Support de Travaux Pratiques Traitement d'images

Niveau 2éme année cycle Ingénieur en Génie Informatique Industrielle.

Etablissement EnetCom Sfax

Année universitaire 2022-2023

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Ecole nationale d'électronique et des télécommunications de Sfax

Support de travaux pratiques
Traitement des images

2022 – 2023

2ème année Ingénieur Génie Informatique Industrielle (2GII)

Responsable du cours :

Ali KHALFALLAH, Maitre-assistant à Enetcom de sfax

Enseignante :

Nessrine ELLOUMI, Assistante Vacataire à Enetcom de sfax

Sommaire

TP1 : Affichage des images

TP2 : Outils pratiques et restauration des images

TP3 : Opérateurs morphologiques et segmentation des images

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Ecole nationale d'électronique et des télécommunications de Sfax

Traitement des images

TP1 : Affichage des images

2022 – 2023

2éme année Ingénieur Génie Informatique Industrielle (2GII)

Responsable du cours:

Ali KHALFALLAH, Maitre-assistant à Enetcom de sfax

Enseignante:

Nessrine ELLOUMI, Assistante Vacataire à Enetcom de sfax

Tps Traitement des images

Les applications de traitements d'image sont diverses dont on peut citer, le codage de l'image, l'amélioration de l'affichage de l'image ou en extraire des informations. Dans ce contexte plusieurs algorithmes, langage, bibliothèque sont disponibles. Durant ces séances de TP nous allons utiliser python sur google Colab pour appliquer nos algorithmes de traitement d'image.

1. Environnement de travail : Google Colab

Les environnements de développement collaboratifs en ligne gagnent en popularité dans le domaine de l'apprentissage automatique et de l'analyse de données. Google Colab, abréviation de Colaboratory, est l'un de ces outils puissants qui offre un accès gratuit aux ressources informatiques. Google Colab est un choix idéal pour les étudiants et les professionnels cherchant à collaborer et à travailler sur des projets de programmation, d'analyse de données et d'apprentissage automatique. Google Colab repose sur la plateforme cloud de Google, offrant ainsi une solution sans tracas pour exécuter du code Python, créer des carnets de notes interactifs et collaborer en temps réel avec d'autres utilisateurs. Dans le cadre de ce TP, nous explorerons les fonctionnalités de base de Google Colab, allant de la création de carnets de notes à l'exécution de code, en passant par l'accès à des bibliothèques populaires telles que Open Source Computer Vision Library (cv2), PIL, Numpy et matplotlib.

2. Les types de variables dans python.

Entiers (int) : Représente des nombres entiers, positifs ou négatifs, sans partie décimale. Exemple : a = 5.

Nombres à virgule flottante (float) : Représente des nombres décimaux. Exemple : b = 3.14.

Chaînes de caractères (str) : Représente une séquence de caractères. Exemple : c = "Bonjour".

Booléens (bool) : Représente les valeurs de vérité True ou False. Exemple : vrai = True.

Listes (list) : Représente une séquence mutable d'éléments. Les éléments peuvent être de différents types. Exemple : liste = [189, True, "Image"].

Tuples (tuple) : Représente une séquence d'éléments. Exemple : tup = (45, 92, 36). Ce type de variable ne permet pas la modification de son contenu après la création.

Dictionnaires (dict) : Représente une collection de paires clé-valeur. Exemple : dictionnaire = {'ID_produit': 3536}

Ensembles (set) : Représente une collection non ordonnée d'éléments uniques. Exemple : ensemble = {6,8,7}.

3. Manipulation des matrice et affichage des données

NumPy : NumPy est une bibliothèque très utilisée pour le calcul numérique en Python. Elle introduit le type de tableau numpy.ndarray, qui offre des fonctionnalités avancées pour les opérations mathématiques sur les tableaux multidimensionnels.

Matplotlib : Matplotlib est une bibliothèque de visualisation de données en 2D qui permet de créer des graphiques statiques, des diagrammes et des visualisations interactives dans le langage de programmation Python. Elle offre une grande flexibilité pour la création de graphiques de qualité professionnelle.

4. Bibliothèques de traitement d'images dans Python

Open-Source Computer Vision Library (OpenCV) : est une bibliothèque très puissante pour le traitement d'images et la vision par ordinateur. Elle offre un large éventail de fonctionnalités, y compris la manipulation d'images, la détection d'objets, la reconnaissance faciale, le suivi d'objets, la stéréovision, etc.

Python Imaging Library (PIL) : est une bibliothèque Python qui permet de travailler avec des images. Cependant, veuillez noter que PIL a été abandonné et n'est plus activement développé. Son successeur, plus récent et maintenu, est la bibliothèque **Pillow**. Pillow est un fork de PIL qui a été créé pour fournir des mises à jour et des améliorations continues à la bibliothèque d'origine. Pillow conserve la plupart des fonctionnalités de PIL tout en ajoutant de nouvelles fonctionnalités.

Scikit-image : est une extension de la bibliothèque Scikit-learn dédiée au traitement d'images. Elle propose des outils pour le filtrage, la segmentation, la transformation géométrique, la mesure des propriétés d'objets, etc.

TP1 : Affichage des images

Introduction

Dans ce TP nous allons nous focaliser l'aspect matricielle de l'image et la notions des espaces colorimétriques. Pour atteindre ce but nous allons utiliser les bibliothèques PIL et Open CV, pour afficher les images nous allons utiliser la bibliothèque Matplotlib.

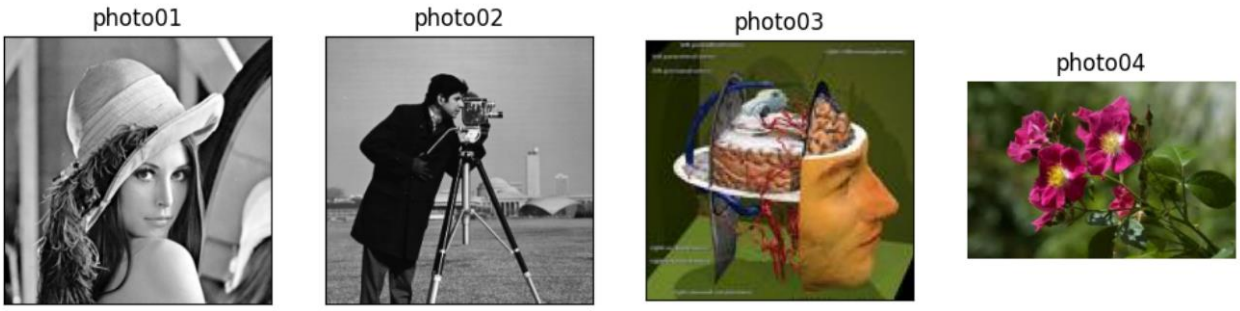
Exercice 1

Créer un fichier EX01.py pour réaliser le travail suivant :

```
from PIL import Image
import matplotlib.pyplot as plt
im1 = Image.open('photo01.bmp')
im2 = Image.open('photo02.bmp')
```

```
im3 = Image.open('photo03.jpg')
im4 = Image.open('photo04.jpg')
titles = ['photo01', 'photo02', 'photo03', 'photo04']
images = [im1, im2, im3, im4]
plt.figure(figsize=(12, 3))
for i in range(4):
    plt.subplot(1, 4, i + 1)
    plt.imshow(images[i], cmap='gray', vmin=0, vmax=255)
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.axis('off')
plt.show()
```

- **Résultat**



Exercice 2

Créer un fichier EX02.py pour réaliser le travail suivant :

1) Lire et afficher l'image "photo01.bmp" (l'affecter à la variable Im).

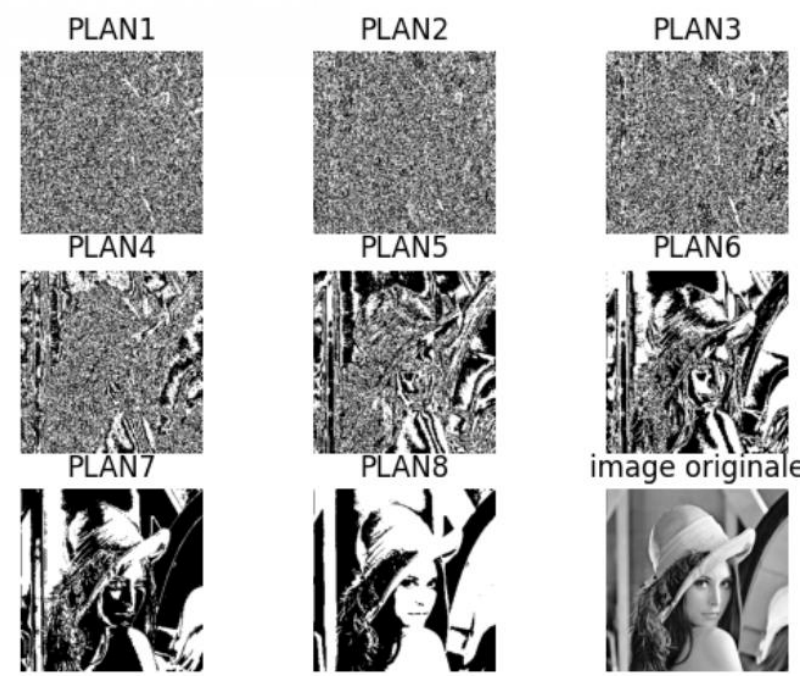
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
Im = Image.open('photo01.bmp')
Im_array = np.array(Im)
plt.imshow(Im_array, cmap='gray')
plt.title('Image originale')
plt.show()
```

2) Affecter à V1, V2, …, V8 les 8 plans de Im.

```
bit_planes = [(Im_array >> i) & 1 for i in range(8)]
fig, ax = plt.subplots(3, 3)
for i in range(8):
    row = i // 3
    col = i % 3
    ax[row, col].imshow(bit_planes[i],
cmap='gray')
    ax[row, col].set_title(f'PLAN{i+1}')
ax[2, 2].imshow(Im, cmap='gray')
ax[2, 2].set_title('image originale')
plt.show()
```

V1	V2	V3
V4	V5	V6
V7	V8	Im

➢ **Résultat**



3) Calculer et afficher la variable suivante : G sachant que

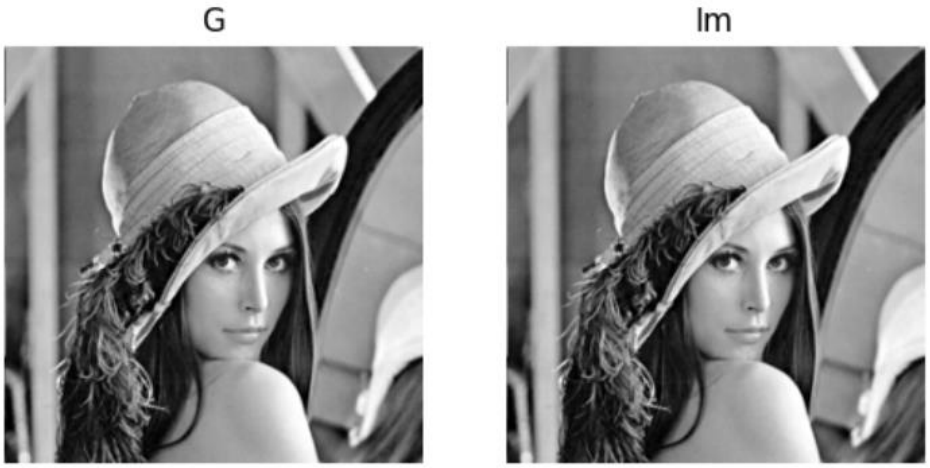
$G=V0*2^0 + ...+V7*2^7$

Im	G

```
V = []
for i in range(8):
    V.append(np.uint8(np.bitwise_and(Im, 2**i) / 2**i))
```

G = V[0]*20 + V[1]*2**1 +……..**
G = np.uint8(G)

➢ **Résultat**



4) Calculer et afficher les variables suivantes : G1, G2 et G3

$G1=V5*2^5 + ...+V7*2^7$

$G2=V4*2^4 + ...+V7*2^7$

$G3=V3*2^3 + ...+V7*2^7$

G_1	G_2
G_3	Im

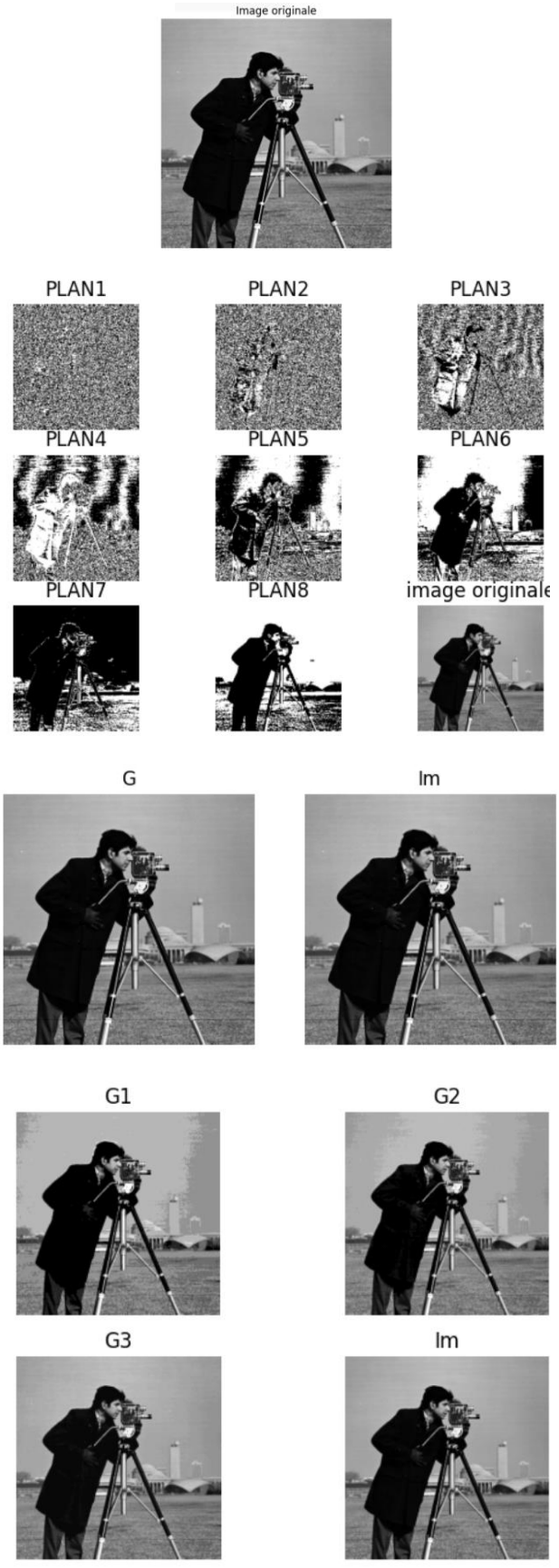
- Afficher les variables en utilisant la bibliothèque **matplotlib**

```
fig, axs = plt.subplots(2, 2)
axs[0, 0].imshow(G1, cmap='gray')
axs[0, 0].set_title('G1')
axs[0, 0].axis('off')

axs[0, 1].imshow(G2, cmap='gray')
axs[0, 1].set_title('G2')
axs[0, 1].axis('off')

axs[1, 0].imshow(G3, cmap='gray')
axs[1, 0].set_title('G3')
axs[1, 0].axis('off')

axs[1, 1].imshow(Im_array, cmap='gray')
axs[1, 1].set_title('Im')
axs[1, 1].axis('off')
plt.show()
```

- **Résultat**

5) Appliquer ce programme sur l'image "photo02.bmp"

- **Résultat**



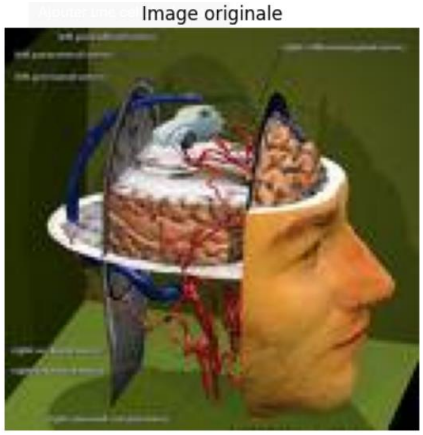
Exercice 3

Créer un fichier EX03.py pour réaliser le travail suivant :

1) Lire l'image "photo03.jpg" (l'affecter à la variable ImCol) et l'afficher (figure 1).

```
Imcolo = Image.open('photo03.jpg')
```

➢ **Résultat**

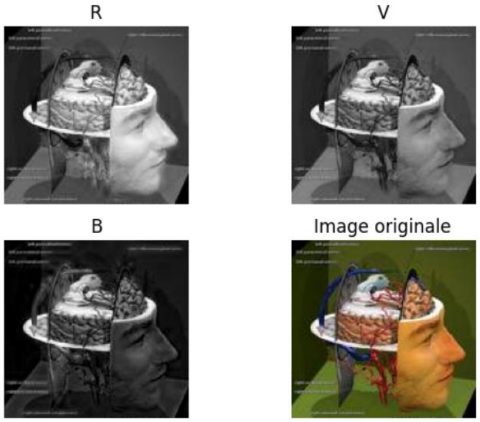


2) Décomposer ImCol en 3 matrices de chrominance R, V et B

```
r, v, b = Imcolo.split()
```

R	V
B	ImCol

➢ **Résultat**



3) Créer une matrice Z de même dimension que R (on appellera nl et nc respectivement le nombre de lignes et celui de colonnes de R). Toutes les valeurs de Z sont nulles.

```
nl, nc = Imcolo.size
z = np.zeros((nl, nc), dtype='uint8')

def filtre_rouge_pil(image):
    imr = image.copy()
    imr_np = np.array(imr)
    imr_np[:, :, 1] = z[:, 1]
    imr_np[:, :, 2] = z[:, 2]
    imr = Image.fromarray(imr_np)
    return imr
```

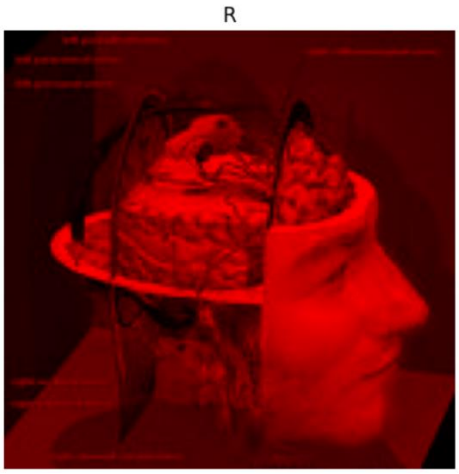
4) Créer une matrice ImR en utilisant la ligne suivante :

```
imr= filtre_rouge_pil(Imcolo)
```

5) Afficher l'image ImR

```
plt.imshow(imr)
plt.title('R')
plt.axis('off')
plt.show()
```

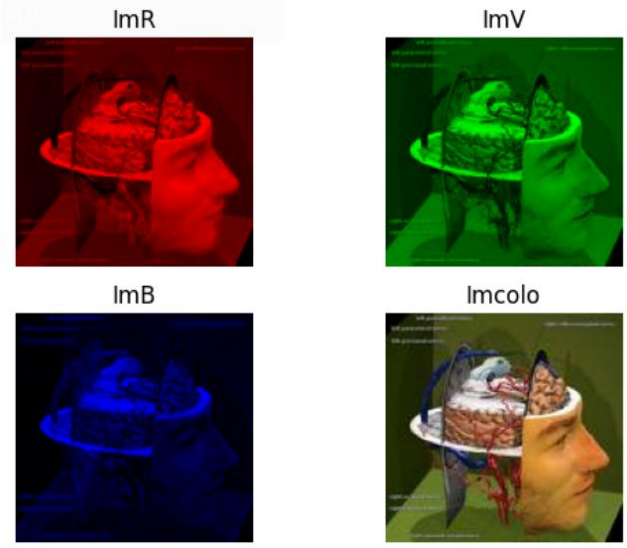
➢ **Résultat**



6) Dans une nouvelle figure, afficher ImColo et ses trois composantes de chrominance (ImR, ImV, ImB).

➢ **Résultat**

ImR	ImV
ImB	ImCol



7) Reconstruire l'image en cours de traitement à partir de R, V et B et l'affecter à la variable ImRec.

```
Imcolo1 = Image.merge('RGB', (r, v, b))
```

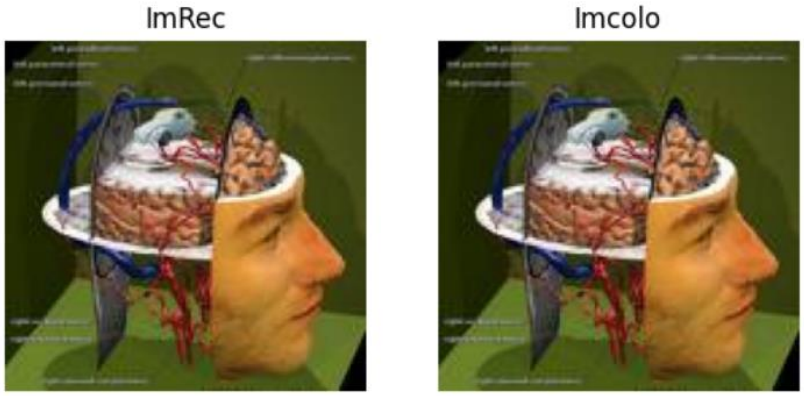
8) Afficher les images ImRec et ImCol dans une nouvelle figure (figure 4).

```
titles = ['ImRec','Imcolo']
```

ImRec	ImCol

```
images = [Imcolo1, Imcolo]
for i in range(2):
    plt.subplot(1,2,i+1),plt.imshow(images[i])
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.axis('off')
plt.show()
```

➢ **Résultat**



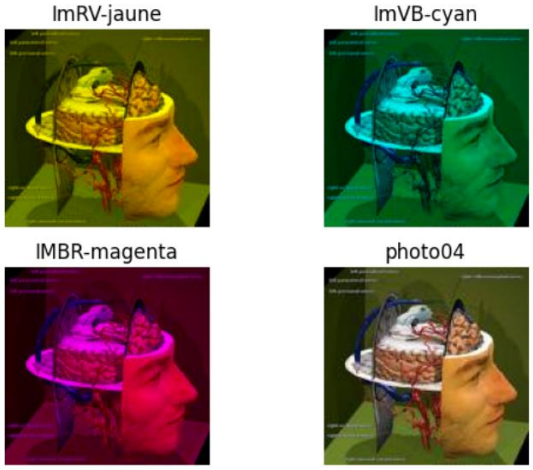
9) Créer une nouvelle image ImRVJaune qui associe à sa composante rouge la matrice R, à sa composante verte la matrice V et à sa composante bleue une matrice de valeurs nulles. Afficher ImRVJaune , ImVBCyan , ImBRMagenta et ImColo dans une même figure (figure 5).

$ImRV_{jaune}$	$ImVB_{Cyan}$
$ImBR_{Magenta}$	ImCol

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
Imcolo = Image.open('photo03.jpg')
plt.imshow(Imcolo)
plt.title('Original Image')
plt.show()
def filtre_function(Imcolo, channel):
    imFiltered = Imcolo.copy()
    pixels = imFiltered.load()
    for i in range(imFiltered.width):
        for j in range(imFiltered.height):
            r, v, b = pixels[i, j]
            if channel == 'RV':
                pixels[i, j] = (r, v, 0)
            elif channel == 'VB':
                pixels[i, j] = (0, v, b)
            elif channel == 'BR':
                pixels[i, j] = (r, 0, b)
    return imFiltered
imRV = filtre_function(Imcolo, 'RV')
imVB = filtre_function(Imcolo, 'VB')
imBR = filtre_function(Imcolo, 'BR')
titles = ['ImRV-jaune', 'ImVB-cyan', 'IMBR-magenta', 'photo04']
images = [imRV, imVB, imBR, Imcolo]
```

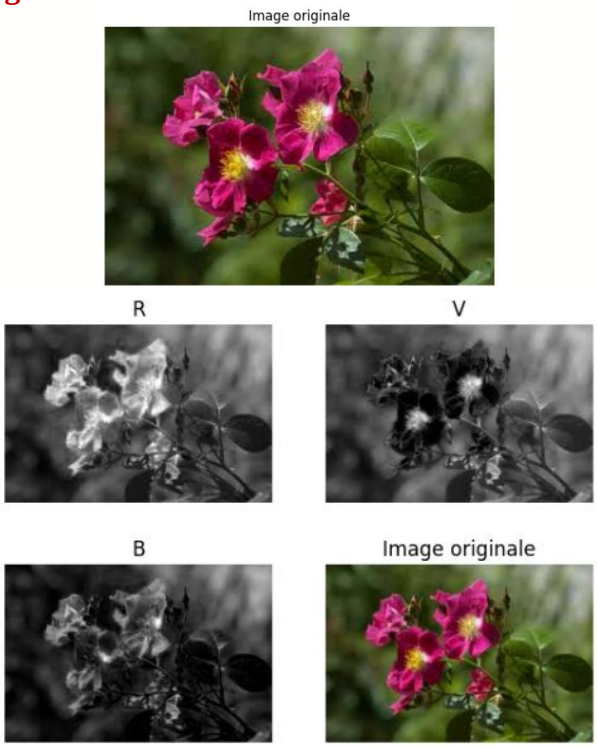
```
for i in range(4):
    plt.subplot(2, 2, i+1), plt.imshow(images[i])
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
    plt.axis('off')
plt.show()
```

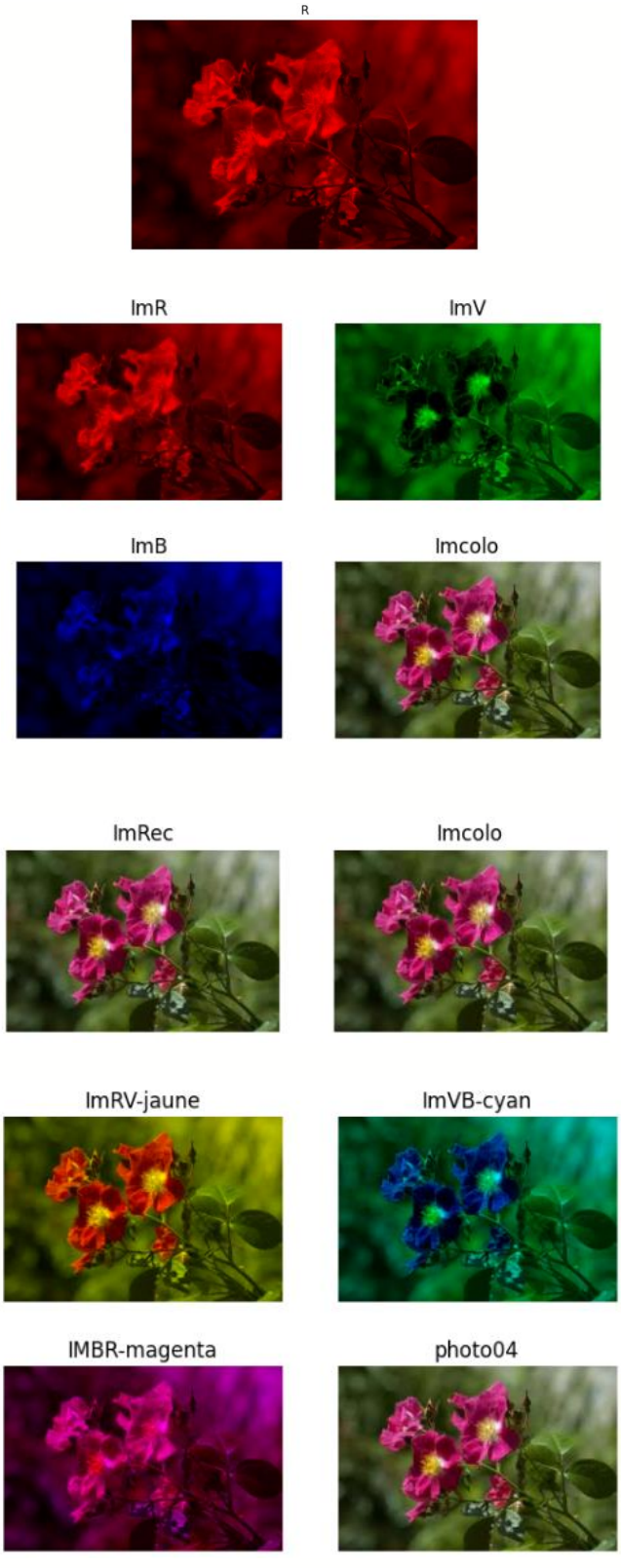
- **Résultat**



10) Appliquer ce programme sur l'image "photo04.jpg"

- **Résultat « photo04.jpg »**





Exercice 4

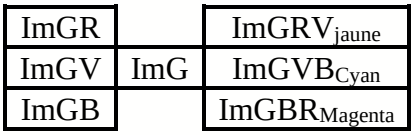
Créer un fichier EX04.py pour réaliser le travail suivant :

1) Lire l'image "photo01.bmp" (l'affecter à la variable ImG) et l'afficher (figure 1).

2) Créer une matrice Vide de mêmes dimensions que ImG et dont les valeurs sont nulles.

3) Créer une image colorée ImGRV dont les composantes rouge et verte sont égales à la matrice ImG et la composante bleu est égale à la matrice Vide.

Afficher ImGRVJaune , ImGVBCyan , ImGBRMagenta et ImG dans une même figure.




```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

ImG = Image.open('photo01.bmp')
nl, nc = ImG.size
ImG = ImG.convert('RGB')
Z = Image.fromarray(np.zeros((nl, nc), dtype=np.uint8))

ImGR = Image.merge('RGB', (ImG.split()[0], z,z)

ImGV = Image.merge('RGB', (z, ImG.split()[1],z)

ImGB = Image.merge('RGB', z, z, ImG.split()[2]))

ImGRV = Image.merge('RGB', (ImG.split()[0], ImG.split()[1],z)

ImGVB = Image.merge('RGB', (z , ImG.split()[1], ImG.split()[2]))

ImGBR = Image.merge('RGB', (ImG.split()[0], z, ImG.split()[2]))

# Afficher les images
fig, axes = plt.subplots(3, 3, figsize=(12, 12))
axes[0, 0].imshow(ImGR)
axes[0, 0].set_title('ImR')
axes[0, 0].axis('off')

axes[1, 0].imshow(ImGV)
axes[1, 0].set_title('ImV')
axes[1, 0].axis('off')
axes[2, 0].imshow(ImGB)
axes[2, 0].set_title('ImB')
axes[2, 0].axis('off')

axes[0, 1].imshow(ImG)
axes[0, 1].set_title('ImColo')
axes[0, 1].axis('off')
axes[1, 1].axis('off')
axes[2, 1].axis('off')
axes[0, 2].imshow(ImGRV)
axes[0, 2].set_title('ImRV Jaune')
axes[0, 2].axis('off')
axes[1, 2].imshow(ImGVB)
axes[1, 2].set_title('ImVB Cyan')
```

```
axes[1, 2].axis('off')

axes[2, 2].imshow(ImGBR)
axes[2, 2].set_title('ImBR Magenta')
axes[2, 2].axis('off')
plt.show()
```

- **Résultat**

Exercice 5

1) Lire l'image "photo04.jpg" (l'affecter à la variable Im) et l'afficher (figure 1).
2) Donner le rôle des fonctions suivante :

```
hsv = Im.convert('HSV')
```

Cette instruction convertit l'image Im du format original (RVB) au format HSV (teinte, saturation, luminosité). HSV est un espace colorimétrique qui représente les couleurs en termes de teinte, de saturation et de luminosité.

```
h, s, v = hsv.split()
```

Cette instruction divise l'image convertie en HSV en trois canaux distincts : teinte (H), saturation (S) et luminosité (V). Ces canaux individuels peuvent être traités séparément pour effectuer diverses opérations de traitement d'image, telles que la modification de la teinte, l'ajustement de la saturation ou le contrôle de la luminosité.

3) Afficher dans la même figure Im, H1, H2, et H3 Avec
H1 = np.mod(H + 1/3, 1)
H2= np.mod(H + 2/3, 1)
H3= np.mod(H + 3/3, 1)

Im	ImH1
ImH2	ImH3

```
import cv2
import numpy as np
Im = cv2.imread('photo04.jpg')
K = cv2.cvtColor(Im, cv2.COLOR_BGR2HSV)
H, S, V = cv2.split(K)
H1 = np.mod(H + 1/3, 1)
H1=cv2.convertScaleAbs(H1*255)
I1 = cv2.merge([H1, S, V])
ImH1 = cv2.cvtColor(I1, cv2.COLOR_HSV2BGR)

# Affichage
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))

axes[0, 0].imshow(cv2.cvtColor(Im, cv2.COLOR_BGR2RGB))
axes[0, 0].set_title('Image originale')

axes[0, 1].imshow(cv2.cvtColor(ImH1, cv2.COLOR_BGR2RGB))
axes[0, 1].set_title('Teinte décalée de 1/3')

axes[1, 0].imshow(cv2.cvtColor(ImH2, cv2.COLOR_BGR2RGB))
axes[1, 0].set_title('Teinte décalée de 2/3')

axes[1, 1].imshow(cv2.cvtColor(ImH3, cv2.COLOR_BGR2RGB))
axes[1, 1].set_title('Teinte décalée de 3/3')
```

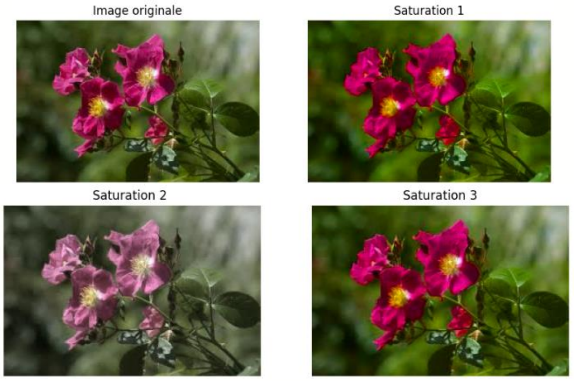
- **Résultat**

4) Afficher dans la même figure Im, ImS1, ImS2 et ImS3 Avec
S1= cv2.multiply(S, 2)
S2= cv2.multiply(S, 0.5)
S3= cv2.multiply(S, 1.5)

```
import cv2
Im = cv2.imread('photo04.jpg')
K = cv2.cvtColor(Im, cv2.COLOR_BGR2HSV)
H, S, V = cv2.split(K)
S1 = cv2.multiply(S, 2)
I1 = cv2.merge((H, S1, V))
ImS1 = cv2.cvtColor(I1, cv2.COLOR_HSV2BGR)
```

Im	ImS_1
ImS_2	ImS_3

➢ **Résultat**

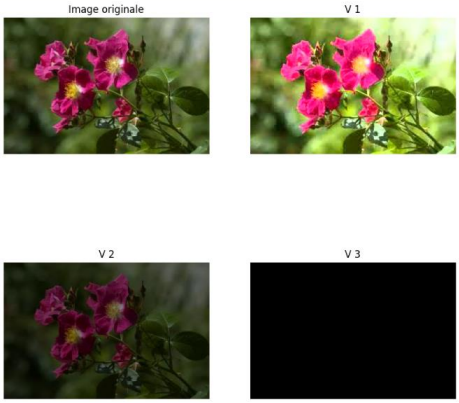


5) Afficher dans la même figure Im, ImV1, ImV2 et ImV3 Avec
V1= cv2.multiply(H, 2)
V2= cv2.multiply(H, 0.5)
V3= cv2.multiply(H, 1.5)

```
import cv2
Im = cv2.imread('photo04.jpg')
K = cv2.cvtColor(Im, cv2.COLOR_BGR2HSV)
H, S, V = cv2.split(K)
V1 = cv2.multiply(V, 2)
I1 = cv2.merge((H, S, V1))
ImV1 = cv2.cvtColor(I1, cv2.COLOR_HSV2BGR)
```

Im	ImV_1
ImV_2	ImV_3

➢ **Résultat**



Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Ecole nationale d'électronique et des télécommunications de Sfax

Traitement des images

TP2 : Outils pratiques et restauration des images

2022 – 2023

2ème année Ingénieur Génie Informatique Industrielle (2GII)

Responsable du cours:

Ali KHALFALLAH, Maitre-assistant à Enetcom de sfax

Enseignante:

Nessrine ELLOUMI, Assistante Vacataire à Enetcom de sfax

TP2 : Outils pratiques et restauration des images

Exercice 1

1) Lire l'image "photo00.bmp" (l'affecter à la variable Im) et l'afficher en utilisant la bibliothèque open Cv.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
Im=cv.imread('photo00.bmp')
```

- **Résultat**

2) Afficher la valeur du pixel (47,1) puis celle du pixel (11,256) de l'image Im

```
print("la valeur du pixel [47,1] est:", Im[47,1])
print("la valeur du pixel [11,256] est:", Im[11,256])
```

- **Résultat**

```
la valeur du pixel [47,1] est: [121 121 121]
la valeur du pixel [4,4] est: [174 174 174]
```

3) Afficher la ligne 4 et la colonne 4 de l'image Im

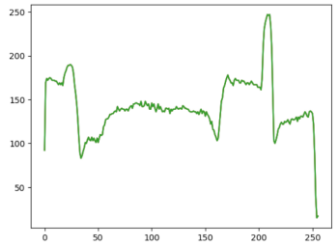
```
print("la valeur du ligne 4 est:", Im[4, :])
print("la valeur du ligne 4 est:", Im[ :, 4])
```

4) Donner le profil de la ligne 4 et la colonne 4 de l'image Im Profil de la ligne 4 :

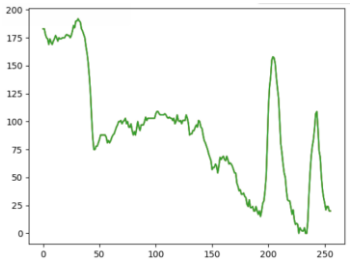
```
a=Im[4,:]
b=Im[:,4]
plt.figure(1)
plt.plot(a)
plt.figure(2)
plt.plot(b)
```

➢ **Résultat**

- Profil de la ligne 4



- Profil de la colonne 4



5) Donner le scanning de l'image Im selon le pixel (47,1) puis selon le pixel (10,255).

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
Im=cv.imread('photo00.bmp', cv.IMREAD_GRAYSCALE)
nl, nc = Im.shape
print(nl,nc)
Val=Im[47,1]
print(Val)
ImScan = np.copy(Im)
for i in range(nl):
  for j in range(nc):
        if Im[i,j]==Val:
        ImScan[i,j]=255
        else:
      ImScan[i,j]=0
plt.imshow(ImScan , cmap='gray')
plt.plot()
```

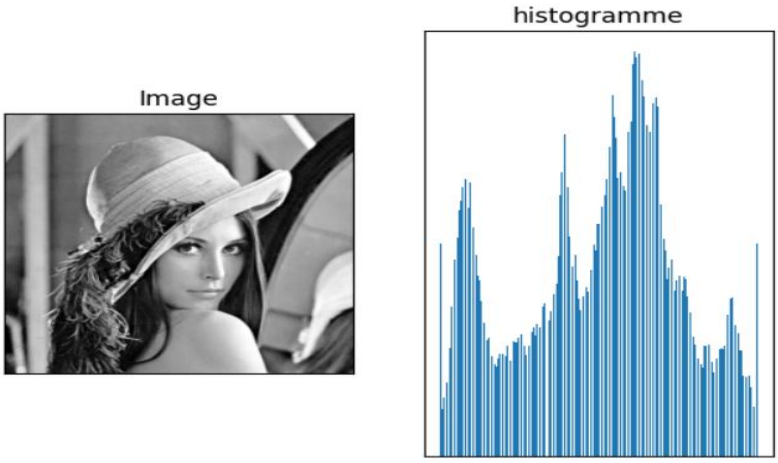
➢ Résultat Scan selon le pixel [47,1]

➢ Résultat Scan selon le pixel [10,255]

6) Donner l'histogramme de Im.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
Im=cv.imread('photo00.bmp', cv.IMREAD_GRAYSCALE)
titles = ['Im','histogramme']
plt.subplot(1,2,1),plt.imshow(Im), plt.title('Image')
plt.xticks([]),plt.yticks([])
plt.subplot(1,2,2),plt.hist(Im.ravel(),256,[0,256])
plt.title('histogramme'), plt.xticks([]),plt.yticks([])
plt.show()
```

➢ Résultat

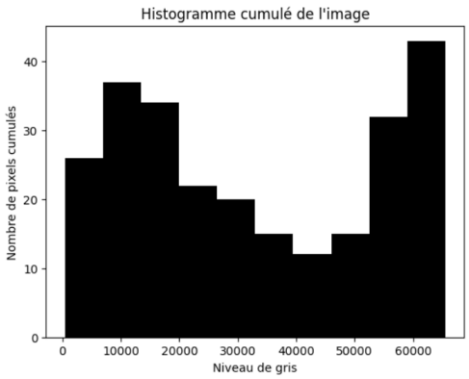


7) Donner l'histogramme cumulé de Im.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
Im=cv.imread('photo00.bmp')
hist = cv2.calcHist([Im], [0], None, [256], [0, 256])
hist_cumulative = np.cumsum(hist)
plt.hist(hist_cumulative, color='black')
plt.title('Histogramme cumulé de l\'image')
plt.xlabel('Niveau de gris')
```

```
plt.ylabel('Nombre de pixels cumulés')
plt.show()
```

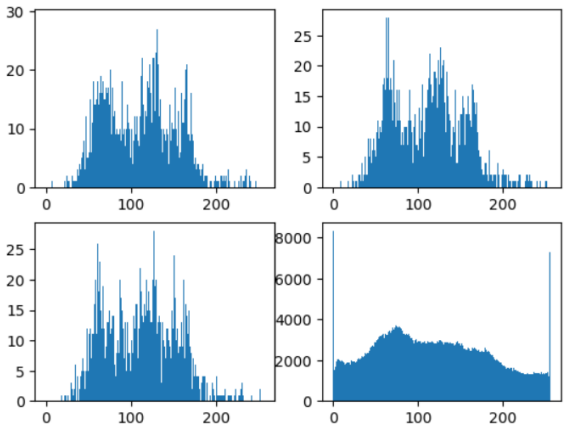
➢ Résultat



8) Donner l'histogramme de " col01.jpg" (l'affecter à la variable col)

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
Im=cv.imread('col01.jpg')

b=Im[:,:][0]
v=Im[:,:][1]
r=Im[:,:][2]
plt.figure(1)
plt.subplot(2,2,1),plt.hist(r.ravel(),256,[0,256])
#plt.figure(2)
plt.subplot(2,2,2),plt.hist(v.ravel(),256,[0,256])
#plt.figure(3)
plt.subplot(2,2,3),plt.hist(b.ravel(),256,[0,256])
#plt.figure(4)
plt.subplot(2,2,4),plt.hist(Im.ravel(),256,[0,256])
plt.show()
```

➢ Résultat



9) Calculer la brillance L et le contraste C des images "photo00.bmp", "photo01.bmp" et "photo02.bmp", (l'affecter à la variable P1, P2 et P3).

• La Luminance (ou brillance) L d'une image est définie comme la moyenne des pixels de l'image. (Une image claire admet une brillance élevée alors qu'une image sombre admet une brillance faible)

• Le contraste quantifie la différence entre les parties claires et les parties sombres de l'image.

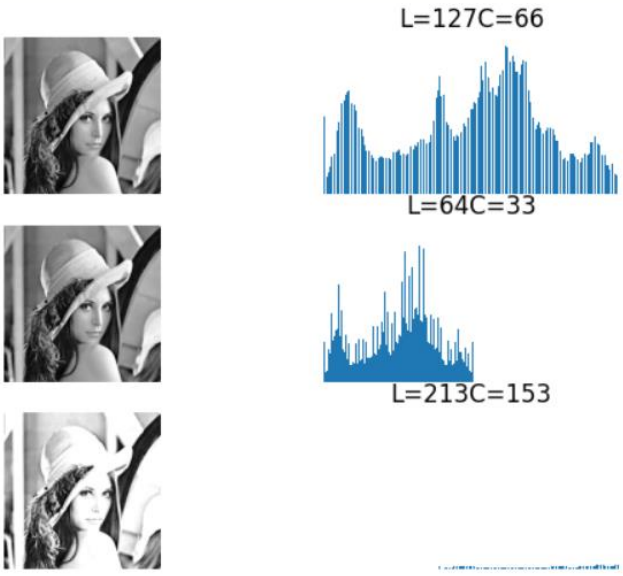
NB :

```
L1=mean(P1),
C1 = sqrt(sum(sum(P1-L1)^2))/(nl*nc)) avec nl, nc= P1.shape
```

$$C = \frac{1}{M * N} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} (1(x,y) - L)^2$$

```
import cv2 as cv
import numpy as np
from numpy import mean
import math
from matplotlib import pyplot as plt
P1=cv.imread('photo00.bmp')
P2=cv.imread('photo01.bmp')
P3=cv.imread('photo02.bmp')
nl1, nc1, channels = P1.shape
nl2, nc2, channels = P2.shape
nl3, nc3, channels = P3.shape
plt.hist(P1.ravel(),256,[0,256])
plt.show()
L1=mean(P1)
print(L1)
v=np.zeros(P1.shape)
v1=np.zeros(P1.shape)
for i in range(nl1):
  for j in range(nc1):
    v[i,j]=P1[i,j]-L1
    v1[i,j]= v[i,j]*v[i,j]
    va=sum(sum(v1)/(nl1*nc1))
    C1=math.sqrt(va)
print(C1)
```

➢ **Résultat de P1, P2, P3**



Exercice 2

Créer un fichier EX02.py pour réaliser le travail suivant :

1) Créer et afficher la matrice M et l'affecter à la variable M

```
import numpy as np
from matplotlib import pyplot as plt
M = np.array([[20, 150, 150, 50], [100, 20, 50, 30], [200, 50, 20, 3]
,[200, 200, 60, 60]])
```

➢ **Résultat**

```
[[ 20 150 150  50]
 [100  20  50  30]
 [200  50  20   3]
 [200 200  60  60]]
```

2) Expliquer le rôle de la fonction `equalizeHist`.

La fonction equalizeHist de OpenCV est utilisée pour égaliser l'histogramme d'une image en niveaux de gris. L'égalisation d'histogramme est un processus qui ajuste la distribution des intensités de l'image afin d'obtenir une distribution plus uniforme et améliorer le contraste d'une image.

3) 4. Créer et afficher la matrice Madjust

$$M_{adjust} = \frac{255 * (M - \min)}{(max - min)} \; ; min = \min(M) \; et \; max = \max(M)$$

```
m1 = np.min(M)
m2 = np.max(M)
Madjust = np.round(255 * (M - m1) / (m2 - m1)).astype(np.uint8)
```

```
print("\nMadjust:")
print(Madjust)
```

➢ **Résultat**

```
Madjust:
[[  0 184 184  42]
 [113   0  42  14]
 [255  42   0  14]
 [255 255  57  57]]
```

4) Expliquer le rôle de la fonction M_{adjust}

La fonction Madjust a pour rôle d'ajuster les valeurs de la matrice M de manière à ce qu'elles soient linéairement réparties sur toute l'échelle de gris (0 à 255).

5) Appliquer l'étirement de l'histogramme, en utilisant la fonction cv2.normalize, sur la matrice M. (l'affecter à la variable Ma)

```
m1 = np.min(M)
m2 = np.max(M)

Ma = cv2.normalize(M, None, 0, 255, cv2.NORM_MINMAX)

print("Madjust:")
print(Ma.astype(np.uint8))
```

6) Comparer visuellement la matrice Madjust et la matrice Ma.

Les résultats sont les même.

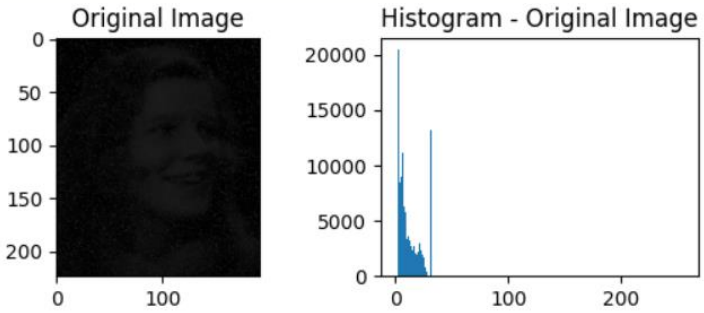
7) Lire et afficher les images "image1.bmp" et "image2.bmp" (l'affecter à la variable Im, ImG) .

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
im = cv.imread('image1.bmp')
imG = cv.imread('image2.bmp')
plt.subplot(1,2,1),plt.imshow(im),plt.show()
plt.subplot(1,2,2),plt.imshow(imG),plt.show()
```

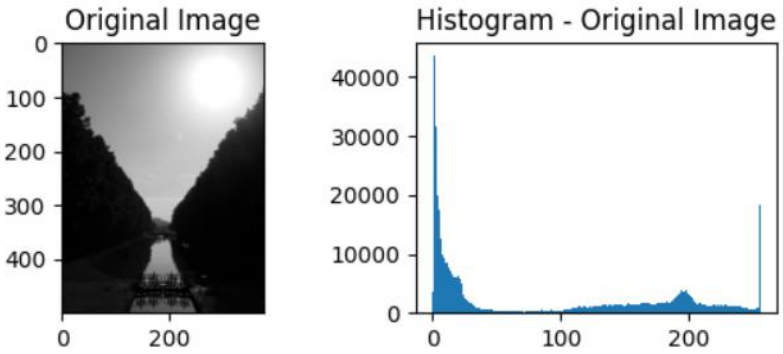
8) Calculer les histogrammes des images Im, ImG.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
im = cv.imread('image1.bmp')
plt.subplot(2,2,1),plt.imshow(im),plt.title('Original Image')
plt.subplot(2,2,2),plt.hist(im.ravel(),256,[0,256]),plt.title('Histog
ram - Original Image')
```

➢ **Résultat im**



➢ **Résultat img**



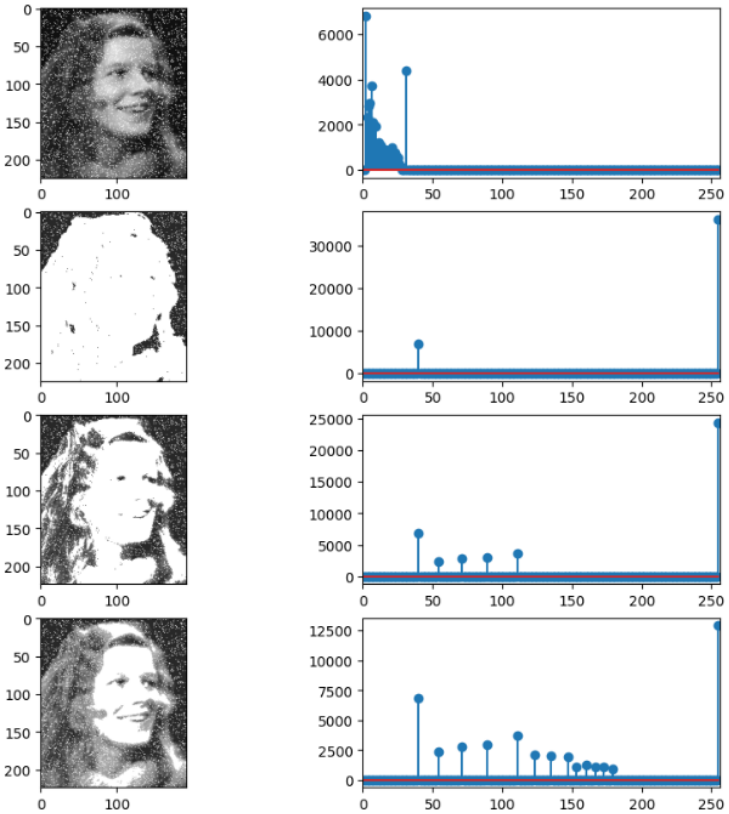
9) Comparer visuellement et en termes d'histogramme l'effet de l'égalisation de l'histogramme sur 4, 8 et 16 raies pour les images en niveaux de gris Im et ImG.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
im = cv2.imread('image1.bmp', cv.IMREAD_GRAYSCALE)
imG = cv2.imread('image2.bmp', cv.IMREAD_GRAYSCALE)


Him, _ = np.histogram(im.flatten(), 256, [0, 256])
HimG, _ = np.histogram(imG.flatten(), 256, [0, 256])


im4 = cv2.equalizeHist(cv2.convertScaleAbs(im, alpha=(255/3)))
Him4, _ = np.histogram(im4.flatten(), 256, [0, 256])

im8 = cv2.equalizeHist(cv2.convertScaleAbs(im, alpha=(255/7)))
Him8, _ = np.histogram(im8.flatten(), 256, [0, 256])


im16 = cv2.equalizeHist(cv2.convertScaleAbs(im, alpha=(255/15)))
Him16, _ = np.histogram(im16.flatten(), 256, [0, 256])

plt.figure(figsize=(10, 10))

plt.subplot(4, 2, 1)
plt.imshow(im, cmap='gray')
```

Im	H_{Im}
Im_4	H_4
Im_8	H_8
Im_{16}	H_{16}

```
plt.subplot(4, 2, 2)
plt.stem(Him)
plt.xlim([0, 256])

plt.subplot(4, 2, 3)
plt.imshow(im4, cmap='gray')

plt.subplot(4, 2, 4)
plt.stem(Him4)
plt.xlim([0, 256])

plt.subplot(4, 2, 5)
plt.imshow(im8, cmap='gray')

plt.subplot(4, 2, 6)
plt.stem(Him8)
plt.xlim([0, 256])

plt.subplot(4, 2, 7)
plt.imshow(im16, cmap='gray')


plt.subplot(4, 2, 8)
plt.stem(Him16)
plt.xlim([0, 256])
```

- **Résultat**

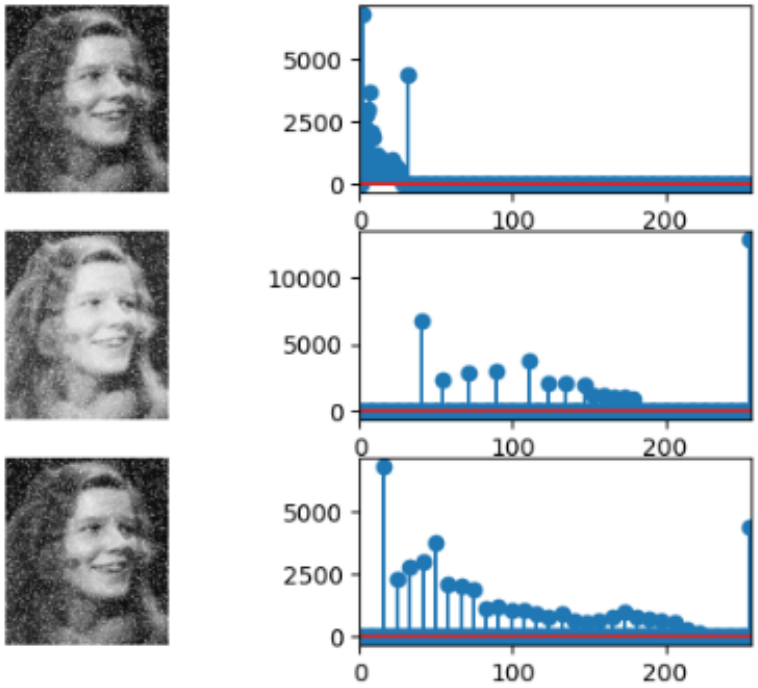


10) Afficher l'image Im, son histogramme, l'image étirée, l'histogramme de l'image étirée, l'image égalisée (R=16) et son histogramme dans une même figure

Im	H_{Im}
Im_{16}	H_{16}
Im_a	H_a

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('image1.bmp', cv2.IMREAD_GRAYSCALE)
Him, bins = np.histogram(img.flatten(), 256, [0, 256])
plt.subplot(3, 2, 2)
plt.stem(Him)
plt.xlim([0, 256])
im16 = cv2.equalizeHist(cv2.convertScaleAbs(im, alpha=(255/15)))
Him16, bins = np.histogram(img16.flatten(), 256, [0, 256])
plt.subplot(3, 2, 4)
plt.stem(Him16)
plt.xlim([0, 256])
ima = cv2.normalize(ima, None, 0, 255, cv2.NORM_MINMAX)
Hima, bins = np.histogram(ima.flatten(), 256, [0, 256])
plt.subplot(3, 2, 6)
plt.stem(Hima)
plt.xlim([0, 256])
plt.subplot(3, 2, 1)
plt.imshow(img, cmap='gray')
plt.subplot(3, 2, 3)
plt.imshow(img16, cmap='gray')
plt.subplot(3, 2, 5)
plt.imshow(ima, cmap='gray')
plt.show()
```

➢ **Résultat**



Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université de Sfax
Ecole nationale d'électronique et des télécommunications de Sfax

Traitement des images

TP 3 : Opérateurs morphologiques et segmentation des images

2022 – 2023

2éme année Ingénieur Génie Informatique Industrielle (2GII)

Responsable du cours:

Ali KHALFALLAH, Maitre-assistant à Enetcom de sfax

Enseignante:

Nessrine ELLOUMI, Assistante Vacataire à Enetcom de sfax

TP3 : Opérateurs morphologiques et segmentation des images

Exercice 1

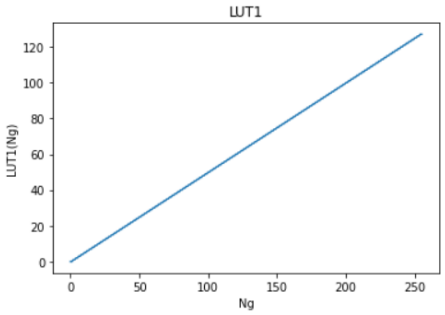
Une LUT (Look-Up Table) ou table de conversion est une fonction qui transforme un niveau de gris initial i en un niveau de gris j sans toucher la structure spatiale de l'image.

- ✓ Transformation linéaire
- ✓ Transformation linéaire avec saturation

1) Créer les tableaux de la conversion suivante

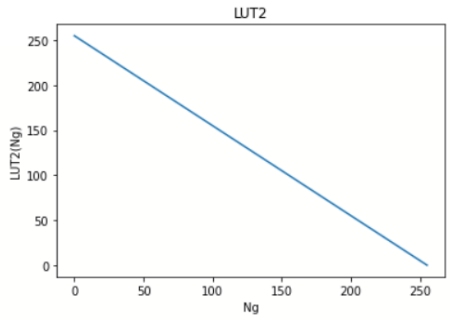
- ✓ **Transformation linéaire**

```
import matplotlib.pyplot as plt
import numpy as np
Ng = np.arange(256)
LUT1 = (0.5*Ng).astype(np.uint8)
plt.plot(Ng, LUT1)
plt.title('LUT1')
plt.xlabel('Ng')
plt.ylabel('LUT1(Ng)')
plt.show()
```



LUT1(Ng)=0.5xNg Ng=[0 255]

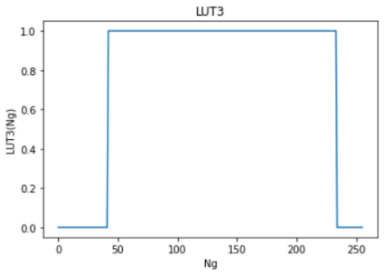
```
import matplotlib.pyplot as plt
import numpy as np
Ng = np.arange(256)
LUT2 = (255 - Ng).astype(np.uint8)
plt.plot(Ng, LUT2)
plt.title('LUT2')
plt.xlabel('Ng')
plt.ylabel('LUT2(Ng)')
plt.show()
```



LUT2 = (255 - Ng) Ng=[0 255]

- ✓ **Transformation linéaire avec saturation**

```
import matplotlib.pyplot as plt
import numpy as np
max_val = 180
min_val = 20
```



LUT3 = 255*(LUT1-min)/(max -min))

```
LUT3 = np.uint8(255*(LUT1-min_val)/(max_val-min_val))
plt.plot(LUT3)
plt.title('LUT3')
plt.xlabel('Ng')
plt.ylabel('LUT3(Ng)')
plt.show()
```

$$LUT_3(Ng) = \frac{255*(Ng - a)}{(a - b)}$$

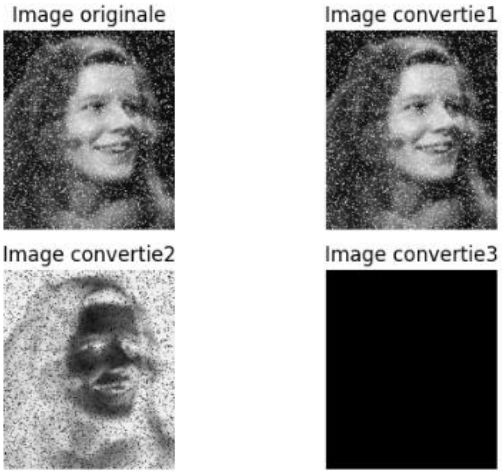
Ng=[0 255] (a=50, b=180)

2) Lire l'image "image1.bmp" (l'affecter à la variable Im).
3) Appliquer les $LUT_{1,2,3}$ sur les images en niveaux de gris image1.bmp et image2.bmp.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

Im = cv2.imread('image1.bmp', cv2.IMREAD_GRAYSCALE)

b = cv2.LUT(Im, LUT1)
c = cv2.LUT(Im, LUT2)
d = cv2.LUT(Im, LUT3)

fig, ax = plt.subplots(2, 2)
ax[0,0].imshow(Im, cmap='gray')
ax[0,0].set_title('Image originale')
ax[0,0].axis('off')
ax[0,1].imshow(b, cmap='gray')
ax[0,1].set_title('Image convertie1')
ax[0,1].axis('off')
ax[1,0].imshow(c, cmap='gray')
ax[1,0].set_title('Image convertie2')
ax[1,0].axis('off')
ax[1,1].imshow(d, cmap='gray')
ax[1,1].set_title('Image convertie3')
ax[1,1].axis('off')
plt.show()
```

Im	Im_1
Im_2	Im_3

➢ **Résultat**



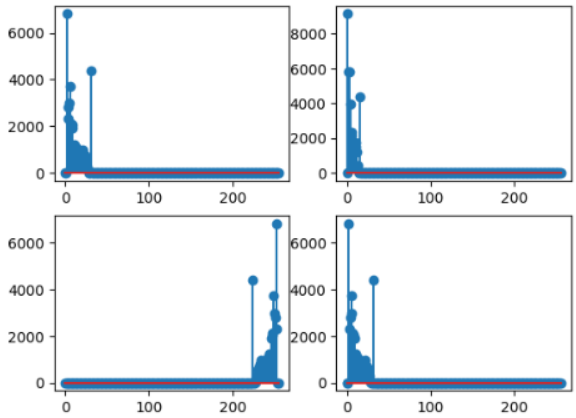
4) Afficher les histogrammes des images

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

H_{Im}	H_{Im1}
H_{Im2}	H_{Im3}

```
Ha, _ = np.histogram(a.flatten(), 256, [0, 256])
Hb, _ = np.histogram(b.flatten(), 256, [0, 256])
Hc, _ = np.histogram(c.flatten(), 256, [0, 256])
Hd, _ = np.histogram(d.flatten(), 256, [0, 256])
plt.subplot(2, 2, 1)
plt.stem(Ha)
plt.subplot(2, 2, 2)
plt.stem(Hb)
plt.subplot(2, 2, 3)
plt.stem(Hc)
plt.subplot(2, 2, 4)
plt.stem(Ha)
```

➢ **Résultat**



Exercice 2

1) Lire l'image "image3.bmp" (l'affecter à la variable Im) et l'afficher (figure 1).

2) Calculer à partir de l'image Im les images :

- ✓ Isp suite au bruit Sel & Poivre de densité 0.01
- ✓ Ipois suite au bruit Poisson
- ✓ Is suite au bruit de Speackle de variance 0.02
- ✓ Ig suite au bruit Gaussien de moyenne 0 et de variance 0.1

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
Im = cv2.imread('image1.bmp', cv2.IMREAD_GRAYSCALE)
Ig = cv2.GaussianBlur(Im, (0, 0), 0.1)
Isp = Im.copy()
cv2.randu(Isp, 0, 255)
Isp = cv2.threshold(Isp, 250, 255, cv2.THRESH_BINARY)[1]
Is = Im.copy()
cv2.randn(Is, 0, 0.02*255)
Is = cv2.add(Is, Im, dtype=cv2.CV_8UC1)
Ipois = np.random.poisson(Im)
```

3) Calculer l'erreur entre Im et chacune des images Isp, Ipois, Is et Ig. En utilisant la fonction « psnr ».

```
from skimage.metrics import peak_signal_noise_ratio
P1 = peak_signal_noise_ratio(Im, Ig)
P2 = peak_signal_noise_ratio(Im, Isp)
P3 = peak_signal_noise_ratio(Im, Is)
P4 = peak_signal_noise_ratio(Im, Ipois)
```

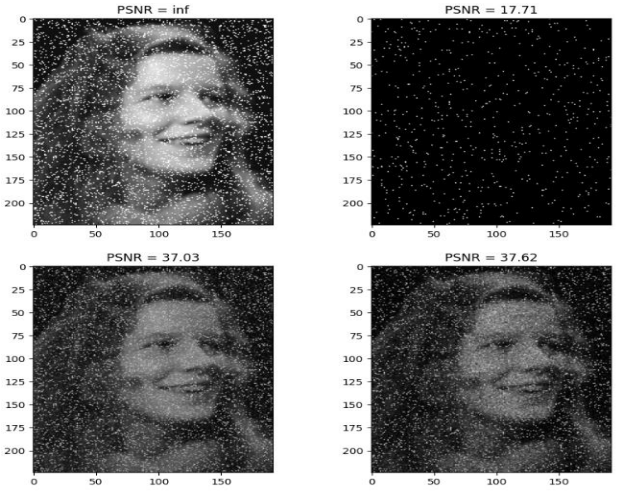
4) Afficher le résultat dans une figure qui contient les 4 images.

```
fig, ax = plt.subplots(2, 2, figsize=(10, 10))
ax[0, 0].imshow(Ig, cmap='gray')
ax[0, 0].set_title('PSNR = {:.2f}'.format(P1))
ax[0, 1].imshow(Isp, cmap='gray')
ax[0, 1].set_title('PSNR = {:.2f}'.format(P2))
ax[1, 0].imshow(Is, cmap='gray')
ax[1, 0].set_title('PSNR = {:.2f}'.format(P3))
ax[1, 1].imshow(Ipois, cmap='gray')
ax[1, 1].set_title('PSNR = {:.2f}'.format(P4))
plt.show()
```

PSNR= PSNR=

Ig	Isp
Is	Ipois

- **Résultat**



5) Appliquer un filtre médian carré 3x3 sur les images Isp, Ipois, Is et Ig en utilisant la fonction **medianBlur**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
Isp = Isp.astype(np.uint8)
Ipois = Ipois.astype(np.uint8)
Is = Is.astype(np.uint8)
Ig = Ig.astype(np.uint8)

Isp1 = cv2.medianBlur(Isp, 3)
Ipois1 = cv2.medianBlur(Ipois, 3)
```

```
Is1 = cv2.medianBlur(Is, 3)
Ig1 = cv2.medianBlur(Ig, 3)
```

6) Calculer l'erreur entre Im et chacune des images Isp1, Ipois1, Is1 et Ig1 filtré en utilisant la fonction « psnr ».

```
P1 = peak_signal_noise_ratio(Im, Ig1)
P2 = peak_signal_noise_ratio(Im, Isp1)
P3 = peak_signal_noise_ratio(Im, Is1)
P4 = peak_signal_noise_ratio(Im, Ipois1)
```

PSNR=	PSNRY1=
Isp	Fy_1
Ipois	Fy_2
Is	Fy_3
Ig	Fy_4

7) Afficher le résultat dans une figure

```
fig, ax = plt.subplots(2, 2, figsize=(10, 10))
ax[0, 0].imshow(Ig1, cmap='gray')
ax[0, 0].set_title('PSNR = {:.2f}'.format(P1))
ax[0, 1].imshow(Isp1, cmap='gray')
ax[0, 1].set_title('PSNR = {:.2f}'.format(P2))
ax[1, 0].imshow(Is1, cmap='gray')
ax[1, 0].set_title('PSNR = {:.2f}'.format(P3))
ax[1, 1].imshow(Ipois1, cmap='gray')
ax[1, 1].set_title('PSNR = {:.2f}'.format(P4))
plt.show()
```

➢ **Résultat**

8) Appliquer un filtre moyeneur carré 3x3 (h=ones(3,3)*1/9) sur les images Isp, Ipois, Is et Ig en utilisant la fonction **filter2D**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.metrics import peak_signal_noise_ratio

h = np.ones((3, 3), dtype=np.float32) / 9.0

Isp2 = cv2.filter2D(Isp, -1, h)
```

```
Ipois2 = cv2.filter2D(Ipois, -1, h)
Is2 = cv2.filter2D(Is, -1, h)
Ig2 = cv2.filter2D(Ig, -1, h)

# Calculer les PSNR pour chaque image filtrée
P1_avg = peak_signal_noise_ratio(Im, Ig2)
P2_avg = peak_signal_noise_ratio(Im, Isp2)
P3_avg = peak_signal_noise_ratio(Im, Is2)
P4_avg = peak_signal_noise_ratio(Im, Ipois2)
```

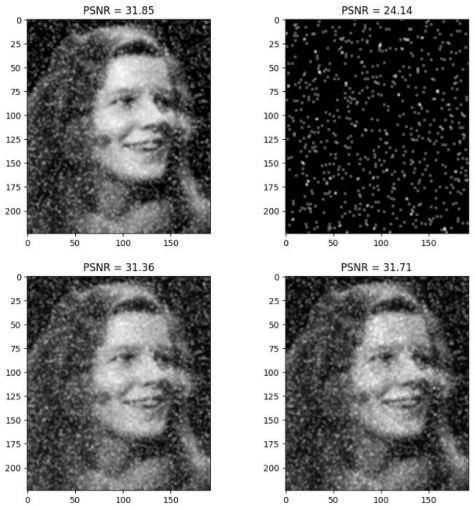
9) Calculer l'erreur entre Im et chacune des images Isp, Ipois, Is et Ig filtré en utilisant la fonction « psnr ».

```
P1_a = peak_signal_noise_ratio(Im, Ig2)
P2_a = peak_signal_noise_ratio(Im, Isp2)
P3_a = peak_signal_noise_ratio(Im, Is2)
P4_a = peak_signal_noise_ratio(Im, Ipois2)
```

10) Afficher le résultat dans une figure

```
fig, ax = plt.subplots(2, 2, figsize=(10, 10))
ax[0, 0].imshow(Ig2, cmap='gray')
ax[0, 0].set_title('PSNR = {:.2f}'.format(P1_a))
ax[0, 1].imshow(Isp2, cmap='gray')
ax[0, 1].set_title('PSNR = {:.2f}'.format(P2_a))
ax[1, 0].imshow(Is2, cmap='gray')
ax[1, 0].set_title('PSNR = {:.2f}'.format(P3_a))
ax[1, 1].imshow(Ipois2, cmap='gray')
ax[1, 1].set_title('PSNR = {:.2f}'.format(P4_a))
plt.show()
```

- **Résultat**



Exercice 3

1- Dilatation

- ✓ Lire l'image "image5.bmp" et la stocker dans la variable I5.
- ✓ Appliquer une dilatation sur I5 en utilisant des éléments structurants carrés de différentes tailles : ES carré 6×6, ES carré 7×7 et ES carré 8×8.
- ✓ Utiliser les commandes python

```
se1 = cv2.getStructuringElement(cv2.MORPH_RECT, (6, 6))
imD1 = cv2.dilate(Im, se1)
```

- ✓ Afficher sur la même figure l'image originale et les images filtrées. Interpréter les résultats.

2- Erosion

- ✓ Appliquer une Erosion sur I5 en utilisant des éléments structurants carrés de différentes tailles : ES carré 6×6, ES carré 7×7 et ES carré 8×8.
  Utiliser les commandes python.

```
se1 = cv2.getStructuringElement(cv2.MORPH_RECT, (6, 6))
imor1 = cv2.erode(Im, se1)
```

- ✓ Afficher sur la même figure l'image originale et les images filtrées. Interpréter les résultats.

3- Ouverture

- ✓ Appliquer une ouverture sur I5 en utilisant des éléments structurants carrés de différentes tailles : ES carré 6×6, ES carré 7×7 et ES carré 8×8. Utiliser les commandes python.

```
imop1 = cv2.morphologyEx(Im, cv2.MORPH_OPEN, se1)
plt.imshow(imop1)
```

- ✓ Afficher sur la même figure l'image originale et les images filtrées. Interpréter les résultats.

4- Fermeture

✓ Appliquer une fermeture sur I5 en utilisant des éléments structurants carrés de différentes tailles : ES carré 6×6, ES carré 7×7 et ES carré 8×8. Utiliser les commandes python

```
imclo1 = cv2.morphologyEx(Im, cv2.MORPH_CLOSE, se1)
plt.imshow(imclo1)
```

✓ Afficher sur la même figure l'image originale et les images filtrées. Interpréter les résultats.

5- Fermeture + Ouverture

✓ Appliquer une fermeture puis une ouverture sur I5 en utilisant des éléments structurants carrés de différentes tailles : ES carré 6×6, ES carré 7×7 et ES carré 8×8 pour la fermeture et l'élément structurant carré de taille 8×8 pour l'ouverture,
Fermeture(ES carré 6×6) + Ouverture(ES carré 8×8)
Fermeture(ES carré 7×7) + Ouverture(ES carré 8×8)
Fermeture(ES carré 8×8) + Ouverture(ES carré 8×8)

✓ Afficher sur la même figure l'image originale et les images filtrées. Interpréter les résultats.

Exercice 4

1) Lire l'image "image1.bmp" (l'affecter à la variable Im).

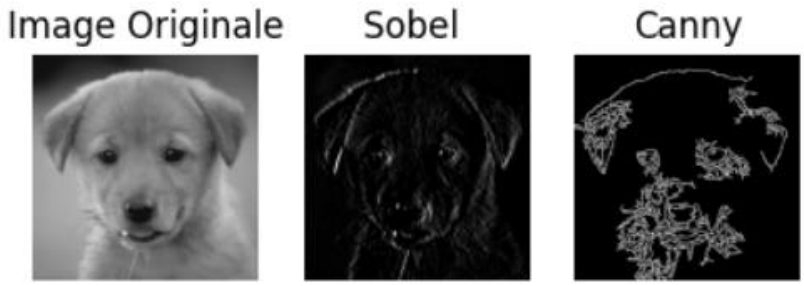
2) Réaliser la détection du contour sur l'image "image2.bmp" par le détecteur de « Prewitt », « Roberts », « Sobel » et « Canny ». Commenter les résultats.

```
import cv2
from matplotlib import pyplot as plt

Im = cv2.imread('image4.bmp', cv2.IMREAD_GRAYSCALE)
```

Im	Sobel	canny

```
Im = Im.astype('uint8')

Im1 = cv2.Canny(Im, 0, 255)
Im2 = cv2.Sobel(Im, cv2.CV_8U, 1, 0, ksize=3)
plt.figure()
plt.subplot(1, 4, 1)
plt.imshow(Im, cmap='gray')
plt.title('Image Originale')
plt.axis('off')

plt.subplot(1, 4, 2)
plt.imshow(Im2, cmap='gray')
plt.title('Sobel ')
plt.axis('off')

plt.subplot(1, 4, 3)
plt.imshow(Im1, cmap='gray')
plt.title('Canny')
plt.axis('off')
plt.show()
```

- **Résultat**



Exercice 5

1) Lire l'image "image2.bmp" (l'affecter à la variable Im)
2) Expliquer le rôle de la fonction K-means.
3) Appliquer l'algorithme K-means sur l'image « "image1.bmp"» pour extraire 2, 3, 4 puis 5 régions de l'image Im .

```
import cv2
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
Im = cv2.imread('image1.bmp', cv2.IMREAD_GRAYSCALE)
nl, nc = Im.shape
T = np.float32(Im.reshape((-1, 1)))
K2 = KMeans(n_clusters=2).fit(T)
ImK2 = np.uint8(K2.labels_.reshape(nl, nc))
K3 = KMeans(n_clusters=3).fit(T)
ImK3 = np.uint8(K3.labels_.reshape(nl, nc))
```

```
K4 = KMeans(n_clusters=4).fit(T)
ImK4 = np.uint8(K4.labels_.reshape(nl, nc))
K5 = KMeans(n_clusters=5).fit(T)
ImK5 = np.uint8(K5.labels_.reshape(nl, nc))
```

4) Commenter le résultat de la segmentation par l'algorithme K-means.

```
fig, ax = plt.subplots(2, 2)
ax[0, 0].imshow(ImK2, cmap='gray')
ax[0, 1].imshow(ImK3, cmap='gray')
ax[1, 0].imshow(ImK4, cmap='gray')
ax[1, 1].imshow(ImK5, cmap='gray')
plt.show()
```

ImK_2	ImK_3
ImK_4	ImK_5

➢ **Résultat**