Travaux Pratiques: Traitement d'images

Compte rendu

RÉALISER PAR:

KARIMALLAH Nouhaila

SOUS L'ENCADREMENT :

Dr. MEDOURI Abdellatif

ANNÉE UNIVERSITAIRE : 2021 – 2022

Introduction :

Dans le cadre de notre projet en module Vision artificielle nous nous sommes intéressés au traitement d'images par le langage de programmation ‘Python’. L’objectif de ce projet était d'étendre nos connaissances scientifiques et technologiques.

Comme nous avons pu nous en rendre compte, le traitement d'images associe l'informatique et les mathématiques. Il permet de modifier une image ou d'en extraire de l'information. Pour l'ensemble du groupe, il s'agissait d'une complète découverte du langage de programmation Python. Notre premier objectif fut de découvrir Python et ainsi de comprendre les méthodes utilisées en python pour manipuler et afficher les images.

Méthodologie de travail :

Il existe de nombreuses manipulations que nous avons exploré tout au long la réalisation de ce rapport, que l'on peut classer comme suit:

- Composition : assembler différentes images pour constituer une nouvelle image;
- Traitement : modification d'une image donnée en couleur par une image binaire ou en niveau de gris ;
- Analyse : extractions d'informations à l’aide des outils de traitement d’une image (Histogramme) ;
- Amélioration : modification de l'image dans le but de la rendre plus agréable à l'œil.

Pour modifier ses images nous avons programmé en Python les différents algorithmes de traitement.

Question 1:

Avec un éditeur de texte, on construit au format BPM les deux images binaires suivantes :

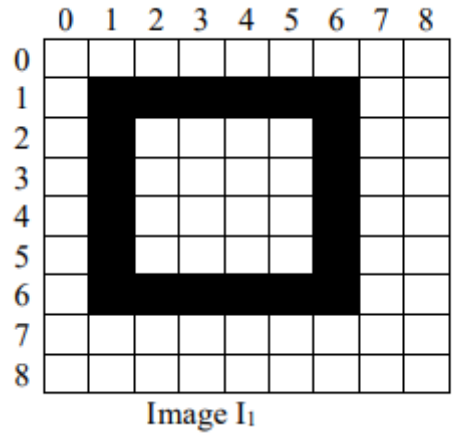


Image I_1

Image-I1 - Bloc-notes

Fichier Edition Format Affichage Aide

```
P4
9 9
111111111
100000011
101111011
101111011
101111011
101111011
100000011
111111111
111111111
```

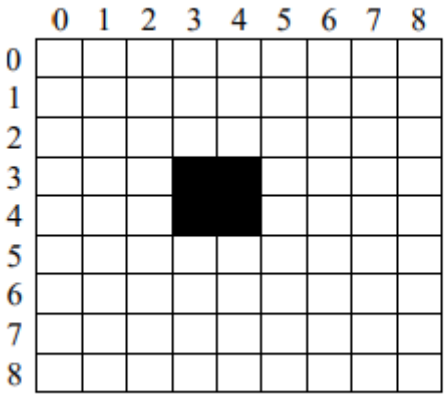


Image I_2

Image-I2 - Bloc-notes

Fichier Edition Format Affichage Aide

```
P4
9 9
111111111
111111111
111111111
111001111
111001111
111111111
111111111
111111111
111111111
```

a- Ensuite nous avons additionné les deux images de la façon suivante :

```
img1=[]
img2=[]

with open('Image-I1.txt') as file:
    for line in file :
        img1.append(line.rstrip('\n'))

with open('Image-I2.txt') as file:
    for line in file :
        img2.append(line.rstrip('\n'))


entete = img1[:2]
entetec=""
for i in range(2):
    entetec += ''.join(str(e) for e in entete[i])
```

```
img1=img1[2:]
img2=img2[2:]


for j in range(0,9):
    l = list(img1[j].strip(" "))
    for i in range(0,9) :
        l[i] = int(l[i])
    img1[j]=l


for j in range(0,9):
    l = list(img2[j].strip(" "))
    for i in range(0,9) :
        l[i] = int(l[i])
    img2[j]=l


img1=np.array(img1)
img2=np.array(img2)


img3=img1+img2
for i in range(len(img3)):
    for j in range(len(img3[i])):
        if img3[i][j] > 1 :
            img3[i][j] = 1

img3l=img3.tolist()
img3lc= [""]*len(img3l)

for i in range(len(img3l)):
    for j in range (len(img3l[i])):
        img3lc [i] = ''.join(str(e) for e in img3l[i])


with open('Image-Iad.txt', 'w') as f:

    f.writelines('\n'.join(entete+img3lc))
```

Le codage généré de l'image résultante sera ainsi :

Il s'agit d'une image contenant que des pixels en couleur blanc.

b- Puis nous avons effectué la soustraction de ces deux images:

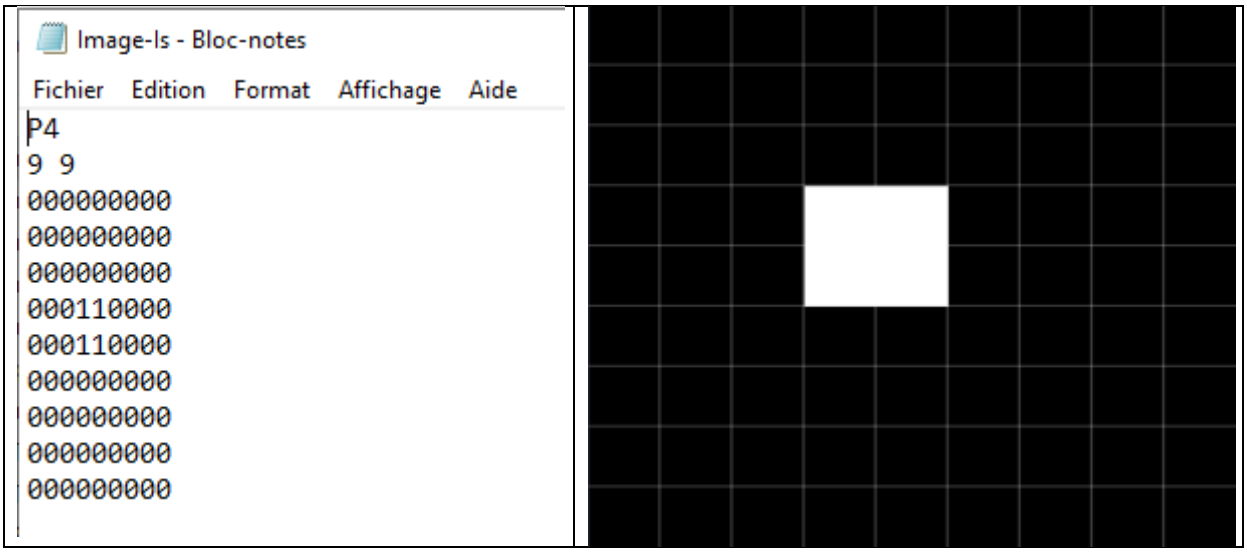
```
img4=img1-img2
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if img4[i][j] < 0 :
            img4[i][j] = 0

img4l=img4.tolist()
img4lc= [""]*len(img4l)

for i in range(len(img4l)):
    for j in range (len(img4l[i])):
        img4lc [i] = ''.join(str(e) for e in img4l[i])

with open('Image-Is.txt', 'w') as f:
    f.writelines('\n'.join(entete+img4lc))
```

En sauvegardant toujours le résultat sous format .txt nous avons obtenu comme résultat :



c- Enfin nous allons multiplier les deux images I1 et I2, pour réaliser cela nous exécutons l'algorithme suivant :

```
l=['P5',  '9  9','15','000000000',  '000000000',  '033445560',  '033445560',  '066554430',  '078978970',
'099887770', '000000000','000000000' ]
with open('Image-I.txt', 'w') as f:
    f.writelines('\n'.join(l))

imgi=[]
with open('Image-I.txt') as file:
    for line in file :
        imgi.append(line.rstrip('\n'))

entete2 = imgi[:3]
entetec2=""
for i in range(3):
    entetec2 += ''.join(str(e) for e in entete2[i])

imgi=imgi[3:]

for j in range(len(imgi)):
    l = list(imgi[j].strip(" "))
    for i in range(0,9) :
        l[i] = int(l[i])
    imgi[j]=l

imgi=np.array(imgi)
print(imgi)
imgi=imgi*2
print(imgi)
for i in range(len(imgi)):
    for j in range(len(imgi[i])):
```

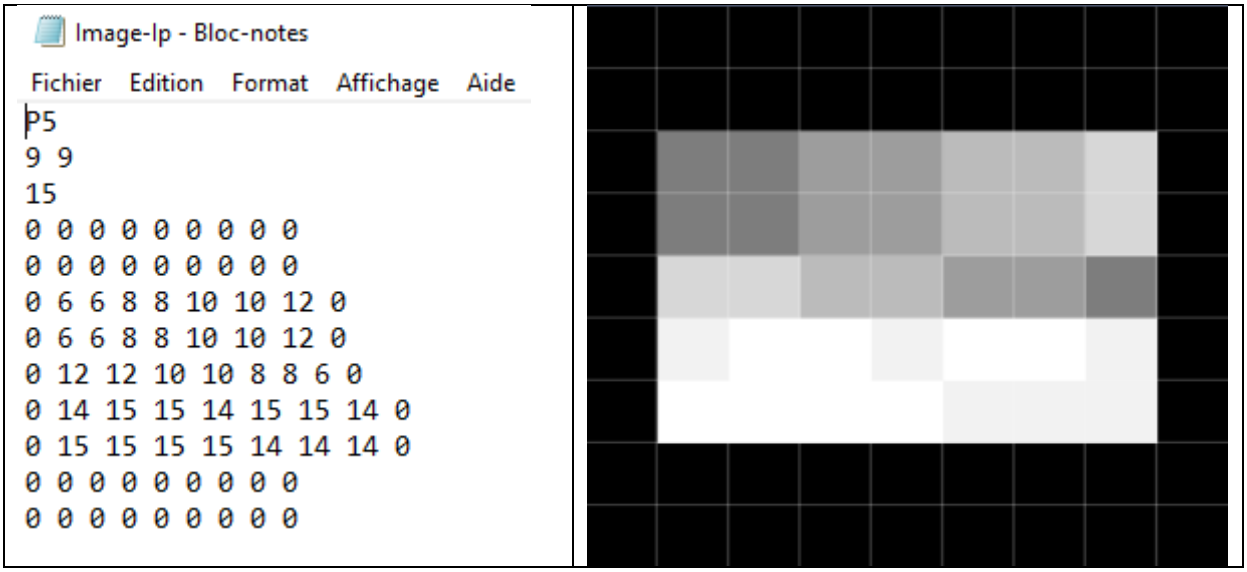
```
        if imgi[i][j] > 15 :
            imgi[i][j] = 15

print(imgi)

imgil=imgi.tolist()
imgilc= [""]*len(imgil)
for i in range(len(imgil)):
    for j in range (len(imgil[i])):
        imgilc [i] = ' '.join(str(e) for e in imgil[i])

with open('Image-Ip.txt', 'w') as f:
    f.writelines('\n'.join(entete2+imgilc))
```

Le résultat de la procédure s'agit d'une image en niveau de gris codé sur 4 bits :



Question 2 :

Afin de tracer un histogramme un histogramme nous utilisant la bibliothèque Matplotlib qui est destinée à tracer et visualiser des données sous formes de graphiques.

En utilisant les différentes méthodes mentionnées dans le code ci-dessous :

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

# lien
image = Image.open('Omar.JPG')
imageTable = np.asarray(image)

plt.subplot(211)
plt.imshow(image)

color = ('r', 'g', 'b')
for i, col in enumerate(color):
    histograme = cv2.calcHist([imageTable], [i], None, [255], [0, 255])
    plt.subplot(212)
    plt.plot(histograme, color=col)
    plt.xlim([0, 255])
plt.show()
```

On aura l'histogramme de l'image couleur du portrait suivant :

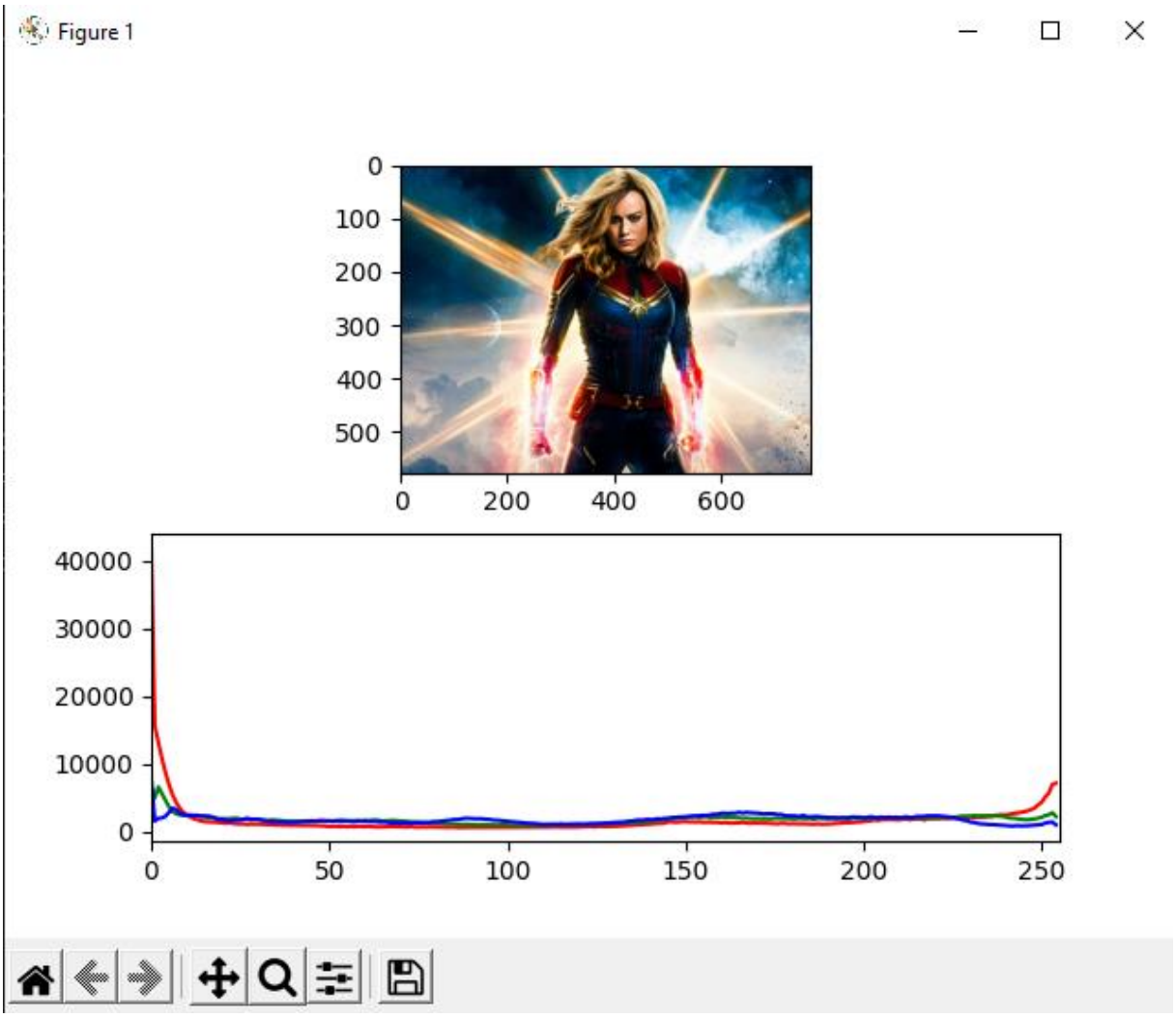


Figure 1:Histogramme Image Couleur

Question 3 & 4 :

Dans le but de transformer une image en utilisant le langage python, nous allons implémenter les fonctionnalités de la bibliothèque cv2 qui permet de diverses fonctionnalités de traitement d'image. Elle dispose de capacités de traitement d'images relativement puissantes, et possède des fonctions utilisable en traitement d'images, notamment : elle offre un support pour quelques fonctions de bases telles que le filtrage, la convolution ou encore la conversion d'espaces couleurs.

En effet pour convertir une image on utilise la fonction cvtColor (param1,param2) prenant deux paramètres le premier est l'image originale et le deuxième est le type de conversion désiré sous forme de code:

- Pour **convertir** de l'espace couleur **RGB** en niveau de **gris** on utilise le code : **cv2.COLOR_BGR2GRAY**.
- Pour **convertir en image binaire**, on utilise le seuillage avec l'opération de 'threshold', qui prend en entrée l'image à transformer **en niveaux de gris** et une valeur de **seuil**.

Comme on part d'une image en niveaux de gris, nous considérerons la valeur du seuil 127, qui se situe

au milieu de l'échelle des valeurs que peut prendre un pixel en niveau de gris (de 0 à 255).

Après la transformation de l'image en une image en niveau de gris ou en image binaire.

Figure 2: image originale en couleur

Figure 3: Image en niveau de gris

Figure 4: Image Binaire

En utilisant la bibliothèque **OpenCV** du langage Python comme le montre le code suivant:

```
histo = input("Voulez vous afficher l'histogramme de l'image convertit en Noir&Blanc (NB) ou en niveau de gris (NG)")
```

```
if(histo == 'NB' or histo == 'NG'):
   if(histo == 'NB'):
      imageOrig = cv2.imread("nouhaila.jpg", cv2.IMREAD_GRAYSCALE)
      #resizing an image
      imageOrig = cv2.resize(imageOrig,(600,600))
      # Conversion en image binaire
      imgconvertie = cv2.threshold(imageOrig, 127, 255,cv2.THRESH_BINARY)[1]
      cv2.imshow('Noir & Blanc',imgconvertie)
      cv2.waitKey(0)
   else :
      imageOrig = cv2.imread("nouhaila.jpg")
      #resizing an image
      imageOrig = cv2.resize(imageOrig,(600,600))
      #convertir vers une image en niveau de gris
      imgconvertie = cv2.cvtColor(imageOrig, cv2.COLOR_BGR2GRAY)
      cv2.imshow('Niveau de gris',imgconvertie)
      cv2.waitKey(0)
```

Figure 5: Transformation en image binaire et en niveau de gris en utilisant le langage Python (Bibliothèque : OpenCV)

On fait appel à la bibliothèque **matplotlib** pour créer l'histogramme associé à chaque transformation (en image binaire ou en image en niveau de gris) via la **méthode** plt.hist(x,bins,range,align,orientation,rwidth,log,color,edgecolor,label) **de la bibliothèque** matplotlib.pyplot **as** plt .

Histogramme :

L'histogramme d'une image mesure la distribution des niveaux de gris dans l'image.

Il est construit de la manière suivante: pour chaque niveau de gris, on compte le nombre de pixels ayant sa valeur. On utilise ainsi la boucle for suivante pour créer une table H contenant les occurrences de chaque niveau de gris. A chaque valeur de niveau de gris compris entre 0 et 255, associe le nombre de pixels ayant cette valeur.

```
s = imgconvertie.shape    #obtenir la taille de l'image
H = np.zeros(shape=(256,1))  #créer un tableau H contient 256 cases tout en les initialisant par des zéros

for i in range(s[0]):       #parcourir les colonnes
   for j in range(s[1]):      #parcourir les lignes
      k = imgconvertie[i,j]
      H[k,0]= H[k,0]+1

plt.hist(H,range=(0,256),bins=20,align="mid",rwidth=0.9,color="b",edgecolor="red",label="occurrence")
plt.title("Histogramme")
plt.xlabel("niveau de gris")
plt.show()
cv2.waitKey(0)
```

Figure 5: Construire l'histogramme d'une image en utilisant le langage Python (Bibliothèque : matplotlib)

Enfin on obtient les histogrammes suivants :

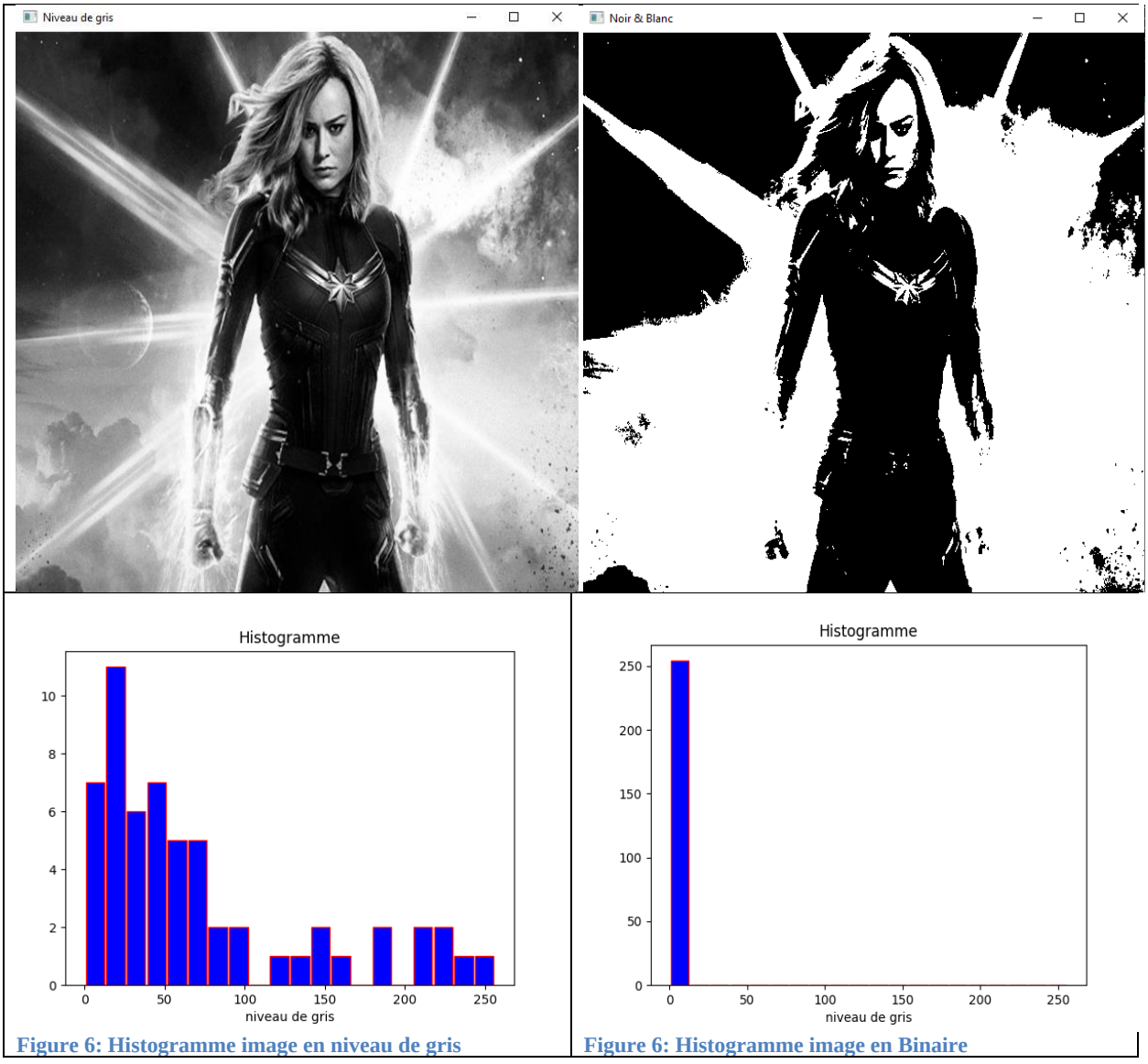


Figure 6: Histogramme image en niveau de gris

Figure 6: Histogramme image en Binaire

On obtient enfin pour l'image en niveau de gris, un histogramme sous forme de bâton chaque bâton représente le nombre de pixels ayant la même valeur de niveau de gris.

Concernant le deuxième histogramme de l'image binaire, on prévoit avoir deux bâtons le premier représente le nombre de pixels ayant la couleur noire et l'autre représente le nombre de pixels ayant la couleur blanche. Pourtant on a obtenu un seul bâton représentant la couleur noire, même si la table H qui regroupe les valeurs d'occurrence de chaque couleur est pleine en niveau de 0 et de 255.

Conclusion :

Pour terminer, nous aimerions tirer des conclusions du travail que nous avons réalisé. Nous avons produit des fonctions variées. Certaines concernent la création et la composition des images, d'autres son traitement et son amélioration. Nous avons donc appris à modifier et traiter les différents aspects d'une image. Nous avons également appris à améliorer une image à travers le filtre de Nagao.

Enfin, afin de mettre en exergue l'ensemble de nos réalisations et de pouvoir les présenter, nous avons créé d'abord un répertoire (*TPTraitementImage*) dans lequel nous avons enregistré nos fichiers Python contenant les différents algorithmes de traitement (format .py).

Sur le plan personnel nous avons découvert l'univers très vaste du traitement d'images et nous avons ainsi pu entrevoir les possibilités et la diversité des choses qu'il permet de faire. Nous avons pu toucher du doigt quelque chose de très utilisé aujourd'hui.

De plus, nous avons découvert un nouveau langage de programmation, et renforcé nos compétences en codage. Au-delà de l'aspect intellectuel, l'aspect humain est aussi important dans ce projet. Il constitue une de nos expériences de gestion de projet, gestion du temps qui sont des choses fondamentales pour un futur ingénieur.