA**

Chapitre 2 : Opérations sur les images

1- Échantillonnage et quantification d' image

Numérisation

image réelle

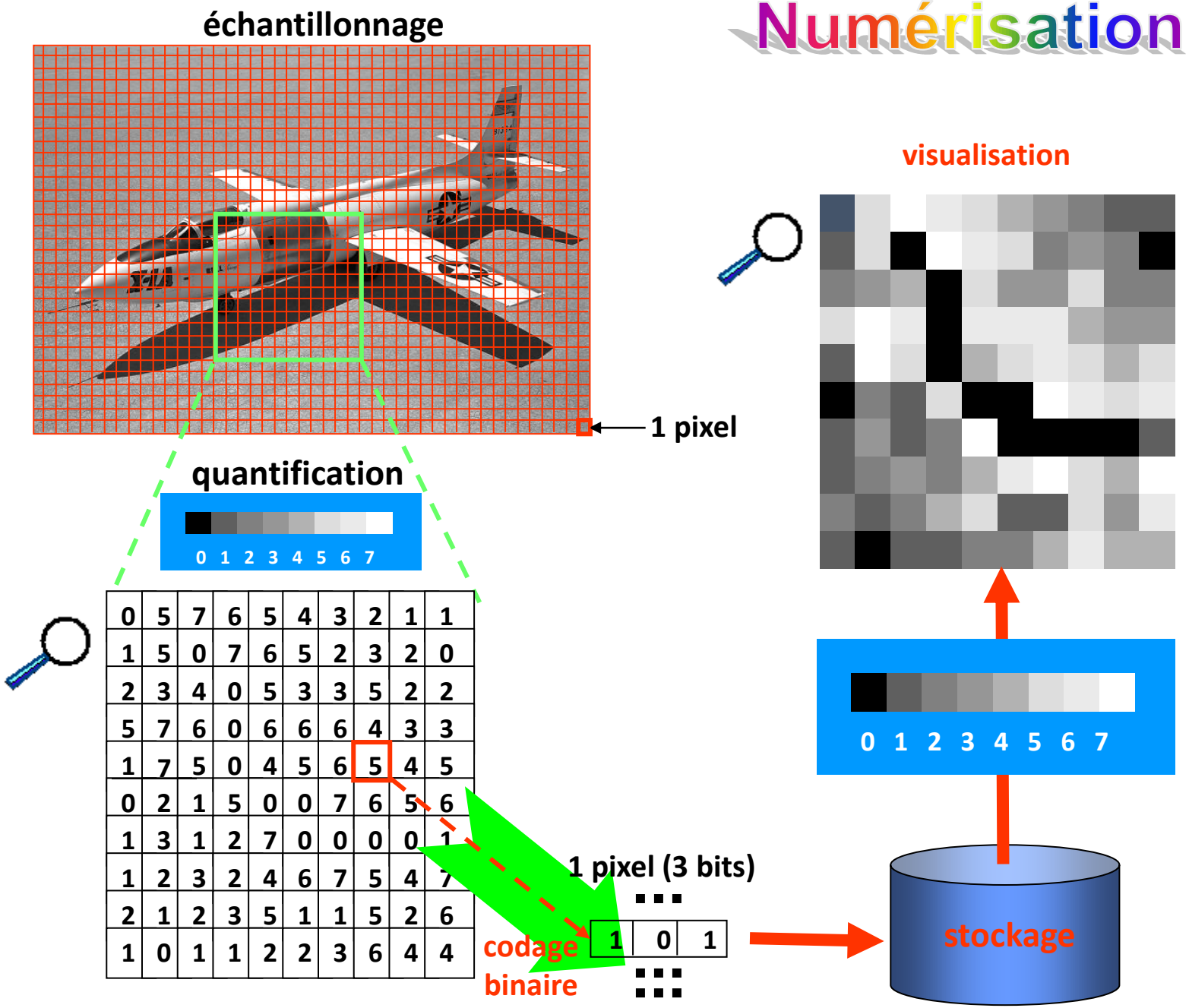
scanner

caméra numérique

numériseurs

Numérisation = Échantillonnage + Quantification

Numérisation



1- Échantillonnage et quantification d’ image

- **L’échantillonage** consiste à représenter l'image par un nombre fini de points. Il peut être vu comme une superposition à l'image une grille régulière formé de petits éléments chacun d'eux va présentera un point de l'image discrète appelé pixel (PICture ELement). L'échantillonnage spatial détermine la taille de chaque point élémentaire de l'image (pixel).). Cette taille est fonction de la résolution du capteur
- **la Quantification** d'une image consiste, pour chaque pixel, à lui associer une valeur discrète d'amplitude. Cette valeur de l'amplitude s'exprime en « bit » et l'action de transformer la valeur numérique de l'amplitude en valeur binaire s'appelle le codage.
- L'intensité I est quantifié sur m bits et peut prendre $L = 2^m$ valeurs : $I \in [0, 2^m-1]$
- Exemple m = 1 bit : 2 valeurs possibles (images binaires)

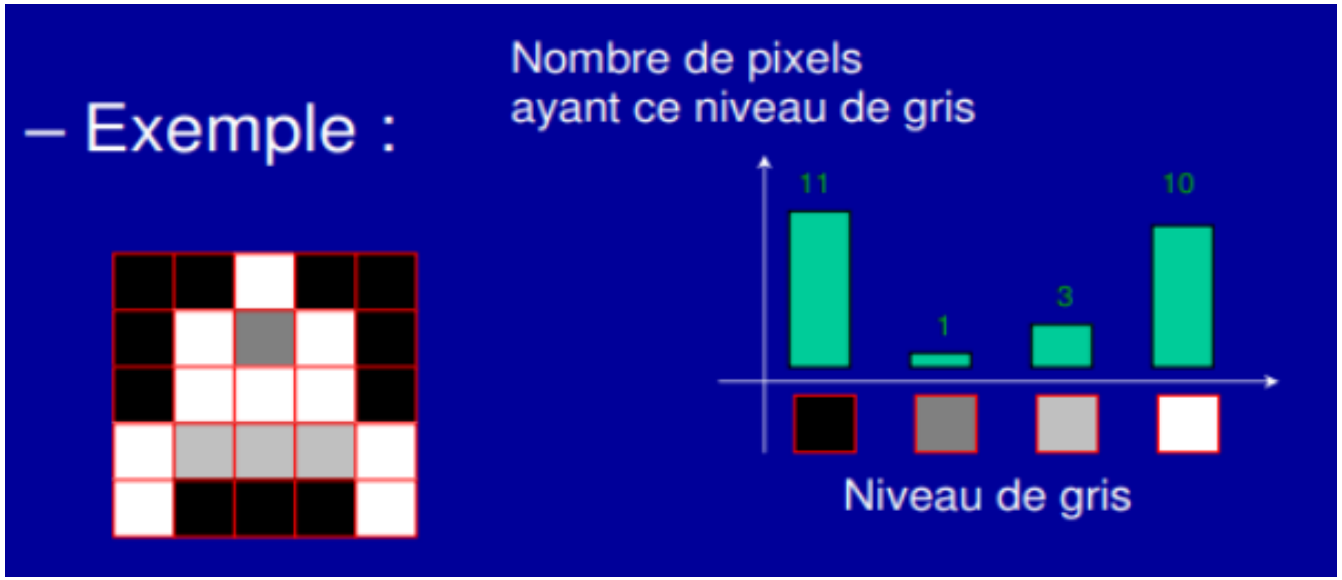
  m = 8 bits: 256 valeurs possibles (images en niveaux de gris)

  m = 16 bits: 65535 valeurs possibles (images en couleurs)

2- Histogramme d’une image

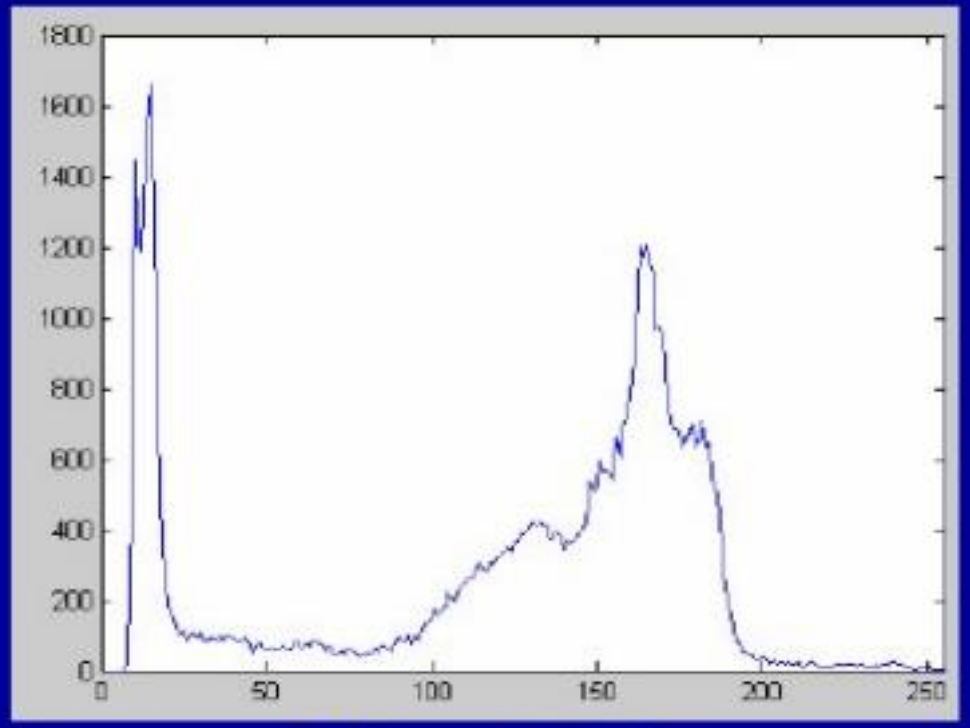
2.1-Définition

- Un **histogramme** est la représentation graphique de la répartition statistique de la luminosité des pixels dans l’image numérique.
- En abscisse, les valeurs proches de 0 indiquent des pixels sombres et celles se rapprochant de 255 les pixels les plus clairs
- En ordonné on trouve le nombre de pixels pour chaque valeur d’intensité lumineuse



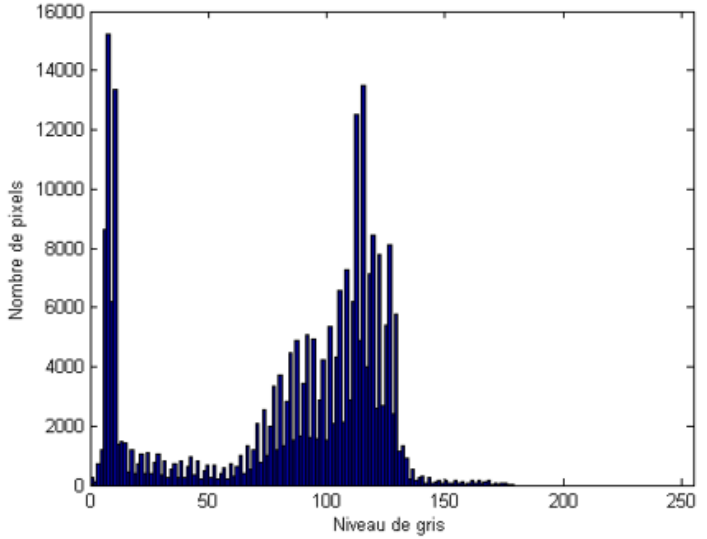
2.1-Définition

Histogramme

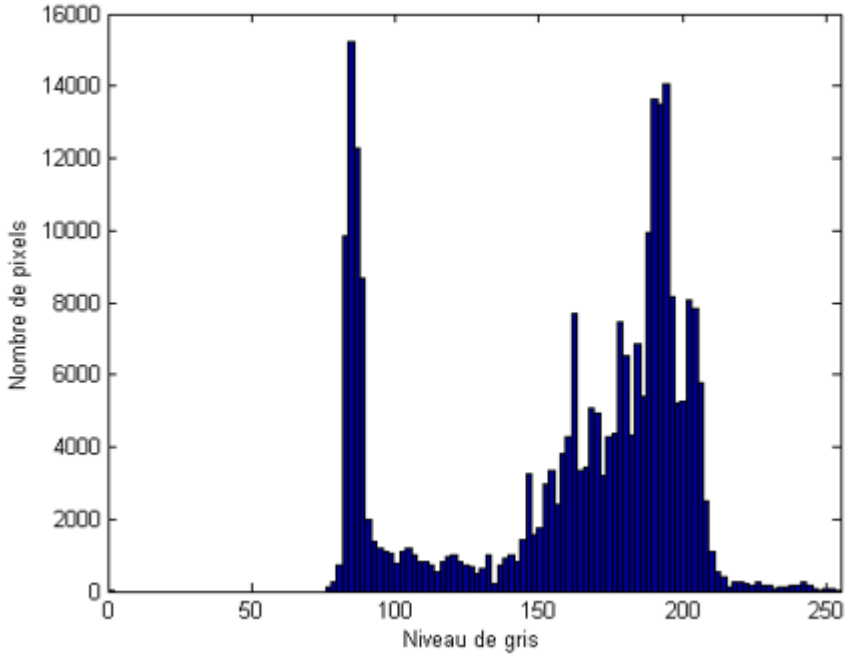


2.1-Définition

- En divisant chaque valeur de l'histogramme par le nombre total de pixels dans l'image on obtient un *histogramme normalisé*. L'histogramme normalisé correspond à une distribution de probabilité empirique (toutes les valeurs sont comprises entre 0 et 1 et la somme des valeurs vaut 1).
- L'histogramme permet d'obtenir rapidement une information générale sur l'apparence de l'image. Une image *visuellement plaisante* aura généralement un histogramme équilibré (proche d'une fonction plate). Par exemple dans l'image ci-dessous, l'histogramme est tassé sur la gauche; l'image est trop sombre :

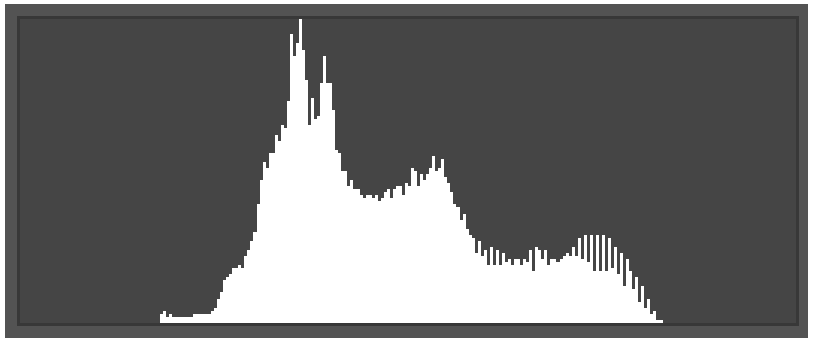


2.1-Définition

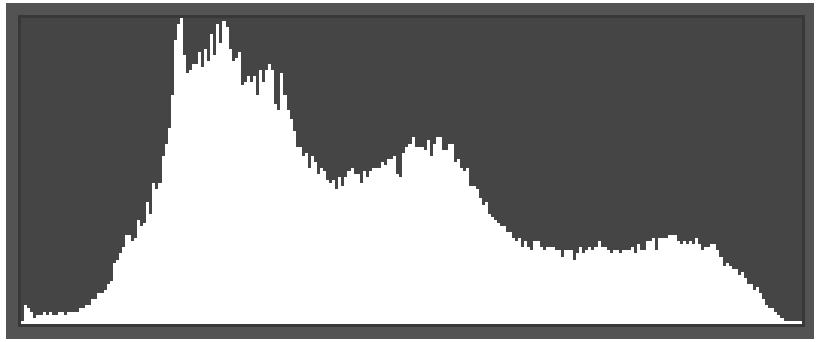


2.2 Exemples d’histogrammes

- histogramme tassé au centre; l’image est grisâtre et manque de contraste :

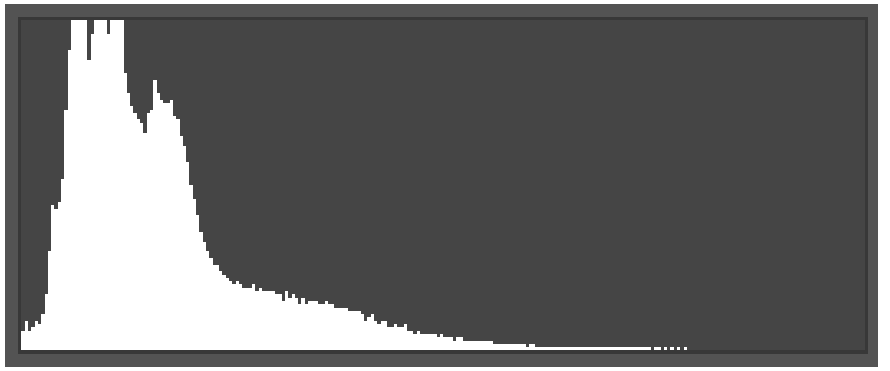


- Histogramme étendu : s’étale sur toute la plage de luminosité

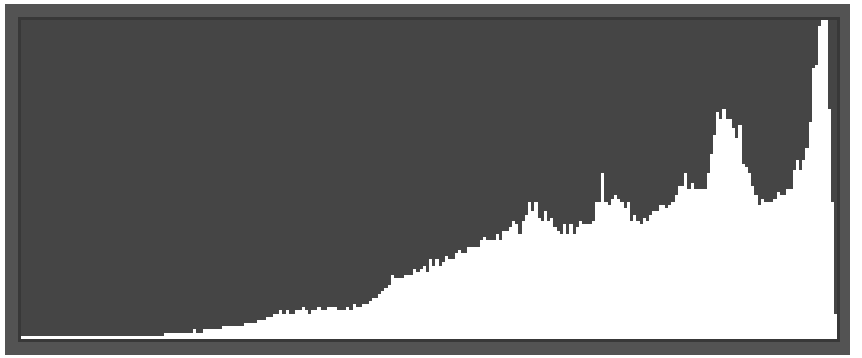


2.2 Exemples d’histogrammes

- histogramme tassé à gauche; **informe d’une photo très sombre**



- Histogramme tassé à droite : image trop lumineuse



2.3 Manipulations d'histogrammes : Étirement

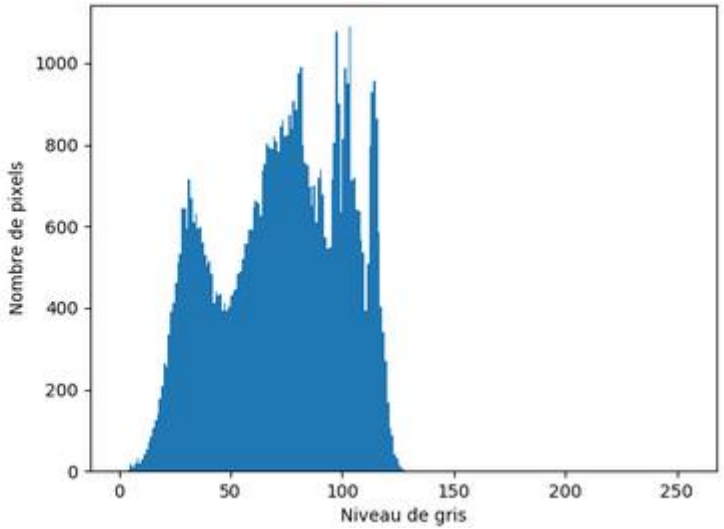
- Lorsque l'histogramme est normalisé, il indique en ordonnée la probabilité p_i de trouver un pixel de niveau de gris i dans l'image :

$\forall i \in \{0,...,255\}$, p_i=nombre de pixels d'intensité i /nombre total de pixels

- L'intensité d'un pixel est alors vue comme une variable aléatoire discrète.

Étirement d'histogrammes

Une première application consiste à corriger la luminosité, ou **exposition**, de l'image



2.3- Manipulations d'histogrammes

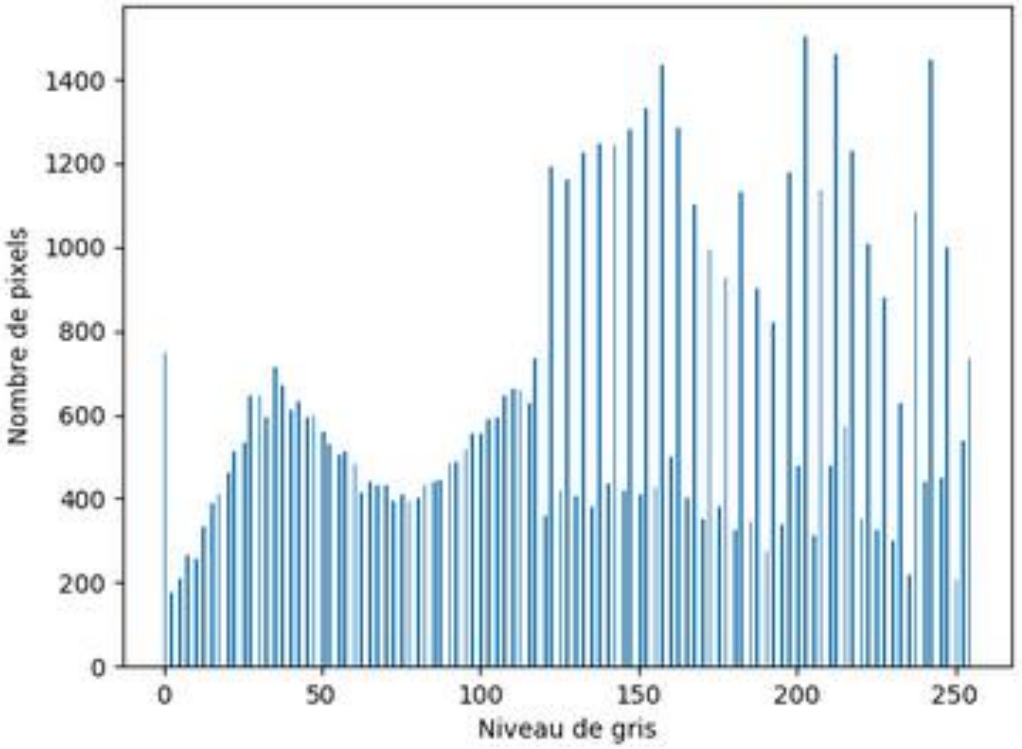
pour corriger les défauts liés à l’exposition d'une image,
il suffit simplement d'étirer son histogramme :
l'objectif est d'étendre les valeurs des niveaux de gris de l'image mal exposée,
majoritairement répartis dans un sous intervalle $[I_{min},I_{max}]\subset[0,255]$
à tout l'intervalle disponible.

$$I'(x,y) = \frac{255 \times (I(x,y) - I_{min})}{I_{max} - I_{min}}$$

où I(x,y) et I’(x,y) désignent les intensités du pixel de coordonnées (x,y) respectivement dans l'image mal exposée et la nouvelle image.

2.3- Manipulations d'histogrammes : Étirement

Notre image après étirement et son histogramme



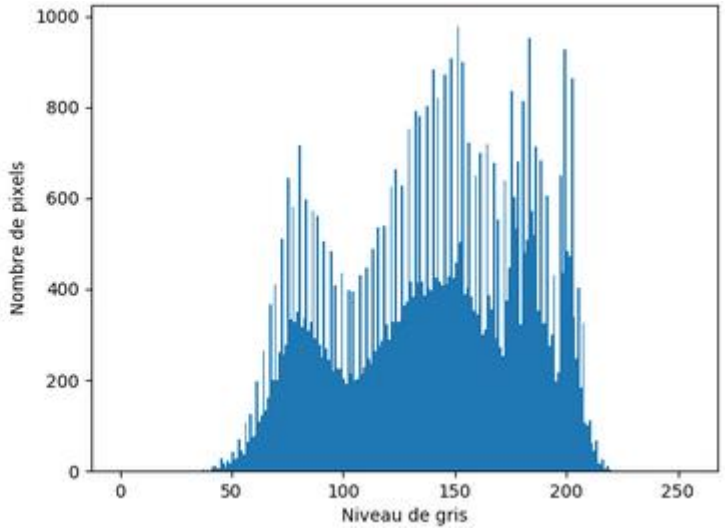
Le résultat est satisfaisant : les pixels se répartissent bien dans tout l'intervalle [0,255] et l'image présente une meilleure luminosité

2.3- Manipulations d'histogrammes : Égalisation

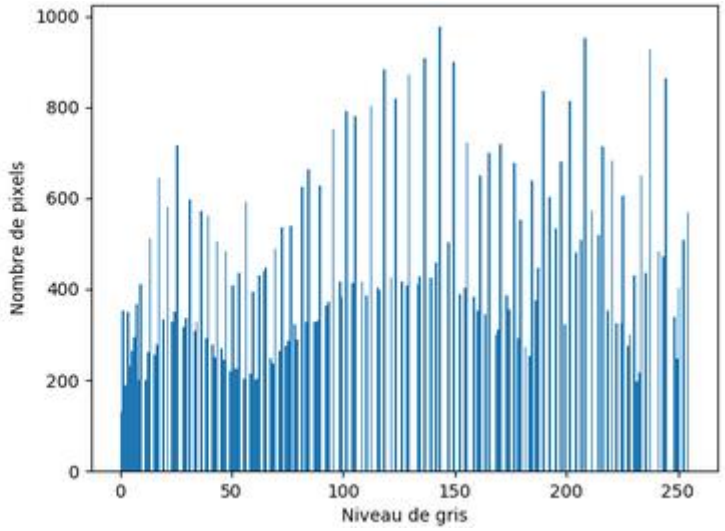
- La deuxième application courante concerne l'amélioration du contraste de l'image.
- Le contraste caractérise la répartition de lumière dans une image : plus une image est contrastée, plus la différence de luminosité entre ses zones claires et sombres est importante.
- **L'égalisation d'histogrammes** est une technique simple permettant de réajuster le contraste d'une image
- L'objectif est donc d'harmoniser la distribution des niveaux de gris de l'image, de sorte que chaque niveau de l'histogramme contienne idéalement le même nombre de pixels.
- Pour cela, nous calculons d'abord l'histogramme cumulé normalisé de l'image, puis nous ajustons la valeur de chaque pixel en utilisant la formule mathématique :

où pi désigne la probabilité qu'un pixel de l'image initiale soit d'intensité i

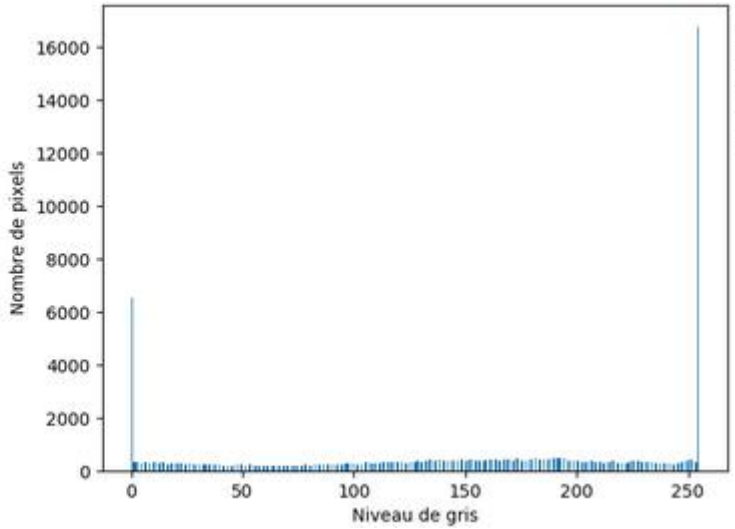
$$I'(x,y) = \lfloor 255. \sum_{i=0}^{I(x,y)} p_i \rfloor$$



Notre image avec peu de contraste et son histogramme



Notre image peu contrastée après égalisation et son histogramme



Notre image avec trop de contraste et son histogramme

2.3- Manipulations d'histogrammes : Égalisation

- Pour améliorer le contraste, on cherche à aplanir l'histogramme

- Etape 1 : Calcul de l'histogramme

$$h(i) \quad i \in [\ 0,\ 255\ ]$$

- Etape 2 : Normalisation de l'histogramme
  (Nbp : nombre de pixels de l'image)

$$h_n(i) = \frac{h(i)}{Nbp} \quad i \in [\ 0,\ 255\ ]$$

- Etape 3 : Densité de probabilité normalisé
  (histogramme cumulé *C(i)*)

$$C(i) = \sum_{j=0}^{i} h_n(j) \quad i \in [\ 0,\ 255\ ]$$

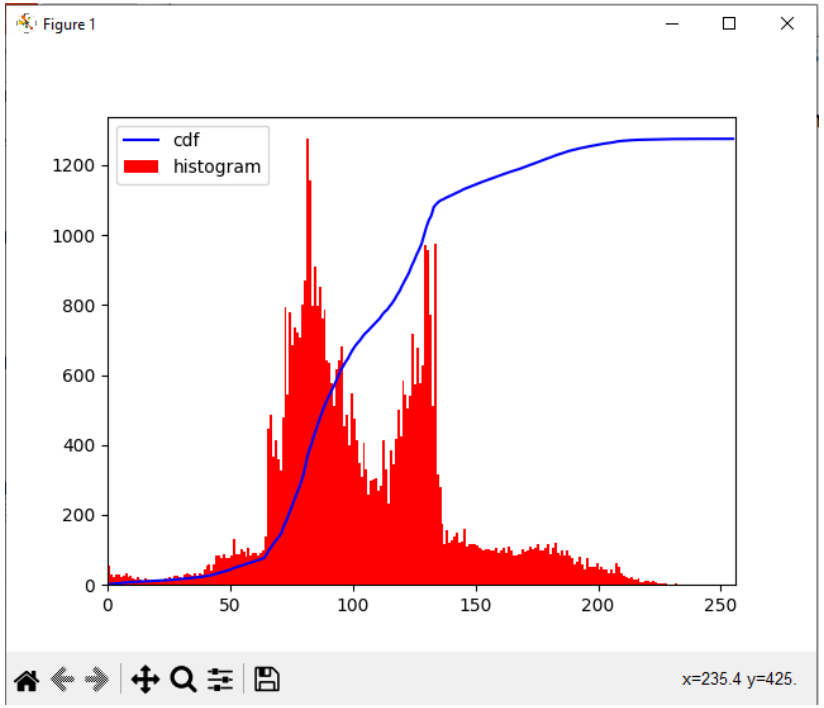
- Etape 4 : Transformation des niveaux de gris de l'image

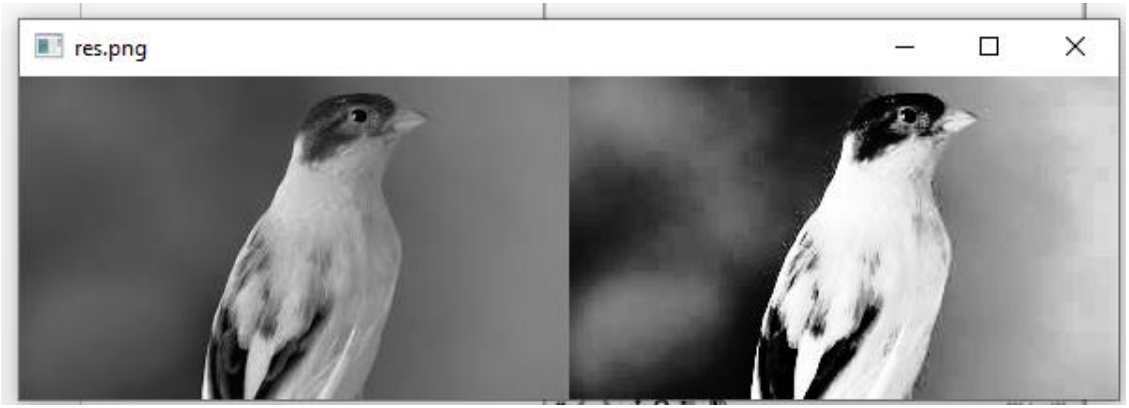
$$f'(x,y) = C(f(x,y)) \times 255$$

Code python

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('C:/Users/intel i7/Desktop/image-ng.jpg', 0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
cv.waitKey(0)
equ = cv.equalizeHist(img)
res = np.hstack((img,equ)) #stacking images side-by-side
cv.imshow('res.png',res)
cv.waitKey(0)
```

Résultat de l'exécution





3- Addition de 2 images

- L’addition de deux images f et g de même taille est une nouvelle image h de même taille dont chaque pixel correspond à la somme des valeurs des pixels des images originales

Image 1 : F(x,y)

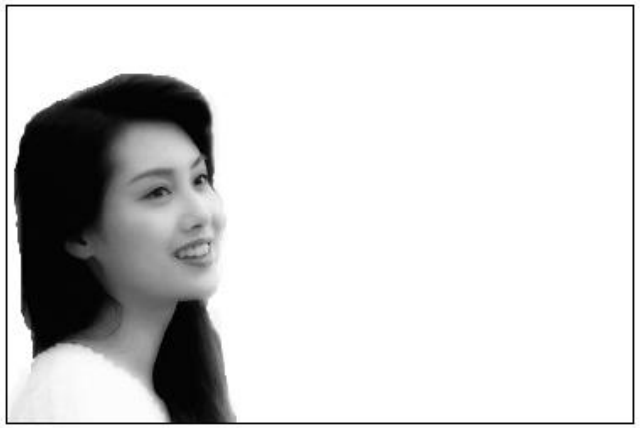


Image 2 : G(x,y)

F(x,y)+G(x,y)

3- Addition de 2 images : OpenCV (superposition ou mélange)

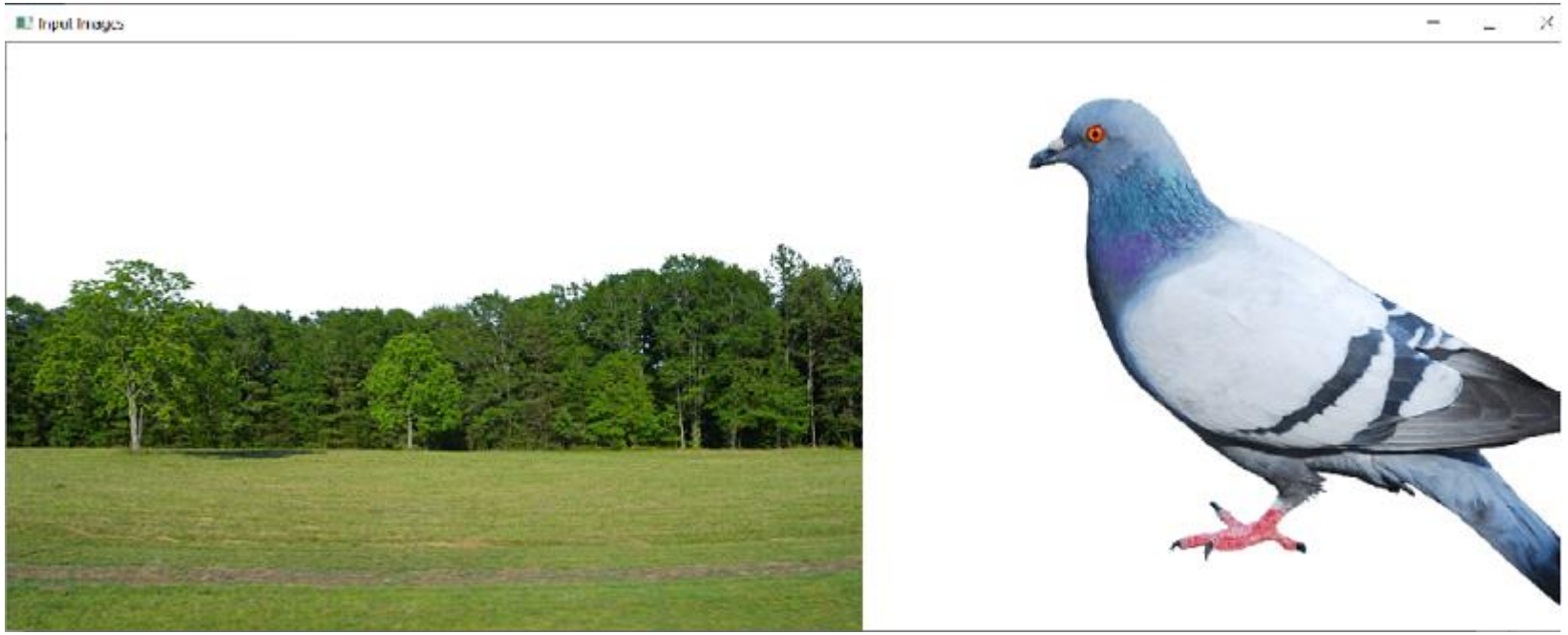
cv.addWeighted(**src1**, **alpha**, **src2**, **beta**, **gamma**[, dst[, dtype]])

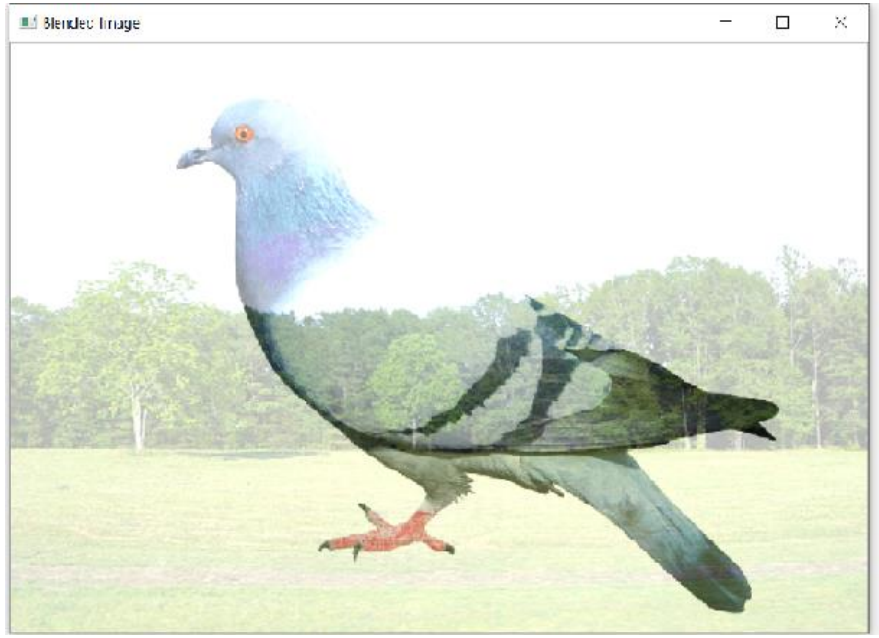
src1: first input array,
alpha: weight of the first array elements,
src2: second input array with the same size and channel number as src1,
beta: weight of the second array elements,
gamma: scalar added to each sum,
dst: output array that has the same size and number of channels as the input arrays,
dtype: optional depth of the output array.

Exemple

```
import cv2
import numpy as np
img1 = cv2.imread('forest.png')
img2 = cv2.imread('pigeon.png')
dst = cv2.addWeighted(img1, 0.5, img2, 0.7, 0)
img_arr = np.hstack((img1, img2))
cv2.imshow('Input Images',img_arr)
cv2.imshow('Blended Image',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output





4- Soustraction de 2 images

image1 - image2

F(x,y) - G(x,y)

image2 - image1

G(x,y) - F(x,y)

4- Soustraction de 2 images OpenCV

```
import cv2
image_one = cv2.imread('_1.jpg')
image_two = cv2.imread('_2.jpg')
result_image = cv2.subtract(image_one, image_two)
cv2.imshow('Final Image', result_image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

5- Le Seuillage

Dans certains cas, l'histogramme peut être un outil efficace pour segmenter une image en deux classes, c'est-à-dire pour distinguer les objets de l'image suivant leur luminosité. En effet, lorsque l'histogramme présente deux modes distincts, on peut définir un seuil (threshold) S entre ces deux modes, et appliquer ensuite un seuillage (thresholding) sur les pixels de l'image :

- si le pixel a une valeur inférieure au seuil, le pixel est dans la classe 0 ;
- si le pixel a une valeur supérieure au seuil, le pixel est dans la classe 1.

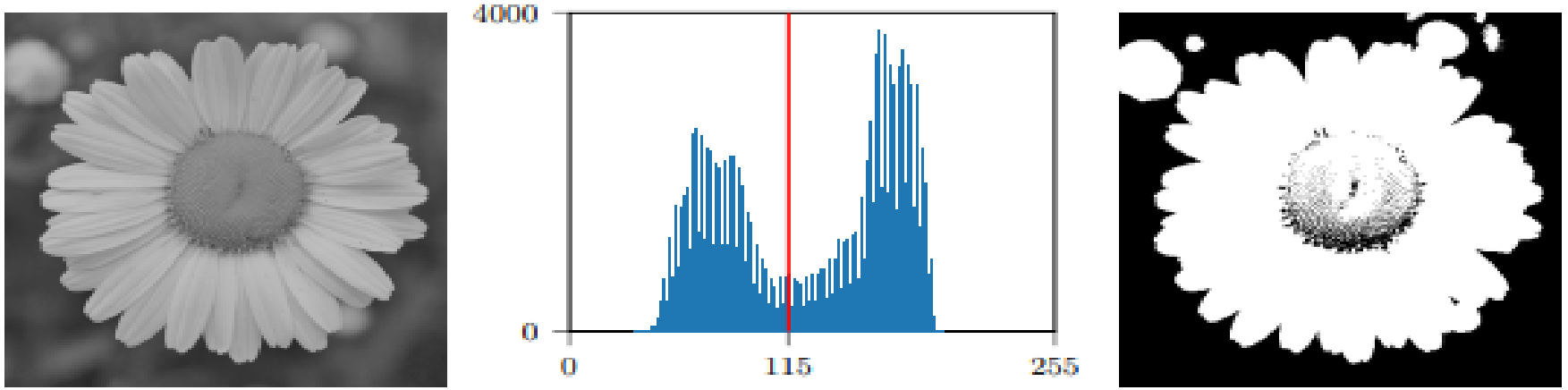


FIGURE 10 – Seuillage avec un seuil égal à 115.

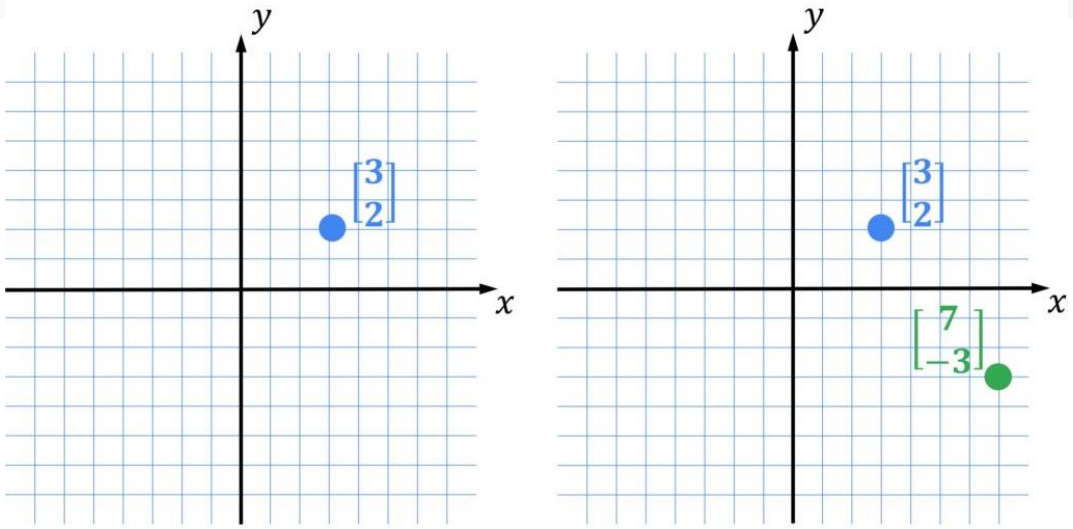
6- Translation d’image

Afin de translater notre image, la première chose que nous devons faire est de créer notre matrice de traduction M.
Cette matrice définira la distance et la direction dans laquelle nous devons décaler les pixels de notre image d'entrée.

Number of pixels we will shift the image left or right

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Number of pixels we will shift the image up or down



$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = M \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} \quad M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3+0+4 \\ 0+2-5 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} 7 \\ -3 \\ 1 \end{bmatrix}$$

6- Translation d’image : OpenCV

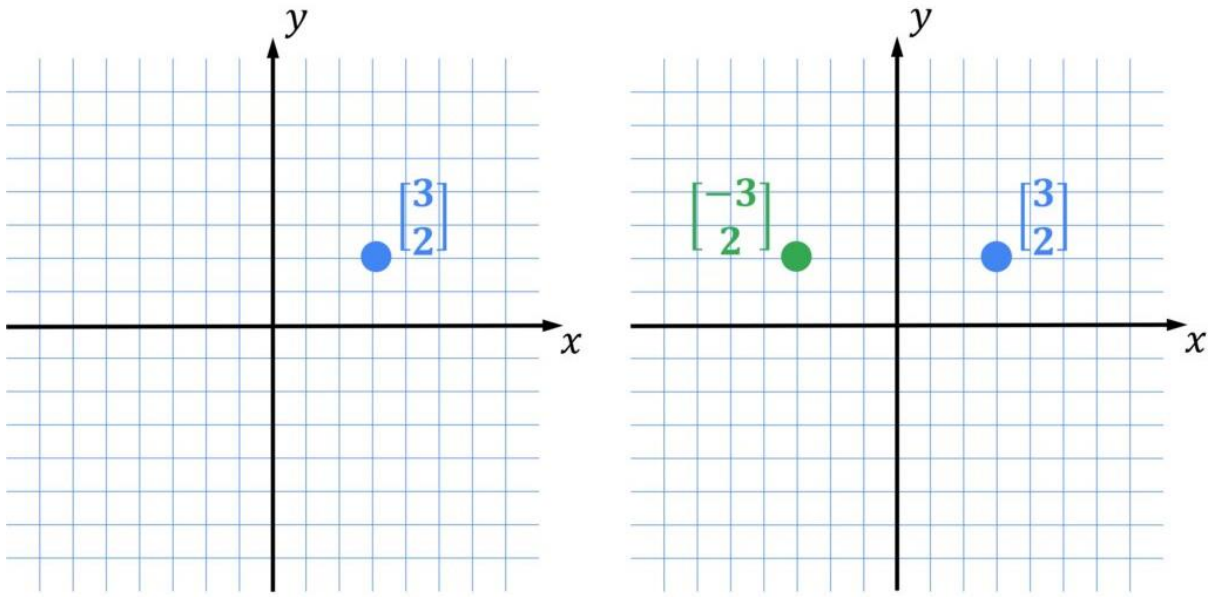
```
import numpy as np
import cv2
img = cv2.imread('messi5.jpg', cv.IMREAD_GRAYSCALE)
rows,cols = img.shape
M = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow('img',dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```python
import cv2 as cv
import numpy as np

img = cv.imread("Images/photography.jpeg")
cv.imshow("Balloons", img)


## Translation
def translation(img, x, y):
    transMAT = np.float32([[1, 0, x], [0, 1, y]])
    dimensions = (img.shape[1], img.shape[0])
    return cv.warpAffine(img, transMAT, dimensions)

translated = translation(img, -100, 100)
cv.imshow("Translated", translated)



cv.waitKey(0)
```

7- Flipping d’image

La matrice que nous utilisons pour effectuer cette opération est appelée matrice de réflexion.

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



$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = M \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} \qquad M = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

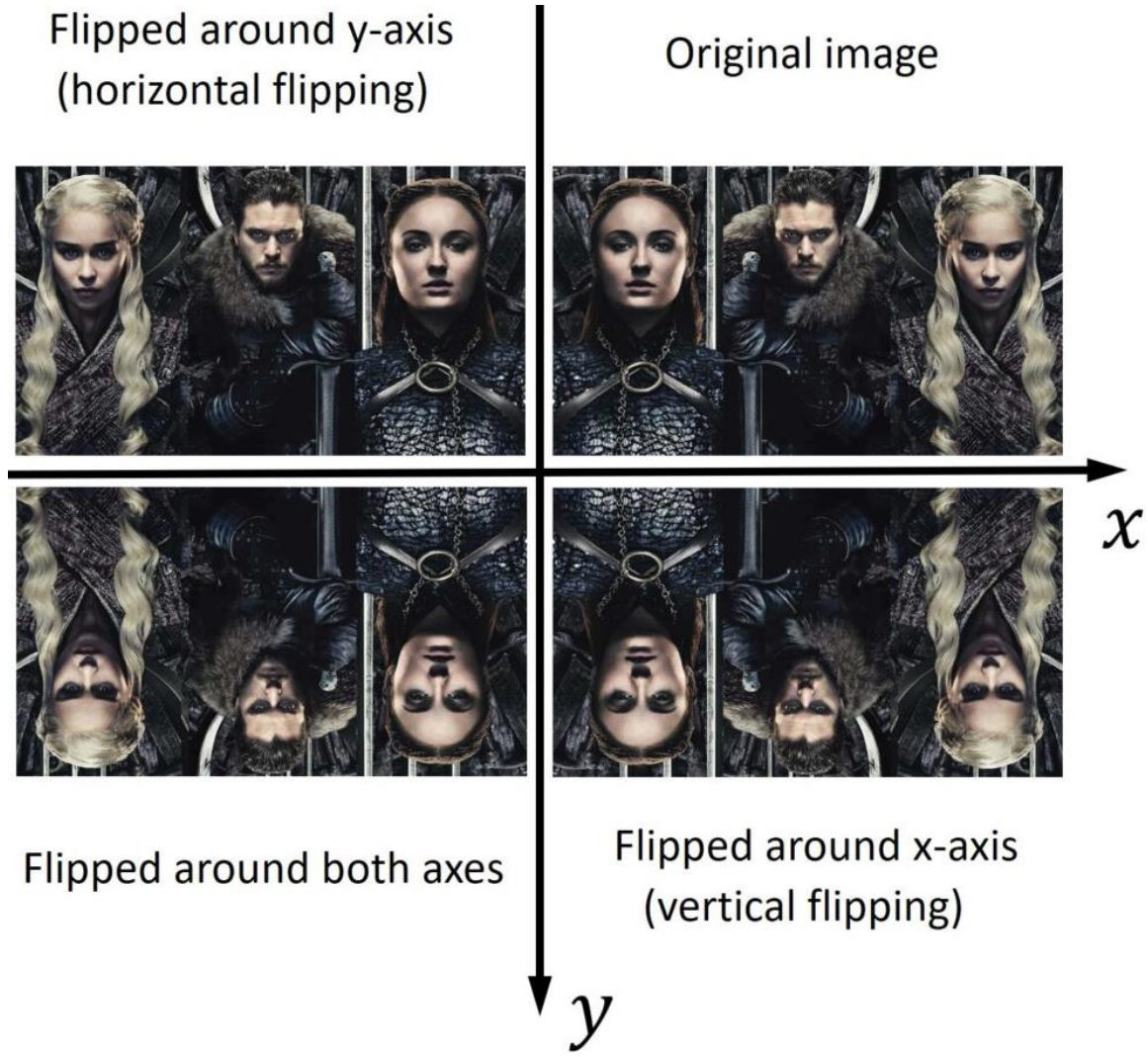
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -3+0+0 \\ 0+2+0 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 1 \end{bmatrix}$$

7- Flipping d’image : OpenCV

```
# Flipping the image around y-axis
flipped = cv2.flip(img, 1)
cv2_imshow(flipped)
```

```
# Flipping the image around x-axis
flipped = cv2.flip(img, 0)
cv2_imshow(flipped)
```

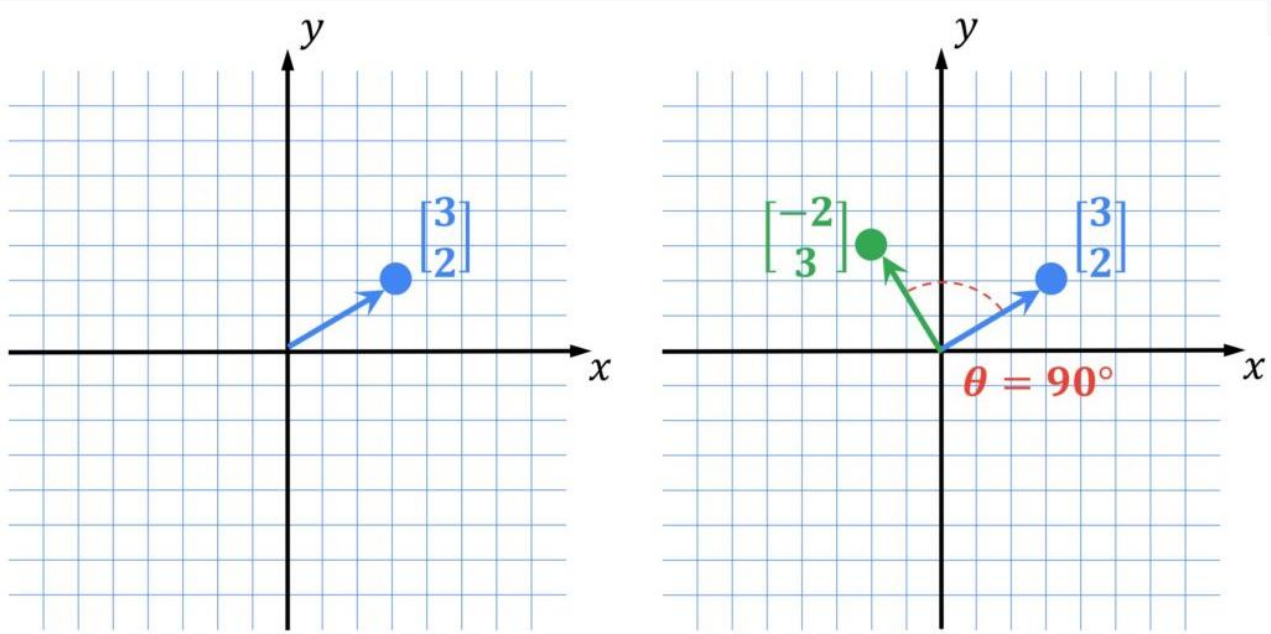
```
# Flipping the image around both axes
flipped = cv2.flip(img, -1)
cv2_imshow(flipped)
```



8- Rotation d’image

Afin de réaliser la rotation de notre image, nous devons calculer la matrice de rotation, avec θ est l’angle de rotation,

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



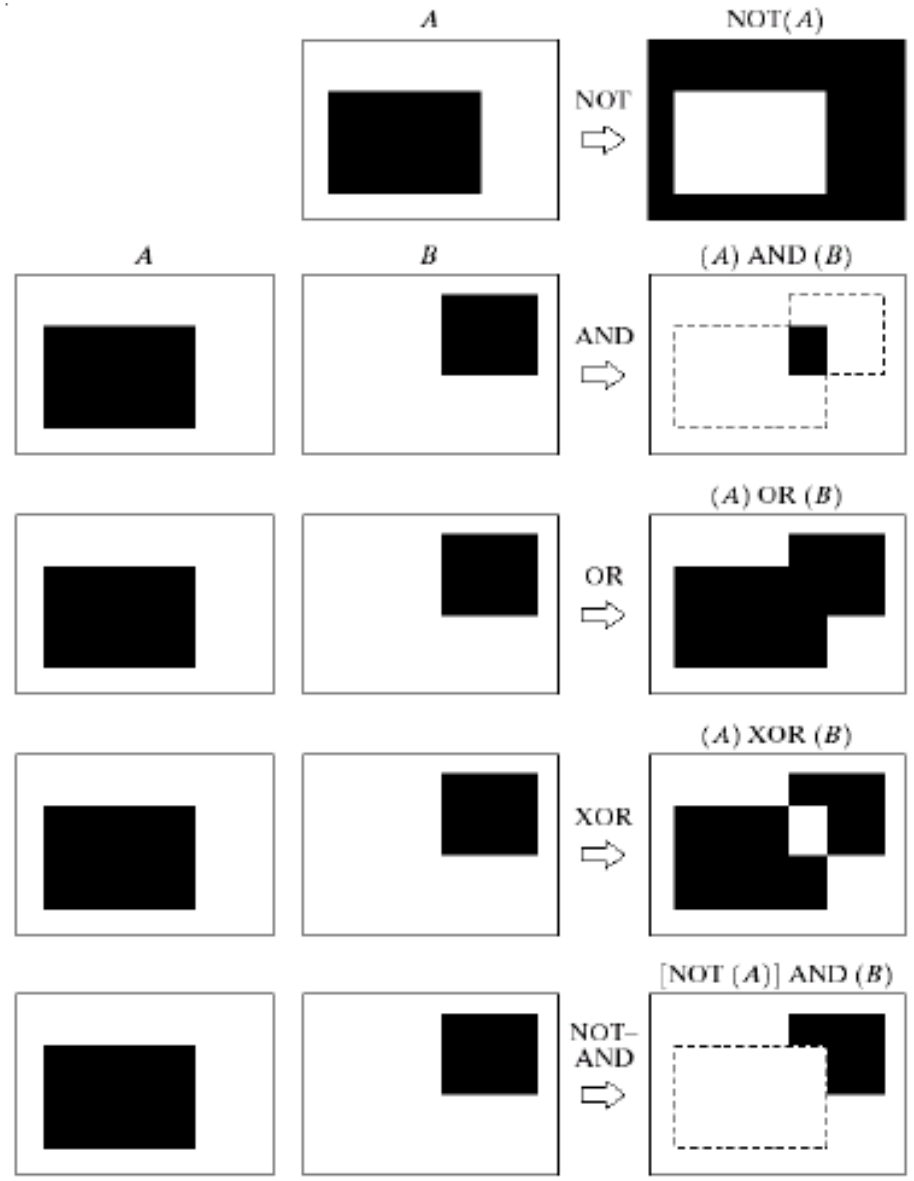
$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = M \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} \quad M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0-2+0 \\ 3+0+0 \\ 0+0+1 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \\ 1 \end{bmatrix}$$

8- Rotation d’image : OpenCV

```
import cv2
import numpy as np
img = cv2.imread('images/input.jpg')
num_rows, num_cols = img.shape[:2]
rotation_matrix = cv2.getRotationMatrix2D((num_cols/2, num_rows/2), 30, 1)
img_rotation = cv2.warpAffine(img, rotation_matrix, (num_cols, num_rows))
cv2.imshow('Rotation', img_rotation)
cv2.waitKey()
```

9- Opérations Logiques



ET logique

- Pour les images en niveaux de gris, l'opérateur logique est appliqué sur la représentation binaire des niveaux de gris, en comparant les bits correspondants.
- 2 niveaux de gris 47 et 252 codés sur 8 bits,

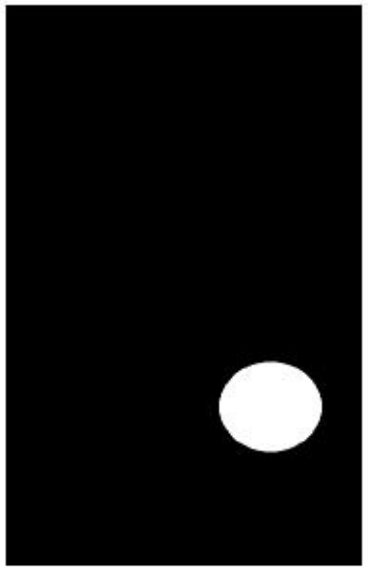
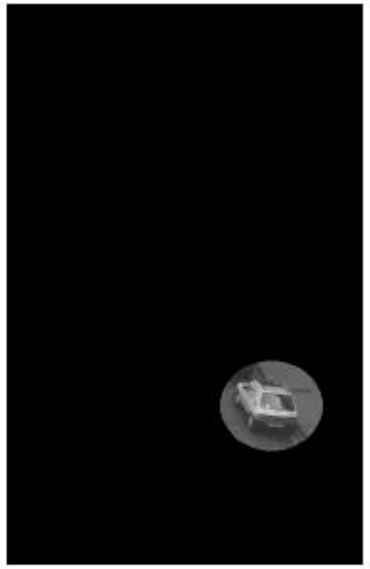
  47 en binaire 00101111

  252 en binaire 11111100

  Résultat 00101100 soit 44

9- Opérations Logiques

ET logique


Utilisation : Masque pour isoler une région

10- Filtrage des images

Filtrage : modification des valeurs de l'image par application d'un opérateur.

Objectifs :

- améliorer la qualité d'une image
- atténuer, supprimer les dégradations (bruit)
- rehausser des contours
- calculer certaines caractéristiques de l'image (gradient, laplacien)

Ces dégradations apparaissent :

- au moment de l'acquisition
- liées à la transmission
- changement de format ou au stockage

Objectif : Fidélité de l'image restaurée (filtrée) avec la scène qu'elle représente

⇒modéliser la dégradation

⇒Connaître les sources de dégradations

10- Filtrage des images

Contexte d'acquisition

- Sur ou sous illumination
- Perturbations des capteurs (mouvement)

Capteur

- Distorsions (géométriques, d'intensité)

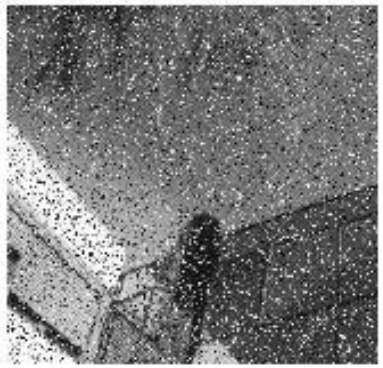
Echantillonnage

- Objet dont la taille est égale au pixel : bruit de poivre et sel

Nature de la scène

- Nuages en imagerie satellitaire
- Images médicales

Image originale



Bruit poivre et sel

Modélisation

Le filtrage de l'image I vise à réduire le bruit et à trouver l'image idéale I' qui aurait été obtenue avec un système d'acquisition parfait.

Le bruit b(x,y) est souvent considéré comme aléatoire

- Bruit additif : $I(x,y)=I'(x,y)+b(x,y)$
- Bruit multiplicatif : $I(x,y)=I'(x,y).b(x,y)$

Bruit Gaussien additif

Bruit multiplicatif

10- Filtrage des images : Bruit de poivre et sel

Un bruit « poivre et sel » d'ordre n est obtenu en ajoutant n pixels blancs et n pixels noirs aléatoirement dans une image.

Originale

5%

15%

30%

10- Filtrage des images : Bruit gaussien

un **bruit gaussien** est un bruit dont la densité de probabilité est une distribution gaussienne (loi normale).

La densité de probabilité p d'une variable aléatoire gaussienne z est la fonction :

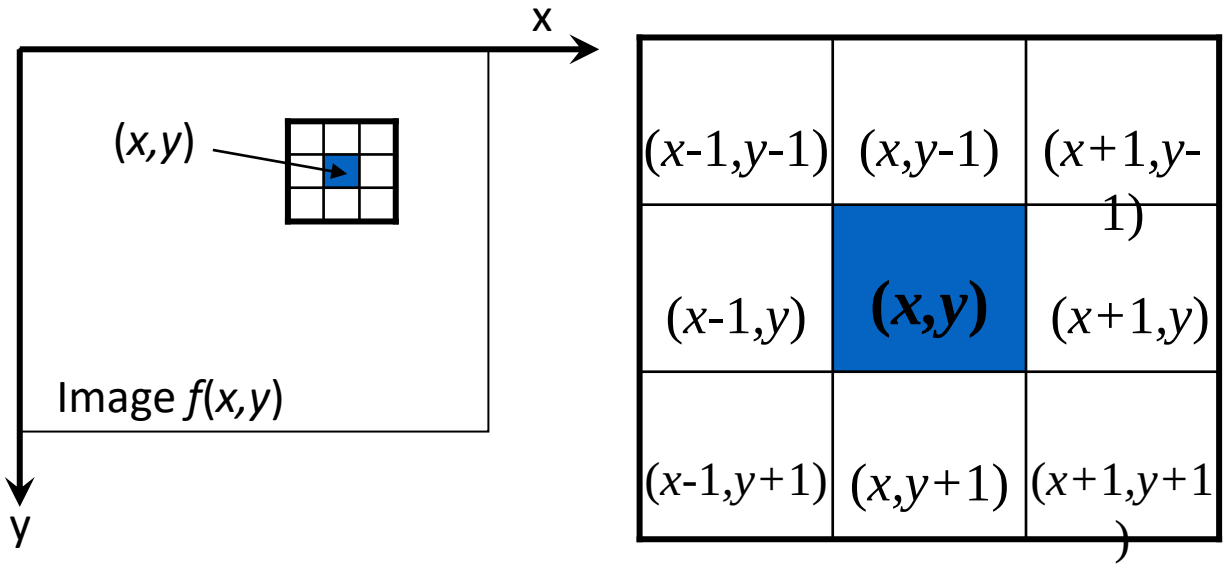
$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(z-\mu)^2}{2\sigma^2}\right]$$

où z représente le niveau de gris, μ la valeur de gris moyenne et σ son écart type

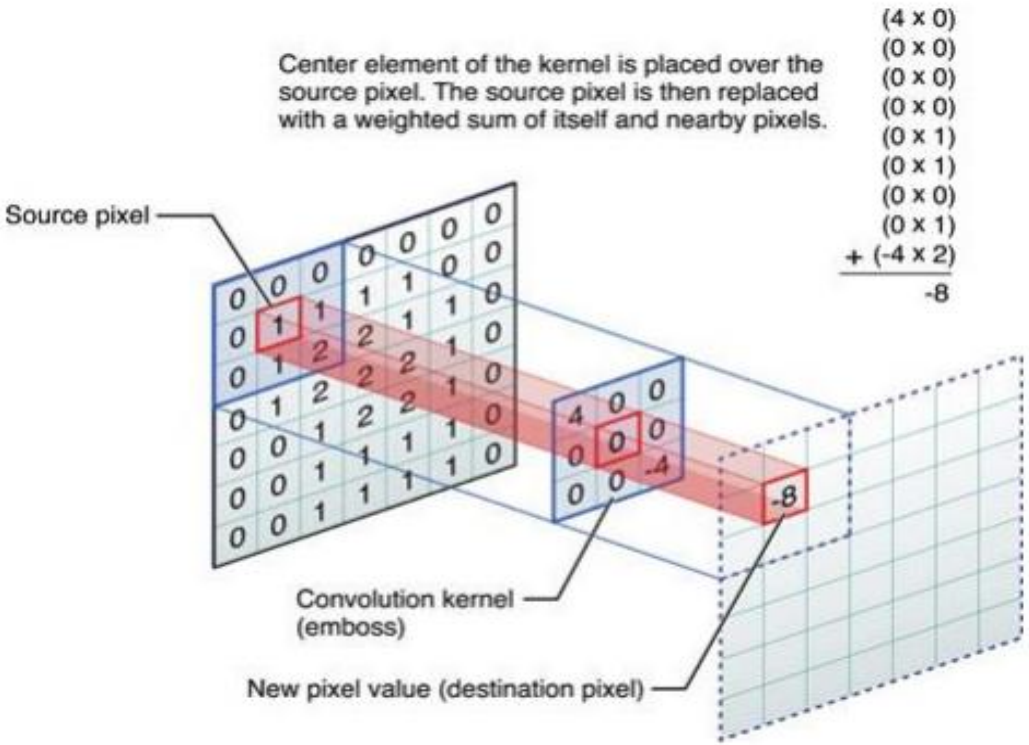
Originale | $\sigma = 20$ | $\sigma = 40$ | $\sigma = 60$

10- Filtrage des images : Convolution

- La convolution consiste en une opération de multiplication de deux matrices de tailles différentes (généralement une petite et une grande), mais de même dimensionnalité semblable (p.ex. 1D, 2D), produisant une nouvelle matrice (également de même dimensionnalité).
- La convolution est donc le traitement d'une matrice (p.ex. une image) par une autre petite matrice appelée matrice de convolution ou noyau (kernel).
- Le filtre parcourt toute la matrice principale (p.ex. l'image) de manière incrémentale et génère une nouvelle matrice constituée des résultats de la multiplication



$(x-1,y-1)$	$(x,y-1)$	$(x+1,y-1)$
$(x-1,y)$	(x,y)	$(x+1,y)$
$(x-1,y+1)$	$(x,y+1)$	$(x+1,y+1)$



10- Filtrage des images : Filtrage Linéaire

- Le filtre est dit linéaire si la valeur du nouveau pixel est une combinaison linéaire des valeurs des pixels du voisinage.
- Le voisinage est une matrice carrée et symétrique autour du pixel considéré. En fonction du filtre utilisé, on pourra obtenir différents effets visuels.
- On classe habituellement les filtres linéaires en 2 familles :

1- **Les filtres passe-bas :** Les filtres lisseurs ou **passe-bas** consistant à atténuer les composantes de l'image ayant une fréquence haute. Ces filtre sont tendance à rendre l'image floue .Ce type de filtrage est généralement utilisé pour atténuer le bruit de l'image, c'est la raison pour laquelle on parle habituellement de lissage. On distingue deux filtres passe-bas, le filtre Moyenneur , et le filtre gaussien

10- Filtrage des images : Filtrage Linéaire

A- Filtre moyenneur

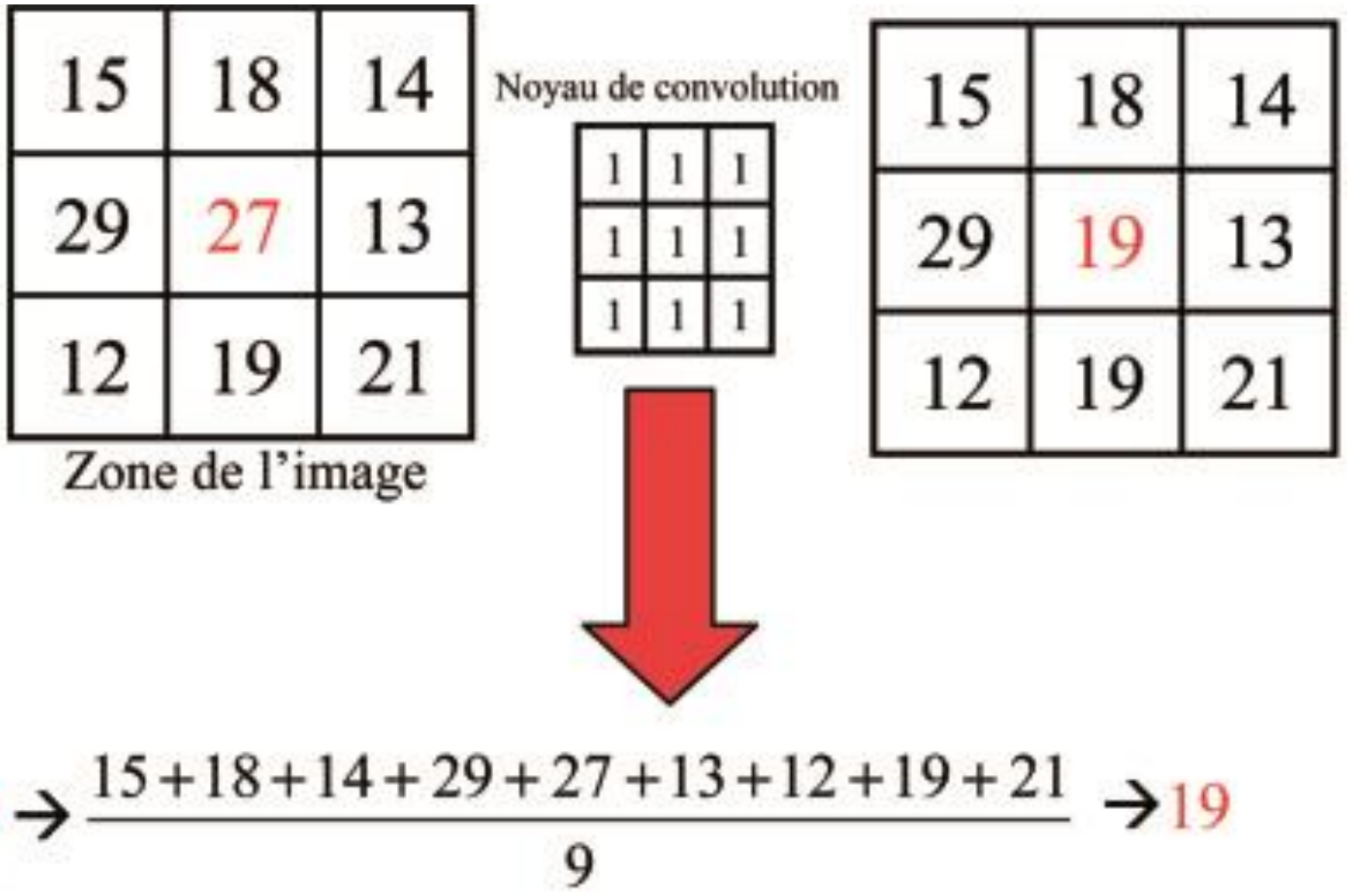
- Objectif : lisser l'image
- Fonctionnement : on remplace la valeur d'un pixel par la moyenne des valeurs des pixels du voisinage
- Noyau de convolution :

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

- Comment choisir la taille/la forme du voisinage ?

10- Filtrage des images : Filtrage Linéaire

Exemple : l'opérateur moyenne



$$27 \rightarrow \frac{15+18+14+29+27+13+12+19+21}{9} \rightarrow 19$$

10- Filtrage des images : Filtrage Linéaire

Image originale

Moyenne 3×3

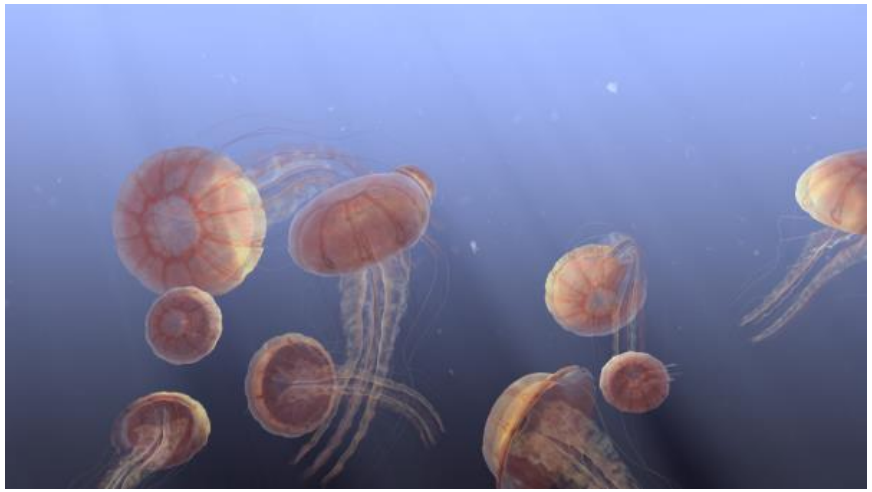
Moyenne 5×5

Moyenne 7×7

Amélioration

Au lieu de faire contribuer tous les pixels également, on peut privilégier les pixels proches du centre

10- Filtrage des images : Filtrage Linéaire



1/9

1	1	1
1	1	1
1	1	1

10- Filtrage des images : Filtrage Linéaire

B- Filtre gaussien

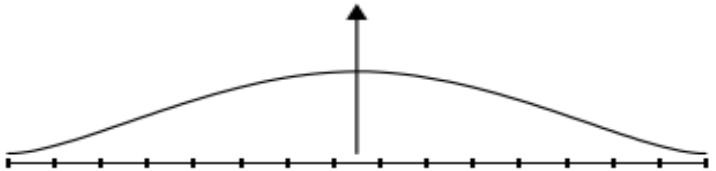
- Objectif : lisser l'image
- Fonctionnement : on remplace la valeur d'un pixel

par la moyenne pondérée des valeurs des pixels du voisinage

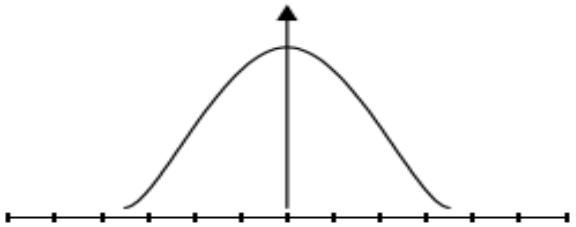
- Noyau de convolution : gaussienne
- Paramètre / Taille du Noyau ?

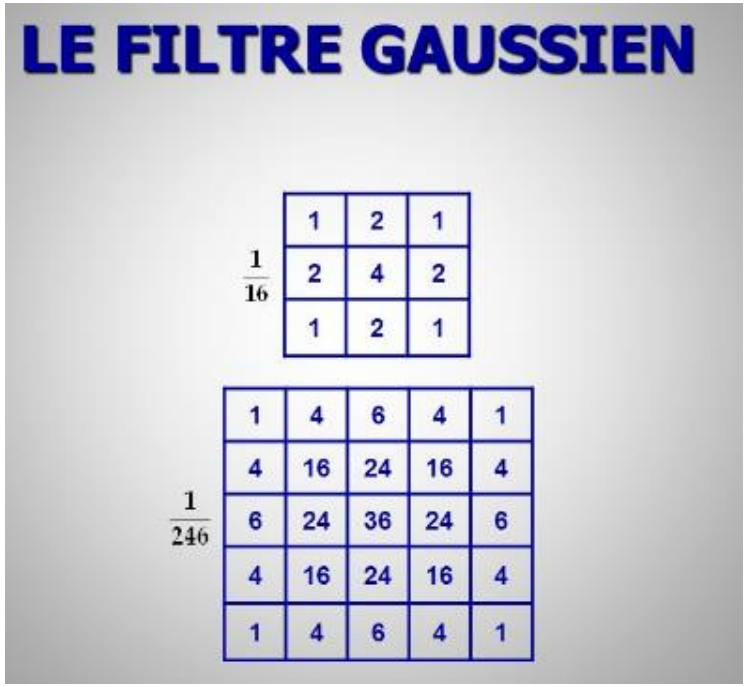
Les coefficients du noyau sont ici calculés en utilisant des pondérations gaussiennes.

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



- Largeur de la fenêtre
  - typiquement 2sigma+1
- avantage
  - filtre paramétrable (sigma)
  - adapter au problème
    - taille de la fenêtre
    - valeur de sigma
- Inconvénient
  - complexité (calcul flottant et non entier)





9- Filtrage des images : Filtrage Linéaire

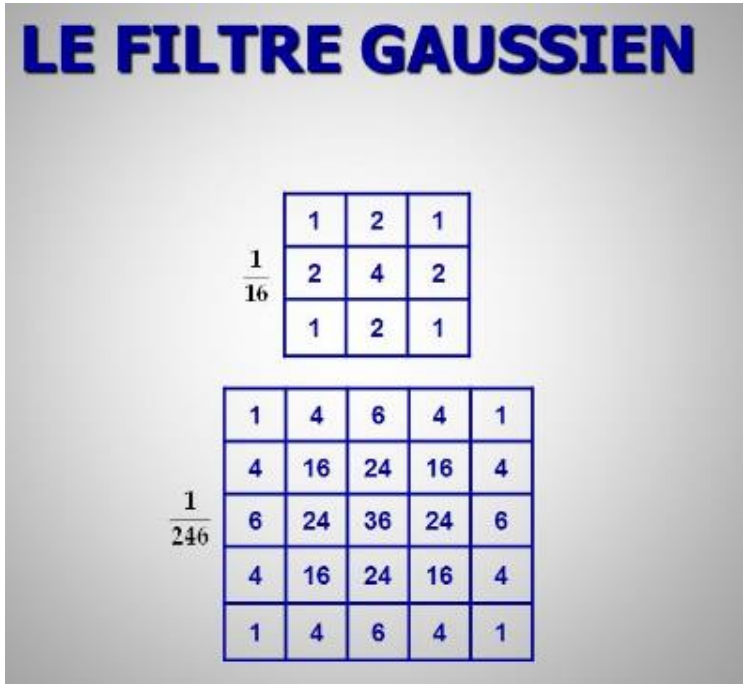
B- Filtre gaussien

Image originale

Gaussienne de variance 0,75

Gaussienne de variance 2,08

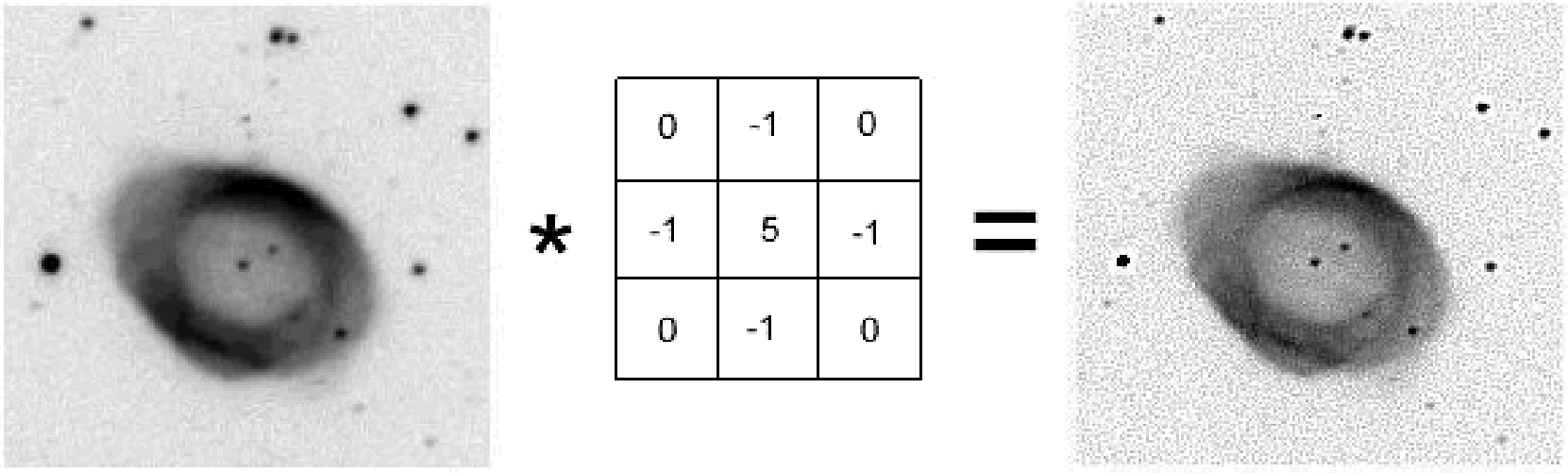
Gaussienne de variance 4,08



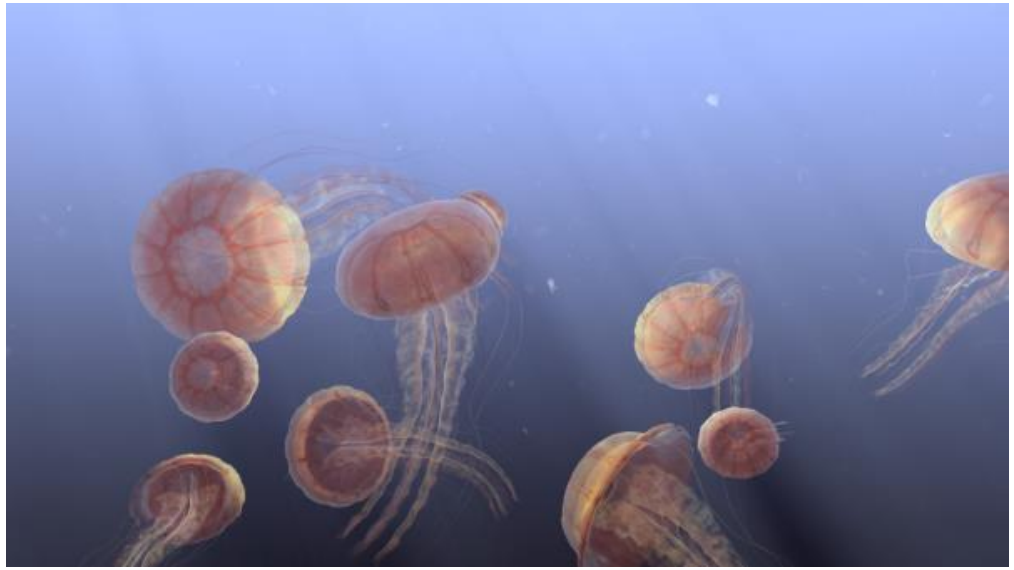
10- Filtrage des images : Filtrage Linéaire

2- **Les filtres passe-haut :**

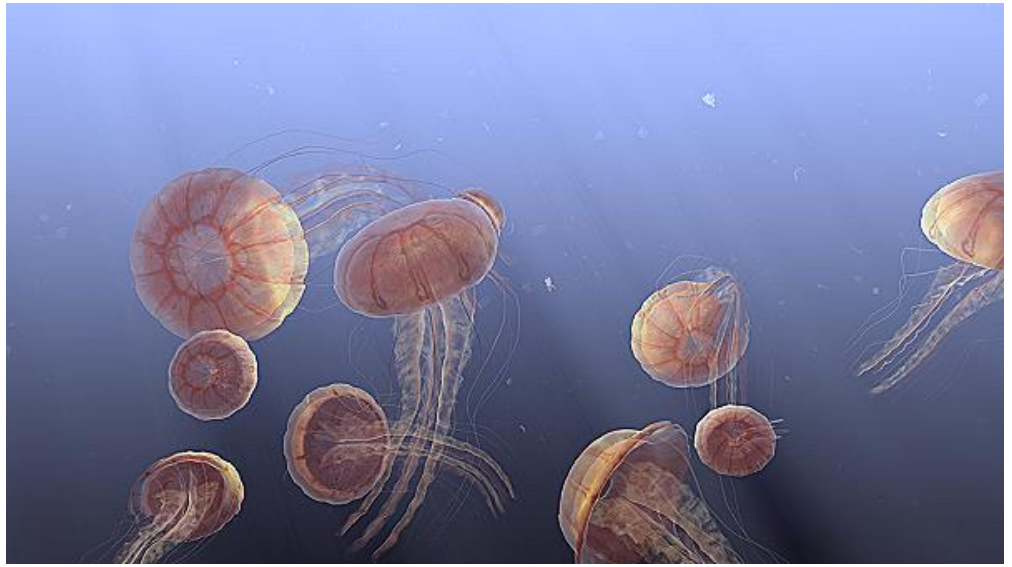
- Les filtres détecteurs de contours ou passe-haut à l'inverse des passe-bas, atténuent les composantes de basse fréquence de l'image et permettent notamment d'accentuer les détails et le contraste
- Ces filtres seront détailler dans la partie de détection de contours.



10- Filtrage des images : Filtrage Linéaire



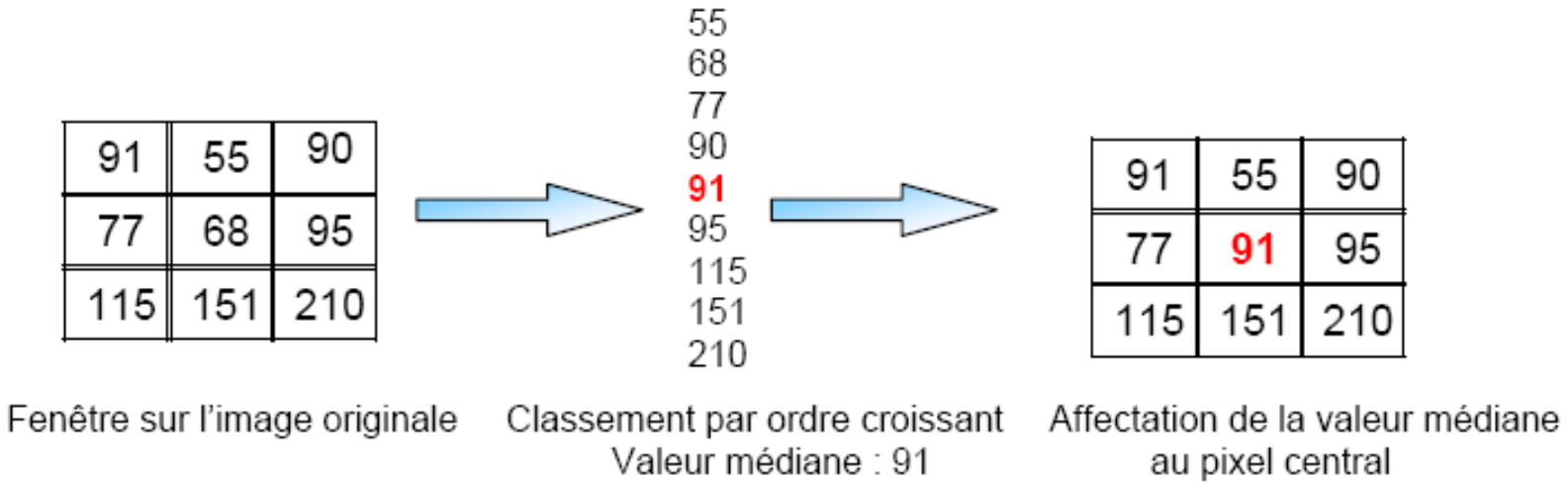
0	-1	0
-1	5	-1
0	-1	0



10- Filtrage des images : Filtrage Non Linéaire

A- Filtre Médian

Le niveau de gris du pixel central est remplacé par la valeur médiane de tous les pixels de la fenêtre centrée sur le pixel.



Avantages:

-il préserve les contours pour les petites tailles du filtre

il est efficace dans le cas du bruit impulsionnel (type poivre et sel).

10- Filtrage des images : Filtrage Non Linéaire

bureau

médian 3x3

médian 5x5

nagao

FLOU DE L 'IMAGE

Filtre MOYENNAGE

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

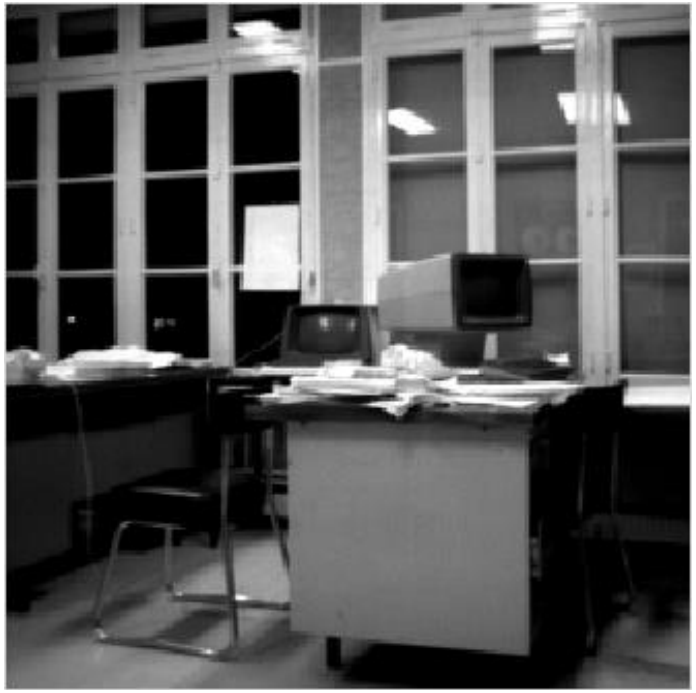


Image originale f(x,y)

Image filtrée g(x,y)

EXTRACTION DES CONTOURS

Filtre LAPLACIEN

	-1	
-1	4	-1
	-1	

Image originale

Image filtrée

REHAUSSEMENT DES CONTOURS

	-1	
-1	5	-1
	-1	

=

1		

+

	-1	
-1	4	-1
	-1	

Image originale

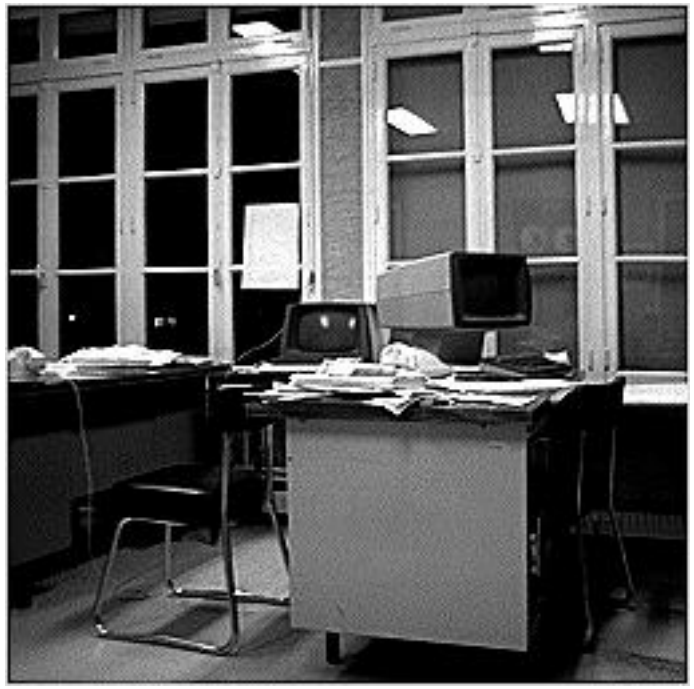
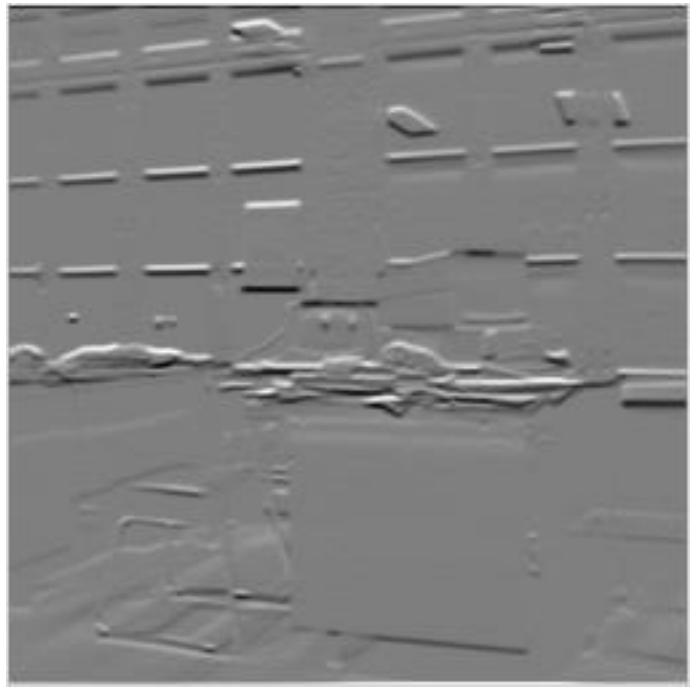


Image rehaussée

RELIEF

Filtre SOBEL

1	2	1
0	0	0
-1	-2	-1



Horizontal

1	0	-1
2	0	-2
1	0	-1

Vertical

Exemple

Originale

Niveaux de gris

Lissage

Binarisée

Inversée

contours-laplacien

contours-sobel

11- OpenCV & Filtrage des images

Image floue

```
import cv2
import numpy as np
# Loading source image
src_image = cv2.imread("dog.jpg")
# Defining the kernel of size 3x3
kernel = np.array([
  [1, 1, 1],
  [1, 1, 1],
  [1, 1, 1]
]) / 9

resulting_image = cv2.filter2D(src_image, -1, kernel)

cv2.imshow("original image", src_image)
cv2.imshow("filter2d image", resulting_image)
cv2.imwrite("Filter2d Blur Image.jpg", resulting_image)
cv2.waitKey()
cv2.destroyAllWindows()
```

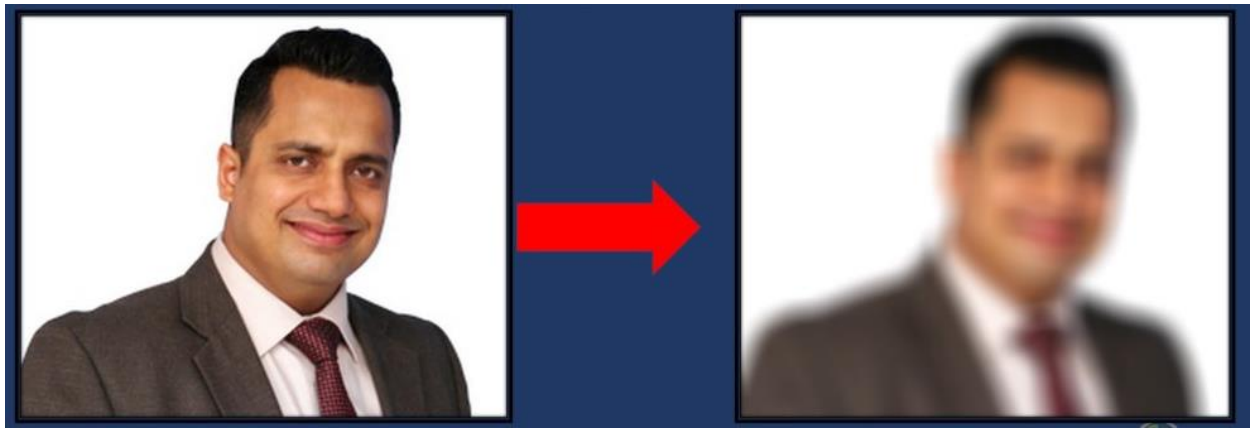
11- OpenCV & Filtrage des images

Détection du contour

```
import cv2
import numpy as np
# Loading source image
src_image = cv2.imread("pug-dog.jpg")
# Defining the kernel of size 3x3
kernel = np.array([
  [-1, -1, -1],
  [-1, 8, -1],
  [-1, -1, -1]
])

resulting_image = cv2.filter2D(src_image, -1, kernel)

cv2.imshow("original image", src_image)
cv2.imshow("filter2d image", resulting_image)
cv2.imwrite("Filter2d Outline Image.jpg", resulting_image)
cv2.waitKey()
cv2.destroyAllWindows()
```

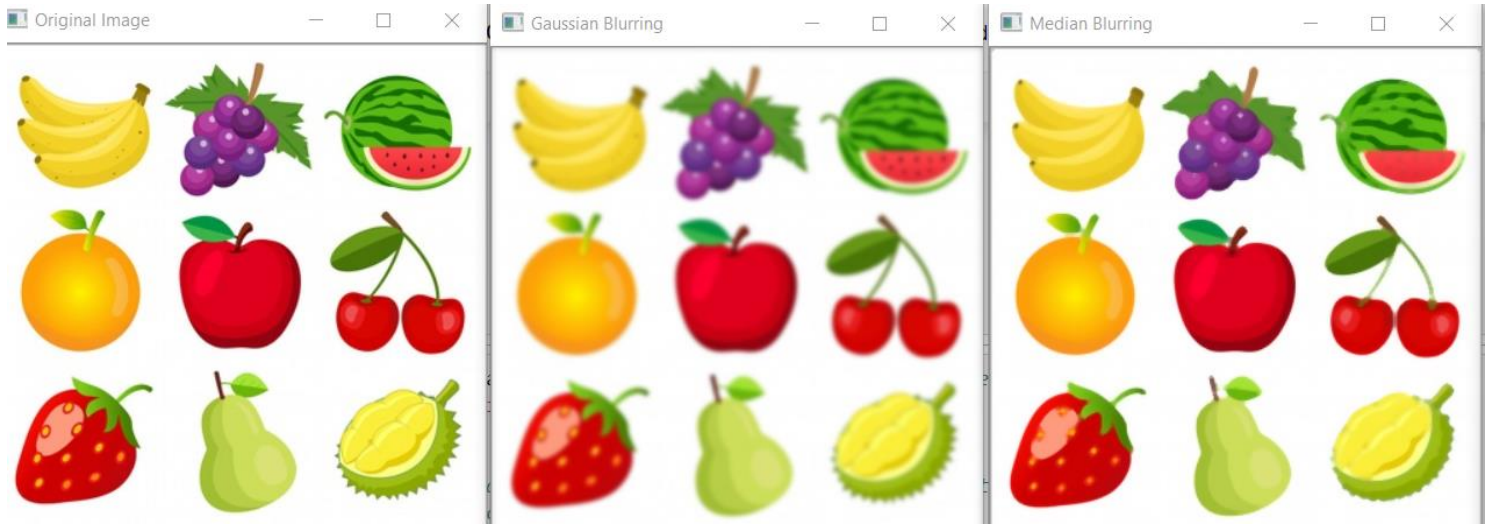
11- OpenCV & Filtrage des images

```
# Python program to explain cv2.blur() method

# importing cv2
import cv2

# path
path = r'C:\Users\Rajnish\Desktop\geeksforgeeks\geeks.png'

# Reading an image in default mode
image = cv2.imread(path)

# ksize
ksize = (10, 10)

# Using cv2.blur() method
image = cv2.blur(image, ksize)

# Displaying the image
cv2.imshow(window_name, image)
```



11- OpenCV & Filtrage des images

```
import cv2
import numpy as np

image = cv2.imread('C://Geeksforgeeks//image_processing//fruits.jpg')

cv2.imshow('Original Image', image)
cv2.waitKey(0)

# Gaussian Blur
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian Blurring', Gaussian)
cv2.waitKey(0)

# Median Blur
median = cv2.medianBlur(image, 5)
cv2.imshow('Median Blurring', median)
cv2.waitKey(0)
```



12- Les Opérations morphologiques

- Parfois, après avoir traité votre image avec un seuil (Threshold), vous avez du bruit indésirable dans l'image binaire résultante. Les opérations morphologiques peuvent aider à supprimer ce bruit.

- Le noyau est une forme simple où l'origine est superposée à chaque pixel de valeur 1 de l'image binaire. OpenCV limite le noyau à une matrice NxN où N est un nombre impair. L'origine du noyau est le centre. Un noyau commun est:

$$kernel = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

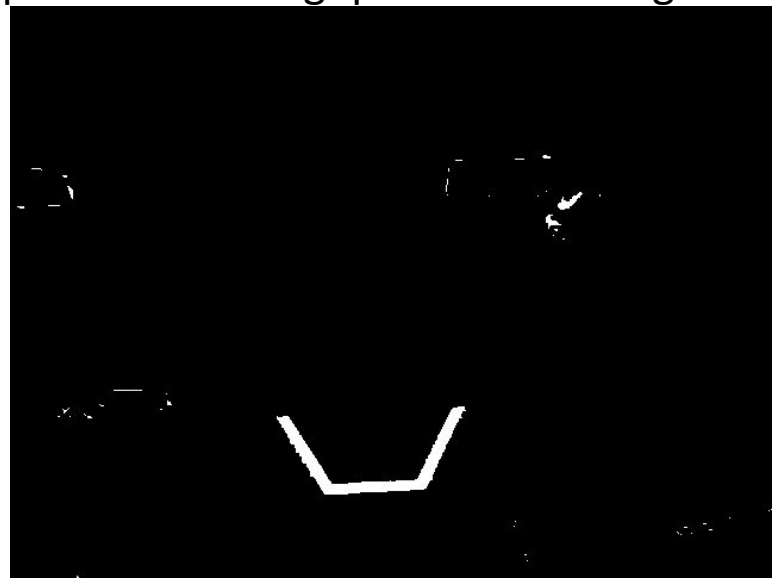
- Différents noyaux peuvent affecter l'image différemment, tels que l'érosion ou la dilatation

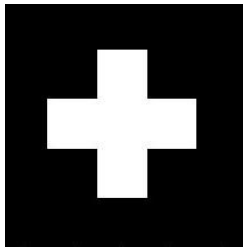
12- Les Opérations morphologiques

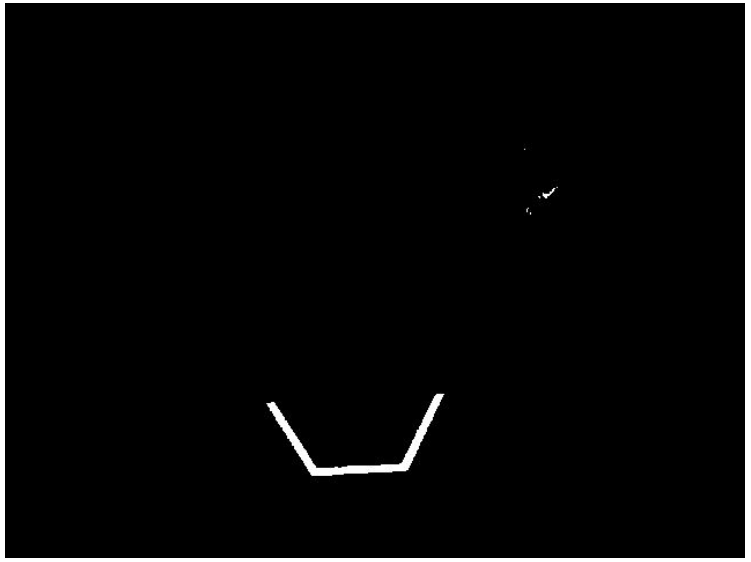
Érosion

L'érosion en vision par ordinateur est similaire à l'érosion sur le sol. Ce processus peut supprimer le bruit de l'arrière-plan.

Correspond à un ET logique sur une image binaire







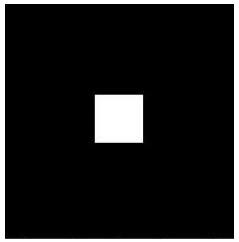


Image originale

Image après érosion

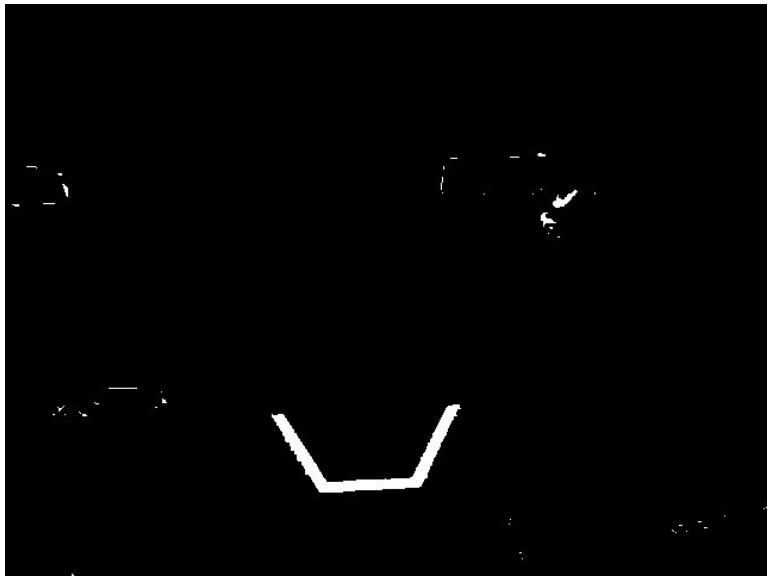
```
kernel = np.ones((3, 3), np.uint8)
binary_img = cv2.erode(binary_img, kernel, iterations = 1)
```

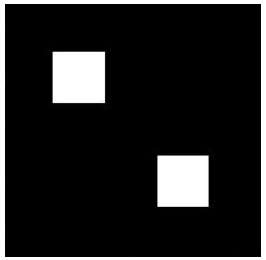
12- Les Opérations morphologiques

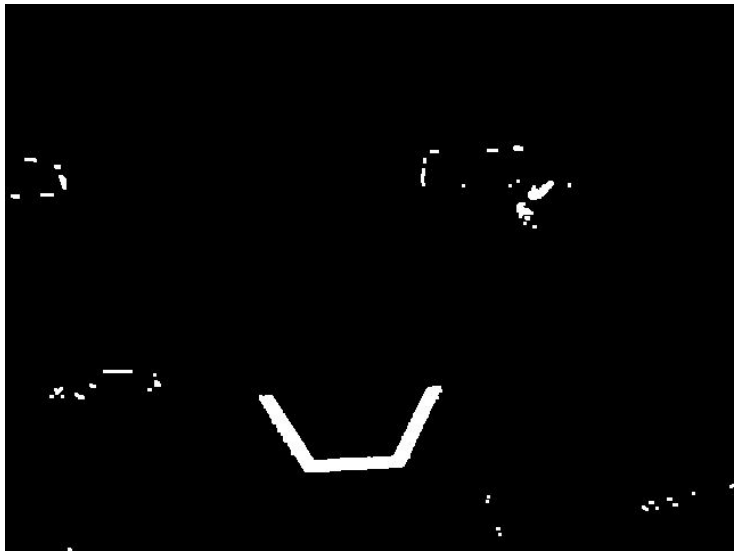
Délatation

La dilatation est le contraire de l'érosion. Ce processus peut contribuer à supprimer de petits trous à l'intérieur d'une plus grande région.

Correspond à un OU logique sur une image binaire







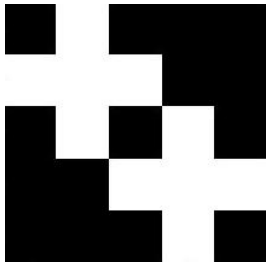


Image originale

Image après érosion

```
kernel = np.ones((3, 3), np.uint8)
binary_img = cv2.dilate(binary_img, kernel, iterations = 1)
```

12- Les Opérations morphologiques

Ouverture & Fermeture

- L'ouverture c'est la composition de l'érosion par un gabarit suivie de la dilatation par ce même gabarit.
  Ce processus supprime le bruit sans affecter la forme des entités plus grandes.
- La fermeture c'est la composition de la dilatation par un gabarit suivie de l'érosion par ce même gabarit

Ce processus supprime les petits trous ou les ruptures sans affecter la forme des entités plus grandes.

```
•. kernel = numpy.ones((3, 3), numpy.uint8);
   newImg = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
• kernel = numpy.ones((3, 3), numpy.uint8);
  newImg = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

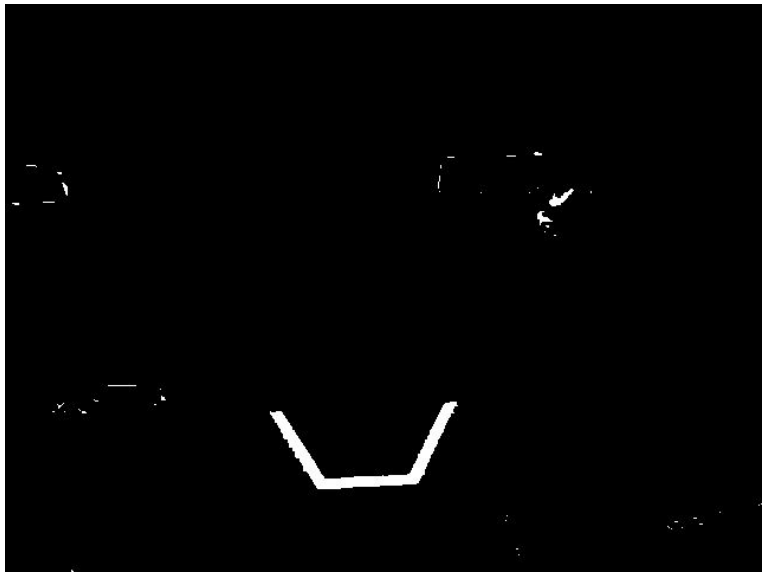
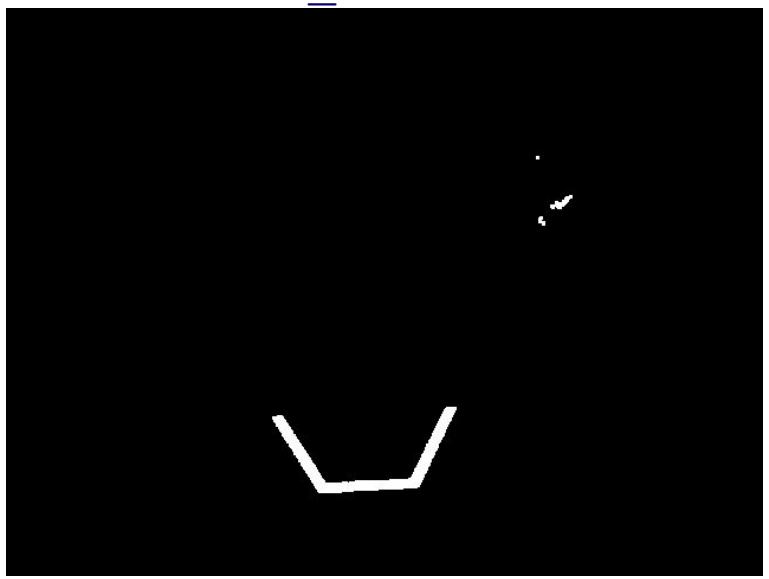
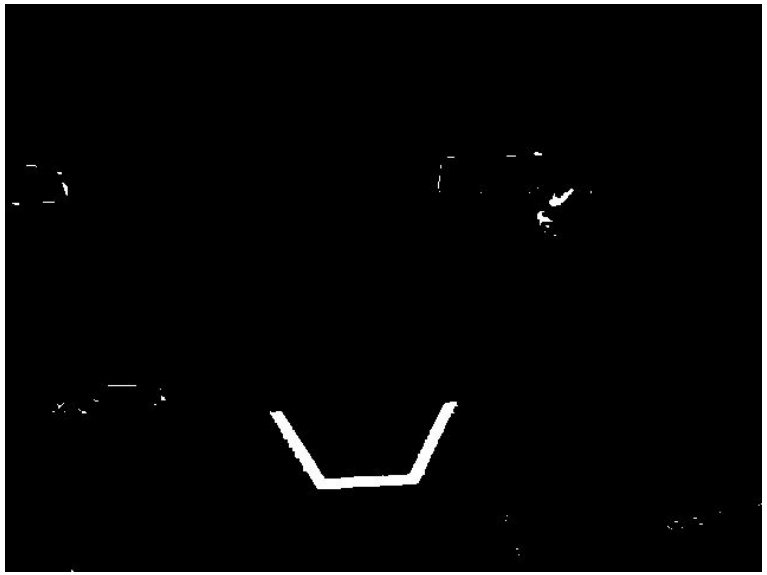


Image originale



ouverture



fermeture

12- Les Opérations morphologiques

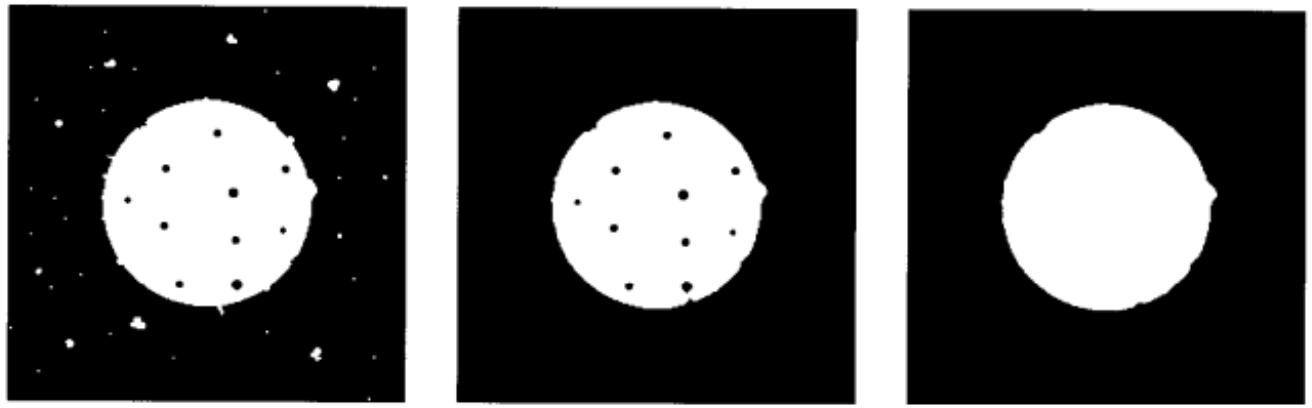


Image Ouverture Fermeture

Application de la morphologie mathématique : Elle vise à déterminer les caractéristiques d'un objet, simplifie l'image en supprimant certaines structures géométriques ainsi que la séparation des objets collés.

Le filtrage d'images : suppression d'objets ne satisfaisant pas certains critères morphologiques

Segmentation : pour obtenir une partition de l'image en ses différentes régions d'intérêt

Mesure d'images : à chaque image (mesure globale) ou à chaque point d'une image (mesure locale) on associe une mesure. Exemple : extraction de caractéristiques fondée sur des critères morphologiques et géométriques. Elle fournit ainsi des outils pour la reconnaissance des formes