codePremiereDeuxiemeParties

June 16, 2021

Présentation et traitement ponctuel d'une image

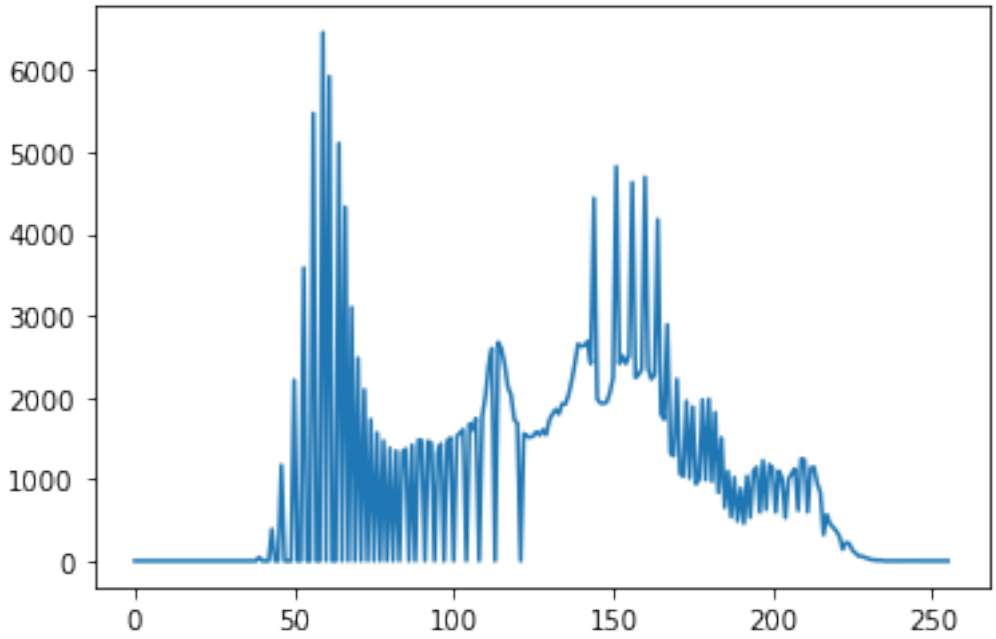
[3]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

#Chargement d'une matrice de pixels
image = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png")␣
 ↪# charge le fichier dans une matrice de pixels couleur
print(image.shape)              # les dimensions de la matrice


#Visualisation de la matrice avec matplotlib
#plt.imshow(matrice[..., ::-1])  #-1 car matplotlib utilise le systeme RVB␣
 ↪alors que OpenCv utilise BRV. Il faut donc d'abord convertir en RVB
#plt.show()

#Visualisation de la matrice avec OpenCV
cv.imshow("Lenna",image)
cv.waitKey(0)
cv.destroyAllWindows()
```

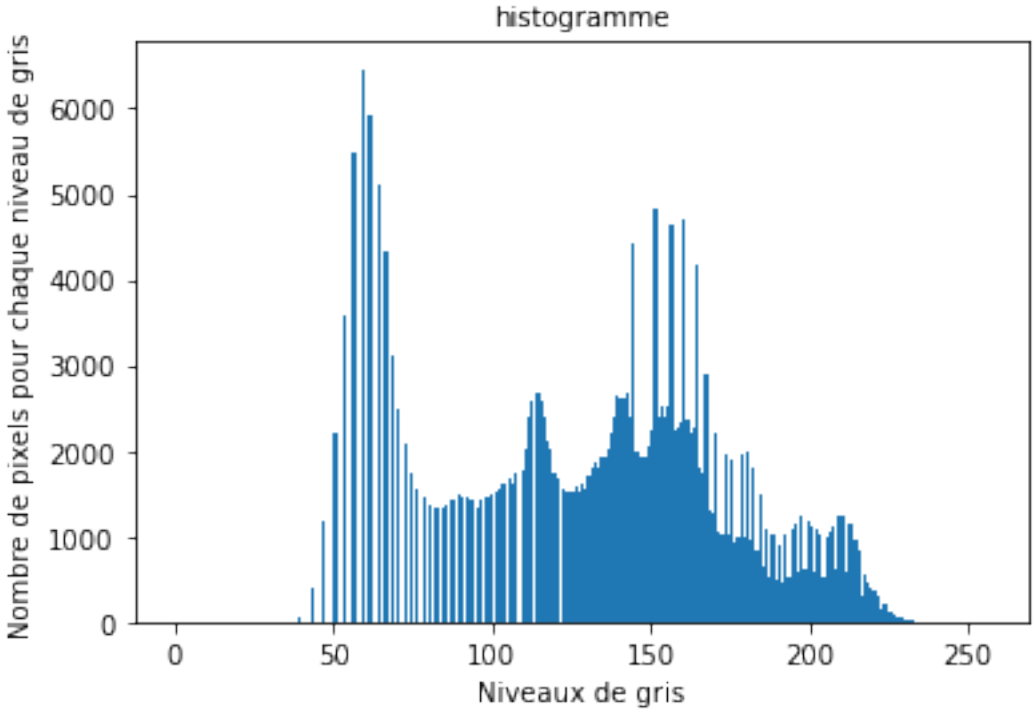
```
(512, 512, 3)
```

[4]:
```
# Chargement du fichier dans une matrice de pixels gris
y = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png", 0)
print(y)

# Dimensions de la matrice
print(y.shape)

cv.imshow("Matrice de pixels gris", y)
cv.waitKey(0)
cv.destroyAllWindows()

# Calcul de l'histogramme de l'image
hist = np.zeros(256, int)       # vecteur de 256 zéros (pour chaque gris)
for i in range(0,image.shape[0]):      # énumère les lignes
```

```
    for j in range(0,image.shape[1]):  # énumère les colonnes
        hist[y[i,j]] += 1

print(hist)
plt.plot(hist)
plt.show()

plt.hist(y.ravel(),256,[0,256])
plt.title("histogramme",fontsize=10)
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()
```

```
[[169 169 168 … 175 162 138]
 [169 169 168 … 175 162 138]
 [169 169 168 … 175 162 138]
 …
 [ 53  53  59 … 115 112 114]
 [ 53  53  64 … 117 118 122]
 [ 53  53  64 … 117 118 122]]
(512, 512)
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0   45    0    0
    0  388    0    0 1171    0    0    0 2211    0    0 3583    0    0
 5472    0    0 6460    0 5922    0    0 5104    0 4325    0 3100    0
 2479    0 2093    0 1730    0 1566    0 1469    0 1377    0 1345    0
 1335 1367    0 1416    0 1479 1468    0 1465 1438    0 1328 1431    0
 1465 1504    0 1522 1564 1605    0 1674 1604 1741    0 1774 2021 2398
 2598    0 2673 2588 2403 2126 2027 1730 1676    0 1554 1522 1510 1529
 1579 1536 1603 1542 1719 1795 1851 1796 1919 1916 2022 2198 2409 2650
 2626 2629 2685 2411 4435 1980 1930 1925 1939 2058 2237 4821 2411 2506
 2411 2507 4627 2240 2277 2346 4692 2352 2221 2278 4174 1798 1729 2888
 1316 1285 2222 1066 1035 1955 1012 1883  941  998 1969  990 1979  971
 1816  836 1502  651 1097  538 1020  487  886  458 1036  532 1093 1145
  595 1227  628 1176 1129  595 1098 1009  527  999 1066 1124  616 1249
 1236  598 1139 1145  947  835  319  567  452  411  364  296  142  225
  213  133   94   59   56   41   28   14   10    6    8    0    1    0
    1    0    0    0    2    0    0    2    0    0    0    0    0    0
    0    0    0    0]
```





[5]:
```
#Trouvons les niveaux de gris extrêmes présents dans l'image (ie les niveaux de␣
↪gris min et max utilisées par l'image)
```

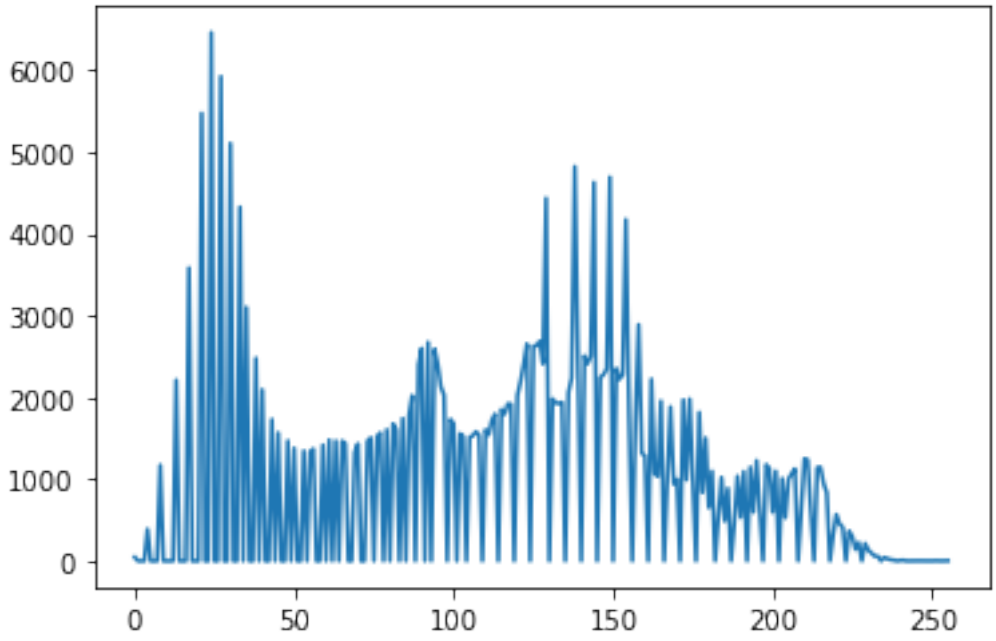
```
extr = list()
for i in range(0,hist.size):
    if hist[i] != 0:
        extr.append(i)
#print (extr)
a,b = extr[0],extr[-1]

print('le min des niveaux de gris est {0:d} et le max est {1:d}'.format(a,b))
print("a=",a,"et b=",b)
```

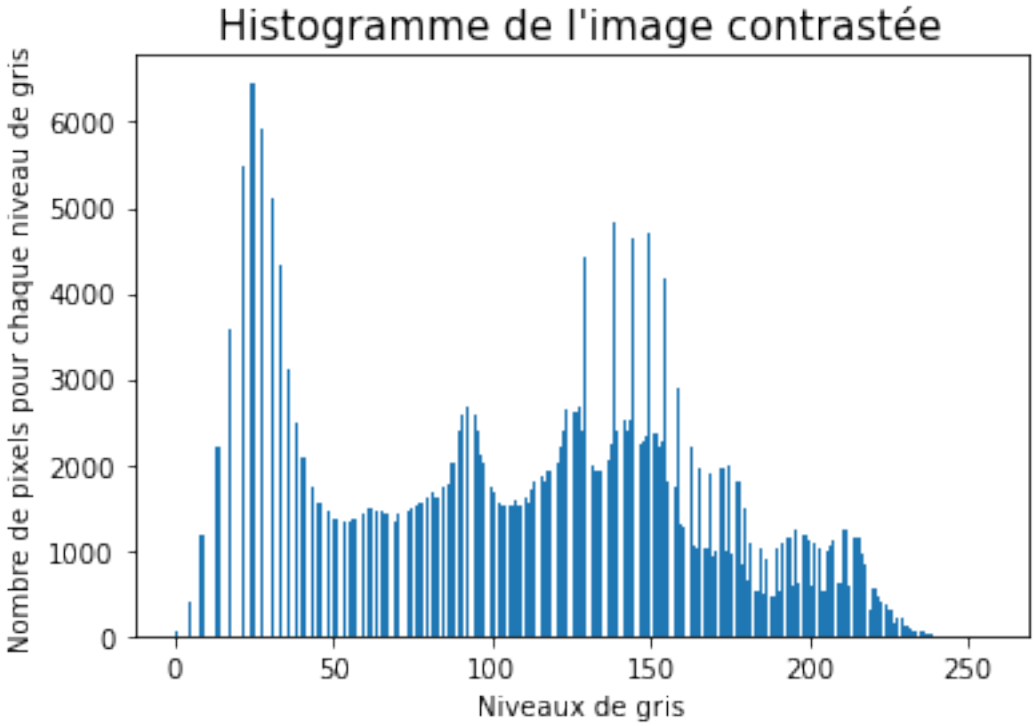
```
le min des niveaux de gris est 39 et le max est 245
a= 39 et b= 245
```

[6]:
```
#Recadrage de dynamique : contraste
#A un niveau de gris f de l'image originelle correspond le niveau t(f) dans␣
 ↪l'image transformée
# Notre fonction de contraste sera définie comme suit:

def contrast(a,b,y):      # y est la matrice des niveaux de gris
    y_new = np.zeros((image.shape[0],image.shape[1]),np.uint8)
    for i in range(0,image.shape[0]):
        for j in range(0,image.shape[1]):
            if y[i,j]<a:
                y_new[i,j]=0
            elif y[i,j]>b:
                y_new[i,j]=255
            else:
                y_new[i,j]=255*(y[i,j]-a)/(b-a)
    return (y_new)

y = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png", 0)
A = contrast(39,245,y)
print(A)

#Nouvel histogramme
hist_new = np.zeros(256, int)        # prépare un vecteur de 256 zéros (pour␣
 ↪chaque gris)
for i in range(0,y.shape[0]):       # énumère les lignes
    for j in range(0,y.shape[1]):  # énumère les colonnes
        hist_new[A[i,j]] += 1

print(hist_new)
plt.plot(hist_new)
plt.show()

plt.hist(A.ravel(),256,[0,256])
plt.title("Histogramme de l'image contrastée", fontsize=15)
```

```
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()
```

```
[[160 160 159 … 168 152 122]
 [160 160 159 … 168 152 122]
 [160 160 159 … 168 152 122]
 …
 [ 17  17  24 …  94  90  92]
 [ 17  17  30 …  96  97 102]
 [ 17  17  30 …  96  97 102]]
[  45    0    0    0  388    0    0    0 1171    0    0    0    0 2211
    0    0    0 3583    0    0    0 5472    0    0 6460    0    0 5922
    0    0 5104    0    0 4325    0 3100    0    0 2479    0 2093    0
    0 1730    0 1566    0    0 1469    0 1377    0    0 1345    0 1335
 1367    0    0 1416    0 1479    0 1468    0 1465 1438    0    0 1328
 1431    0    0 1465 1504    0 1522 1564    0 1605    0 1674 1604    0
 1741    0 1774 2021    0 2398 2598    0 2673    0 2588 2403 2126 2027
    0 1730 1676    0 1554 1522    0 1510 1529 1579 1536    0 1603 1542
 1719 1795    0 1851 1796 1919 1916    0 2022 2198 2409 2650    0 2626
 2629 2685 2411 4435    0 1980 1930 1925 1939    0 2058 2237 4821 2411
    0 2506 2411 2507 4627    0 2240 2277 2346 4692    0 2352 2221 2278
 4174 1798    0 1729 2888 1316 1285    0 2222 1066 1035 1955    0 1012
 1883  941  998    0 1969  990 1979  971    0 1816  836 1502  651 1097
    0  538 1020  487  886    0  458 1036  532 1093    0 1145  595 1227
  628    0 1176 1129  595 1098    0 1009  527  999 1066 1124    0  616
 1249 1236  598    0 1139 1145  947  835    0  319  567  452  411    0
  364  296  142  225    0  213  133   94   59   56    0   41   28   14
   10    0    6    8    0    1    0    0    1    0    0    0    0    2
    0    0    0    2]
```


6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

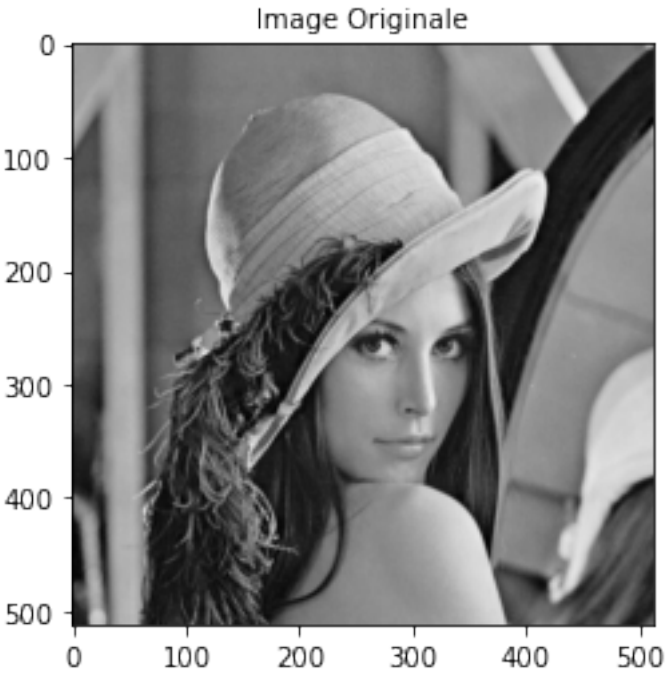

Histogramme de l'image contrastée
Nombre de pixels pour chaque niveau de gris
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
Niveaux de gris

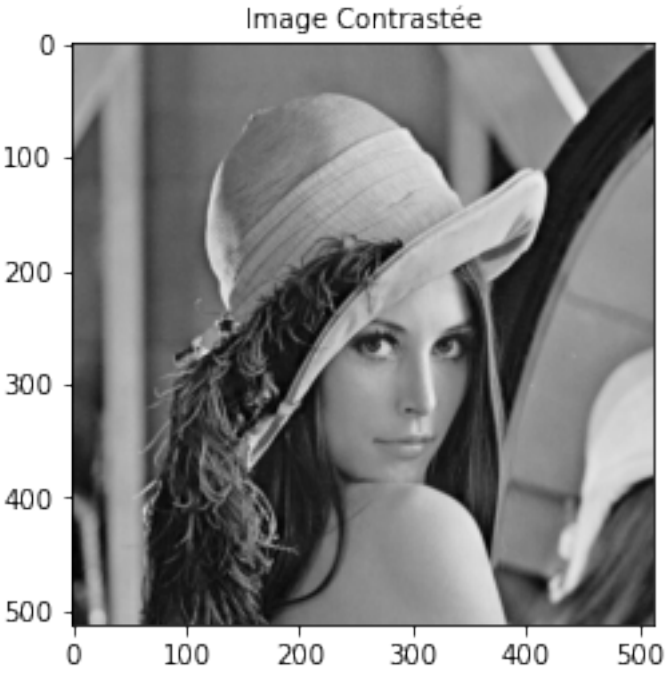
[7]:
```
#Trouvons les niveaux de gris extrêmes présents dans l'image contrastée (ie les␣
 ↪niveaux de gris min et max utilisées par l'image)
extr = list()
for i in range(0,hist_new.size):
    if hist_new[i] != 0:
        extr.append(i)
#print (extr)
a,b = extr[0],extr[-1]

print('le min des niveaux de gris est {0:d} et le max est {1:d}'.format(a,b))
```

```
le min des niveaux de gris est 0 et le max est 255
```

On voit que dans l'image contrastée, tous les niveaux de gris ont été utilisés, ce qui est le résultat attendu.

[8]:
```
#Affichage des images contrastrée et originelle

#Matrice originelle
plt.imshow(y, cmap = "gray" )   # affiche la matrice de niveaux de gris
plt.title("Image Originale",fontsize=10)
plt.show() # ouvre la fenêtre d'affichage et attend la fin de l'interaction␣
 ↪utilisateur

cv.imshow("Image Originale",y)


#Matrice contrastée
plt.imshow(A, cmap = "gray" )   # affiche la matrice de niveaux de gris␣
 ↪contrastée
plt.title("Image Contrastée",fontsize=10)
plt.show() # ouvre la fenêtre d'affichage et attend la fin de l'interaction␣
 ↪utilisateur

cv.imshow("Image Contrastee",A)
cv.waitKey(0)
cv.destroyAllWindows()
```



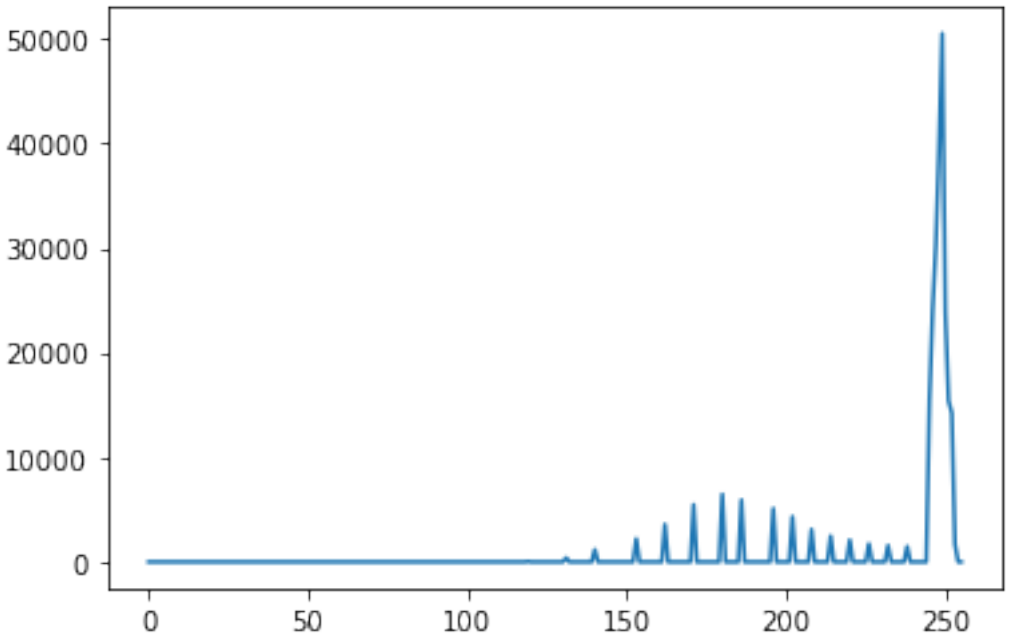


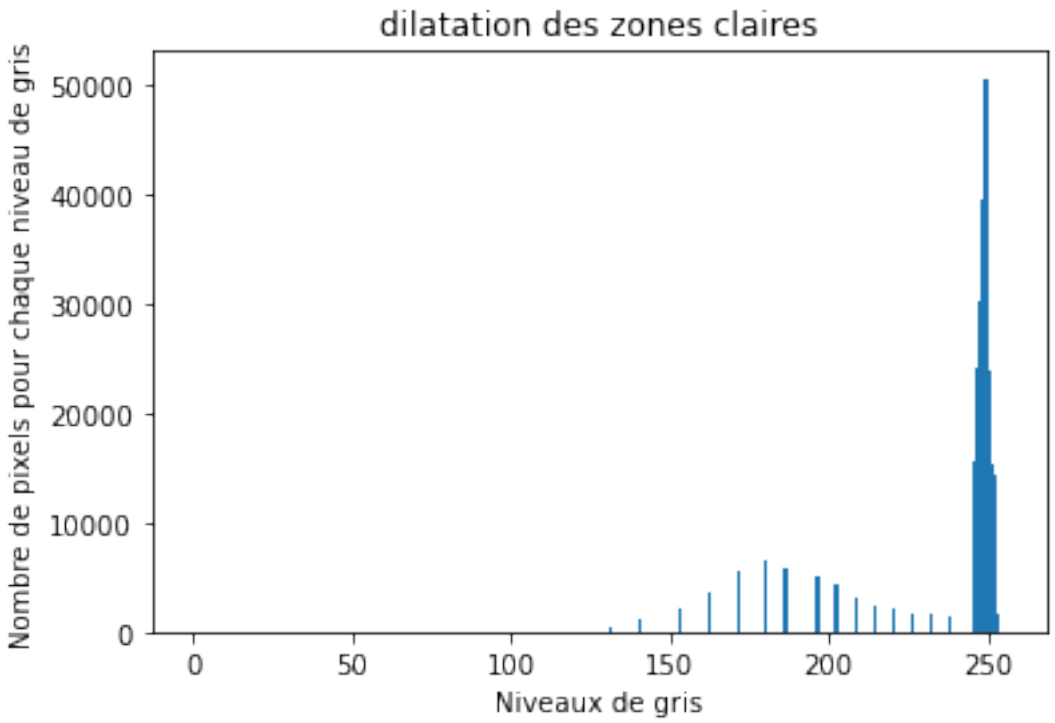
[9]:
```
#rehaussement des contrastes
```

```
def r_contrast(a,b,y):  # y est la matrice des niveaux de gris
    y_rc = np.zeros((image.shape[0],image.shape[1]),np.uint8)
    for i in range(0,image.shape[0]):
        for j in range(0,image.shape[1]):
            if (y[i,j]>=0 and y[i,j]<=a):
                y_rc[i,j]=((b*y[i,j])/a)
            elif (y[i,j]>=a and y[i,j]<=255):
                y_rc[i,j]=((255-b)*y[i,j]+255*(b-a))/(255-a)
            else:
                y_rc[i,j]=y[i,j]
    return(y_rc)

#v = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/chat.jpg", 0)
v = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png", 0)

#dilatation de la dynamique des zones claires avec a = 80 et B = 245
B = r_contrast(80,245,v)
print(B)
## Nouvel histogramme et affichage des images

hist_hc = np.zeros(256, int)       # prépare un vecteur de 256 zéros (pour␣
 ↪chaque gris)
for i in range(0,image.shape[0]):      # énumère les lignes
    for j in range(0,image.shape[1]):  # énumère les colonnes
        hist_hc[B[i,j]] += 1

print(hist_hc)
plt.plot(hist_hc)
plt.show()

plt.hist(B.ravel(),256,[0,256])
plt.title("dilatation des zones claires",fontsize = 12)
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()
```

```
[[250 250 250 … 250 249 248]
 [250 250 250 … 250 249 248]
 [250 250 250 … 250 249 248]
 …
 [162 162 180 … 247 246 246]
 [162 162 196 … 247 247 247]
 [162 162 196 … 247 247 247]]
[    0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0    45
    0     0     0     0     0     0     0     0     0     0     0   388
    0     0     0     0     0     0     0     0  1171     0     0     0
    0     0     0     0     0     0     0     0     0  2211     0     0
    0     0     0     0     0     0  3583     0     0     0     0     0
    0     0     0  5472     0     0     0     0     0     0     0     0
 6460     0     0     0     0     0  5922     0     0     0     0     0
    0     0     0     0  5104     0     0     0     0     0  4325     0
    0     0     0     0  3100     0     0     0     0     0  2479     0
    0     0     0     0  2093     0     0     0     0     0  1730     0
    0     0     0     0  1566     0     0     0     0     0  1469     0
    0     0     0     0     0 15449 24143 30290 39528 50515 23776 15391
14239  1690     5     0]
```





[10]:
```
#Affichage de la matrice de rehaussement de contraste

plt.imshow(y, cmap = "gray" )   # affiche la matrice de niveaux de gris
plt.title("Image Originale")
plt.show()

cv.imshow("Image Originale",y)

plt.imshow(B, cmap = "gray" ) # affiche la matrice avec dilatation des zones␣
 ↪claires
plt.title("Dilatation des zones  claires avec a=80 et b=245")
plt.show()

cv.imshow("Dilatation des zones claires",B)
cv.waitKey(0)
cv.destroyAllWindows()
```

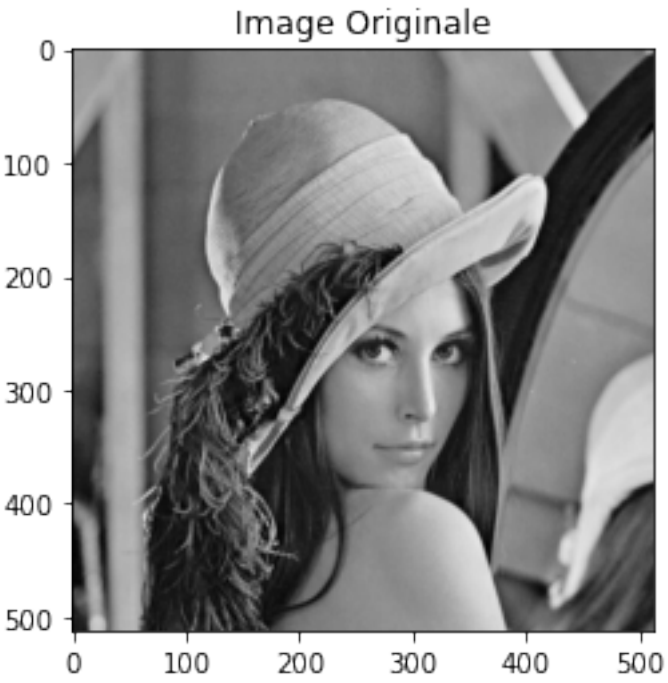
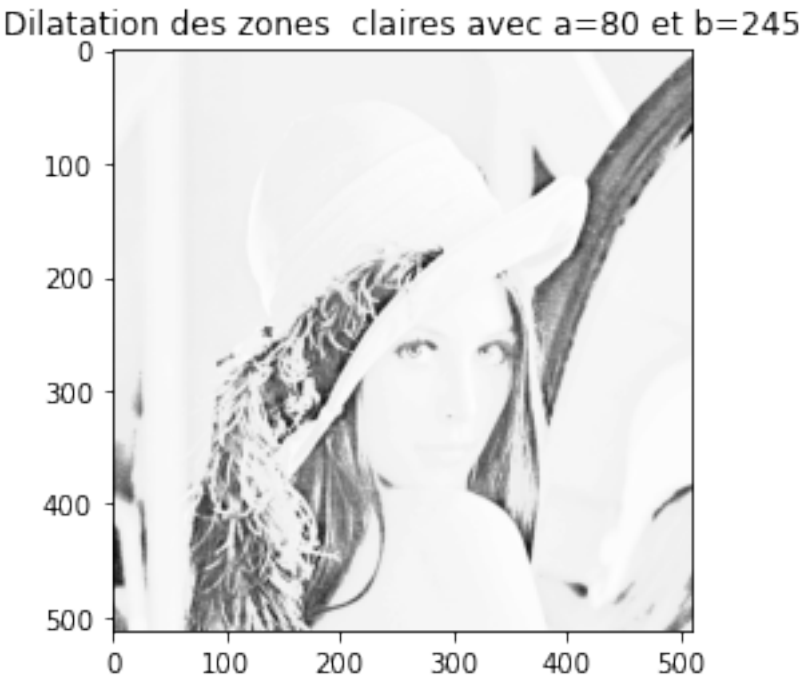


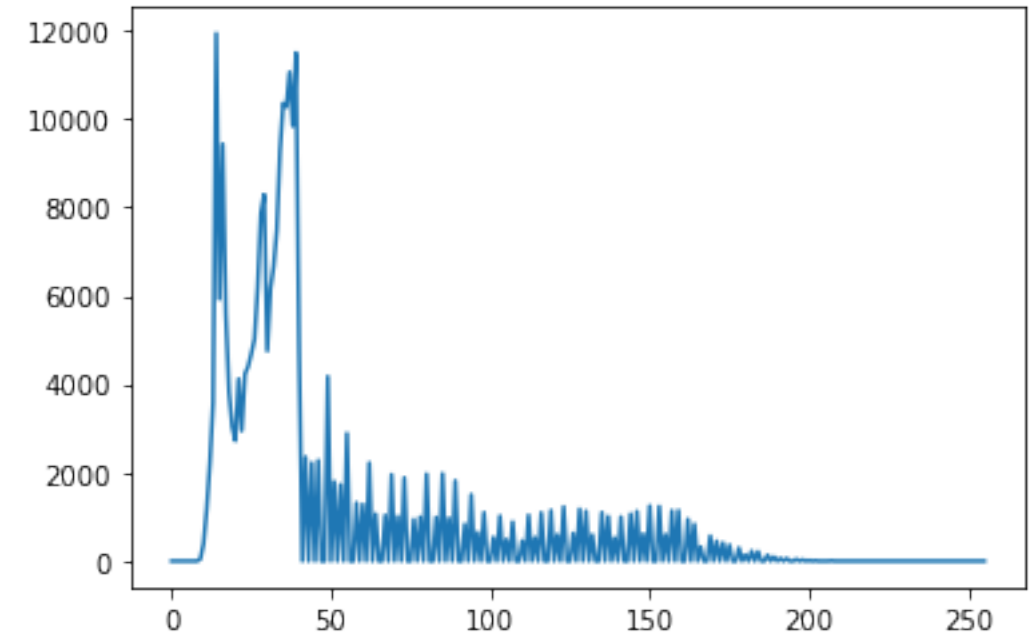
Image Originale
0
100
200
300
400
500
0
100
200
300
400
500

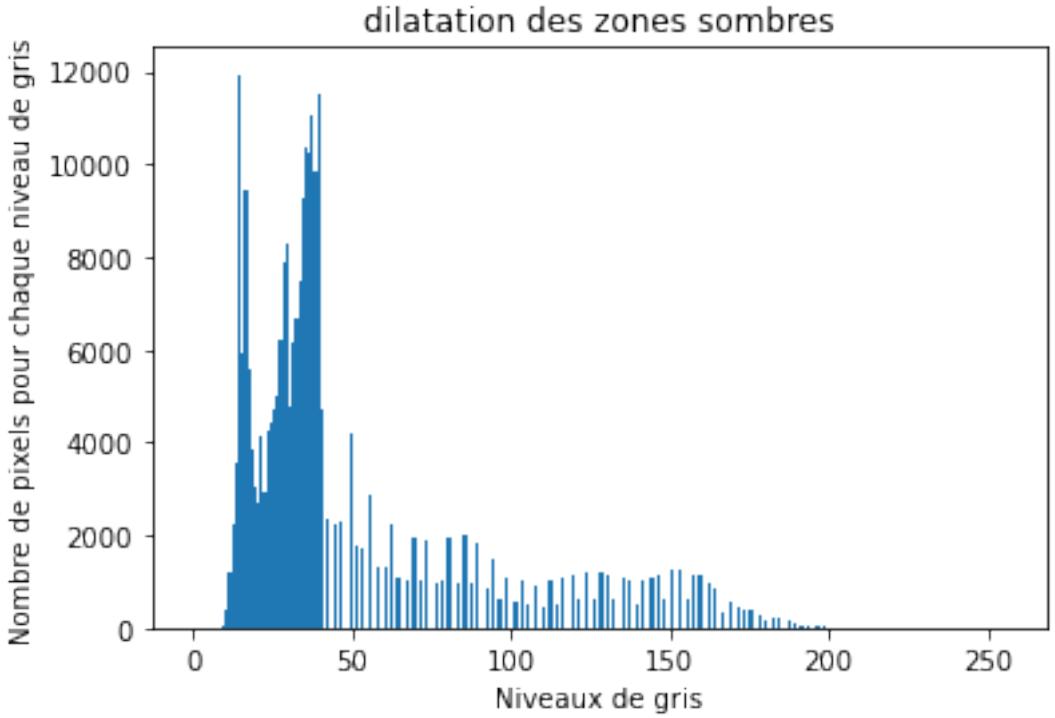

Dilatation des zones claires avec a=80 et b=245
0
100
200
300
400
500
0
100
200
300
400
500

[11]:
```
#dilatation de la dynamique des zones sombres avec a = 160 et b = 40

C = r_contrast(160,40,v)
print(C)
## Nouvel histogramme et affichage des images

hist_hc = np.zeros(256, int)       # prépare un vecteur de 256 zéros (pour␣
 ↪chaque gris)
for i in range(0,image.shape[0]):      # énumère les lignes
    for j in range(0,image.shape[1]):  # énumère les colonnes
        hist_hc[C[i,j]] += 1

print(hist_hc)
plt.plot(hist_hc)
plt.show()

plt.hist(C.ravel(),256,[0,256])
plt.title("dilatation des zones sombres",fontsize = 12)
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()
```

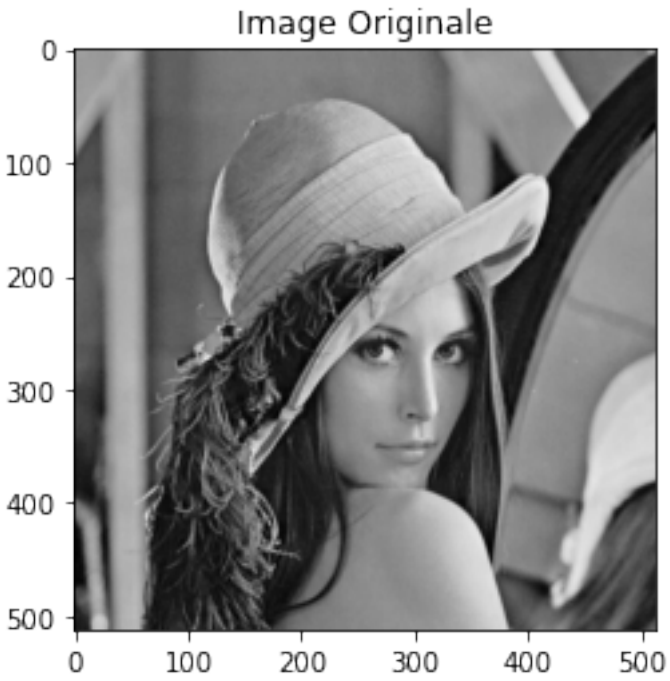
```
[[60 60 58 … 73 44 34]
 [60 60 58 … 73 44 34]
 [60 60 58 … 73 44 34]
 …
 [13 13 14 … 28 28 28]
 [13 13 16 … 29 29 30]
 [13 13 16 … 29 29 30]]
[    0     0     0     0     0     0     0     0     0    45   388  1171
  2211  3583 11932  5922  9429  5579  3823  3035  2722  4118  2947  4231
  4400  4691  5019  6193  7859  8286  4752  6154  6659  7482  9279 10351
 10270 11055  9835 11490  4692     0  2352     0  2221     0  2278     0
     0  4174     0  1798     0  1729     0  2888     0     0  1316     0
  1285     0  2222     0  1066     0     0  1035     0  1955     0  1012
     0  1883     0     0   941     0   998     0  1969     0     0   990
     0  1979     0   971     0  1816     0     0   836     0  1502     0
   651     0  1097     0     0   538     0  1020     0   487     0   886
     0     0   458     0  1036     0   532     0  1093     0     0  1145
     0   595     0  1227     0     0   628     0  1176     0  1129     0
   595     0     0  1098     0  1009     0   527     0   999     0     0
  1066     0  1124     0   616     0  1249     0     0  1236     0   598
     0  1139     0  1145     0     0   947     0   835     0   319     0
     0   567     0   452     0   411     0   364     0     0   296     0
   142     0   225     0   213     0     0   133     0    94     0    59
     0    56     0     0    41     0    28     0    14     0    10     0
     0     6     0     8     0     0     0     0     1     0     0     0
     1     0     0     0     0     0     0     0     0     2     0     0
```

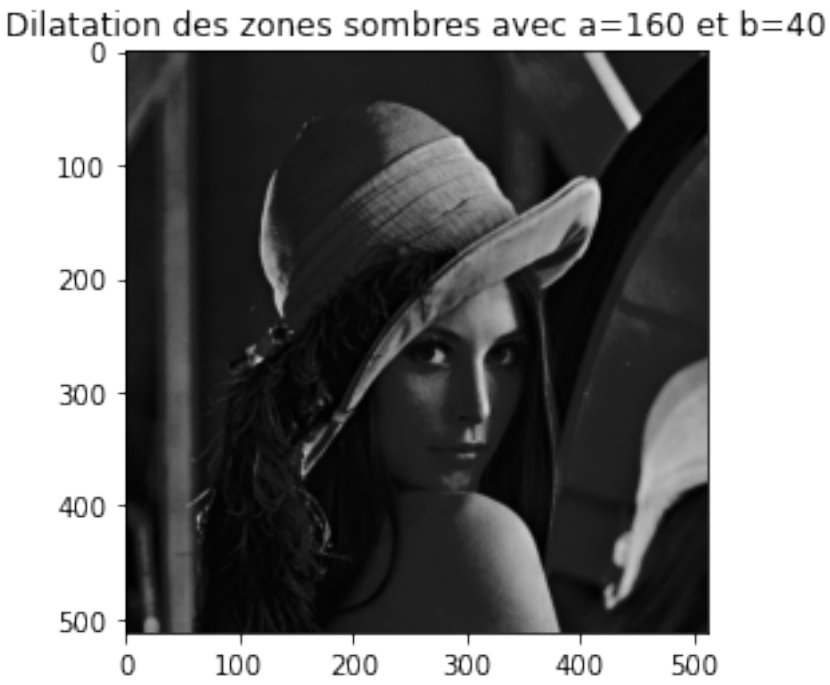
```
0    0    0    0    2    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0]
```





[12]:
```
#Affichage de la matrice de rehaussement de contraste

plt.imshow(y, cmap = "gray" )   # affiche la matrice de niveaux de gris
plt.title("Image Originale")
plt.show()

cv.imshow("Image Originale",y)

plt.imshow(C, cmap = "gray" ) # affiche la matrice avec dilatation des zones␣
 ↪sombres
plt.title("Dilatation des zones sombres avec a=160 et b=40")
plt.show()

cv.imshow("Dilatation des zones sombres",C)
cv.waitKey(0)
cv.destroyAllWindows()
```

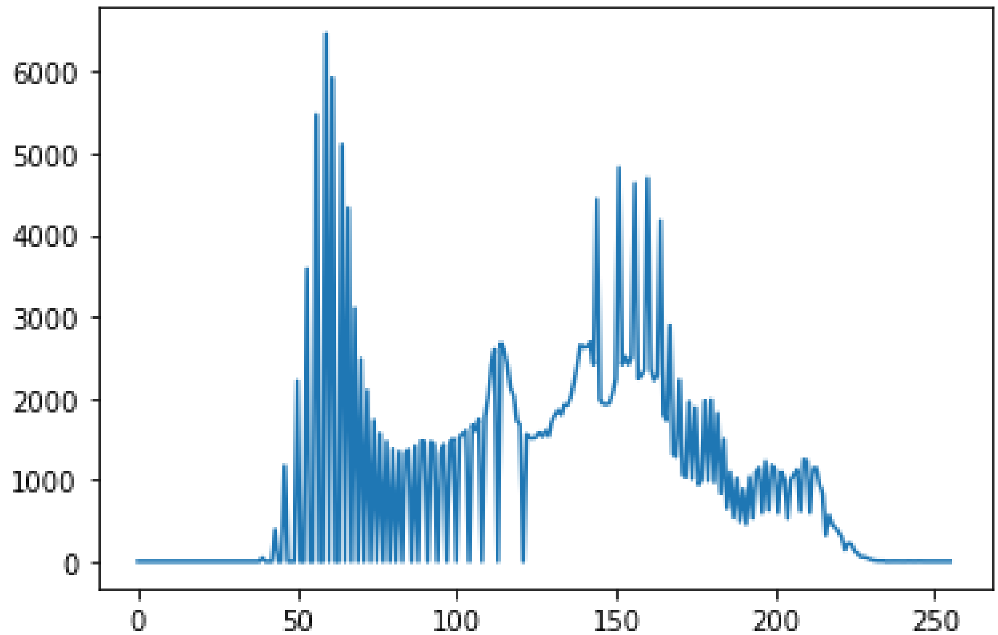




[13]:
```
###Egalisation de l'histogramme
image = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png")
y = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png",0)

# Calcule l'histogramme de l'image
histo = np.zeros(256, int)      # prépare un vecteur de 256 zéros
for i in range(0,image.shape[0]):       # énumère les lignes
    for j in range(0,image.shape[1]):   # énumère les colonnes
        histo[y[i,j]] = histo[y[i,j]] + 1
print(histo)
plt.plot(histo)
plt.show()

# Calcule l'histogramme cumulé hc
hc = np.zeros(256, int)         # prépare un vecteur de 256 zéros
hc[0] = histo[0]
for i in range(1,256):
    hc[i] = histo[i] + hc[i-1]

# Normalise l'histogramme cumulé
nbpixels = y.size
hc = (hc / nbpixels) * 255
print(hc)
plt.plot(hc)
```

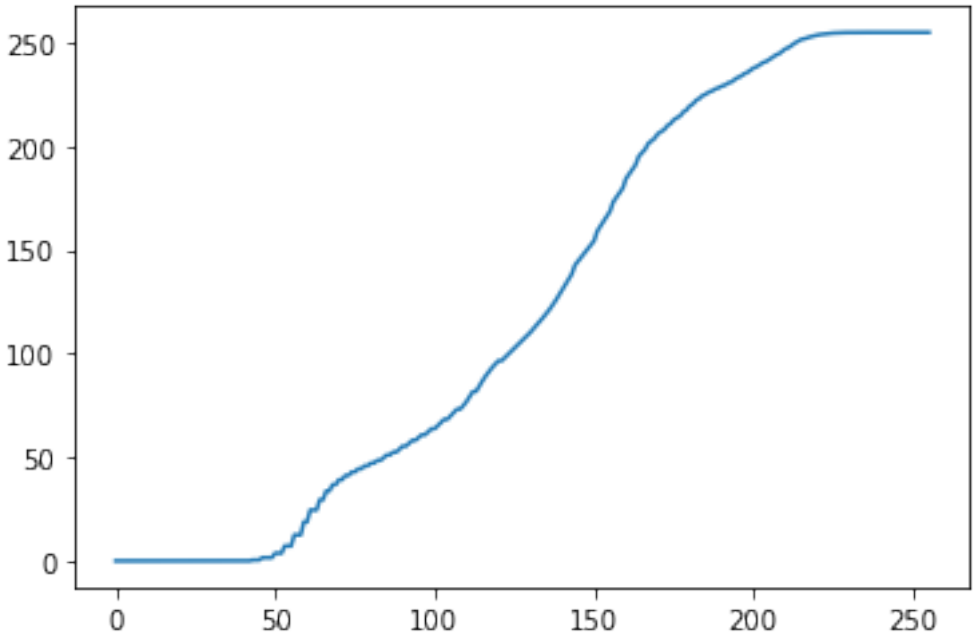
```
plt.show()

# Utilise hc comme table de conversion des niveaux de gris
for i in range(0,y.shape[0]):        # énumère les lignes
    for j in range(0,y.shape[1]):    # énumère les colonnes
        y[i,j] = hc[y[i,j]]
print(y)
cv.imshow("Image après égalisation", y)
plt.hist(y.ravel(),256,[0,256])

plt.title("histogramme égalisé", fontsize=10)
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()

cv.waitKey(0)
cv.destroyAllWindows()
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0   45    0    0
    0  388    0    0 1171    0    0    0 2211    0    0 3583    0    0
 5472    0    0 6460    0 5922    0    0 5104    0 4325    0 3100    0
 2479    0 2093    0 1730    0 1566    0 1469    0 1377    0 1345    0
 1335 1367    0 1416    0 1479 1468    0 1465 1438    0 1328 1431    0
 1465 1504    0 1522 1564 1605    0 1674 1604 1741    0 1774 2021 2398
 2598    0 2673 2588 2403 2126 2027 1730 1676    0 1554 1522 1510 1529
 1579 1536 1603 1542 1719 1795 1851 1796 1919 1916 2022 2198 2409 2650
 2626 2629 2685 2411 4435 1980 1930 1925 1939 2058 2237 4821 2411 2506
 2411 2507 4627 2240 2277 2346 4692 2352 2221 2278 4174 1798 1729 2888
 1316 1285 2222 1066 1035 1955 1012 1883  941  998 1969  990 1979  971
 1816  836 1502  651 1097  538 1020  487  886  458 1036  532 1093 1145
  595 1227  628 1176 1129  595 1098 1009  527  999 1066 1124  616 1249
 1236  598 1139 1145  947  835  319  567  452  411  364  296  142  225
  213  133   94   59   56   41   28   14   10    6    8    0    1    0
    1    0    0    0    2    0    0    2    0    0    0    0    0    0
    0    0    0    0]
```

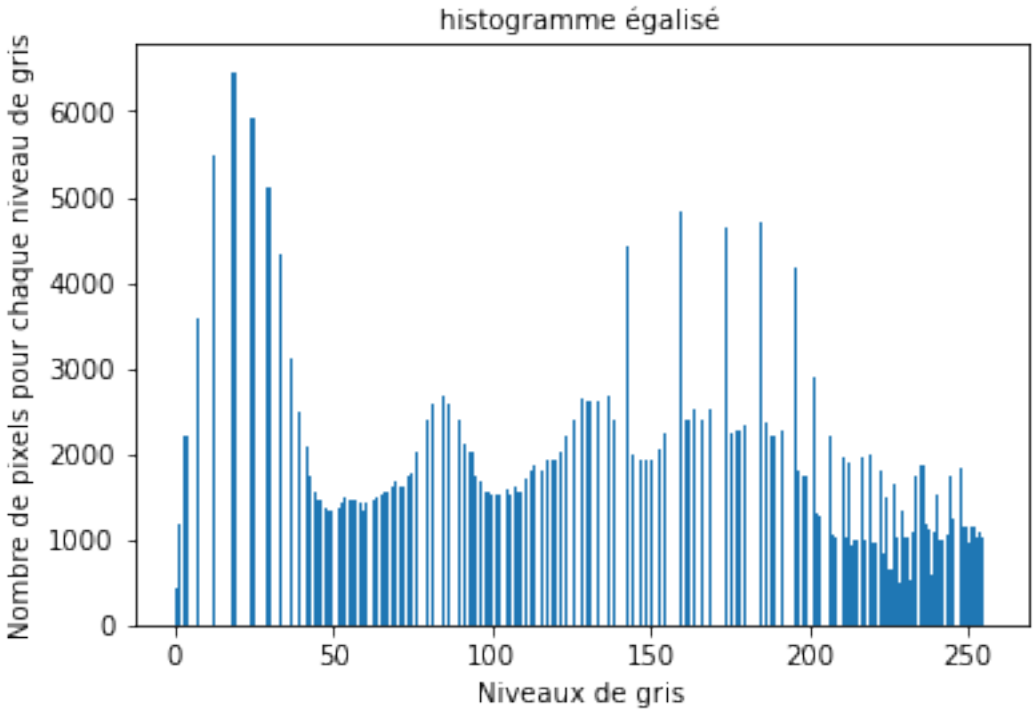



```
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.37736511e-02
 4.37736511e-02 4.37736511e-02 4.37736511e-02 4.21199799e-01
 4.21199799e-01 4.21199799e-01 1.56028748e+00 1.56028748e+00
 1.56028748e+00 1.56028748e+00 3.71103287e+00 3.71103287e+00
 3.71103287e+00 7.19638824e+00 7.19638824e+00 7.19638824e+00
 1.25192642e+01 1.25192642e+01 1.25192642e+01 1.88032150e+01
 1.88032150e+01 2.45638275e+01 2.45638275e+01 2.45638275e+01
 2.95287323e+01 2.95287323e+01 3.37358665e+01 3.37358665e+01
 3.67513847e+01 3.67513847e+01 3.91628265e+01 3.91628265e+01
 4.11987877e+01 4.11987877e+01 4.28816414e+01 4.28816414e+01
 4.44049644e+01 4.44049644e+01 4.58339310e+01 4.58339310e+01
 4.71734047e+01 4.71734047e+01 4.84817505e+01 4.84817505e+01
 4.97803688e+01 5.11101151e+01 5.11101151e+01 5.24875259e+01
 5.24875259e+01 5.39262199e+01 5.53542137e+01 5.53542137e+01
 5.67792892e+01 5.81781006e+01 5.81781006e+01 5.94699097e+01
 6.08619118e+01 6.08619118e+01 6.22869873e+01 6.37500000e+01
 6.37500000e+01 6.52305222e+01 6.67518997e+01 6.83131599e+01
```

```
6.83131599e+01 6.99415398e+01 7.15018272e+01 7.31953812e+01
7.31953812e+01 7.49210358e+01 7.68869591e+01 7.92196083e+01
8.17468071e+01 8.17468071e+01 8.43469620e+01 8.68644333e+01
8.92019463e+01 9.12700081e+01 9.32417679e+01 9.49246216e+01
9.65549469e+01 9.65549469e+01 9.80665970e+01 9.95471191e+01
1.01015968e+02 1.02503300e+02 1.04039268e+02 1.05533409e+02
1.07092724e+02 1.08592701e+02 1.10264854e+02 1.12010937e+02
1.13811493e+02 1.15558548e+02 1.17425251e+02 1.19289036e+02
1.21255932e+02 1.23394032e+02 1.25737381e+02 1.28315163e+02
1.30869598e+02 1.33426952e+02 1.36038780e+02 1.38384075e+02
1.42698212e+02 1.44624252e+02 1.46501656e+02 1.48374195e+02
1.50260353e+02 1.52262268e+02 1.54438305e+02 1.59127922e+02
1.61473217e+02 1.63910923e+02 1.66256218e+02 1.68694897e+02
1.73195801e+02 1.75374756e+02 1.77589703e+02 1.79871769e+02
1.84435902e+02 1.86723804e+02 1.88884277e+02 1.91100197e+02
1.95160446e+02 1.96909447e+02 1.98591328e+02 2.01400623e+02
2.02680759e+02 2.03930740e+02 2.06092186e+02 2.07129135e+02
2.08135929e+02 2.10037651e+02 2.11022072e+02 2.12853756e+02
2.13769112e+02 2.14739914e+02 2.16655254e+02 2.17618275e+02
2.19543343e+02 2.20487881e+02 2.22254391e+02 2.23067608e+02
2.24528675e+02 2.25161934e+02 2.26229038e+02 2.26752377e+02
2.27744579e+02 2.28218307e+02 2.29080162e+02 2.29525681e+02
2.30533447e+02 2.31050949e+02 2.32114162e+02 2.33227959e+02
2.33806744e+02 2.35000305e+02 2.35611191e+02 2.36755142e+02
2.37853374e+02 2.38432159e+02 2.39500237e+02 2.40481739e+02
2.40994377e+02 2.41966152e+02 2.43003101e+02 2.44096470e+02
2.44695683e+02 2.45910645e+02 2.47112961e+02 2.47694664e+02
2.48802624e+02 2.49916420e+02 2.50837612e+02 2.51649857e+02
2.51960163e+02 2.52511711e+02 2.52951393e+02 2.53351192e+02
2.53705273e+02 2.53993206e+02 2.54131336e+02 2.54350204e+02
2.54557400e+02 2.54686775e+02 2.54778214e+02 2.54835606e+02
2.54890079e+02 2.54929962e+02 2.54957199e+02 2.54970818e+02
2.54980545e+02 2.54986382e+02 2.54994164e+02 2.54994164e+02
2.54995136e+02 2.54995136e+02 2.54996109e+02 2.54996109e+02
2.54996109e+02 2.54996109e+02 2.54998055e+02 2.54998055e+02
2.54998055e+02 2.55000000e+02 2.55000000e+02 2.55000000e+02
2.55000000e+02 2.55000000e+02 2.55000000e+02 2.55000000e+02
2.55000000e+02 2.55000000e+02 2.55000000e+02 2.55000000e+02]
```




```
[[203 203 202 … 212 188 125]
 [203 203 202 … 212 188 125]
 [203 203 202 … 212 188 125]
 …
 [  7   7  18 …  86  81  84]
 [  7   7  29 …  91  93  98]
 [  7   7  29 …  91  93  98]]
```



[14]:
```
#Affichage de l'image égalisée
matG = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png",0)
plt.imshow(matG, cmap = "gray" )   # affiche la matrice de niveaux de gris
plt.title("Image Originale")
plt.show()

cv.imshow("Image Originale",matG)

plt.imshow(y, cmap = "gray" )   # affiche la matrice après égalisation
plt.title("Image après égalisation")
plt.show()

cv.imshow("Image apres egalisation", y)
cv.waitKey(0)
cv.destroyAllWindows()
```

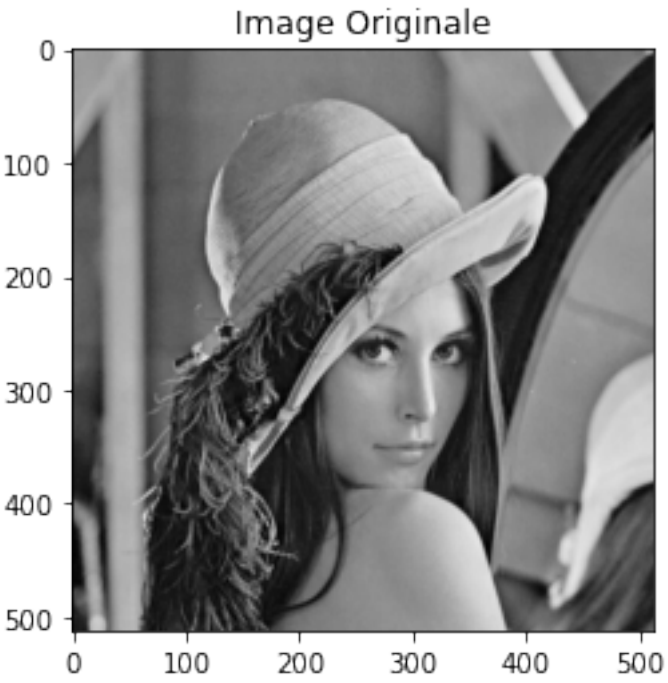


Image Originale
0
100
200
300
400
500
0
100
200
300
400
500

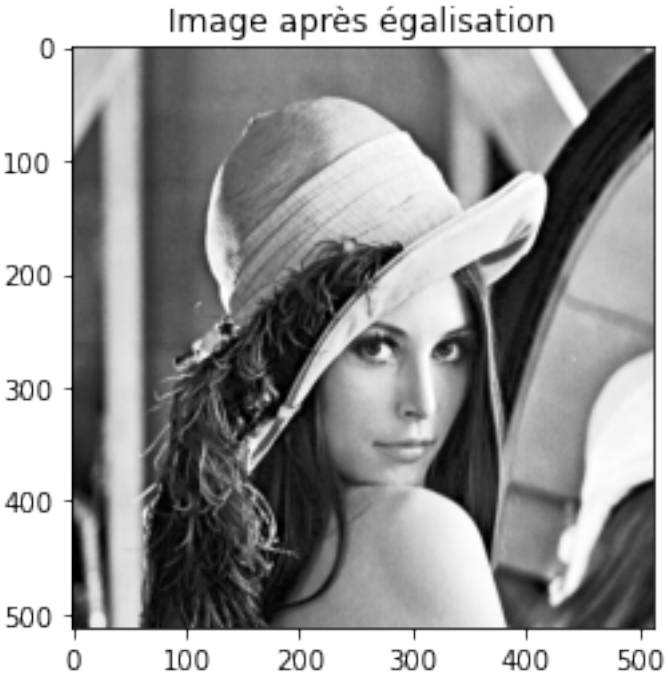
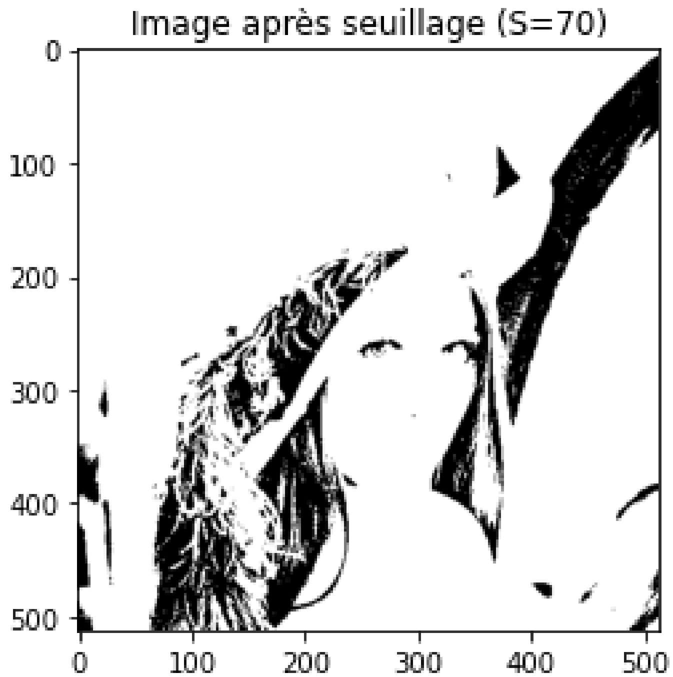


Image après égalisation
0
100
200
300
400
500
0
100
200
300
400
500

[15]:
```
###Binarisation-seuillage
#Le but de la binarisation d'une image est d'affecter un niveau uniforme aux pixels pertinents et d'éliminer les autres.
#Le seuillage consiste à affecter le niveau 255 aux pixels dont la valeur est supérieure à un seuil S et le niveau 0 aux autres.

def seuillage(S,y):    #Q: Sur quoi se base t-on pour choisir le seuil?
    y_seuil =  np.zeros((image.shape[0],image.shape[1]),np.uint8)
    for i in range(0,y.shape[0]):
        for j in range(0,y.shape[1]):
            if (y[i,j]>S):
                y_seuil[i,j] = 255
            else :
                y_seuil[i,j] = 0
    return (y_seuil)

y = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png",0)
D = seuillage (70,y)
print(D)

plt.imshow(D, cmap="gray")
plt.title("Image après seuillage (S=70)")
plt.show()

cv.imshow("Image apres seuillage (S=70)",D)
cv.waitKey(0)
cv.destroyAllWindows()

# Histogramme
hist = np.zeros(256, int)       # prépare un vecteur de 256 zéros (pour chaque gris)
for i in range(0,image.shape[0]):      # énumère les lignes
    for j in range(0,image.shape[1]):  # énumère les colonnes
        hist[D[i,j]] += 1

print(hist)
plt.hist(D.ravel(),256,[0,256])
plt.title("Histogramme après seuillage (S=70)",fontsize = 12)
plt.xlabel("Niveaux de gris")
plt.ylabel("Nombre de pixels pour chaque niveau de gris")
plt.show()
```

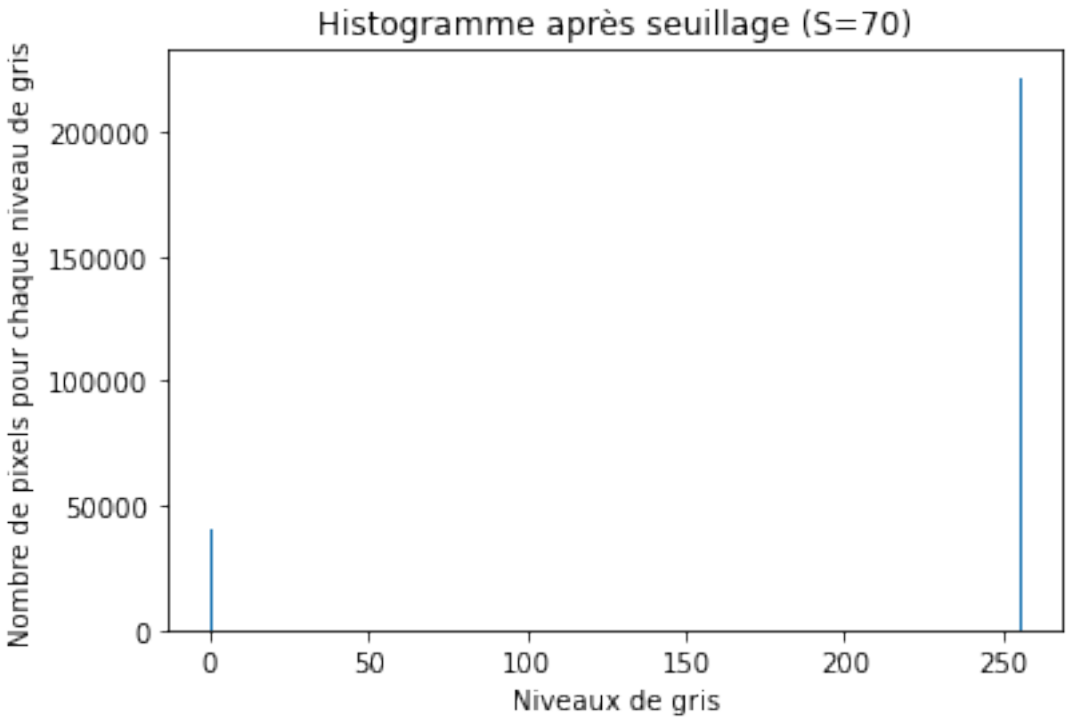
```
[[255 255 255 … 255 255 255]
 [255 255 255 … 255 255 255]
 [255 255 255 … 255 255 255]
 …
 [  0   0   0 … 255 255 255]
```

```
[  0   0   0 … 255 255 255]
[  0   0   0 … 255 255 255]]
```




```
[ 40260      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
```

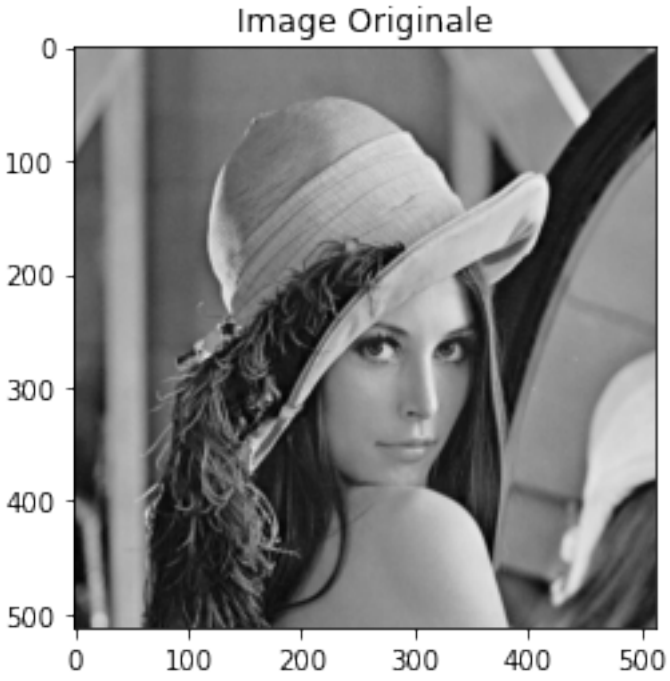
```
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0 221884]
```



[16]:
```
#Détermination automatique de la valeur du seuillage en maximisant la variance intraclasse

import numpy as np
import cv2 as cv

img = cv.imread("D:/Master_IM/Cours_S2/Projet_Traitement_dImages/Lenna.png",cv.IMREAD_GRAYSCALE)
fenetre = 'Image binaire OTSU'
cv.namedWindow(fenetre)
thresh,imgbin=cv.threshold(img,0,255,cv.THRESH_OTSU)
cv.imshow(fenetre,imgbin)
print( 'Valeur seuil OTSU ', thresh)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
Valeur seuil OTSU  124.0
```

[17]:
```
#Affichage de l'image après seuillage

plt.imshow(img,cmap = "gray" )   # affiche la matrice de niveaux de gris
plt.title("Image Originale")
plt.show()

cv.imshow("Image Originale",img)

plt.imshow(imgbin, cmap = "gray" )   # affiche la matrice après seuillage
plt.title("Image avec valeur seuil OTSU (S=124)")
plt.show()

cv.imshow("Image avec valeur seuil OTSU (S=124)",imgbin)
cv.waitKey(0)
cv.destroyAllWindows()
```



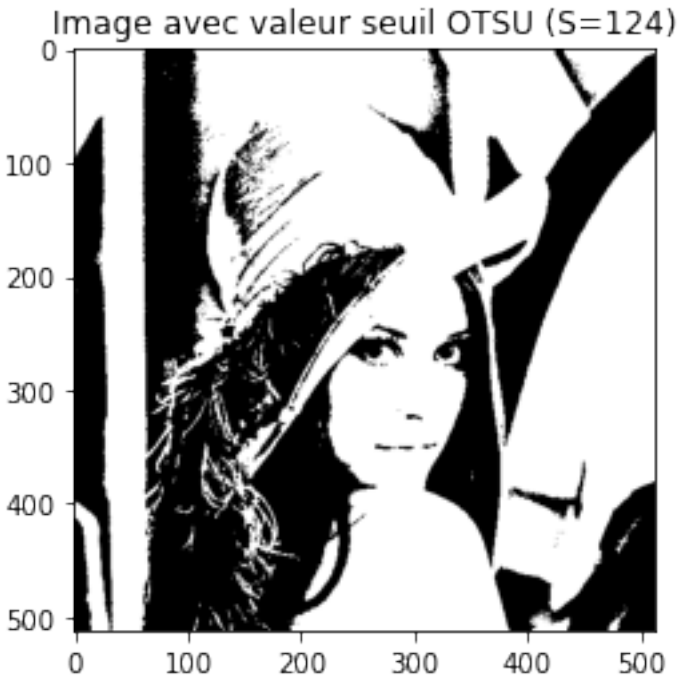


Image avec valeur seuil OTSU (S=124)
0
100
200
300
400
500
0
100
200
300
400
500