

Algoritmo 5

Muñoz Nuñez Ian Emmanuel

Ecualización mediante histograma
Visión Robótica

Imagen usada para el algoritmo



Ejercicio 1

```

# Se importa la libreria opencv para leer y manipular la imagen
import cv2
# Se importa la libreria pyplot de matplotlib para calcular y mostrar los histogramas
import matplotlib.pyplot as plt

# Se lee la imagen 'imagen.jpg' ubicada en el mismo directorio del programa
imagen = cv2.imread("imagen.jpg")
# Se convierte la imagen original a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se asignan los valores de ancho y alto a las variables
grises_m, grises_n = grises.shape[0:2]
# Se calcula el histograma de la imagen original
histograma = cv2.calcHist([grises], [0], None, [100], [0, 256]).flatten()/(grises_m*grises_n)

# Se cambia el tamaño de la imagen a la mitad del tamaño original y se asigna a la variable
# 'imagen_2'
imagen_2 = cv2.resize(imagen, None, fx=0.5, fy=0.5)
# Se convierte la variable 'imagen_2' a escala de grises
grises_2 = cv2.cvtColor(imagen_2, cv2.COLOR_BGR2GRAY)

# Se asignan los valores de ancho y alto a las variables
grises_2_m, grises_2_n = grises_2.shape[0:2]
# Se calcula el histograma de la variable 'imagen_2'
histograma_2 = cv2.calcHist([grises_2], [0], None, [100], [0, 256]).flatten()/(grises_2_m*grises_2_n)

# Se cambia el tamaño de la imagen a un cuarto del tamaño original y se asigna a la variable
# 'imagen_3'
imagen_3 = cv2.resize(imagen, None, fx=0.25, fy=0.25)
# Se convierte la variable 'imagen_3' a escala de grises
grises_3 = cv2.cvtColor(imagen_3, cv2.COLOR_BGR2GRAY)

# Se asignan los valores de ancho y alto a las variables
grises_3_m, grises_3_n = grises_3.shape[0:2]
# Se calcula el histograma de la variable 'imagen_3'
histograma_3 = cv2.calcHist([grises_3], [0], None, [100], [0, 256]).flatten()/(grises_3_m*grises_3_n)

# Se muestra la imagen original con opencv
cv2.imshow("Imagen original", imagen)
# Se muestra la imagen escalada a 1/2 con opencv
cv2.imshow("Imagen 1/2", imagen_2)
# Se muestra la imagen escalada a 1/4 con opencv
cv2.imshow("Imagen 1/4", imagen_3)

# Se agrega el histograma de la imagen original al pyplot
plt.bar(range(len(histograma)), histograma)
# Se agrega el titulo 'Imagen original' al pyplot
plt.title('Imagen original')
# Se muestra el pyplot

```

```
plt.show()
# Se agrega el histograma de la imagen 2 al pyplot
plt.bar(range(len(histograma_2)), histograma_2)
# Se agrega el titulo 'Imagen 1/2' al pyplot
plt.title("Imagen 1/2")
# Se muestra el pyplot
plt.show()
# Se agrega el histograma de la imagen 3 al pyplot
plt.bar(range(len(histograma_3)), histograma_3)
# Se agrega el titulo 'Imagen 1/4' al pyplot
plt.title("Imagen 1/4")
# Se muestra el pyplot
plt.show()

# Opencv espera a que el usuario presione una tecla para terminar el programa
cv2.waitKey(0)
```

Para el ejercicio 1, primero se importan las librerías necesarias, como **opencv** y **matplotlib**. Luego se lee la imagen "imagen.jpg", que se encuentra en el mismo directorio del programa, esta se convierte de BGR a escala de grises, después se obtienen el ancho y alto de la imagen con la función "shape()" y se calcula el histograma de la imagen. Se cambia el tamaño de la imagen a la mitad y se asigna a la variable `imagen_2`, se convierte a escala de grises y se calcula su histograma. Se cambia el tamaño de la imagen a un cuarto de su tamaño original y se convierte a escala de grises, luego se calcula su histograma. Por último se muestran la imagen original, la reducida 1/2 y la reducida 1/4 con **opencv** y se muestran los 3 histogramas con **pyplot**, al final **opencv** espera a que el usuario presione una tecla para terminar el programa.

Imagen original



Imágenes del código del ejercicio 1

```

File Edit Selection View Go Run Terminal Help
main.py - algoritmo_5 - Visual Studio Code
main.py > ...
1 # Se importa la librería opencv para leer y manipular la imagen
2 import cv2
3 # Se importa la librería pyplot de matplotlib para calcular y mostrar los histogramas
4 import matplotlib.pyplot as plt
5
6 # Se lee la imagen 'imagen.jpg' ubicada en el mismo directorio del programa
7 imagen = cv2.imread("imagen.jpg")
8 # Se convierte la imagen original a escala de grises
9 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
10
11 # Se asignan los valores de ancho y alto a las variables
12 grises_m, grises_n = grises.shape[0:2]
13 # Se calcula el histograma de la imagen original
14 histograma = cv2.calcHist([grises], [0], None, [100], [0, 256]).flatten()/(grises_m*grises_n)
15
16 # Se cambia el tamaño de la imagen a la mitad del tamaño original y se asigna a la variable
17 # 'imagen_2'
18 imagen_2 = cv2.resize(imagen, None, fx=0.5, fy=0.5)
19 # Se convierte la variable 'imagen_2' a escala de grises
20 grises_2 = cv2.cvtColor(imagen_2, cv2.COLOR_BGR2GRAY)
21
22 # Se asignan los valores de ancho y alto a las variables
23 grises_2_m, grises_2_n = grises_2.shape[0:2]
24 # Se calcula el histograma de la variable 'imagen_2'

```

Ln 67, Col 1 Spaces: 4 UTF-8 CRLF Python

```

File Edit Selection View Go Run Terminal Help
main.py - algoritmo_5 - Visual Studio Code
main.py > ...
24 # Se calcula el histograma de la variable 'imagen_2'
25 histograma_2 = cv2.calcHist([grises_2], [0], None, [100], [0, 256]).flatten()/(grises_2_m*grises_2_n)
26
27 # Se cambia el tamaño de la imagen a un cuarto del tamaño original y se asigna a la variable
28 # 'imagen_3'
29 imagen_3 = cv2.resize(imagen, None, fx=0.25, fy=0.25)
30 # Se convierte la variable 'imagen_3' a escala de grises
31 grises_3 = cv2.cvtColor(imagen_3, cv2.COLOR_BGR2GRAY)
32
33 # Se asignan los valores de ancho y alto a las variables
34 grises_3_m, grises_3_n = grises_3.shape[0:2]
35 # Se calcula el histograma de la variable 'imagen_3'
36 histograma_3 = cv2.calcHist([grises_3], [0], None, [100], [0, 256]).flatten()/(grises_3_m*grises_3_n)
37
38 # Se muestra la imagen original con opencv
39 cv2.imshow("Imagen original", imagen)
40 # Se muestra la imagen escalada a 1/2 con opencv
41 cv2.imshow("Imagen 1/2", imagen_2)
42 # Se muestra la imagen escalada a 1/4 con opencv
43 cv2.imshow("Imagen 1/4", imagen_3)
44
45 # Se agrega el histograma de la imagen original al pyplot
46 plt.bar(range(len(histograma)), histograma)
47 # Se agrega el título 'Imagen original' al pyplot

```

Ln 67, Col 1 Spaces: 4 UTF-8 CRLF Python

```

File Edit Selection View Go Run Terminal Help
main.py - algoritmo_5 - Visual Studio Code
main.py > ...
47 # Se agrega el título 'Imagen original' al pyplot
48 plt.title("Imagen original")
49 # Se muestra el pyplot
50 plt.show()
51 # Se agrega el histograma de la imagen 2 al pyplot
52 plt.bar(range(len(histograma_2)), histograma_2)
53 # Se agrega el título 'Imagen 1/2' al pyplot
54 plt.title("Imagen 1/2")
55 # Se muestra el pyplot
56 plt.show()
57 # Se agrega el histograma de la imagen 3 al pyplot
58 plt.bar(range(len(histograma_3)), histograma_3)
59 # Se agrega el título 'Imagen 1/4' al pyplot
60 plt.title("Imagen 1/4")
61 # Se muestra el pyplot
62 plt.show()
63
64 # OpenCV espera a que el usuario presione una tecla para terminar el programa
65 cv2.waitKey(0)
66 cv2.destroyAllWindows()
67


```

Ln 67, Col 1 Spaces: 4 UTF-8 CRLF Python

Imagen original

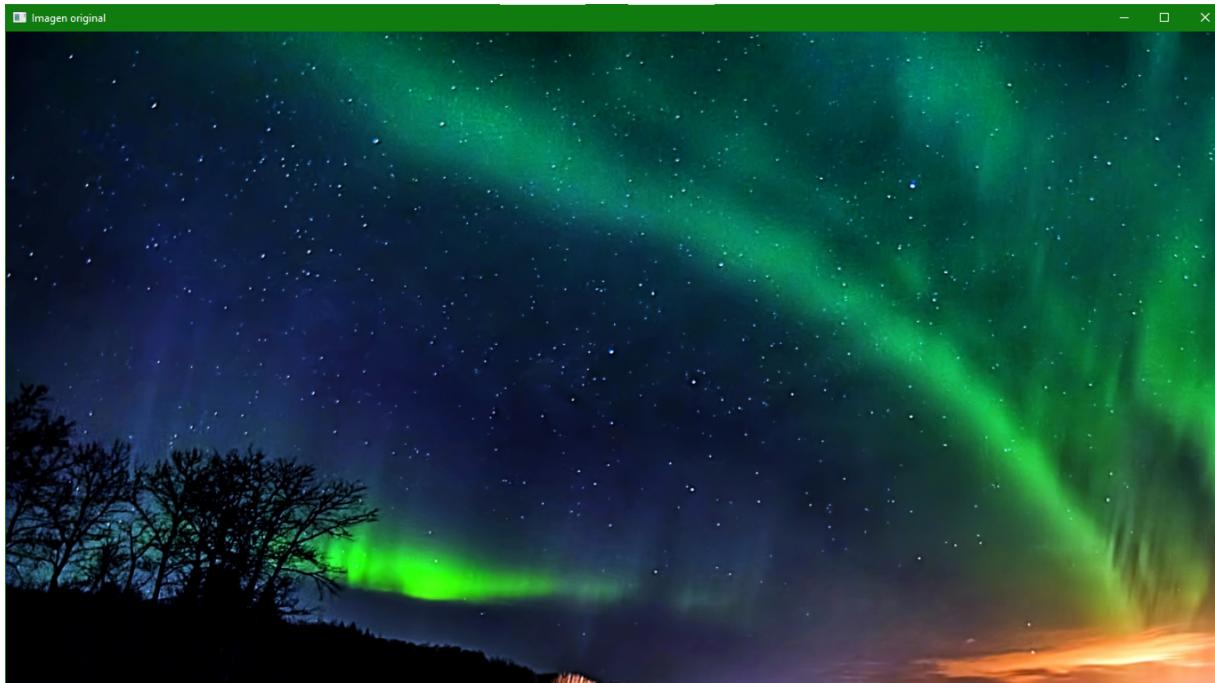


Imagen reducida $\frac{1}{2}$

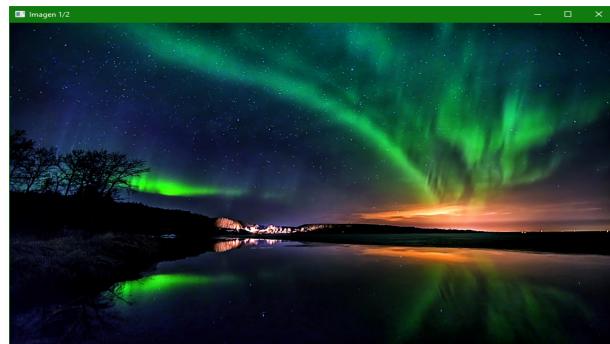
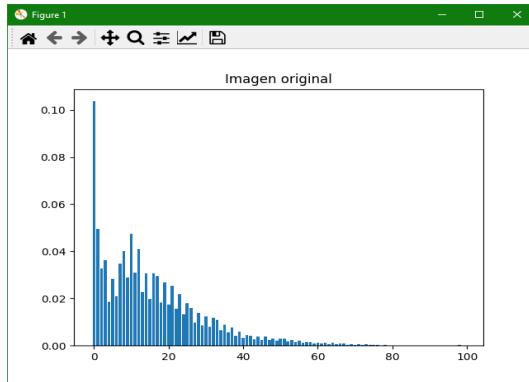


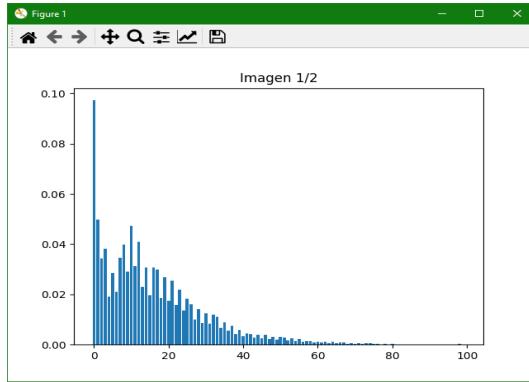
Imagen reducida $\frac{1}{4}$



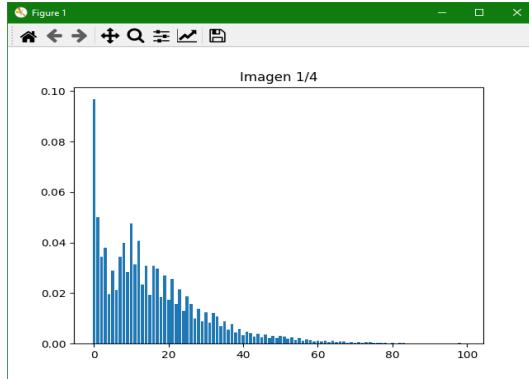
Histograma de la imagen original



Histograma de la imagen reducida $\frac{1}{2}$



Histograma de la imagen reducida $\frac{1}{4}$



Note que los histogramas en este ejercicio son muy parecidos, me imagino que es porque no se hace ningún cambio en su color, pero si se puede notar un cambio en sus dimensiones (aunque no muy grande) pues se cambio el tamaño de la imagen.

Ejercicio 2

```

# Se importa la libreria opencv para leer y manipular la imagen
import cv2
# Se importa la libreria pyplot de matplotlib para calcular y mostrar los histogramas
import matplotlib.pyplot as plt

# Se lee la imagen 'imagen.jpg' ubicada en el mismo directorio del programa
imagen = cv2.imread("imagen.jpg")
imagen = cv2.resize(imagen, None, fx=0.25, fy=0.25)
# Se convierte la imagen original a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se asignan los valores de ancho y alto a las variables
grises_m, grises_n = grises.shape[0:2]
# Se calcula el histograma de la imagen original
histograma = cv2.calcHist([grises], [0], None, [100], [0, 256]).flatten()/(grises_m*grises_n)

# Se dividen los 3 canales de la imagen con la funcion split()
b, g, r = cv2.split(imagen)
# Se multiplica el canal R por 1.5
r = cv2.multiply(r, 1.5)
# Se suma 50 al canal R
r = cv2.add(r, 50)

# Se multiplica el canal G por 1.5
g = cv2.multiply(g, 1.5)
# Se suma 50 al canal G
g = cv2.add(g, 50)

# Se multiplica el canal B por 1.5
b = cv2.multiply(b, 1.5)
# Se suma 50 al canal B
b = cv2.add(b, 50)

# Se mezclan los canales R, G y B modificados
imagen_rgb = cv2.merge((b, g, r))
# Se convierte la imagen RGB a escala de grises
grises_rgb = cv2.cvtColor(imagen_rgb, cv2.COLOR_BGR2GRAY)
# Se asignan los valores de ancho y alto a las variables
grises_rgb_m, grises_rgb_n = grises_rgb.shape[0:2]
# Se calcula el histograma de la imagen RGB
histograma_rgb = cv2.calcHist([grises_rgb], [0], None, [100], [0, 256]).flatten()/(grises_rgb_m*grises_rgb_n)

# Se ecualiza el canal R
r = cv2.equalizeHist(r)
# Se ecualiza el canal G
g = cv2.equalizeHist(g)
# Se ecualiza el canal B
b = cv2.equalizeHist(b)

```

```

# Se mezclan los canales R, G y B ecualizados
imagen_equal = cv2.merge((b, g, r))
# Se convierte la imagen ecualizada a escala de grises
grises_equal = cv2.cvtColor(imagen_equal, cv2.COLOR_BGR2GRAY)

# Se asignan los valores de ancho y alto a las variables
grises_equal_m, grises_equal_n = grises_equal.shape[0:2]
# Se calcula el histograma de la imagen ecualizada
histograma_equal = cv2.calcHist([grises_equal], [0], None, [100], [0,
256]).flatten()/(grises_equal_m*grises_equal_n)

# Se muestra la imagen original con opencv
cv2.imshow("Imagen original", imagen)
# Se muestra la imagen modificada con opencv
cv2.imshow("Imagen RGB", imagen_rgb)
# Se muestra la imagen ecualizada con opencv
cv2.imshow("Imagen ecualizada", imagen_equal)

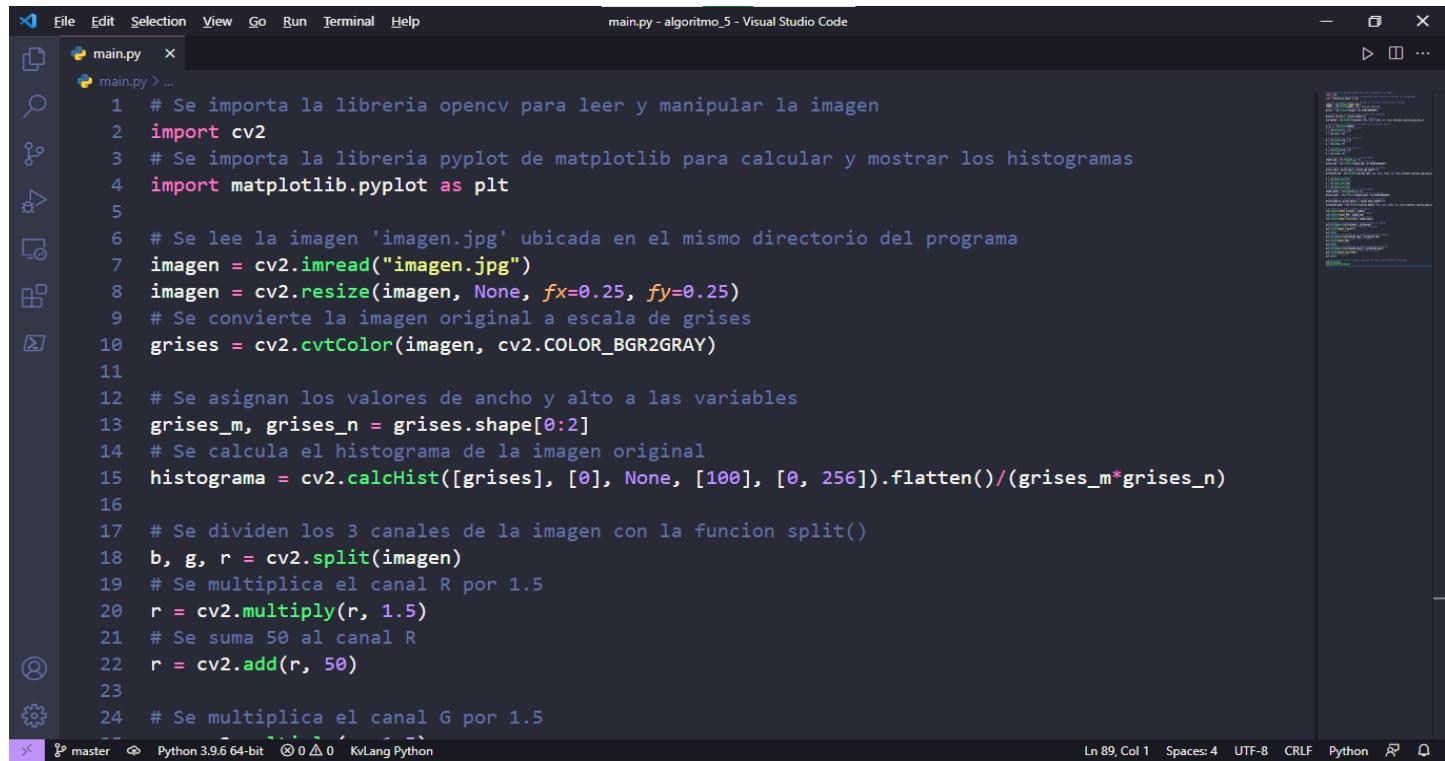
# Se agrega el histograma de la imagen original al pyplot
plt.bar(range(len(histograma)), histograma)
# Se agrega el titulo 'Imagen original' al pyplot
plt.title("Imagen original")
# Se muestra el pyplot
plt.show()
# Se agrega el histograma de la imagen modificada al pyplot
plt.bar(range(len(histograma_rgb)), histograma_rgb)
# Se agrega el titulo 'Imagen RGB' al pyplot
plt.title("Imagen RGB")
# Se muestra el pyplot
plt.show()
# Se agrega el histograma de la imagen ecualizada al pyplot
plt.bar(range(len(histograma_equal)), histograma_equal)
# Se agrega el titulo 'Imagen ecualizada' al pyplot
plt.title("Imagen ecualizada")
# Se muestra el pyplot
plt.show()

# Opencv espera a que el usuario presione una tecla para terminar el programa
cv2.waitKey(0)

```

Para el algoritmo del ejercicio 2, primero se importan las librerías necesarias, como ***opencv*** y ***matplotlib***. Luego se lee la imagen "imagen.jpg" que se encuentra en el mismo directorio del programa, esta se escala a un cuarto de la misma, pues es muy grande para poder verla fácilmente, luego se cambia a escala de grises para poder calcular su histograma. Después se dividen los canales R, G y B con la función "split()", cada canal es multiplicado y sumado por valores distintos. Se mezclan los canales en una sola imagen y esta se cambia a escala de grises para también calcular su histograma. Luego se ecualizan los tres canales y se mezclan en una nueva imagen, al igual que las anteriores se convierte a escala de grises para calcular su histograma. Por último se muestran las imágenes con ***opencv*** y se muestra cada histograma con ***pyplot***, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

Imágenes del código del ejercicio 2



```

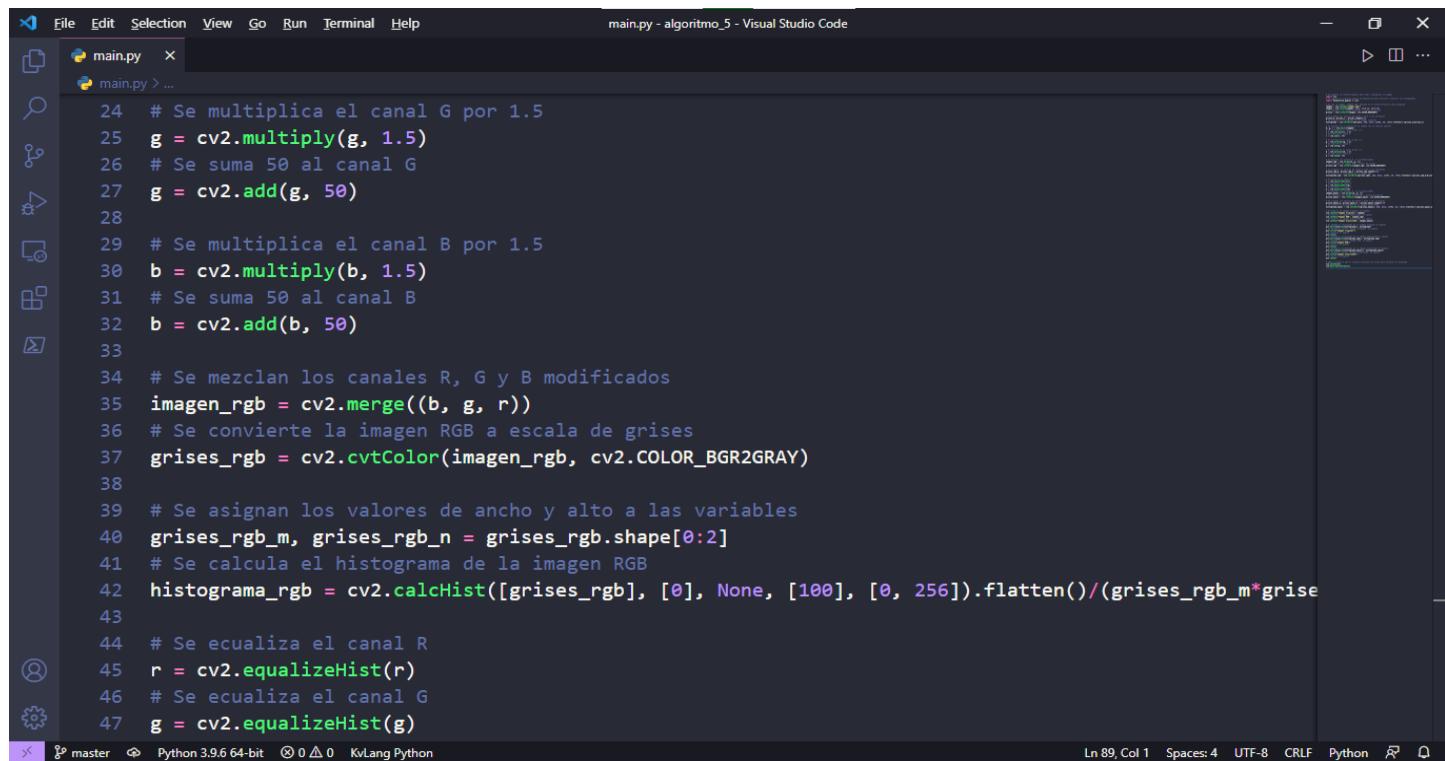
File Edit Selection View Go Run Terminal Help
main.py - algoritmo_5 - Visual Studio Code

main.py
main.py > ...

1 # Se importa la libreria opencv para leer y manipular la imagen
2 import cv2
3 # Se importa la libreria pyplot de matplotlib para calcular y mostrar los histogramas
4 import matplotlib.pyplot as plt
5
6 # Se lee la imagen 'imagen.jpg' ubicada en el mismo directorio del programa
7 imagen = cv2.imread("imagen.jpg")
8 imagen = cv2.resize(imagen, None, fx=0.25, fy=0.25)
9 # Se convierte la imagen original a escala de grises
10 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
11
12 # Se asignan los valores de ancho y alto a las variables
13 grises_m, grises_n = grises.shape[0:2]
14 # Se calcula el histograma de la imagen original
15 histograma = cv2.calcHist([grises], [0], None, [100], [0, 256]).flatten()/(grises_m*grises_n)
16
17 # Se dividen los 3 canales de la imagen con la funcion split()
18 b, g, r = cv2.split(imagen)
19 # Se multiplica el canal R por 1.5
20 r = cv2.multiply(r, 1.5)
21 # Se suma 50 al canal R
22 r = cv2.add(r, 50)
23
24 # Se multiplica el canal G por 1.5
25 g = cv2.multiply(g, 1.5)
26 # Se suma 50 al canal G
27 g = cv2.add(g, 50)
28
29 # Se multiplica el canal B por 1.5
30 b = cv2.multiply(b, 1.5)
31 # Se suma 50 al canal B
32 b = cv2.add(b, 50)
33
34 # Se mezclan los canales R, G y B modificados
35 imagen_rgb = cv2.merge((b, g, r))
36 # Se convierte la imagen RGB a escala de grises
37 grises_rgb = cv2.cvtColor(imagen_rgb, cv2.COLOR_BGR2GRAY)
38
39 # Se asignan los valores de ancho y alto a las variables
40 grises_rgb_m, grises_rgb_n = grises_rgb.shape[0:2]
41 # Se calcula el histograma de la imagen RGB
42 histograma_rgb = cv2.calcHist([grises_rgb], [0], None, [100], [0, 256]).flatten()/(grises_rgb_m*grises_rgb_n)
43
44 # Se iguala el canal R
45 r = cv2.equalizeHist(r)
46 # Se iguala el canal G
47 g = cv2.equalizeHist(g)

Ln 89, Col 1 Spaces: 4 UTF-8 CRLF Python ⚙️

```



```

File Edit Selection View Go Run Terminal Help
main.py - algoritmo_5 - Visual Studio Code

main.py
main.py > ...

24 # Se multiplica el canal G por 1.5
25 g = cv2.multiply(g, 1.5)
26 # Se suma 50 al canal G
27 g = cv2.add(g, 50)
28
29 # Se multiplica el canal B por 1.5
30 b = cv2.multiply(b, 1.5)
31 # Se suma 50 al canal B
32 b = cv2.add(b, 50)
33
34 # Se mezclan los canales R, G y B modificados
35 imagen_rgb = cv2.merge((b, g, r))
36 # Se convierte la imagen RGB a escala de grises
37 grises_rgb = cv2.cvtColor(imagen_rgb, cv2.COLOR_BGR2GRAY)
38
39 # Se asignan los valores de ancho y alto a las variables
40 grises_rgb_m, grises_rgb_n = grises_rgb.shape[0:2]
41 # Se calcula el histograma de la imagen RGB
42 histograma_rgb = cv2.calcHist([grises_rgb], [0], None, [100], [0, 256]).flatten()/(grises_rgb_m*grises_rgb_n)
43
44 # Se iguala el canal R
45 r = cv2.equalizeHist(r)
46 # Se iguala el canal G
47 g = cv2.equalizeHist(g)

Ln 89, Col 1 Spaces: 4 UTF-8 CRLF Python ⚙️

```

main.py - algoritmo_5 - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
main.py x
main.py > ...
47 g = cv2.equalizeHist(g)
48 # Se ecualiza el canal B
49 b = cv2.equalizeHist(b)
50 # Se mezclan los canales R, G y B ecualizados
51 imagen_equal = cv2.merge((b, g, r))
52 # Se convierte la imagen ecualizada a escala de grises
53 grises_equal = cv2.cvtColor(imagen_equal, cv2.COLOR_BGR2GRAY)
54
55 # Se asignan los valores de ancho y alto a las variables
56 grises_equal_m, grises_equal_n = grises_equal.shape[0:2]
57 # Se calcula el histograma de la imagen ecualizada
58 histograma_equal = cv2.calcHist([grises_equal], [0], None, [100], [0, 256]).flatten()/(grises_equal_m * grises_equal_n)
59
60 # Se muestra la imagen original con opencv
61 cv2.imshow("Imagen original", imagen)
62 # Se muestra la imagen modificada con opencv
63 cv2.imshow("Imagen RGB", imagen_rgb)
64 # Se muestra la imagen ecualizada con opencv
65 cv2.imshow("Imagen ecualizada", imagen_equal)
66
67 # Se agrega el histograma de la imagen original al pyplot
68 plt.bar(range(len(histograma)), histograma)
69 # Se agrega el titulo 'Imagen original' al pyplot
70 plt.title("Imagen original")
```

Ln 89, Col 1 Spaces: 4 UTF-8 CRLF Python

main.py - algoritmo_5 - Visual Studio Code

```
File Edit Selection View Go Run Terminal Help
main.py x
main.py > ...
70 plt.title("Imagen original")
71 # Se muestra el pyplot
72 plt.show()
73 # Se agrega el histograma de la imagen modificada al pyplot
74 plt.bar(range(len(histograma_rgb)), histograma_rgb)
75 # Se agrega el titulo 'Imagen RGB' al pyplot
76 plt.title("Imagen RGB")
77 # Se muestra el pyplot
78 plt.show()
79 # Se agrega el histograma de la imagen ecualizada al pyplot
80 plt.bar(range(len(histograma_equal)), histograma_equal)
81 # Se agrega el titulo 'Imagen ecualizada' al pyplot
82 plt.title("Imagen ecualizada")
83 # Se muestra el pyplot
84 plt.show()
85
86 # Opencv espera a que el usuario presione una tecla para terminar el programa
87 cv2.waitKey(0)
88 cv2.destroyAllWindows()
89 |
```

Ln 89, Col 1 Spaces: 4 UTF-8 CRLF Python

Imagen original

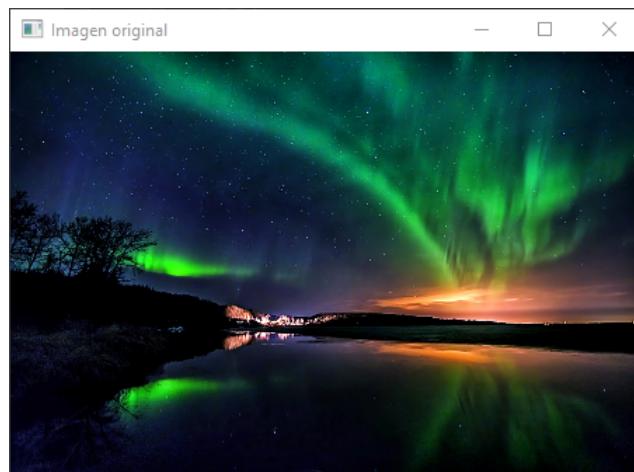


Imagen modificada 1

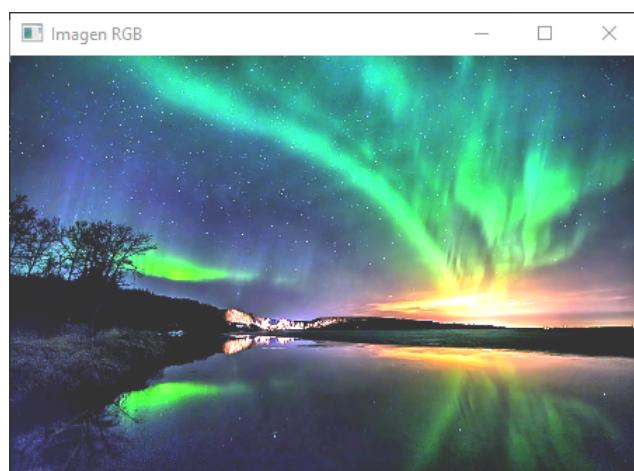
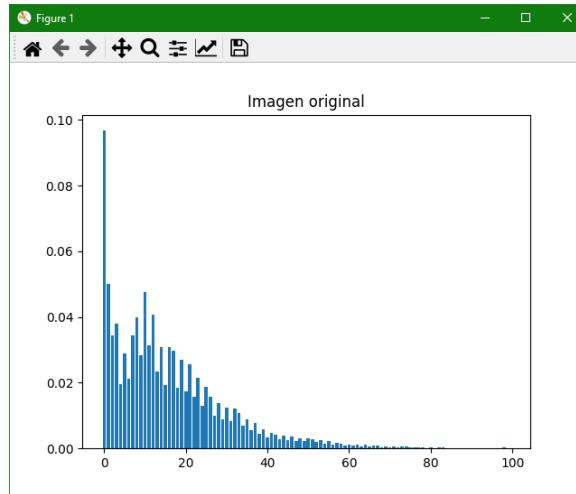


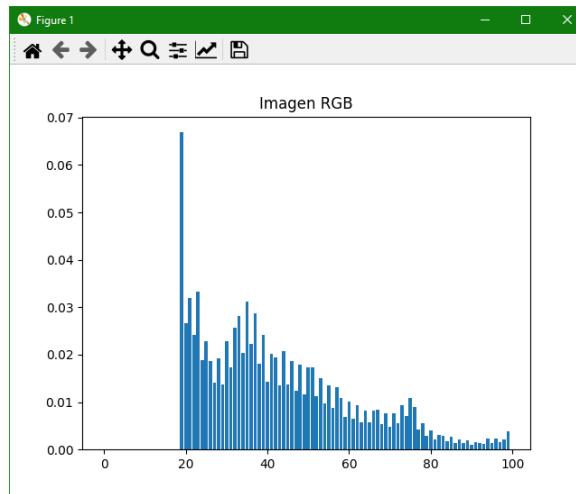
Imagen ecualizada 1



Histograma de la imagen original



Histograma de la imagen modificada 1



Histograma de la imagen ecualizada 1

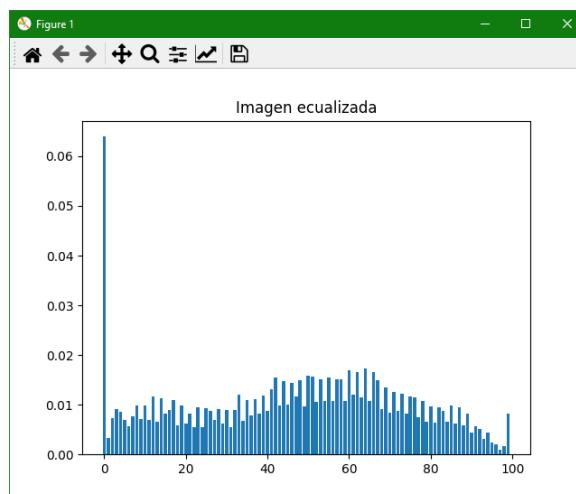


Imagen original

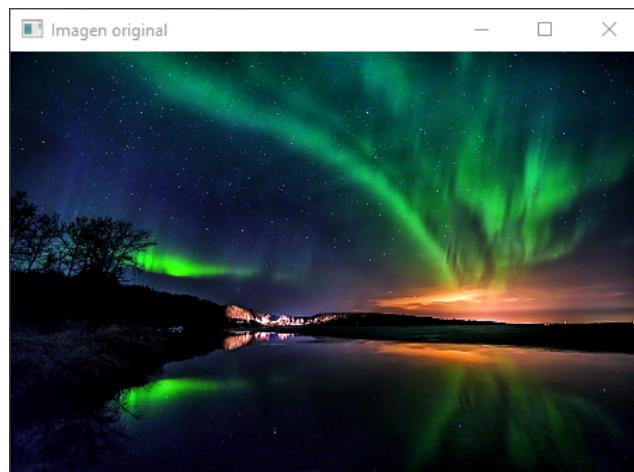


Imagen modificada 2

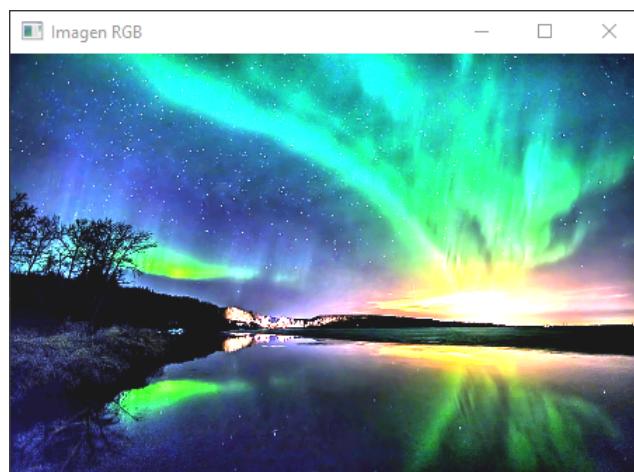
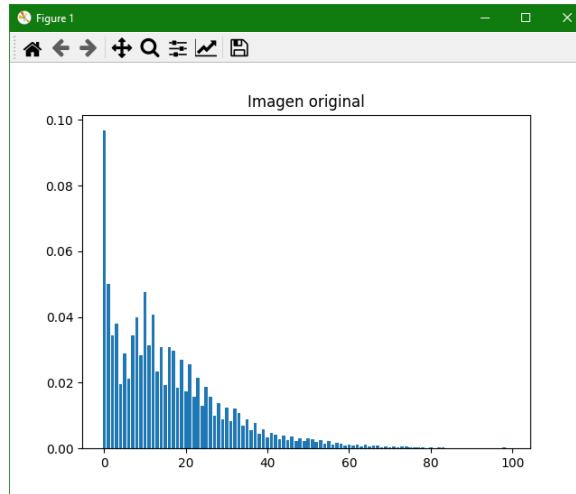


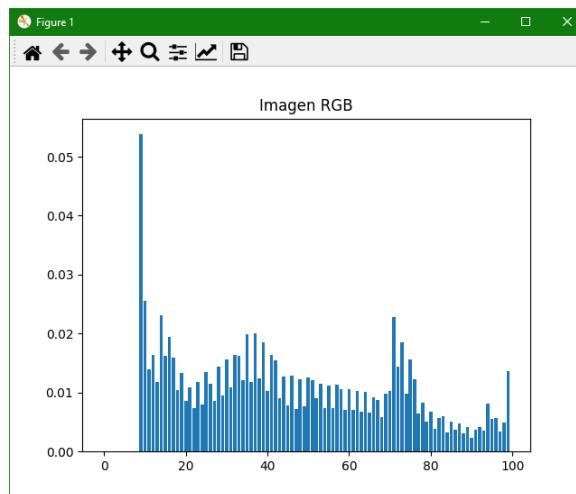
Imagen ecualizada 2



Histograma de la imagen original



Histograma de la imagen modificada 2



Histograma de la imagen ecualizada 2

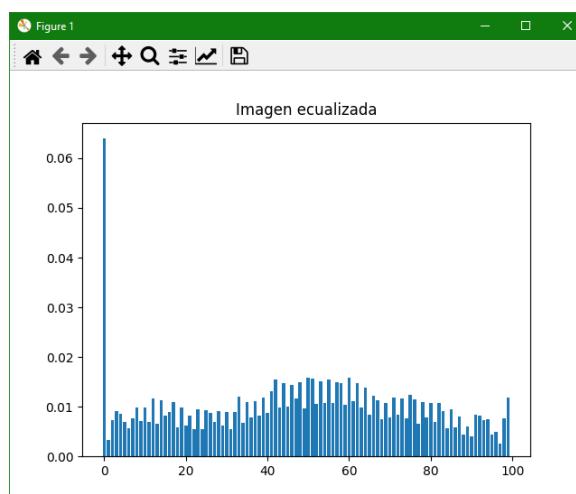


Imagen original

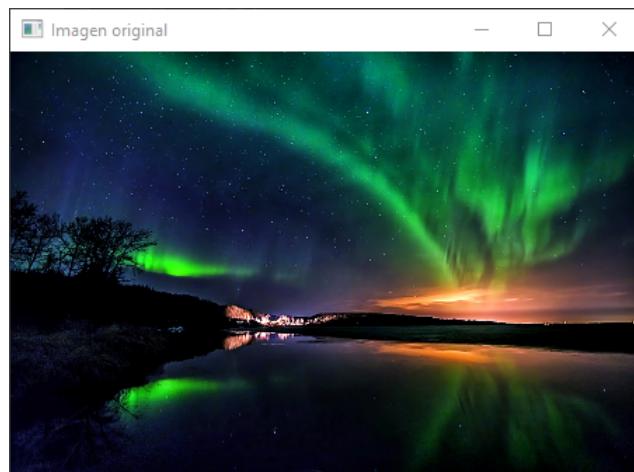


Imagen modificada 3

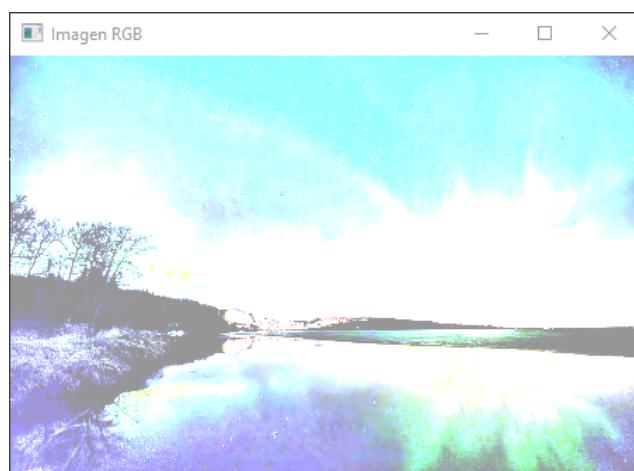
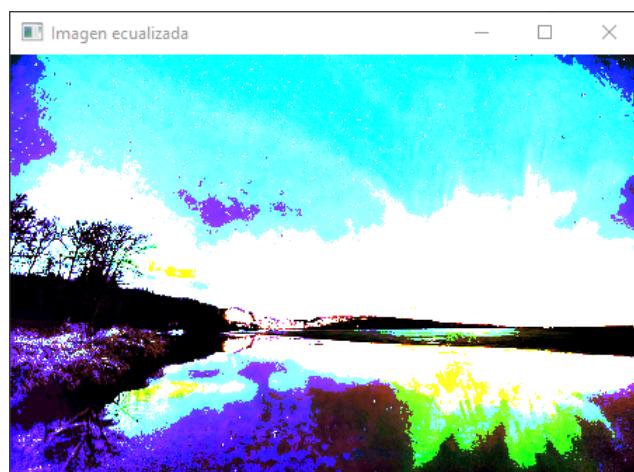
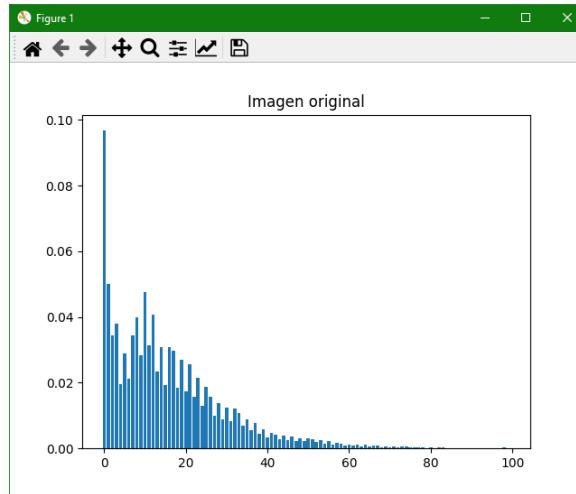


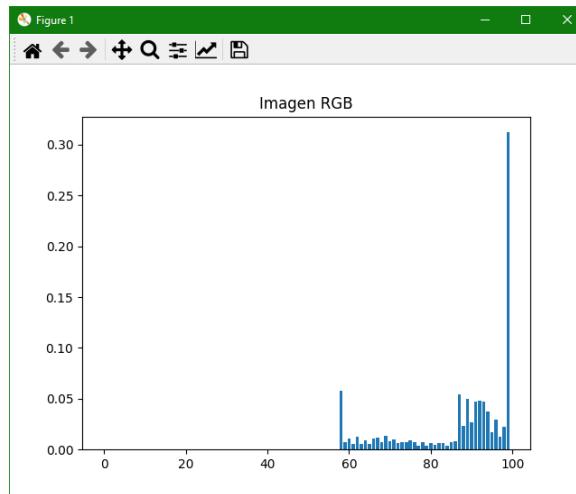
Imagen ecualizada 3



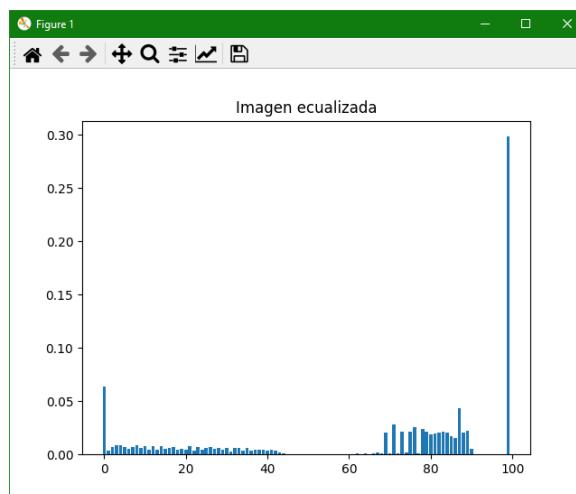
Histograma de la imagen original



Histograma de la imagen modificada 3



Histograma de la imagen ecualizada 3



En este ejercicio si se noto un cambio muy grande entre cada uno de los histogramas, pues esta vez no se modificó la imagen, sino que se modificaron los valores de esta, ya sea que se hayan hecho más brillantes o más oscuros, por lo que se puede concluir que el cambio a la exposición de la imagen depende de las modificaciones que se le haga a los valores de los pixeles, no a la cantidad de estos.