GI2

PROJET :

TRAITEMENT D'IMAGES

Encadrée par : Prof. Abdellatif MEDOURI

Réalisée par : BELOUARRAQ Mohammed
KHALISS Hicham

QUESTION 1

Pour cette question, l'éditeur de texte **Visual Studio Code** a été utilisé. Cet outil offre une interface moderne, une grande flexibilité et un large éventail de fonctionnalités adaptées aux développeurs.
Les résultats ont été validés à travers une analyse approfondie. Par ailleurs, nous avons généré des images binaires en noir et blanc au format PBM (Portable Bit Map), spécifiquement conçu pour ce type de représentation.

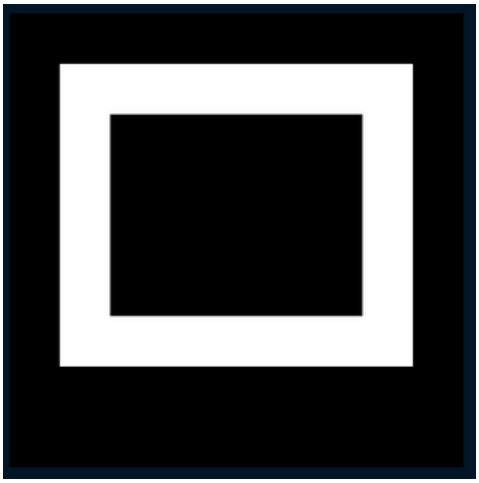
01

QUESTION 1

Lors de l'exécution du fichier quest1_2.py, les images au format BMP seront sauvegardées dans un dossier nommé Images, tandis que les matrices correspondantes seront enregistrées dans un dossier nommé BinMatrix.

```
● ❯ source venv/bin/activate
● ❯ python quest1_2.py
○ ~/Desktop/Project_VA ❯ 6s  Project_VA
```

L'image I1 :



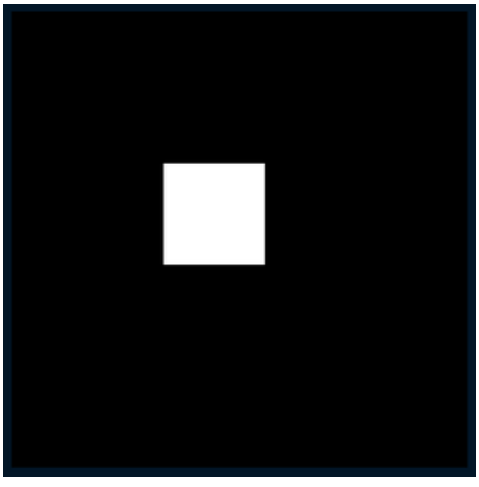
Image_I1.bmp

BinMatrix > Image_I1.txt

```
1   M1
2   # Image I1
3   9 9
4   0 0 0 0 0 0 0 0 0
5   0 1 1 1 1 1 1 1 0
6   0 1 0 0 0 0 0 1 0
7   0 1 0 0 0 0 0 1 0
8   0 1 0 0 0 0 0 1 0
9   0 1 0 0 0 0 0 1 0
10  0 1 1 1 1 1 1 1 0
11  0 0 0 0 0 0 0 0 0
12  0 0 0 0 0 0 0 0 0
13
```

Image_I1.txt

L'image I2 :



Image_I2.bmp

BinMatrix > Image_I2.txt

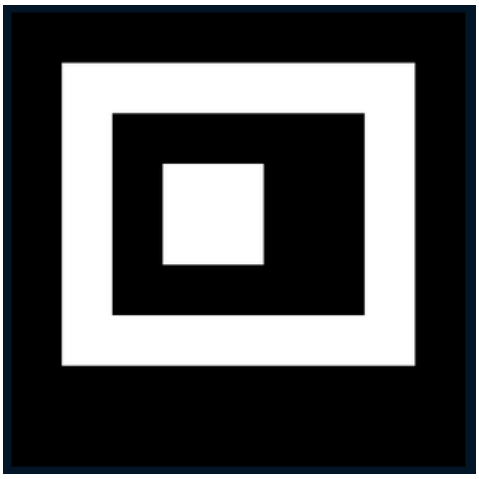
```
1   M2
2   # Image I2
3   9 9
4   0 0 0 0 0 0 0 0 0
5   0 0 0 0 0 0 0 0 0
6   0 0 0 0 0 0 0 0 0
7   0 0 0 1 1 0 0 0 0
8   0 0 0 1 1 0 0 0 0
9   0 0 0 0 0 0 0 0 0
10  0 0 0 0 0 0 0 0 0
11  0 0 0 0 0 0 0 0 0
12  0 0 0 0 0 0 0 0 0
13
```

Image_I2.txt

QUESTION 1-A

Calculons l'image Iad = I1 + I2, en utilisant la relation :
A (i, j) = min [I1(i, j) + I2(i, j) ; 1]

Apres l'execution on obtient:



Addition_Image.bmp

BinMatrix > Addition_Matrix.txt

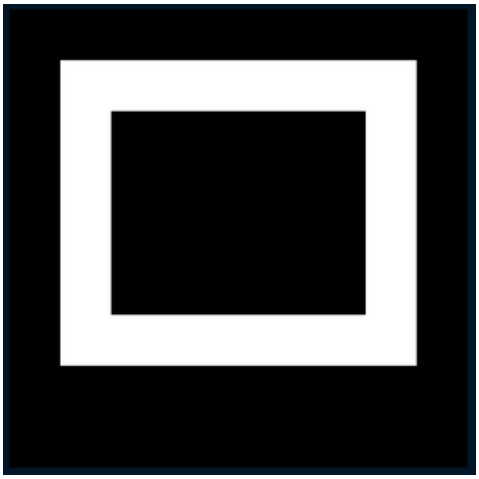
```
1   Result Image
2   # Addition of matrices I1 and I2
3   9 9
4   0 0 0 0 0 0 0 0 0
5   0 1 1 1 1 1 1 1 0
6   0 1 0 0 0 0 0 1 0
7   0 1 0 1 1 0 0 1 0
8   0 1 0 1 1 0 0 1 0
9   0 1 0 0 0 0 0 1 0
10  0 1 1 1 1 1 1 1 0
11  0 0 0 0 0 0 0 0 0
12  0 0 0 0 0 0 0 0 0
13
```

Addition_Matrix.txt

QUESTION 1-B

Calculons l'image Is = I1 - I2, en utilisant la relation :
S (i, j) = max (I1(i, j) – I2(i, j) ; 0)

Apres l'execution on obtient:



Soustraction_Image.bmp

BinMatrix > Soustraction_Matrix.txt

```
1   Result Image
2   # Soustraction of matrices I1 and I2
3   9 9
4   0 0 0 0 0 0 0 0 0
5   0 1 1 1 1 1 1 1 0
6   0 1 0 0 0 0 0 1 0
7   0 1 0 0 0 0 0 1 0
8   0 1 0 0 0 0 0 1 0
9   0 1 0 0 0 0 0 1 0
10  0 1 1 1 1 1 1 1 0
11  0 0 0 0 0 0 0 0 0
12  0 0 0 0 0 0 0 0 0
13
```

Soustraction_Matrix.txt

QUESTION 1-C

Pour calculer l'image Ip résultante de la multiplication par 2 d'une image en niveaux de gris codée sur 4 bits, nous appliquons une transformation pixel par pixel à l'aide de la relation suivante :

M (i, j) = Min (I (i, j) * 2 ; L-1))
Avec L-1=16-1=15

On a l'image suivant:

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	3	3	4	4	5	5	6	0
3	0	3	3	4	4	5	5	6	0
4	0	6	6	5	5	4	4	3	0
5	0	7	8	9	7	8	9	7	0
6	0	9	9	8	8	7	7	7	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

Image I

Après l'exécution du code, l'image résultante est générée et sauvegardée dans le dossier nommé **BinMatrix**. Cette image représente le résultat de la transformation ou du traitement appliqué, conformément aux instructions du programme.

BinMatrix > Multiplication_Matrix.txt

```
1   Result Image
2   # Multiplication of matrix I by 2
3   9 9
4    0  0  0  0  0  0  0  0  0
5    0  0  0  0  0  0  0  0  0
6    0  6  6  8  8 10 10 12  0
7    0  6  6  8  8 10 10 12  0
8    0 12 12 10 10  8  8  6  0
9    0 14 15 15 14 15 15 14  0
10   0 15 15 15 15 14 14 14  0
11   0  0  0  0  0  0  0  0  0
12   0  0  0  0  0  0  0  0  0
13
```

Multiplication_Matrix.txt

QUESTION 2

Dans cette question, nous avons écrit un programme pour générer l'histogramme d'une image couleur. Pour cela, nous avons utilisé la bibliothèque **PIL** (Python Imaging Library) pour ouvrir et manipuler des images, notamment pour accéder aux canaux de couleur (rouge, vert, bleu) via la méthode **getchannel**. À l'aide de **Matplotlib**, nous avons calculé et visualisé les histogrammes de ces canaux, qui montrent la fréquence des niveaux de couleur de 0 à 255. Chaque canal est tracé sur un même graphique avec une couleur distincte (rouge, vert, bleu), ce qui facilite l'analyse de la distribution des couleurs dans l'image. Bien que **NumPy** et **SciPy** aient été importées, elles n'ont pas été utilisées dans ce code, mais restent utiles pour des traitements mathématiques et d'images dans d'autres cas.

02

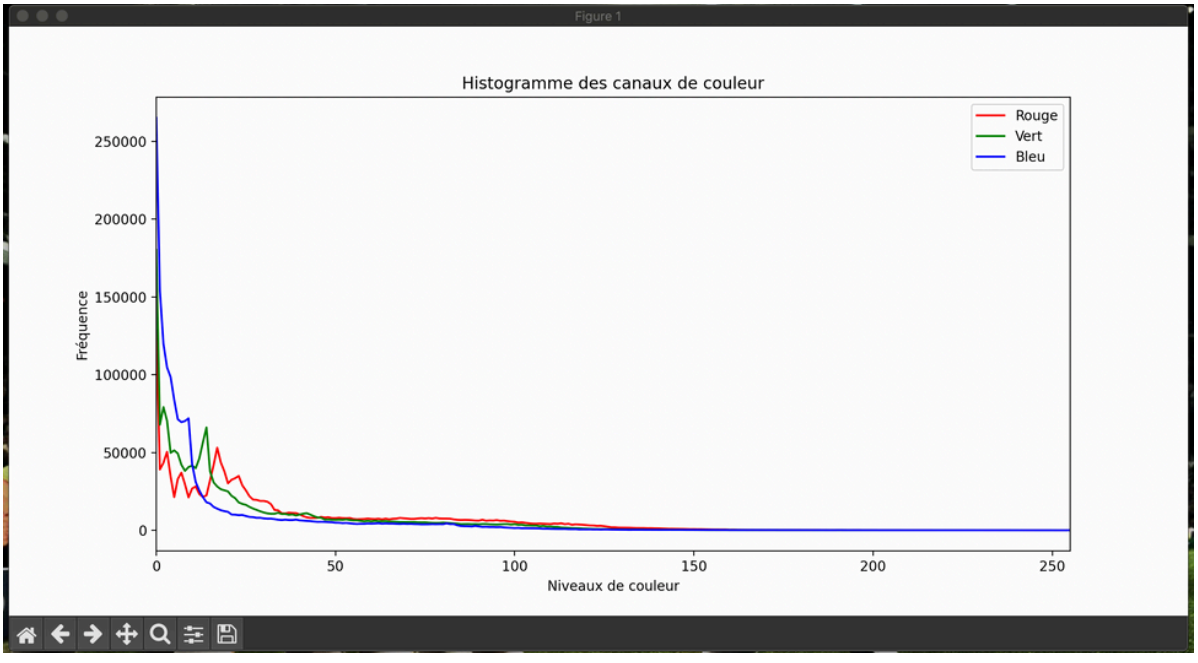
QUESTION 2

Cette partie du code génère l'histogramme des trois canaux de couleur (rouge, vert, bleu) d'une image après son chargement. En utilisant PIL pour extraire les canaux et Matplotlib pour afficher les histogrammes, chaque couleur est représentée par une barre correspondant à la fréquence des valeurs de pixel. Cela permet de visualiser la distribution des couleurs dans l'image et d'analyser sa composition en termes de luminosité et de saturation.

```python
# QUEST 2
def plot_color_histogram(image_path):
    image = Image.open(image_path)

    red_channel = image.getchannel('R')
    green_channel = image.getchannel('G')
    blue_channel = image.getchannel('B')

    red_hist = red_channel.histogram()
    green_hist = green_channel.histogram()
    blue_hist = blue_channel.histogram()

    plt.figure(figsize=(12, 6))
    plt.title("Histogramme des canaux de couleur")
    plt.xlabel("Niveaux de couleur")
    plt.ylabel("Fréquence")

    plt.plot(red_hist, color='red', label='Rouge')
    plt.plot(green_hist, color='green', label='Vert')
    plt.plot(blue_hist, color='blue', label='Bleu')

    plt.xlim([0, 255])
    plt.legend()
    plt.show()

image_path = "/Users/mbelouar/Desktop/Project_VA/Images/BELOUARRAQ.jpg"
plot_color_histogram(image_path)
```

L'image "BELOUARRAQ.jpg" se trouve dans le répertoire "**Project_VA/Images**"

Apres l'execution on obtient:



QUESTION 3

Dans cette question, nous avons créé une fonction, traiter_image.py, permettant de manipuler des images en les convertissant soit en niveaux de gris, soit en version binaire. Cette fonction offre la possibilité de choisir entre ces deux modes de conversion. Après chaque conversion, le script affiche l'image résultante, permettant ainsi d'observer la distribution des intensités des pixels. Pour cela, nous avons utilisé deux bibliothèques principales : **Pillow (PIL)**, qui nous permet d'ouvrir et de traiter les images, et **Matplotlib**, utilisée pour afficher les images et tracer les histogrammes. La conversion en niveaux de gris permet de spécifier le nombre de bits pour la quantification, influençant la qualité de l'image. Par ailleurs, la conversion binaire transforme l'image en une version où les pixels sont uniquement noirs ou blancs.

03

QUESTION 3

- Fonction **convertir_en_niveaux_de_gris**

Cette fonction prend une image, la convertit en niveaux de gris, puis ajuste les valeurs de gris en fonction du nombre de bits spécifié.

```python
def convertir_en_niveaux_de_gris(image_path, nombre_bits):
    image = Image.open(image_path).convert('L')
    image = np.array(image)
    max_val = (2 ** nombre_bits) - 1
    image = (image / 255.0 * max_val).astype(np.uint8)
    image_pil = Image.fromarray(image)
    return image_pil
```

- Fonction **convertir_en_binaire**

Cette fonction convertit l'image en une version binaire (noir et blanc) en utilisant un seuil de 128 pour séparer les pixels.

```python
def convertir_en_binaire(image_path):
    image = Image.open(image_path).convert('L')
    image = np.array(image)
    image_binaire = (image > 128).astype(np.uint8) * 255
    image_binaire_pil = Image.fromarray(image_binaire)
    return image_binaire_pil
```

- Fonction **traiter_image**

C'est la fonction principale où l'utilisateur choisit s'il souhaite une image binaire ou en niveaux de gris. L'utilisateur est également invité à entrer le nombre de bits pour la conversion en niveaux de gris. Selon le choix, la fonction appelle l'une des fonctions de conversion précédentes.

```python
def traiter_image(image_path):
    choix = input("Souhaitez-vous une image (binaire) ou (niveau de gris)? ").strip().lower()

    if choix == "niveau de gris":
        nombre_bits = int(input("Entrez le nombre de bits pour l'image en niveaux de gris (1-8
        if nombre_bits < 1 or nombre_bits > 8:
            print("Le nombre de bits doit être entre 1 et 8.")
            return None
        image = convertir_en_niveaux_de_gris(image_path, nombre_bits)
        image.show()
        image.save("image_niveaux_de_gris.bmp")
        print("Image en niveaux de gris sauvegardée sous 'image_niveaux_de_gris.bmp'.")
        return image

    elif choix == "binaire":
        image = convertir_en_binaire(image_path)
        image.show()
        image.save("image_binaire.bmp")
        print("Image binaire sauvegardée sous 'image_binaire.bmp'.")
        return image

    else:
        print("Choix invalide. Veuillez entrer 'binaire' ou 'niveau de gris'.")
        return None

traiter_image(image_path)
```

QUESTION 3

EXÉCUTION :

L'utilisateur doit choisir entre deux options : entrer '**niveau de gris**' pour convertir l'image en niveaux de gris, ou '**binaire**' pour la convertir en version binaire. L'image est déjà définie par défaut dans le code. Si l'utilisateur souhaite changer l'image, il doit modifier le chemin de l'image dans le code.

```
# QUEST 3
image_path = "/Users/mbelouar/Desktop/Project_VA/Images/BELOUARRAQ2.jpg"
```

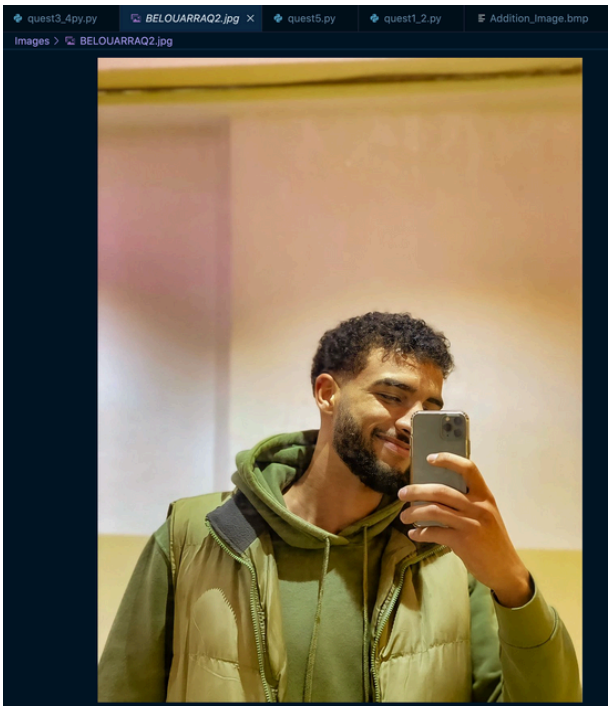
```
❯ python quest3_4py.py
Souhaitez-vous une image (binaire) ou (niveau de gris)?
```

Pour convertir une image en **niveau gris**, le programme va demander le nombre de bit ; De 1 à 8

```
❯ python quest3_4py.py
Souhaitez-vous une image (binaire) ou (niveau de gris)? niveau de gris
Entrez le nombre de bits pour l'image en niveaux de gris (1-8):
```

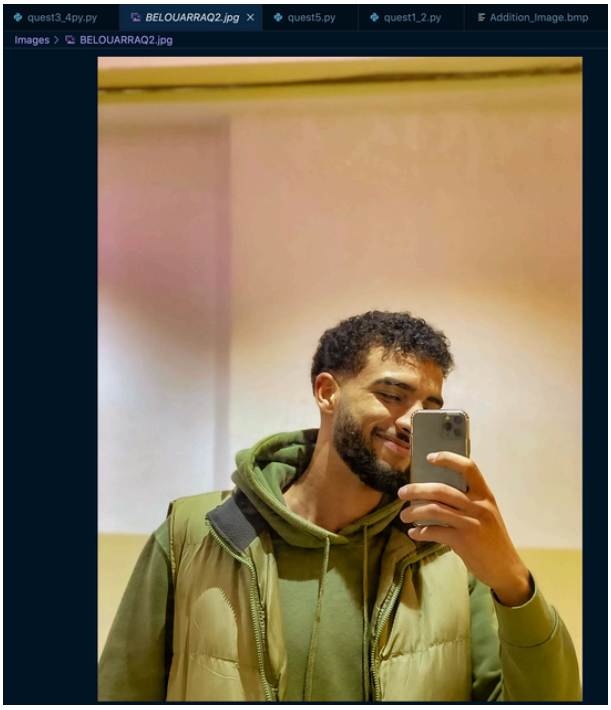
Nous allons distinguer deux cas si l'utilisateur entre '**niveau de gris**' selon le nombre de bits que l'utilisateur entre pour la conversion de l'image :

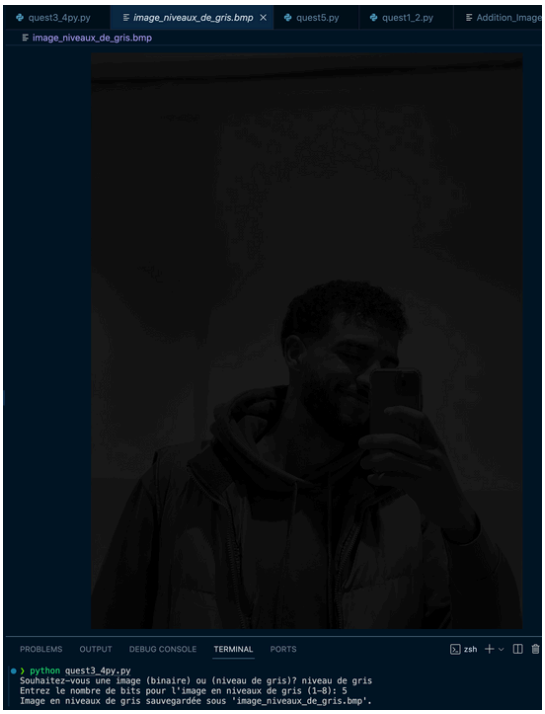
- Si l'utilisateur entre **8 bits**, l'image sera convertie en niveaux de gris avec une précision maximale, ce qui permet une gamme complète de niveaux de gris (de 0 à 255)



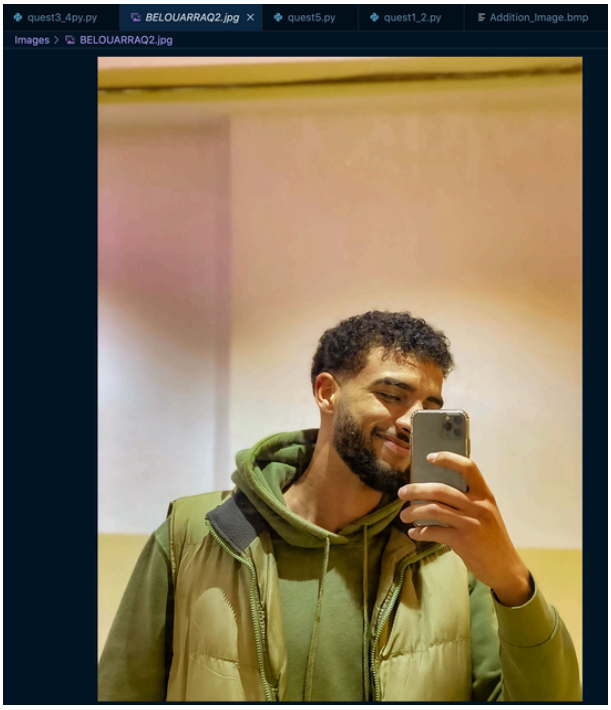
QUESTION 3

- Si l'utilisateur entre **5 bits**, l'image sera convertie en niveaux de gris avec une précision réduite, ce qui limite les niveaux de gris à une gamme plus restreinte (de 0 à 31).





Si l'utilisateur entre "**binaire**", le programme convertira l'image en une image binaire où les pixels seront soit noirs, soit blancs. Cela est réalisé en utilisant une valeur de seuil (128 dans le code)



QUESTION 4

Dans cette question, l'objectif est d'écrire un programme permettant de générer **l'histogramme** d'une image en **niveaux de gris**. En plus des bibliothèques utilisées dans les questions précédentes, nous avons intégré la bibliothèque **NumPy**. NumPy est une bibliothèque Python spécialisée dans la manipulation de matrices et de tableaux multidimensionnels, et elle offre également des fonctions mathématiques puissantes pour travailler avec ces tableaux.

04

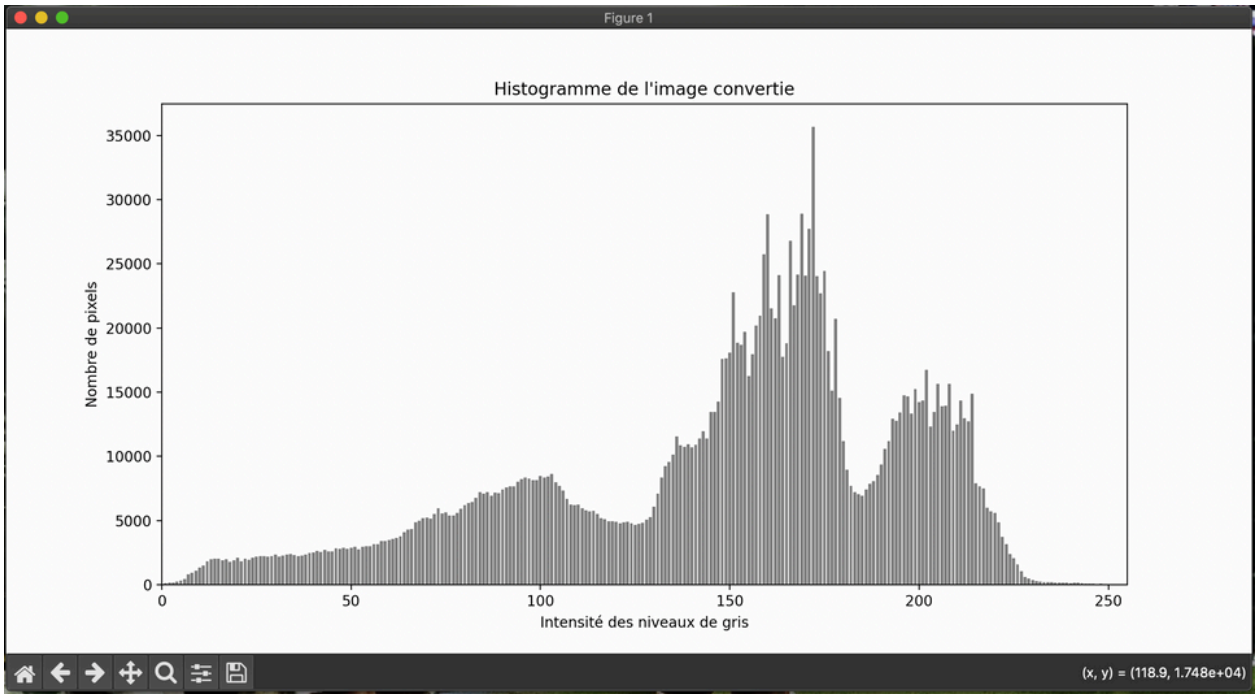
QUESTION 4

- Fonction **plot_histogram**

La fonction plot_histogram génère l'histogramme d'une image en niveaux de gris ou binaire

```python
# QUEST 4
def plot_histogram(image):
    histogram = image.histogram()
    plt.figure(figsize=(12, 6))
    plt.title("Histogramme de l'image convertie")
    plt.xlabel("Intensité des niveaux de gris")
    plt.ylabel("Nombre de pixels")

    plt.bar(range(len(histogram)), histogram, color='gray')

    plt.xlim([0, 255])
    plt.show()

converted_image = traiter_image(image_path)

if converted_image:
    plot_histogram(converted_image)
```

Apres l'execution on obtient:



QUESTION 5

Dans cette question, nous allons créer un programme destiné à appliquer le filtre de **Nagao** à une image en niveaux de gris. Ce filtre est un outil de lissage non linéaire, dont l'objectif principal est de réduire les détails dans les zones homogènes de l'image tout en accentuant la cohérence de ces zones. Il améliore ainsi la netteté des contours et le contraste des parties homogènes de l'image.
Le fonctionnement du filtre consiste à remplacer la valeur d'un pixel par la moyenne des pixels d'une fenêtre autour de celui-ci. Cette fenêtre est choisie parmi plusieurs options pré-définies.
L'image est traitée en utilisant une fenêtre de 5x5 pixels, avec le pixel à traiter positionné au centre. Autour de ce pixel, neuf sous-fenêtres de 3x3 pixels sont définies, chacune contenant neuf pixels, y compris celui à modifier.

05

QUESTION 5

La fonction **nagao_filter** applique le filtre de Nagao à un bloc de pixels de taille 5x5. Elle divise le bloc en 9 sous-régions de 3x3 pixels, incluant le pixel central.
Pour chaque sous-région, elle calcule la variance et sélectionne celle ayant la variance la plus faible (indiquant la plus grande homogénéité). Ensuite, elle remplace le pixel central du bloc par la moyenne des pixels de cette sous-région sélectionnée.

```python
# Fonction pour appliquer le filtre de Nagao
def nagao_filter(block):
    block = block.reshape((5, 5))

    regions = [
        block[:3, :3],  # Haut-gauche 3x3
        block[:3, 1:4], # Haut-centre 3x3
        block[:3, 2:],  # Haut-droite 3x3
        block[1:4, :3], # Milieu-gauche 3x3
        block[1:4, 1:4],# Centre 3x3
        block[1:4, 2:], # Milieu-droite 3x3
        block[2:, :3],  # Bas-gauche 3x3
        block[2:, 1:4], # Bas-centre 3x3
        block[2:, 2:]   # Bas-droite 3x3
    ]

    # Calculer la variance et la moyenne de chaque région
    min_variance = float('inf')
    mean_value = 0
    for region in regions:
        variance = np.var(region)
        if variance < min_variance:
            min_variance = variance
            mean_value = np.mean(region)

    return mean_value

input_image_path = "/Users/mbelouar/Desktop/Project_VA/image_niveaux_de_gris.bmp"
output_image_path = "/Users/mbelouar/Desktop/Project_VA/Images/nagao_filtered_image.png"

try:
    # Ouvrir l'image et convertir en niveaux de gris
    image = Image.open(input_image_path).convert('L')

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

    # Appliquer le filtre de Nagao
    filtered_image_array = generic_filter(image_array, nagao_filter, size=(5, 5))

    # Convertir le tableau filtré en image
    filtered_image = Image.fromarray(filtered_image_array.astype(np.uint8))

    # Sauvegarder et afficher l'image
    filtered_image.save(output_image_path)
    filtered_image.show()

    print(f"L'image filtrée a été sauvegardée sous '{output_image_path}'.")

except Exception as e:
    print(f"Erreur lors du traitement : {e}")
```

QUESTION 5

Apres l'execution
on obtient:

l'exécution de ce programme **peut prendre un certain temps**, car le filtre de Nagao doit traiter chaque pixel de l'image en analysant les fenêtres 5x5 autour de chaque pixel