R 5.A.06

2025 – 2026

Sensibilisation à la programmation multimédia

Corrigé du TD n° 2
Traitements Images

ANNE Jean-François
D'après le TD de F. DOIDY

Sensibilisation à la programmation multimédia

Programmation des images

Le but de ce TP est de se familiariser avec la programmation d'images en python.

A. Traitements de base :

Dans ce TP nous allons analyser l'impact d'une diminution de la quantification ou de l'échantillonnage d'une image. Nous verrons ensuite comment suréchantillonner une image.

1°) Exercice 1 : Lire une image et afficher ses caractéristiques :

Ecrire un programme python qui :

- Lit l'image « Lena.png »
- Afficher l'image
- Affiche la valeur du pixel situé au centre de l'image ?
- Affiche la valeur minimale des pixels de l'image, la valeur maximale des pixels de l'image ?

```
from PIL import Image
import numpy as np

# Charger l'image
image = Image.open("D:/Cours/Electronique/DUT
Info/JFA/BUT/R5.A.06/TDs/TD2/Images/Lena1.png").convert("RGB")

# Afficher l'image
display(image)

# Obtenir les dimensions de l'image
largeur, hauteur = image.size
# Trouver les coordonnées du pixel au centre de l'image
x_central = largeur // 2
y_central = hauteur // 2
pixel_central = image.getpixel((x_central, y_central))

# Convertir l'image en tableau NumPy
img_array = np.array(image)

# Calculer la moyenne des canaux RGB pour chaque pixel
avg_array = img_array.mean(axis=2)

# Trouver la valeur minimale et maximale des moyennes
min_avg_value = np.min(avg_array)
max_avg_value = np.max(avg_array)

# Trouver toutes les coordonnées des pixels ayant la plus petite et la plus grande
moyenne
coords_min_pixels = np.argwhere(avg_array == min_avg_value)
coords_max_pixels = np.argwhere(avg_array == max_avg_value)

# Récupérer les tuples (R, G, B) des pixels ayant les moyennes minimales et
maximales
```

```
min_pixel_tuples = [tuple(img_array[tuple(coord)]) for coord in coords_min_pixels]
max_pixel_tuples = [tuple(img_array[tuple(coord)]) for coord in coords_max_pixels]

# Affichage des résultats
print("Les coordonnées du pixel central sont : ", (x_central, y_central))
print("La valeur du pixel au centre de l'image est : ", pixel_central)

# Résultats pour les pixels avec la plus petite moyenne
print("\nNombre de pixels avec la plus petite moyenne RGB : ",
len(coords_min_pixels))
print("La plus petite moyenne de valeurs RGB est : ", min_avg_value)
print("Tuples de valeurs RGB des pixels ayant la plus petite moyenne : ",
min_pixel_tuples)
print("Coordonnées des pixels avec la plus petite moyenne RGB : ",
coords_min_pixels)


# Résultats pour les pixels avec la plus grande moyenne
print("\nNombre de pixels avec la plus grande moyenne RGB : ",
len(coords_max_pixels))
print("La plus grande moyenne de valeurs RGB est : ", max_avg_value)
print("Tuples de valeurs RGB des pixels ayant la plus grande moyenne : ",
max_pixel_tuples)
print("Coordonnées des pixels avec la plus grande moyenne RGB : ",
coords_max_pixels)
```

```
Les coordonnées du pixel central sont : (275, 276)
La valeur du pixel au centre de l'image est : (175, 59, 70)

Nombre de pixels avec la plus petite moyenne RGB : 1
La plus petite moyenne de valeurs RGB est : 27.0
Tuples de valeurs RGB des pixels ayant la plus petite moyenne : [(45, 0, 36)]
Coordonnées des pixels avec la plus petite moyenne RGB : [[214 286]]

Nombre de pixels avec la plus grande moyenne RGB :  1
La plus grande moyenne de valeurs RGB est : 236.33333333333334
Tuples de valeurs RGB des pixels ayant la plus grande moyenne : [(255, 243, 211)]
Coordonnées des pixels avec la plus grande moyenne RGB : [[293 127]]
```

2°) ***Exercice 2 : Effectuer une rotation de 90° de l'image :***

Ecrire un programme python qui :

- Lit l'image « Lena.png »
- Afficher l'image
- Effectuer une rotation de 90° de l'image

```
from PIL import Image
import numpy as np
```

```
# Charge l'image
image = Image.open(
    "D:\Cours\Electronique\DUT Info\JFA\BUT\R5.A.06\TDs\TD2\Images\Lena.png")

# Afficher l'image
# Afficher l'image
display(image)

# Rotation de l'image de 90 degrés
image_rotate = image.rotate(90)

# Afficher l'image
display(image_rotate)
```

3°) ***Exercice 3 : Flip et négatif***

Ecrire un programme python qui :

- Lit l'image « Lena.png »
- Afficher l'image
- Fait un Flip droite/gauche de l'image : miroir horizontal
- Et Affiche le négatif d'une image

```
from PIL import Image, ImageOps
import numpy as np

# Charge l'image
image = Image.open(
    "D:\Cours\Electronique\DUT Info\JFA\BUT\R5.A.06\TDs\TD2\Images\Lena.png")

# Afficher l'image
display(image)

# Flip droite-gauche de l'image
image_flip = image.transpose(method=Image.FLIP_LEFT_RIGHT)

# Afficher l'image
display(image_flip)

# Convertit l'image en tableau NumPy
```

```
image_array = np.array(image)

# Calcule le négatif de l'image en inversant les valeurs des pixels
image_negative = Image.fromarray(255 - image_array)

# Affiche l'image négative
display(image_negative)
```

4°) ***Exercice 4 : Qualité d'une image***

Les images cameraman1.png et cameraman2.png représentent l'image d'un photographe.

L'une de ces images est bruitée. Ecrire un programme python qui détermine laquelle est bruitée et comment quantifier ce bruit ?

Indice : calculer la moyenne des valeurs des pixels et l'écart-type des valeurs de pixels.

```
from PIL import Image, ImageOps
import numpy as np

# Charge l'image 1
image1 = Image.open(
    "D:\Cours\Electronique\DUT Info\JFA\BUT\R5.A.06\TDs\TD2\Images\cameraman1.png")
# Charge l'image 2
image2 = Image.open(
    "D:\Cours\Electronique\DUT Info\JFA\BUT\R5.A.06\TDs\TD2\Images\cameraman2.png")

# Afficher les images
display(image1)
display(image2)

# Convertir les images en tableaux Numpy
arr1 = np.array(image1)
arr2 = np.array(image2)
```

```
# Extraire la région en haut à gauche (région uniforme)
# mesure1 = arr1[0:80, 0:80]
# mesure2 = arr2[0:80, 0:80]
mesure1 = arr1
mesure2 = arr2

# Calcul de la moyenne dans la région
moyenne1 = np.mean(mesure1)
moyenne2 = np.mean(mesure2)
print(f"La moyenne de l'image1 est : {moyenne1:.2f}")
print(f"La moyenne de l'image2 est : {moyenne2:.2f}")

# Calcul de l'écart-type dans la région
std1 = np.std(mesure1)
std2 = np.std(mesure2)
print(f"L'écart type de l'image1 est : {std1:.2f}")
print(f"L'écart type de l'image2 est : {std2:.2f}")
```

```
Sur l'image complète :
                La moyenne de l'image1 est : 118.90
                La moyenne de l'image2 est : 121.44
                L'écart type de l'image1 est : 62.21
                L'écart type de l'image2 est : 70.68

Sur la région Uniforme :
                La moyenne de l'image1 est : 161.82
                La moyenne de l'image2 est : 161.88
                L'écart type de l'image1 est : 4.62
                L'écart type de l'image2 est : 43.15
```

- ⇨ La moyenne dans la région uniforme est à peu près la même dans les 2 images.
- ⇨ L'écart-type, dans la région 'uniforme', est très différent entre les deux images.

Pour une zone totalement homogène (ie de même couleur) l'écart type est nul.
Plus l'écart-type est élevé plus la valeur des pixels s'éloigne de la moyenne.
Donc, avec du bruit, l'écart type augmente : c'est donc l'image cameraman2 qui est bruitée.

5°) ***Exercice 5 : Luminosité d'une image.***

Ecrire un programme python qui demande, en entrée, la valeur de translation de gris à effectuer.

- ➢ Afficher l'image d'origine et l'image modifiée.

```python
from PIL import Image

# Charger l'image
image = Image.open("Images/cameraman1.png")

# Demander à l'utilisateur la valeur de translation de gris
```

```
translation = int(
    input("Entrez la valeur de translation de gris à appliquer : "))

# Afficher l'image originale
display(image)

print("Vous avez décalé le niveau de gris de :",translation)

# Appliquer la translation de gris
new_img = image.convert('RGB')
new_img = new_img.point(lambda x: x + translation)

# Enregistrer l'image avec la translation de gris appliquée
new_img.save("lum_lena.jpg")

# Afficher l'image avec la translation de gris appliquée
display(new_img)
```

Vous avez décalé le niveau de gris de : 30

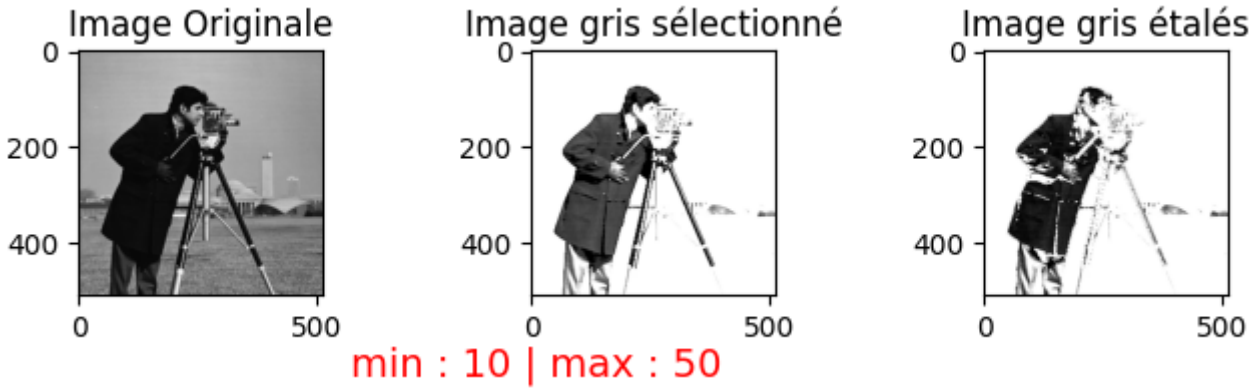
6°) ***Exercice 6 : Contraste d'une image***

Ecrire un programme python qui :

- Lit l'image « cameraman1.png »
- Qui demande en entrée la valeur minimale puis la valeur maximale de gris à conserver.
- Créer l'image qui ne contient que les pixels compris entre les 2 valeurs saisies : les pixels de valeur élevée seront mis à la valeur maximale saisie par l'utilisateur. Cela correspondra à l'image 2.
- Recréer une image (image 3) qui correspond à l'image précédente pour laquelle les valeurs de gris vont de 0 à 255.
- Afficher sur la même figure, côte à côte, l'image 1, l'image 2 et l'image3. Mettre un titre à chaque image. Indiquer, en rouge, sur l'image 2, les valeurs minimale et maximale utilisées.
- Que permet ce type de fonction ?
- Déterminer l'histogramme de l'image

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

# Charger l'image
image1 = Image.open('Images/cameraman1.png').convert('L')
arr1 = np.array(image1)

# Demander les valeurs minimale et maximale à conserver
```

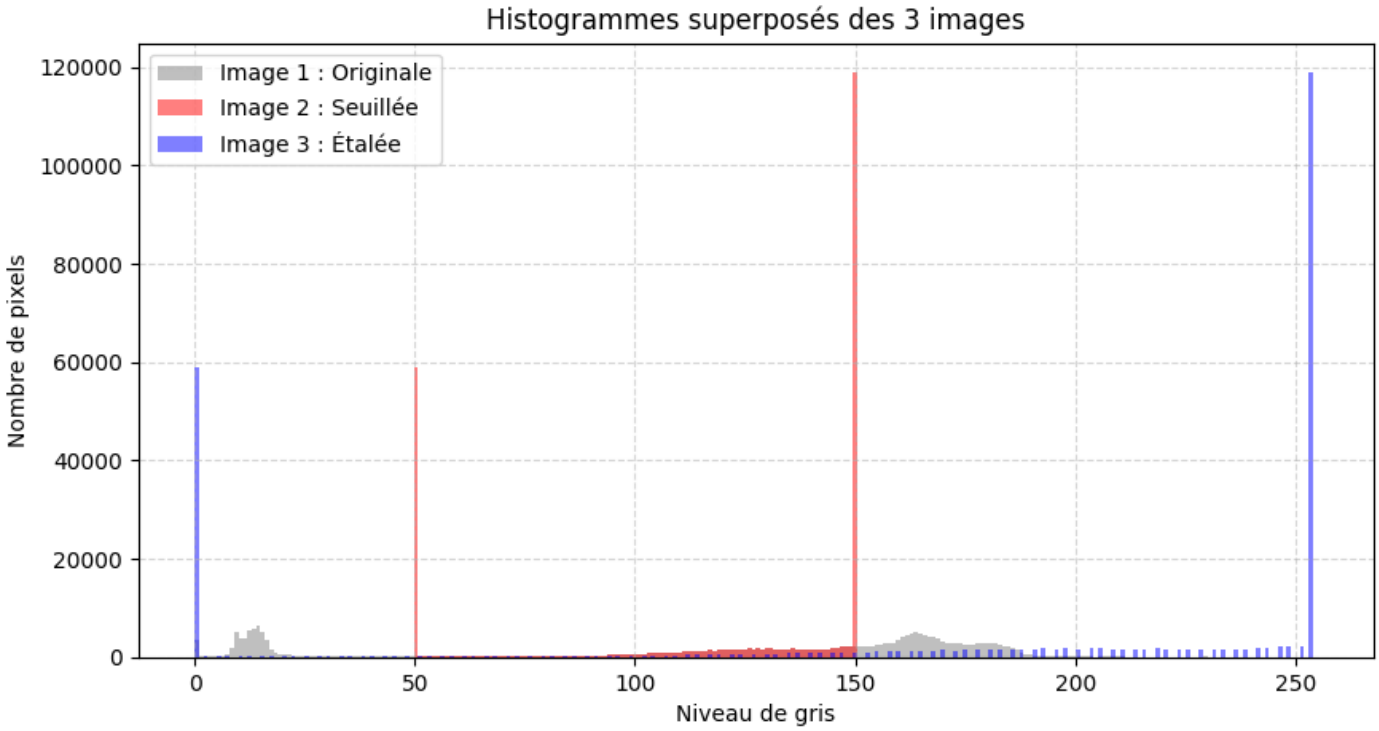
```
minimal = int(input("Valeur minimale à conserver ? "))
maximal = int(input("Valeur maximale à conserver ? "))

# Appliquer la sélection de gris
arr2 = np.zeros_like(arr1)
arr2[arr1 >= minimal] = arr1[arr1 >= minimal]
arr2[arr1 >= maximal] = maximal

# Appliquer l'étalonnage de gris
arr3 = (arr2 - minimal) * (255 / (maximal - minimal))
arr3 = np.clip(arr3, 0, 255)

# Convertir les tableaux NumPy en images PIL
image2 = Image.fromarray(arr2)
image3 = Image.fromarray(np.uint8(arr3))

# Afficher les résultats
plt.subplot(1, 3, 1)
plt.imshow(image1, cmap='gray')
plt.title('Image Originale')

plt.subplot(1, 3, 2)
plt.imshow(image2, cmap='gray')
plt.title('Image gris sélectionné')
plt.text(10, 10, f'min : {minimal} | max : {maximal}',
         color='red', fontsize=14)

plt.subplot(1, 3, 3)
plt.imshow(image3, cmap='gray')
plt.title('Image gris étalés')

plt.show()
```



a) ***Nouvelle version avec Histogramme :***

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

# --- 1. Lecture de l'image ---
image1 = cv2.imread('Images/cameraman1.png', cv2.IMREAD_GRAYSCALE)

if image1 is None:
    raise FileNotFoundError("L'image 'cameraman1.png' est introuvable dans le
dossier courant.")

# --- 2. Demande des valeurs min et max ---
vmin = int(input("Entrez la valeur minimale de gris à conserver (0-255) : "))
vmax = int(input("Entrez la valeur maximale de gris à conserver (0-255) : "))

if not (0 <= vmin < vmax <= 255):
    raise ValueError("Les valeurs doivent être dans l'intervalle 0-255 et vmin <
vmax.")

# --- 3. Création de l'image 2 (seuillage) ---
image2 = np.clip(image1, vmin, vmax)

# --- 4. Création de l'image 3 (étalement linéaire du contraste) ---
image3 = ((image2 - vmin) * (255 / (vmax - vmin))).astype(np.uint8)

# --- 5. Affichage côte à côte ---
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(image1, cmap='gray', vmin=0, vmax=255)
plt.title("Image 1 : Originale")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(image2, cmap='gray', vmin=0, vmax=255)
plt.title("Image 2 : Seuillée")
plt.axis('off')
plt.text(10, 20, f"vmin = {vmin}", color='red', fontsize=10, weight='bold')
plt.text(10, 40, f"vmax = {vmax}", color='red', fontsize=10, weight='bold')
```

```
plt.subplot(1, 3, 3)
plt.imshow(image3, cmap='gray', vmin=0, vmax=255)
plt.title("Image 3 : Étalée (0–255)")
plt.axis('off')

plt.tight_layout()
plt.show()

# --- 6. Superposition des histogrammes ---
plt.figure(figsize=(10, 5))

plt.hist(image1.ravel(), bins=256, range=(0, 255), color='gray', alpha=0.5,
label='Image 1 : Originale')
plt.hist(image2.ravel(), bins=256, range=(0, 255), color='red', alpha=0.5,
label='Image 2 : Seuillée')
plt.hist(image3.ravel(), bins=256, range=(0, 255), color='blue', alpha=0.5,
label='Image 3 : Étalée')

plt.title("Histogrammes superposés des 3 images")
plt.xlabel("Niveau de gris")
plt.ylabel("Nombre de pixels")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```



7°) ***Egalisation de l'histogramme***

L'égalisation de l'histogramme est une technique de traitement d'images utilisée pour améliorer le contraste des images. Elle consiste à redistribuer les intensités des pixels d'une image de manière à ce que l'histogramme résultant soit uniforme.

L'histogramme d'une image représente la distribution des niveaux de gris dans l'image. Si l'histogramme est concentré sur une plage restreinte de niveaux de gris, cela signifie que l'image a un faible contraste, c'est-à-dire qu'il y a peu de différence entre les niveaux de gris des pixels. L'égalisation de l'histogramme vise à étendre cette plage de niveaux de gris pour obtenir une image avec un contraste plus élevé.

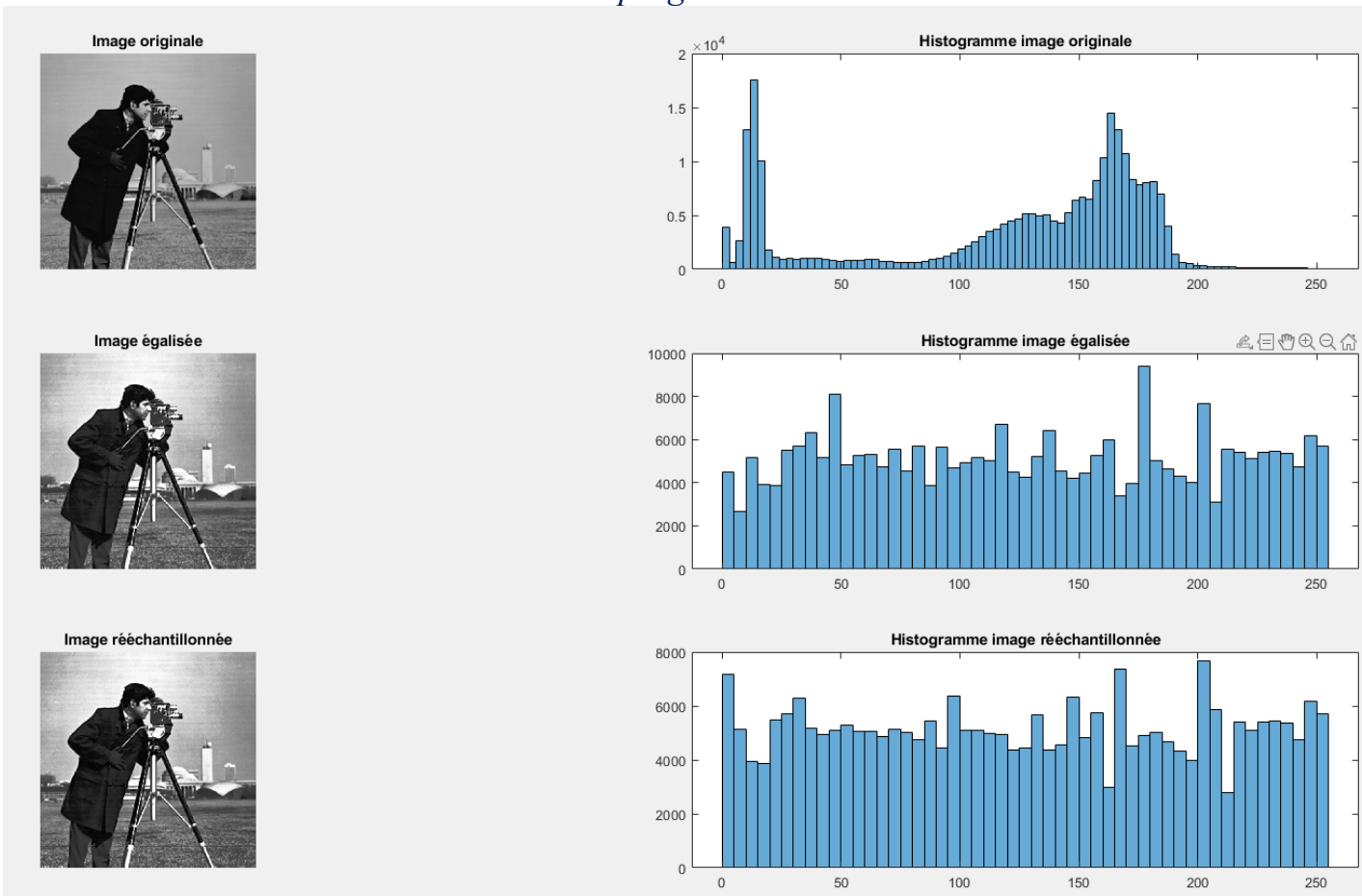
Pour égaliser l'histogramme d'une image, on calcule d'abord l'histogramme cumulatif normalisé de l'image, qui représente la proportion de pixels ayant une intensité inférieure ou égale à chaque niveau de gris. Ensuite, on calcule une transformation qui mappe chaque niveau de gris de l'image originale sur un nouveau niveau de gris, de manière à ce que la distribution des niveaux de gris dans l'image égalisée soit uniforme. Cette transformation est appliquée à tous les pixels de l'image, ce qui produit une image avec un meilleur contraste.

Il est possible d'égaliser l'histogramme, c'est-à-dire de remplacer la valeur de chaque voxel par le nombre de voxels ayant une valeur inférieure ou égale au voxel / nombre total de voxels * valeur maximal des pixels de l'image.

- Afficher les images et leurs histogrammes respectifs l'une à côté de l'autre (fonction subplot()). Mettre un titre à chaque image.

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

# Lecture de l'image
ima = np.array(Image.open('Images/cameraman1.png'))

# Calcul de l'histogramme
histo, bins = np.histogram(ima, bins=np.arange(257))
```

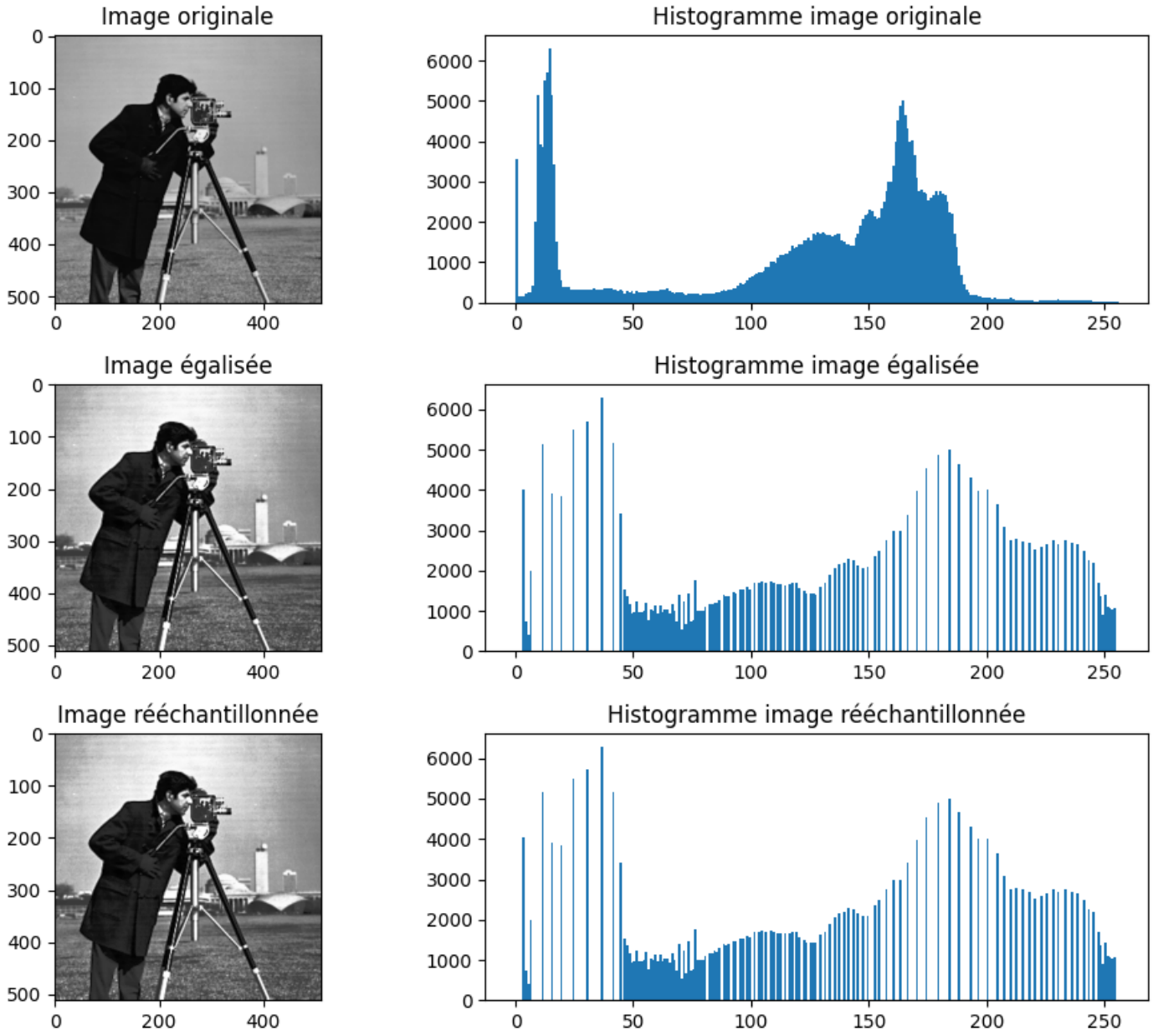
```
# Sauvegarde des valeurs de l'histogramme
values = histo.copy()

# Egalisation de l'image
ima2 = np.zeros_like(ima)
for i in range(ima.shape[0]):
    for j in range(ima.shape[1]):
        ima2[i, j] = np.sum(values[0:ima[i, j]+1])/np.sum(values)*255

# Conversion en type uint8
ima2 = np.uint8(ima2)

# Affichage des images et des histogrammes
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs[0, 0].imshow(ima, cmap='gray')
axs[0, 0].set_title('Image originale')
axs[0, 1].hist(ima.ravel(), bins=np.arange(257))
axs[0, 1].set_title('Histogramme image originale')
axs[1, 0].imshow(ima2, cmap='gray')
axs[1, 0].set_title('Image égalisée')
axs[1, 1].hist(ima2.ravel(), bins=np.arange(257))
axs[1, 1].set_title('Histogramme image égalisée')
plt.show()

# Calcul de la valeur minimale et maximale de l'image égalisée
vmin = np.min(ima2)
print(f"La valeur minimale est : {vmin}")
vmax = np.max(ima2)
print(f"La valeur maximale est : {vmax}")
```



- Quelles sont les valeurs minimale et maximale de l'image obtenue.
- Créer une nouvelle image en partant de l'image égalisée précédemment en utilisant toutes les valeurs de niveaux de gris possibles.
- Afficher les trois images et leurs histogrammes respectifs l'une à côté de l'autre. Mettre un titre aux images et aux histogrammes.
- Quel est l'intérêt de l'égalisation de l'histogramme

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

# Lecture de l'image
ima = np.array(Image.open('Images/cameraman1.png'))

# Calcul de l'histogramme
histo, bins = np.histogram(ima, bins=np.arange(257))

# Sauvegarde des valeurs de l'histogramme
values = histo.copy()

# Egalisation de l'image
ima2 = np.zeros_like(ima)
for i in range(ima.shape[0]):
    for j in range(ima.shape[1]):
        ima2[i, j] = np.sum(values[0:ima[i, j]+1])/np.sum(values)*255

# Conversion en type uint8
ima2 = np.uint8(ima2)
```

```
# Rééchantillonnage de l'image
reduced_image = np.zeros_like(ima)
for i in range(ima.shape[0]):
    for j in range(ima.shape[1]):
        reduced_image[i, j] = np.sum(ima2 <= ima2[i, j])/np.size(ima)*vmax

# Conversion en type uint8
reduced_image = np.uint8(reduced_image)

# Calcul de l'histogramme rééchantillonné
histo_reduced, bins_reduced = np.histogram(reduced_image, bins=np.arange(257))

# Affichage des images et des histogrammes
fig, axs = plt.subplots(3, 2, figsize=(10, 8))
axs[0, 0].imshow(ima, cmap='gray')
axs[0, 0].set_title('Image originale')
axs[0, 1].hist(ima.ravel(), bins=np.arange(257))
axs[0, 1].set_title('Histogramme image originale')
axs[1, 0].imshow(ima2, cmap='gray')
axs[1, 0].set_title('Image égalisée')
axs[1, 1].hist(ima2.ravel(), bins=np.arange(257))
axs[1, 1].set_title('Histogramme image égalisée')
axs[2, 0].imshow(reduced_image, cmap='gray')
axs[2, 0].set_title('Image rééchantillonnée')
axs[2, 1].hist(reduced_image.ravel(), bins=np.arange(257))
axs[2, 1].set_title('Histogramme image rééchantillonnée')

plt.tight_layout()
plt.show()

# Calcul de la valeur minimale et maximale de l'image égalisée
vmin = np.min(ima2)
print(f"La valeur minimale est : {vmin}")
vmax = np.max(ima2)
print(f"La valeur maximale est : {vmax}")
```




```
La valeur minimale est : 3
La valeur maximale est : 255
```

Quel est l'intérêt de l'égalisation de l'histogramme
Augmenter le contraste d'une image

8°) ***Exercice 8 : Images en 3 dimensions***

Dans le répertoire scanner se trouvent 175 images d'un examen médical.

Ces images correspondent à des coupes transverses (voir figure ci-dessous).

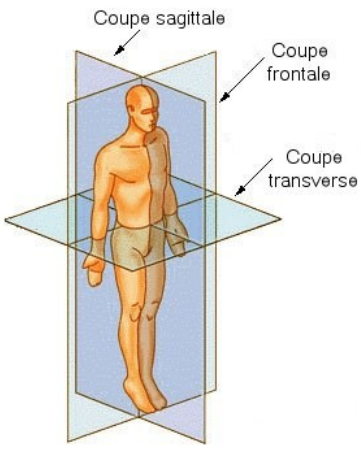
- Charger les images pour former une matrice en 3 dimensions.
- Afficher côte à côte, la coupe transverse, la coupe sagittale et la coupe frontale à une position que vous aurez saisie.

Par ex., si vous saisissez la valeur 50, alors vous afficherez :

- ✓ La coupe transverse passant par le plan (:,:,50),
- ✓ La coupe frontale passant par le plan (50, :, :),
- ✓ La coupe sagittale passant par le plan (:, 50, :).

- Mettre un titre à chaque coupe.

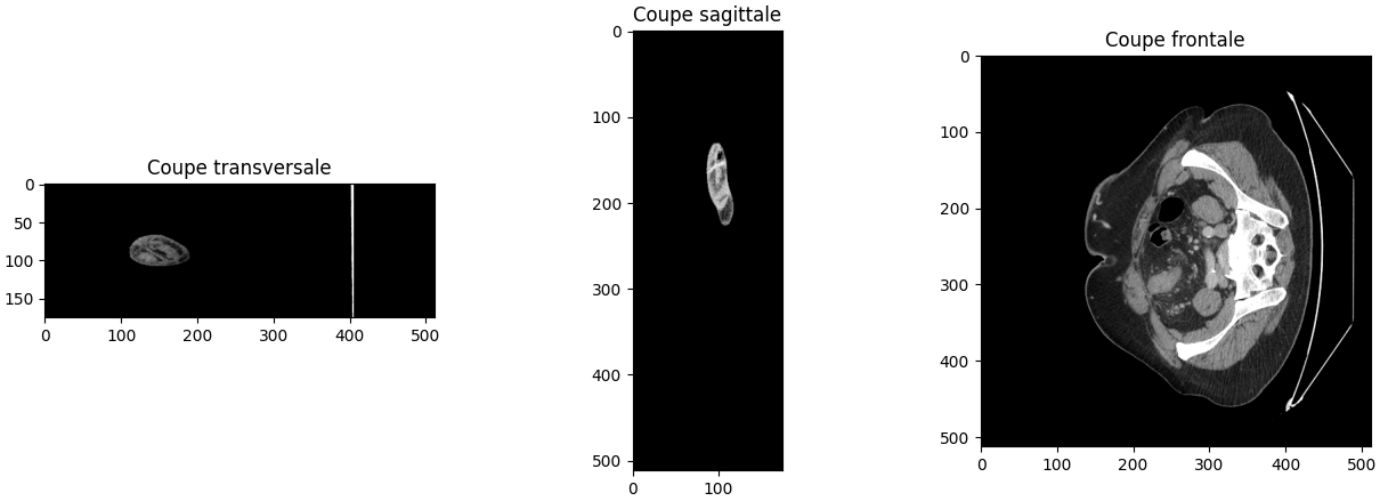
- Afficher sur l'orientation de chaque coupe :
- Coupe transverse : « gauche » à droite, « droite » à gauche, « antérieur » en haut, « postérieur » en bas.
- Coupe frontale : « antérieur » en haut, « postérieur » en bas, « gauche » à droite, « droite » à gauche
- Coupe sagittale : « supérieur » en haut, « inférieur » en bas, « antérieur » à gauche, « postérieur » à droite



Source : https://fr-academic.com/dic.nsf/frwiki/1343661

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

# Chemin du répertoire contenant les images PNG
directory = "Scanner_png"

# Liste des fichiers PNG dans le répertoire
files = [file for file in os.listdir(
    directory) if file.lower().endswith(".png")]

# Tri des fichiers dans l'ordre numérique
files = sorted(files, key=lambda x: int(os.path.splitext(x)[0]))

# Chargement des images dans une liste
volume = []
for file in files:
    image = Image.open(os.path.join(directory, file))
    volume.append(np.array(image))

volume = np.array(volume, dtype=np.uint8)

plan = int(input('Position du plan de coupe ? (0 < valeur < 175) '))

# Création des matrices passant par "plan"
# La 1ère coordonnée est en antéro-postérieur
# La 2ème coordonnée est en droite-gauche

matrice1 = volume[:, :, plan-1]
```

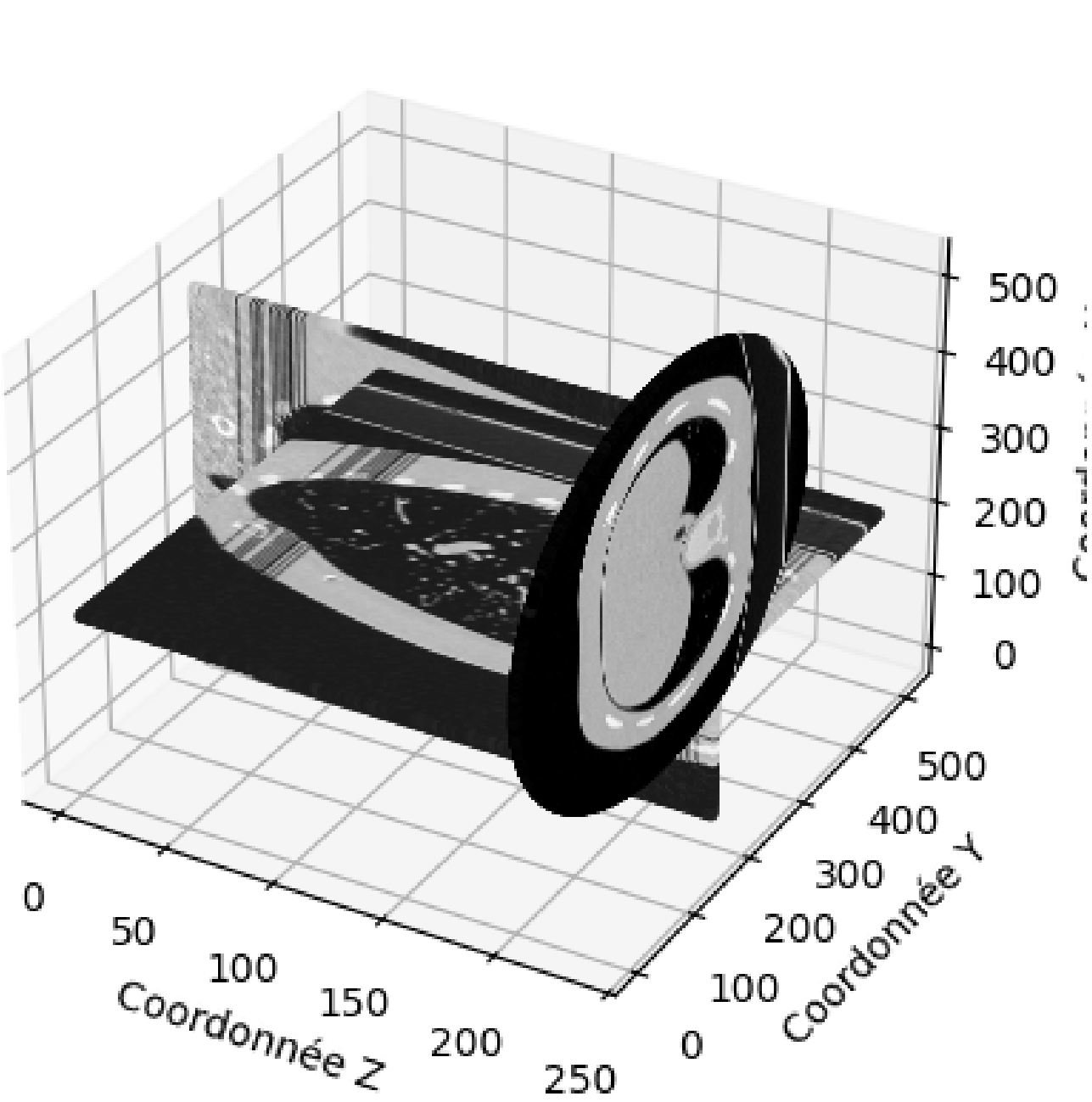
```
matrice2 = volume[:, plan-1, :]
matrice2 = np.transpose(matrice2)

matrice3 = volume[plan-1, :, :]
matrice3 = np.transpose(matrice3)

# Affichage des coupes transversale, sagittale et frontale
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(matrice1, cmap='gray')
axs[0].set_title('Coupe transversale')
axs[1].imshow(matrice2, cmap='gray')
axs[1].set_title('Coupe sagittale')
axs[2].imshow(matrice3, cmap='gray')
axs[2].set_title('Coupe frontale')

plt.show()
```



9°) ***Plan de coupe transversal en 3D :***

```
import os
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from PIL import Image
# Chemin du répertoire contenant les images PNG
directory = "new2_png"

# Liste des fichiers PNG dans le répertoire
files = [file for file in os.listdir(
    directory) if file.lower().endswith(".png")]

# Tri des fichiers dans l'ordre numérique
files = sorted(files, key=lambda x: int(os.path.splitext(x)[0]))

# Chargement des images dans une liste
volume = []
for file in files:
    image = Image.open(os.path.join(directory, file)).convert('L')
```

```
        volume.append(np.array(image))

volume = np.array(volume, dtype=np.uint8)

# Nombre d'images dans le répertoire
num_slices = len(volume)
print('Position du plan de coupe ? (0 < valeur < ', num_slices, ') ')
plan = int(input())

# Création de la grille 3D pour le plan de coupe transversal
gridx = np.zeros_like(volume)
gridy = np.zeros_like(volume)
gridz = np.zeros_like(volume)
gridx[plan-1, :, :] = volume[plan-1, :, :]
gridy[:, plan-1, :] = volume[:, plan-1, :]
gridz[:, :, plan-1] = volume[:, :, plan-1]

# Affichage de la grille 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Obtenir les coordonnées 3D
x1, y1, z1 = np.where(gridx)
x2, y2, z2 = np.where(gridy)
x3, y3, z3 = np.where(gridz)

# Obtenir les valeurs de pixel
valuesx = gridx[x1, y1, z1]
valuesy = gridy[x2, y2, z2]
valuesz = gridz[x3, y3, z3]

# Afficher les voxels
ax.scatter(x1, y1, z1, c=valuesx, cmap='gray', marker='.')
ax.scatter(x2, y2, z2, c=valuesy, cmap='gray', marker='.')
ax.scatter(x3, y3, z3, c=valuesz, cmap='gray', marker='.')

ax.set_xlabel('Coordonnée Z')
ax.set_ylabel('Coordonnée Y')
ax.set_zlabel('Coordonnée X')
ax.set_title('Plan de coupe transversal en 3D')

plt.show()
```


Plan de coupe transversal en 3D
500
400
300
200
100
0
500
400
300
200
100
0
Coordonnée Y
0
50
100
150
200
250
Coordonnée Z

B. Webographie

- *https://openclassrooms.com/fr/courses/5060661-initiez-vous-aux-traitements-de-base-des-images-numeriques*
- *https://data.lhncbc.nlm.nih.gov/public/Visible-Human/Additional-Head-Images/MR_CT_DICOM/MRI/PD/index.html*