In [53]:

```
# 1. Seleccionar 2 imagenes que requieran una mejora de contraste
# 2. Aplica 2 tecnicas que permitan mejorar el contraste
# una con dev propio y otra con OpenCV
# 3. Aplica las dos tecnicas a las 2 imagenes
# 4. Describe como seleccionaste la mejor tecnica para el ajuste de intensidad
# 5. Describe caracteristicas de las tecnicas seleccionadas
# 6. Compara y concluye
```

In [54]:

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import random
```

Ajuste de intensidad - Denis Zelaya

Definir imagenes a utilizar

In [126]:

```
img01 = cv2.imread("r1.png", cv2.IMREAD_GRAYSCALE)
img02 = cv2.imread("p2.jpg", 1)
img04 = cv2.imread("fondo-ojo.jpg", 1)
```

Crear funciones reutilizables

In [56]:

```
# Funcion para calcular el histograma de los 3 canales de una imagen
# https://www.geeksforgeeks.org/python-opencv-cv2-calchist-method/
def plot_rgb_histogramTodo(image):
    # Dividir la imagen en canales RGB
    b, g, r = cv2.split(image)

    # Calcular histograma de cada canal
    hist_b = cv2.calcHist([b], [0], None, [256], [0, 256])
    hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
    hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])

    # Agregar plots
    plt.plot(hist_b, color='blue', label='Azul')
    plt.plot(hist_g, color='green', label='Verde')
    plt.plot(hist_r, color='red', label='Rojo')


    # Titulos de referencia
    plt.title('Histogramas')
    plt.xlabel('Intensidad')
    plt.ylabel('Frecuencia')
    plt.legend()

    # Mostrar
    plt.show()
```

In [57]:

```
# Funcion para abstraer el codigo para mostrar una imagen
def imshow(title="Image", image = None, size = 10):

    # Obtener el ancho y alto de la imagen
    w, h = image.shape[0], image.shape[1]

    aspect_ratio = w/h
    plt.figure(figsize = (size * aspect_ratio, size))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.show()
```

In [101]:

```
# Mostrar varias imagenes en la misma fila
def show_images(images_list: list):

    # Ancho y Alto
    plt.figure(figsize=(20, 5))

    # Iterar el arreglo y hacer destructuring de image y title
    for i, (image, title) in enumerate(images_list):

        # Definir subplot, pasandole numero de filas, columnas, indice
        plt.subplot(1, 3, i + 1)

        # Imagen a mostrar
        imshow(str(title), image)
        plt.imshow(image, cmap="gray")

        # Agregar el titulo
        plt.title(title)

        # Ocultar ejes
        plt.axis("off")

    plt.show()
```

In [103]:

```
# Mostrar 3 imagenes en la misma fila
def show_three_images(img1=None, title1="", img2=None, title2="", img3=None, title3=""):
    plt.figure(figsize=(15, 5))

    # Imagen 1
    plt.subplot(1, 3, 1)
    plt.imshow(img1, cmap='gray')
    plt.title(title1)
    plt.axis('off')

    # Imagen 2
    plt.subplot(1, 3, 2)
    plt.imshow(img2, cmap='gray')
    plt.title(title2)
    plt.axis('off')

    # Imagen 3
    plt.subplot(1, 3, 3)
    plt.imshow(img3, cmap='gray')
    plt.title(title3)
    plt.axis('off')

    plt.show()
```

In [82]:

```
# Invertir una imagen util para mejorar la interpretación de un ultrasonido
def invert_color(image = None, title = ""):

    # Obtener ancho y alto
    h, w = image.shape[0], image.shape[1]

    # Generar un arreglo de 0, con las dimensiones de la imagen
    imgInvert = np.zeros((h, w), np.uint8)

    for j in range(h):
        for i in range(w):
            newValue = 0

            # Obtener el valor actual del pixel en el indice [j,i]
            valor_actual = image[j, i]

            # Ajustar los valores para obtener el resultado deseado
            if (valor_actual > 200):
                newValue = int((255 - image[j, i]) - 100)

            if (valor_actual < 30):
                newValue = int((255 - image[j, i]) + 50)
            else:
                newValue = int((255 - image[j, i]))

            # En la posicion del pixel asignamos el valor generado
            # La funcion de numpy.clip nos permite limitar los valores con el rango defin
            imgInvert[j, i] = np.clip(newValue, 0, 255)



    # Mostrar el resultado
    # imshow(title, imgInvert)
    return imgInvert
```

In [115]:

```
# Mejorar el contraste de una imagen, la he parametrizado para ajustar segun la necesidad
def mejora_contraste(image = None, azul = 0, verde = 10, rojo = 0):
    # Definir constantes a sumar a cada canal
    c_azul = azul
    c_verde = verde
    c_rojo = rojo

    # Obtener los canales de una imagen cv2.split(img) -> Retorna las 3 capas BGR
    canal_azul, canal_verde, canal_rojo = cv2.split(image)

    # Crear constantes para guardar la suma de nuestros parametros + el valor del actual
    c_azul_const = np.clip(canal_azul.astype(int) + c_azul, 0, 255).astype(np.uint8)
    c_verde_const = np.clip(canal_verde.astype(int) + c_verde, 0, 255).astype(np.uint8)
    c_rojo_const = np.clip(canal_rojo.astype(int) + c_rojo, 0, 255).astype(np.uint8)

    # Fusionar nuestros 3 canales procesados
    imagen_final = cv2.merge((c_azul_const, c_verde_const, c_rojo_const))

    # Mostrar imagen
    # plt.imshow(cv2.cvtColor(imagen_final, cv2.COLOR_BGR2RGB))
    # plt.title('imagen')
    # plt.show()

    return imagen_final
```

In [193]:

```
# Restar intensidad a una imagen
def restar_intensidad(image = None, valor = 40):

    # Generar matriz de 0 y la multiplicamos * 40
    # Cada indice de nuestro arreglo tendra el valor de
    M = np.ones(image.shape, dtype = "uint8") * valor

    # Generar nueva imagen, realizando una funcion para cambiar el color de nuestro pixel
    # Al pixel actual le restamos nuestro valor parametrizado
    img_resta = np.clip(img02.astype(np.int16) - M, 0, 255).astype(np.uint8)

    # Mostrar imagen
    plt.imshow(cv2.cvtColor(img_resta, cv2.COLOR_BGR2RGB))
    plt.title('resta')
    plt.show()

    # Retornar la imagen obtenida
    return img_resta
```

In [217]:

```
def ecualizar_imagen(imagen = None):
    b, g, r = cv2.split(imagen)

    # Ecualizar el canal de valor (V)
    equ_b = cv2.equalizeHist(b)
    equ_g = cv2.equalizeHist(g)
    equ_r = cv2.equalizeHist(r)

    # Fusionar los canales nuevamente en la imagen HSV
    equ_image = cv2.merge([equ_b,equ_g,equ_r])
    imshow("ecualizada",equ_g)

    # hist = cv2.calcHist(equ_image, [0], None, [256], [0, 256])
    # plt.plot(hist)

    return equ_image
```

In [63]:

```
def plot_rgb_histograms(image):
    # Dividir la imagen en canales RGB
    b, g, r = cv2.split(image)

    # Calcular los histogramas de cada canal
    hist_b = cv2.calcHist([b], [0], None, [256], [0, 256])
    hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
    hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])

    # Graficar los histogramas
    plt.figure(figsize=(10, 5))

    plt.subplot(3, 1, 1)
    plt.plot(hist_b, color='blue')
    plt.title('Histograma Canal Azul')

    plt.subplot(3, 1, 2)
    plt.plot(hist_g, color='green')
    plt.title('Histograma Canal Verde')

    plt.subplot(3, 1, 3)
    plt.plot(hist_r, color='red')
    plt.title('Histograma Canal Rojo')

    plt.tight_layout()
    plt.show()
```

In [64]:

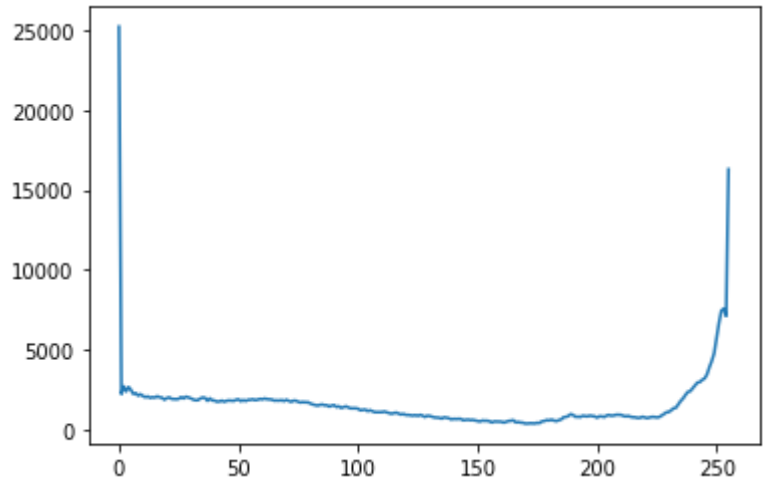
```
# Mostrar todos los canales
def plot_rgb_histogramTodo(image):
    # Dividir la imagen en canales RGB
    b, g, r = cv2.split(image)

    # Calcular histograma de cada canal
    hist_b = cv2.calcHist([b], [0], None, [256], [0, 256])
    hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
    hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])

    plt.plot(hist_b, color='blue', label='Azul')
    plt.plot(hist_g, color='green', label='Verde')
    plt.plot(hist_r, color='red', label='Rojo')
    plt.title('Histogramas')
    plt.xlabel('Intensidad')
    plt.ylabel('Frecuencia')
    plt.legend()
    plt.show()
```

In [98]:

```
# Correcion gama
# https://docs.opencv.org/3.4/d2/de8/group__core__array.html
def contraste_openCV(image = None, value = 1.0):
    img_corrected = cv2.pow(image, value)
    ## imshow(title, img_corrected)
    return img_corrected
```

In [ ]:

```
# Correccion Gama
#https://es.wikipedia.org/wiki/Correcci%C3%B3n_gamma
def generar_histograma(image = None):
    hist_full = cv2.calcHist([image], [0], None, [256], [0,256])
    plt.plot(hist_full)
    plt.show()
```



Out[66]:

```
array([[25219.],
       [ 2233.],
       [ 2711.],
       [ 2397.],
       [ 2674.],
       [ 2490.],
       [ 2238.],
       [ 2288.],
       [ 2134.],
       [ 2221.],
       [ 2111.],
       [ 2029.],
       [ 2087.],
       [ 1991.],
       [ 2035.],
       [ 2009.],
       [ 2096.],
       [ 2021.],
       [ 2009.],
       [ 1878.],
       [ 1975.],
       [ 2026.],
       [ 1930.],
       [ 1919.],
       [ 1911.],
       [ 1932.],
       [ 2041.],
       [ 1956.],
       [ 2070.],
       [ 2023.],
       [ 1969.],
       [ 1892.],
       [ 1846.],
       [ 1852.],
       [ 1950.],
       [ 2011.],
       [ 1995.],
       [ 1807.],
       [ 1962.],
       [ 1835.],
       [ 1824.],
       [ 1755.],
       [ 1768.],
       [ 1824.],
       [ 1751.],
       [ 1807.],
       [ 1833.],
       [ 1821.],
       [ 1804.],
       [ 1873.],
       [ 1901.],
       [ 1783.],
       [ 1858.],
       [ 1794.],
       [ 1869.],
       [ 1871.],
       [ 1902.],
       [ 1851.],
       [ 1880.],
       [ 1927.],
       [ 1901.],
```

```
       [ 1954.],
       [ 1909.],
       [ 1903.],
       [ 1882.],
       [ 1822.],
       [ 1835.],
       [ 1821.],
       [ 1854.],
       [ 1790.],
       [ 1874.],
       [ 1812.],
       [ 1743.],
       [ 1821.],
       [ 1815.],
       [ 1747.],
       [ 1704.],
       [ 1734.],
       [ 1711.],
       [ 1722.],
       [ 1644.],
       [ 1585.],
       [ 1544.],
       [ 1508.],
       [ 1552.],
       [ 1579.],
       [ 1529.],
       [ 1533.],
       [ 1446.],
       [ 1512.],
       [ 1538.],
       [ 1420.],
       [ 1467.],
       [ 1360.],
       [ 1419.],
       [ 1460.],
       [ 1390.],
       [ 1338.],
       [ 1337.],
       [ 1343.],
       [ 1316.],
       [ 1217.],
       [ 1244.],
       [ 1261.],
       [ 1164.],
       [ 1220.],
       [ 1161.],
       [ 1101.],
       [ 1100.],
       [ 1095.],
       [ 1101.],
       [ 1134.],
       [ 1085.],
       [ 1049.],
       [  998.],
       [ 1042.],
       [ 1067.],
       [ 1014.],
       [  951.],
       [  986.],
       [  924.],
       [  925.],
```

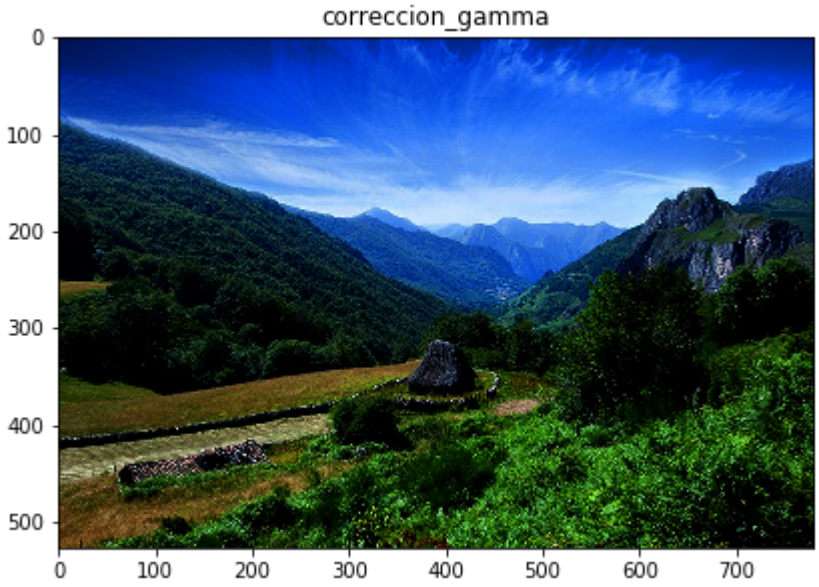
```
       [  906.],
       [  884.],
       [  913.],
       [  871.],
       [  926.],
       [  882.],
       [  797.],
       [  874.],
       [  858.],
       [  811.],
       [  768.],
       [  754.],
       [  747.],
       [  696.],
       [  774.],
       [  747.],
       [  722.],
       [  646.],
       [  681.],
       [  666.],
       [  680.],
       [  678.],
       [  595.],
       [  615.],
       [  632.],
       [  602.],
       [  615.],
       [  584.],
       [  565.],
       [  514.],
       [  575.],
       [  559.],
       [  563.],
       [  531.],
       [  454.],
       [  515.],
       [  508.],
       [  500.],
       [  499.],
       [  448.],
       [  498.],
       [  546.],
       [  563.],
       [  595.],
       [  458.],
       [  482.],
       [  450.],
       [  437.],
       [  387.],
       [  388.],
       [  406.],
       [  399.],
       [  392.],
       [  431.],
       [  416.],
       [  527.],
       [  546.],
       [  604.],
       [  581.],
       [  629.],
       [  571.],
```

```
       [  540.],
       [  599.],
       [  641.],
       [  781.],
       [  802.],
       [  856.],
       [  977.],
       [  919.],
       [  801.],
       [  819.],
       [  792.],
       [  872.],
       [  853.],
       [  833.],
       [  884.],
       [  853.],
       [  838.],
       [  751.],
       [  841.],
       [  837.],
       [  789.],
       [  888.],
       [  922.],
       [  875.],
       [  892.],
       [  937.],
       [  933.],
       [  930.],
       [  868.],
       [  831.],
       [  863.],
       [  810.],
       [  783.],
       [  777.],
       [  737.],
       [  746.],
       [  808.],
       [  771.],
       [  727.],
       [  769.],
       [  804.],
       [  796.],
       [  768.],
       [  791.],
       [  879.],
       [  981.],
       [ 1091.],
       [ 1107.],
       [ 1213.],
       [ 1339.],
       [ 1371.],
       [ 1577.],
       [ 1803.],
       [ 1956.],
       [ 2178.],
       [ 2368.],
       [ 2410.],
       [ 2594.],
       [ 2780.],
       [ 2951.],
       [ 2980.],
```

```
        [ 3123.],
        [ 3203.],
        [ 3462.],
        [ 3908.],
        [ 4298.],
        [ 4785.],
        [ 5755.],
        [ 6671.],
        [ 7442.],
        [ 7574.],
        [ 7118.],
        [16296.]], dtype=float32)
```

In [67]:

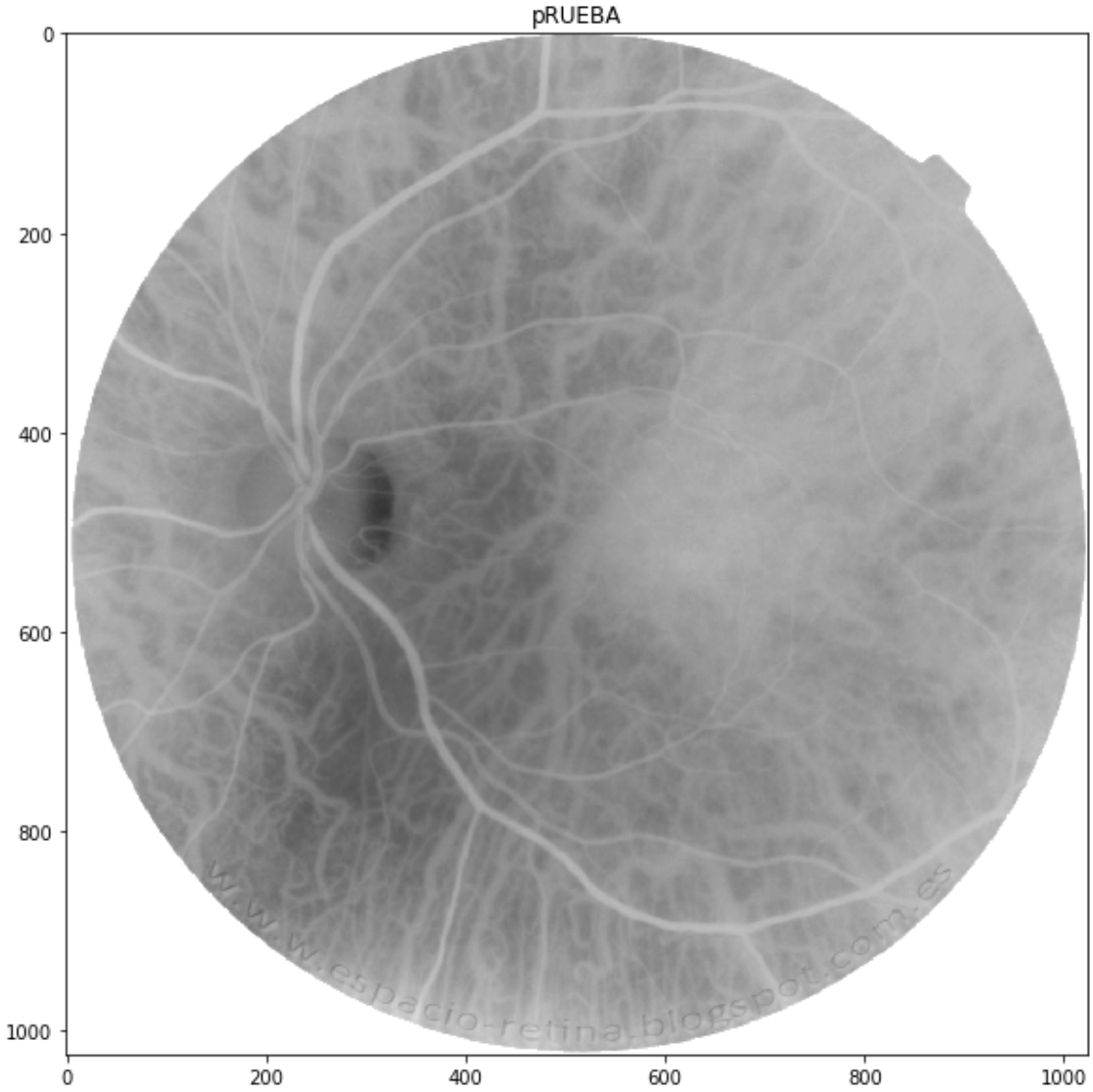
```
# Correccion Gama
def correccion_gamma(gamma = 0.0, imagen = None):
    nueva = np.empty((1, 256), np.uint8)

    for i in range(256):
        nueva[0,i] = np.clip(pow(i / 255.0, gamma) * 255.0, 0, 255)

    res = cv2.LUT(imagen, nueva)
    return res

imshow("correccion_gamma", correccion_gamma(2.0, img02))
```



Procesar imagen 1

In [94]:

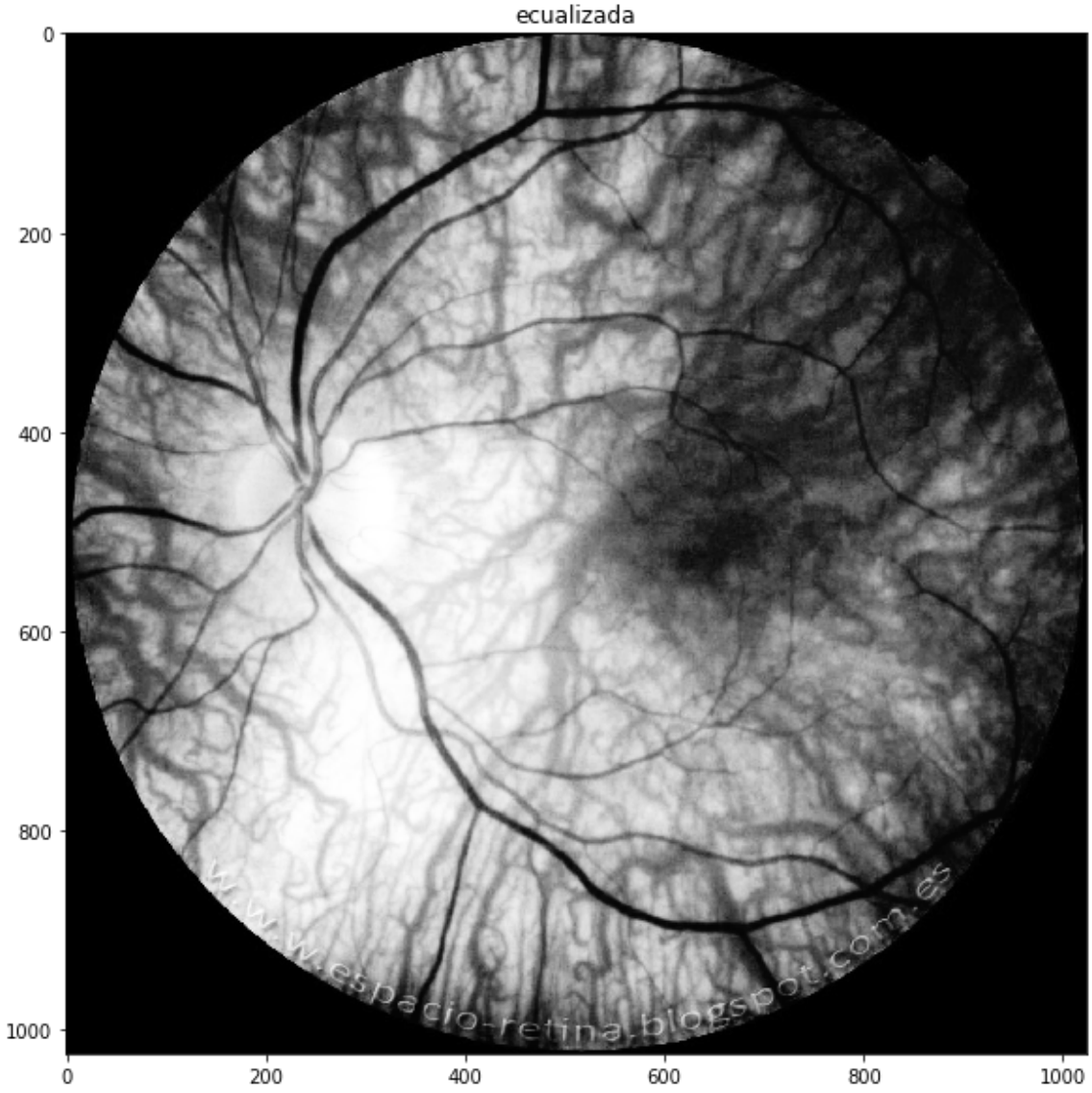
```
# imshow("img niebla", img02)
# imshow("pRUEBA", img04)
imshow("pRUEBA", invert_color(img04, ""))

# imshow("IMG REFERENCIA", img01)
```



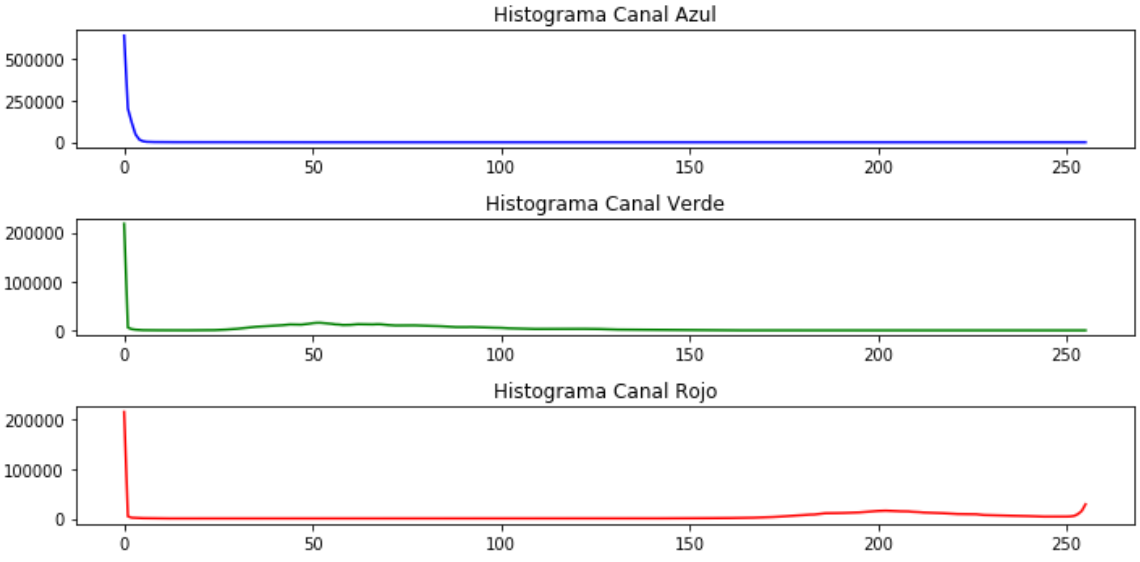
Ajustes de intensidad y contraste muestra 1

In [221]:

```
# IMG 01
img4_open_cv = contraste_openCV(img04, 2)
img4_propio = mejora_contraste(img04, azul = -80, verde = 80, rojo = 0)

show_three_images(img04, "Imagen Original", img4_propio, "Metodo propio", img4_open_cv, "

# Otras pruebas
img04Equ = ecualizar_imagen(img04)
imshow("Ecualizar imagen", img04Equ)

# Mostrar histogramas
plot_rgb_histograms(img04)
plot_rgb_histogramTodo(img04)
plot_rgb_histogramTodo(img4_propio)
```

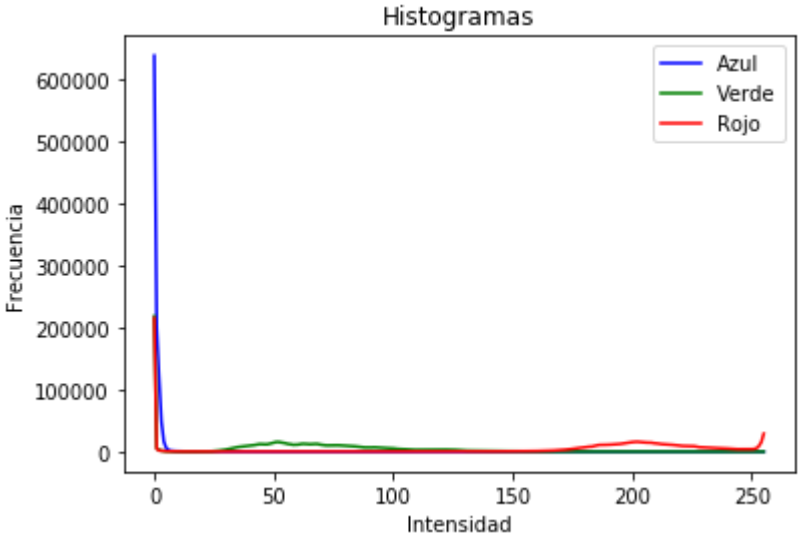

ecualizada
0
200
400
600
800
1000
0
200
400
600
800
1000
www.espacio-retina.blogspot.com.es

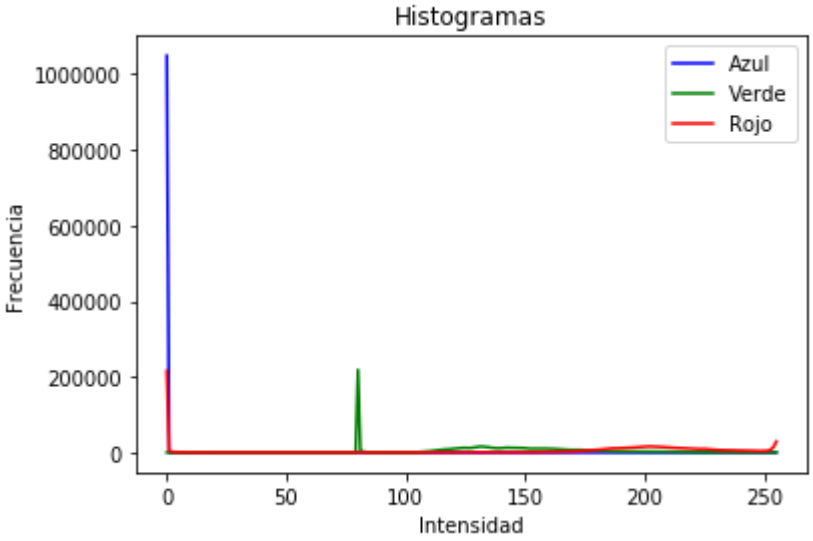

Ecualizar imagen
0
200
400
600
800
1000
www.espacio-retina.blogspot.com.es


Histograma Canal Azul
500000
250000
0
0
50
100
150
200
250

Histograma Canal Verde
200000
100000
0
0
50
100
150
200
250

Histograma Canal Rojo
200000
100000
0
0
50
100
150
200
250

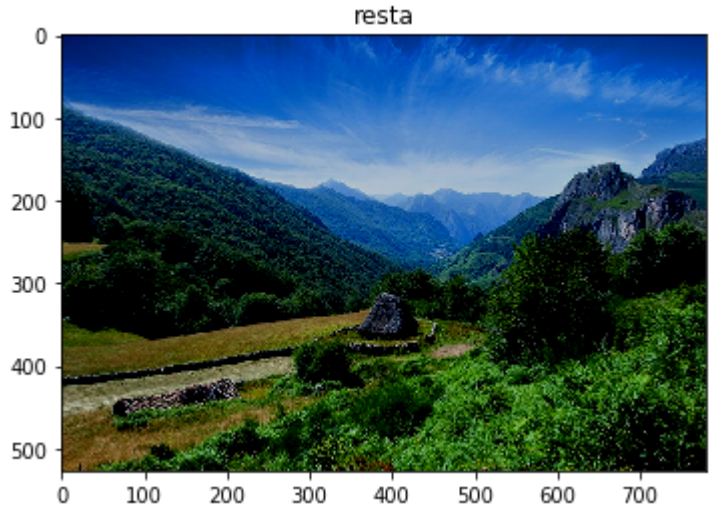

Histogramas
Azul
Verde
Rojo
Frecuencia
Intensidad
0
100000
200000
300000
400000
500000
600000
0
50
100
150
200
250

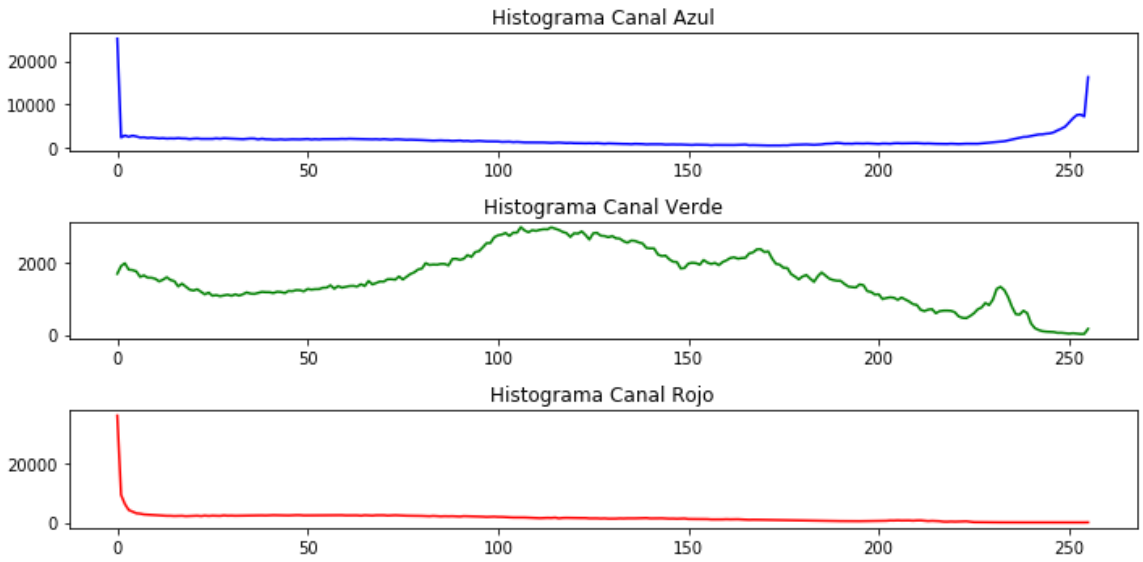

Histogramas
Azul
Verde
Rojo
Frecuencia
Intensidad
0
200000
400000
600000
800000
1000000
0
50
100
150
200
250

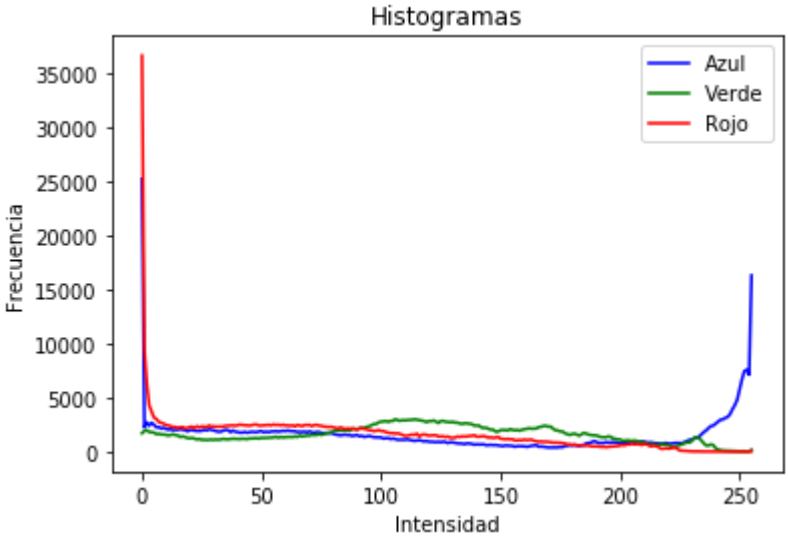
Ajustes en contraste open cv y metodo propio referencia 2

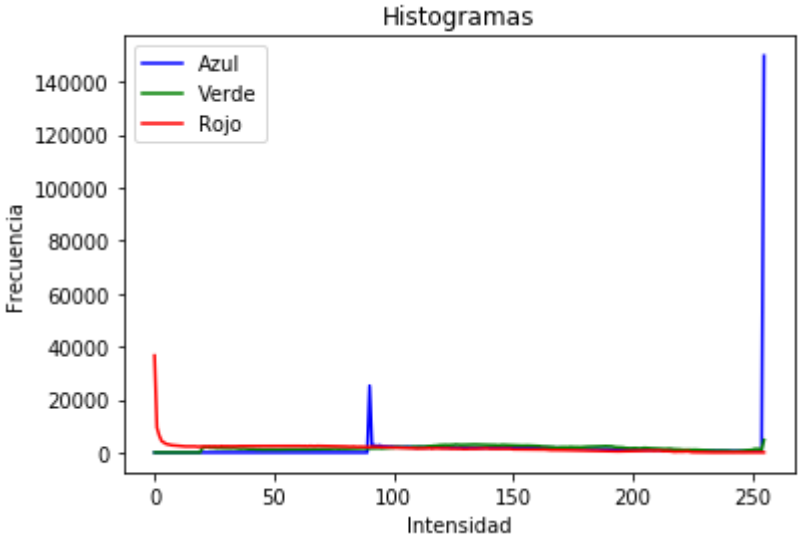
In [202]:

```
# IMG 01
img2_open_cv = contraste_openCV(img02, 2)
img2_propio = mejora_contraste(img02, azul = 90, verde = 20, rojo = 0)

# plot_rgb_histogramTodo(img02)

show_three_images(img02, "Imagen Original", img2_propio, "Metodo propio", img2_open_cv, "
restar_intensidad(img02, valor=50)
plot_rgb_histograms(img02)

plot_rgb_histogramTodo(img02)
plot_rgb_histogramTodo(img2_propio)
```





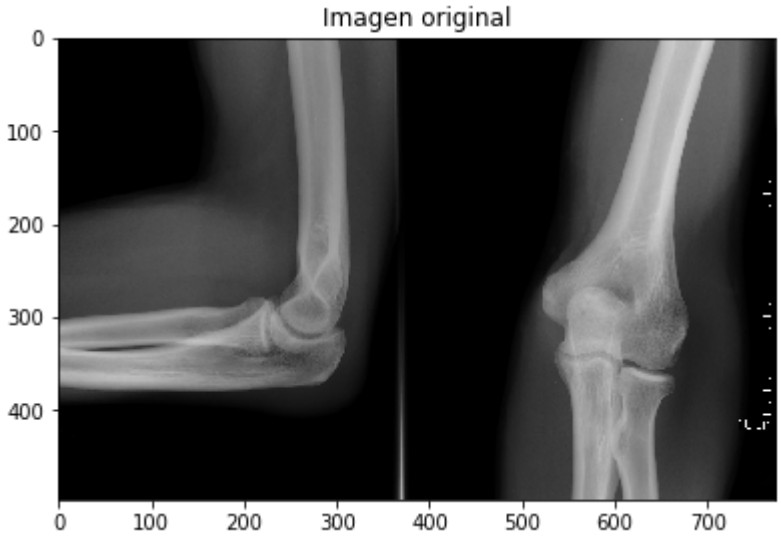

Histogramas
Azul
Verde
Rojo
Frecuencia
Intensidad
0
5000
10000
15000
20000
25000
30000
35000
0
50
100
150
200
250

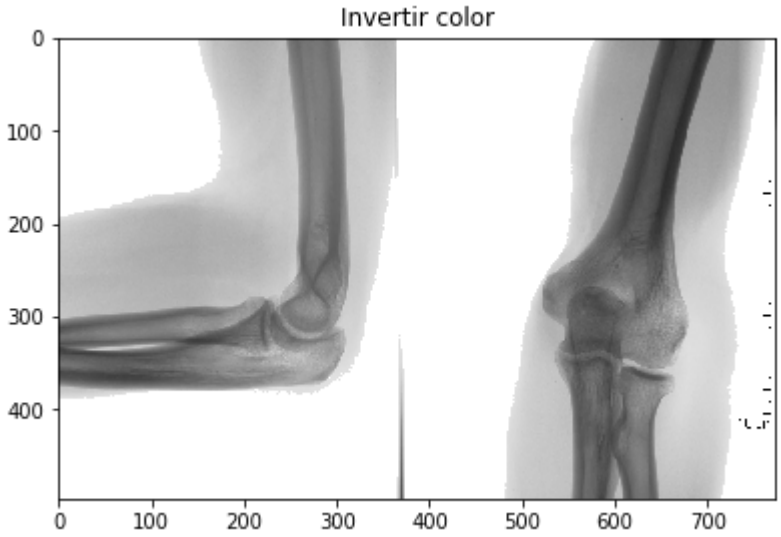

Histogramas
Azul
Verde
Rojo
Frecuencia
Intensidad
0
20000
40000
60000
80000
100000
120000
140000
0
50
100
150
200
250

Tambien me parecio interesante crear un método para invertir los colores de una imagen y tenes la flexibilidad de customizar en cada iteración.

In [190]:

```
imshow("Imagen original", img01)
imshow("Invertir color", invert_color(img01, "Invertir color"))
```





Mis conclusiones.

Respondiendo a las preguntas planteadas en el problema, concluyo que no hay una técnica única para cada caso, sino que, como profesionales del área, debemos indagar y probar en el momento aquella que mejor se ajuste al caso. En términos de eficiencia y rapidez, todas las que ya están hechas y empaquetadas en bibliotecas son mucho mejores. Sin embargo, al querer obtener un resultado más específico, se vuelve muy necesario desarrollar nuestras propias soluciones

Referencias

https://docs.opencv.org/4.8.0/ (https://docs.opencv.org/4.8.0/)

https://matplotlib.org/ (https://matplotlib.org/)

https://numpy.org/doc/stable/ (https://numpy.org/doc/stable/)

En lo particular intente no buscar en otras fuentes mas que el contenido visto en las clases, documentación oficial de open cv y el ingenio propio.