

Proyecto Integrador Unidad 2
Muñoz Nuñez Ian Emmanuel
Proyecto Integrador
Visión Robótica

Imágenes usadas en el proyecto



Imagen subexpuesta

```

# Se importa la libreria opencv para leer, mostrar y manipular la imagen
import cv2
# Se importa la libreria numpy como np para calcular el valor maximo del histograma
import numpy as np
# Se importa la libreria pyplot para mostrar el hisotgrama de la imagen
import matplotlib.pyplot as plt

# Se carga la imagen 'subexpuesta.jpg' que se encuentra en el mismo directorio del programa
imagen = cv2.imread("subexpuesta.jpg")
# Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
if imagen.shape[0] >550 or imagen.shape[1] >1100:
    imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
# La imagen se convierte a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se obtienen el alto y el ancho de la imagen
alto, ancho = imagen.shape[0:2]
# Se calcula el histograma de la imagen
histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)

# Se obtiene el valor maximo del histograma
maximo = np.argmax(histograma)

# Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
if maximo == 4 and histograma[4] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
    imagen_equal = cv2.merge((b, g, r))

    # Se muestra la imagen ecualizada
    cv2.imshow("Imagen ecualizada", imagen_equal)

    # Se agrega el texto 'Sobreeexpuesta' a la imagen
    cv2.putText(imagen, "Sobreeexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
elif maximo == 0 and histograma[0] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
imagen_equal = cv2.merge((b, g, r))

# Se muestra la imagen ecualizada
cv2.imshow("Imagen ecualizada", imagen_equal)

# Se agrega el texto 'Subexpuesta' a la imagen
cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si no se cumple ninguna de las condiciones anteriores...
else:
    # Se agrega el texto 'Buena exposicion' a la imagen
    cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Se muestra la imagen original
cv2.imshow("Imagen", imagen)

# Se agrega el histograma al pyplot
plt.bar(range(len(histograma)), histograma)
# Se muestra el pyplot
plt.show()

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

```

Para el código del algoritmo primero se importan las librerías necesarias para realizarlo, como ***opencv***, ***numpy*** y ***pyplot***. Luego se carga la imagen 'subexpuesta.jpg' que se encuentra en el mismo directorio del programa, si la imagen tiene un alto mayor a 550 o un ancho mayor a 1100 se escala la imagen a la mitad, esta se convierte a escala de grises y se calcula su histograma. Después de calcular el histograma se obtiene el valor máximo de este, si el valor es 4 y el histograma en la posición 4 es mayor a 0.3 significa que está sobreexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Sobreexpuesta". Si el valor máximo es 0 y el histograma en la posición 0 es mayor a 0.3 significa que está subexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Subexpuesta". Si no se cumple ninguna de las condiciones anteriores solo se agregara a la imagen original el texto "Buena exposición". Por último, se muestra la imagen original con el texto, se agrega el histograma al ***pyplot*** y se muestra, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

Imágenes del código

```

main.py M X
main.py
1 # Se importa la libreria opencv para leer, mostrar y manipular la imagen
2 import cv2
3 # Se importa la libreria numpy como np para calcular el valor maximo del histograma
4 import numpy as np
5 # Se importa la libreria pyplot para mostrar el histograma de la imagen
6 import matplotlib.pyplot as plt
7
8 # Se carga la imagen 'subexpuesta.jpg' que se encuentra en el mismo directorio del programa
9 imagen = cv2.imread("subexpuesta.jpg")
10 # Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
11 if imagen.shape[0] > 550 or imagen.shape[1] > 1100:
12     imagen = cv2.resize(imagen, None, None, fx=0.5, fy=0.5)
13 # La imagen se convierte a escala de grises
14 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
15
16 # Se obtienen el alto y el ancho de la imagen
17 alto, ancho = imagen.shape[0:2]
18 # Se calcula el histograma de la imagen
19 histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)
20
21 # Se obtiene el valor maximo del histograma
22 maximo = np.argmax(histograma)
23
24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...

```

```

24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
25 if maximo == 4 and histograma[4] > 0.3:
26     # Se dividen los canales con la funcion split
27     b, g, r = cv2.split(imagen)
28     # Se ecualiza el canal R
29     r = cv2.equalizeHist(r)
30     # Se ecualiza el canal G
31     g = cv2.equalizeHist(g)
32     # Se ecualiza el canal B
33     b = cv2.equalizeHist(b)
34
35     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
36     imagen_equal = cv2.merge((b, g, r))
37
38     # Se muestra la imagen ecualizada
39     cv2.imshow("Imagen ecualizada", imagen_equal)
40
41     # Se agrega el texto 'Sobreexpuesta' a la imagen
42     cv2.putText(imagen, "Sobreexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
43
44 # Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
45 elif maximo == 0 and histograma[0] > 0.3:
46     # Se dividen los canales con la funcion split
47     b, g, r = cv2.split(imagen)

```

```

47     b, g, r = cv2.split(imagen)
48     # Se ecualiza el canal R
49     r = cv2.equalizeHist(r)
50     # Se ecualiza el canal G
51     g = cv2.equalizeHist(g)
52     # Se ecualiza el canal B
53     b = cv2.equalizeHist(b)
54
55     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
56     imagen_equal = cv2.merge((b, g, r))
57
58     # Se muestra la imagen ecualizada
59     cv2.imshow("Imagen ecualizada", imagen_equal)
60
61     # Se agrega el texto 'Subexpuesta' a la imagen
62     cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
63
64 # Si no se cumple ninguna de las condiciones anteriores...
65 else:
66     # Se agrega el texto 'Buena exposicion' a la imagen
67     cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
68
69 # Se muestra la imagen original
70 cv2.imshow("Imagen", imagen)

```

```

71
72     # Se agrega el histograma al pyplot
73     plt.bar(range(len(histograma)), histograma)
74     # Se muestra el pyplot
75     plt.show()
76
77     #opencv espera a que el usuario presione una tecla para terminar el programa
78     cv2.waitKey(0)
79

```

Imagen subexpuesta

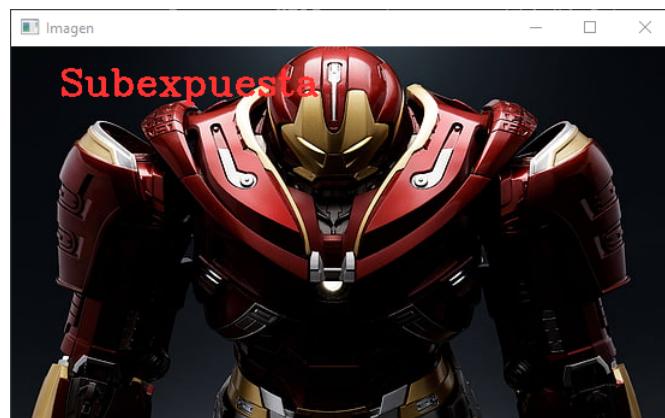


Imagen subexpuesta ecualizada



Histograma de la imagen subexpuesta

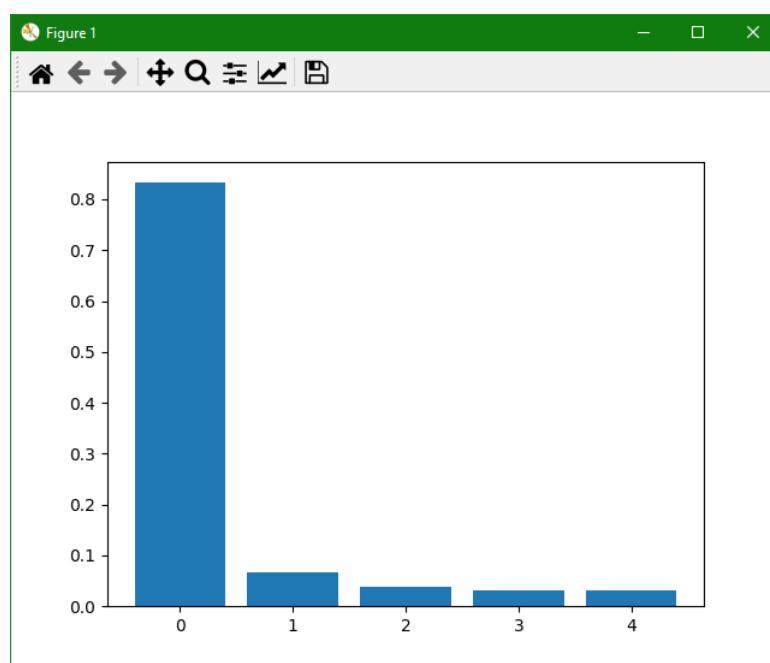


Imagen sobreexpuesta

```

# Se importa la libreria opencv para leer, mostrar y manipular la imagen
import cv2
# Se importa la libreria numpy como np para calcular el valor maximo del histograma
import numpy as np
# Se importa la libreria pyplot para mostrar el hisotgrama de la imagen
import matplotlib.pyplot as plt

# Se carga la imagen 'sobreexpuesta.jpg' que se encuentra en el mismo directorio del programa
imagen = cv2.imread("sobreexpuesta.jpg")
# Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
if imagen.shape[0] >550 or imagen.shape[1] >1100:
    imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
# La imagen se convierte a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se obtienen el alto y el ancho de la imagen
alto, ancho = imagen.shape[0:2]
# Se calcula el histograma de la imagen
histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)

# Se obtiene el valor maximo del histograma
maximo = np.argmax(histograma)

# Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
if maximo == 4 and histograma[4] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
    imagen_equal = cv2.merge((b, g, r))

    # Se muestra la imagen ecualizada
    cv2.imshow("Imagen ecualizada", imagen_equal)

    # Se agrega el texto 'Sobreexpuesta' a la imagen
    cv2.putText(imagen, "Sobreexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
elif maximo == 0 and histograma[0] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
imagen_equal = cv2.merge((b, g, r))

# Se muestra la imagen ecualizada
cv2.imshow("Imagen ecualizada", imagen_equal)

# Se agrega el texto 'Subexpuesta' a la imagen
cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si no se cumple ninguna de las condiciones anteriores...
else:
    # Se agrega el texto 'Buena exposicion' a la imagen
    cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Se muestra la imagen original
cv2.imshow("Imagen", imagen)

# Se agrega el histograma al pyplot
plt.bar(range(len(histograma)), histograma)
# Se muestra el pyplot
plt.show()

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

```

Para el código del algoritmo primero se importan las librerías necesarias para realizarlo, como ***opencv***, ***numpy*** y ***pyplot***. Luego se carga la imagen 'sobreexpuesta.jpg' que se encuentra en el mismo directorio del programa, si la imagen tiene un alto mayor a 550 o un ancho mayor a 1100 se escala la imagen a la mitad, esta se convierte a escala de grises y se calcula su histograma. Después de calcular el histograma se obtiene el valor máximo de este, si el valor es 4 y el histograma en la posición 4 es mayor a 0.3 significa que está sobreexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Sobreexpuesta". Si el valor máximo es 0 y el histograma en la posición 0 es mayor a 0.3 significa que está subexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Subexpuesta". Si no se cumple ninguna de las condiciones anteriores solo se agregara a la imagen original el texto "Buena exposición". Por último, se muestra la imagen original con el texto, se agrega el histograma al ***pyplot*** y se muestra, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

Imágenes del código

```

1 # Se importa la libreria opencv para leer, mostrar y manipular la imagen
2 import cv2
3 # Se importa la libreria numpy como np para calcular el valor maximo del histograma
4 import numpy as np
5 # Se importa la libreria pyplot para mostrar el histograma de la imagen
6 import matplotlib.pyplot as plt
7
8 # Se carga la imagen 'sobreexpuesta.jpg' que se encuentra en el mismo directorio del programa
9 imagen = cv2.imread("sobreexpuesta.jpg")
10 # Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
11 if imagen.shape[0] > 550 or imagen.shape[1] > 1100:
12     imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
13 # La imagen se convierte a escala de grises
14 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
15
16 # Se obtienen el alto y el ancho de la imagen
17 alto, ancho = imagen.shape[0:2]
18 # Se calcula el histograma de la imagen
19 histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)
20
21 # Se obtiene el valor maximo del histograma
22 maximo = np.argmax(histograma)
23
24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...

```

```

24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
25 if maximo == 4 and histograma[4] > 0.3:
26     # Se dividen los canales con la funcion split
27     b, g, r = cv2.split(imagen)
28     # Se ecualiza el canal R
29     r = cv2.equalizeHist(r)
30     # Se ecualiza el canal G
31     g = cv2.equalizeHist(g)
32     # Se ecualiza el canal B
33     b = cv2.equalizeHist(b)
34
35     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
36     imagen_equal = cv2.merge((b, g, r))
37
38     # Se muestra la imagen ecualizada
39     cv2.imshow("Imagen ecualizada", imagen_equal)
40
41     # Se agrega el texto 'Sobreexpuesta' a la imagen
42     cv2.putText(imagen, "Sobreexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
43
44 # Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
45 elif maximo == 0 and histograma[0] > 0.3:
46     # Se dividen los canales con la funcion split
47     b, g, r = cv2.split(imagen)

```

```

47     b, g, r = cv2.split(imagen)
48     # Se ecualiza el canal R
49     r = cv2.equalizeHist(r)
50     # Se ecualiza el canal G
51     g = cv2.equalizeHist(g)
52     # Se ecualiza el canal B
53     b = cv2.equalizeHist(b)
54
55     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
56     imagen_equal = cv2.merge((b, g, r))
57
58     # Se muestra la imagen ecualizada
59     cv2.imshow("Imagen ecualizada", imagen_equal)
60
61     # Se agrega el texto 'Subexpuesta' a la imagen
62     cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
63
64 # Si no se cumple ninguna de las condiciones anteriores...
65 else:
66     # Se agrega el texto 'Buena exposicion' a la imagen
67     cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
68
69 # Se muestra la imagen original
70 cv2.imshow("Imagen", imagen)

```

```

71
72     # Se agrega el histograma al pyplot
73     plt.bar(range(len(histograma)), histograma)
74
75     # Se muestra el pyplot
76     plt.show()
77
78 #opencv espera a que el usuario presione una tecla para terminar el programa
79 cv2.waitKey(0)

```

Imagen sobreexpuesta

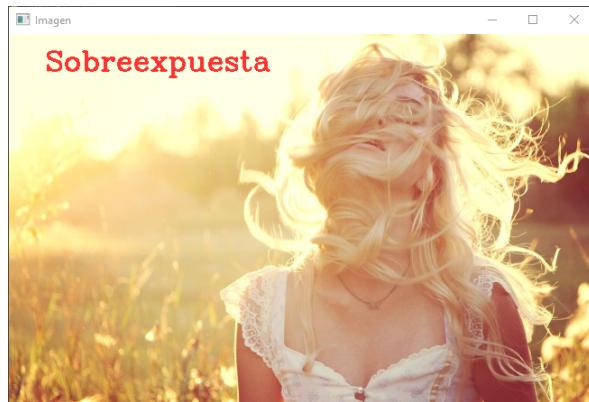
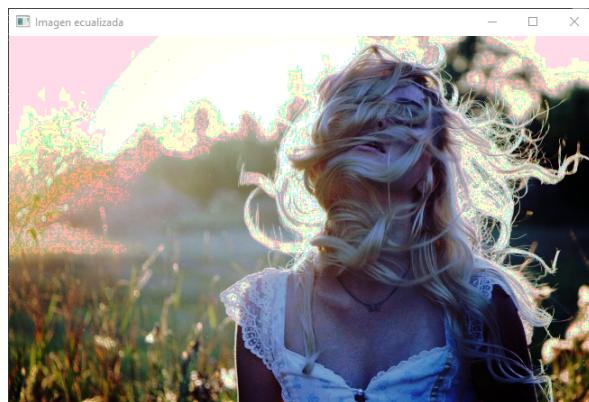


Imagen sobreexpuesta ecualizada



Histograma de la imagen sobreexpuesta

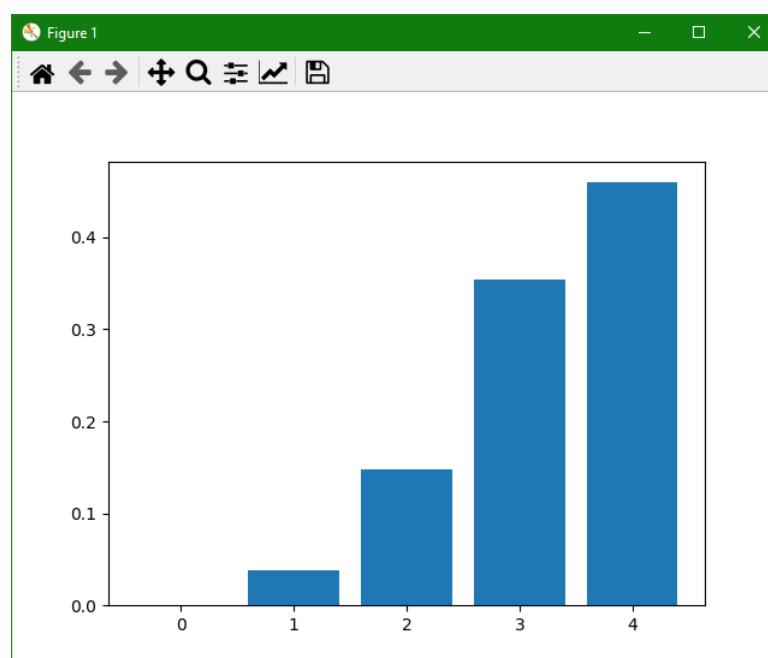


Imagen con buena exposición

```

# Se importa la libreria opencv para leer, mostrar y manipular la imagen
import cv2
# Se importa la libreria numpy como np para calcular el valor maximo del histograma
import numpy as np
# Se importa la libreria pyplot para mostrar el hisotgrama de la imagen
import matplotlib.pyplot as plt

# Se carga la imagen 'imagen.jpg' que se encuentra en el mismo directorio del programa
imagen = cv2.imread("imagen.jpg")
# Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
if imagen.shape[0] >550 or imagen.shape[1] >1100:
    imagen = cv2.resize(imagen, None, fx=0.5, fy=0.5)
# La imagen se convierte a escala de grises
grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)

# Se obtienen el alto y el ancho de la imagen
alto, ancho = imagen.shape[0:2]
# Se calcula el histograma de la imagen
histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)

# Se obtiene el valor maximo del histograma
maximo = np.argmax(histograma)

# Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
if maximo == 4 and histograma[4] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
    imagen_equal = cv2.merge((b, g, r))

    # Se muestra la imagen ecualizada
    cv2.imshow("Imagen ecualizada", imagen_equal)

    # Se agrega el texto 'Sobredespuesta' a la imagen
    cv2.putText(imagen, "Sobredespuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
elif maximo == 0 and histograma[0] >0.3:
    # Se dividen los canales con la funcion split
    b, g, r = cv2.split(imagen)
    # 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 genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
imagen_equal = cv2.merge((b, g, r))

# Se muestra la imagen ecualizada
cv2.imshow("Imagen ecualizada", imagen_equal)

# Se agrega el texto 'Subexpuesta' a la imagen
cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Si no se cumple ninguna de las condiciones anteriores...
else:
    # Se agrega el texto 'Buena exposicion' a la imagen
    cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)

# Se agregan las iniciales a la imagen
cv2.putText(imagen, "M. N. I. E", (40, 550), cv2.FONT_SCRIPT_COMPLEX, 4, (0, 0, 0), 2)

# Se muestra la imagen original
cv2.imshow("Imagen", imagen)

# Se agrega el histograma al pyplot
plt.bar(range(len(histograma)), histograma)
# Se muestra el pyplot
plt.show()

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

```

Para el código del algoritmo primero se importan las librerías necesarias para realizarlo, como ***opencv***, ***numpy*** y ***pyplot***. Luego se carga la imagen 'imagen.jpg' que se encuentra en el mismo directorio del programa, si la imagen tiene un alto mayor a 550 o un ancho mayor a 1100 se escala la imagen a la mitad, esta se convierte a escala de grises y se calcula su histograma. Después de calcular el histograma se obtiene el valor máximo de este, si el valor es 4 y el histograma en la posición 4 es mayor a 0.3 significa que está sobreexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Sobreexpuesta". Si el valor máximo es 0 y el histograma en la posición 0 es mayor a 0.3 significa que está subexpuesta, por lo que se va a ecualizar la imagen original, se mostrara la imagen ecualizada y se agregara a la imagen original el texto "Subexpuesta". Si no se cumple ninguna de las condiciones anteriores solo se agregara a la imagen original el texto "Buena exposición" y las iniciales "M. N. I. E". Por último, se muestra la imagen original con el texto, se agrega el histograma al ***pyplot*** y se muestra, al final ***opencv*** espera a que el usuario presione una tecla para terminar el programa.

Imágenes del código

```

main.py - proyecto_integrador_1 - Visual Studio Code
main.py
1 # Se importa la libreria opencv para leer, mostrar y manipular la imagen
2 import cv2
3 # Se importa la libreria numpy como np para calcular el valor maximo del histograma
4 import numpy as np
5 # Se importa la libreria pyplot para mostrar el histograma de la imagen
6 import matplotlib.pyplot as plt
7
8 # Se carga la imagen 'Imagen.jpg' que se encuentra en el mismo directorio del programa
9 imagen = cv2.imread("Imagen.jpg")
10 # Si la imagen en altura es mayor a 550 y en ancho es mayor a 1100 se escala a la mitad de esta
11 if imagen.shape[0] > 550 or imagen.shape[1] > 1100:
12     imagen = cv2.resize(imagen, None, None, fx=0.5, fy=0.5)
13 # La imagen se convierte a escala de grises
14 grises = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
15
16 # Se obtienen el alto y el ancho de la imagen
17 alto, ancho = imagen.shape[0:2]
18 # Se calcula el histograma de la imagen
19 histograma = cv2.calcHist([grises], [0], None, [5], [0, 256]).flatten()/(alto*ancho)
20
21 # Se obtiene el valor maximo del histograma
22 maximo = np.argmax(histograma)
23
24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...

```

```

main.py - Visual Studio Code
main.py
24 # Si el valor maximo es igual a 4 y la posicion 4 del histograma es mayor a 0.3...
25 if maximo == 4 and histograma[4] > 0.3:
26     # Se dividen los canales con la funcion split
27     b, g, r = cv2.split(imagen)
28     # Se ecualiza el canal R
29     r = cv2.equalizeHist(r)
30     # Se ecualiza el canal G
31     g = cv2.equalizeHist(g)
32     # Se ecualiza el canal B
33     b = cv2.equalizeHist(b)
34
35     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
36     imagen_equal = cv2.merge((b, g, r))
37
38     # Se muestra la imagen ecualizada
39     cv2.imshow("Imagen ecualizada", imagen_equal)
40
41     # Se agrega el texto 'Sobreexpuesta' a la imagen
42     cv2.putText(imagen, "Sobreexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
43
44 # Si el valor maximo es igual a 0 y la posicion 0 del histograma es mayor a 0.3...
45 elif maximo == 0 and histograma[0] > 0.3:
46     # Se dividen los canales con la funcion split
47     b, g, r = cv2.split(imagen)

```

```

main.py - Visual Studio Code
main.py
47     b, g, r = cv2.split(imagen)
48     # Se ecualiza el canal R
49     r = cv2.equalizeHist(r)
50     # Se ecualiza el canal G
51     g = cv2.equalizeHist(g)
52     # Se ecualiza el canal B
53     b = cv2.equalizeHist(b)
54
55     # Se genera la imagen ecualizada mezclando todos los canales ecualizados con la funcion merge
56     imagen_equal = cv2.merge((b, g, r))
57
58     # Se muestra la imagen ecualizada
59     cv2.imshow("Imagen ecualizada", imagen_equal)
60
61     # Se agrega el texto 'Subexpuesta' a la imagen
62     cv2.putText(imagen, "Subexpuesta", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
63
64 # Si no se cumple ninguna de las condiciones anteriores...
65 else:
66     # Se agrega el texto 'Buena exposicion' a la imagen
67     cv2.putText(imagen, "Buena exposicion", (40, 40), cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 255), 2)
68
69 # Se agregan las iniciales a la imagen
70 cv2.putText(imagen, "M. N. I. E.", (40, 550), cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 4, (0, 0, 0), 2)
71
72 # Se muestra la imagen original
73 cv2.imshow("Imagen", imagen)
74
75 # Se agrega el histograma al pyplot
76 plt.bar(range(len(histograma)), histograma)
77 # Se muestra el pyplot
78 plt.show()
79
80 #opencv espera a que el usuario presione una tecla para terminar el programa
81 cv2.waitKey(0)
82

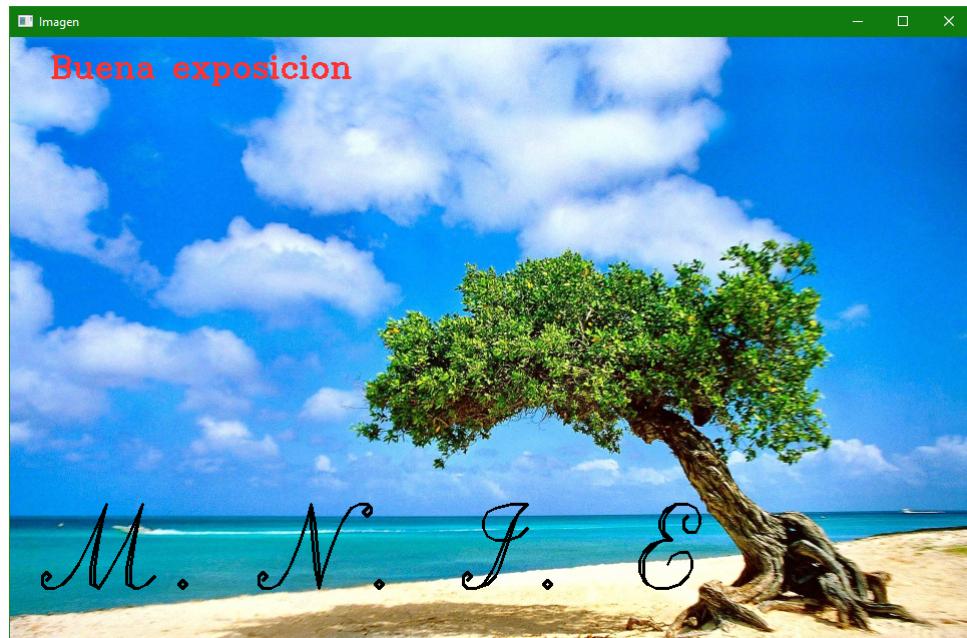
```

```

main.py - Visual Studio Code
main.py
75 # Se agrega el histograma al pyplot
76 plt.bar(range(len(histograma)), histograma)
77 # Se muestra el pyplot
78 plt.show()
79
80 #opencv espera a que el usuario presione una tecla para terminar el programa
81 cv2.waitKey(0)
82

```

Imagen con buena exposición



Histograma de la imagen

