UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ingeniería

Programa de Ingeniería Mecatrónica

LABORATORIO N° 03

"OPENCV Y OPERACIONES DE PUNTO"

DESARROLLO DE GUIA DE LABORATORIO

Procesamiento de Señales e Imágenes

ESTUDIANTE(S) :

Paredes Guevara Darwin Jeferson

Quintana Ramos Enrique Freund

Villanueva Esquivel William Alexis

DOCENTE :

Ing. Asto Rodríguez Emerson

CICLO :

2023 - II

Trujillo, Perú

2023

INDICE

RESUMEN

En el presente laboratorio se trabajó las operaciones de punto sobre imágenes con OpenCv,dentro de lo cual vimos ejemplos de los códigos necesarios para obtener el negativo de unaimagen, la transformación logarítmica, gamma, y arbitraria, es decir, crear nuestra propia transformación, además de la encriptación y vimos como escribir o dibujar sobre imágenes.Por último, vimos cómo se trabajan los histogramas, histogramas acumulados y ecualizaciones de una imagen y cómo nos dan información sobre esta. Con todo ello pudimos aplicar los códigos correspondientes a una imagen para aplicarle una transformación arbitraria, escribir un mensaje encriptados y además usar la ecualización de una imagen para que esta adopte el histograma de otra. Al terminar con los códigos respectivos a cada caso, se logró obtener una imagen transformada por una función creada,otra con un mensaje encriptado y por última una imagen que tenía el histograma de otra.

1. DESARROLLO DEL LABORATORIO

1.1. Resultados de la experiencia

a) Resultado 1

Implementamos una función con la forma mostrada en la figura 1, para modificaruna imagen arbitraria.

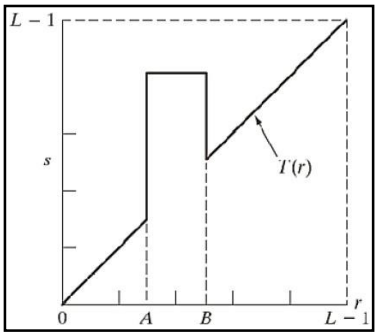

Imagen 1: Forma de la función de transformación. (Gonzalez, 2008)

El resultado es el mostrado en la imagen 2:

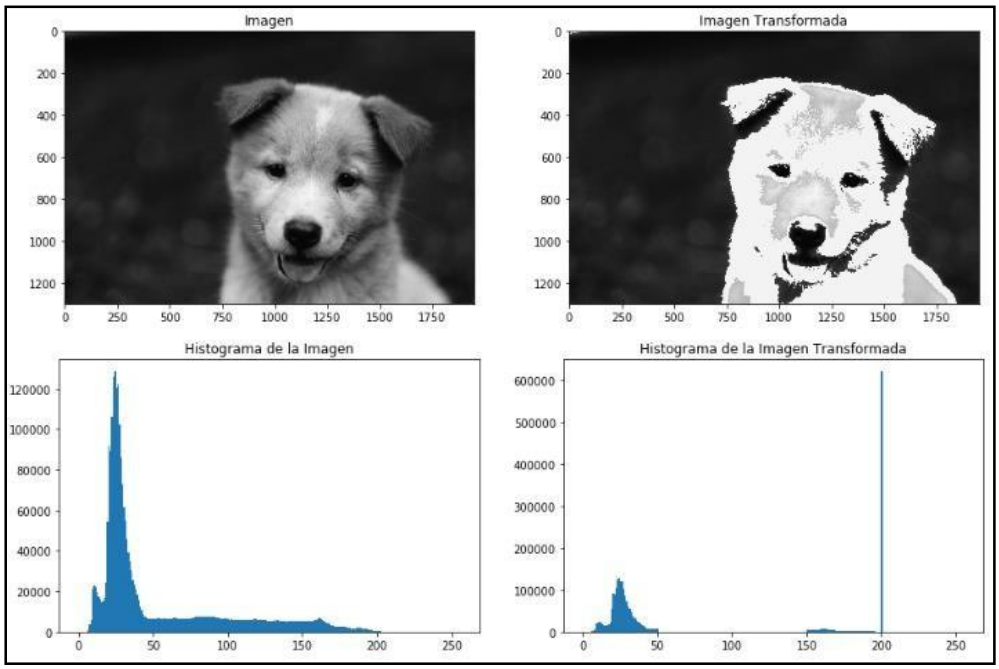

Imagen 2: Histograma de la imagen transformada.

El código trabajado mediante Jupyter Notebook se encuentra en el anexo A1.

b) Resultado 2

Encriptamos un mensaje dentro de una imagen usando descomposición en capas de bits. El resultado es el mostrado en la imagen 3:

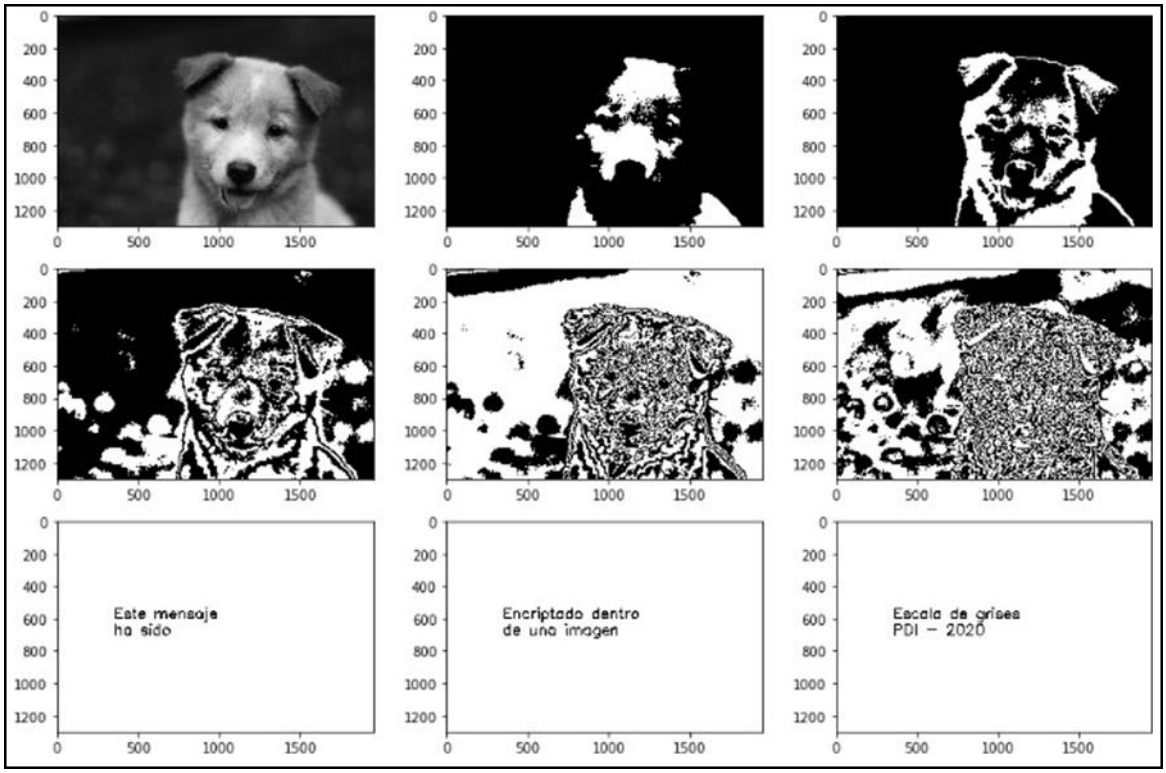

Imagen 3: Mensaje encriptado en las últimas capas de la imagen.

El código trabajado mediante Jupyter Notebook se encuentra en el anexo A2.

c) Resultado 3

Implementa el algoritmo de especificación de histograma de una imagen en escala de grises, cuyo procedimiento se resume en la figura 4.

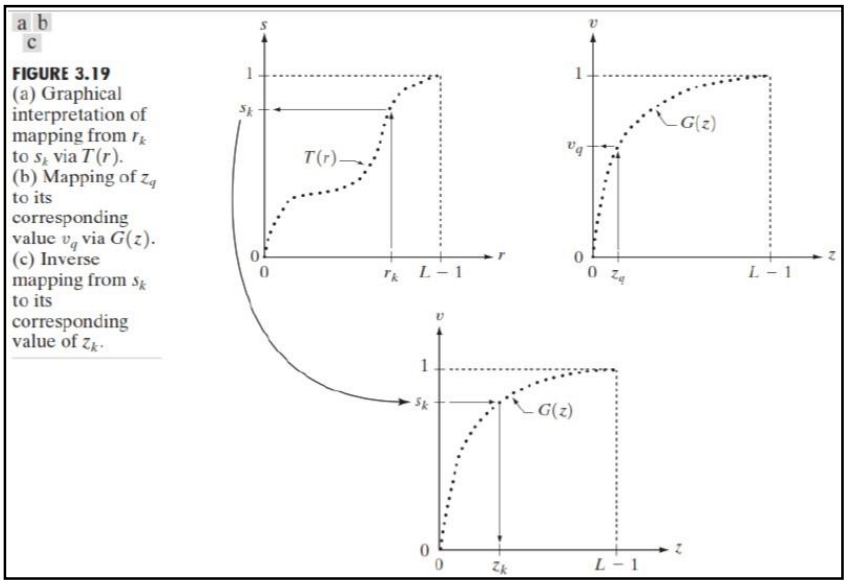

Imagen 4: Algoritmo de especificación de histograma.

El resultado es el mostrado en la imagen 5:

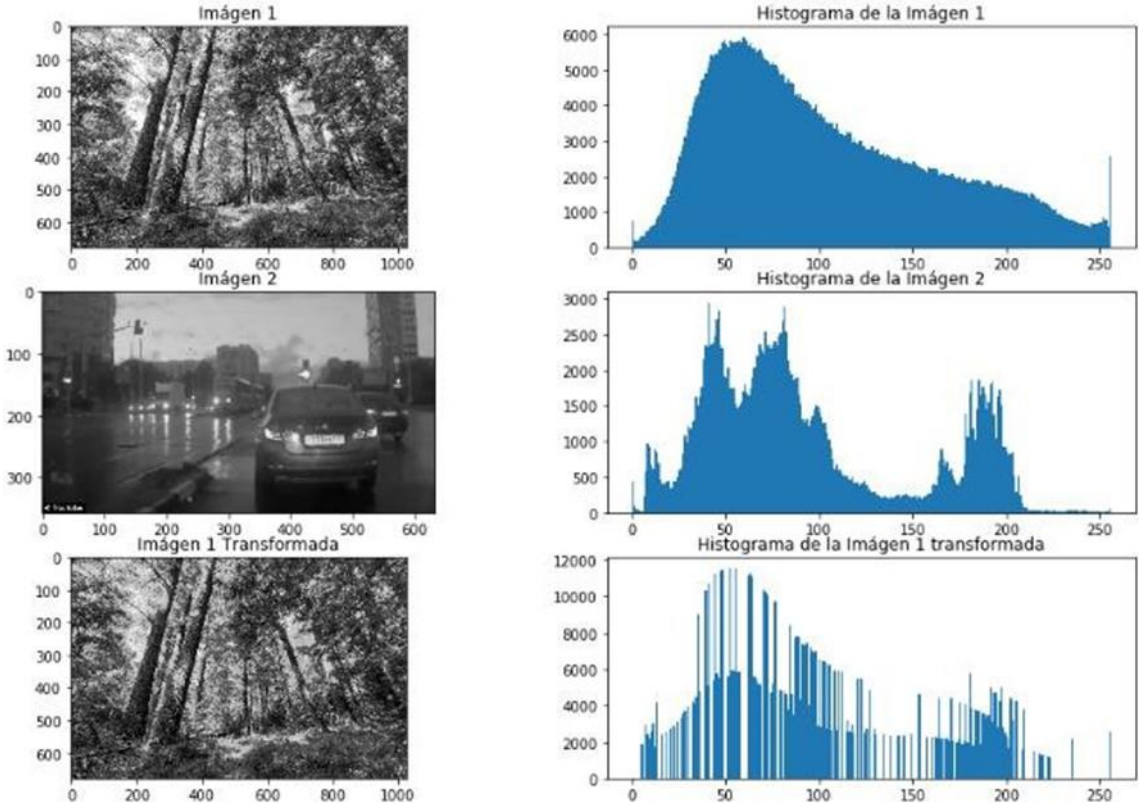

Imagen 5: Histogramas de la imagen 1, imagen 2 e imagen 1 transformada.

El código trabajado mediante Jupyter Notebook se encuentra en el anexo A3.

1.2. Desarrollo de test de comprobación

a) Averigüe como crear un histograma en 2d con OpenCV.

Se usa la función cv2.calcHist(). Para los histogramas de color, necesitamos convertirla imagen de BGR a HSV. Para los histogramas 2D, sus parámetros se modificarán de la siguiente manera:

channels = [0, 1]. porque necesitamos procesar el plano H y S.

bin = [180, 256]. 180 para el plano H y 256 para el plano S.

range = [0, 180, 0, 256]. El valor del tono se encuentra entre 0 y 180 y la saturación seencuentra entre 0 y 256.

El código por implementar sería es el siguiente:

```
import cv2
import numpy as np

img = cv2.imread('ejemplo.jpg')
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hist = cv2.calcHist([hsv], [0, 1], None, [180, 256], [0, 180, 0, 256])
```

b) Defina una función que ingresando un rango de entrada (ej. 100, 120), se cree una función para expandir ese rango, en todo el rango dinámico de 0 a 255.

```
import cv2
import numpy as np

def expandir_rango(rango_entrada):
    minimo, maximo = rango_entrada
    rango_deseado = (0, 255)
    return lambda valor: int(np.interp(valor, rango_entrada, rango_deseado))

mi_funcion = expandir_rango((100, 120))
print(mi_funcion(110))  # Output: 127
```

c) Definir una función para aumentar o reducir el brillo de una imagen, según valor especificado.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# definimos la funcion
def funcion (F, range_in=(0, 255), range_out=(0, 255), gamma=1):
    F = np.clip(F, range_in[0], range_in[1])
    G = ((F - range_in[0]) / (range_in[1] - range_in[0])) ** gamma
    G = G * (range_out[1] - range_out[0]) + range_out[0]
    return np.uint8(G)

# ruta de la imagen
imgSRC = '/IMAGENES_VIRTUAL/practicas/Imagenes/ajedrez.jpg'
img = cv2.imread(imgSRC, cv2.IMREAD_GRAYSCALE)
# brillo de la imagen
brillo_up = funcion(img, gamma=2.0) # Aumentar brillo
brillo_down = funcion(img, gamma=0.3) # Disminuir brillo
# figuras de las imagenes
plt.figure(figsize=(12, 6))
plt.subplot(131)
plt.imshow(img, cmap="gray")
plt.title("Imagen Original")
plt.axis('off')
plt.subplot(132)
plt.imshow(brillo_up, cmap="gray")
plt.title("Aumentar Brillo")
plt.axis('off')
plt.subplot(133)
plt.imshow(brillo_down, cmap="gray")
plt.title("Disminuir Brillo")
plt.axis('off')
plt.show()
```

2. RECOMENDACIONES Y CONCLUSIONES

2.1. Recomendaciones

a) Al importar la imagen, primero hay que asegurarse si la carpeta de su ubicaciónestá dentro de la carpeta de trabajo del jupyter notebook que estamos trabajando,así como asegurarse que el nombre y el formato de la imagen sean los correctos.

b) Al momento de crear una función de transformación, hay que plotearla para observar en la gráfica el valor en el que se transforma cada bit de color. Además,es mejor aplicar la transformación con "LTU" que con bucles "for". Por último,imprimir las imágenes antes y luego de la transformación para notar que la función se efectúe correctamente.

c) Cuando se encripta las capas de la imagen blanca en la original, es preferible quese haya escrito un mensaje en las capas de la imagen blanca antes de encriptarlasen la imagen original porque en caso contario cuando se imprimen sale una capade color negro.

d) Los ciclos "for" para la búsqueda en cada equalización debe hacerse porseparado debido a que Jupyter Notebook no responde cuando se hace más de 2 ciclos "for" seguidos.

2.2. Conclusiones

a) Las operaciones de procesamiento de imágenes punto por punto se pueden utilizar para transformar imágenes según una determinada función, así como codificar mensajes y obtener histogramas de otras imágenes.
b) Los histogramas de imágenes nos proporcionan mucha información valiosa sobre el color de la imagen, como base para crear funciones de transformación.
c) Cuando trasladamos el histograma de una imagen a otra, esto no se consigue del todo; Lo que sucede es que el gráfico se distorsiona en una curva que generalmente se parece a la curva que queremos.
d) Surge el problema de que hay colores en la ecualización de nuestra imagen que noencontramos en la ecualización de la imagen de la cual queremos adoptar su histograma, y esto hace que cuando hacemos la función de transformación para adoptar el histograma deseado, muchos valores se acumulan en 0, formándose unacurva con un enorme pico en 0. Para solucionar dicho error, a los colores que no seencuentren en la otra ecualización se los deja en su mismo valor. Efectuando dicha solución, se obtiene una curva más continua y por ende un histograma más similaral que queremos alcanzar.

3. REFERENCIAS BIBLIOGRÁFICAS

- ✓ Gonzalez, R. &. (2008). *Digital image processing.* New Jersey: Parson.
- ✓ Matplotlib.org (s.f.). (26 de Julio de 2020). *Matplotlib Documentation*. Obtenido dehttps://matplotlib.org/
- ✓ NumPy.org. (s.f.). (26 de Julio de 2020). *NumPy Documentation*. Obtenido de NumpyDocumentation.
- ✓ OpenCV.org. (s.f.). (26 de Julio de 2020). *OpenCV python tutorials*.
- ✓ Unipython. (05 de Abril de 2018). *HISTOGRAMAS 2D*. Obtenido de https://unipython.com/histogramas-2d/

ANEXOS

ANEXO A1

```
# Importamos librerías:
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Implementamos una función arbitraria T(r):
T = np.zeros((256))
A = 50
B = 150
H = 200

for i in range(256):
    if i <= A:
        T[i] = i
    elif A < i < B:
        T[i] = H
    else:
        T[i] = i

plt.plot(T)
plt.grid(True)

# Importamos la imagen:
imagen = cv2.imread("IMAGENES/puppy.jpg", 0)

# Aplicamos la transformación T(r) a la imagen:
s = cv2.LUT(imagen, T)

# Imprimimos:
```

```
plt.figure(figsize=(15,10))
plt.subplot(221)
plt.imshow(imagen, cmap="gray")
plt.title("Imagen")
plt.subplot(222)
plt.imshow(s, cmap="gray")
plt.title("Imagen Transformada")
plt.subplot(223)
plt.hist(imagen.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen")
plt.subplot(224)
plt.hist(s.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen Transformada")
plt.show()
```

ANEXO A2

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Importamos la imagen:
imagen = cv2.imread("IMAGENES/puppy.jpg", 0)

# Creamos imagen blanca con las dimensiones de nuestra imagen
imagen_w = np.ones((imagen.shape), dtype=np.uint8) * 255

# Descomponemos la imagen en 8 capas:
imgg_b7 = np.bitwise_and(imagen, 128)
imgg_b6 = np.bitwise_and(imagen, 64)
imgg_b5 = np.bitwise_and(imagen, 32)
imgg_b4 = np.bitwise_and(imagen, 16)
imgg_b3 = np.bitwise_and(imagen, 8)
imgg_b2 = np.bitwise_and(imagen, 4)
imgg_b1 = np.bitwise_and(imagen, 2)
imgg_b0 = np.bitwise_and(imagen, 1)

# Descomponemos la imagen blanca en 8 capas:
imgw_b7 = np.bitwise_and(imagen_w, 128)
imgw_b6 = np.bitwise_and(imagen_w, 64)
imgw_b5 = np.bitwise_and(imagen_w, 32)
imgw_b4 = np.bitwise_and(imagen_w, 16)
imgw_b3 = np.bitwise_and(imagen_w, 8)
imgw_b2 = np.bitwise_and(imagen_w, 4)
imgw_b1 = np.bitwise_and(imagen_w, 2)
imgw_b0 = np.bitwise_and(imagen_w, 1)

# Escribimos el texto en las primeras 3 capas de la imagen blanca:
cv2.putText(imgw_b2, "Este mensaje", (350,600),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)
cv2.putText(imgw_b2, "ha sido", (350,700),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)
cv2.putText(imgw_b1, "Encriptado dentro", (350,600),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)
cv2.putText(imgw_b1, "de una imagen", (350,700),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)
cv2.putText(imgw_b0, "Escala de grises", (350,600),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)
cv2.putText(imgw_b0, "PDI - 2020", (350,700),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0,255,255), 8)

# Intercambiamos las primeras 3 capas de la imagen con las de la imagen blanca:
imgg_b2 = imgw_b2
imgg_b1 = imgw_b1
imgg_b0 = imgw_b0

# Imprimimos todas las capas de la imagen
plt.figure(figsize=(15,10))
plt.subplot(331); plt.imshow(imagen, cmap="gray")
plt.subplot(332); plt.imshow(imgg_b7, cmap="gray")
plt.subplot(333); plt.imshow(imgg_b6, cmap="gray")
plt.subplot(334); plt.imshow(imgg_b5, cmap="gray")
plt.subplot(335); plt.imshow(imgg_b4, cmap="gray")
plt.subplot(336); plt.imshow(imgg_b3, cmap="gray")
plt.subplot(337); plt.imshow(imgw_b2, cmap="gray")
plt.subplot(338); plt.imshow(imgg_b1, cmap="gray")
plt.subplot(339); plt.imshow(imgg_b0, cmap="gray")
plt.title("Imagen Encriptada")
plt.imshow(imgg_b7 + imgg_b6 + imgg_b5 + imgg_b4 + imgg_b3 + imgg_b2 +
imgg_b1 + imgg_b0, cmap="gray")
plt.show()
```

ANEXO A3

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Imágenes en escala de grises:
imagen_1_gray = cv2.imread("IMAGENES/bosque.jpg", 0)
imagen_2_gray = cv2.imread("IMAGENES/car_plate.jpg", 0)

# Obtengo las equalizaciones de las imágenes:
equ_imagen_1_gray = cv2.equalizeHist(imagen_1_gray)
equ_imagen_2_gray = cv2.equalizeHist(imagen_2_gray)

# Imprimimos las imágenes, sus histogramas y los histogramas acumulados de
las imágenes y sus equalizaciones:
plt.figure(figsize=(15,15))

plt.subplot(421)
plt.imshow(imagen_1_gray, cmap="gray")
plt.title("Imagen 1 GRIS")

plt.subplot(422)
plt.imshow(imagen_2_gray, cmap="gray")
plt.title("Imagen 2 GRIS")

plt.subplot(423)
plt.hist(imagen_1_gray.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen 1")

plt.subplot(424)
plt.hist(imagen_2_gray.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen 2")

plt.subplot(425)
```

```
plt.hist(imagen_1_gray.ravel(), 256, [0,256], cumulative=True)
plt.title("Histograma acumulado de la Imagen 1")

plt.subplot(426)
plt.hist(imagen_2_gray.ravel(), 256, [0,256], cumulative=True)
plt.title("Histograma acumulado de la Imagen 2")

plt.subplot(427)
plt.hist(equ_imagen_1_gray.ravel(), 256, [0,256], cumulative=True)
plt.title("Histograma acumulado de la Imagen 1 Equalizada")

plt.subplot(428)
plt.hist(equ_imagen_2_gray.ravel(), 256, [0,256], cumulative=True)
plt.title("Histograma acumulado de la Imagen 2 Equalizada")

plt.show()

# Obtenemos la función que relaciona los colores de la Imagen 1 y su
Equalización:
F = list(range(256))

# Creamos el vector que tendrá los valores del eje y:
F1 = np.zeros((256))

# Buscamos cada color de la Imagen 1 con su respectivo de Equalización 1 y los
guardamos en F1:
for i in range(imagen_1_gray.shape[0]):
    for j in range(imagen_1_gray.shape[1]):
        F1[imagen_1_gray[i,j]] = equ_imagen_1_gray[i,j]

# Graficamos la transformación de los colores de la Imagen 1 a los de
Equalización 1:
plt.plot(F,F1)
plt.show()
```

```
# Obtenemos la función que relaciona los colores de la Imagen 2 y su
Equalización:
# Creamos el vector que tendrá los valores del eje y:
F2 = np.zeros((256))

# Buscamos cada color de la Imagen 2 con su respectivo de Equalización 2 y los
guardamos en F2:
for i in range(imagen_2_gray.shape[0]):
    for j in range(imagen_2_gray.shape[1]):
        F2[imagen_2_gray[i,j]] = equ_imagen_2_gray[i,j]

# Graficamos la transformación de los colores de la Imagen 2 a los de
Equalización 2:
plt.plot(F,F2)
plt.show()

# Obtenemos la función que relaciona los colores de la Imagen 1 y la Imagen 2:
# Creamos el vector que tendrá los valores del eje y:
F3 = np.zeros((256))

# Rellenamos F3 con valores 0-256 para mantener valores de F1 que no tengan
su par en F2:
for i in range(F1.shape[0]):
    F3[i] = i
    for i in range(F1.shape[0]):
        for j in range(F2.shape[0]):
            if F2[j] == F1[i]:
                F3[i] = j

# Graficamos la transformación de los colores de la Imagen 1 a la Imagen 2:
plt.plot(F,F3)
plt.show()
```

```
# Aplicamos la transformación F3:
imagen_gray_T = cv2.LUT(imagen_1_gray, F3)

# Imprimimos la Imagen 1, Imagen 2 e Imagen 1 transformada con sus
histogramas:
plt.figure(figsize=(15,10))

plt.subplot(321)
plt.imshow(imagen_1_gray, cmap="gray")
plt.title("Imagen 1")
plt.subplot(322)
plt.hist(imagen_1_gray.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen 1")

plt.subplot(323)
plt.imshow(imagen_2_gray, cmap="gray")
plt.title("Imagen 2")
plt.subplot(324)
plt.hist(imagen_2_gray.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen 2")

plt.subplot(325)
plt.imshow(imagen_gray_T, cmap="gray")
plt.title("Imagen 1 Transformada")
plt.subplot(326)
plt.hist(imagen_gray_T.ravel(), 256, [0,256])
plt.title("Histograma de la Imagen 1 transformada")

plt.show()
```